ADAPTIVE PARTICLE SWARM OPTIMIZATION SCHEDULING MODEL FOR JOBS ON CLOUD

Alaba T. Owoseni [1], Olatunde Iyaniwura [1], Lukman Adebayo Ogundele [2]

[1] Department of Mathematical Sciences, Kings University, Odeomu, Nigeria
[2] Department of Computer Science, Osun State College of Education, Ilesa, Nigeria

Corresponding Author: Alaba T. Owoseni, at.owoseni@kingsuniversity.edu.ng

ABSTRACT: Cloud computing is a special network-based computation that makes computational services available to customers on demand. In this computational paradigm, resources are managed by organizations that have all it takes to purchase and manage them since they are cost intensive and as such, the needy of computer-based services pay the organizations as virtualized resources are used to provide services to the needy. On cloud, effective and efficient computation depends on the technique used in scheduling tasks among computational resources especially the virtual machines and this has called for many researches from various quarters. In this paper, we propose an improved particle swarm optimization algorithm whose parameters (cognitive and social coefficients and inertial weight) are adaptive as a measure against premature convergence of the algorithm. CloudSim simulator was used to simulate the operation of the algorithm and the performance was compared with classic particle swarm optimization method for better result.

KEYWORDS: Adaptive particle swarm optimization, inertial weight, cloud computing, cognitive coefficient, social coefficient, makespan.

1. INTRODUCTION

Cloud computing is a technical distributed computing paradigm that occurs over computer network(s) in which computational resources at data centers are used to provide services to customers on demand and at a cost. It is also a term used to describe the on demand, elastic, and scalable services offered over a computer network [Hus14] usually on the internet as a global computer network. The rationale behind this computing could be attributed to the high cost of management for high-level computational resources for all users who at the same time yearn for high taste of services from these resources. Therefore, organization that has the strength of managing the resources do that and provide the high taste services to users at pay-per-use deal. There are various forms of cloud. These are public cloud (Amazon, and Microsoft provided clouds on the internet), private cloud and a hybrid of the afore-mentioned forms [Hus14]. The public cloud resides over a public computer network while the private is on a private network.

On a cloud, many tasks are been processed by virtualized resources at high tastes as demanded by users. These tastes include, minimum execution time, high data transfer rate, low energy consumption [J+13], maximum throughput, and high reliability. With these demands, it is a big task for cloud providers to ensure that appropriate and effective algorithms are used to ensure allocation of tasks to resources. The need for effectiveness and efficiency on the cloud has been a major concern to many researchers and various attempts have been made in devising both static and dynamic task-resource allocation algorithms. The static algorithms are based on prior knowledge of resources' computational power before scheduling while the dynamic algorithms are independent of such knowledge.

Proven facts have shown that the dynamic algorithms are better than the static ones and they include the biologically inspired algorithms [DA15]. Among the biologically inspired algorithms are particle swarm optimization (PSO), artificial bee colony, ant colony, cuckoo, bat algorithm and genetic algorithm. The genetic algorithm is similar to particle swarm optimization algorithm both in its design and operation as both are stochastic, population and evolutionary based. However, benefits like implementation simplicity, less computational power (since computational cost for mutation and crossover are exclusive) [FY17] have all made this algorithm preferable to genetic algorithm. On the other hand, premature convergence as attributed to PSO has been a setback for its use and also a challenge to its effectiveness. In this paper, an improved PSO whose parameters are optimized so as to solve the problem of premature convergence is proposed in scheduling tasks among virtual machines that reside on the cloud. The study is concerned with scheduling objective of reducing the makespan of the tasks presented at a particular time to the resources for execution. Other objectives like load balancing, data transfer and bandwidth are possible features of cloud that we recommend for upcoming researchers to enhance.

2. RELATED WORK

From the advent of cloud computing, various attempts have been made by scientists to provide algorithms that would help optimize resources on the cloud. In this section of the paper, a trend of these works is categorically reviewed.

2.1. Heuristic and Non-biologically Inspired Algorithms

First come first served and round robin algorithms are two conventional algorithms used in scheduling tasks among resources. The former allocates tasks based on their arrival time while the later uses time slice among resources to manage tasks allocation.
Min-min algorithm that selects the task with the least computational need and allocates it to the resource with the least completion time was considered in [KG11]. This algorithm targets small jobs to be accomplished firstly, which in turn increases the risk of starvation for big tasks or delay of their execution [RVR12]. This algorithm favors small tasks and disfavor big tasks.
In other to achieve a better computing throughput in terms of makespan and load balancing in a cloud environment, a new version of min-min algorithm (ILBMM) was proposed [PP16]. The theoretical analysis of the algorithm showed that under all possible situations, the ILBMM was capable of decreasing completion time of tasks, improving load balance of resources and gained better overall performance than Min-min algorithm.
Max-min algorithm that corrects the shortcomings of the Min-min algorithm was revised in [SM14]. In this study, the algorithm firstly allocates the highest computation-power- demanding task to resources with the least completion time. That is highest task goes to the best resource and this gives priority to bigger tasks and not the smaller tasks. Although this algorithm as believed could perform better than the Min-min or any improved version of Min-min but the priority for big tasks could also make small tasks to be postponed indefinitely [SM14].
Various works on the improvement of max-min algorithm as needs to get better task-resource allocation evolved. One of them is resource-aware scheduling algorithm (RASA) that uses the min-min strategy to execute small tasks before large ones and applies the max-min strategy to avoid delays in the execution of the big tasks and to support concurrency in the execution of large and small tasks [SR09].
Also, a unique modification to the improved max-min algorithm was proposed in [SM14]. In the study, two algorithms were proposed on improved max-min where instead of selecting the largest task; a task just greater than average execution time is selected and assigned to the resource which gives minimum completion time. The experimental results in the study showed that the new algorithms scheduled jobs with lower makespan. In the algorithms, the first uses arithmetic mean for dependent tasks while independent tasks are allocated using the second algorithm that uses geometric mean for computing average task.
An enhanced Max-min algorithm was proposed in [UP13]. The motivation for this study was as a result of large makespan that might result from a worst case of improved Max-min where largest task is allocated to the least resource and the degree of the largest task is high as it cost the least resource much time. Here, the enhanced algorithm is based on comprehensive study of the impact of improved Max-min task scheduling algorithm in cloud computing. It assigns task with average execution time (average or nearest greater than average task) to resource that produces minimum completion time.
[SA15] analyzed various scheduling algorithms that efficiently schedule computational tasks on cloud. Their result showed that Max-min algorithm in space-shared mode is more efficient than other scenarios (that is, Max-min in timeshared mode, Min-Min algorithm in both space-shared and timeshared modes). Also, in the study, it was proven that Min-min algorithm provides minimum delays in processing of tasks while Max-min provides proper utilization of the virtual machine and load balancing.
A hybrid of modified Max-min and classic Min-min algorithms were used in [RP14]. In the study, modified Max-min was based on the execution time and not on completion time as a selection basis. The result of the study could produce an algorithm that balances load with lower makespan than what RASA and classic Max-min would have produced.
A V-heuristic scheduling algorithm in service-oriented resource broker for improving computation applications performance was presented in [Y+13]. In the scheduling algorithm, three different heuristic scheduling algorithms (MCT- minimum completion time, Min-Min and Max-Min) were used to perform meta-job execution. Meta-job total processing time, average resource utilization rate and load balancing produced by their proposed scheduling algorithm were compared against that produced by classic heuristic scheduling algorithms with physical network supported. The simulation results obtained from the simulator showed that performance metrics of computation applications could be improved significantly when using V-heuristic scheduling algorithm.
In all researches on non-biologically inspired algorithms, static scheduling has always been the other of the day as it demands that prior knowledge

of the resources and tasks are needed and used for scheduling. However, dynamic algorithms are better in operation as they optimize effectively.

2.2. Meta-Heuristic and Biologically Inspired Algorithms

Among the meta-heuristic algorithms ever used in scheduling tasks on cloud or grid is the ant colony algorithm (ACO) [MSN10] [U+12] [T+13] [AM11]. This algorithm implements the foraging behavior of ants as they explore and exploit the optimal path to source of food. Using this algorithm, the virtual machines are compared to the nodes, ants are compared to software agents that traverse among virtual machines to locate tasks for them, and resources are similar to the food sources. Ant traverses nodes to allocate the jobs to various resources in an optimized mode. Results in the works showed that they are effective in providing an optimal schedule of tasks on cloud. However, when virtual machines are more even on optimized ant colony algorithm, colonies may be developed which may make an ant to follow pheromone of another colony [BDB16].

Artificial bee colony (ABC) algorithm that implements intelligent behavior of honey bees swarm during foraging was proposed in [FZ16] [KK14]. In the earlier work, the scheduler was designed to achieve a well-balanced load across virtual machines for maximizing the throughput and deliver minimum makespan. Experimental results as simulated using CloudSim showed its effectiveness to optimize load balancing and task scheduling compared with both of first come first served (FCFS) and ACO.

Also, an artificial bee colony algorithm was considered in [KG14]. In the study, scheduling objectives considered were quality of service (QoS), minimum total task finishing time, mean task finishing time and load balancing time and results showed its effective performance. However, out of all probabilistic meta-heuristic algorithms commonly used for optimization on cloud, ABC shows better scheduling when the objective mainly relies on and not on computational power costs [BDB16].

A genetic scheduling algorithm that schedules at every task scheduling cycle was devised in [S+12]. The algorithmic function iterates genetic operations to make an optimal task schedule. Experimental results showed effectiveness and efficiency of the genetic algorithm-based task scheduling model in comparison with existing task scheduling models, like round-robin task scheduling model, the load index-based task scheduling model, and the artificial bee colony-based task scheduling model.

A new scheduling concept was proposed in [1 PA12], where Min-min and Max-min algorithms are combined with genetic algorithm to improve its performance. Also, a modified genetic algorithm for single user jobs in which the fitness is developed to encourage the formation of solutions to achieve the time minimization and compared it with existing heuristics was proposed [SA12]. Experimental results showed that, under the heavy loads, the proposed algorithm exhibited a good performance.

Particle swarm optimization [KE95] is another prominent algorithm used in scheduling resources on the cloud [M+13]. It implements the natural schooling behavior of fish and schooling behavior of birds. Reasons for its prominence and preference to genetic algorithm include, faster convergence, its simplicity to implement, reduced cost of execution coupled with great performance. However, its shortcoming is its premature convergence. Many attempts have been made to remove this bottleneck in the operation of the PSO by optimizing its parameters.

One of the improvements on PSO was a study where a hybrid of particle swarm optimization and simulated annealing was implemented [MPG14]. In the study, simulated annealing was used to slow down the convergence of the swarm and to increase the swarm's probability of reaching the global optima.

Due to problems in PSO like premature convergence, an improved algorithm PSO using vector differential operator in differential evolution (DE) algorithm to prevent premature convergence was proposed [GK16]. The improved algorithm could improve the utilization of resources while minimizing the makespan of the given tasks. In [PM15], a hybrid algorithm of PSO and Min-min algorithm for solving workflow scheduling problem in cloud environment focusing on makespan was also executed.

3. RESEARCH METHODOLOGY

This section discusses the methods and materials used in the paper. The materials include both hardware and software tools as shortly discussed.

3.1. Materials

CloudSim 3.0.3 (a java-based simulator) was used for the simulation of the algorithm under discourse. The choice of the simulator was due to its popularity and effectiveness in achieving a real-world-like cloud environment. The simulator was executed on a personal computer (PC) with specification: Intel (R) Pentium(R) M processor with 2.16GHz, 4.0GB of random-access memory, 64-bit Operating System (Windows 10 Pro), and 80GB of free hard disk space.

3.2. Methods

Here, various techniques used in the study are discussed shortly:

a. *Particle Encoding*
The first task in using particle swarm optimization is particle encoding. It has to do with mapping the particle position to the problem solution. Here, a particle position represents a vector of integers with dimension "N", where "N" is the number of tasks to schedule at a particular point in time and the values (represent the resources that tasks are assigned to) of the integers range from "1" to "M". Where "M" is the total number of resources among which the tasks are to be scheduled. Table 2, shows a particular representation for six tasks and three resources (virtual machines).

Table 1: A Task-Resource Allocation

Task	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**
Resource	1	1	2	3	2	3

b. *Swarm Generation and Initialization*
Swarm is generated from particles that are randomly generated using the above particle encoding method. Here, each particle is a candidate solution of the problem and the goal is to iteratively derive the best fitness function which turns to the best particle. Also, initialization of swarm in PSO is a vital phase. In this paper, initial positions and velocities of particles are generated randomly. The values for the positions and velocities range between one and zero.

c. *Proposed APSO algorithm*
In this paper, the steps in classic algorithm for PSO and the methods for updating position and velocity are still used but, the inertia weight and acceleration coefficients generation are not randomly generated as usual but they are time varying. The steps are:
Step 1: Set particle dimension as equal to the size of ready tasks in {Ti}∈T;
Step 2: Initialize position vector and velocity vector randomly;
Step 3: For each particle, calculate its fitness value as in equation (3);
Step 4: If the fitness value is better than the previous pbest (particle best position), set the current fitness value as the new pbest;
Step 5: Select the best particle from all of the particles as gbest (particle global best position);
Step 6: For all particles, calculate velocity using equation (4) and update their positions using equation (5);
Step 7: If reaching to the maximum iteration or getting the ideal result stops, otherwise, go to Step 3.

d. *Fitness Function*
The fitness function measures the degree of optimization that a particle posses. Here, the particle with the least fitness function is the best particle. The expected execution time (EETT) for a task T is mathematically represented as:

$$EET_T = \frac{l}{c} \qquad (1)$$

where, l is the length of Task T, and C is the computational strength of any resource measured in millions instructions per second.
The makespan for a set of tasks assigned to a resource (k) can be mathematically represented as:

$$Makespan_T = \sum_{i=1}^{n} \left(EE_{T_i} + E_{T_i}\right) \qquad (2)$$

where, n is the total number of tasks in the set of tasks to allocate and ETi is the waiting time for task Ti before it was executed by resource k.
The objective fitness function is represented as:

$$f = min_{makespan} \qquad (3)$$

$$v_i^{k+1} = \omega v_i^k + c_1 rand_1 * \left(pbest_i - x_i^k\right) + c_2 rand_2 * \left(gbest_i - x_i^k\right) \qquad (4)$$

$$x_i^{k+1} = x_i^k + v_i^k \qquad (5)$$

In this paper, the position denotes the objective fitness function.

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min}) \frac{g}{G} \qquad (6)$$

The inertia weight ω in equation (6), as introduced by Eberhart and Shi [ES20], where it linearly decreases with the iterative generations and g is the generation index representing the current number of evolutionary generations, while G is a predefined maximum number of generations. Here, the maximal and minimal weights ωmax and ωmin are set to 0.9 and 0.4, respectively [Z+09] [SE98] [SE99].
The adaptive acceleration coefficients are adapted from [ZLL15]. They are mathematically represented below:

$$c1 = c1_{ini} - AcFator * c_{Acc} - AsFactor * c_{Ass} \qquad (7)$$

$$c2 = c2_{ini} - AcFator * c_{Acc} - AsFactor * c_{Ass} \qquad (8)$$

where,
$c1_{ini}$ and $c2_{ini}$ are constants with values 2.5, and 1.5 respectively.
c_{Acc} and c_{Ass} range from 0.4 to 0.6,
AcFator and AsFactor are mathematically represented as (9) and (10):

$$AcFator = \frac{F(gbest_T)}{F(gbest_{T-1})} \quad (9)$$

$$AsFator = \frac{F(gbest_T)}{avgF(X_T)} \quad (10)$$

where,
$F(gbest_T)$ is the best fitness of current iteration,
$F(gbest_{T-1})$ is the best fitness of previous iteration,
$avgF(X_T)$ is the average fitness of current iteration.

4. EXPERIMENT AND RESULTS

The adaptive particle swarm optimization algorithm was simulated on a CloudSim simulator with a Dell PC with specification: Intel (R) Pentium(R) M processor with 2.16GHz, 4.0GB of random-access memory, 64-bit Operating System (Windows 10 Pro), and 80GB of free hard disk space.
During the simulation, ten swarm sizes N=10, 20, 30, 40, 50, 60, 70, 80, 90, 100 were used to comparatively evaluate the performances of classical PSO and adaptive PSO. A swarm size of 60 was the best case in the two instances, so it was used in all subsequent tests. The algorithm parameters for classic PSO like: c1 and c2 are tested with values ranging from [0.1, 0.7] in increments of 0.2, and the inertial weight w, during the iterations it reduced from wmax to wmin.
Parameter wmin values of 0.1, 0.3, and 0.5 and wmax was set to 0.5, 0.7, and 0.9. Settings of c1 = 0.7, c2 =0.1, wmin = 0.3, and wmax = 0.7 worked best. In the case of adaptive PSO, parameters were generated as expressed in equations (7) and (8).
The total numbers of cloudlets were set to value ranging from 10 to 100, the processing times of cloudlets are uniformly distributed in [5, 10] and the memory requirement in [50, 100]. The numbers of Data centers are from 5 to 20, the total available memory is also uniformly distributed in [250, 500]. Simulation results demonstrate that more iterations or number of particles obtain the better solution since more solutions were generated as presented on Table 2 and Table 3.

5. CONCLUSION

This study has proposed an improved particle swarm optimization method in scheduling tasks on cloud for better makespan. The improvement was introduced through the optimization of the parameters of the algorithm that is, inertial weight, social coefficient and cognitive coefficient so as to prevent premature convergence problem that is found in PSO and at the same time improve on the makespan. Unlike the randomly generated parameters, time varying parameters have been proposed in this study and the algorithm was simulated using CloudSim as a simulator.

Table 2: Simulation Results with 10 Particles

Cloudlets & Data Centers	Number of Iterations					
	100		300		500	
	PSO	APSO	PSO	APSO	PSO	APSO
10,5	1260	1258	1254	1251	1203	1200
20,5	2108	2105	2101	2100	1994	1991
30,10	2913	2900	2910	2870	2900	2876
40,10	4391	4311	4279	4267	4267	4234
50,15	5742	5741	5160	5156	5151	5145
60,15	6853	6845	6851	6834	6848	6576
70,20	8159	8156	8156	8150	8141	8000
80,20	8911	8900	8909	8900	8900	8564
90, 20	9959	9930	9963	9945	9943	9854
100,20	10909	10870	10866	10856	10863	10651

Table 3: Simulation Results with 20 Particles

Cloudlets & Data Centers	Number of Iterations					
	100		300		500	
	PSO	APSO	PSO	APSO	PSO	APSO
10,5	1260	1250	1241	1240	1203	1200
20,5	2101	2101	2100	2099	1994	1991
30,10	2914	2900	2907	2807	2903	2900
40,10	4389	4379	4266	4256	4273	4263
50,15	5739	5729	5152	5150	5156	5146
60,15	6830	6830	6820	6819	6819	6809
70,20	8161	8151	8158	8148	8139	8119
80,20	8907	8903	8907	8907	8907	8901
90, 20	9960	9950	9931	9931	9949	9942
100,20	10908	10910	10866	10856	10873	10870

Table 4: Simulation Results with 30 Particles

Cloudlets & Data Centers	Number of Iterations					
	100		300		500	
	PSO	APSO	PSO	APSO	PSO	APSO
10,5	1259	1249	1241	1240	1203	1200
20,5	2100	2100	2100	20998	1994	1991
30,10	29142	2900	2907	2807	2903	2900
40,10	4387	4379	4266	4257	4273	4263
50,15	5736	5729	5152	5150	5156	5146
60,15	6830	6827	6820	6819	6819	6809
70,20	8159	8150	8158	8149	8139	8119
80,20	8907	8901	8907	8907	8907	8901
90, 20	9958	9940	9931	9929	9949	9942
100,20	10906	10910	10866	108565	10873	10870

REFERENCES

[AM11] **S. Aranganatham, K. M. Mehta** - *An ACO Algorithm for Scheduling data intensive application with various QOS requirements*, International journal of computer Applications (0973-8887) Volume 27, No. 10, pp 1-5, 2011.

[BDB16] **A. A. Buhussain, R. E. De Grande, A. Boukerche** - *Performance Analysis of Bio-Inspired Scheduling Algorithms for Cloud Environments*, Proceeding of IEEE International Parallel and

Distributed Processing Symposium Workshops IPDPSW, 2016.

[DA15] **J. Delsy, S. Anoop** - *Bio-Inspired Scheduling of High-Performance Computing Applications in Cloud: A Review*, International Journal of Computer Science and Information Technologies (IJCSIT), Volume 6 (1), pp. 485-487, 2015.

[ES20] **R. Eberhart, Y. Shi** - *Comparing inertia weights and constriction factors in particle swarm optimization*, Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), vol. 1, no. 7, pp. 84, 2000.

[FY17] **N. Frederic, Y. Yang** - *Analysis of Particle Swarm Optimization and Genetic Algorithm based on Task Scheduling in Cloud Computing Environment*, International Journal of Advanced Computer Science and Applications, Volume 8, No 1, pp. 19-25, 2017.

[FZ16] **R. Fatemeh, K. Zamanifar** - *Task Scheduling Based on Load Balancing Using Artificial Bee Colony in Cloud Computing Environment*, International Journal of Advanced Biotechnology and Research (IJBR), Vol 7, Special Issue-Number 5, pp. 1058-1069, 2016.

[GK16] **B. Gomathi, K. Karthikeyan** - *A Task Scheduling Based on Hybrid Self Organizing Migrating Algorithm in Cloud Environment*, Asian Journal of Information Technology, volume 15, issue 19, pp. 3703-3707, 2016.

[Hus14] **S. A. Hussein** - *Optimizing Cloudlet Scheduling and Wireless Sensor Localization using Computational Intelligence Techniques*, Thesis, The University of Toledo, 2014.

[J+13] **H. Jing, W. Kai, K. L. Lok, M. Seungbeom, M. Melody** - *A Tunable Workflow Scheduling Algorithm Based on Particle Swarm Optimization for Cloud Computing*, The International Journal of Soft Computing and Software Engineering (JSCSE), Volume 3, No. 3, pp. 351-358, 2013.

[KE95] **J. Kennedy, R. C. Eberhard** - *Particle swarm optimization*, Proceedings of IEEE International Conference on Neural Networks, pp.1942-1948, Piscataway, NJ, USA, 1995.

[KG11] **T. Kokilavani, D. Ga** - *Load Balanced Min-min Algorithm for Static Meta-Task Scheduling in Grid Computing*, International Journal of Computer Applications, Number 2, Article 7, 2011.

[KG14] **R. S. Kumar, S. Gunasekaran** - *Improving Task Scheduling in Large Scale Cloud Computing Environment using Artificial Bee Colony Algorithm*, International Journal of Computer Applications, Volume 103, No.5, pp. 29-32, 2014.

[KK14] **B. Kruekaew, W. Kimpan** - *Virtual Machine Scheduling Management on Cloud Computing Using Artificial Bee Colony*, Proceedings of the International Multi-Conference of Engineers and Computer Scientists, Hong Kong, China, March 12 - 14, 2014.

[MPG14] **B. Moitree, D. Pradipta, G. Gautam -** *Hybrid of Particle Swarm Optimization and Simulated Annealing for Multidimensional Function Optimization*, International Journal of Information Technology, Volume 20, No 1, 2014.

[MSN10] **P. Mathiyalagan, S. Suriya, S. NSivnaam** - *Modified Ant Colony Algorithm for Grid Scheduling*, International Journal on Computer science and Engineering Volume. 02, No. 02, pp. 132-139, 2010.

[M+13] **A. T. Medhat, E. Ashraf, E. K. Arabi, F. A. Torkey -** *Cloud Task Scheduling Based on Ant Colony Optimization*, The 8th International Conference on Computer Engineering & Systems (ICCES), pp.64-68, 2013.

[PA12] **K. Pardeep, V. Amandeep** - *Independent Task Scheduling in Cloud Computing by Improved Genetic Algorithm*, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Volume 2, Issue 5, 2012.

[PM15] **D. Purnima, K. Mala** - *Workflow Scheduling Using Hybrid Discrete Particle Swarm Optimization (HDPSO) in Cloud Computing Environment*, International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), Volume 3, Issue 12, pp. 12301-12307, 2015.

[PP16] **C. Prerit, S. C. Partha** - *An Improved Min-Min Task Scheduling Algorithm for Load Balancing in Cloud Computing*, International Journal on Recent and Innovation Trends in Computing and Communication, Volume 4, Issue 4, pp. 60-64, 2016.

[RP14] **K. Rajwinder, L. Pawan** - *Load Balancing in Cloud System using Max-Min and Min-Min Algorithm*, International Journal of Computer Applications, pp. 31-34, 2014.

[RVR12] **D. Richa, Y. Vibhash, S. Rahul** - *Review of Various Scheduling Algorithms in Cloud Environment*, Matrix Academic International Online Journal of Engineering and Technology, Volume 3, Issue 2, pp. 32-36, 2012.

[SA12] **K. Shaminder, V. Amandeep** - *An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment*, I. J. Information Technology and Computer Science, 10, pp. 74-79, 2012.

[SA15] **K. Sanjay, M. Atul** - *Application of Min-Min and Max-Min Algorithm for Task Scheduling in Cloud Environment Under Time Shared and Space Shared VM Models*, International Journal of Computing Academic Research (IJCAR), Volume 4, pp.182-190, 2015.

[SE98] **Y. Shi, R. C. Eberhart** - *A modified particle swarm optimizer*, Proceedings of IEEE World Congress on Computational Intelligence, pp. 69-73, 1998.

[SE99] **Y. Shi, R. C. Eberhart** - *Empirical study of particle swarm optimization*, in Proceedings of IEEE Congress on Evolutionary Computation, Volume 3, pp. 1945–1950, 1999.

[SM14] **B. Santhosh, D. H. Manjaiah** - *An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing*, International Journal of Innovative Research in Computer and Communication Engineering, Volume 2, Special Issue 2, pp. 84-88, 2014.

[SR09] **P. Saeed, E. Reza**, - *RASA: A New Grid Task Scheduling Algorithm*, International Journal of Digital Content Technology and its Applications, Volume 3, pp. 91-99, 2009.

[S+12] **H. J. Sung, Y. K. Tae, K. K. Jae, S. L. Jong** - *The Study of Genetic Algorithm-based Task Scheduling for Cloud Computing*, International Journal of Control and Automation Vol. 5, No. 4, pp. 157-162, 2012.

[T+13] **A. M. Tawfeek, E. Ashraf, E. Arabi, A. F. Torkey** - *An ant Algorithm for cloud Task scheduling*, International Workshop on Cloud Computing and Information Security (CCIC2013), 2013.

[UP13] **B. Upendra, N. R. Purvi** - *Enhanced Max-min Task Scheduling Algorithm in Cloud Computing*, International Journal of application or Innovation in Engineering & Management (IJAIEM) Volume 2, Issue 4, pp. 259-264, 2013.

[U+12] **S. G. Umarani, V. M. Uma, P. Shanthi, S. Arul** - *Tasks Scheduling using Ant Colony Optimization*, Journal of Computer Science 8 (8), pp. 1549-3636, Science Publications, 2012.

[Y+13] **Y. Yichao, Z. Yanbo, S. Zhili, C. Haitham** - *Heuristic Scheduling Algorithms for Allocation of Virtualized Network and Computing Resources*, Journal of Software Engineering and Applications, Volume 6, pp. 1-13, 2013.

[ZLL15] **S. Zhao, X. Lu, X. Li** - *Particle Swarm Optimization with Time Varying Parameters for Scheduling in Cloud Computing*, MATEC Web of Conferences 28, pp. 06001-1-06001-6, 2015.

[Z+09] **Z. H. Zhan, J. Zhang, Y. Li, H. S. Chung** - *Adaptive particle swarm optimization*, IEEE Transactions on Systems Man, and Cybernetics - Part B: Cybernetics, 39 (6), pp. 1362-1381, 2009.