International Journal of Advanced Science and Technology
Vol.96 (2016), pp.71-82
http://dx.doi.org/10.14257/ijast.2016.96.07

Glowworm Swarm Optimization (GSO) for Cloud Jobs Scheduling

Dolagi Izzat Esa and Adil Yousif

University of Science and Technology-Omdurman-Sudan
adiluofk@gmail.com

Abstract

Cloud computing is a new technology provides computing resources as services, and allows users to access these resources via the Internet without the need to own knowledge and experience, or even control of infrastructure that support these services. Job scheduling is considered one of the main issues in cloud computing. The main task of job scheduling is how to find an optimal mapping of set of jobs to a set of available resources. Unsuitable mapping of jobs to resources usually leads to inefficient cloud performance. The current methods for cloud job scheduling process produce acceptable solution but not optimal solution. This paper proposes a new job scheduling mechanism using Glowworm Swarm Optimization (GSO). The proposed mechanism aims to find the best mapping in order to minimize the execution time of jobs. The proposed mechanism based on information of jobs (cloudlets) and resources (virtual machines) such as length of jobs, speed of resources and identifier for both. The scheduling function in the proposed job scheduling mechanism firstly creates a set of jobs and resources to generate the population by assigning the jobs to resources randomly and evaluates the population using fitness values which represent the execution times of jobs. Secondly the function used iterations to regenerate populations based on glowworms behavior 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 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.

Keywords: *Cloud, Job Scheduling, Metaheuristic, Glowworm Swarm Optimization*

1. Introduction

Cloud computing is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer technology. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. It relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network. At the foundation of cloud computing is the broader concept of converged infrastructure and shared services [1].

NIST define of Cloud computing as "a model for 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"[2].

Cloud Computing has become a widely accepted paradigm for high performance computing, because in Cloud Computing all type of IT facilities are provided to the users

ISSN: 2005-4238 IJAST

as a service. The services of the cloud are provided through the Internet. In Cloud Computing the term Cloud is used for the service provider, which holds all types of resources for storage, computing etc. Mainly three types of services are provided by the cloud. First is Infrastructure as a Service (IaaS), which provides cloud users the infrastructure for various purposes like the storage system and computation resources. Second is Platform as a Service (PaaS), which provides the platform to the clients so that they can make their applications on this platform. Third is Software as a Service (SaaS), which provides the software to the users and hence the user don't need to install the software on their own machines and they can use the software directly from the cloud. Cloud Computing provides many benefits: it results in cost savings because there is no need of initial installation of much resource; it provides scalability and flexibility, the users can increase or decrease the number of services as per requirement; maintenance cost is very less because all the resources are managed by the Cloud providers[3].

Job scheduling is one of the major activities performed in all the computing environments. Cloud computing is one the upcoming latest technology which is developing drastically. To efficiently increase the working of cloud computing environments, job scheduling is one the tasks performed in order to gain maximum profit. The goal of scheduling algorithms in distributed systems is spreading the load on processors and maximizing their utilization while minimizing the total task execution time Job scheduling, one of the most famous optimization problems [4, 5]. Job scheduling has been considered as one of crucial problems in cloud computing. An optimized scheduler would improve many factors in scheduling of tasks 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 Firefly Algorithm[6].

Suppose that R= $\{r_1, r_2, r_3 \ldots r_s\}$are s cloud resources and J= $\{j_1, j_2, j_3 \ldots j_m\}$ are m independent jobs. The speed of each resource is expressed in form of MIPS (Million Instructions Per Second), and the length of each job is expressed in the form of number of instructions. The problem is how to allocate the submitted jobs to the available resource in order to complete the jobs efficiently and to minimize the execution time, such that an optimum execution time is achieved. The objective of this paper is to propose a new job scheduling mechanism with modified distance to minimize the execution time based on Glowworm Swarm Optimization algorithm and to evaluate the proposed mechanism using CloudSim simulator.

This paper contains six sections. Section two describes the job scheduling on cloud computing. Section three describes the Glowworm Swarm Optimization (GSO) algorithm. Section four describes the proposed scheduling mechanism. Section five presents the evaluation and experimentation process. Section six provides the Conclusion and Future work.

2. Cloud Job Scheduling

One of the major concerns in cloud computing is job scheduling. It is an extensive research area in cloud computing. As in Figure (1), job scheduling of customers' tasks means how to allocate resources to these tasks. Thus, the required tasks can be accomplished in minimum time according to time defined in user request. Most researches that used in grid computing can also be used in cloud computing setting. The central task of job scheduling system is to determine the most suitable resources for the user's jobs in a cloud computing. Scheduling in cloud computing can be divided into two main views, from the cloud computing users (CCU) and from the cloud computing service provider (CCSP). From the user's point of view, the scheduling algorithm should generally reduce both the execution time and cost. On the other hand, the main concern of

the cloud service provider is that the scheduling algorithm should improve the resource utilization and reduce the maintenance cost along with energy consumption[7].

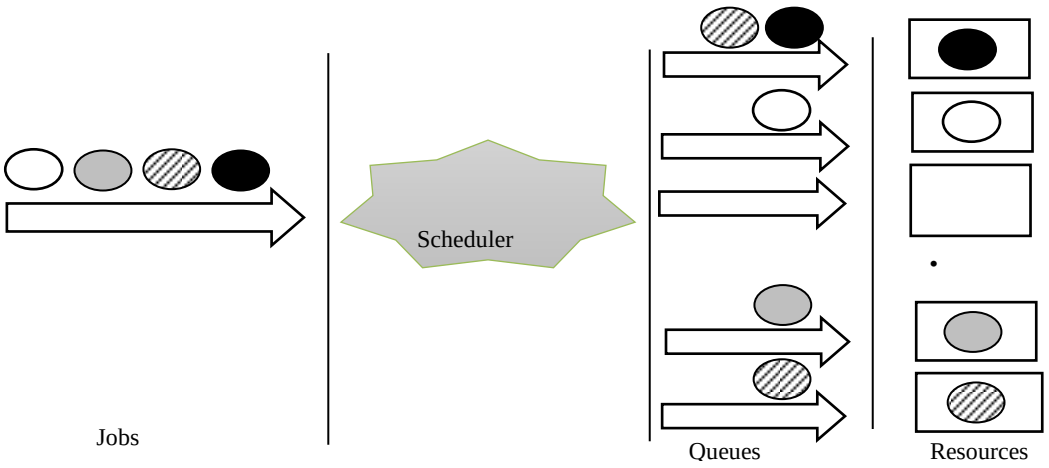

Figure 1. Job Scheduling

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. Swarm intelligence (SI) is the collective behavior of decentralized, self-organized systems, natural or artificial. The concept is employed in work on artificial intelligence.SI systems consist typically of a population of simple agents or interacting locally with one another and with their environment. The inspiration often comes from nature, especially biological systems, including ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling. 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 [8-11].

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)

$$Li(t) = (1-\rho)Li(t-1) + \gamma J(xi(t)) \quad (1)$$

3.2. Movement-phase

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

$$j \in N_{ij} \text{ iff } d_{ij} < rd_i(t) \text{ and } L_j(t) > L_i(t) \quad (2)$$

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

$$p_{ij}(t) = \frac{L_j(t) - L_i(t)}{\sum_{k \in N_i(t)} L_k(t) - L_i(t)} \quad (3)$$

The glowworm i position is updated using equation (4)

$$X_i(t+1) = X_i(t) + s\left(\frac{X_j(t) - X_i(t)}{\|X_j(t) - X_i(t)\|}\right) \quad (4)$$

where s is the step size.

3.3. Local-decision Range Update Rule

The neighborhood range is updated using equation (5)

$$rdi(t+1) = \min\{rs, \max\{0, rdi(t) + \beta(nt - |Ni(t)|)\}\} \quad (5)$$

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 [12, 13].

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 neighbours (nt), the sensor range (rs) and a constant value (β)[14].

Parameters values of Glowworm Algorithm that are Kept Constant for all experiments are described in Table 1 [8, 9, 11, 15-17].

Table 1. Glowworm Optimization Parameters

P	γ	β	n_t	S	L_0
0.4	0.6	0.08	5	0.03	5

The basic Glowworm Swarm Optimization (GSO) Algorithm as follows[9, 17]
Set number of dimensions = m
Set number of glowworms = 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** $L_i(0) = L_0$
for i = 1 to n **do** $rd_i(0) = r_0$
Set maximum iteration number = iter max
Set t = 1
while (t ≤ iter max) **do** {
for each glowworm i **do** % Luciferin-update phase
$L_i(t) = (1-\rho)L_i(t-1) + \gamma J(x_i(t))$
for each glowworm i **do** % Movement-phase {
$N_i(t) = \{j : d_{ij}(t) < rd_i(t); L_i(t) < L_j(t)\}$
for each glowworm $j \in N_i(t)$ **do**
$$p_{ij}(t) = \frac{L_j(t) - L_i(t)}{\sum k \in N_i(t)\, L_k(t) - L_i(t)}$$
j = select glowworm(p)
$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right)$$
$rd_i(t+1) = \min\{r_s, \max\{0, rd_i(t) + \beta(nt - |N_i(t)|)\}v\}$ }
$t \leftarrow t + 1$ }

4. Proposed Glowworm Swarm Optimization for Cloud Job Scheduling

In the proposed mechanism, Glowworm Swarm Optimization is employed in solving the problem of independent job scheduling and allocation of the jobs on resources. Each glowworm is a solution for allocation of jobs $\overrightarrow{X_{ij}}$, $i = (1,2,3,\ldots,n)$ $j = (1,2,3,\ldots,k)$, each element inside the glowworm population vector is a random number between 1 to m where:

m is the total number of resources.

n is number of glowworms.

k is number of jobs that represent the length of each glowworm.

We represented resources as a vector for storing the speed of each resource $\overrightarrow{R_i}$, $i = (1,2,3,\ldots,m)$ and also jobs as a vector for storing the length of each job $\overrightarrow{J_i}$, $i = (1,2,3,\ldots,k)$, then assigned the luciferin for each glowworms Li(0) = L0 and local decision range rdi(0) = r0 for each glowworms as the same in the beginning. Then the proposed mechanism calculated the fitness function $F(\overrightarrow{X_{ij}})$ for each glowworm by dividing each job length by the resource speed that the job is allocated to, and then find the summation of the division results, then find the maximum fitness, Glowworm that has maximum fitness either move randomly or not move at all. The next step is to calculate the distance between each two glowworms which is the number of non-corresponding elements in the glowworm population[18] and store it in $\overrightarrow{D_{ij}}$, $i = (1,2,3,\ldots,n)$ $j = (1,2,3,\ldots,n)$.

Finally the movement for all glowworm, which starts with the update of the luciferin for each glowworm according to equation (1):

$$Li(t) = (1-\rho)Li(t-1) + \gamma F(xi(t)) \quad (1)$$

Where:

Li(t) is the new luciferin level for glowworm i.

Li(t − 1) is the previous luciferin level for glowworm i.

ρ is the luciferin decay constant.

γ is the luciferin enhancement constant.

F(xi(t)) is the fitness function value for glowworm i at current glowworm position (xi) at iteration t.

Then each glowworm i determine the neighbor glowworm j according the higher luciferin and in the same local decision range $j \in N_{ij} \text{ iff } d_{ij} < rd_i(t) \text{ and } L_j(t) > L_i(t)$, neighbors j that belongs to glowworm i decision range calculate the probability by equation (2)

$$p_{ij}(t) = \frac{L_j(t) - L_i(t)}{\sum_{k \in N_i(t)} L_k(t) - L_i(t)} \quad (2)$$

Where:

j is one of the neighbor group Ni(t) of glowworm i.

$L_j(t)$is the luciferin levels for glowworm j.

$L_i(t)$is the luciferin levels for glowworm i.

Then glowworm i moves towards a neighbor j with the higher probability and updated the position of glowworm i by equation (3)

$$X_i(t+1) = X_i(t) + s\left(\frac{X_j(t) - X_i(t)}{\|X_j(t) - X_i(t)\|}\right) \quad (3)$$

Where:
$X_i(t+1)$ is the new position for the glowworm i.
$X_i(t)$ is the current position for the glowworm i.
s is a step size constant.
Finally update the local decision range by equation (4)

$$rdi(t+1) = \min\{rs, \max\{0, rdi(t) + \beta(nt - |Ni(t)|)\}s\} \quad (4)$$

Where:
rdi(t + 1) is the new local decision range for glowworm i
rdi(t) is the previous local decision range for glowworm i
β is a constant parameter.
rs is the constant radial sensor range.
nt is a constant parameter used to control the number of neighbors.
|Ni(t)| is the actual number of neighbors.

4.1. Pseudo Code for the Proposed GSO Algorithm

Begin
Initialize parameter: $l_0, r_0, \rho, \gamma, \beta$ *,nt,rd,m,t, iter_max.*
Generate initial population of glowworms $\overrightarrow{X_{iJ}}$, $i = (1,2,3,\ldots.,n)$ $j = (1,2,3,\ldots.,k)$.
Set maximum iteration number=iter_max.
Set t=1
For *each resource* ***do***
Set speed for each resource $\overrightarrow{R_i}$
end for
for *each job* ***do***
Set length for each job $\overrightarrow{J_i}$
end for
while*(t<=iter_max)*
for *each glowworm I* ***do***
Set luciferin for each glowworms as the same $L_i(0) = L_0$
end for
for *each glowworm i* ***do***
Set local decision range for each glowworms as the same $rd_i(0) = r_0$
end for
for *each glowworm i* ***do***
Compute Fitness function $F(\overrightarrow{X_{iJ}})$
end for
for *each glowworm i* ***do***
for *each glowworm j* ***do***
Compute the distance between glowworm i and glowworm j $\overrightarrow{D_{iJ}}$
end for
end for
for *each glowworm i* ***do*** *// Luciferin-update phase*
$L_i(t) = (1-\rho)L_i(t-1) + \gamma F(x_i(t))$
end for
for *each glowworm* ***i do*** *// Movement-phase*
Find the neighbors j
$N_i(t) = \{j : d_{ij}(t) < rd_i(t);\ L_i(t) < L_j(t)\}$
For *each neighbor glowworm* $j \in N_i(t)$ ***do***
Calculate the probability

$$p_{ij}(t) = \frac{L_j(t) - L_i(t)}{\sum k \in N_i(t) L_k(t) - L_i(t)}$$

end for

Glowworm i move towards a neighbor j: Select glowworm j based on maximum p_{ij}

end for

Update position of glowworm i

$$X_i(t+1) = X_i(t) + s\left(\frac{X_j(t) - X_i(t)}{\|X_j(t) - X_i(t)\|}\right)$$

Update Local-decision range for glowworm i

$$rd_i(t+1) = \min\{rs, \max\{0, rd_i(t) + \beta(nt - |N_i(t)|)\}\}$$

end for

$t \leftarrow t+1$

end while

5. Evaluation and Experimentation

This paper proposed a new scheduling mechanism on cloud computing using Glowworm Swarm Optimization algorithm.

To evaluate the proposed Glowworm Swarm Optimization mechanism for cloud job scheduling this study implemented the algorithm using CloudSim simulator. Different scenarios are experimented to study various aspect of the mechanism. Execution time is used as measurement as illustrated previously. The experimentation phase scenarios are simulated as presented in the related works. The experiments setup and configuration and the values of the algorithm parameters are assigned based on the related works and the literature of the algorithm.

5.1. The First Scenario

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

Table 2. The Execution Time of Ten Iterations in First Scenario

Iteration	Execution Time
1	219.5833863
2	125.1402317
3	110.2781259
4	108.7364261
5	108.7364261
6	97.40285329
7	97.40285329
8	97.40285329
9	97.40285329
10	97.40285329

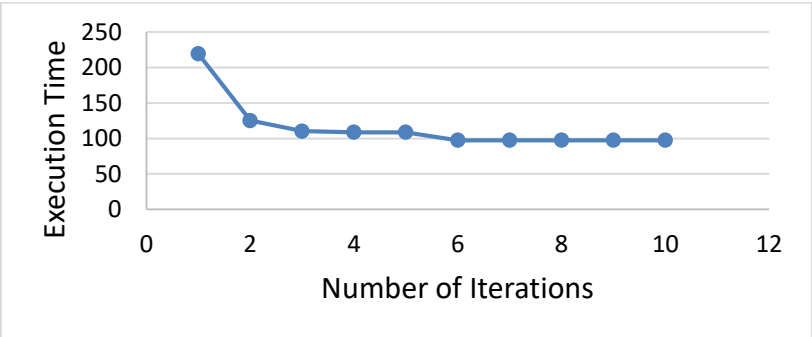

Figure 2. CloudSim Simulation Result in First Scenario

As described in Table 2 and Figure 2, the result of the initial execution time is 219.5833863 gradually decreased until it reached 97.40285329 in the last iteration.

5.2. The Second Scenario

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

Table 3. The Execution Time of Ten Iterations in Second Scenario

Iteration	Execution Time
1	155.7707116
2	155.7707116
3	155.7707116
4	155.7707116
5	141.4966578
6	141.4966578
7	141.4966578
8	133.7815208
9	133.7815208
10	133.7815208

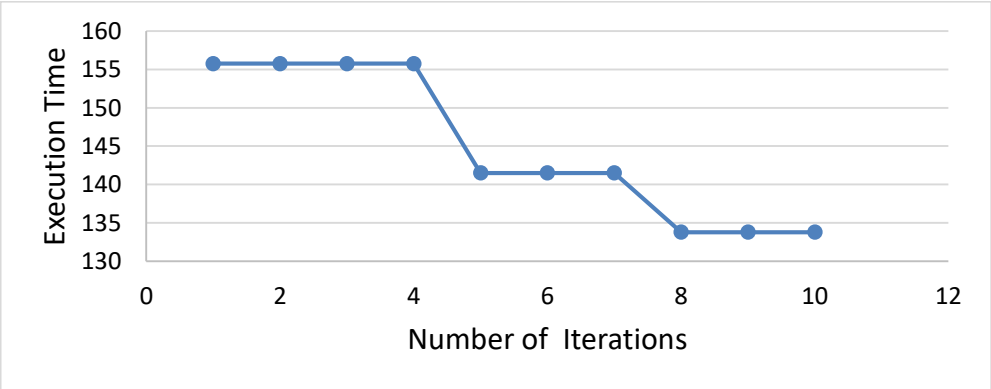

Figure 3. CloudSim Simulation Result in Second Scenario

As described in Table 3 and Figure 3, the result of the initial execution time is 155.7707116gradually decreased until it reached 133.7815208.

5.3. The Third Scenario

This scenario presents a comparison of the execution time between GSO and FCFS algorithm with the same number of jobs and resources (150 jobs and 70 resources).

Table 4. The Comparison between GSO and FCFS Algorithm

Algorithm	Execution Time
FCFS	898.3541286
Random solution	611.8881145
GSO	478.0460448

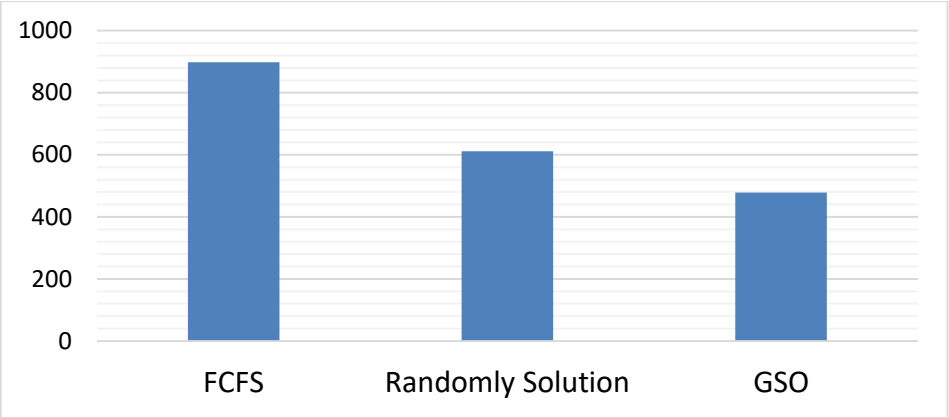

Figure 4. Comparison of Simulation Result in Third Scenario

As described in Table 4 and Figure 4, as the proposed GSO has the shortest execution time. While the random distribution of jobs on the resources has the worst execution time. FCFS has larger execution time than the random distribution. GSO performed better than FCFS in term of execution time.

5.4. The Fourth Scenario

In this scenario, different numbers of jobs and resources are considered to evaluate the proposed mechanism and to examine the performance of the proposed mechanism in different workload and from different perspectives.

Table 5. Average Execution Time between GSO and FCFS Algorithm

	FCFS	GSO
Time 1 (20 jobs and 10 resources)	49.28968253968254	36.602777777777774
Time 2 (60 jobs and 30 resources)	179.8496139276117	152.12253170980117
Time 3 (100 jobs and 50 resources)	324.9602669164712	287.8011214968593
Time 4 (120 jobs and 80 resources)	462.28344311491	292.8789355882527
Time 5 (150 jobs and 100 resources)	599.9205338329427	331.3020118099746
Average Time	323.2607080663236	220.14147567653310

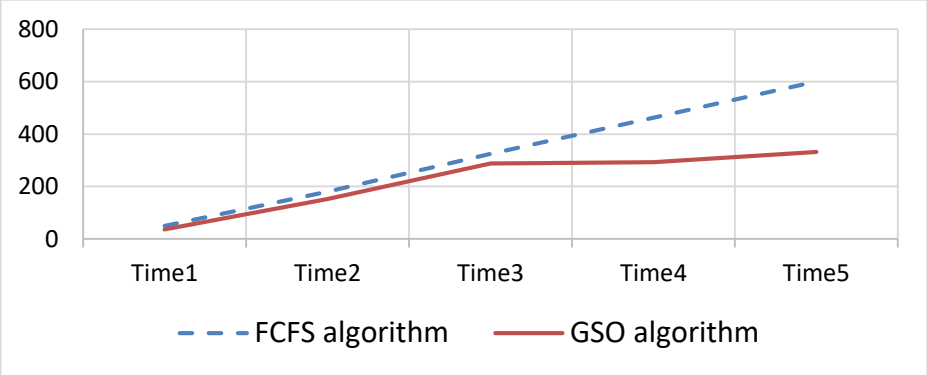

Figure 5. Comparison of Simulation Result in Fourth Scenario

As described in Table 5 and Figure 5, in this case a comparison between GSO and FCFS algorithm is conducted after computing the average times for 5 executions, The GSO performed better than FCFS based on the execution time. As the problem size increases by adding more and more jobs the effectiveness of the proposed mechanism became better and better. The previous results revealed that the proposed Glowworm

Swarm Optimization algorithm for job scheduling in cloud optimizes the fitness value. As in all scenarios the fitness increase significantly which indicates that the execution time is decreased and enhanced.

6. Conclusion and Future Work

This research proposed a new job scheduling mechanism to solve the scheduling problems by minimizing the execution time of jobs using Glowworm Swarm Optimization (GSO). The research work started with a mapping process to map the cloud job scheduling mechanism with the GSO. In this mapping each job scheduling solution represents a glowworm. A random number of solutions have been selected to represent the initial glowworm population. The execution time of each scheduling is considered the fitness function of the schedule (glowworm). In each iteration the schedules (glowworms) move based on the Luciferin of the scheduling to generate a new population with enhanced fitness. To find the best solution in each iteration we select the schedule that has the best fitness. The evaluation of the proposed job scheduling mechanism is based on CloudSim simulator, conducting various experiments and a comparison with FCFS algorithm. The proposed GSO mechanism has significantly reduced the execution times of the cloud jobs. Using small number of jobs, the proposed mechanism has the best execution time. Certainly, when increasing the number of jobs the execution time has increased. However, the execution time of proposed mechanism is still less than First Come First Serves (FCFS). The results revealed that the proposed mechanism outperforms FCFS algorithm in minimizing the execution time of jobs.

References

[1] R. Kaur and S. Kinger, "Analysis of Job Scheduling Algorithms in Cloud Computing", International Journal of Computer Trends and Technology (IJCTT), vol. 9, **(2014)**, pp. 379-386.
[2] P. Mell and T. Grance, "The NIST definition of cloud computing", **(2011)**.
[3] P. Kumar and A. Verma, "Scheduling using improved genetic algorithm in cloud computing for independent tasks", Proceedings of the International Conference on Advances in Computing, Communications and Informatics, **(2012)**, pp. 137-142.
[4] P. Salot, "A survey of various scheduling algorithm in cloud computing environment", IJRET: International Journal of Research in Engineering and Technology, ISSN, **(2013)**, pp. 2319-1163.
[5] M. Aboalama and A. Yousif, "Enhanced Job Scheduling Algorithm for Cloud Computing Using Shortest Remaining Job First (SRJF)", International Journal of Computer Science & Management Studies, vol. 15, **(2015)**, pp. 65-68.
[6] D. I. Esa and A. Yousif, "Scheduling Jobs on Cloud Computing using Firefly Algorithm", International Journal of Grid and Distributed Computing, vol. 9, **(2016)**, pp. 149-158.
[7] M. Maqableh, H. Karajeh, and R. e. Masa'deh, "Job Scheduling for Cloud Computing Using Neural Networks", Communications and Network, vol. 2014, **(2014)**.
[8] Z. Huang and Y. Zhou, "Using glowworm swarm optimization algorithm for clustering analysis", Journal of Convergence Information Technology, vol. 6, **(2011)**, pp. 78-85.
[9] K. Krishnanand and D. Ghose, "Glowworm swarm optimization for multimodal search spaces", in Handbook of Swarm Intelligence, ed: Springer, **(2011)**, pp. 451-467.
[10] K. Krishnanand and D. Ghose, "Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions", Swarm intelligence, vol. 3, **(2009)**, pp. 87-124.
[11] K. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications", Multiagent and Grid Systems, vol. 2, **(2006)**, pp. 209-222.
[12] A. G. Karegowda and M. Prasad, "A Survey of applications of glowworm swarm optimization algorithm", IJCA Proceedings on International Conference on Computing and information Technology 2013, **(2013)**, pp. 39-42.
[13] I. Aljarah and S. Ludwig, "A MapReduce based glowworm swarm optimization approach for multimodal functions", Swarm Intelligence (SIS), 2013 IEEE Symposium, **(2013)**, pp. 22-31.
[14] R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey", International Journal of Bio-Inspired Computation, vol. 3, **(2011)**, pp. 1-16.
[15] G. Zhao, Y. Zhou, and Y. Wang, "The glowworm swarm optimization algorithm with local search operator", Journal of Information & Computational Science, vol. 9, **(2012)**, pp. 1299-1308.

[16] Y. Zhou, Y. Wang, S. He and J. Wu, "A novel double glowworm swarm co-evolution optimization algorithm based levy flights", Appl. Math, vol. 8, **(2014)**, pp. 355-361.
[17] Y. Zhou, Q. Luo and J. Liu, "Glowworm Swarm Optimization for Optimization Dispatching System of Public Transit Vehicles", Journal of Theoretical & Applied Information Technology, **(2013),** vol. 52.
[18] A. Kumbharana and G. M. Pandey, "Solving travelling salesman problem using firefly algorithm", International Journal for Research in Science & Advanced Technologies, vol. 2, **(2013)**, pp. 53-57.