Balanced PSO Algorithm for Task Scheduling in Cloud Computing

Lipika Datta 1[a*], Dr. G. Thippanna 2[b]
[a]*Research Scholar, Computer Science and Engineering Department, NIILM University, Haryana, India*
[b]*Associate Professor, Computer Science and Engineering Department, NIILM University, Haryana, India*

***Abstract*—Cloud computing represents a contemporary computing paradigm where IT services are delivered over the Internet on demand. Task scheduling in cloud environments is a critical concern as it involves mapping application tasks to available resources to optimize performance. One prominent algorithm for task scheduling in the cloud is Particle Swarm Optimization (PSO). In PSO, tasks are allocated to resources aiming to minimize computation costs. However, to enhance efficiency further, a Proposed Balanced Particle Swarm Optimization (PBPSO) algorithm has been introduced and implemented. Unlike traditional PSO, PBPSO aims to minimize not only computation costs but also execution time by strategically allocating tasks to available resources. The evaluation of PBPSO considers various time and cost parameters, analyzing their impact on performance metrics such as utilization, speedup, and efficiency. Results from the implementation demonstrate that PBPSO outperforms traditional PSO in optimizing task scheduling in cloud computing environments.**

Keywords—task scheduling; cloud computing; PSO; particle swarm optimization; balanced PSO

I. INTRODUCTION

The distribution of resources between individuals and businesses is currently a prominent concern across diverse sectors. Task scheduling is crucial in cloud computing as it manages resources to enhance system stability, directly impacting user satisfaction. Consequently, the development of various task scheduling algorithms has become a significant area of focus.

Efficient integration of existing task scheduling algorithms can reduce task completion times, meet service quality expectations of users, and improve system load balancing. Cloud computing operates on the principle of resource allocation on demand. It establishes extensive resource pools and selects optimal resources based on user requirements. Virtualization technology centralizes diverse resources, managed automatically by specialized software, thereby relieving users of administrative concerns beyond their tasks.

Under this framework, the relationship between task processing times and costs becomes inseparable. Cloud computing often handles vast tasks, making resource scheduling a critical concern. Cost management, load balancing, and service quality are essential considerations in scheduling tasks effectively [1]. Cloud computing represents a novel computing paradigm aimed at addressing the limitations of restricted resources and significantly reducing costs associated with purchasing, maintaining, and managing physical infrastructure [2,3]. It is built upon advancements in distributed computing, parallel computing, and grid computing [4]. It has gained widespread popularity for its ability to provide IT infrastructure, platforms, and applications as services accessible via the internet [5].

Cloud computing serves a diverse range of sectors effectively. Firstly, it offers practical business models for small computational science and engineering research groups, which often lack the resources and expertise to manage complex computational and data infrastructure for their research [6]. Secondly, it provides substantial benefits to IT organizations by relieving them of the mundane tasks of setting up and maintaining basic hardware and software infrastructures. This frees IT teams to focus on innovation and creating business value [5]. Thirdly, cloud computing presents an appealing opportunity for end users, enabling them to access personal data, programs, storage, and application development platforms from various devices such as PCs, laptops, smartphones, and PDAs through on-demand services provided by cloud providers. This accessibility allows users to leverage advanced technologies without needing deep expertise in each one [7]. Ultimately, cloud computing aims to reduce costs and empower consumers to concentrate on their core business activities rather than being hindered by IT obstacles.

Particle Swarm Optimization (PSO) is an intelligent algorithm inspired by the collective behavior of animals, initially introduced by Kennedy and Eberhart [8]. In PSO, each particle possesses both a position and a velocity. The particle's position at any given time is influenced by its personal best position (pbest) and the best position found by any particle in the entire problem space (gbest). The effectiveness of a particle is evaluated based on a fitness value determined by the specific problem being addressed.

Scheduling refers to the method through which threads, processes, tasks, or data flows are allocated access to system resources (such as processor time, communication bandwidth, and overall system utilization) according to user requirements. A robust scheduling algorithm is crucial in modern systems where multitasking (simultaneous execution of multiple processes) and multiplexing (simultaneous transmission of multiple flows) are essential functionalities [8].

The remainder of the paper is organized as follows. Section II discusses related work and Section III formulates the scheduling model. The optimization model is described in section IV. The flowchart of the proposed algorithm is presented in Section V, and in Section VI the obtained results are presented. Finally, Section VII concludes the paper.

II. RELATED WORK DONE

The authors in [9] have proposed a heuristic approach that combines the modified analytic hierarchy process (MAHP), bandwidth aware divisible scheduling (BATS) + BAR optimization, longest expected processing time preemption

(LEPT), and divide-and-conquer methods to perform task scheduling and resource allocation. In this approach, each task is processed before its actual allocation to cloud resources using an MAHP process. The resources are allocated using the combined BATS + BAR optimization method, which considers the bandwidth and load of the cloud resources as constraints. In addition, the proposed system preempts resource-intensive tasks using LEPT preemption. In [10], the authors have introduced a formal definition of CloudSim simulator, including its architecture, reasons for adopting modeling and simulation, and pros and cons. Moreover, CloudSim versions and how to implement the cloud environment using CloudSim are explained. Further, they have demonstrated the cloudlet scheduler policies: TimeShared, SpaceShared, and Dynamic Workload approach for VM scheduler. TimeShared policy was compared on the basis of some performance parameters in terms of average turnaround time, throughput, total execution time, and total simulation time. They have found that these parameters outperformed in Dynamic Workload cloudlet scheduler policy than TimeShared and SpaceShared approach for the TimeShared VM scheduler policy. The authors in their paper [11] have described and discussed the basic computing resources scheduling and allocation algorithms in cloud computing, in addition to the working mechanism. This paper also presents several experiments conducted based on CloudSim simulation toolkit to assess and evaluate the performance of these scheduling algorithms on Cloud Computing like infrastructure. Furthermore, they introduced and explained the CloudSim simulator design, architecture, and proposed two new scheduling algorithms to enhance the existing ones, and highlighted the weaknesses and/or effectiveness of those algorithms. An enhanced version of the Round Robin algorithm, called Smarter Round Robin (SRR) has been introduced in [12]. The improvements authors brought to the Round Robin algorithm aimed to inject a smarter layer to the existing algorithm to adapt the algorithm to different situations that came with the new delivery model of Cloud Computing as well as reducing the run-time of big data processing. The previous proposed version of the Round Robin algorithm in the paper [13] used a dynamic CPU quantum based on the average burst time, whereas the time quantum would be recalculated each time there were tasks on the waiting list. Authors [12] have demanded that the new perception exhibited great benefits but was unable to detect the changes happening on the waiting list, therefore, even if there were only two tasks on the waiting list and their burst times were very low the algorithm instructed the CPU to recalculate the time quantum thus resulting a high number of context switches and average waiting time. The paper [14] presents an alternative approach to task scheduling problems. It mainly focused on time utilization and resource monitoring. The proposed method is Enhanced Resource Monitor and Scheduler (ERMS) which focuses on both monitoring of available resources and scheduling of user tasks. The Periodic push action method is used to get the status of the virtual machine. The fitness value is calculated and scheduling is done. In the paper [15], a simplified version of a particle swarm optimization-based job scheduling algorithm is proposed for scheduling jobs in the cloud environment to minimize the makespan.

In summary, by using appropriate and effective scheduling schemes, the execution time of the tasks can be minimized and the cloud resources can be fully utilized, which finally increases the availability and scalability of the entire cloud system. Our work here proposes a Balanced PSO methodology to address the issue of job scheduling in a cloud environment with load balancing as an objective.

Objective of the paper

The objective is to find a task-resource mapping instance M such that the highest cost and highest time among all the virtual machines is minimized and load balance is achieved.

III. Proposed Balanced PSO Task Scheduling Model

In this paper a deterministic scheduling approach is considered, i.e., all information about the tasks and their relation to each other such as execution time and precedence relation are known to the scheduling algorithm in advance. The main objective is to minimize the total task completion time (execution time + waiting time + transfer time).

Let the cloud computing system consist of a set of m *VM*s (Eq. 1),

$$VM = \{VM_i : i = 1, 2, 3 \ldots m\} \quad (1)$$

The application is represented by a Directed Acyclic Graph (DAG).

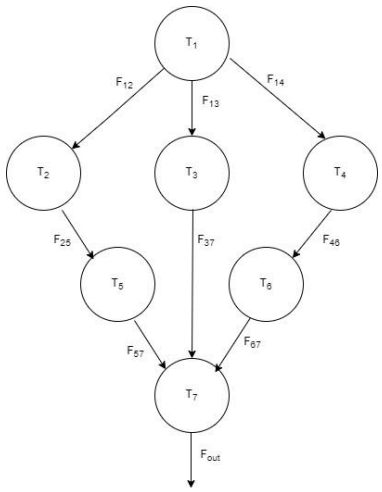

Fig. 1. An application consisting of 7 tasks represented as DAG

A Directed Acyclic Graph (DAG) is represented by $G = (V, E)$, where $i = \{T1, ..., Tn\}$ is the set of tasks, E represents the dependencies among the tasks. The aim is to map every task to a set $VM = \{ VM_1, VM_2, \ldots, VM_m\}$ of m *VM*s. Each task T_i has a weight W_i associated with it, which is the amount of time the task takes to execute on any one of the m *VM*s. Each directed edge E_{ik} indicates dependency between the two tasks T_i and T_k that it connects. $F_{j,k}$ in Eq. 2 represents that the data is produced by T_j *a*nd consumed by T_k.

$$F_{jk} = (T_j, T_k) \quad \in E \quad (2)$$

Assumptions:

1. The average computation time of a task T_k on a compute resource VM_j for a certain size of input is considered known'
2. The cost of unit data access $d_{i,j}$ from a resource i to a resource j is known.
3. $d_{i,j} = d_{j.i}$ for all $i, j \in N$ (3)
4. $d_{i,j} + d_{j.k} > d_{i,k}$ for all $i, j, k \in N$ (4)

The cost of computation of a task on a compute host is inversely proportional to the time it takes for computation on that resource. The transfer cost can be calculated according to the bandwidth between the sender and receiver sites. By considering an application DAG with a set of tasks $T = \{1, ..., k\}$, a set of storage sites $S = \{1, ..., i\}$, and a set of virtual machines $VM = \{1, ..., j\}$ the objective is to allocate tasks to virtual machines (*VM*s) in a way that both the maximum cost and maximum time among all *VM*s are minimized, ensuring a balanced load across all *VM*s i.e., to find a task-resource mapping instance *M* such that, when estimating the total cost and total time for each virtual machine VM_j , the highest cost and highest time among all the virtual machines is minimized and load balance is achieved by following these steps:

A. *Parameters and Variables:*

- Let T_i be the set of tasks.
- Let VM_j be the set of virtual machines.
- Let T_{ij} be the time required to execute task T_i on VM_j.
- Let *W* be the node weight i.e., the cost of execution of a task T_k on compute resource *j*.
- Let $C_{ex}(M)_j$ denotes the total cost of all the tasks assigned to a compute by VM_j.
- Let $C_{tr}(M)_j$ be the total access cost (including transfer cost) between tasks assigned to a virtual machine VM_j and those that are not assigned to that virtual machine in the mapping *M*.
- Let the average cost of communication of unit data between two resources is given by $d_{M(k1),M(k2)}$.
- The cost of communication is applicable only when two tasks have file dependency between them, i.e. when $e_{k1,k2} > 0$.
- Total cost $C_{total}(M)_j$ for a virtual machine VM_j is the sum of execution cost and cost for transfer of resources for particle *j* for a given assignment *M*.
- Let $Cost(M)$ denotes the maximum C_{total}.
- Let $E_{vmt}(M)$ be the total virtual machine time
- Let α be the relative weight among cost and transfer time.

The goal of the assignment is to minimize this cost.

B. *Constraints*

- Each task T_i must be assigned to exactly one VM_j.
- Transmission time is equivalent to the utmost limit of the operational time by any *VM*.
- The total load (cost and time) should be balanced across all *VM*s.

C. *Formulation of Mathematical Models*

The execution cost, transfer cost, total cost and transfer time are calculated as given in Eq. 5 through 1_1.

$$C_{ex}(M)_j = \sum_k w_k \qquad \forall(k) = j \qquad (5)$$

$$C_{tr}(M)_j = \sum_{k1 \in T} \sum_{k2 \in T} d_{M(k1),M(k2)} * e_{k1,k2}$$

$$\forall M(k1) = j \text{ and } M(k2) \neq j \qquad (6)$$

$$C_{total}(M)_j = C_{ex}(M)_j + C_{tr}(M)_j \qquad (7)$$

$$Cost(M)_j = \max(C_{total}(M)_j) \quad \forall j \in P \qquad (8)$$

$$E_{vmt}(M) = \sum_k w_k \qquad k = 1,\ldots, N \qquad (9)$$

$$TransTime(M) = \max(E_{vmt}(M)_k) \qquad \forall j \in N \quad (10)$$

IV. Optimization Model

The fitness function aims to minimize the weighted sum of both the cost and transfer time parameters within the system. It is calculated as stated in Eq. 11.

$$\alpha * (Cost(M)) + (1-\alpha) * (TransTime(M)) \qquad (11)$$

According to PSO, the population is a set of particles in a problem space. Particles are initialized randomly; each particle will have a fitness value. In each iteration, this value is evaluated by a fitness function and is to be optimized. Each particle knows its best position pbest and the best position so far among the entire group of particles gbest. The pbest of a particle is the best result (fitness value) so far reached by the particle, whereas gbest is the best particle in terms of fitness in all population. The evaluation is carried out in a loop until the results converge or the user-specified stopping criteria is reached.

$$\text{pbest}(i,t) = \arg\min_{k=1,\ldots,t} [f(P_i(k))], \qquad i \in \{1 \ldots Np\} \qquad (12)$$

$$\text{gbest}(t) = \arg\min_{i=1,\ldots,N_p} [f(P_i(k))] \qquad (13)$$

In each iteration, the velocity and the position of particles will be updated as follows:

$$V_i(t+1) = \omega V_i(t) + c_1 r_1(\text{pbest}(i,t) - P_i(t)) + c_2 r_2(\text{gbest}(t) - P_i(t)) \qquad (14)$$

$$P_i(t+1) = P_i(t) + V_i(t+1) \qquad (15)$$

A. *Variables Used*

The variables used in formulating the equations [12-15] for optimization according to PBPSO algorithm are given in Table I.

TABLE I. Variables

$V_i(t)$:	velocity of particle i at iteration t
$V_i(t+1)$:	velocity of particle i at iteration t + 1
ω	inertia weight between 0.9 to 0.1
c_1 and c_2	positive acceleration coefficients
r_1 and r_2	random numbers between 0 and 1
$P_i(t)$:	current location of particle i at iteration t
$P_i(t+1)$:	location of particle i at iteration t + 1
$\text{pbest}(i,t)$:	best position of particle I at iteration t
$\text{gbest}(t)$	position of best particle in a population

V. Flowchart of PBPSO

The following flowchart represents the working principle of PBPSO.

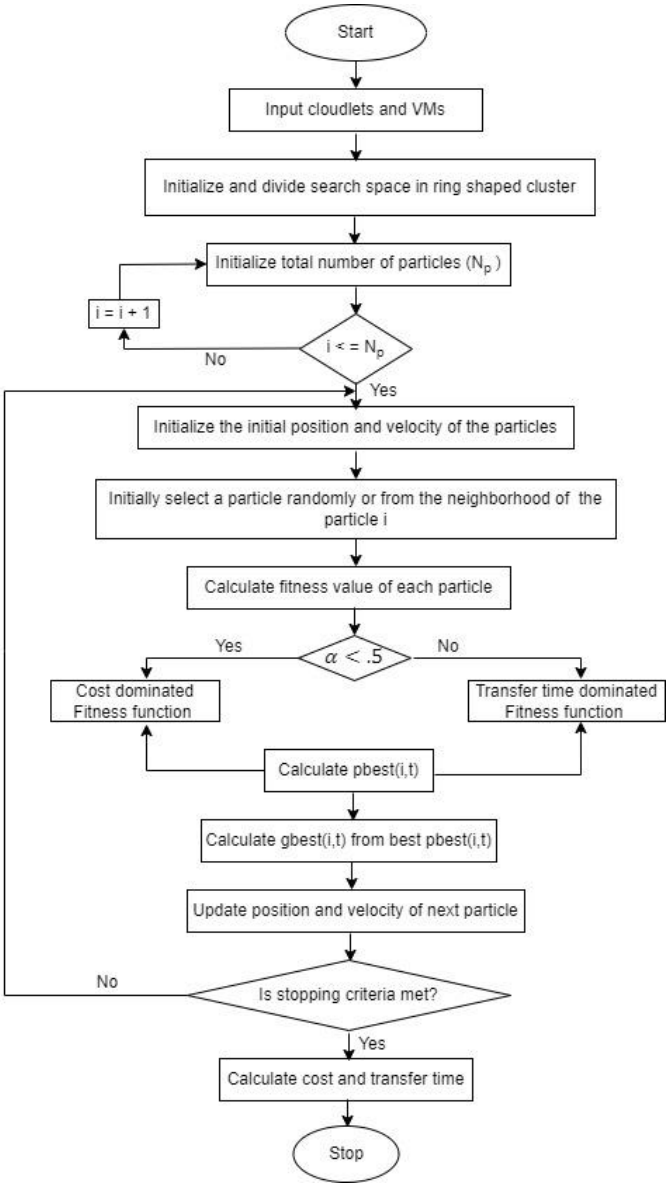

Fig 2. Flowchart of PBPSO

VI. Result and Analysis

The proposed scheduling model described is implemented on CloudSim and also existing algorithms (Min-Min, First Come First Serve, and Particle Swarm Optimization). The method involves employing JSwarm module for simulation execution. Specifically, the current task assigns ten particles, representing virtual devices. Each particle is characterized by its resources: maximum position, minimum velocity, and inertia.

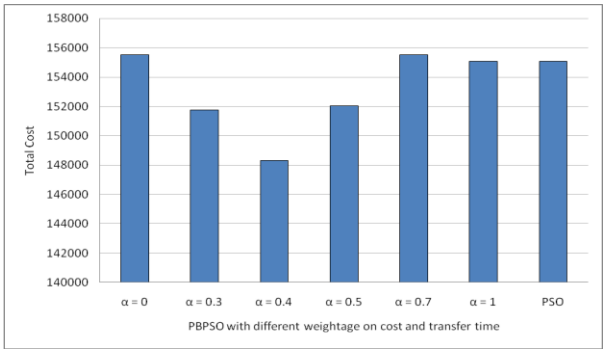

Fig 3. Total Cost vs. Different Approaches when No. of iterations is 700 between the existing PSO and proposed PBPSO algorithms

It is observed that for $\alpha = 0.4$ total cost is minimized.

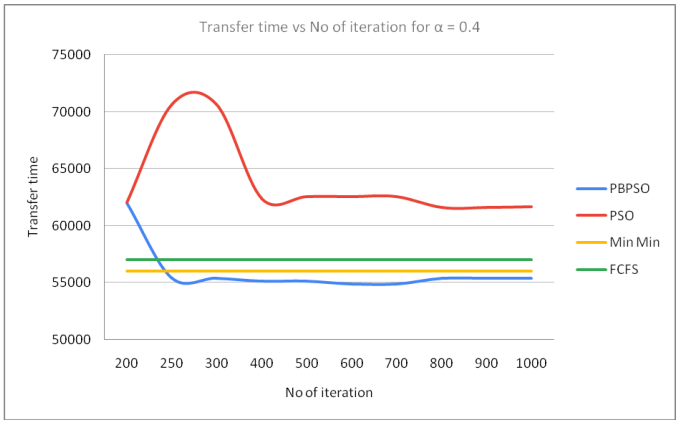

Fig 4. Transfer Time for proposed PBPSO and existing algorithms for 200 to 1000 iterations at ($\alpha = 0.4$).

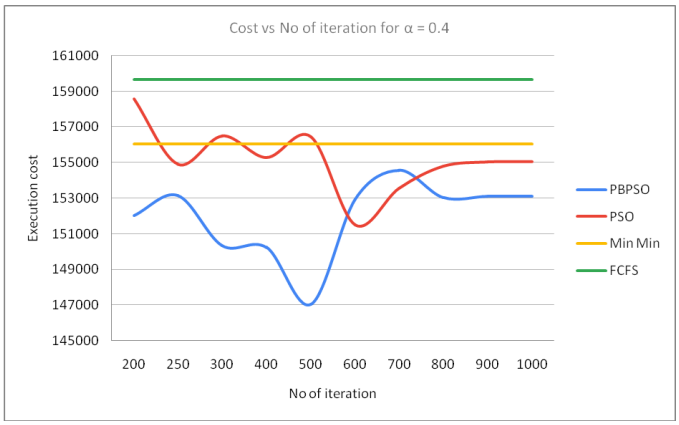

Fig 5. Cost for proposed PBPSO and existing algorithms for 200 to 1000 iterations at ($\alpha = 0.4$).

From Fig. 3, it is observed that the total cost i.e. (Execution cost + Transfer time) is minimum when α is chosen as 0.4. So, it reflects that transfer time is dominating execution cost. It is also observed from Fig. 4 that transfer time is less in PBPSO than PSO and other algorithms for all iterations. Fig. 5 reveals that Execution cost in PBPSO is better than PSO in almost all iterations.

VII. Conclusion

Based on the findings, static algorithms like FCFS and Min-Min focus on maximizing overall machine resource utilization. In contrast, dynamic algorithms efficiently harness the full computational power of the machine while ensuring appropriate resource allocation. Swarm intelligence algorithms allocate ample resources and swarm capacity to each mission or cloudlet. Analyzing the data from these tables, it becomes evident that the PBPSO algorithm significantly outperforms existing methods. By optimizing resource utilization within the system, PBPSO achieves maximum exploitation, thereby enhancing overall effectiveness.

References

[1] M. Cusumano, "Cloud computing and SaaS as new computing platforms," Communications of the ACM, vol. 53, no. 4, pp. 27–29, 2010.

[2] D.-K. Kang, S.-H. Kim, C.-H. Youn, and M. Chen, "Cost adaptive workflow scheduling in cloud computing," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication - ICUIMC '14*, 2014, pp. 1– 8.

[3] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[4] B. P. Rimal, E. Choi, and I. Lumb, “A Taxonomy and Survey of Cloud Computing Systems,” in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009, pp. 44–51.

[5] R. Buyya, S. Pandey, and C. Vecchiola, “Cloudbus Toolkit for Market-Oriented Cloud Computing,” in *Proceedings of the 1st International Conference on Cloud Computing*, M. G. Jaatun, G. Zhao, and C. Rong, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 24–44.].

[6] H.-L. Truong and S. Dustdar, “Cloud computing for small research groups in computational science and engineering: current status and outlook,” *Computing*, vol. 91, no. 1, pp. 75–91, Jan. 2011.

[7] I. Attiya and X. Zhang, “Cloud Computing Technology: Promises and Concerns,” *Int. J. Comput. Appl.*, vol. 159, no. 9, pp. 32–37, Feb. 2017.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4.

[9] Task scheduling and resource allocation incloud computing using a heuristic approach”, Mahendra Bhatu Gawali and Subhash K. Shinde, Journal of Cloud Computing: Advances, Systems and Applications (2018).

[10] Cloud Computing Modeling and Simulation using CloudSim Environment”, Mohammad Oqail Ahmad and Rafiqul Zaman Khan, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

[11] Cloud Computing CPU Allocation and Scheduling Algorithms Using CloudSim Simulator”, Gibet Tani Hicham, El Amrani Chaker, International Journal of Electrical and Computer Engineering (IJECE), Vol. 6, No. 4, August 2016, pp. 1866~1879, ISSN: 2088-8708

[12] Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data”, Hicham Gibet Tani, Chaker El Amrani, Journal of Data Mining and Digital Humanities, ISSN 2416-5999, hal-01443713, 2017

[13] G. T. Hicham, EL.Chaker, Cloud Computing CPU Allocation and Scheduling Algorithms using CloudSim Simulator, International Journal of Electrical and Computer Engineering, Vol 6, No 4, 2016.

[14] Task scheduling in cloud environment using enhanced resource monitor and scheduler algorithm”, V.Seethalakshmi, Dr. V. Govindasamy, Dr. V. Akila, N. Iyswarya, J.Hari haran, G. Vigneshwaran, International Conference on Computation of Power,Energy, Information and Communication (ICCPEIC), 2019.

[15] Ibrahim Attiya, Xiaotong Zhang, “ A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing”, International Journal of Computer Applications (0975 – 8887) Volume 163 – No 9, April 2017