An Improved GSO based Task Scheduling (IGSOTS) Algorithm for Load Balancing in Cloud Environment

R. Raja[1], A. Karthikeyan[2]
Associate Professor[1], Assistant Professor[2]
Department of CSE[1, 2],
St. Martin's Engineering College[1, 2],
Secunderabad, Telengana, India.
nsraja1984@gmail.com[1], keyanruvan@yahoo.co.in[2]

Abstract: Computing services have become cheaper, more efficient and more widespread than ever with the exponential advancement of the computing and storage systems or the popularity of the internet. Task scheduling is a big issue in cloud systems, specifically influencing the usage of services. If an appropriate load balance is done in the cloud, successful utilization of resources is made. To fix the load balancing problem, a novel cloud load balancing using the Improved Glow-worm Swarm Optimization Task Scheduling (IGSOTS) algorithm is proposed. In addition, in order to achieve better performance, the objective function is built on the basis of three criteria, including completion time, processing costs and load. Finally, the efficiency of the proposed technique is measured in terms of various parameters, including the cost of execution and the time of execution. Experimental findings demonstrate that the proposed solution produces improved load balancing performance compared to other approaches.

Keywords: Load balancing, Task Scheduling, IGSO algorithm, Virtual machine.

1. INTRODUCTION

In the academic and business, cloud infrastructure is making substantial strides. The company is innovative in contrast to history and philosophy, but relies on consumer needs. This is a web-based registry with funds for reimbursement for all use premises [1] [2]. It invention is intended as a service to end clients to deliver distribute, virtualized and versatile properties. It will enhance the complete identification of processing as an asset soon as possible. Cloud Stages allows consumers to connect think-tank services as virtual computers by virtualization creativity [3]. In the modern environment Virtual machine (Computer Machine), which is the category of a few single fathered computational processes [4], the virtualization of an operating structure typically infers. Clients that use a large number of virtual machines, but it is difficult to physically allocate assignments to record assets in clouds. So we require a competent estimate for cloud activity preparation.

In cloud circumstances, task scheduling is a major challenge. This is used to schedule activities for efficient utilization of assets by allocating resources to different assets explicitly to overcome the load and execution of frameworks. An effective task scheduling strategy [5] requires consumer needs to be met, and the whole system needs to be strengthened. In any case, as the number of clients is increasing, and in the event that an adequate load balancing is not carried out, the cloud processing execution is decreasing at that stage [6]. Calculations and instruments for the competent load balancing are important for the fulfilment of the cloud processing requirements for task scheduling systems. Load balancing [7] is the way the total load of a distributed system is transferred across nodes to ensure the node is not overwhelmed, and nodes are not packed. Therefore, load balancing in a cloud environment means that there is no overloading of VMs when certain VMs are under load or do absolutely no job.

Load balancing attempts to speed up the pace of uses execution. It also guarantees the soundness of the Framework. It is a good program of reinforcement due to the bombing over. The main downside to using traditional methods to balance the load within a cloud situation is that the lion's share is attempting to migrate overloaded VM [8]. This VM movement technique has a few disadvantages: (1) It aims to transfer cluttered memory after internet based VM has been duplicated, (2) For both essential physical engine and new host PM, it takes a lot of memory, (3) It wants to prevent crucial VM through triggering down time for VM. A number of researchers have approached this issue using the optimization technique to resolve such pitfalls at the moment. The previously suggested optimization strategy is the genetic algorithm [9], simulated annealing, bee colony algorithm [10], particle swarm optimisation [11].

In this paper, we present load balancing based task scheduling using Improved Glow-worm Swarm Optimization (IGSO) to increase the efficiency of executing individual scheduled tasks. The cost of executing a task considers the time of completion and the price of using the VM. Completion period involves the task's waiting time and execution time and meeting user criteria should be within the user-defined deadline. The performance of the proposed IGSO based task scheduling (IGSOTS) algorithm, is evaluated with a comparative simulation study with Firefly algorithm (FA), and Dragonfly algorithm (DA). The structure of the paper is as follows: Chapter 2 presents the literature survey. Load Balancing System with solution framework is described in Chapter 3. Chapter 4 describes about Basic Glow-worm swarm optimization (GSO) and Chapter 5 discussed about the Improved Glow-worm swarm optimization (IGSO) algorithm. The result and analysis is presented in Chapter 6. Chapter 7 describes the conclusion.

2. LITERATURE SURVEY

He et al. [12] developed a cloud computing task scheduling framework to improve response time and transfer time for task scheduling problems. A technique for efficient resource usage, total time of work, average expense and average energy used was demonstrated with PSO-based Adaptive Multi-objective Task Scheduling (AMTS). The adaptive acceleration coefficient was introduced for the conservation of particle diversity.

In addition, Zhang and Zhou [13] developed a system focused on a two-stage strategy to optimize task scheduling efficiency and eliminate non-reasonable task allocation in clouds. Throughout the first stage, a classifier based on the Bayes classifier design method has been used to identify activities based on statistical scheduling data utilizing virtual machines (VMs) to save the creation of VMs. Dynamically, tasks with concrete VMs are balanced at the second stage. Interactive algorithm task planning was effectively discovered. Chunlin et al.[14] have found an effective method for load-balancing cloud-specific resource scheduling for mobile users by increasing the chances of achieving service level agreements with the local cloud service pools. In the hybrid mobile cloud device, the system status information such as mobile application preferences, resources, server load in the cloud data center was used to improve the resource usage and mobile user experience quality.

Following this, Guo et al.[15] introduced a queuing model for heterogeneous and dynamic workloads. The VM scheduling was conceived as an action-making method in a queuing cloud computing environment where the control parameter was the vector of VM configurations and the optimization goal was the latency value in terms of relative task finishing period. To evaluate the solutions, an online low-complex scheme incorporating the shortest-job-first (SJF) buffering and the min-min best-fit (MMBF) scheduling algorithms, i.e. SJF-MMBF, was obtained. In the same way, Niknam et al.[16] provided a method for determining the replication factor for each task in the acyclic synchronous data flow (SDF) graph, so that by distributing workloads among more parallel tasks with a lower utilization in the transformed graph obtained, the left processor capacity could be effectively exploited, thus reducing the number of processors needed to schedule. Mostly because, these works struggle to resolve the question of overload when scheduling the job.

Neelima et al. [17] implemented ADA based multi-objective Load balancing method to acquire a well-balanced load through virtual machines and operate minimum of time and expense In addition, this algorithm demonstrates the ADA benefits in maximizing the task planning and distribution of resources in a cloud computing framework.

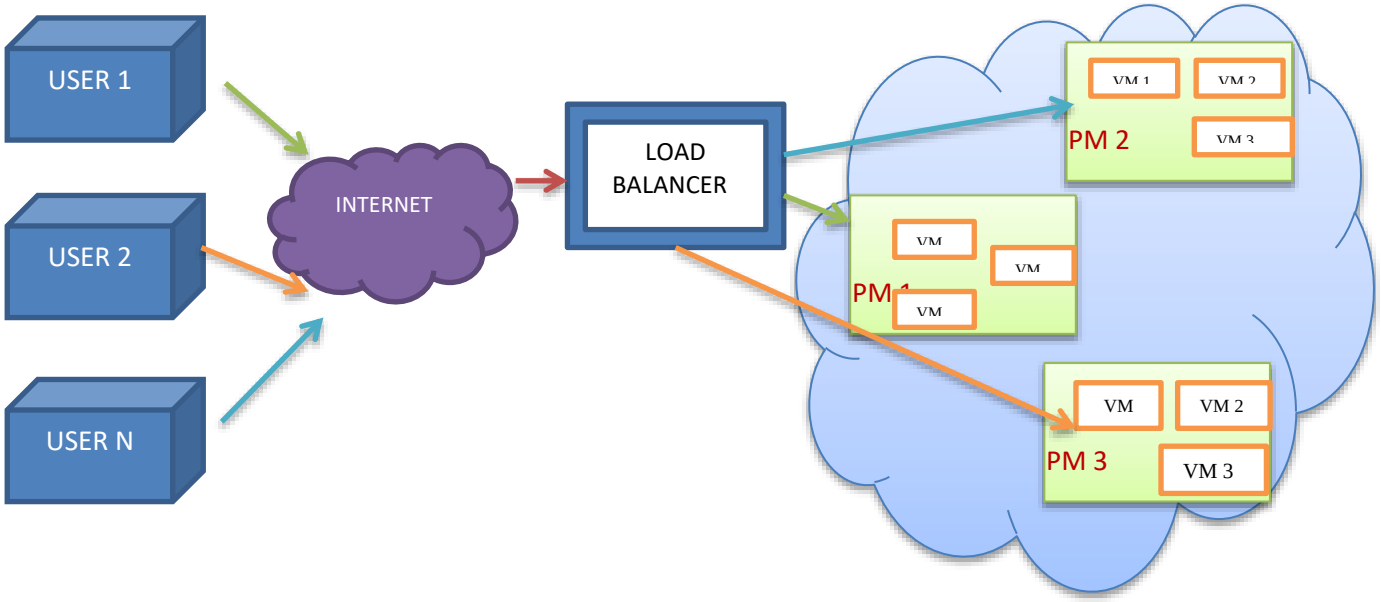


Figure.1 Load Balancing System

3. LOAD BALANCING SYSTEM

The proposed load balancing design model as shown in Figure 1. The key aim of the method is to allocate every function in compliance with the capability of VM to VM. The load balance algorithms aid to delete the activities from the filled Server and transfer them to the prepared Machine. The proposed system consisting of an astronomically large number of data centers also known as physical machines (PMs), and each PM has certain virtual machines (VMs) to perform the tasks of the user. Growing cloud user has a variety of different tasks to perform on the VM. Here the tasks are allocated to VM using a load balancing algorithm.

The proposed algorithm of load balancing continuously tests the load of all VMs in the cloud. The processing time of each task depends on the load of the VM. Although the execution time of one job is distinct from the other, the load of the VM is transferred.

Consider cloud C, which consists of 'i' number of the physical machine (P) and each 'P' consist of 'j' number of virtual machines (V).

Likewise, in the cloud, 'k' numbers of user are registered, and each user consists of 'l' number of tasks. The primary goal of this research is to reduce Task Execution Period (TET), Task Execution Expense (TEC), and allocate the workload to the virtual computer uniformly throughout the cloud. The Task Execution Time (TET) can be calculated as follows:

$$TET=\sum_{l=1}^{Number\ of\ task}(TET\ of\ V\ X\ Size\ of\ \ T) \qquad (1)$$

The Task Execution Cost (TEC) can be calculated as follows:

$$TEC=\sum_{l=1}^{Number\ of\ task}\ TET \qquad (2)$$

The load should be managed to get the correct scheduling procedure. The machine requires significant time and cost for carrying out the mission without a sufficient load.

4. BASIC GSO ALGORITHM (GSO)

The Glow-worm Swarm Optimization (GSO) is an initial swarm intelligence optimization algorithm that imitates glow worm's flashing behaviour. GSO is a swarm of glow-worm's that are randomly dispersed in the object function search space. As authority is not regulated at a single level, it is more scalable [18]. GSO algorithm was implemented [19] for complex allocation of VMs in cloud computing.

GSO is based on the behavior of glow-worm. A glow-worm that generates more light (high luciferin) means it is similar to the real location and has a high objective feature value. Each glow-worm selects a neighbor with a higher luciferin value than its own, according to a quantitative estimate. Such movements are focused entirely on local knowledge. It then splits the glow-worm into identity groups that discover further optimizations of the defined objective function. GSO algorithm begins by placing the glow-worm's randomly in the workspace, and all the glow-worm possesses the same amount of luciferin.

5. IMPROVED GSO BASED TASK SCHEDULING (IGSOTS) ALGORITHM

IGSOTS algorithm is used to check for VM and minimizes cost of execution. Each glow-worm represents a VM and the execution cost is value of VM luciferin. They also move into their neighbors with higher luciferin than their own, depending on the type of glow-worm. But in our algorithm, a VM is drawn to its neighbor with the lowest cost of execution, which is the opposite of the glow-worm's characteristics.

5.1 Initialization of control Parameters

There are a variety of glow-worm's which were regarded as candidate solutions to the problem in the initial population. The first population is randomly generated.

The most common parameters used in IGSOTS algorithm are given below

Glow-worm individual	n
Number of Dimension	m
Number of Decision variables	d
Step size	s
Number of iteration	iter_max
The initial value of luciferin	ℓ_0
The initial value of the radial range	r_0
luciferin decay constant	ρ (0 <ρ< 1)
luciferin enhancement fraction	γ

Table.1 Notations used in IGSOTS

ρ	γ	β	n_t	S	ℓ_0
0.4	0.6	0.08	5	0.03	5

Table. 2 Values of parameters fixed in all the simulations

β is a model constant, n_t is a constant parameter used to restrict the neighborhood set size. The quantity of $r_0$is made equal to r_s in all experiments. Initialize a set of glow-worm in the solution space. The number of glow-worm's is predefined in the initialization process of the algorithm parameters. The initial glow-worm set consists of a mixture of only the candidate solutions that satisfy all the constraints of the problem.

Glow-worm swarm S, which consists of m glow-worm's, is distributed in the objective function search space. Each Virtual machine VM_j (j=1…m) is assigned a random position PM_j inside the given function search space. Glow-worm VM_j carry its own luciferin level Lj, and has the vision range called local decision range rdj.

Evaluate all the glow-worm by the pre-defined problem objective functions. Every mixture of objective functions can be used concurrently to test the glow-worm as shown in the question formulation. All measured objective function values are now analysed at the same time.

5.2 Calculation of Luciferin value for all glow-worms

All the VM (glow-worm) carry an equal luciferin level (L_0). The rd radial sensor range r_s are initialized with the same value (r_0). The objective function at the existing glow-worm location (PM_j) is measured during the luciferin level update, and then the luciferin level for all VMs (glow-worm's) is used for the updated objective function values. The updated luciferin level L_j is calculated using the following equation:

$$\boldsymbol{L_j(t) = (1-\rho)L_j(t-1) + \gamma F\left(p_j(t)\right)} \quad \textbf{(3)}$$

5.3 Identify a set of possible neighborhood solutions

Each VM_j (glow-worm) explores its own neighborhood region to extract the neighbors of VM_j that have the lowest luciferin level by applying the following rule:

$$\boldsymbol{z \in N_j(t) \ \ iff \ \ d_{jz} < rd_j(t) \ and \ L_z(t) < L_j(t)} \quad \textbf{(4)}$$

Where d is the distance from one VM to another VM and z is one of the closer glow-worm's to glow-worm j which is having less number of task, $\boldsymbol{N_j(t)}$ is the neighborhood set, $\boldsymbol{d_{jz}}$ is the Euclidean distance between VM_j and VM_z, $\boldsymbol{rd_j(t)}$ is the local decision range for VM_j, and $\boldsymbol{L_z(t)}$ and $\boldsymbol{L_j(t)}$ are the luciferin levels for virtual machine VM_j and VM_z, respectively.

5.4 Calculation of the probability for each possible neighborhood solutions

To select the best neighbor from the neighborhood set, the probabilities j for all neighbors is calculated using the following equation

$$\boldsymbol{prob_{jz} = \frac{L_z(t) - L_j(t)}{\sum_{k \in N_j(t)} L_k(t) - L_j(t)}} \quad \textbf{(5)}$$

5.5 Selection of the best neighbor from the neighborhood set

Every glowworm selects the direction of movement using the roulette wheel method, with the reater likelihood of a glowworm being selected from the neighbourhood.

$$x_i(t+1) = x_i(t) + s * \left(\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right) + V \quad (6)$$

Where V represents number of Virtual Machine.

5.6 Calculation of the updated Glow-worm and Evaluation of the overall fitness value

Identify a unique neighborhood solution for each glowworm with the probability value calculated in step 5.3. Unique neighbourhood solution is selected based on the higher probability value of the corresponding glowworm among the each set of neighbourhood solutions.

5.7 Calculate the updated neighborhood range value

At the end of the GSO iteration, the local decision range $\boldsymbol{rd_j}$ is adjusted by the following equation

$$rd_j(t) = \min\{rs, max[0, rd_j(t-1) + \beta(nt - |N_j(t-1)|)]\} \quad (7)$$

Where $rd_j(t-1)$is the previous$\boldsymbol{rd_j}$,$\boldsymbol{r_s}$ is the radial sensor range constant, $\boldsymbol{\beta}$ is a model constant, nt is a constant parameter used to restrict the neighborhood set size, and $|N_j(t)|$ is the actual neighborhood set size. Update the algorithm iteration value and repeat the steps 5.2 to 5.6. After completing the predefined number of iterations, the highest ranked glowworm will be considered as the optimal solution for the given problem.

6. RESULT AND ANALYSIS

In this chapter the data gathered from the proposed IGSOTS load balancing technique are provided. Many tests are performed to validate the efficiency of the methodology introduced in this paper for load balancing inside the cloud environment. So we model the proposed study using Java with Cloud Sim software and a variety of tests have been conducted.

The principal focus of the proposed approach is to allocate every task to VM based on VM's potential. VM is allocated the role based on execution time, execution cost and load. To evaluate the efficiency of the proposed work we compare our proposed work with some other algorithm. Three specific settings for the experimental findings are used. Initially we assigned 5 physical machines, 15

virtual machines with 50 numbers of tasks. Next we assigned 15 physical machines, 45 virtual machines with 100 numbers of tasks. Finally we checked with 25 physical machines, 60 virtual machines with 125 numbers of tasks

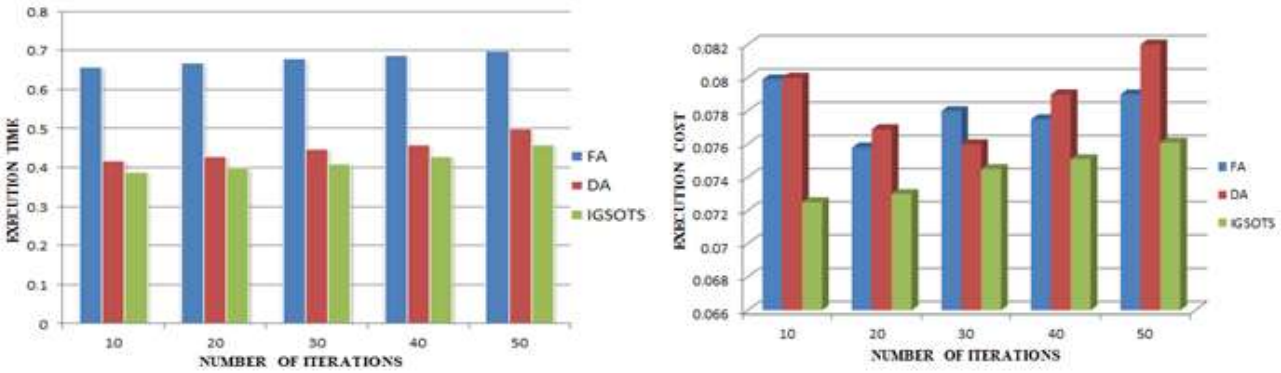


Fig. 2a Fig.2b

Fig. 2a&b Performance Analysis of Total Execution Time and Total Execution Cost

The effects of the proposed approach as well as of the existing FA and DA for the design on various iterations can be seen in Figures 2a and 2b. The performance analysis based on TET was shown in Figure 2a. For the system to be successful the execution time must be less. By Analysing Fig. 2b Compared to FA and DA our proposed approach achieves less time. Similarly, an overview of output based on TEC is given in Fig. 2b. The total cost of implementation of the proposed IGSOTS approach has lower cost of execution which makes it superior to other comparative approaches. Hence, it was obvious from the result that the proposed IGSOTS approach has outperformed FA and DA methods.

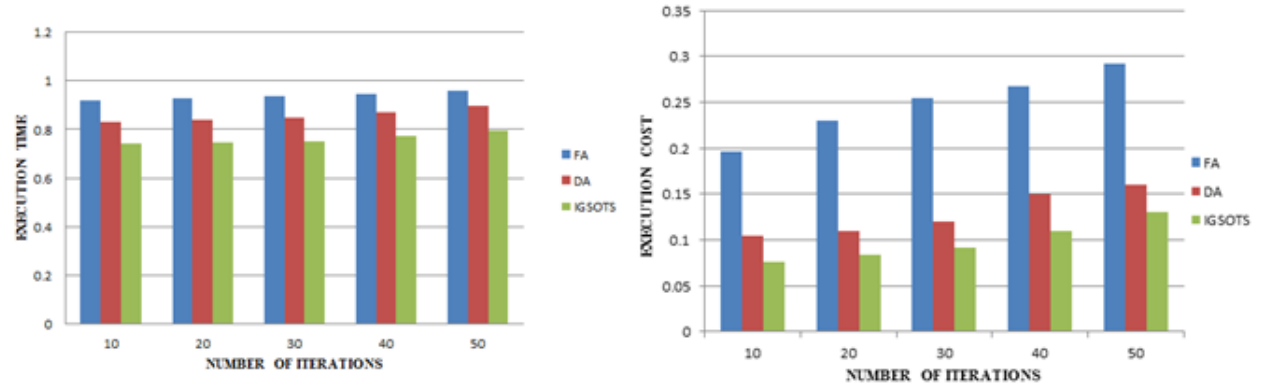
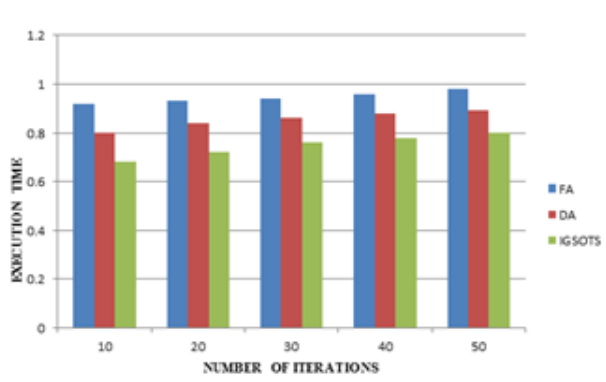


Fig. 3a Fig.3b

Fig. 3a&3b Performance Analysis of Total Execution Time and Total Execution Cost

Here, for experimental research, we used 100 tasks, 15 physical machines, and 45 virtual machines. We plan our tasks using this PM and VM. The performance analysis of IGSOTS and existing FA and DA configuring methodologies in various iterations are shown in Fig. 3a & b. The key aim is to prepare the function in VM on a load basis. The load is one of the most important programming parameters. Only the correct expected output is given by the correct balanced VM. Analysing Fig.3a, Compare with other approaches, our proposed method meets the minimum time to execute the process.



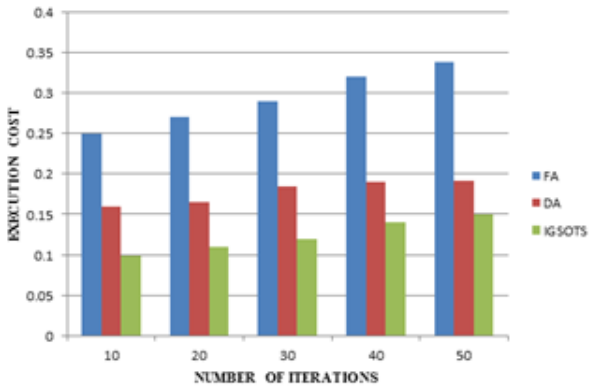


Fig. 4a **Fig.4b**

Fig. 4a&4b Performance Analysis of Total Execution Time and Total Execution Cost

Here in this section, we examine performance of the proposed approach using 25 physical machines and 60 virtual machines with 125 tasks used. Fig. 4a & b demonstrate the performance comparison of the proposed methodology and existing FA and DA methodologies for different iterations. The output analysis was shown in Fig. 4a based on TET. Likewise we have achieved the lowest TEC for this design in Fig. 4b. By the end, our proposed work clearly understands producing better outcomes relative to other works.

7 CONCLUSION

This paper describes an IGSOTS-based load balancing method and this mechanism was explored in simulated tests. The strategy method that we are implementing is structured to get a balanced load through virtual machines and to execute with a minimal time and expense. Therefore, IGSOTS offers a cloud-based system for streamlined task management and resource usage. The simulated results have shown that the proposed IGSOTS methodology is compared with the existing methodology (FA and DA) in terms of execution time and implementation costs.

REFERENCES

1. J. Koomey, Growth in data center electricity use 2005 to 2010, A report by Analytical Press, completed at the request of The New York Times (2011) 9.
2. C. Pettey, Gartner estimates ict industry accounts for 2 percent of global co2 emissions, Dostupnona: https://www. gartner.com/newsroom/id/503867 14 (2007) 2013.
3. Zhu, X., Yang, L.T., Chen, H., Wang, J., Yin, S., Liu, X.: Real-time tasks oriented energy-aware scheduling in virtualized clouds. TOCC 2(2), 168–180 (2014)
4. Z. G. Chen, Z. H. Zhan, H. H. Li, K. J. Du, J. H. Zhong, Y. W. Foo, Y. Li, and J. Zhang, “Deadline constrained cloud computing resources scheduling through an ant colony system approach,” in Proc. Int. Conf. Cloud Computing Research and Innovation, 2015, pp. 112-119.
5. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Struct. Multi. Optim. 26(6), 369–395 (2004).
6. Zhan S, Huo H. Improved PSO-based task scheduling algorithm in cloud computing. J Inform Comput Sci. 2012;9:3821-3829.
7. Shahapure, N.H., Jayarekha, P.: Load balancing with optimal cost scheduling algorithm. In: 2014 International Conference on Computation of Power and Energy, Information and Communication (ICCPEIC) 2014, IEEE (2014)
8. Zhao, S., Lu, X., Li, X.: Quality of service-based particle swarm optimization scheduling in cloud computing. In: Wong W. (eds.) Proceedings of the 4th International Conference on Computer Engineering and Networks. Lecture Notes in

Electrical Engineering, vol. 355. Springer, Cham (2015)

9. Liu, P., Zhu, Y.: Multi-dimensional constrained cloud computing task scheduling mechanism based on genetic algorithm. Int. J. Online Eng. (iJOE) 9(S6), 15–18 (2013)
10. Tang, Q., Li, Z., Zhang, L.: An effective discrete artificial bee colony algorithm with idle time reduction techniques for two sided assembly line balancing problem of type-II. Comput. Ind. Eng. 97, 146–156 (2016)
11. Ramezani, F., Lu, J., Hussain, F.K.: Task-based system load balancing in cloud computing using particle swarm optimization. Int. J. Parallel Program. 42(5), 739–754 (2014).
12. He, H., Xu, G., Pang, S., Zhao, Z.: AMTS: adaptive multi-objective task scheduling strategy in cloud computing. China Commun. 13(4), 162–171 (2016).
13. Zhang, P.Y., Zhou, M.C.: Dynamic cloud task scheduling based on a two-stage strategy. IEEE Trans. Autom. Sci. Eng. 15(2),772–783 (2018).
14. Chunlin, L., Min, Z., Youlong, L.: Efficient load-balancing aware cloud resource scheduling for mobile user. Comput. J. (2017). https://doi.org/10.1093/comjnl/bxx037.
15. Guo, M., Guan, Q., Ke, W.: Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload.IEEE Access 6, 15178–15191 (2018).
16. Niknam, S., Wang, P., Stefanov, T.: Resource optimization for real-time streaming applications using task replication. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37(11), 2755–2767 (2018).
17. P. Neelima, A. Rama Mohan Reddy, An efficient load balancing system using adaptive dragonfly algorithm in cloud computing, Cluster Computing, Springer, Feb 2020,
18. K. Krishnanand and D. Ghose. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. Multiagent and Grid Sys., 2(3):209-222, 2006.
19. D. Alboaneen, H. Tian_eld, and Y. Zhang. Glowworm swarm optimisation algorithm for virtual machine placement in cloud computing. In Int. IEEE Conf. on Ubiquitous Intelligence Comp., Advanced and Trusted Comp., Scalable Comp. and Communications, Cloud and Big Data Comp., Internet of People, and Smart World Congress,(Toulouse, 18-21 July), pages 808-814.IEEE, 2016.