IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-10, Oct 2018 E-ISSN: 2347-2693

IMPROVED RESOURCE AWARE HYBRID META-HEURISTIC ALGORITHM FOR TASK SCHEDULING IN CLOUD ENVIRONMENT

D. Gupta[1*], H.J.S. Sidhu[2]

[1]CSE, Desh Bhagat University, Mandi Gobindgarh, Punjab, India
[2]CSE, Desh Bhagat University, Mandi Gobindgarh, Punjab, India

[*]*Corresponding Author: dineshgupta@ptu.ac.in, Tel.: +91-88474-41684*

Available online at www.ijcseonline.org

Accepted: 15/Oct/2018, Published: 31/Oct/2018

***Abstract*—** Popularity of services over cloud can be estimated from the fact that all major mobile and computer system manufacturers are providing free cloud services to their client on purchase of their product. Easy and fast access to internet services have resulted in increased usage of cloud infrastructure. Also, the amount of data being generated and shared over cloud has increased multi-folds. With ever increasing users, resource provider is aware that they cannot afford wastage or misutilization of its resources. Hence, selection of right resource for fulfilling the request of the customer is vital. Scheduling of tasks over cloud is a key research area. Meta-Heuristic algorithms provide efficient solution to this problem. But, each meta-Heuristic algorithm individually suffers from inherent draw backs. So, there is a need to design a scheduling algorithm that does not suffer from the inherent draw backs of any individual meta-heuristic algorithm and is aware of the current utilization of resources in the cloud. In this paper, an improved resource aware hybrid meta-heuristic scheduling algorithm has been designed which reduces the overall Makespan time, Transfer cost and Response time. It also takes into consideration the current utilization of resources in the cloud.

***Keywords*—** ACO, PSO, VM, Data Centre, Cloud Computing, Cloud, Meta-Heuristic, NIC, Makespan, Response time, Transfer cost.

I. INTRODUCTION

Cloud computing 'as a service' provides *on-demand access to shared pool of resources over Internet in a self-service, dynamically scalable manner [1]*. Cloud computing is fast gaining its feet's in the area of high performance distributed computing. Cloud computing provides large scalability with the ability to easily add and assign resources to different users on demand. It effectively supports multitenancy by facilitating shared access to same resource. The hardware need to homogeneous and it fully supports integration of independent devices. Cloud computing is growing fast and its benefits have not reached to its peak as of yet. Hence, much research is required across a broad array of topics. *One of the important research issues which need to be focused for its efficient performance is scheduling. The goal of scheduling is to map tasks to appropriate resources that optimize one or more objectives [1]*. Scheduling in cloud computing belongs to a category of problems known as NP-hard problem due to large solution space and thus it takes a long time to find an optimal solution. *There are no algorithms which may produce optimal solution within polynomial time to solve these problems*. In cloud environment, it is preferable to find suboptimal solution, but in short period of time [1]. **Section 2** gives a brief introduction to the problem of cloud scheduling. **Section 3** gives an overview of the related work done in task scheduling in cloud environment. **Section 4** and **section 5** provide a brief overview of Particle Swarm Optimization & Ant Colony Optimization respectively. **Section 6** explains the methodology for the proposed algorithm in detail. Results are given in **section 7**. **Section 8** then concludes the outcome of the paper and discusses the scope of future work based on the outcome of this paper.

II. CLOUD SCHEDULING

Cloud scheduling is an optimization problem that deals with optimal allocation of resources among given tasks in a finite time to achieve desired quality of service. The aim is to build a schedule that specifies when and on which resource each task will be executed [2]. There are no available algorithms that may produce optimal solution within polynomial time for Cloud scheduling problem. Meta-heuristic based techniques [3] deal with these problems by providing near optimal solutions within reasonable time.

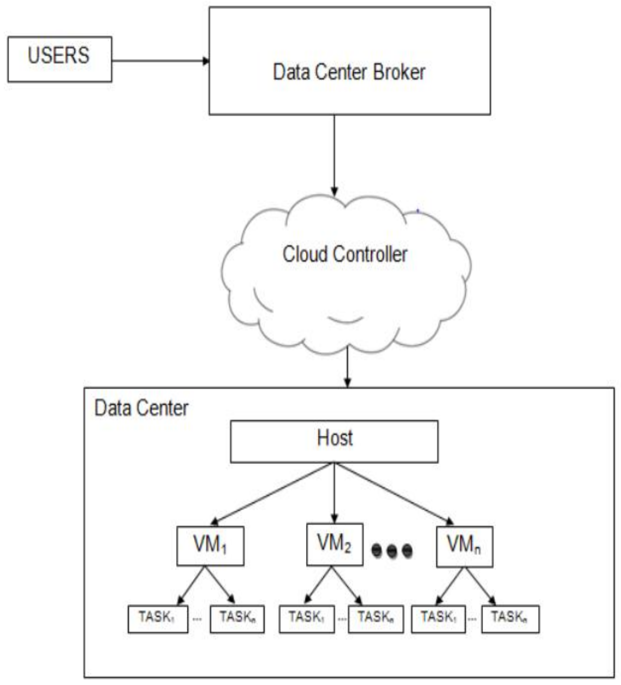

Fig 1: Cloud Scheduling Architecture [4]

III. RELATED WORK

Problem independent Bio-inspired Meta-heuristic algorithms have been used successfully to solve many complex problems in the field of computer networks, data mining, robotics, information security, image processing etc. Still a lot of work can be done to improve the efficiency of these algorithms with the help of innovative ideas or thoughts [5].
Ant Colony Optimization (ACO), a Meta-heuristic algorithm can be used to solve the problem of scheduling tasks to different resources in cloud environment. ACO, which is classified as a stochastic algorithm provides fair scheduling as compared to other static algorithms like algorithm First Come First Serve, Round Robin, etc. [6].

An extensive survey and comparative analysis of various scheduling algorithms for cloud and grid environments based on three popular metaheuristic techniques: Ant colony optimization (ACO), genetic algorithm (GA) and Particle Swarm optimization (PSO) is given in [1]. Metaheuristic techniques are problem independent problems and are used to solve complex problems that cannot be solved in polynomial hence, Hence, Meta-heuristic techniques take more time as compared to deterministic algorithms to provide approximate solutions. Still a lot of work needs to be done towards improving the convergence speed and quality of the solution [1].

A dynamic resource scheduling algorithm, Autonomous agent based load balancing algorithm (A2LB) provides scalability and reliability by offering better resource utilization, and better response time [7]. It is based on three agents. Load agent maintains the information related to load on different resources at a point of time. Channel agent select the best alternative resource where the task can be migrated if the selected resources is overloaded. Finally, the Migration agent is responsible for assigning the task to the newly selected resource [7].

Algorithms for task scheduling may be optimized from the point of view of either the customer or service provider. Goal Oriented Task Scheduling (GOTS) is categorized as service provider optimization algorithm. The objective of this algorithms is to schedule the tasks over different resources in such a way that it generates maximum profit, while using least resource provisioning [8].

Cloud task scheduling problem relates to scheduling of tasks on resources subject to some constraints to optimize some objective function. Scheduling allows optimal allocation of resources among given tasks in a finite time to achieve desired quality of service. The aim is to build a schedule that specifies when and on which resource each task will be executed [2].

Cloud task scheduling policy based on Ant Colony Optimization (ACO) algorithm outperforms FCFS and RR algorithms. But the ACO fails to improve the imbalance factor beyond a factor [9].

Significant advances have been made in the emerging field of nature-inspired computing Nature Inspired Computing algorithms. A lot of research has been field of computational intelligence (CI) and it was stated that NIC and CI intersect. It was further stated that swarm intelligent provide computation intelligence [10].

Metaheuristic algorithms are a natural solution to the problem of task scheduling in cloud environment. But these algorithms as such do not provide a complete solution. A hybrid of Particle Swarm optimization and Ant Colony optimization for load balancing of tasks on cloud resources is proposed in [11]. The result showed that the hybrid algorithm provided better results than Ant Colony Optimization & Particle Swarm Optimization. But the hybrid algorithm could be improved even further by adding resource awareness parameters for better and fair treatment [11].

IV. OVERVIEW OF PARTICLE SWARM OPTIMIZATION (PSO)

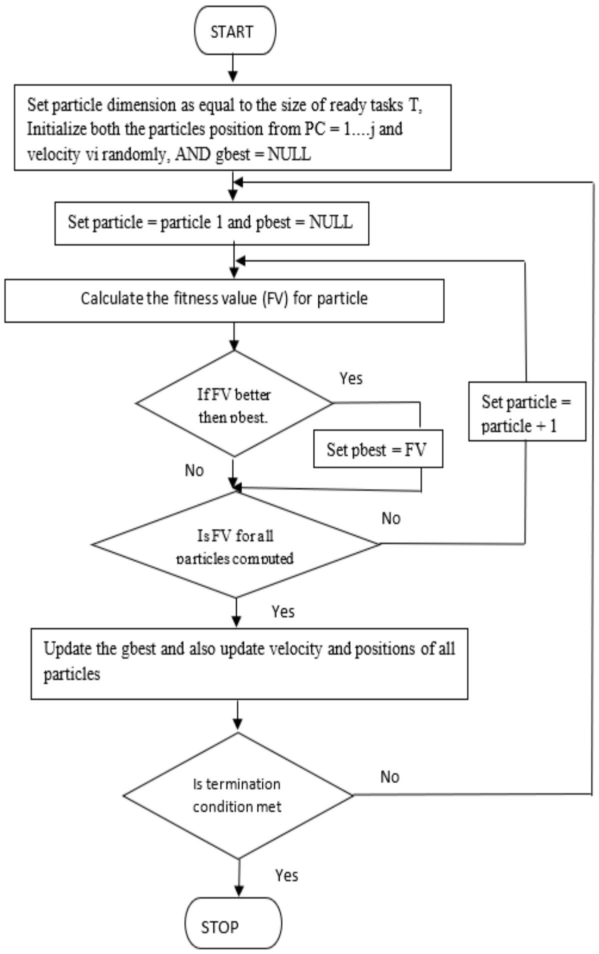
START
Set particle dimension as equal to the size of ready tasks T, Initialize both the particles position from PC = 1….j and velocity vi randomly, AND gbest = NULL
Set particle = particle 1 and pbest = NULL
Calculate the fitness value (FV) for particle
If FV better then pbest.
Yes
Set pbest = FV
No
Set particle = particle + 1
Is FV for all particles computed
No
Yes
Update the gbest and also update velocity and positions of all particles
Is termination condition met
No
Yes
STOP

Fig 2: Flowchart of Particle Swarm Optimization

Particle Swarm Optimization (PSO) is influenced by the social behaviour of animals such as a group of birds searching for food source or a group of fish protecting themselves from a hunter [12]. A particle in this algorithm is referenced to an animal. PSO algorithm is based on two factors: position & velocity. Each particle in the swarm is referenced to by its position that keeps changing with time till it finds the best position or solution. The position of particles in a solution space represents a solution for the problem [1]. Velocity decides the movement of each particle in the swarm. The performance of a particle is measured by a fitness value, which is problem specific. PSO has gained popularity due to its simplicity and its usefulness in broad range of applications with low computational cost. PSO was originally developed for continuous optimization problems. Hence, there is a need to make necessary encoding to solve discrete optimization problems such as scheduling. The first step in PSO is to encode the problem. One method is to represent a particle using a vector representing mapping of resource to a task [13]. Velocities can also be represented using a vector. PSO has fewer primitive mathematical operators than other metaheuristic algorithms which results in lesser convergence time.

V. OVERVIEW OF ANT COLONY OPTIMIZATION (ACO)

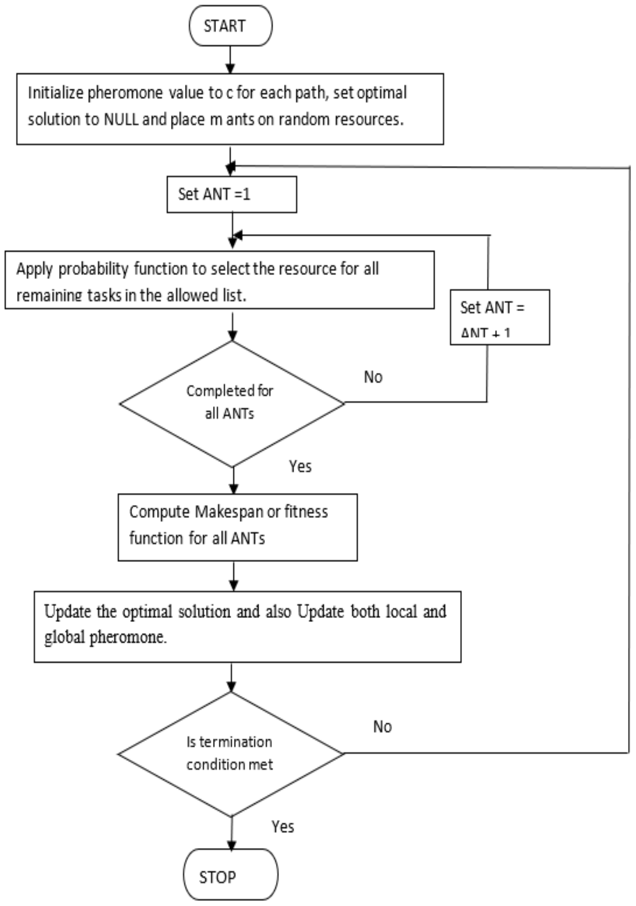
START
Initialize pheromone value to c for each path, set optimal solution to NULL and place m ants on random resources.
Set ANT =1
Apply probability function to select the resource for all remaining tasks in the allowed list.
Set ANT = ANT + 1
Completed for all ANTs
No
Yes
Compute Makespan or fitness function for all ANTs
Update the optimal solution and also Update both local and global pheromone.
Is termination condition met
No
Yes
STOP

Fig 3: Flowchart of Ant Colony Optimization

Ant Colony Optimization (ACO) is a meta-heuristic algorithm. It is inspired by the behaviour of real ants looking for the shortest path between their colonies and source of food. While the ants move from their colony to food source, each ant leaves chemical called pheromones on the path [14]. Pheromone intensity increases with each passing ant drops with the evaporation of pheromone. The pheromone intensity is highest on the shortest path and it is this high intensity that helps ants to recognize smaller paths to the food source [14]. Ant Colony Optimization algorithm can be effectively used for solving the task scheduling problem in cloud.

VI. PROPOSED ALGORITHM

The pheromone value for all paths between each task and resource is set to a constant value at the start of each iteration of the ACO algorithm. It somehow fails to take into account the current load of each resource into consideration for the tasks previously assigned to these resources. In the proposed method, the pheromone value for each path is set based on the degree of imbalance between all resources based on already assigned tasks to each resource. Initially the tasks are scheduled using the Particle Swarm Optimization. Even though the scheduling is done with the help of PSO, the pheromone values are updated as per the schedule prepared using PSO. Once the set number of tasks are scheduled using PSO, remaining tasks are scheduled using PSO [11]. With this the ACO is expected to produce better results as it based on resource awareness. Pheromone value for each path at the start of iterations is set as follows:

For all VM's, i= 0 to sizeof(VmList)-1, in the sorted VM list,

Compute the Degree of Imbalance DI = $\frac{T_i - T_0}{T_{avg}}$ [9]

Set pheromone value for paths between each task and resource VM_i = c – DI, where c is some constant value.

All VM's are then sorted on the basis of length of all cloudlets assigned to them. T_0 has minimum length and pheromone value for let's say T_4 depends on T_0, T_1, T_2, T_3 & T_4.

Also, the probability function in ACO is computed as follows,

$$P^k_{ij}(t) = \left[\frac{[\tau_{ij}(t)]^\alpha * [n_{ij}]^\beta}{\sum s \epsilon\, allowed_k \; [\tau_{is}(t)]^\alpha * [n_{is}]^\beta}\right] \text{ if } j \epsilon\, allowed\ k$$

[9]

0 Otherwise

where $[n_{ij}]^\beta$ = 1/ d_{ij}

d_{ij} is the total time needed by $task_i$ to finish on $resource_j$, which is sum of expected execution time of a $task_i$ on $resource_j$ & the time to transfer the $task_i$ to $resource_j$. It is expressed as follows:

$$d_{ij} = \frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j} + \frac{Input\ File\ Size}{VM\_bw_j} \quad [9]$$

In the equation above, **Transfer cost** = $\frac{Input\ File\ Size}{VM\_bw_j}$

where Input File Size id the total length of the file & VM_bw_j stands for band width of $resource_j$

To reduce the transfer cost along with the overall execution time, it is important to sperate the expected execution and the transfer cost as follows:

$d1_{ij} = \frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j}$

$d2_{ij} = \frac{Input\ File\ Size}{VM\_bw_j}$

now $[n_{ij}]^\beta$ = $[(1/ d1_{ij})^{\beta1} + (1/ d1_{ij})^{\beta2}]$

and hence, β = β1 + β2

```
Algorithm
Input:
        CloudletList: List of all clouldlets received.
        VmList: List of all VM's
Main_Function Hybrid_TS
// initially schedule a small no. of incoming tasks using PSO [number of tasks can be fixed or decided on the basis of percentage of total tasks]. In the proposed work, cloudlets = 2 times the size of VM list are initially passed to PSO scheduling algorithm
  1. PSOCloudletList = CloudletList [1: Sizeof (VmList)* 2]
  2. Call PSO_TS (PSOCloudletList, VmList)
  3. ACOCloudletList = CloudletList – PSOCloudletList
// Scheduling based ACO algorithm
  4. temp_List_of_Cloudlet = null, temp_ACO_List_of_Cloudlet = ACOCloudletList and n= size of VMs list
  5. while temp_ACO_List_of_Cloudlet not empty
        if (size of temp_ACO_List_of_Cloudlet greater than n)
              Transfer the first arrived n Cloudlets from temp_List_of_Cloudlet and put them on temp_List_of_Cloudlet
        else
              Transfer all Cloudlets from temp_List_of_Cloudlet and put them on temp_List_of_Cloudlet
        end If
  6. Call ACO_TS (temp_ACO_List_of_Cloudlet, VmList)
  7. temp_ACO_List_of_Cloudlet = temp_ACO_List_of_Cloudlet - temp_List_of_Cloudlet
  8. Compute the resource utilization (VM utilization), and if it is less than 80%
        a. Compute the degree of imbalance factor between the VM's.
            i. For each VM in the VM List, compute the length of all cloudlets assigned to each VM in the VM List.
           ii. Sort the VM's on the basis of length and create a list UL_VmList of all underloaded VM's and List of Cloudlets OL_VM_CloudletList of Cloudlets in the execution list of overloaded VM's
          iii. Call PSO_TS (OL_VM_CloudletList, UL_VmList) and transfer Cloudlets from overloaded loaded VM to Least loaded VM in the sorted list.
  End Do
Function PSO_TS (CloudletList, VmList)
     1. Set particle dimension as equal to the size of ready tasks T.
     2. Initialize particles position randomly from PC = 1.....j and velocity vi randomly.
```

3. For each particle, calculate its fitness value.
4. If the fitness value is better than the previous best pbest, set the current fitness value as the new pbest.
5. Perform Steps 3 and 4 for all particles and select the best particle as gbest.
6. For all particles, calculate velocity and update their positions.
7. If the stopping criteria or maximum iteration is not satisfied, repeat from Step 3 & 4.

Function ACO_TS (CloudletList, VmList)

```
1. Sort the VM's on the basis of length. i.e. length of cloudlets assigned to each VM.
2. Initialize pheromone value for each path between tasks and resources as follows:
3. For all VM's, i= 0 to sizeof(VmList)-1, in the sorted VM list,
        Compute the Degree of Imbalance DI = $\frac{T_i - T_0}{T_{avg}}$
        Set pheromone value for paths between each task and resource VM$_i$ = c – DI, where c is some constant value.
4. Set optimal solution to NULL and place m ants on random resources.
5. Repeat for each ant
        a. Put the starting resource of first task in tabu list and all other tasks in allowed list.
        b. Based on the probability function or transition rule, select the resource for all remaining tasks in the allowed list.
6. Compute fitness of all ants which in this case is Makespan time.
7. Replace the optimal solution with the ant's solution having best fitness value if its value of better than previous optimal solution.
8. Update both local and global pheromone.
9. Stop when the termination condition is met and print the optimal solution.
```

Degree of imbalance (DI) can be computed using equation 1 & 2. It measures the imbalance between VMs [9].

$$T_i = \frac{TL\_Tasks}{Pe\_num_j * Pe\_mips_j} \quad (1)$$

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (2)$$

TL_Tasks refers to the length of all tasks assigned to VM$_i$. T_{max}, T_{min} and T_{avg} refer to the average of T_i among all VMs [9].

Resource/VM utilization can be measured using equation 3:

$$\left[\frac{\sum_{i=1}^{n} \text{time taken by resource } i \text{ to finish all jobs}}{Makspan\ X\ n}\right] \quad (3) \qquad [1]$$

Where n is total numbers of resources or VMs.

VII. RESULTS & DISCUSSION

CloudSim Libraries & NetBeans IDE is used for implementing the task scheduling. CloudSim is an open source software under the GPL license developed in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, at the Computer Science and Software Engineering Department of the University of Melbourne. It frees the developers from the issues related to cloud-based infrastructures and services and focus on only the systems design issues that they wish to investigate. CloudSim is a simulation tool where you can test the performance of provisioning policies in a reliable and free of cost platform. CloudSim is a library consisting of classes for describing various cloud components like data centres, computational resources, virtual machines, applications, users, and policies to manage scheduling and provisioning. Net Beans IDE is a free, open source, integrated development environment (IDE) that enables you to develop desktop, mobile and web applications. NetBeans in this project is used to execute the CloudSim libraries which are written using Java.

Table 1: Parameters Setting of Cloudsim

Entity Type	Parameters	Values
Task(Cloudlet)	Length of Task	20000-400000
	Total Number of Task	150-250
Virtual Machine	Total Number of VMs	8
	MIPS	1024-4096
	VM Memory (RAM)	128-512
	Bandwidth	500-2000
	Cloudlet Scheduler	Time_shared and Space_shared
	Number of PEs Requirement	2
Data Centre	Number of Datacenter	1
	Number of Host	3
	VM Scheduler	Time_shared and Space_shared

Table 2: Makespan Time with fixed VMs = 8

Cloudlets	ACO	PSO	A2LB	Hybrid of PSO and ACO [11]	Proposed Algorithm
150	270	250	240	180	160
175	309	300	285	220	198
200	325	335	310	240	210
225	355	370	335	260	226
250	403	425	375	295	253

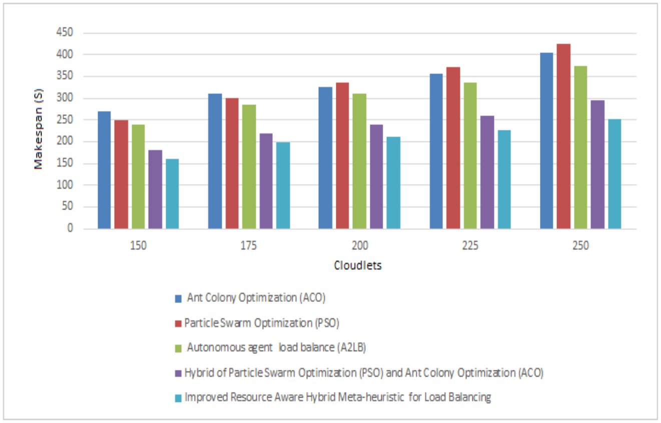

Fig 4: Makespan time with fixed VMs = 8

Overall Makespan of the proposed algorithm is much better than other Meta-Heuristic algorithms and also the Hybrid of PSO & ACO proposed in [11]. Main contributors to this improvement in Makespan time are use of resource utilization as balance factor and giving high weightage to transfer cost in the probability function. Overall transfer cost is shown in next figure.

Table 3: Total transfer cost with fixed VMs = 8

Cloudlets	ACO	PSO	A2LB	Hybrid of PSO and ACO [11]	Proposed Algorithm
150	2053	2115	2090	2053	1880
175	2395	2488	2472	2393	2311
200	2737	2840	2826	2738	2509
225	3080	3195	3171	3075	2620
250	3422	3547	3522	3421	2679

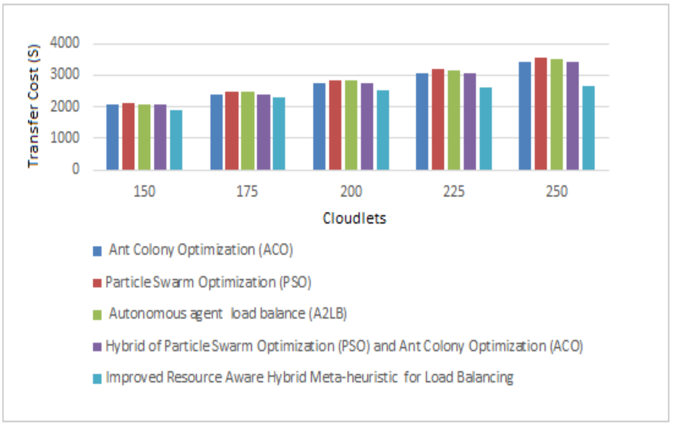

Fig 5: Total Transfer cost with fixed VMs = 8

Transfer cost is much lower than other scheduling algorithms. Lower transfer costs lead to reduced overall processing time. The overall processing time of any cloudlet includes execution time of that cloudlet on particular VM and transfer cost.

Table 4: Total Response Time with fixed VMs = 8

Cloudlets	ACO	PSO	A2LB	Hybrid of PSO and ACO [11]	Propose d Algorit hm
150	20273	20109	20214	20145	17892
175	23651	23608	23623	23388	21012
200	26354	26490	26017	25715	23780
225	30814	31111	30242	29837	27554
250	33788	34245	33191	32651	30234

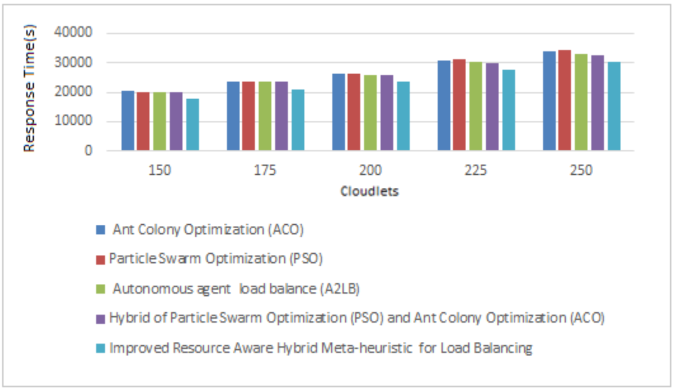

Fig 6: Total Transfer cost with fixed VMs = 8

It is clear that the proposed algorithm also results in lower response time.

VIII. CONCLUSION

In this paper an "Improved Resource Aware Hybrid Meta-Heuristic Scheduling Algorithm" is proposed for solving the problem of task scheduling in cloud. The main objective of this algorithm is to provide the updated load on each resource at every iteration so that the choice of resources for next set of tasks is based on resource awareness. In addition to that, the proposed algorithm focuses on two aspects of total execution time of each task on a resource individually. The total execution time is based on execution/computation time and the transfer time/cost. the proposed algorithm is hybrid of Ant Colony Optimization and Particle Swarm Optimization. The results show that the Makespan time of the proposed algorithm is better than the Meta-Heuristic algorithms (ACO, PSO), A2LB (Autonomous Agent based Load balancing and Hybrid of Ant Colony Optimization & Particle Swarm Optimization proposed in [11]. Also, the transfer cost was computed for all the algorithms and the transfer cost of proposed algorithm is much lesser as compared to Meta-Heuristic algorithms (ACO, PSO), A2LB (Autonomous Agent based Load balancing and Hybrid of Ant Colony

Optimization & Particle Swarm Optimization proposed in [11]. The proposed algorithm also results in less response time as compared to other algorithms. Hence it can be concluded that the performance of the proposed algorithm in terms of Makespan time, Transfer cost & Response time is much better then Meta-Heuristic algorithms (ACO, PSO), A2LB (Autonomous Agent based Load balancing and Hybrid of Ant Colony Optimization & Particle Swarm Optimization proposed in [11]. In future the further improvement can be gained by changing the objective function to facilitate other performance criteria's as well. Also, the implementation can be shifted to real environment to test online data.

REFERENCES

[1] M. Kalra, S. Singh, "A review of metaheuristic scheduling technique in cloud computing", Cairo University, Egyptian Informatics Journal, Vol. 16, issue 3, pp 275-295, 2015.

[2] Karger D, Stein C, Wein J, "Scheduling Algorithms. Algorithms and Theory of Computation Handbook". Chapman & Hall/CRC, p 20, 2010.

[3] Talbi E. G., "Metaheuristics: From Design to Implementation", Wiley, Canada, p 593, 2009.

[4] V. Behal, A. Kumar, "Comparative Study of Load Balancing Algorithms in Cloud Environment using Cloud Analyst", International Journal of Computer Applications (0975 – 8887) Volume 97– No.1, July 2014.

[5] G. Singh, A. Kaur, "Bio Inspired Algorithms: An Efficient Approach for Resource Scheduling in Cloud Computing", International Journal of Computer Applications (0975–8887) Volume 116, No. 10, April 2015.

[6] S. Selvaraj, J. Jaquline, "Ant Colony Optimization Algorithm for Scheduling Cloud Tasks", International Journal of Computer Technology & Applications, Vol 7(3), pp 491-494, May-June 2016.

[7] A. Singh, D. Juneja, M. Malhotra, "Autonomous Agent Based Load Balancing Algorithm in Cloud Computing", In the proceedings of 2015 ELSEVIER Procedia Computer Science International Conference on Advanced Computing Technologies and Applications (ICACTA- 2015), At Mumbai, India, Vol. 45, pp 832-841, 2015.

[8] J. Thaman, M. Singh, "Current Perspective in Task Scheduling Techniques in cloud Computing: A Review", International Journal in Foundations of Computer Science & Technology (IJFCST) Vol. 6, No. 1, January 2016.

[9] M. Tawfeek, A. El-Sisi, A. Keshk, F. Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization" , The International Arab Journal of Information Technology, Vol. 12, No. 2, March 2015.

[10] N. Siddique, H. Adeli, "Nature Inspired Computing: An Overview and Some Future Directions", Cognitive Computing, Vol. 7, Issue 6, pp 706-714, 2015.

[11] D. Gupta, H.J.S. Sidhu, "Hybrid Task Scheduling Algorithm Based on ANT Colony Optimization and Particle Swarm Optimization for Cloud Environment", International Journal of Computer Sciences and Engineering (2347-2693), Vol. 6, Issue. 2, pp. 324-328, 2018.

[12] J. Kennedy, R. Eberhart, "Particle swarm optimization" In proceedings of IEEE International Conference on Neural Networks, vol. 4, pages 1942–1948, 1995.

[13] Guo L, Zhao S, Shen S, Jiang C, "Task scheduling optimization in cloud computing based on heuristic Algorithm". Journal of Networks, Vol 7, pp. 547–553, 2012

[14] Dorigo M. and Stützle T., "Ant colony optimization". MIT Press; 319 p, 2014.

Authors Profile

Mr. Dinesh Gupta, did his M. Tech in IT from Department of CSE GNDU Amritsar, India. Currently he is pursuing his Ph. D in CSE from Desh Bhagat University. He has more than 8 years of experience in teaching. Currently he is working as Assistant Professor in department of CSE, IKGPTU, India. He has more than 8 publications in leading research Journal.

Mr C H Lin pursed Bachelor of Science and Master of Science from University of New York, USA in year 2009. He is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Telecommunication, University of New York, USA since 2012. He is a member of IEEE & IEEE computer society since 2013, a life member of the ISROSET since 2013 and ACM since 2011. He has published more than 20 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. His main research work focuses on Cryptography Algorithms, Network Security, Cloud Security and Privacy, Big Data Analytics, Data Mining, IoT and Computational Intelligence based education. He has 5 years of teaching experience and 4 years of Research Experience.