TASK SCHEDULING WITH FIREFLY ALGORITHM IN CLOUD COMPUTING

Omid Jafarzadeh-Shirazi *1

1* Department of Computer Science & Engineering, Shiraz University, Shiraz, Iran.

*Corresponding author e-mail: ojafarzadehs@cse.shirazu.ac.ir,omid.jafarzadeh1988@gmail.com

ABSTRACT: *Task scheduling problem in cloud computing has become an active research topic due to the tremendous growth in the use of cloud computing. Cloud computing is a heterogeneous system and it holds and processes large amount of complex data. Scheduling tasks efficiently can lead to better performance and more throughputs in the system. In this paper, in order to minimize the cost of processing time, a firefly scheduling algorithm is proposed to schedule tasks in cloud computing environments. Firefly algorithm is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. We considered two costs for each task: communication and computation. A comparison is made between proposed algorithm and particle swarm optimization algorithm in task scheduling. Experimental results show that firefly task scheduling algorithm is more suitable to cloud computing environments.*

Keywords: Firefly algorithm, task scheduling, cloud computing, attractiveness function

1. INTRODUCTION

Cloud computing is the latest distributed computing paradigm that attracts increasing interests of researchers in the area of distributed and parallel computing [1], service oriented computing [2] and software engineering [3].

The new emergence of cloud computing technologies provides a method to deal with complex applications that manipulate large amounts of data and need high performance capabilities. Clouds are a type of parallel and distributed system, consisting of inter-connected and virtualized computers. These computers can be dynamically provisioned as per user's requirements [4]. Berkeley [5] defined cloud as "*both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services*". The software services provided to users have long been referred to as *Software as a Service* or *SaaS*. The datacenter hardware and software infrastructures are called a cloud. When a cloud is made available in a pay-as-you-go manner to the general public, it is called a public cloud.

One of the main challenges in distributed systems, especially cloud computing systems is task scheduling challenge. Task scheduling refers to assigning user requests to underlying resources effectively. Regarding the fact that the scheduling is known as a NP-hard problem, many heuristic solutions have been proposed so far to find a near-optimum solution for the scheduling problem.

In this paper, we propose a task scheduling solution based on firefly algorithm. Firefly algorithm is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. Metaheuristic is a heuristic designed to find, generate, select searching manner for finding a solution to an optimization problem. Firefly algorithm has many advantages such as:

- Its convergence rate is too high.
- Each firefly works individually and finds a better position for itself in consideration with its current position as well as the position of other fireflies. Thus, it escapes from the local optima and finds global optimum.
- It provides different levels of robustness in comparison to other metaheuristic algorithms

The remainder of this paper is organized as follows: Section 2 mentions a set of related works in the area of task scheduling in cloud. Section 3 discusses the proposed solution that is based on firefly algorithm. Section 4 presents the experimental results and finally section 5 concludes the paper.

2. RELATED WORKS

Assigning cloud resources to users' tasks is a key technical challenge that is called task scheduling. Different methods have been proposed to schedule tasks on a distributed system, efficiently. Scholars have proposed FCFS, greedy, genetic and many other algorithms [6-11] to solve this problem. T. Kosar *et al* [12] proposed a scheduler in the Grid that guarantees task scheduling activities can be queued, scheduled, monitored and managed in a fault tolerant manner. J. M. Cope *et al*, [13] proposed a task scheduling strategy for urgent computing environments to guarantee robustness of data. T. Xie [14] proposed an energy-aware strategy for task scheduling in RAID-structured storage systems.

Genetic Algorithm (GA) is another suitable approach that has been used in task scheduling problem in cloud computing. GAs can solve optimization problems by iterative evolutions over generations of solutions. Although GA is more time consuming than list heuristics, it is acceptable for applications with long runtime. In addition, the speed of GA can be accelerated by using parallel GA technologies [15].

R. Bossche *et al* [16] have proposed a genetic algorithm as the optimization method for a new scheduler that provides better makespan and better balanced load across all nodes in comparison to FIFO and delay scheduling. In 2010, an optimal scheduling policy based on linear programming to outsource deadline constraint workloads in a hybrid cloud scenario was proposed [17]. In 2011, S. Tayal [18] proposed an algorithm based on Fuzzy-GA optimization that evaluates the entire group of tasks in a job queue on basis of prediction of execution time of tasks assigned to certain processors and makes the scheduling decision. Wang *et al*, [19] extend the previous works and propose the Look-Ahead scheduling algorithm with Reliability-Driven (RD) reputation. To evaluate the reliability of resources, RD reputation considers the runtime of tasks by using the task failure rate (task failures per unit time) of resources to define the reputation. It also provides a real-time reputation that can be used to evaluate the reliability of each task directly using the exponential failure model. Based on the RD reputation, they propose Look-Ahead Genetic Algorithm (LAGA) to intelligently optimize both makespan and reliability for a workflow application.

Another approach to tackle task scheduling problem is Particle Swarm Optimization (PSO) algorithm. PSO is a self-adaptive global search based optimization technique introduced by Kennedy and Eberhart [22]. The algorithm is similar to other population-based algorithms like GAs but, there is no direct re-combination of individuals of the population.

Zhan, S. B *et al*, [23] proposed an improved PSO task scheduling algorithm. PSO algorithm was utilized by simulated annealing algorithm for faster global convergence. Guo *et al.* [24] formulate a model for task scheduling and also present a PSO algorithm based on small position value rule.

Salman *et al.* [25] have shown that the performance of PSO algorithms are faster than GA in solving static task assignment problems for homogeneous distributed computing systems based on their test cases. Lei et al. [26] have also shown that the PSO algorithm is able to get better schedule than GA based on their simulated experiments for Grid computing. In addition, the results presented by Tasgetiren *et al.* [27] have provided evidence that PSO algorithm was able to improve 57 out of 90 best known solutions provided by other well known algorithms to solve the sequencing problems.

Ant colony optimization (ACO) is another algorithm that has been used widely in cloud computing. Zhu et al. [28] studied resource scheduling problem and also presented an ant colony optimization based solution for the problem. Nishant *et al.* [29] informs readers about ACO algorithm capabilities and then proposes an ACO-based load balancing solution for cloud computing network.

3. Proposed Scheduling Solution

In this paper, we propose a firefly-based algorithm to schedule tasks in cloud computing environments. The rest of this section introduces the firefly algorithm, then introduces the proposed algorithm, and finally presents the evaluation results.

3.1. Firefly Algorithm

Firefly algorithm is a nature-inspired metaheuristic optimization algorithm. This algorithm has been implemented in chemistry [30], clustering [31], image processing [32] etc. Each firefly is considered as a solution and each one has a flash. Fireflies with less light intensity will be moved towards fireflies with more light intensities. This procedure will be continued till expected answer is reported.

The primary purpose for a firefly's flash is to act as a signal system to attract other fireflies. The brightness should be associated with the objective function. The objective function is the main function for evaluating solutions (fireflies). More brightness shows the solution is more appropriate. Xin-She Yang formulated this firefly algorithm by assuming that [33]:

- All fireflies are unisexual, so that one firefly will be attracted to all other fireflies
- Attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by (and thus move to) the brighter one; however, the brightness can decrease as their distance increases
- If there are no fireflies brighter than a given firefly, it will move randomly

Firefly algorithm starts with random solutions. Depending on the problem, each firefly is considered as an answer and contains an array with size of L. These solutions are called population. Afterwards, fireflies will be evaluated by an objective function. Fireflies that represent a better solution look brighter.

The pseudo code of firefly algorithm is described as shown in Algorithm 1.

```
1.  Generate an initial population of fireflies
2.  Formulate light intensity (I) so that it is associated with
    objective function (f)
3.  Define absorption coefficient γ
4.  While (t < MaxGeneration)
    4.1  For all fireflies i
         4.1.1  For all fireflies j
                4.1.1.1  If light intensity of firefly i is greater than
                         firefly j then,
                         • Move firefly j to firefly i
                         • Vary attractiveness with distance (r)
                         • Evaluate new solutions and update light
                           intensity
5   Rank fireflies and find the current best
```

Algorithm 1: The general steps of the firefly algorithm

In the pseudo code presented in Algorithm 1, *t* shows iteration in learning cycle and *MaxGeneration* is the maximum number that the algorithm is allowed to continue the learning phase.

3.2. Proposed Scheduling Algorithm

In this section, we will introduce firefly algorithm for task scheduling in cloud computing. For simplicity in describing our new firefly algorithm, we considered the following three idealized rules:

- All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex
- Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly.
- The brightness of a firefly is affected or determined by the landscape of the objective function. For a maximization problem, the brightness can simply be proportional to the value of the objective function.

We modeled the task scheduling problem as a graph where $T= \{T_1, T_2, ..., T_n\}$ represents the tasks of an application and $E=\{C_{ij}\}$ indicates the information exchange between tasks T_i and T_j. The edge weigh e_{ij} between node T_i and T_j denotes the information exchange between these pair of tasks. The node weigh w_i corresponds to the work capacity of the node. Figure 1 shows a sample task graph. A task's processing cost will be varied according to the task's assignment to different processors. In this paper, our goal is how to minimize the communication and the execution time.

Light intensity shows attractiveness of each firefly. According to the definition of the firefly algorithm, there should be an attraction between the fireflies i.e., the tasks. The attraction is based on the affinity possessed by node to the request. The attraction is controlled by the decision parameters defined by the proposed approach. The proposed approach defines decision parameters for each node and the parameters are considered as the attributes.

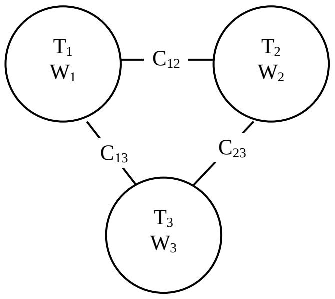


Figure 1.Task scheduling graph. C_{ij} is cost of information exchange between node i and j

The nodes attractiveness is measured by equation (1)

$$attr(n_i)_j = \frac{p_j}{cpu_i + disk_i + dlink_i} \quad (1)$$

where:

- *attr(n$_i$)$_j$* represents the attraction of task *j* to resource *i*
- *cpu$_i$* is CPU rate of node *i*
- *mem$_i$* is memory rate of node *i*
- *dlink$_i$* is transition rate of node *i*
- *p$_j$* represent processing time of task *j*

We store the parameters of each resource in a table like Table 1.

Table 1. Information about each resource

Resource	CPU Rate	Memory Rate
r_1	C_1	M_1
r_2	C_2	M_2
.	.	.
.	.	.
r_n	C_n	M_n

The process of selecting the node with least cost is inspired from the firefly algorithm in such way that, the least distinct firefly will possess similar characteristics. Inspired from the theory, we subject a distance calculation between the nodes in the scheduling queues. Before proceeding to the calculation, we find the node with least *attr(n$_i$)* value. The node with least *attr(n$_i$)* value is considered as the pivot point for the queue to calculate the least distinct nodes. The distance value of the node is calculated based on the Cartesian distance, which is given by equation (2).

$$Dist = \sqrt{\sum_{i=1}^{k}(n_i - n_j)^2} \quad (2)$$

The distance is presented using the expression distance in which n_i is the selected node and n_j is the comparing node. Once all the distance values have been calculated between the node values, the nodes are rearranged according to the least distinct node to the pivot node.

4. SIMULATIVE RESULTS

The proposed scheduling algorithm is based on firefly algorithm and its application is in optimizing task scheduling in cloud computing. The processing of the proposed approach is explained in Section 3, now, in this Section, we show the experimental analysis of the proposed scheduling algorithm by considering a simulated cloud network through Cloudsim [22] simulator and java programming. Cloudsim is a framework for modeling and simulation of cloud computing infrastructures and services.

In implementation of firefly task scheduling algorithm in Cloudsim, the following classes in Cloudsim were assisted:

- *Task* class: this class was developed for simulating the behavior of tasks in cloud computing
- *DatacenterBroker* class: this class represents a broker acting on behalf of a user. It hides VM management.
- *VM* class: this class represents a virtual machine. It runs inside a host, sharing host list with other VMs.
- *Datacenter* class: This class deals with handling of VMs

Figure (2) shows classes which were developed to implement the proposed algorithm.

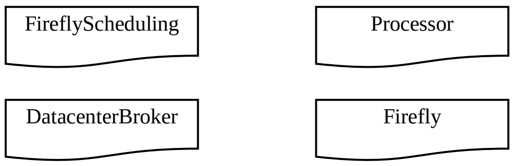


Figure2. Classes which were developed for implementation of proposed algorithm

FireflyScheduling class consists of functions and parameters that are related to task scheduling part.

In implementation of the algorithm, we considered resources with different processing and memory abilities. So, we developed Processor class which contains of different processing resources as tabulated in Table 2.

Table 2: Different processor types that are considered in the Processor class.

Ref.	Vendor	Processor Model
1.	Intel	Core_i7_Extreme_Edition_3960X
2.	Intel	Core_i7_Extreme_Edition_980X
3.	Intel	Core_2_Extreme_QX9770
4.	Intel	Core_2_Extreme_X6800
5.	Intel	Pentium_4_Extreme_Edition
6.	AMD	FX_8150_Eight_core
7.	AMD	Phenom_II_X6_1100T
8.	AMD	Athlon_FX_60_Dual_core
9.	AMD	Athlon_FX_57
10.	AMD	E_350_Dual_core

Datacenter Broker class was implemented for the purpose of encapsulation. This class is responsible for managing resources and assigning them to each task. Also this class controls scheduling algorithm as well.

Firefly class was dedicated for implementing behavior of fireflies in the proposed algorithm.

For evaluation of firefly algorithm, we used NASA Ames Research Center dataset. This dataset includes the features tabulated in Table 3.

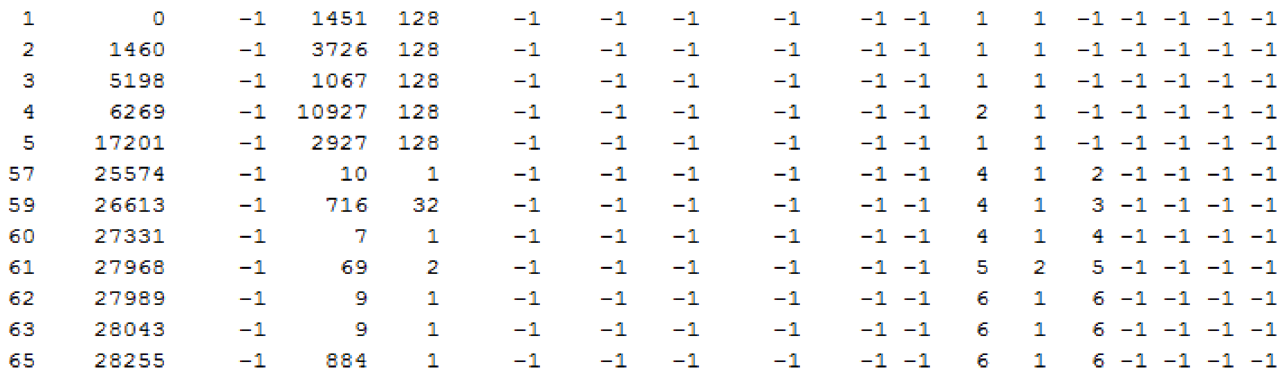
Table 3: Features of the NASA Ames Research Center dataset

Ref	Attribute	Value
1.	Record number	42264
2.	Preemption	No
3.	Start Time	Fri Oct 01 00:00:03 PDT 1993
4.	End Time	Fri Dec 31 23:03:45 PST 1993
5.	User Groups	Normal & System Personnel

Figure 3 below demonstrates the format of the dataset.
In experimental tests, we evaluated our proposed algorithm with different number of resources. We compared result of firefly algorithm with particle swarm optimization (PSO) task scheduler. Table 4 represents the results:

Table 4. Comparison between PSO and proposed algorithm

PSO (second)	Proposed algorithm (second)	Number of resources	Number of cloudlets
44.235	40.430	4	89
25.115	22.320	8	89
19.889	17.650	10	89



```
 1       0    -1   1451  128    -1    -1   -1     -1    -1 -1   1    1  -1 -1 -1 -1 -1
 2    1460    -1   3726  128    -1    -1   -1     -1    -1 -1   1    1  -1 -1 -1 -1 -1
 3    5198    -1   1067  128    -1    -1   -1     -1    -1 -1   1    1  -1 -1 -1 -1 -1
 4    6269    -1  10927  128    -1    -1   -1     -1    -1 -1   2    1  -1 -1 -1 -1 -1
 5   17201    -1   2927  128    -1    -1   -1     -1    -1 -1   1    1  -1 -1 -1 -1 -1
57   25574    -1     10    1    -1    -1   -1     -1    -1 -1   4    1   2 -1 -1 -1 -1
59   26613    -1    716   32    -1    -1   -1     -1    -1 -1   4    1   3 -1 -1 -1 -1
60   27331    -1      7    1    -1    -1   -1     -1    -1 -1   4    1   4 -1 -1 -1 -1
61   27968    -1     69    2    -1    -1   -1     -1    -1 -1   5    2   5 -1 -1 -1 -1
62   27989    -1      9    1    -1    -1   -1     -1    -1 -1   6    1   6 -1 -1 -1 -1
63   28043    -1      9    1    -1    -1   -1     -1    -1 -1   6    1   6 -1 -1 -1 -1
65   28255    -1    884    1    -1    -1   -1     -1    -1 -1   6    1   6 -1 -1 -1 -1
```

Figure 3. Format of dataset used in this work

5. CONCLUSION

Task 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 etc. Firefly algorithm is a nature-inspired metaheuristic optimization algorithm. This algorithm has been implemented in chemistry, clustering, image processing. In this paper, we proposed a firefly task scheduling algorithm in cloud computing. We implemented proposed algorithm in Cloudsim simulator which is developed with Java programming language. Also a comparison between proposed algorithm and particle swarm optimization algorithm was reported in the paper as well.

REFERENCES

[1] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, “Cloud control with distributed rate limiting,” *ACM SIGCOMM Computer Communication Review*, **37**(4), p. 337, (2007).

[2] D. Ardagna and B. Pernici, “Adaptive Service Composition in Flexible Processes,” *IEEE Transactions on Software Engineering*, **33**(6), pp. 369–384, (2007).

[3] K. Bhattacharya, M. Bichler, and S. Tai, “ICSE Cloud 09: First international workshop on software engineering challenges for Cloud Computing,” *31st International Conference on Software Engineering - Companion Volume*, (2009).

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” *Future Generation Computer Systems*, **25**(6), pp. 599–616, (2009).

[5] E. Deelman and A. Chervenak, “Data Management Challenges of Data-Intensive Scientific Workflows,” *Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)* (2008).

[6] B. Xu, C. Zhao, E. Hu, B. Hu, "Job scheduling algorithm based on Berger model In cloud environment", *Advances in Engineering Software*, **42**(7), pp.419-425, (2011).

[7] W. Wang, G. Zeng, D. Tang, J. Yao, "Cloud-DLS: Dynamic trusted scheduling for Cloud computing", Expert Systems with Applications, **39**(3), pp. 2321-2329, (2012).

[8] D. Cenk Erdil, "Simulating peer-to-peer cloud resource scheduling", *Peer-to-Peer networking and Applications*, **5**(3), pp.219-230, (2012).

[9] B. Li, A Meina Song, Junde Song, "A Distributed QoS-Constraint Task Scheduling Scheme in Cloud Computing Environment: Model and Algorithm", *Advances in Information Sciences and Service Sciences*, **4**(5), pp. 283-291, (2012).

[10] L. Deboosere, Bert Vankeirsbilck, Pieter Simoens, Filip De Turck, Bart Dhoedt, Piet Demeester, "Efficient resource management for virtual desktop cloud computing", *The Journal of Supercomputing*, **62**(2), pp. 741-767, (2012).

[11] S. Abrishami, Mahmoud Naghibzadeh, "Deadline-constrained workflow scheduling in software as a service Cloud", *Scientia Iranica*, **19**(3), pp.680-689, (2012).

[12] T. Kosar and M. Livny, “Stork: making data placement a first class citizen in the grid,” *24th International Conference on Distributed Computing Systems, Proceedings*. (2004).

[13] J. M. Cope, N. Trebon, H. M. Tufo, and P. Beckman, “Robust data placement in urgent computing environments,” *IEEE International Symposium on Parallel & Distributed Processing*, (2009).

[14] T. Xie, “SEA: A Striping-Based Energy-Aware Strategy for Data Placement in RAID-Structured Storage Systems,” *IEEE Transactions on Computers*, **57**(6), pp. 748–761, (2008).

[15] D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, and B.-S. Lee, “Efficient Hierarchical Parallel Genetic Algorithms using Grid computing,” *Future*

Generation Computer Systems, **23**(4), pp. 658–670, (2007).

[16] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads," *IEEE 3rd International Conference on Cloud Computing*, (2010).

[17] Tayal, S."Tasks Scheduling Optimization for the Cloud Computing Systems‖". In: (IJAEST) *International Journal of Advanced Engineering Sciences and Technologies*, **5**(2), pp. 111-115. (2011).

[18] Y. Ge and G. Wei, "GA-Based Task Scheduler for the Cloud Computing Systems,", *International Conference on Web Information Systems and Mining* (2010).

[19] X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, **27**(8), pp. 1124–1134, (2011).

[20] X. Wang, R. Buyya, J. Su, Reliability-oriented genetic algorithm for workflow applications using max-min strategy, in: 9th IEEE International Symposium on Cluster Computing and theGrid (CCGrid 2009), *IEEE Computer Society: Los Alamitos, CA, USA, Shanghai, China*,(2009).

[21] X. Wang, C. S. Yeo, R. Buyya, J. Su, "Reliability-driven repu-tation based scheduling for public-resource computing using ga" IEEE 23rd International Conference on Advanced Informa-tion Networking and Applications (AINA 2009*), IEEE Com-puter Society: Los Alamitos, CA, USA, Bradford, UK*, (2009).

[22] Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, (1995).

[23] Zhan, S. B., and H. Y. Huo. "Improved PSO-based task scheduling algorithm in cloud computing." *Journal of Information & Computational Science* **9**(13) 3821-3829.(2012).

[24] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm," *JNW*, **7**(3), (2012).

[25] Salman. "Particle swarm optimization for task assignment problem". *Microprocessors and Microsystems*, **26**(8):363–371, (2002).

[26] L. Zhang, Y. Chen, R. Sun, S. Jing, and B. Yang." A task scheduling algorithm based on pso for grid computing". *International Journal of Computational Intelligence Research*, **4**(1), (2008).

[27] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyil-maz. "Aparticle swarm optimization algorithm for make span and total flow time minimization in the permutation flow shop sequencing problem". *European Journal of Operational Re-search*, **177**(3):1930–1947, (2007).

[28] Zhu, Linan, Qingshui Li, and Lingna He. "Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm." *IJCSI International Journal of Computer Science Issues* **9**(5). (2012).

[29] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, Nitin, and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization," *UKSim 14th International Conference on Computer Modelling and Simulation*,(2012).

[30] S.-E. K. Fateen, A. Bonilla-Petriciolet, and G. P. Rangaiah, "Evaluation of Covariance Matrix Adaptation Evolution Strategy, Shuffled Complex Evolution and Firefly Algorithms for phase stability, phase equilibrium and chemical equilibrium problems," *Chemical Engineering Research and Design*, **90**(12), pp. 2051–2071, (2012).

[31] J. Senthilnath, S. N. Omkar, and V. Mani, "Clustering using firefly algorithm: Performance study," Swarm and Evolutionary Computation, **1**(3), pp. 164–171. (2011).

[32] T. Hassanzadeh, H. Vojodi, and F. Mahmoudi, "Non-linear Grayscale Image Enhancement Based on Firefly Algorithm," *Swarm, Evolutionary, and Memetic Computing*, pp. 174–181, (2011).

[33] Y., Xin-She. *"Firefly algorithm."* Engineering Optimization 221-230. (2010).