Sudan University of Science and Technology

College of Graduate Studies

Master of Information Technology

A Comparative Study of Swarm Intelligence (SI) Task Scheduling Algorithms in Cloud Computing

دراسة مقارنة بين خوارزميات السرب الذكي لجدولة المهام في الحوسبة السحابية

A Thesis Submitted in Partial Fulfillment of the requirements for the degree of M.Sc.in Information Technology

By
Isra Faisal Abdalla Elhag

Supervisor
Dr. Adil Yousif

October 2017

الآيـــــة

قال تعالى:" إِنَّ فِي السَّمَاوَاتِ وَالْأَرْضِ لَآيَاتٍ لِلْمُؤْمِنِينَ (3) وَفِي خَلْقِكُمْ وَمَا يَبُثُّ مِنْ دَابَّةٍ آيَاتٌ لِقَوْمٍ يُوقِنُونَ (4) "

(سورة الجاثية)

صدق الله العظيم

ACKNOWLEDGEMEN

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

Foremost, I want to offer this endeavor to our GOD Almighty for the wisdom he bestowed upon me, the strength, peace of my mind and good health in order to finish this research.

I would like to express my special gratitude and thanks to my adviser, Dr Adil Yousif for imparting his knowledge and expertise in this study.

I would like to express my gratitude towards my family for the encouragement which helped me in completion of this research. My supportive sister, who is always by my side when times I needed her most and helped me a lot in making this study, and my lovable brothers who served as my inspiration to pursue this undertaking

I am highly indebted to Sudan University of Science and Technology Faculty of Graduate Studies for their guidance and constant supervision as well as for providing necessary information regarding this research & also for their support in completing this endeavor

My thanks and appreciations also go to my colleague and people who have willingly helped me out with their abilities

Abstract

Cloud Computing refers to computing services over the internet and deals with varied different virtualization resources. The task scheduling plays a crucial role in enhancing the performance of cloud computing. The issue with task scheduling is distribution of tasks within the system in a manner that optimize the performance of overall system and minimize the execution time. To achieve such good plan the provider need to evaluate and choose among different algorithms to allocate and schedule the available resources. The challenging decision of choosing the proper algorithm is taking based on different performance metrics for task scheduling. In this research the focus is concentrated on task Execution Time as criteria for evaluating among the chosen algorithms. The selected mechanisms contain information of jobs (cloudlets) and resources (virtual machines) such as length of jobs, speed of resources and identifier for both. In order to generate the population, first, set of jobs and resources were created, then the execution times of jobs were computed as a fitness values. Second, the algorithms iterated to themselves in order to regenerate populations to produce the best job schedule that gives the minimum execution time of jobs. The methodology of this research is based on simulation of the selected mechanisms using the Java Language and CloudSim simulator. The comparison and analysis of different task scheduling algorithms has been discussed in this research on the basis of time execution. The results revealed that when having small sizes of scheduling problems PSO take the lead. However, in case of large size of jobs, Cat Swarm Optimization significantly outperforms the considered Particle Swarm Optimization, Firefly Algorithm and Glowworm Swarm Optimization in terms of execution time.

المستخلص

يقصد بالحوسبه السحابية: خدمات الحوسبة المقدمة عن طريق الإنترنت والتي تتعامل مع العديد من الموارد الافتراضية المختلفة. يعتمد تحسين اداء الحوسبة السحابية بصورة اساسية على طريقة جدولة المهام في السحابة. والهدف من جدولة المهام هو توزيع المهام داخل النظام بطريقة تؤدي الى تحسين الأداء العام في النظام وتقليل وقت التنفيذ. ولتنفيذ مهمة الجدولة بطريقة فعاله يجب على مقدم الخدمه التقييم والاختيار بين عدة خوارزميات تستخدم في جدولة تخصيص الموارد. وهناك مقاييس مختلفه لقياس اداء الخوارزمية. وتكمن صعوبة التقييم والإختيار في أنها تعتمد على مؤشرات مختلفه في اداء الخوارزمية.ولأغراض هذا البحث تم التركيز على وقت التنفيذ كمحدد للتقييم بين الخوارزميات المختاره. كل خوارزمية تحتوي على معلومات المهمة الداخلهللنظام (cloudlet) والموارد المتاحة (virtual machines) تختص هذه المعلومات بحجم المهمه وسرعة الموارد المخصصة لتلك المهمه ومحدداتهما. لتمثيل المجتمع (population) أولاً تقوم بإفتراض عدد من المهام والموارد, ومن ثم تخصيص المهام على حسب الموارد المتاحه بطريقة عشوائية, ويتم حساب وقت التنفيذ لكل مهمه. ثانياً تقوم الخوارزمية بتكرار العملية مرة اخرى للحصول على أفضل جدولة للمهام في أقل زمن تنفيذ. يتبع هذا بحث منهجية محاكاة الخوارزميات المقترحة باستخدام لغة الجافا وCloudSimulator. تمت مقارنة وتحليل خوارزميات جدولة المهام على اساس زمن التنفيذ وأظهرت النتائج أن خوارزميةPSO لها اقل زمن تنفيذ في حالة عدد المهام القليل, اماCSOكان لها أقل زمن تنفيذ في حالة عدد المهام الكثيرة مقارنة مع الخوارزميات الثلاثة الاخرى.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviation/Symbols	Explanation
CPU	Central Processing Unit
PSO	Particle Swarm Optimization
CSO	Cat Swarm Optimization
FSO	Firefly Swarm Optimization
GSO	Glowworm Swarm Optimization
QoS	Quality of Service
RA	Resource Allocation
FIFO	First In First Out
IO	Input Output
SLAs	Service Level Agreements
CloudSim	Cloud Simulator
SI	Swarm Intelligence
PBest	Particle Best
GBes	Global Best
V	Velocity
W	Weight
SMP	Seeking Memory Pool
CDC	Count of Dimension to Change
IT	Information Technology
FA	Firefly Algorithm
GA	Genetic Algorithm
ACO	Ant colony optimization
FCFS	First Com First Servers
VM	Virtual machine
MIPS	Million Instruction per second
β	Constant Parameter
α	Alpha
P	Luciferin decay Constant
Y	Luciferin Enhancement

Chapter One

Introduction

1.1 Introduction

This chapter introduce the research work, describe the problem background, problem statement, the research objective and the thesis structure.

1.2 Problem Background

In recent years, there has been a dramatic increase in the popularity of cloud computing systems that rent computing resources on-demand, bill on a pay-as-you-go basis, and multiplex many users on the same physical infrastructure. Cloud computing that has become an increasingly important trend, is a virtualization technology that uses the internet and central remote servers to offer the sharing of resources that include infrastructures, software, applications and business processes to the market environment to fulfill the elastic demand(Ngenzi and Nair, 2015).These cloud computing environments provide an illusion of infinite computing resources to cloud users so that they can increase or decrease their resource consumption rate according to the demands Two players in cloud computing environments, cloud providers and cloud users, pursue different goals; providers want to maximize revenue by achieving high resource utilization, while users want to minimize expenses while meeting their performance requirements. However, it is difficult to allocate resources in a mutually optimal way. Moreover, ever-increasing heterogeneity and variability of the environment poses even harder challenges for both parties(Quan et al., 2011).

Cloud computing data centres are emerging as new candidates for replacing traditional data centres that are growing rapidly in both number and capacity to meet the increasing demands for computing resources and storages(Quan et al., 2011). Large Cloud datacenters comprise of many thousands of servers and most of the time these servers are underutilized. The massive amount of wastage of resources in Cloud datacenters results in resource management problems. The challenges related

to datacenters with a particular emphasis on how new virtualization technologies can be used to simplify deployment, improve resource efficiency and reduce the number of usage of physical servers(Ngenzi and Nair, 2015).

The computing resources, either software or hardware, are virtualized and allocated as services from providers to users. Since the consumers may access applications and data of the "Cloud" from anywhere at any time, it is difficult for the cloud service providers to allocate the cloud resources dynamically and efficiently(Patil and Mehrotra, 2012).

Cloud is developing day by day and faces many challenges, one of them is scheduling. Scheduling refers to a set of policies to control the order of work to be performed by a computer system. A good scheduler adapts its scheduling strategy according to the changing environment and the type of task. There has been various types of scheduling algorithm existing in distributed computing system, and job scheduling is one of them. The main advantage of job scheduling algorithm is to achieve a high performance computing and the best system throughput. Scheduling manages availability of CPU memory and good scheduling policy gives maximum utilization of resource(Agarwal and Jain, 2014).

In cloud computing, the underlying large-scale computing infrastructure is often heterogeneous. To maximize cloud utilization, the capacity of application requirements shall be calculated so that minimal cloud computing infrastructure devices shall be procured and maintained. Given access to the cloud computing infrastructure, applications shall allocate proper resources to perform the computation with minimum time and infrastructure cost.

Scheduling is a difficult task in cloud computing environment because a cloud provider has to take care of many users according to their different QoS needs. Every task could have varied parameters like needed information, desired completion time, expected execution time, job priority etc..,(SundarRajan et al., 2016).

Management of these resources requires efficient planning and proper layout. While designing an algorithm for resource provisioning on cloud the developer must take into consideration different cloud scenarios and must be aware of the issues that are to be resolved by the selected algorithm(Katyal and Mishra, 2014).There are many promising methods to solve Job scheduling problems inspired from the nature. For sake of this research we focus on four algorithms, which are: Particle Swarm Optimization (PSO), Cat Swarm Optimization (CSO), Firefly Algorithm (FA) and Glowworm Swarm Optimization (GSO).

1.3 Problem statement

Many cloud datacenters have problems in understanding and implementing the techniques to manage, allocate and migrate the resources in their premises. The consequences of improper resource management may result into underutilized and wastage of resources which may also result into poor service delivery in these datacenters. Multiple resource types in datacenters make the situation even more complex, thus a careful planning for relocation is necessary. To achieve such good plan, the provider, need to evaluate and choose among different algorithms to allocate and schedule the available resources. The challenging decision of choosing the proper algorithm is taken based on different performance metrics for task scheduling. In this research the focus is concentrated on task Execution Time as criteria for evaluating among the chosen algorithms (Esa and Yousif, 2016a) (Katyal and Mishra, 2014).

1.4 Research Question

1- Are data center providers fully aware of the perspective of task scheduling algorithms?

2- What is the best swarm intelligence technique used in scheduling, to achieve the minimum task execution time?

1.5 Research Objectives

1. To provide a perspective on the domain of task scheduling in cloud data centers by summarizing different methods used.

2. Evaluating the swarm intelligence task scheduling algorithms in accordance to task execution time.

1.6 Research Scope

The resources allocation techniques in cloud data center and comparative study.

1.7 Research importance

Researches in the field of Resource allocation in Cloud Datahas become increasingly popular worldwide, and there is a need to shift from traditional working environment to achieve the ultimate electronic solution in solving our problem and processing our data, thus the efforts from the researchers and providers should be increase in this field. The contribution of this research is to provide a starting point for researchers and developers who want to evaluate and develop thetask scheduling techniques in Cloud data center, specially, in Sudan since few researches done to cover this field.

1.8 Thesis Structure

This thesis contains six chapters. Chapter two gives an overall idea of cloud computing and job scheduling in cloud computing. Chapter three describes the research methodology. Chapter four presents the swarm intelligence algorithms under the study for task scheduling. Chapter five Simulation Results and Performance Analysis. Chapter six provides the conclusion and recommendation.

Chapter Two

Literature Review

2.1 Introduction

In order to achieve our objectives, set before, we need to review recent researches that were conducted in the same field, here below are some of selected studies.

2.2 Cloud computing definition

Cloud application is very popular in recent years. Specially, cloud computing has emerged as a promising approach to rent IT infrastructure on a short-term pay-per-usage basis. With cloud computing, companies can scale up to massive capacities in an instant without having to invest in new infrastructure, train new personnel, or license new software. cloud computing is of a particular benefit to small-medium size business who wish to completely outsource their data center infrastructure, or large companies who wish to get peak load capacity without increasing the higher cost of building large data center internally(Tsai et al., 2013).

Cloud computing is defined as a model enabling ubiquitous, 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 (Mohamaddiah et al., 2014).Also, it can be defined as a computing paradigm, where a large pool of systems is connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly(Sarga, 2012).

Other defined it as a new technology that provides resources as an elastic pool of services in a pay-as-you-go model. Whether it is storage space, computational power, or software, customers can get it over the internet from one of

the cloud service providers. In both instances, services consumers use, what they need on the internet and pay only for what they use(Mohammad et al., 2012).

Cloud computing provides a shared pool of various resources including data storage space, networks, computer processing power, and user applications. The cloud services consist of highly optimized virtual datacenter and provide various hardware, software and information resources for use. The aim of cloud computing environment is to optimally use the available computing resources. Virtualization greatly helps in valuable utilization of resources and build an effective system. A cloud consist of several data centers, servers, clients which are interconnected in an efficient way(Kaur and Dhindsa, 2016).Resource allocation is a complicated task in cloud computing environment because there are many alternative computers with varying capacities.

2.3 Definitions of Resource allocation

Resource allocation is process of assigning the available resources in an economic, efficient and effective way. Resource allocation is the scheduling of the available resources and available activities required by those activities while taking into consideration both the resource availability and the project time. Resource provisioning and allocation solves that problem by allowing the service providers to manage the resources for each individual request of resource(Rajasekar and Manigandan, 2015).

In cloud computing, Resource Allocation (RA) is the process of assigning available resources to the needed cloud applications over the internet. Resource allocation starves services if the allocation is not managed precisely.(Anuradha and Sumathi, 2014).

In cloud computing, resource allocation (RA) is a field that is taken into account in many computing areas such as datacenter management, operating

systems, and grid computing. RA deals with the division of available resources between cloud users and applications in an economic and effective way(Alnajdi et al., 2012).

2.4 Cloud Scheduling

Scheduling is the vital task in cloud computing environment. Scheduling means the set of rules and mechanisms for controlling the order of work to be performed by computing systems. There are numerous types of scheduling algorithms and task scheduling being the significant one. In cloud task scheduling is the major problem. The scheduling of tasks means an optimal usage of available resources. The main purpose of scheduling is to achieve the high performance, reduce the waiting time, increase system throughput and so on. Task scheduling is a challenging issue in cloud computing because it is parallel and distributed architecture. The task completion time determination is difficult in cloud because the tasks may be distributed between more than one Virtual machine (Kaur and Dhindsa, 2016).

Traditional cloud scheduling algorithms typically aim to minimise and decrease the time and cost for processing all tasks scheduled. However, in cloud computing environment, computing capability varies from different resources and the cost of the resource usage. Therefore, it is important to take into consideration the cost. A scheduling algorithm is implemented by programmers to schedule the task with maximum estimated gain or profit and execute the task in the queue.

2.5 Need for Cloud Scheduling

In cloud computing, users may utilize hundreds of thousands of virtualized resources and it is impossible for everyone to allocate each task manually. Due to commercialization and Virtualization, cloud computing left the task scheduling complexity to virtual machine layer by utilizing resources virtually. Hence to assign the resources to each task efficiently and effectively, Scheduling plays an important role in cloud computing(Goyal and Sharma, 2016).

2.6 The different types of cloud scheduling are

• User level scheduling: User level scheduling comprises of market based and auction based scheduling. FIFO scheduling, priority based, non-pre-emptive scheduling etc. are used in user level scheduling.

• Cloud Service Scheduling: Cloud service scheduling is classified at user level and system level. At user level, it mainly considers the service regarding problems between the provider and the user. At system level, scheduling and resource management is done. In addition to real time satisfaction, fault tolerance, reliability, resource sharing and QoS parameters are also taken under consideration.

• Static and Dynamic Scheduling: Static scheduling permits pre-fetching of required data and pipelining of distant stages of task execution. Static scheduling imposes minimum runtime overhead. In case of dynamic scheduling, information of the job components or task is not known before. Thus the execution time of the task may not be known and the allocation of tasks is done only as the application executes.

• Heuristics Scheduling: In cloud environment, heuristic based scheduling can be done to optimize results. More accurate results can be built by heuristic methods.

• Workflow Scheduling: For the management of workflow execution, workflow scheduling is done.

• Real Time Scheduling: Real Time Scheduling in cloud environment is done to increase the throughput and to decrease the average response time instead of meeting deadline(Goyal and Sharma, 2016).

2.7 Load Balancing in Cloud Computing Environment

Load balancing in cloud computing provides an efficient solution to various issues residing in cloud computing environment set-up and usage. Load balancing must take into account two major tasks, one is the resource provisioning or resource allocation and other is task scheduling in distributed environment(Katyal and Mishra, 2014).

2.8 Task scheduling

There are various types of task scheduling algorithm. The main goal of a scheduling algorithm is to achieve high computing performance and best system throughput. Traditional scheduling algorithms cannot operate in cloud environment (because of overhead costs), thus providers have resorted to heuristic or hybrid algorithms to fill this gap. Effectiveness of task scheduling has a direct effect on the quality of cloud, thus many algorithms have been developed to resolve this particular problem. In some studies, algorithms have been developed to optimize the resource efficiency(Chalack and Germi).

In order to efficiently and cost effectively schedule the tasks and data of applications onto these cloud computing environments, application schedulers have different policies that vary according to the objective function: minimize total execution time, minimize total cost to execute, balance the load on resources used while meeting the deadline constraints of the application, and so forth(Pandey et al., 2010).

2.9 Performance metrics for task scheduling

A good scheduling algorithm always considers benefits of both the parties the cloud users and the service providers. The algorithms should try to reduce both the cost and power consumption as well as provide better performance. Scheduling algorithms must consider Load balancing, energy consumption, user's fairness and security while providing services. Below is an overview of common Performance metrics for task scheduling recommended by different researches in the field:

1. Execution Time

The CPU time or burst time spent by the computer system for execution of a task is known as execution time, including the time consumed to provide system services for task execution(Ali and Alam, 2016). In other way, the exact time taken to execute the given tasks. A good scheduling algorithm ultimately aims to minimize execution time(Yadav and Mandoria, 2017).

2. Response Time

The amount of time taken by the system to reply the user task very first time for required service. That service may also be something from a memory fetch, to a disk IO, to an elaborate database question, or loading a full web page. Response time of the system should be minimum(Ali and Alam, 2016).

3. Makespan

[Syed and Mansaf], the amount of time, from start to finish for completing a set of tasks. The makespan is the maximum time to complete all jobs(Ali and Alam, 2016). Whereas, [Ashwani and Hardwari] refers it as the aggregate consummation

time of all tasks in the job queue. A good scheduling algorithm dependably tries to diminish the makespan(Yadav and Mandoria, 2017).

4. Throughput

Throughput uses the consideration of total number of tasks, which are implemented successfully. In cloud computing, throughput means some tasks completed in a certain time period. Minimum throughput is required for task scheduling (Madni et al., 2017).

5. Resource Utilization

In addition to response time and throughput, another parameter for performance measurement of a system is resource consumption. How much amount of system resources are busy? is track using resource utilization. Scheduling algorithm should increase the utilization of resources(Ali and Alam, 2016).

6. Load Balancing

[Ashwani and Hardwari]It is the strategy for dissemination of the whole load in a cloud network crosswise over various nodes furthermore, connects so that at once no nodes and connections remain under loaded while a few nodes or connections are over-loaded. Most of the scheduling algorithms try to keep the load balanced in a cloud network in order to increase the efficiency of the system(Yadav and Mandoria, 2017).

7. Fault Tolerance

Fault tolerance is the property that allows for a procedure to continue running effectively within the incident of the failure in its components. It is an important parameter to check the capability of any system(Ali and Alam, 2016).

8. Energy Consumption

Energy consumption is the amount of resource energy used to produce the output. Energy consumption should be minimal(Ali and Alam, 2016).

Energy utilization in cloud data centers is a present issue that ought to be considered with more care nowadays. Numerous scheduling algorithms were developed for diminishing power consumption and enhancing execution and consequently making the cloud services green(Yadav and Mandoria, 2017).

9. Scalability

It is a characteristic of a system, model or function that describes its ability to manage and participate in below multiplied or increasing workload. A process that scales well might be competent to hold or even broaden its level of efficiency when tested by using higher operational needs(Ali and Alam, 2016).

10. Performance

The accomplishment of a given task measured against pre-set identified requirements of completeness, cost, accuracy, and velocity. In computing performance is measured by the time and cost, a system should complete a user task

in less time and minimum cost of services. A performance should be considered at both sides user and provider by scheduling algorithm(Ali and Alam, 2016).

11. Quality of Service

Best of provider considers user involvement restrictions like meeting cut-off date, efficiency, execution price, make span, and so forth. Everything are outlined in Service Level Agreement (SLAs) which is a contract file outlined between the cloud user and cloud service provider. Input constraints such as meeting execution cost, deadline, performance, cost, makespan, etc enhances quality of service(Ali and Alam, 2016).

12. Cost

Cost means the total payment generate against the utilization or usage of resources, which is paid to the cloud providers by the cloud users. The main determination is to the growth of revenue and profit for cloud providers while reducing the expenses for cloud user with efficient utilization (Madni et al., 2017).

Job scheduling, one of the most famous optimization problems. Job scheduling has been considered as one of crucial problems in cloud computing. An optimized scheduler would improve many factors in scheduling of jobs in a cloud system such as throughput and performance. Different Approaches have tried to solve this problem like Genetic algorithm, Ant colony optimization, Particle swarm optimization and etc. which are considered types of swarm Intelligence techniques.

2.10 Swarm Intelligence (SI)

Swarm intelligence models are referred to as computational models inspired by natural swarm systems. To date, several swarm intelligence models based on

different natural swarm systems have been proposed in the literature, and successfully applied in many real-life applications. Examples of swarm intelligence models are: Ant Colony Optimization , Particle Swarm Optimization, Artificial Bee Colony, Bacterial Foraging, Cat Swarm Optimization, Artificial Immune System, and Glowworm Swarm Optimization (Ahmed and Glasgow, 2012).In this research, we will primarily focus on four of the most popular swarm intelligences models, namely, Particle Swarm Optimization, Cat Swarm Optimization, Glowworm Swarm Optimization and Firefly Swarm Optimization.

2.10.1 Particle Swarm Optimization(PSO)

PSO is a population-based optimization technique that finds solution to a problem in a search space by modeling and predicting insect social behavior in the presence of objectives. The general term “particle” is used to represent birds, bees or any other individuals who exhibit social behavior as group and interact with each other. Under PSO, multiple candidate solutions –called particles– coexist and indirectly collaboratesimultaneously.Eachparticle“flies”intheproblemsearchspacelookingfor the optimal position to land. A particle adjusts its position as time passes according to its own experience as well as according to the experience of neighbor particles. Moreover, particles are essentially described by two characteristics: the particle position, which defines where the particle is located with respect to other solutions in the search space, and the particle velocity, which defines the direction and how fast the particle should movetoimproveitsfitness.Thefitnesssofaparticleisanumberrepresentinghowclose a particle is to the optimum point compared to other particles in the search space(Pacini et al., 2014).

2.10.2 Firefly Algorithm (FA)

This firefly algorithm has been designed based on the inspiration on the swarm behavior of fireflies. Fireflies are generally known to exist as groups and they

are said to have a swarm kind of behavior. The blinking light in the fireflies is their attribute of attractiveness mainly used for the purpose of attracting mates and to defend themselves from other predators. The swarm of fireflies usually moves in the direction of the brightest one. All the other fireflieswith lower light intensities move toward the ones with higher light intensities. So as the distance between the fireflies goes on increasing, the light intensity also increases(SundarRajan et al., 2016).

2.10.3 Cat Swarm Optimization (CSO)

In this CSO heuristic optimization algorithm, created based on the inspiration towards the swarm behavior of cats. Cats that generally have swarm behavior are said to have two modes of behavior namely 1. seeking mode and 2. Tracking mode

Seeking mode: In seeking mode the cat stays idle and only has position whereas they do not have velocity.

Tracking Mode: In tracing mode the cat is in motion and is said to possess both position and velocity.

This algorithm wholly lies on two modes of operation. The fitness factor for each cat is calculated and the best one is picked out. The best cat is stored in memory and it is updated with the next best cat. Here in the Cat swarm algorithm the virtual machines are disguised as cats(SundarRajan et al., 2016).

2.10.4 Glowworm swarm optimization (GSO)

Glowworm swarm optimization (GSO), introduced by Krishnanand and Ghose in 2005 for simultaneous computation of multiple optima of multimodal functions. GSO is a new optimization algorithm, inspired by nature, which imitates the behavior of the lighting worms. The agents in GSO are thought of as glowworms

that carry a luminescence quantity called luciferinLi(t) along with them. The glowworms encode the fitness of their current locations, evaluated using the objective function, into a luciferin value that they broadcast to their neighbors. The glowworm identifies its neighbors and computes its movements by exploiting an adaptive neighborhood, which is bounded above by its sensor rangerdi(t). Each glowworm selects, using a probabilistic mechanism, a neighbor that has a luciferin value higher than its own and moves toward it. These movements—based only on local information and selective neighbor interactions—enable the swarm of glowworms to partition into disjoint subgroups that converge on multiple optima of a given multimodal function. Each iteration consists of a luciferin-update phase followed by a movement-phase based on a transition rule and Local-decision range update phase (Esa and Yousif, 2016a).

2.11 Related works

A number of task scheduling algorithms have been proposed by many researchers. In this research we focus on four algorithms which are Particle Swarm Optimization, Cat Swarm Optimization, Firefly algorithms and Glowworm Swarm Optimization. However, after reviewing these studies it's been found that proposed algorithms were discussed separately. In this section previous studies were listed to show their relevance to the proposed mechanism.

In this paper, a simplified version of particle swarm optimization (PSO) algorithm is proposed to solve the job scheduling problem in cloud computing environment. To evaluate the performance of the proposed approach, this study compares the proposed PSO strategy with genetic algorithm (GA), by having both of them implemented on CloudSim toolkit. The results obtained demonstrate that the presented PSO algorithm can significantly reduce the makespan of job scheduling problem compared with the other metaheuristic algorithm evaluated in this paper(Attiya and Zhang, 2017).

The study on comparison of ACO and PSO has been presented in this paper by analysing the optimization methods of each algorithm. Both optimization techniques are assigned with a specific task to allocate resources within minimum execution time by analysing the makespan to measure the throughput. PSO is considered as best optimization with low computational cost(BOOBA and GOPAL).

In cloud computing environment, there are a large number of users, which lead to huge amount of tasks to be processed by system. In order to make the system complete the service requests efficiently, how to schedule the tasks becomes the focus of cloud computing Research. A task scheduling algorithm based on PSO and ACO for cloud computing is presented in this paper. First, the algorithm uses particle swarm optimization algorithm to get the initial solution quickly, and then according to this scheduling result the initial pheromone distribution of ant colony algorithm is generated. Finally, the ant colony algorithm is used to get the optimal solution of task scheduling. The experiment simulated on CloudSim platform shows that the algorithm has good effect in real-time performance and optimization capability. It is an effective task scheduling algorithm(Wang and Chen).

This paper proposes a new job scheduling mechanism using Firefly Algorithm to minimize the execution time of jobs. The proposed mechanism based on information of jobs and resources such as length of job speed of resource and identifiers. Different settings have been considered in the evaluation and experimentation phase to examine the proposed mechanism in different workloads. The results revealed that the proposed mechanism minimizes the execution time significantly. Furthermore, the proposed mechanism outperformed the FCFS algorithm(Esa and Yousif, 2016b).

The proposed mechanism aims to find the best mapping in order to minimize the execution time of jobs. The methodology of this research is based on simulation of the proposed mechanism using the CloudSim simulator. The evaluation process of the proposed mechanism started with a set of different experiments. These experiments revealed that, the proposed mechanism minimized the execution time of

jobs. The proposed mechanism is compared with the First Come First Servers (FCFS) algorithm and experimental results revealed that the proposed mechanism has a better performance than FCFS for minimizing the execution time of jobs(Esa and Yousif, 2016a).

In a paper written by Bilgaiyan and others the authors presented a scheduling technique based on a relatively new swarm-based approach known as Cat Swarm Optimization. This technique shows considerable improvement over PSO in terms of speed of convergence(Bilgaiyan et al., 2014).

Chapter Three
Research Methodology

3.1 Introduction

This chapter describes all phases of research methods that have been applied to develop the proposed mechanisms and tools used in the research work.

3.2 Operational Framework

This research aims to develop a comparative study for PSO, CSO, FA and GSO in term of Execution Time. Based on simulation of the proposed mechanisms using the Java Language and CloudSim simulator. CloudSim toolkit is a tool for modeling and simulation of cloud computing environment. It supports dynamic creation information of jobs (cloudlets) and resources (virtual machines) such as length of jobs, speed of resources and identifier for both. In order to generate the population, first, set of jobs and resources were created, and jobs were assigned to resources randomly, then the execution times of jobs was computeas a fitness values. Second, iterations were used by algorithms to regenerate populations to produce the best job schedule that gives the minimum execution time of jobs.

The operational framework of the study is described in Figure (3.1) and the following subsection illustrates this framework.

3.2.1 Problem Formulation

This research aims to evaluate PSO, CSO, FA and GSO in term of Execution Time.

3.2.2 Proposal Writing

In this step research objectives and the overall research plan were set. The research methodology that will be employed in the research work is described in details and initial results are presented.

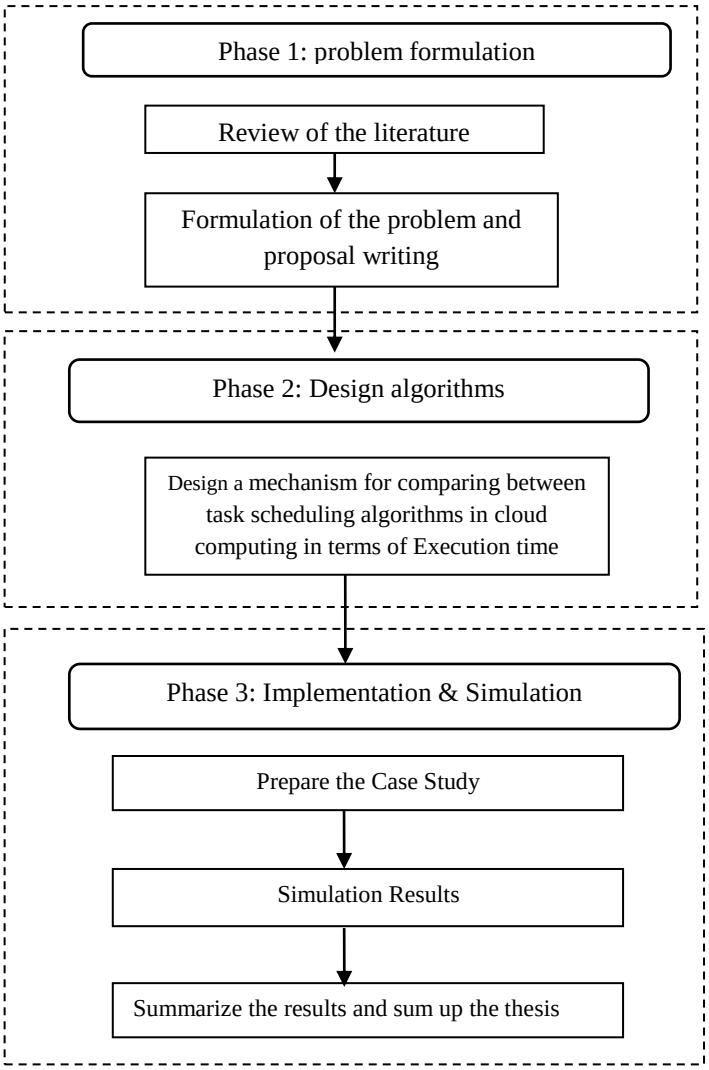

Figure (3.1): Research Operational Framework

3.2.3 Design of Proposed Framework

In the design phase we focus on how to enhance the performance of cloud computing in term of execution time by studying all selected algorithms and compare them to find the best algorithms that minimize the execution time significantly.

3.3 Implementation

In this phase the design of the proposed mechanism and its application tool was done using CloudSim simulator implemented in Eclipse by using Java language. This phase started by preparing the case study that will be used in the testing process. CloudSim toolkit is a tool for modeling and simulation of cloud computing environment.

3.4 Tool Used in This Methodology

Use CloudSim simulator implemented in Eclipse by using Java language.

Chapter Four

Swarm Intelligence Technique

4.1 Introduction

In this chapter, four of swarm intelligence techniques (PSO, CSO, FA and GSO) were explained in depth, followed by Pseudo code and flowchart for each.

4.2 Swarm Intelligence (SI)

Swarm Intelligence (SI) has received increasing attention lately among researchers, and refers to the collective behavior that emerges from social insects' swarms to solve complex problems. Hence, researchers have proposed algorithms for combinatorial optimization problems. Moreover, scheduling in Clouds is also a combinatorial optimization problem, and hence schedulers exploiting SI have been proposed(Pacini et al., 2014).

4.2.1 Particle Swarm Optimization (PSO)

The particle swarm optimization algorithm was first proposed in 1995 by James Kennedy and Russell C. Eberhart. PSO is a method for optimizing hard numerical functions on metaphor of social behavior of flocks of birds and schools of fish. The original PSO algorithm is discovered through simplified social model simulation. It was first designed to simulate birds seeking food which is defined as a cornfield vector. The bird would find food through social cooperation with other birds around it and expanded to multidimensional search(Surekha and Sumathi, 2011).

PSO has particles which represent candidate solutions of the problem, each particle searches for optimal solution in the search space, each particle or candidate solution has a position and velocity. A particle updates its velocity and position

based on its inertia, own experience and gained knowledge from other particles in the swarm, aiming to find the optimal solution of the problem.

In every iteration, each particle is updated by following two “best” values. The first one is best solution it has achieved; its value is called pbest. Another “best” value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When particle takes part of the population as its neighbors, the best value is the local best and called lbest. In the local population, each particle keeps track of the best position lbest attained by its local neighboring particles. For the global population, the best position gbest is determined by any particles in the entire swarm. Thus the gbest model is a special case of the lbest model. Peng- Yeng Yin.

After finding the two best values, the particle updates its velocity and positions with following equation (1) and (2).

$$v[] = \omega * v[] + c_1 \, rand() * (pbest[] - present[]) + c_2 rand() * (gbest[] - present[]) \quad (1)$$

$$present[] = present[] + v[] \quad (2)$$

Where,

- v []: The velocity for the i th particle, represents the distance to be traveled by this particle from the current position.
- ω inertia weights usually 0.8 to 0.9.
- rand () is a random number between $(0,1)$
- c1, c2 are learning factors. Usually $c1 = c2 = 2$.
- Present []: The location of the ith particle i.e., particle position.
- Pbest []: The best previous position of the ith particle is recorded and represented as pbest [].

Gbest []: The index of the best particle among all the particles in the population is represented by gbest [].

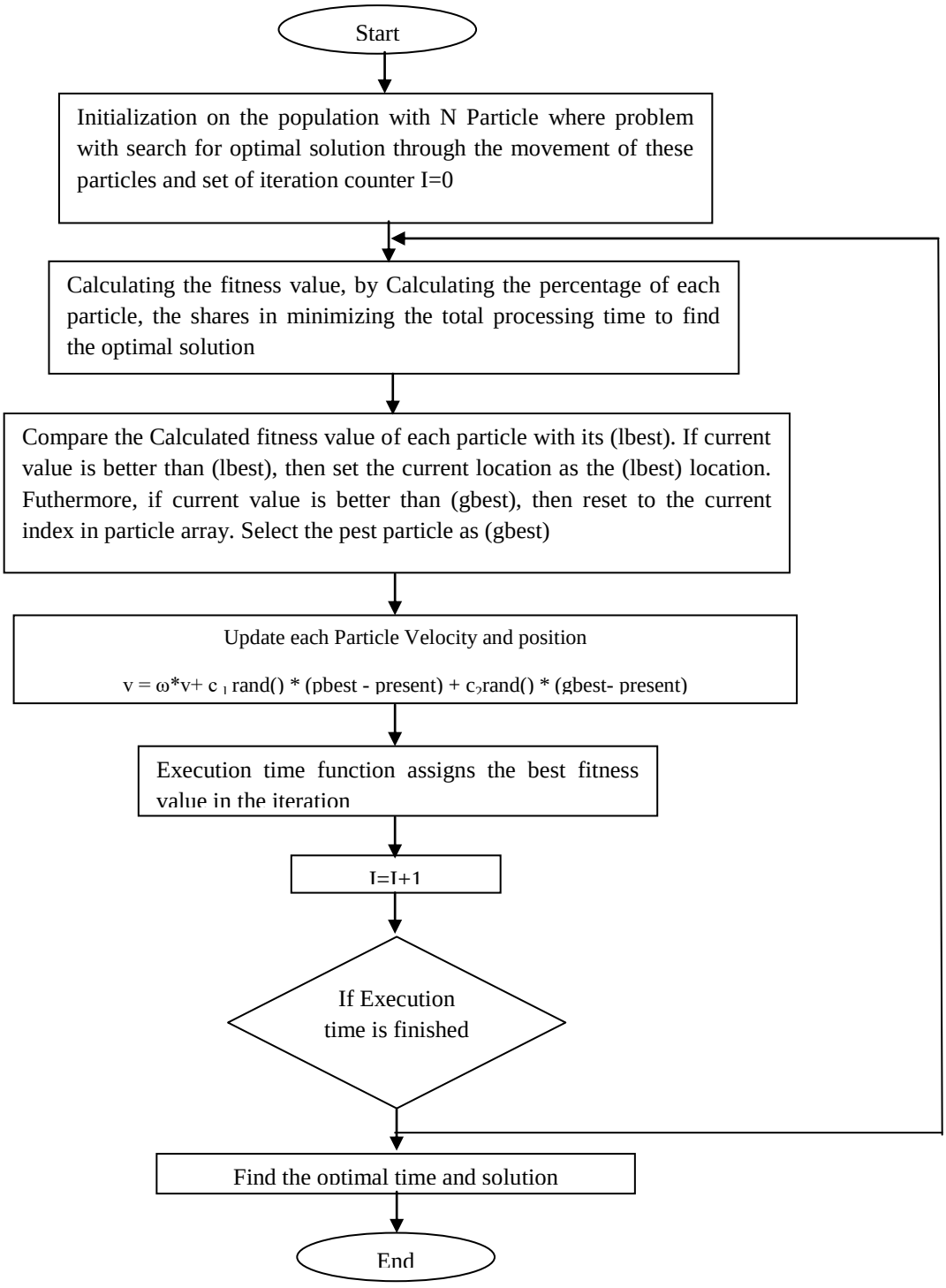

Figure (4.1): Flowchart of Particle Swarm Optimization

4.2.1.1 Pseudo Code for PSO algorithm

Input the scheduling problem
Setup the parameters
Generate a swarm of particles with random positions and velocities
Calculate the fitness value of each particle in the swarm
Select the particle with best fitness value from all particles as global best
while termination criterion is not met do
for each particle i do
Update the particle's velocity
Check the velocity boundaries for each component of velocity−vector
Update the particle's position
Round off the real values in particle's position into the nearest integer
Evaluate the fitness of the particle
if $F(X_i) < F(P_i)$ then
Update the global best
end if
end for
end while
Output the best particle (schedule) as the final solution

4.2.2 Cat Swarm Optimization (CSO)

A new swarm-based evolutionary algorithm named Cat Swarm Optimization (CSO) has been introduced by Chuand Tsai in 2007. It is inspired and Tsai in 2007. It is inspired Cats exhibit two modes of behavior - 1) Seeking mode, in which cats do not move. They just stay in a certain position and sense for the next best move, thus having only state and notvelocity. 2) Tracing mode, in which cats move to their next best positions with some velocity, representing how the cats chase their target.

The proposed algorithm uses an initial population of N cats among which some are in seeking mode while others are in tracing mode, according to MR. Each cat represents a task-resource mapping, which is updated as per the mode that the cat is in. Assessing the fitness value of the cats leads to finding the mapping having

minimum cost. In each iteration, a new set of cats is chosen to be in tracing mode. The final solution, represented by the best position among the cats, gives the best mapping that has the minimum cost among all mappings.

Seeking mode: This represents the majority of cats that search the global space while being in a resting state by intelligence position updating. Here the algorithm uses two basic factors - SMP and CDC. SMP (seeking memory pool) represents the number of copies to be made for each cat. CDC (count of dimension to change) defines how many of the allocations are to be altered in a single copy. The general steps are as follows:

Step 1. Create j copies of the i^{th} cat as represented by SMP.
Step 2. Modify CDC dimension of each copy randomly
Step 3. Evaluate fitness of each copy
Step 4. Find the best solutions among all copies that is the mapping have minimum Execution time
Step 5. Randomly choose a solution among them and replace it for the i^{th} cat

Tracing mode: This represents the cats that are in a fast moving mode and search the local space by moving towards the next best position with high energy. The general steps are as follows:

Step 1. Find the velocity v_i^{t+1} for the i^{th} cat as per

$$v_i^{t+1} = w.v_i^t + r1.c1.(x_{best} - x_i^t)$$

There w is the inertia weight, r1 is a random number such that $0 \leq r1 \leq 1$ and c1 is the acceleration constant. v_i^t is the previous velocity,x_{best} is the best location and x_i^t is the current location.
Step 2. Update position for the cat as per

$$x_i^{t+1} = x_i^t + x_i^{t+1}$$

Where x_i^t is the current position.
Step 3. Check if the position goes out of the defined range. If so, assign the boundary value of the position.
Step 4. Assess the fitness value for the cats.
Step 5. Update the solution set with the best position of the current iteration.

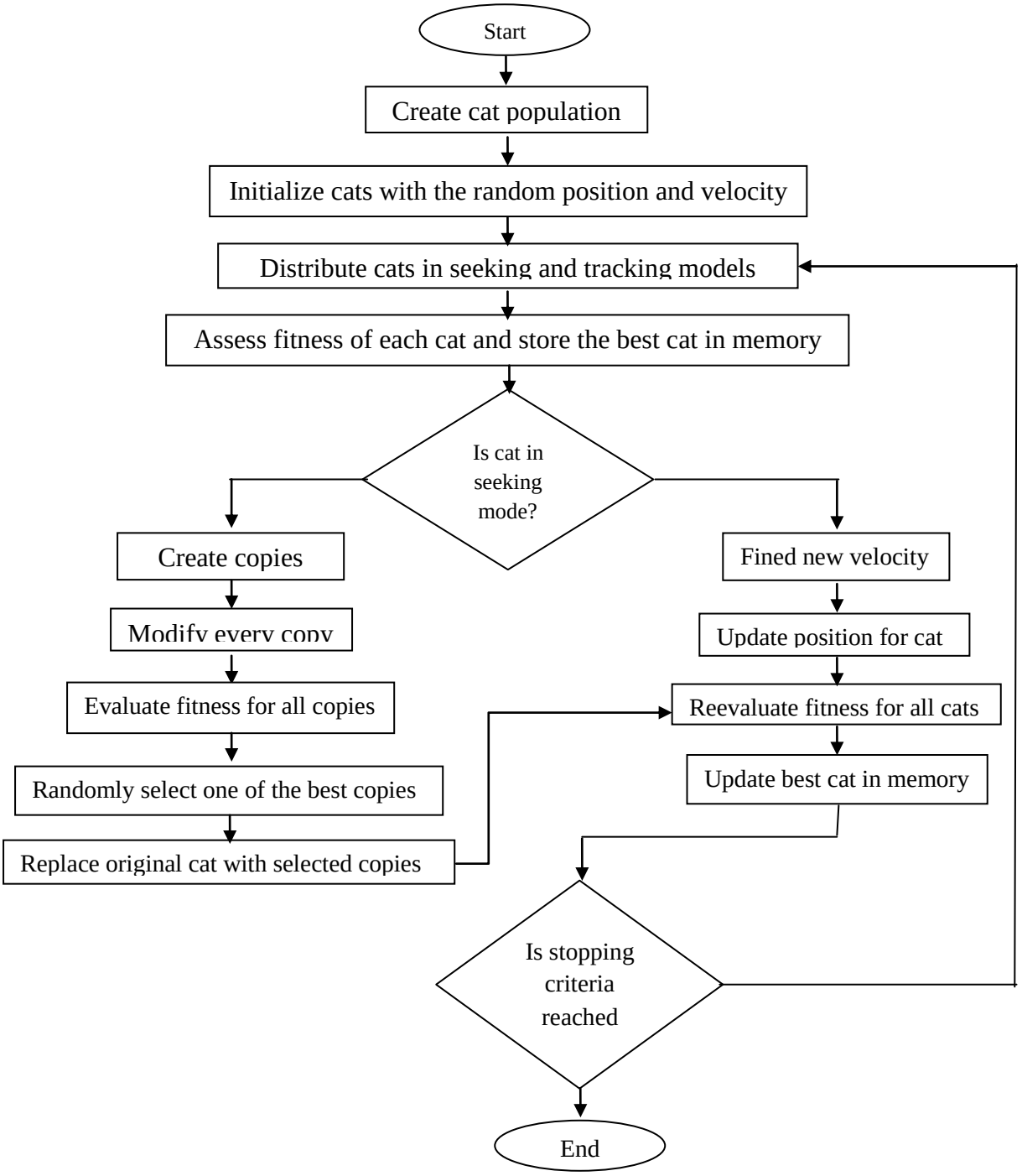

Figure (4.2) Flowchart of CSO algorithm(Bilgaiyan et al., 2014)

4.2.3 Glowworm Swarm Optimization algorithm (GSO)

Glowworm swarm optimization (GSO), introduced by Krishnanand and Ghose in 2005 for simultaneous computation of multiple optima of multimodal functions. GSO is a new optimization algorithm, inspired by nature, which imitates the behavior of the lighting worms. SI systems consist typically of a population of simple agents or interacting locally with one another and with their environment. The agents in GSO are thought of as glowworms that carry a luminescence quantity called luciferinLi(t) along with them. The glowworms encode the fitness of their current locations, evaluated using the objective function, into a luciferin value that they broadcast to their neighbors. The glowworm identifies its neighbors and computes its movements by exploiting an adaptive neighborhood, which is bounded above by its sensor rangerdi(t). Each glowworm selects, using a probabilistic mechanism, a neighbor that has a luciferin value higher than its own and moves toward it. These movements—based only on local information and selective neighbor interactions—enable the swarm of glowworms to partition into disjoint subgroups that converge on multiple optima of a given multimodal function. Each iteration consists of a luciferin-update phase followed by a movement-phase based on a transition rule and Local-decision range update phase(Esa and Yousif, 2016a).

4.2.3.1 Luciferin-update-phase

At time t, the location of the glowworm i is xi(t), and its corresponding value of the objective function at glowworm i's location at time t is J(xi(t)). The luciferin level associated with glowworm i at time t is given by equation (1)

$$\mathrm{Li(t)} = (1-\rho)\mathrm{Li(t-1)} + \gamma \mathrm{J(xi(t))} \qquad (1)$$

4.2.3.2 Movement-phase

Find the neighbors j for each glowworm i: Ni(t) using equation(2)

$$\mathrm{j} \in N_{ij} \text{ iff } d_{ij} < rdi(\mathrm{t}) and\ Lj(t) > Li(t) \qquad (2)$$

Each Glowworm i moves towards a neighbor j with a certain probability computed by equation (3)

$$\mathrm{Pij(t)} = \frac{Lj(t) - Li(t)}{\sum k \in Ni(t) Lk(t) - Li(t)}$$

The glowworm i position is updated using equation (4)

$$\mathrm{Xi(t + 1)} = \mathrm{Xi(t)} + \mathrm{S}\left(\frac{Xj(t) - Xi(t)}{|Xj(t) - Xi(t)|}\right)$$

where S is the step size.

4.2.3.3 Local-decision Range Update Rule

The neighborhood range is updated using equation (5)

$$r_d^i(\mathrm{t} + 1) = \min\{\mathrm{rs}, \max\{0, \mathrm{rdi(t)} + \beta(\mathrm{nt} - |\mathrm{Ni(t)}|)\}\} \qquad (5)$$

where β is a constant parameter, rs is the constant radial sensor range, nt is a parameter used to control the number of neighbors and |Ni(t)| is the actual number of neighbors.

At the beginning, all the glowworms contain an equal quantity of luciferin l0 and the same neighborhood decision range r0. Each iteration consists of a luciferin update phase followed by a movement phase based on a transition rule. Other

involved parameters are the luciferin decay constant (ρ), the luciferin enhancement constant (γ), the step size (s), the number of neighbors (nt), the sensor range **(**rs) and a constant value (β).

Parameters values of Glowworm Algorithm that are Kept Constant for all experiments are described in Table 1(Esa and Yousif, 2016a).

Table (4.1) Glowworm Optimization Parameters

P	**Γ**	**β**	**nt**	**S**	**L_0**
0.4	0.6	0.08	5	0.03	5

4.2.3.4 Pseudo Code for the GSO Algorithm(Kaipa and Ghose, 2017)

Glowworm Swarm Optimization (GSO) Algorithm

Set number of dimensions m
Set number of glowworm n
let s be the step size
Let xi(t) be the location of glowworm i at time t
Deploy-agents-randomly
For i=1 to n do $li(o) = lo$
$r_d^i(0) = r_0$
Set maximum iteration number = $iter\_max$;
Set t = 1 ;
While ($t \leq iter\_max$) do:
{
For each glowworm i do: % luciferin-update phase

$$\mathrm{Li(t)} = (1-\rho)\mathrm{Li(t-1)} + \gamma \mathrm{J(xi(t))}$$

For each glowworm i do: % Movement phase
{

$$n_i(t) = \{j : d_{ij}(t) < r_d^i(t); l_i(t) < l_i(t)\};$$

For each glowworm $j \in N_i(t)$do :

$$\mathrm{Pij(t)} = \frac{Lj(t) - Li(t)}{\sum k \in Ni(t) Lk(t) - Li(t)}$$

j = select-glowworm (p);

$$\mathrm{Xi(t+1)} = \mathrm{Xi(t)} + \mathrm{S}\left(\frac{Xj(t) - Xi(t)}{|Xj(t) - Xi(t)|}\right)$$

$$r_d^i(\mathrm{t+1}) = \min\{\mathrm{rs}, \max\{0, \mathrm{rdi(t)} + \beta(\mathrm{nt} - |\mathrm{Ni(t)}|)\}\}$$

}
t= t+1
}

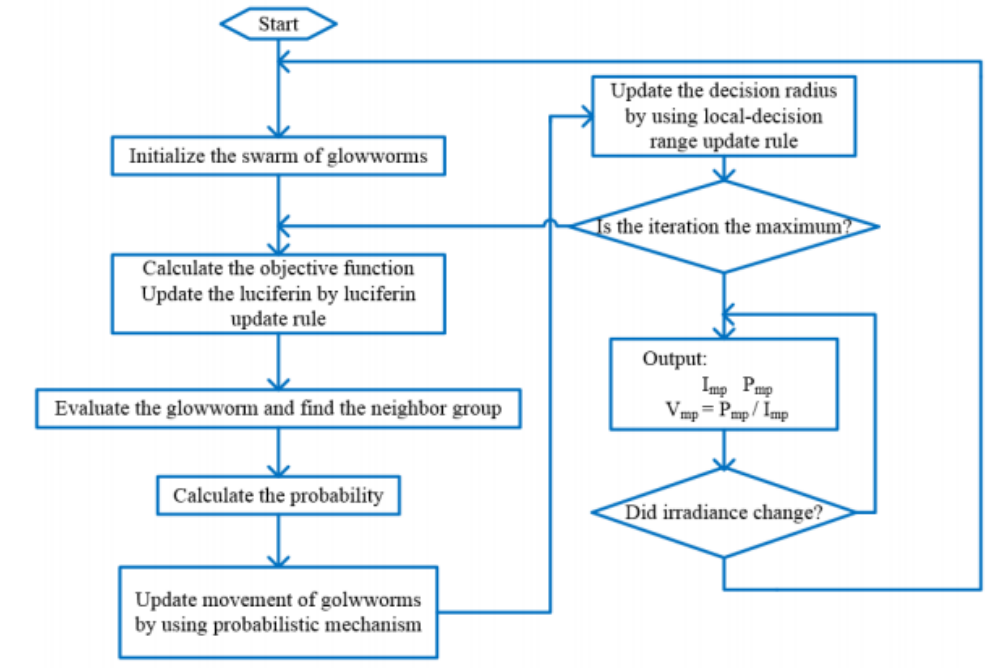

Figure (4.3): Flow chart of Glowworm Optimization(Jin et al., 2017)

4.2.4 Firefly Algorithm (FA)

The Firefly algorithm was introduced by Dr. Xin She yang at Cambridge University in 2007 which was inspired by the mating or flashing behavior of fireflies. Although the algorithm has many similarities with other swarm based algorithms such as Particle Swarm Optimization, Artificial Bee Colony Optimization and Ant Colony Optimization, the Firefly algorithm has proved to be much simpler both in concept and implementation(Hashmi et al., 2013).

The firefly algorithm is based on three main principles:

1. All fireflies are unisex, implying that all the elements of a population can attract each other.

2. The attractiveness between fireflies is proportional to their brightness. The firefly with less bright will move towards the brighter one. If no one is brighter than a particular firefly, it moves randomly. Attractiveness is proportional to the brightness which decreases with increasing distance between fireflies.

3. The brightness or light intensity of a firefly is related with the type of function to be optimized. In practice, the brightness of each firefly can be directly proportional to the value of the objective function(Francisco et al., 2014).

4.2.3.1 Firefly Algorithm for Cloud Job Scheduling

In the proposed mechanism the study used Firefly Algorithm in solving the problem of job scheduling and allocation of jobs to resources. Each firefly is a solution for allocation of jobs $\mathrm{Xij}, \mathrm{i} = (1,2,3, \dots, n)\ j = (1,2,3, \dots, k)$, each element inside the firefly population vector is a random number between 1 to s where:
s is the total number of resources.
n is number of fireflies.
k is number of jobs that represent the length of each firefly.

The study represents resources as a vector that stores the speed of each resource$\mathrm{Ri}, \mathrm{i} = (1,2,3, \dots, s)$, and also jobs as a vector that stores the length of each job $\mathrm{Ji}, \mathrm{i} = (1,2,3, \dots, k)$, then we calculated the fitness function $F(Xij)$for each firefly by dividing each job length by the resource speed that the job is allocated to. The next step is to find the summation of the division results. This followed by finding the maximum fitness value. The firefly that has maximum fitness either moves randomly or does not move at all. The distance between each two fireflies is the

number of non-corresponding elements[14] in the firefly population is calculated and stored in $Dij, i = (1,2,3,\dots,n)\ j = (1,2,3,\dots,k)$ vector, and then calculate the attractiveness $\beta ij, i = (1,2,3,\dots,n)\ j = (1,2,3,\dots,k)$, for each Firefly from the fitness of the firefly by the equation $\beta ij = F(Xij) * e^{-\gamma Dij^2}$, where γ is fixed light absorption coefficient and e is exponential constant.

Finally, firefly moves towards the brightest based on the attractiveness by the equation

$$Xij = Xij + \beta ij\, e^{-\gamma Dij^2}(Xi - Xj) + (rand - \frac{1}{2})$$

Where α is randomization parameter between 0 and 1.

Pseudo Code for FA

Begin
Initialize parameter: t, itra_max, α, γ.
Generate initial population of fireflies $Xij, i = (1,2,3,\dots,n)\ j = (1,2,3,\dots,k)$,
Set maximum iteration number=iter_max.
Set t=1
For each resource do
Set speed for each resource R_i
end for
for each job do
Set length for each jobs J_i
end for
while (t<=iter_max)
for each firefly i do
Compute Fitness function $F(Xij)$
end for
foreach firefly i do
for each firefly jdo
Compute the distance between firefly i and firefly j

$$\beta ij = F(Xij) * e^{-\gamma Dij^2}$$

end for

end for

for each firefly i do

for each firefly j do

find the max attractiveness and its position

end for

for each job to firefly i do

Move firefly i towards firefly has max attractiveness using

$$Xij = Xij + \beta ij\, e^{-\gamma Dij^2}(Xi - Xj) + (\text{rand} - \frac{1}{2})$$

end for

end for

t←t+1

end while(Esa and Yousif, 2016b)

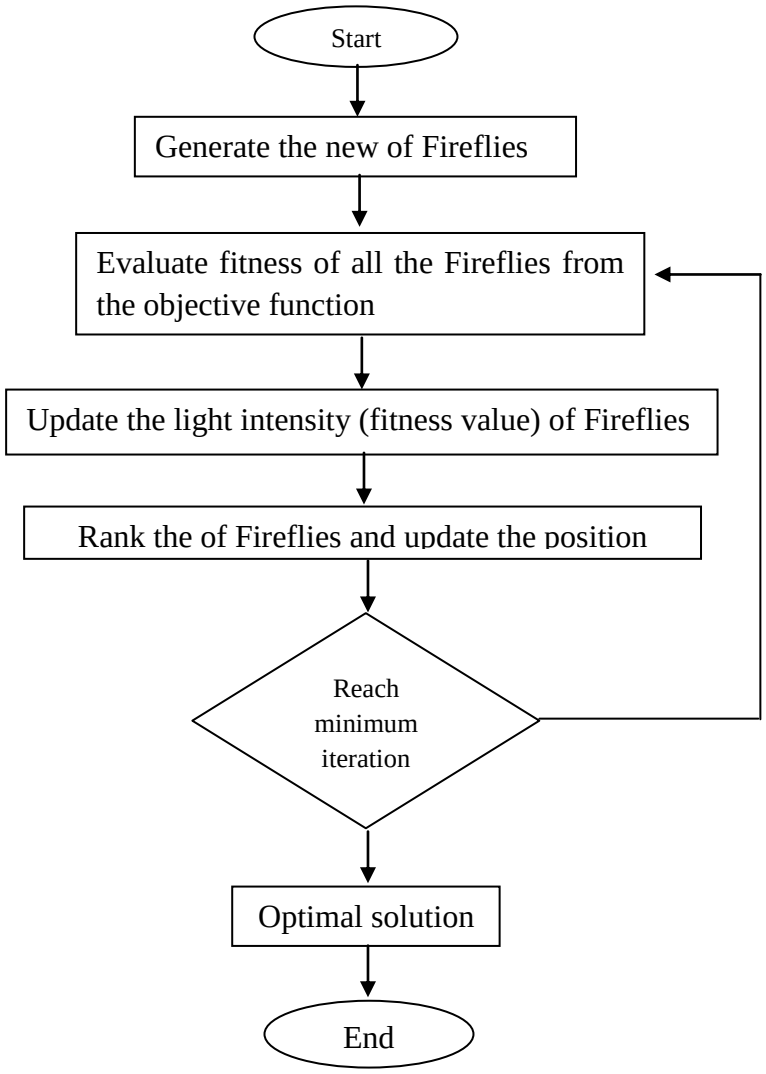

Figure (4.4) Flowchart of Firefly Algorithm(SundarRajan et al., 2016)

4.3 CloudSim

For measuring the execution time of task scheduling algorithms simulation environment are required. CloudSim is the most efficient tool that can be used for modeling of Cloud. During the lifecycle of a Cloud, CloudSim allows VMs to be managed by hosts which in turn are managed by datacenters.

Cloudsim provides architecture with four basic entities. These entities allow user to set-up a basic cloud computing environment and measure the execution time of task scheduling algorithms. A typical Cloud modeled using CloudSim consists of following four entities Datacenters, Hosts, Virtual Machines and Application as well as System Software. Datacenters entity has the responsibility of providing Infrastructure level Services to the Cloud Users. They act as a home to several Host Entities or several instances hosts' entities aggregate to form a single Datacenter entity. Hosts in Cloud are Physical Servers that have pre-configured processing capabilities. Host is responsible for providing Software level service to the Cloud Users. Hosts have their own storage and memory. Processing capabilities of hosts is expressed in MIPS (million instructions per second). They act as a home to Virtual Machines or several instances of Virtual machine entity aggregate to form a Host entity. Virtual Machine allows development as well as deployment of custom application service models. They are mapped to a host that matches their critical characteristics like storage, processing, memory, software and availability requirements. Thus, similar instances of Virtual Machine are mapped to some instance of a Host based upon its availability. Application and System software are executed on Virtual Machine on-demand(Katyal and Mishra, 2014).

Chapter Five

Simulation Results and Performance Analysis

5.1 Introduction

To evaluate the four chosen Swarm Optimization mechanisms for cloud job scheduling this study implemented the four algorithms using CloudSim simulator. Different scenarios were experimented to measure the execution time of each mechanism. The experimentation phase scenarios are simulated as presented in this section.

5.2 Particle Swarm Optimization (PSO)

Three scenarios have been considered to evaluate particle swarm optimization as follows:

5.2.1 The First Scenario

In this scenario, the study considered number of 50 jobs and number of 20 resources.

Table (5.1): The Execution Time of Ten Iterations in First Scenario for PSO

Iteration number	Execution Time
1	153.31738157353482
2	75.39274624154655
3	75.39274624154655
4	75.39274624154655
5	75.39274624154655
6	75.39274624154655
7	75.39274624154655
8	75.39274624154655
9	75.39274624154655
10	75.39274624154655

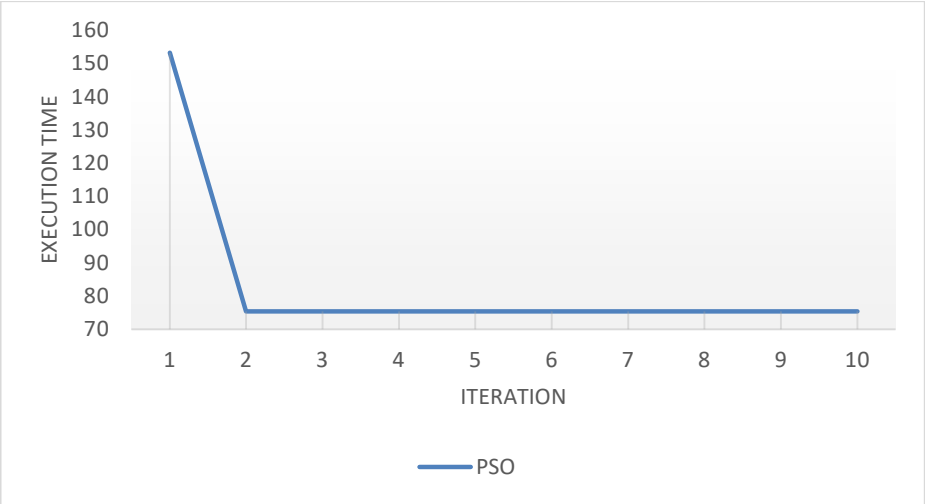

Figure (5.1): The Execution Time of Ten Iterations in First Scenario for PSO

As described in Table (5.1) and Figure (5.1) the result of the initial execution time was 153.31738157353482, and itgradually decreased until it reached 75.39274624154655. Which indicates a better performance from the second iteration.

5.2.2 The Second Scenario

In this scenario the study considered number of 60 jobs and number of 30 resources.

Table (5.2): The Execution Time of Ten Iterations in Second Scenario for PSO

Iteration number	Execution Time
1	212.7475833943387
2	110.69590540178777
3	106.57142970548544
4	106.57142970548544
5	106.57142970548544
6	106.57142970548544
7	106.57142970548544
8	106.57142970548544
9	106.57142970548544
10	106.57142970548544

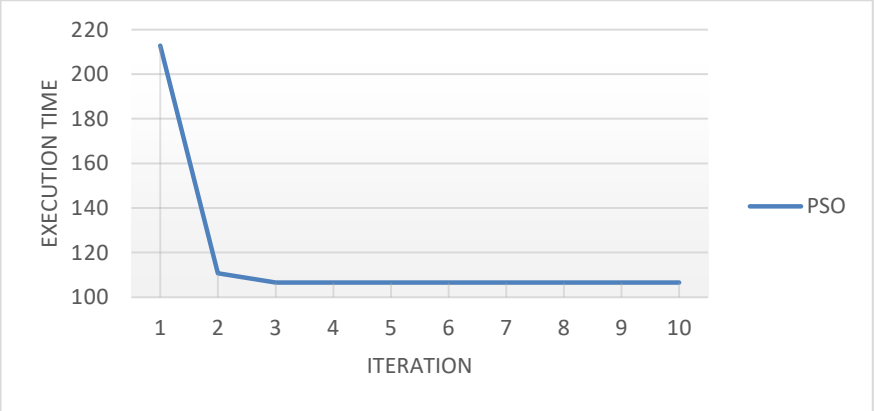

Figure (5.2): The Execution Time of Ten Iterations in Second Scenario for PSO

As described in Table (5.2) and Figure (5.2) the result of the initial execution time was 212.7475833943387then it decreased sharply to106.57142970548544from the second iteration. This indicates that the PSO algorithm succeed in reducing the execution time.

5.2.3 The Third Scenario

In this scenario we considered number of 150 jobs and number of 70 resources.

Table (5.3): The Execution Time of Ten Iterations in Third Scenario for PSO

Iteration number	Execution Time
1	620.5561493178764
2	595.2909102335573
3	595.2909102335573
4	595.2909102335573
5	595.2909102335573
6	595.2909102335573
7	485.75975051088005
8	485.75975051088005
9	485.75975051088005
10	485.75975051088005

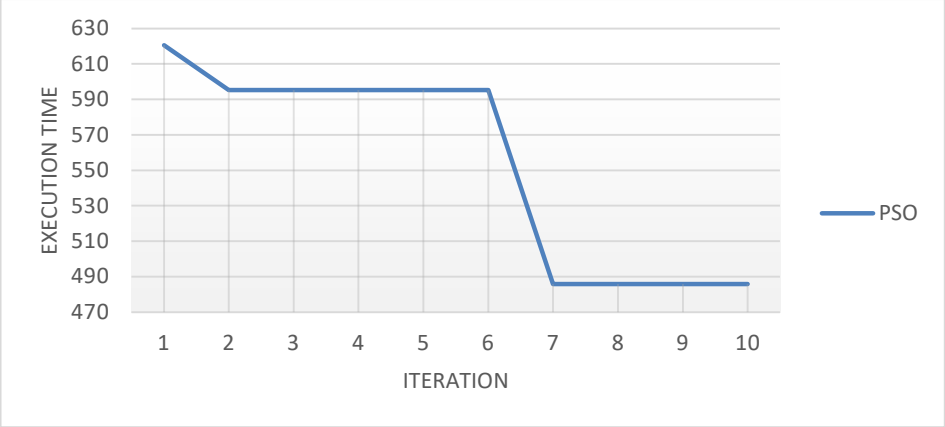

Figure (5.3): The Execution Time of Ten Iterations in Third Scenario for PSO

As described in Table (5.3) and Figure (5.3) the result of the initial execution time was 620.5561493178764, which reduced to 595.2909102335573, the algorithm maintained this time till the 6thiteration, where it sharply changed to 485.75975051088005 till the 10th iteration.

5.3 CatSwarm Optimization (CSO)

Three scenarios have been considered to evaluate Cat swarm optimization as follows:

5.2.3 The First Scenario

In this scenario, the study considered number of 50 jobs and number of 20 resources.

Table (5.4): The Execution Time of Ten Iterations in First Scenario for CSO

Iteration number	Execution Time
1	164.5409268062519
2	118.49798472511321
3	112.2652032736785
4	112.2652032736785
5	112.2652032736785
6	112.2652032736785
7	112.2652032736785
8	112.2652032736785
9	107.00915060172028
10	107.00915060172028

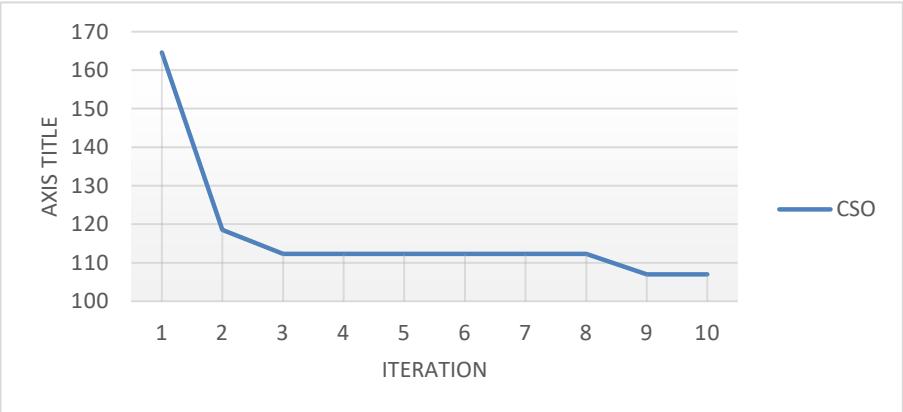

Figure (5.4): The Execution Time of Ten Iterations in First Scenario for CSO

As described in Table (5.4) and Figure (5.4) the result of the initial execution time was 164.5409268062519 gradually decreased until it reached 107.00915060172028. This indicates that the CSO algorithm reduced the execution time.

5.2.3 The Second Scenario

In this scenario we considered number of 60 jobs and number of 30 resources.

Table (5.5): The Execution Time of Ten Iterations in Second Scenario for CSO

Iteration number	Execution Time
1	201.48353261049226
2	118.6076512831467
3	111.01993021078034
4	111.01993021078034
5	111.01993021078034
6	110.73966750285199
7	110.73966750285199
8	104.5230702819497
9	102.04418576064688
10	102.04418576064688

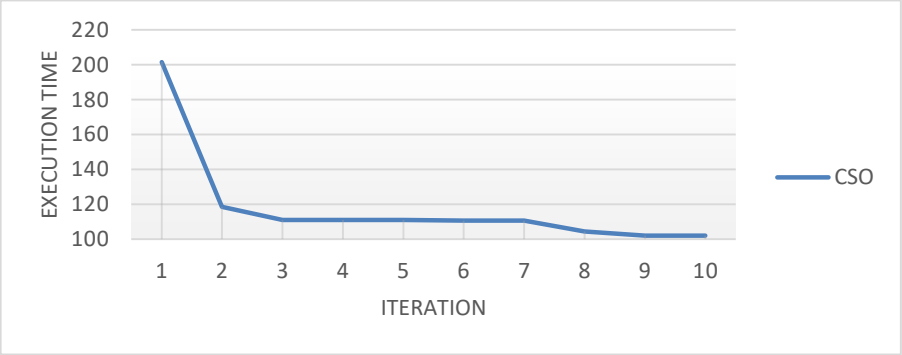

Figure (5.5): The Execution Time of Ten Iterations in Second Scenario for CSO

As described in Table (5.5) and Figure (5.5) the result of the initial execution time was 201.48353261049226, which sharply decreased until it reached 118.6076512831467. This indicates that the CSO algorithm decreased the execution time from the second iteration and continued to enhance its performance till the last iteration with time of 102.04418576064688.

5.3.3 The Third Scenario

In this scenario we considered number of 150 jobs and number of 70 resources.

Table (5.6): The Execution Time of Ten Iterations in Third Scenario for GSO

Iteration number	Execution Time
1	571.4923707790051
2	369.88142445275207
3	369.88142445275207
4	312.253347266184
5	312.253347266184
6	312.253347266184
7	288.5688397467151
8	288.5688397467151
9	288.5688397467151
10	288.5688397467151

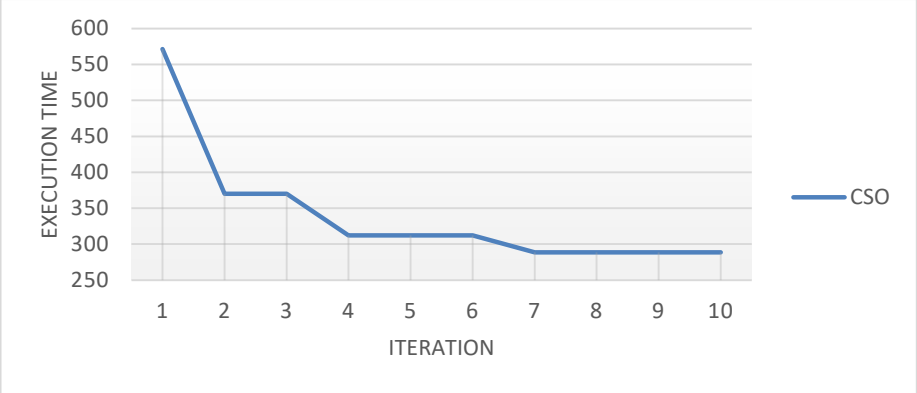

Figure (5.6): The Execution Time of Ten Iterations in Third Scenario for GSO

As described in Table (5.6) and Figure (5.6) the result of the initial execution time was 571.4923707790051, it gradually decreased until it reached 288.5688397467151.

5.4 Firefly Algorithms (FA)

Three scenarios have been considered to evaluate Firefly Algorithms as follows:

5.2.3 The First Scenario

In this scenario, the study considered number of 50 jobs and number of 20 resources.

Table (5.7): The Execution Time of Ten Iterations in First Scenario for FA

Iteration number	Execution Time
1	184.79477586397093
2	184.79477586397093
3	184.79477586397093
4	160.9562009670756
5	160.9562009670756
6	160.9562009670756
7	160.9562009670756
8	160.9562009670756
9	160.9562009670756
10	160.9562009670756

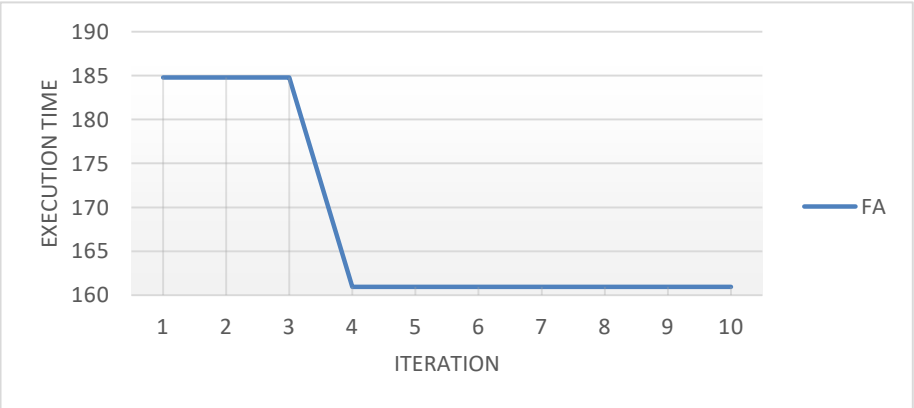

Figure (5.7): The Execution Time of Ten Iterations in First Scenario for FA

As described in Table (5.7) and Figure (5.7) the result of the initial execution time was 184.79477586397093 for the first third iterations the it drastically

decreased until it reached 160.9562009670756, in the 4th iteration and maintained it till the last iteration.

5.2.3 The Second Scenario

In this scenario we considered number of 60 jobs and number of 30 resources.

Table (5.8): The Execution Time of Ten Iterations in Second Scenario for FA

Iteration number	Execution Time
1	222.1725185469669
2	222.1725185469669
3	198.99817797756182
4	176.119549514586
5	176.119549514586
6	176.119549514586
7	176.119549514586
8	176.119549514586
9	176.119549514586
10	176.119549514586

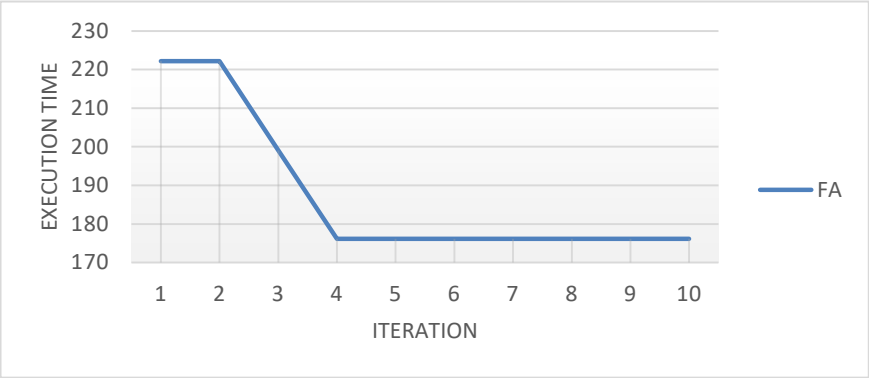

Figure (5.8): The Execution Time of Ten Iterations in Second Scenario for FA

As described in Table (5.8) and Figure (5.8) the result of the initial execution for the first and second iteration was 222.1725185469669. Then it decreased sharply until it reached 176.119549514586 and the same time till the end maintained.

5.4.3 The Third Scenario

In this scenario we considered number of 150 jobs and number of 70 resources.

Table (5.9): The Execution Time of Ten Iterations in Third Scenario for FA

Iteration number	Execution Time
1	691.1954378511617
2	691.1954378511617
3	691.1954378511617
4	691.1954378511617
5	691.1954378511617
6	691.1954378511617
7	651.1540671966757
8	651.1540671966757
9	651.1540671966757
10	543.3117858574376

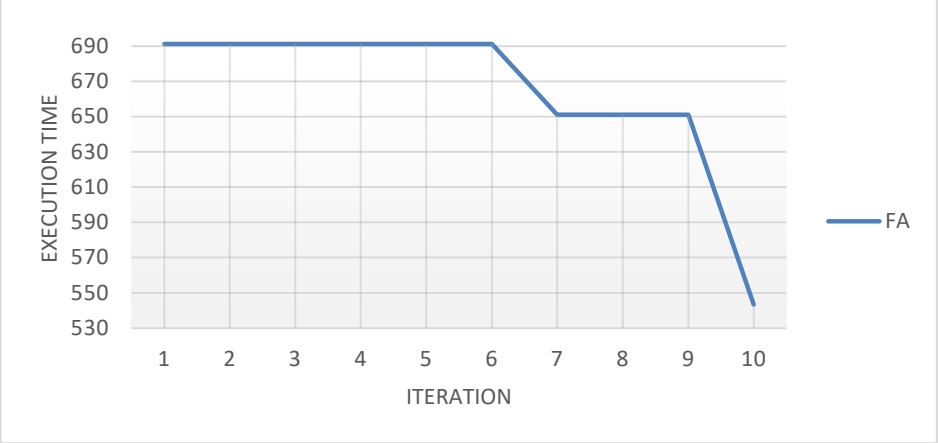

Figure (5.9): The Execution Time of Ten Iterations in Third Scenario for FA

As described in Table (5.9) and Figure (5.9) the algorithm failed to reduce the initial execution time 691.1954378511617 till the 6^{th} iteration where it reduced gradually until it reached 651.1540671966757. However, the algorithm succeeds in reducing the time sharply to 543.3117858574376 at the last iteration.

5.5 Glowworm Swarm Optimization (GSO)

Three scenarios have been considered to evaluate Glowworm Swarm Optimization as follows:

5.2.3 The First Scenario

In this scenario, the study considered number of 50 jobs and number of 20 resources.

Table (5.10): The Execution Time of Ten Iterations in First Scenariofor GSO

Iteration number	Execution Time
1	193.41970639279785
2	193.41970639279785
3	193.41970639279785
4	180.46191974249658
5	180.46191974249658
6	180.46191974249658
7	163.3355664682965
8	163.3355664682965
9	163.3355664682965
10	163.3355664682965

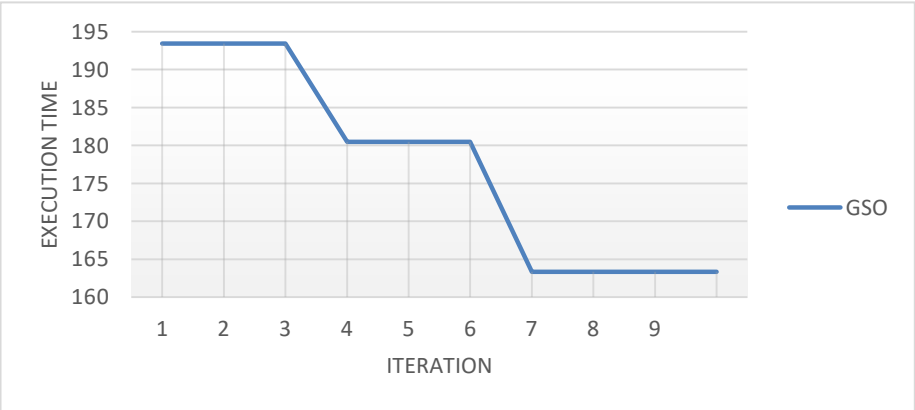

Figure (5.10): The Execution Time of Ten Iterations in First Scenario for GSO

As described in Table (5.10) and Figure (5.10) the result of the initial execution time was 193.41970639279785, which is gradually decreased after each two iterations until it reached 163.3355664682965, after the 7th iteration and maintained the same execution time till the last iteration.

5.2.3 The Second Scenario

In this scenario we considered number of 60 jobs and number of 30 resources.

Table (5.11): The Execution Time of Ten Iterations in Second Scenario for GSO

Iteration number	Execution Time
1	193.41970639279785
2	193.41970639279785
3	193.41970639279785
4	180.46191974249658
5	180.46191974249658
6	180.46191974249658
7	163.3355664682965
8	163.3355664682965
9	163.3355664682965
10	163.3355664682965

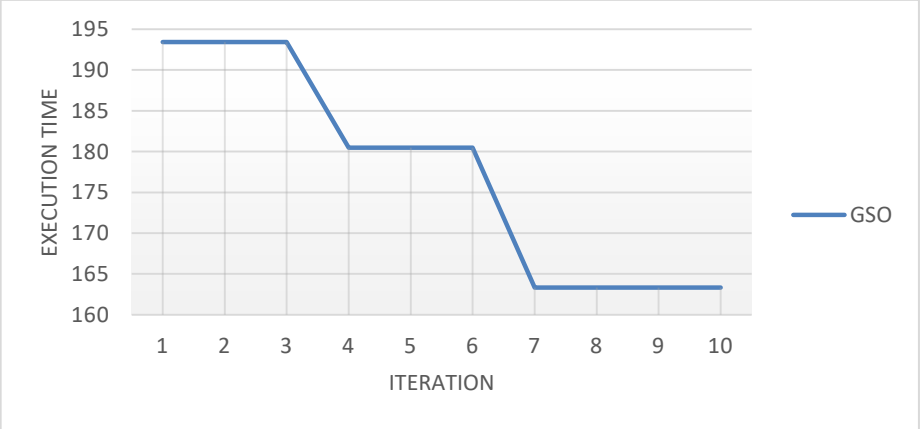

Figure (5.11): The Execution Time of Ten Iterations in Second Scenario for GSO

As described in Table (5.11) and Figure (5.11) the result of the initial execution time was 193.41970639279785, which change gradually after each two

iterations until it reached 163.3355664682965, in the 7th iteration and maintained it till the last iteration.

5.5.3 The Third Scenario

In this scenario we considered number of 150 jobs and number of 70 resources.

Table (5.12): The Execution Time of Ten Iterations in First Scenario for GSO

Iteration number	Execution Time
1	713.5431209169094
2	713.5431209169094
3	612.0312774122625
4	612.0312774122625
5	580.8035105449121
6	580.8035105449121
7	580.8035105449121
8	580.8035105449121
9	580.8035105449121
10	485.4380432574824

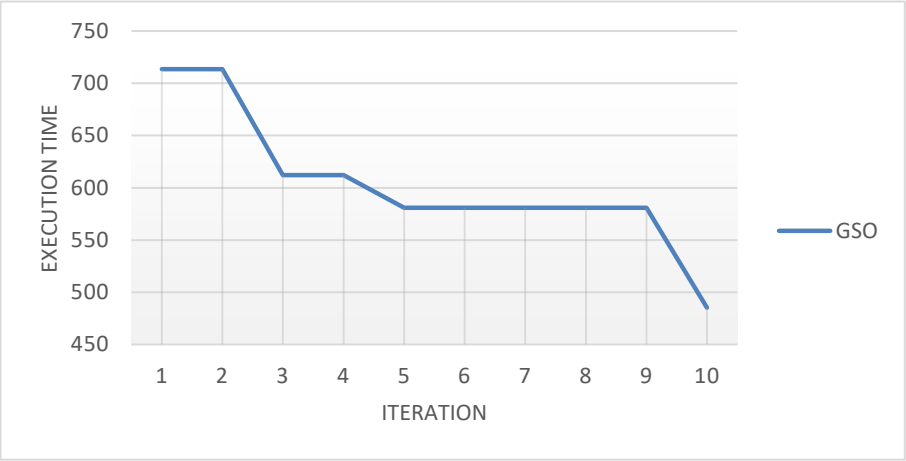

Figure (5.12): The Execution Time of Ten Iterations in First Scenario for GSO

As described in Table (5.12) and Figure (5.12) the result of the initial execution time was 713.5431209169094 gradually decreased until it reached 485.4380432574824.

5.6 Simulation Results and Performance Analysis

Four scenarios have been considered to evaluate and comparative between the algorithms that have been selected as follows:

5.6.1 Comparative in first Scenario

This scenario compares the execution time between the four selected algorithms with the same number of jobs and resources (50 jobs and 20 resources).

Table: (5.13) Comparative in first Scenario

Iteration number	PSO	CAT	Firefly	Glowworm
1	153.31738157353482	164.5409268062519	184.79477586397093	193.41970639279785
2	75.39274624154655	118.49798472511321	184.79477586397093	193.41970639279785
3	75.39274624154655	112.2652032736785	184.79477586397093	193.41970639279785
4	75.39274624154655	112.2652032736785	160.9562009670756	180.46191974249658
5	75.39274624154655	112.2652032736785	160.9562009670756	180.46191974249658
6	75.39274624154655	112.2652032736785	160.9562009670756	180.46191974249658
7	75.39274624154655	112.2652032736785	160.9562009670756	163.3355664682965
8	75.39274624154655	112.2652032736785	160.9562009670756	163.3355664682965
9	75.39274624154655	107.00915060172028	160.9562009670756	163.3355664682965
10	75.39274624154655	107.00915060172028	160.9562009670756	163.3355664682965

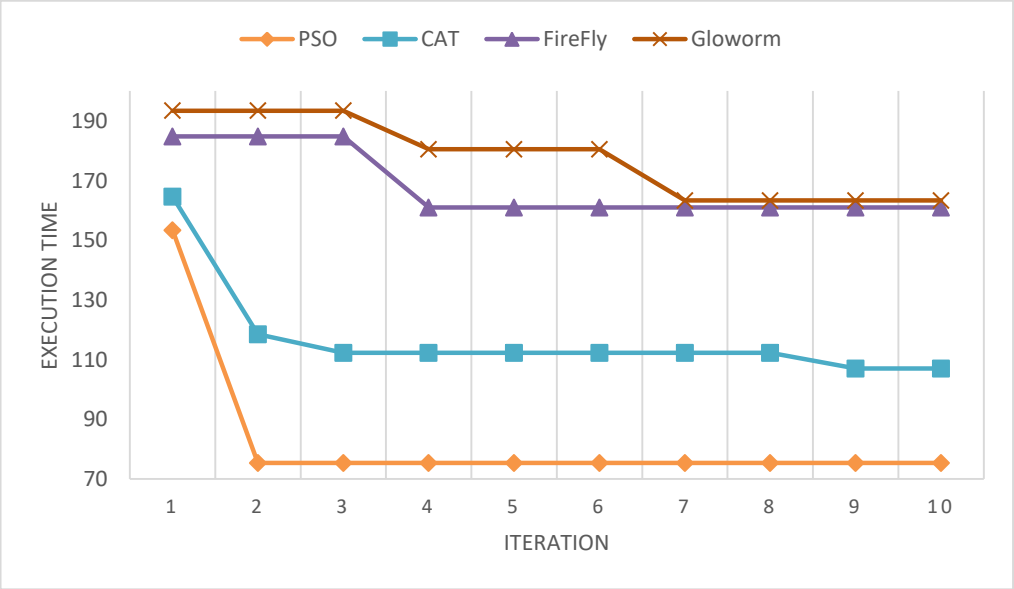

Figure (5.13): Comparative in First Scenario

As described in Table (5.13) and Figure (5.13), in this case a comparison between PSO, CSO, FA and GSO is conducted after computing the execution times for 10iterations, The PSO achieved the best performance in term of execution time among the four algorithms, followed by CSO,FA, and GSO. In each iteration the execution time was decreased and the effectiveness of the proposed mechanisms became better and better.

5.6.2 Comparative in Second Scenario

This scenario compares the execution time between the four proposed Algorithms with the same number of jobs and resources (60 jobs and 30 resources).

Table: (5.14) Comparative in Second Scenario

Iteration number	PSO	CAT	Firefly	Glowworm
1	212.7475833943387	201.48353261049226	222.1725185469669	193.54897425420936
2	110.69590540178777	118.6076512831467	222.1725185469669	193.54897425420936
3	106.57142970548544	111.01993021078034	198.99817797756182	193.54897425420936
4	106.57142970548544	111.01993021078034	176.119549514586	173.16782011570015
5	106.57142970548544	111.01993021078034	176.119549514586	173.16782011570015
6	106.57142970548544	110.73966750285199	176.119549514586	130.67495856826483
7	106.57142970548544	110.73966750285199	176.119549514586	130.67495856826483
8	106.57142970548544	104.5230702819497	176.119549514586	130.67495856826483
9	106.57142970548544	102.04418576064688	176.119549514586	130.67495856826483
10	106.57142970548544	102.04418576064688	176.119549514586	130.67495856826483

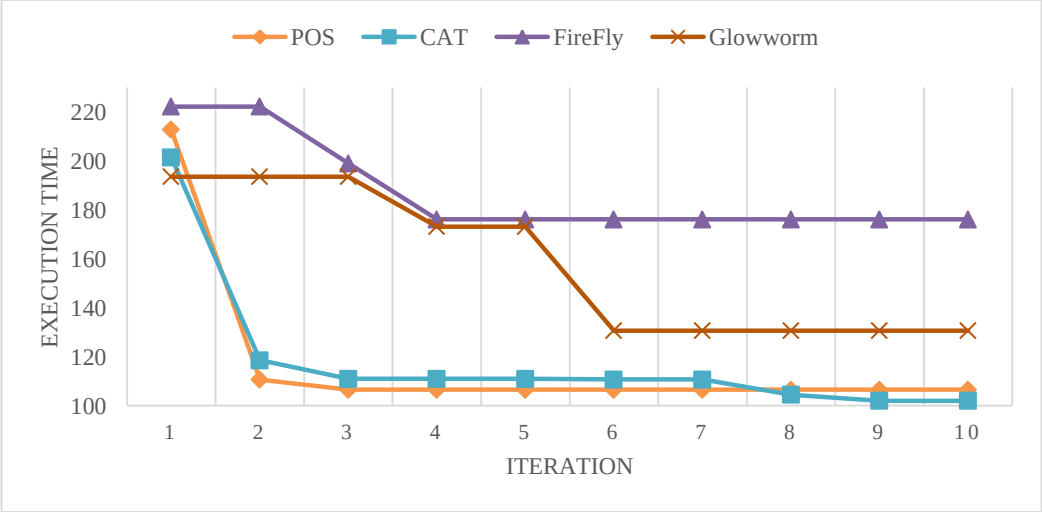

Figure (5.14): Comparative in Second Scenario

As described in Table (5.14) and Figure (5.14), in this case a comparison between PSO, CSO, FA and GSO is conducted after computing the execution times for 10iterations. In term of execution time PSO and CSO achieved the shortest time, with minor differences between the two, where CSO overcome PSO at the last three iterations. The two other algorithms (FA and GSO) stood behind.

5.6.3 Comparative in third Scenario

This scenario compares the execution time between the four proposed Algorithms with the same number of jobs and resources (150 jobs and 70 resources).

Table (5.15): Comparative in third Scenario

Iteration number	PSO	CAT	Firefly	Glowworm
1	620.5561493178764	571.4923707790051	691.1954378511617	713.5431209169094
2	595.2909102335573	369.88142445275207	691.1954378511617	713.5431209169094
3	595.2909102335573	369.88142445275207	691.1954378511617	612.0312774122625
4	595.2909102335573	312.253347266184	691.1954378511617	612.0312774122625
5	595.2909102335573	312.253347266184	691.1954378511617	580.8035105449121
6	595.2909102335573	312.253347266184	691.1954378511617	580.8035105449121
7	485.75975051088005	288.5688397467151	651.1540671966757	580.8035105449121
8	485.75975051088005	288.5688397467151	651.1540671966757	580.8035105449121
9	485.75975051088005	288.5688397467151	651.1540671966757	580.8035105449121
10	485.75975051088005	288.5688397467151	543.3117858574376	485.4380432574824

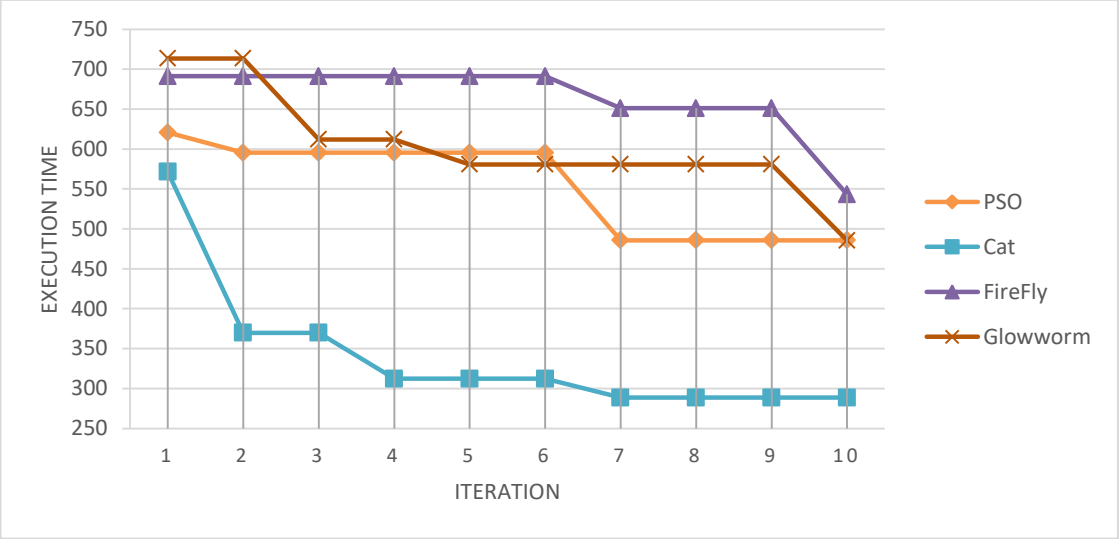

Figure (5.15): Comparative in third Scenario

As described in Table (5.15) and Figure (5.15), in the case of 150 jobs and 70 resources comparison between PSO, CSO, FA and GSO showed the CSO algorithm achieved the best performance in term of execution time. While the three other algorithms varied in their performance, where PSO and GSO reached the same

execution time at last iteration. Glowworm started with the longest execution time in the first two iterations and it started to perform better in (4-9) iteration till it reached the same execution time of PSO in the last iteration, where they reach the second position after CSO. Firefly maintained the same performance till the sixth iteration where it started decrease the time. Despite this change it couldn't overcome the other algorithms and came at the last position.

5.6.4 The Fourth Scenario

In this scenario, different numbers of jobs and resources are considered to evaluate the Swarm Intelligence mechanisms and to examine the performance of the SI mechanisms in different workload.

Table (5.16): The Fourth Scenario

Time	PSO	CAT	Firefly	Glowworm
Time 1 (20 jobs and 10 resources)	30.958730158730155	37.19404761904762	40.01666666666667	38.14960317460318
Time 2 (60 jobs and 30 resources)	106.57142970548544	102.04418576064688	176.119549514586	163.3355664682965
Time 3 (100 jobs and 50 resources)	271.66686481732916	165.85901883246106	307.9912744616086	350.2123042458601
Time 4 (120 jobs and 80 resources)	300.4969150858924	182.83863086605297	309.76665483892725	264.49118590922234
Time 5 (150 jobs and 100 resources)	325.1070806948886	214.88563695884247	691.1954378511617	580.8035105449121

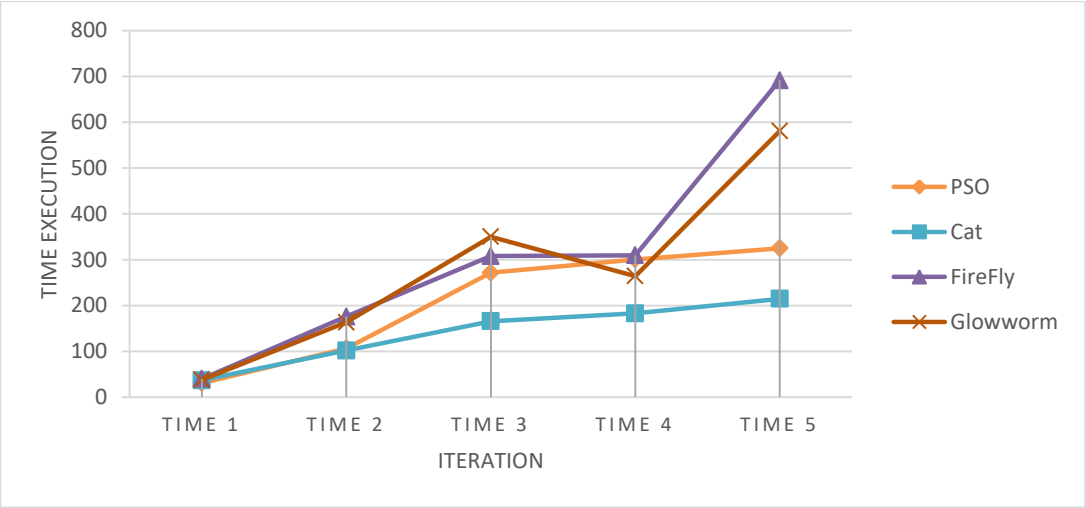

Figure (5.16): The Fourth Scenario

As described in Table (5.16) and Figure (5.16), in this case a comparison between PSO, CSO, FA and GSO is conducted by computing the execution times for each algorithm in different Scenarios. At time1 the PSO had the shortest execution time, however, as the problem size increases by adding more and more jobs the effectiveness of PSO decreased, and CSO came to take the lead in terms of execution time.

5.7 Discussion:

This research aims to provide a perspective on the domain of task scheduling in cloud data centers by summarizing different methods used and to evaluating the swarm intelligence task scheduling algorithms in accordance to task execution time. After conducting simulation using different scenarios to evaluate the selected techniques, the results revealed that when having small sizes of scheduling problems PSO take the lead, however in case of large size of jobs Cat Swarm Optimization significantly outperforms the considered PSO, FA and GSO algorithms in terms of execution time. There have not been any studies conducted comparing the four selected techniques in term of execution time. However, there was a study which compared only two of the techniques in term of the speed of convergence. The study showed that CSO considerable improvement over PSO. This research also shows that CSO over performed PSO and other selected algorithms, but in term of execution

time. However, the results finding from this research were limited to small sized jobs, which may differ in real cloud environment. For future research the job size may be increased and the CSO might be compared to different algorithms using different criteria.

Chapter Six

Conclusions and Recommendation

6.1 Conclusions

In cloud computing, the resources should be utilized effectively and consequently scheduling should consider the resource utilization to decrease the execution time and thereby increasing the throughput of the system. In this research, a simplified four of swarm optimization-based job scheduling algorithms (particles warm optimization, Cat swarm optimization, Firefly algorithm and Glowworm swarm optimization) were implemented for scheduling of jobs in cloud environment in order to minimize the execution time. The performance of the selected methods was compared to each other through carrying out extensive simulation tests. The selected algorithms were tested among four different scenarios. In each scenario the number of jobs and allocation resources were changed. In each scenario, the selected algorithms iteration changes their plans 10 times to come up with the best performance. The execution time is computed in each iteration to compare between the algorithms. Simulation results revealed that when having small sizes of scheduling problems PSO take the lead, however in case of large size of jobs Cat Swarm Optimization significantly outperforms the considered PSO, FA and GSO algorithms in terms of execution time.

6.2 Recommendation

1. Since results revealed that Cat has the best performance among selected algorithm, in the future Cat could be compared to different algorithms rather than (PSO, FA and GSO) to come up with the best algorithm ever and generalize the result.

2. In the future, research works can address other important factors such as the flow time, overall execution cost and load balancing during the scheduling of tasks and jobs.

Refrences

AGARWAL, D. & JAIN, S. 2014. Efficient optimal algorithm of task scheduling in cloud computing environment. *arXiv preprint arXiv:1404.2076*.

AHMED, H. & GLASGOW, J. 2012. Swarm intelligence: concepts, models and applications. *School Of Computing, Queens University Technical Report*.

ALI, S. A. & ALAM, M. A relative study of task scheduling algorithms in cloud computing environment. Contemporary Computing and Informatics (IC3I), 2016 2nd International Conference on, 2016. IEEE, 105-111.

ALNAJDI, S., DOGAN, M. & AL-QAHTANI, E. 2012. ASurvey ON RESOURCE ALLOCATION IN CLOUD COMPUTING.

ANURADHA, V. & SUMATHI, D. A survey on resource allocation strategies in cloud computing. Information Communication and Embedded Systems (ICICES), 2014 International Conference on, 2014. IEEE, 1-7.

ATTIYA, I. & ZHANG, X. 2017. A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing. *International Journal of Computer Applications*, 163.

BILGAIYAN, S., SAGNIKA, S. & DAS, M. Workflow scheduling in cloud computing environment using cat swarm optimization. Advance Computing Conference (IACC), 2014 IEEE International, 2014. IEEE, 680-685.

BOOBA, B. & GOPAL, T. COMPARISON OF ANT COLONY OPTIMIZATION & PARTICLE SWARM OPTIMIZATION IN GRID ENVIRONMENT.

CHALACK, V. A.&GERMI, I. Resource Allocation in Cloud Environment Using Approaches Based Particle Swarm Optimization.

ESA, D. I. & YOUSIF, A. 2016a. Glowworm Swarm Optimization (GSO) for Cloud Jobs Scheduling. *International Journal of Advanced Science and Technology*, .82-71 ,96

ESA, D. I. & YOUSIF, A. 2016b. Scheduling Jobs on Cloud Computing using Firefly Algorithm. *International Journal of Grid and Distributed Computing,* 9**,** 149-158.

FRANCISCO, R. B., COSTA, M. F. P. & ROCHA, A. M. A. Experiments with Firefly Algorithm. International Conference on Computational Science and Its Applications, 2014. Springer, 227-236.

GOYAL, G. & SHARMA, D. 2016. Profit Based Swarm Intelligence Task SchedulingFor Cloud Computing: Review.

GROLINGER, K., HIGASHINO, W. A., TIWARI, A. & CAPRETZ, M. A. 2013. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: advances, systems and applications,* 2**,** 22.

HASHMI, A., GOEL, N., GOEL, S. & GUPTA, D. 2013. Firefly algorithm for unconstrained optimization. *IOSR J Comput Eng,* 11**,** 75-78.

JIN, Y., HOU, W., LI, G. & CHEN, X. 2017. A Glowworm Swarm Optimization-Based Maximum Power Point Tracking for Photovoltaic/Thermal Systems under Non-Uniform Solar Irradiation and Temperature Distribution. *Energies,* 10**,**.541

KAIPA, K. N. & GHOSE, D. 2017. Glowworm Swarm Optimization: Algorithm Development. *Glowworm Swarm Optimization.* Springer.

KATYAL, M. & MISHRA, A. 2014. A comparative study of load balancing algorithms in cloud computing environment. *arXiv preprint arXiv:1403.6918*.

KAUR, A. & DHINDSA, D. K. S. 2016. Analysis of Task Scheduling Algorithms using CloudComputing.

MADNI, S. H. H., LATIFF, M. S. A., ABDULLAHI, M. & USMAN, M. J. 2017. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. *PloS one,* 12**,** e0176321.

MOHAMADDIAH, M. H., ABDULLAH, A., SUBRAMANIAM, S. & HUSSIN, M. 2014. A survey on resource allocation and monitoring in cloud computing. *International Journal of Machine Learning and Computing,* 4**,** 31.

MOHAMMAD, S., BREß, S. & SCHALLEHN, E. Cloud Data Management: A Short Overview and Comparison of Current Approaches. Grundlagen von Datenbanken, 2012. 41-46.

NGENZI, A. & NAIR, S. R. 2015. Dynamic resource management in Cloud datacenters for Server consolidation. *arXiv preprint arXiv:1505.00577*.

PACINI, E., MATEOS, C. & GARCÍA GARINO, C. 2014. Dynamic scheduling based on particle swarm optimization for cloud-based scientific experiments. *CLEI Electronic Journal,* 17**,** 3-3.

PANDEY, S., WU, L., GURU, S. M. &BUYYA, R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. Advanced information networking and applications (AINA), 2010 24th IEEE international conference on, 2010. IEEE, 400-407.

PATIL ,M. S. D. & MEHROTRA, D. S. 2012. Resource allocation and Scheduling in the Cloud. *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS),* 1.

QUAN, D. M., BASMADJIAN, R., DE MEER, H., LENT, R., MAHMOODI, T., SANNELLI, D., MEZZA, F., TELESCA, L. & DUPONT, C. 2011. Energy efficient resource allocation strategy for cloud data centres. *Computer and information sciences II.* Springer.

RAJASEKAR, B. & MANIGANDAN, S. 2015. An Efficient Resource Allocation Strategies in Cloud Computing. *International Journal of Innovative Research in Computer and Communication Engineering,* 3**,** 1239-1244.

SARGA, L. 2012. Cloud computing: an overview. *Journal of Systems Integration,* 3**,** 3.

SUNDARRAJAN, R., VASUDEVAN, V. & MITHYA, S. Workflow scheduling in cloud computing environment using firefly algorithm. Electrical,

Electronics, and Optimization Techniques (ICEEOT), International Conference on, 2016. IEEE, 955-960.

SUREKHA, P. & SUMATHI, S. 2011. PSO and ACO based approach for solving combinatorial Fuzzy Job Shop Scheduling Problem. *Int. J. Comp. Tech. Appl,* 2**,** 112-120.

TSAI, J.-T., FANG, J.-C. & CHOU, J.-H. 2013. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research,* 40**,** 3045-3055.

WANG, C. & CHEN, K. Research on the task scheduling algorithm optimization based on hybrid PSO and ACO in cloud computing.

YADAV, A. K. & MANDORIA, H. L. 2017. Study of Task Scheduling Algorithms in the

Cloud Computing Environment: A Review.