Received: January 26, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multi Objective Task Scheduling Using Modified Ant Colony Optimization in Cloud Computing

Gogi Reddy Narendrababu Reddy[1]* **Singamsetty Phanikumar[2]**

[1]*Gunampalli Narayanamma Institute of Technology and Science, India*
[2]*Gandhi Institute of Technology and Management University, India*
* Corresponding author's Email: narendraphd2017@gmail.com

Abstract: Cloud computing is the development of distributed computing, parallel computing, and grid computing, or defined as a commercial implementation of such computer science concepts. One of the main issues in a cloud computing environment is Task scheduling (TS). In Cloud task scheduling, many Non deterministic Polynomial time-hard optimization problem, and many meta-heuristic (MH) algorithms have been proposed to solve it. A task scheduler should adapt its scheduling strategy to changing environment and variable tasks. This paper amends a cloud task scheduling policy based on Modified Ant Colony Optimization (MACO) algorithm. The main contribution of recommended method is to minimize makespan and to perform Multi Objective Task Scheduling (MOTS) process by assigning pheromone amount relative to corresponding virtual machine efficiency. MACO algorithm improves the performance of task scheduling by reducing makespan and degree of imbalance comparatively lower than a basic ACO algorithm by its multi-objective and deliberate nature. Experimental outcomes have shown that proposed MACO to have makespan 350 milliseconds and average utilization of 0.51 for a set of 100 tasks.

Keywords: Meta-heurestic, Modified ant colony optimization, Multi objective task scheduling, Non deterministic polynomial time-hard optimization problem, Task scheduling.

1. Introduction

In recent years, data centres play an important role in global computing platforms such as sending frequent mail, and other operations like search, read, write, etc. Cloud computing is inter-based computing system, which is highly dynamic. Task Scheduling (TS) and resource allocation are main issues in cloud computing [1]. Based on the aspects of Platform as a Service (PaaS), Software as a Service (SaaS), Infrastructure as a Service (IaaS), these computing platforms provides a collection of host networks through the internet. According to Information Technology Laboratory, the definition of cloud computing is the process of sharing their computer processing resources and data to other devices which are in demand [2]. Cloud computing is an on-demand resource for networks, servers, programs and services with high speed and little effort in interaction of users. One of the recent challenges in a computing platform is TS [3]. The general issue in TS is NP-hard optimization problems like travelling salesman problem and combinatorial problems like integer programming and addressing problems. These issues occur in the allocation of hundreds and thousands of Virtual Machines (VM) to cloud resources that cause a delay in the performance of the TS [4]. To overcome these problems in the TS, Ant Colony Optimization algorithm (ACO) has been implemented [5].

In TS process data centres, brokers, VM and cloudlets have been created for performing the task required by using the Cloudsim toolkit to satisfy customer's demand [6]. Cloudsim is a framework for simulation of cloud infrastructure, where data centres are resource provider, broker's help in the creation and destruction of VMs, cloudlets perform

and submit tasks to provide information about RAM size, bandwidth and number of CPU allocations [7]. Here the traditional ACO algorithm finds best shortest path by representing graphs. The major purpose of this algorithm helps to lessen the makespan of performing task [8]. This algorithm has the capability to adopt according to dynamic applications, for example, it can modify its character from an artificial ant to real ant. Comparing to FCFS (First Come First Serve) and RR (Round Robin) algorithm, ACO finds better solutions for travelling salesman problems and reduce makespan [9]. During runtime overhead, task load gets increase along with lack of rapid adaptability which results in more execution time and decrease in convergence rate will enact as a major pitfall to basic ACO [10].

In this article, a MACO optimization algorithm is put forth in order to achieve anticipated performance in the cloudsim framework by means of selecting multiple processors, reducing makespan and high convergence speed in minimum time. MACO focuses on deliberate nature of assigning pheromones to virtual machines according to its corresponding efficiency and in assigning of tasks based on processing speed, makespan, bandwidth etc. Hence, a cloudsim system can be able to perform humongous load of tasks with greater efficiency.

This paper is composed as follows. Section 2 describes various methodologies employed in ACO algorithm to enhance its effective functionality. The modified ACO algorithm with its standard structure and workflow along with its fundamental functions and specific modifications are given in Section 3. Section 4 presents results and experimental setup, which shows comparison study between ACO and Modified ACO. Conclusion with future work is represented in Section 5.

2. Literature review

H. Cui, Y. Li, X. Liu, N. Ansari, and Y. Liu [11] presented a multi objective model with constraints for tackling the TS issue in cloud computing. The multi-objective model takes task processing mode of cloud system into account, and illustrates scheduling performance in terms of makespan, flow time and reliability by using Queuing theory and Markov process. Proposed method of Genetic Algorithm-based Chaotic Ant Swarm (GA-CAS) algorithm, in which four operators and natural selection are applied to solve this embarrassed multi-objective optimization problem. But issue involved in such method is design of operator cost is bit high and complex, that leads to lack of balance in load to processors.

H. He, G. Xu, S. Pang, and Z. Zhao [12] presented PSO based Adaptive Multi Objective Task Scheduling (AMTS) Strategy. Objective of advanced strategy is maximizing the resource utilization minimizes the task completion time, energy consumption and average cost. To retain particle diversity, the adaptive acceleration coefficient was implemented to show improved results of PSO algorithm achieve quasi-optimal solutions for cloud TS problem, but the algorithm cost was slightly higher is a typical pitfall to AMTS strategy.

C.W. Tsai, W.C. Huang, M.H. Chiang, M.C. Chiang, andC.S. Yang [13] presented hyper heuristic scheduling algorithm (HHSA) to find superior scheduling solutions for cloud computing systems. Diversity recognition and development discovery operators are utilized by HHSA to dynamically conclude that which primary level is to be employed in searching superior candidate results. Results illustrate that proposed algorithm can be realistic to different cloud computing system that moderates makespan of TS combined with other scheduling algorithms estimated on both CloudSim and Hadoop. Some of the drawbacks in this methodology are to find a candidate solution for high level heuristic which is not possible. Moreover, it requires effective operators to achieve a better performance.

L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara [14] have presented the multi objective optimization (MOO) algorithm to handle TS issues and requirements of biodiversity in cloud computing. Author introduces resource cost model, which was mirroring the link between users' resource costs and budget costs. Depends on this model multi objective optimization scheduling method was proposed. Their method presented better performance with regards to the parameters like makespan, cost, deadline violation rate, resource utilization. Experimental results showed that MOO method recovered only half percent in finest case development, which is not efficient at all in cloudsim environment.

S.K. Panda, and P.K. Jana [15] presented the four sorts of TS calculations such as Cloud Z-Score Normalization (CZSN), Cloud Decimal Scaling Normalization (CDN), Cloud Distribution Scaling Normalization (CDSN) and Cloud Nearest Radix Scaling Normalization (CNRSN) for heterogeneous multi cloud condition. Initial two calculations depend on conventional standardization systems, particularly z-score and decimal scaling and

following two calculations depend on two recently arranged standardization procedures, such as distribution scaling and nearest radix scaling. The critical issue is proposed a method that is estimated through simulation by determining makespan and average cloud utilization but not consider the task execution which is necessary to manipulate efficiency of Cloudsim architecture.

Muthu, A.B.A. and Enoch, S [16], stated a bacterial foraging optimization algorithm combined with genetic algorithm (GABFO) to realize reliability and scheduling difficulties in cloud processes. The foremost subscription of GABFO is to introduce an optimization tactic to plan the tasks proficiently and assign the resources in an efficient manner.

Nagaraju, D. and Saritha, V [17] presented the multi objective genetic algorithm for mobile cloud (MOGAMCC) environment. In the direction of devicing the MOGAMCC, the cloudsim toolkit was protracted with the MOGA and its task planning approach governs the optimal scheduling policy. A single point crossover system is engaged for the development of new population. The pitfall of this approach is it takes very high response time.

Loganathan, S., Saravanan, R. and Mukherjee, S [18] introduced a job scheduling approach is proposed to allocate job to a VM of active hosts by considering job classification and pre-emption. So that minimalizing the number of host used in provision in turn decreases the energy consumption in the Cloud datacenter. In this algorithm, classifying the task in to three different kinds and allotted based on pre-emption policy with the first available time of resource in host. It suffers from the drawback of more resource utilization and SLA violation.

In this article, a proposed MACO algorithm helps to decrease makespan significantly in order to resolve the above mentioned drawbacks and to make cloudsim

3. Problem definition

The scheduling issue refers to decrease the overall execution time of a set of tasks and balancing the workload. Moreover, inside the cloudsim, multiprocessor scheduling maps a set of tasks to the number of processors in order to minimize the execution time. Major problems with task scheduling are makespan and the degree of imbalance. The current paper focuses on these two problems.

4. Task scheduling algorithm based on modified ant colony optimization in CloudSim

The TS method creates a highly ordered scheduling mechanism for the cloud network in order to optimize the use of resources available in the network. The cloudlets are assigned to every VM according to their respective allocation policy. Then the cloudlets wait for a processor with more cost, span and some processors sit idle in some VMs. These will increase both the manipulation cost and makespan. These are very critical in the case of processing the huge amount of data. So, in this paper MACO algorithm is employed to reduce the makespan in TS. The MOTS process is utilized to reduce the makespan and balance the entire system workload and this helps to handle both the parameter without affecting one another without collapsing fundamental structures and efficient task scheduling routine of cloudsim structure.

4.1 Cloudsim structure of task scheduling

The CloudSim allows simulation of modeling like IaaS, PaaS and SaaS, which offer basic components like Hosts, VMs, and applications. These models provide the different services. CloudSim provides an extensible modeling framework and seamless simulation platform with a greater application performance. CloudSim enrols itself as a library for modeling typical cloud computing scenarios and support whole virtual framework and endorse an effective computing platform. It offers many essential classes that define by computable shared resources, data centres that hold them, hosts are present in data centres and each and every host comprised of a set of virtual machine to process their corresponding activities or scheduled tasks. Mainly VMs have two methods like allocation and scheduling. By using allocation and scheduling factor, one can emphasize a new strategy that governs the use of the cloud. VM efficiently frames the TS algorithm and load balancing algorithm. These algorithms optimize the makespan and balance the workload of the entire system. The Fig.1 represents the cloudsim structure of TS as shown below which includes:

Provinces: It similar locations in terms of geography where the cloud could serve its customers, provision shared data to them, by means of analytics in cloud computing. This province (region) includes all other elements of Cloudsim structure.

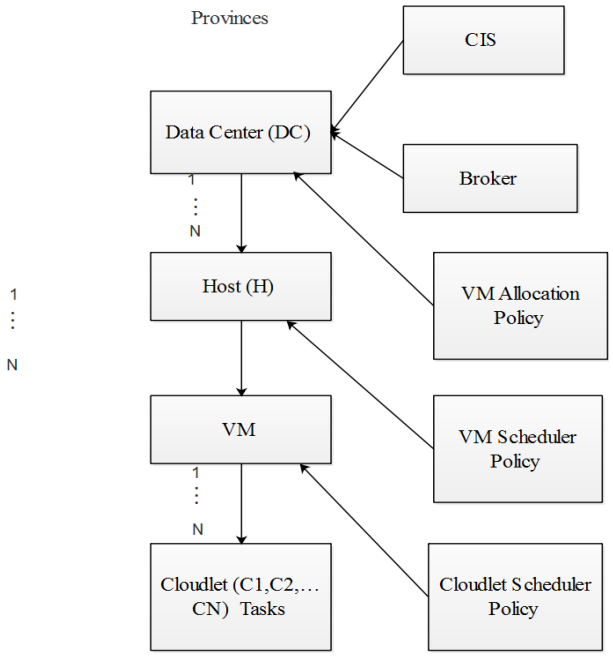

Figure.1 CloudSim structure of TS

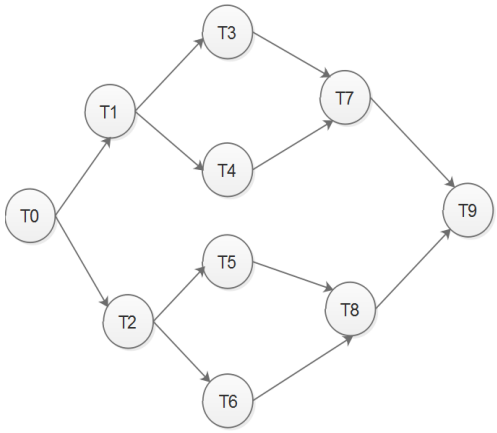

Figure.2 Work flow of task model

Datacentres: It resembles framework services offered by a variety of services in Provisional's of cloud and also comprises of a collection of computable servers or hosts. It shall be either homogeneous or heterogeneous depends upon the character of the individual data centre. It comprises of all data about host, user, VM and every individual task.

User: It simulates a set of users normally taken as a single component at the run time of the simulation and its duties is to cause traffic to modeling.

Hosts: It simulates physically present components in the real time environment for example storage devices and computers.

Virtual Machine: It resembles a real time machine or computer which processes the task in order to produce desired output. The method of processing and its data regarding task are all taken from Datacentre.

VM Allocation Policy: It simulates allocation algorithm for every individual VM to be allocated on the host will be depending on this policy.

VM scheduler: It simulates the type of scheduling policy to be followed by the system, it may be either space shared or time shared policy to provision processor to every individual VM.

Cloudlet: It models the cloud-based application services and it's a new architectural element that extends cloud computing infrastructure. The main purpose of the cloudlet is supporting resource intensive and interactive mobile applications by providing powerful computing resources to mobile devices with lower latency.

Broker: Broker determines the selection of data center from which the service is to be provisioned in response to which the user request and it enacts itself as a mediator between all other components in the toolkit.

Scheduling the workflow of tasks efficiently becomes a challenging issue in the Cloud Computing environment, because the scheduling decides performance of the applications. ACO algorithm is employed to solve the scheduling problem. The ACO algorithm helps to find the optimal resource allocation for tasks in cloud system and also helps to minimize the makespan of tasks in the entire system.

4.2 Work Flow of Scheduling Model

In this model represents the basic structure of a set of task workflow and the set of nodes $\{T_0, T_1, T_2, \ldots\ldots, T_n\}$ represents the workflow of tasks. The set of edges $E = \{(T_i, T_j) | T_i, T_j \in T\}$ represents precedence constraints between two tasks in the workflow. If task T_i has a directed edge pointing to task T_i, T_j is called the parent task of T_j and T_j is the child task of T_i. The Fig. 2 contains ten tasks to be processed and these tasks are labelled from T0 to T9. Here, each task is equally distributed to the processor so, the cloud computing system easily achieves the load balancing in the entire system.

In workflow of cloudsim the major issue to be faced is the selection of VM to every individual task in an efficient manner which considers about processing capability of every individual processor and also about processor status whether it is idle or not to resolve these challenges ACO had been coined.

4.3 Ant colony optimization algorithms

In ACO scheme ants are initially placed at all VMs randomly and then they are initialized with a level of pheromone according to MIPS, bandwidth and a number of processors according to their initial VM. Then ants are allowed to move from one VM to another randomly by a process called the selection of the next VM. In this process, ants are not selected their next VM according to other trails or global pheromone. Ant will always follow, initial stage selects VM randomly, then comparing with the predefined probability, if it goes more than that random VM or it goes to another VM. With the help of Tabu table history, the ant will check whether the VMs are visited or not.

In case VMs are already visited means repeat the selection process otherwise visit that selected VM and marked it has in the Tabu table.

The probability of an ant for choosing next VM can be given as follows.

$$P_{ij}(VM) = \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}}{\sum S \in allowed_j [\tau_{is}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}} \quad (1)$$

Where

$P_{ij}(VM) \rightarrow$ Probability to choose another VM by an ant;

$\tau_{ij}(t) \rightarrow$ Pheromone amount deposited between VMs at indices i and j;

$\eta_{ij}(t) \rightarrow$Visibility for heuristic algorithm found by its calculation which expresses execution time and processing speed of VM;

Parameters α, β provides a description about the relative pheromone deposit and visibility of other pheromones respectively.

Earlier research works have attempted to formulate a new kind of ACO method based on finding even optimized ways of working with bandwidth and traffic, but all of the existing left out of the initial state of idle processors. Also, those are not concern about the proper feeding of tasks to each and every processor on VM to avoid overloading. These are all the aims drawn up for a proposed method, to solve instead of looking upon a cluster of processors as VM, it will try to optimize for each and every single processor inside every VM. A proposed method will perform ACO optimization for all processors in VMs based on their capability of every individual processor and store it up as an optimized list by ACO optimization. The advanced algorithm also focuses the multi objective task; it will balance the multiple metrics.

4.4 Modified ant colony optimization algorithm

In this paper, the main objective of the proposed algorithm is to improve the efficiency of TS, reduction of make span and also balances the multi objective tasks. Initially, all ants are presented on processors randomly. Then, all the ants provide pheromone, depending on pheromone values ants are allowed to move. An ant places pheromone in a processor according to its processing speed, the amount of traffic to reach, the number of processors present in that processor and all. Then it registers that VM on its tour history similarly all ants tend to move in a similar way. By this proposed method developer can feed the number of tasks to the more processing VM and less number to slow processing VM. Thus developer need not worry about utilization of resources by the framework and left alone of some best working processors in their VM.

The pseudo code for the proposed scheme is shown below:

Pseudo Code: Algorithm: Modified Ant Colony Optimization Algorithm
Procedure: MACO[A.Vm,k,Cldt]
$k \leftarrow \emptyset : Vm \leftarrow (Vm1, Vm2, \dots\dots, Vmn)$
$A \leftarrow Ant1, Ant2, \dots\dots, AntM$
$clt \leftarrow (clt1, clt2, \dots\dots\dots, cltn)$
$P \leftarrow Pheremone\ pr \leftarrow probability$
$B \leftarrow Broker\ D \leftarrow DataCenter$
$cldsim[Vm, cldt] \leftarrow cldShedMod(mips, numpes) \leftarrow MACO[A.K]$
Class cldsim[Vm,cldt]
function Main()
Vm= new Vm(vmid, B-id,mips,numpes,cldShedMod(mips,numpes));
end function
end class
Class cldShedMod [mips,numpes] extends cldShed
$rcl \leftarrow$Rem Cld
function CldFin(rcl)
getcloudletfinishedlist().add(rcl);
end function
function updateVmprocessing
for $rgl \leftarrow cloudlet\ to\ Finish$
getCloudletExelist().remove(rgl);
cloudletFinish(rgl)
end for
end function
function cloudletSubmit(cld, filetranstime)
for $i \leftarrow MACO, k$
rcl.setMachineAndPeId(H-id,i);
end for
end function

```
end class
class MACO[A,k]
                    A ← Ant
pr ←Prob(A)
functioncomp_dist()
for i ∈ Pelist.size() do
for j ∈ Pelist.size() do
dist[i][j]= cld.size/pe.Mips;
end for
end for
end function
function main()
BT ← BestTour
k ← index of BT
for i ← BT.length − 1
for j ← BT.length − 1
if(BT[i]>BT[j])
Tmp=k[i];
 k[i]=k[j];
k[j]=Tmp;
end if
end for
    End for
end class
```

The proposed scheme can be evaluated in several ways and it can be compared with an ACO scheme in order to know about its performance, efficiency, and better utilization of resources against several existing frameworks some of them are makespan, a number of iterations took reduce makespan. The MACO process balances the makespan and an entire system load, in balancing the time it does not affect the one parameter to another. It will reduce the makespan as well as balance the system load efficiently. Factors that took influence over the efficiency of a cloudsim process are:

Degree of Imbalance DI: In order to reduce overloading of tasks to a processor, and utilization of Bandwidth, RAM, and MIPS in order to reduce traffic for cloudlets. The DI is used to measure imbalance among processors to spread workload evenly for every processor,

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

Where T_{max} and T_{min} are highest and shortest time span took for a cloudlet to process in the proposed framework and T_{avg} is given as the average time span took by the system to process one cloudlet. Thus, DI will interpret the balance of workload among different processors in order of their equally spread working time for the overall task.

Execution time (Makespan) T_E: If workload of j^{th} task which is hosted on i^{th} resource of processing capacity P_i then execution time of j^{th} a task is $\frac{W_i}{P_i}$. It is assumed that the execution time is proportional to the execution cost. The X_{ji} indicates the binary variable for the position of the task j and if the task was hosted on the resource i, Hence,

$$T_E = \left(\frac{W_i}{P_i}\right) \times X_{ji} \quad (3)$$

5. Experimental Result

The proposed algorithm was implemented in Java NetBeans 8.2 version and 32 bit operating system, 8GB RAM. In this paper, MACO achieved reduction of makespan and multi objective TS process. The purpose of MACO is to minimize the makespan and balancing the workload in TS. The proposed algorithm is better compare to the other traditional algorithm such as basic ACO, GA, and PSO. In the following experiments, the basic ACO and MACO algorithm with different iterations, associated with the makespan of 10-100 tasks set. The next parameter of DI in the following experiments,

The Fig. 3 indicates the number of iterations like 10, 20, up to 100 along the X axis and makespan 0, 1, 2, 3, 4 along the Y axis. The graph represents that the proposed algorithm is better than the basic ACO and other existing modified ACO (ACO_PSO) [19]. It means MACO balance the entire system load perfectly.

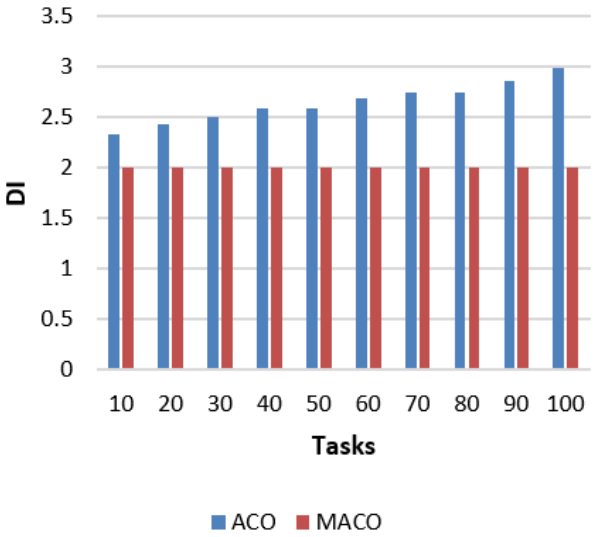

Figure.3 Number of iterations vs makespan

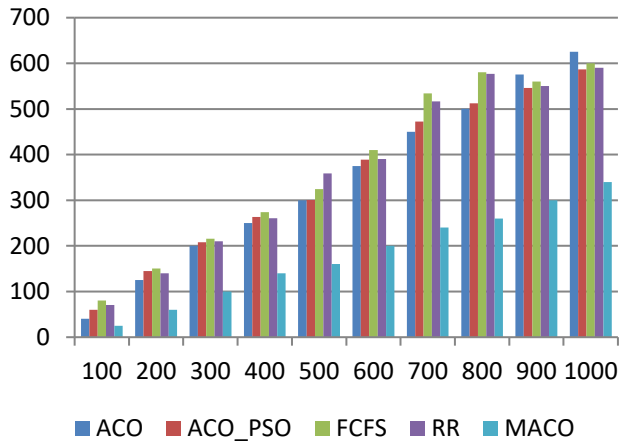

Figure.4 A number of tasks vs Makespan

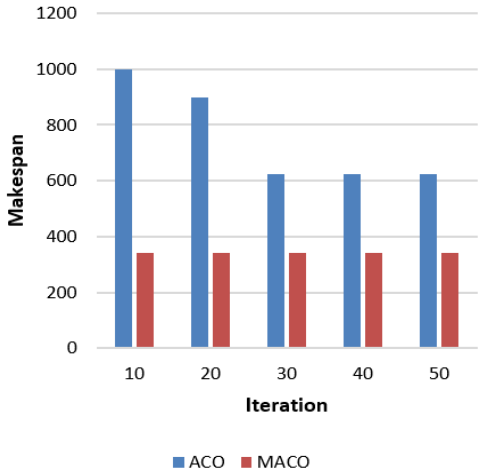

Figure.5 Number of Iterations vs Makespan

The Fig.4 indicates the number tasks like 10 to 100 along x axis and makespan of each task like 100-700 along the X-Y axis. According to this graph, MACO achieves reduction of makespan better than ACO.

Fig.5 represents the number iterations like 10-50 along the X axis and makespan along the Y axis. The graph indicates that the performance of MACO algorithm is better than the basic ACO algorithm. It means that the proposed algorithm achieves better performance.

A TS algorithm based on MACO is implemented and the algorithm is run for 8 problem instances with the number of processors as 8. Ten trials are done for each problem instance with ACO and the average value of wait time of tasks and utilization of each processor are obtained.

Table 1 indicates the individual processor utilization obtained for a single trial by varying number of tasks and Table 2 by varying trial runs up to 10 for task sets of fixed size 150. Utilization of available resources by each processor is computed and results are tabulated. In Table 1 utilization will become 1 due to full utilization after heavy loads more than efficiency of processors and in Table 2 some utilization of processors will be higher cause of their lack of efficiency in cloudlets at a time or lower MIPS.

The basic ACO algorithm increases the overhead task at runtime and its lack rapid adaptivity. So, it increases execution time and decreases the convergence rate. So, MACO algorithm is utilized to overcome these problems. Also it won't take into account about individual processors in a machine and leaves some idle processors in VM therefore leads in reducing efficiency of virtual machine and hence it is followed by many enhancements to get more efficient ACO which is a Modified ACO algorithm.

Table 1. Individual processor utilization for a task sets of different size using ACO in a single trial

PE\tasks	5	10	15	20	25
10	0.326	0.652	0.978	1	1
15	0.217	0.434	0.333	0.859	1
20	0.163	0.326	0.489	0.652	0.815
25	0.026	0.083	0.125	0.167	0.208
30	0.0195	0.065	0.097	0.134	0.163

Table 2. Individual processor utilization for a task set of size = 150 using ACO

Processor	Trail 1	Trail2	Trail3	Trail 4	Trail5	Trail6	Trail7	Trail8	Trail9	Trail 10	Avg Utilization
1	0.656	0.700	0.885	0.406	0.763	0.773	0.298	0.364	0.573	0.364	0.578
2	0.582	0.674	0.663	0.842	0.686	0.548	0.466	0.721	0.478	0.364	0.466
3	0.395	0.671	0.289	0.695	0.734	0.704	0.824	0.710	0.562	0.601	0.824
4	0.670	0.313	0.689	0.590	0.562	0.549	0.705	0.714	0.788	0.689	0.705
5	0.428	0.729	0.593	0.659	0.788	0.601	0.669	0.278	0.818	0.593	0.669
6	0.417	0.524	0.605	0.644	0.351	0.535	0.653	0.818	0.590	0.605	0.653
7	0.742	0.524	0.410	0.466	0.466	0.337	0.466	0.321	0.788	0.410	0.466
8	0.688	0.780	0.584	0.367	0.538	0.328	0.513	0.590	0.705	0.584	0.513

6. Conclusion

In this paper, MACO algorithm was used for achieving TS with reduced makespan in a cloud environment. A drawback of ACO algorithm is that makespan value may not vary throughout the runtime and bit slow convergence time. So now MACO adopted for solving this problem and to improve the convergence speed. ACO also utilizes random amount of pheromone assigned to virtual machines but in our MACO deliberate amount of pheromone according to the efficiency of corresponding virtual machine only will be allocated to make better attraction to effective VM. The experimental results showed that the MACO reduced the makespan effectively and also achieved the MOTS process efficiently. The plan execution evaluated in the cloudsim framework. The proposed method achieved a better result in terms of reduction of makespan. Whether the sizes of the tasks are same or not, MACO can handle all conditions. MACO gave better performance compared to the basic ACO, GA, PSO. In future, the cloud framework will be improved to handle varying behaviour of VMs along with allocation to schedule cloudlets.

References

[1] A. Yadav and S.B. Rathod, "Priority based task scheduling by mapping conflict-free resources and optimized workload utilization in cloud computing", In*: Proc. of International Conf. on Computing Communication Control and Automation*, pp.1-6, 2016.

[2] M. Anuradha and S. Selvakumar, "ACO Based Task Scheduling Algorithm for Hybrid Cloud", *International Journal of Emerging Technology in Computer Science & Electronics*, Vol.13, No.1, pp.976-1353, 2015.

[3] S. Xue, M. Li, X. Xu, J. Chen, and S. Xue, "An ACO-LB algorithm for task scheduling in the cloud environment", *Journal of Software*, Vol.9, No.2, pp.466-473, 2014.

[4] R.G. Babukartik and P. Dhavachelvan, "Hybrid Algorithm using the advantage of ACO and Cuckoo Search for Job Scheduling", *International Journal of Information Technology Convergence and Services*, Vol.2, No.4, pp.25, 2012.

[5] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang, and D. Wu, "A novel collaborative optimization algorithm in solving complex optimization problems", *Soft Computing*, Vol.21, No.15, pp.4387-4398, 2017.

[6] S. Sharma and P. Kuila, "Design of Dependable Task Scheduling Algorithm in Cloud Environment", In: *Proc. of Third International Symposium on Women in Computing and Informatics*, pp.516-521, 2015.

[7] M.A. Tawfeek, A. El-Sisi, A.E. Keshk, and F.A. Torkey, "Cloud task scheduling based on ant colony optimization", In: *Proc. of International Conf. on Computer Engineering & Systems*, pp.64-69, 2013.

[8] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization", In: *Proc. of Sixth Annual International Conf. on Chinagrid*, pp.3-9, 2011.

[9] A. Razaque, N.R. Vennapusa, N. Soni, and G.S. Janapati, "Task scheduling in Cloud computing", In: *Proc. of International Conf. OnLong Island Systems, Applications and Technology*, pp.1-5, 2016.

[10] K.N. Baxodirjonovich and T.Y. Choe, "Dynamic Task Scheduling Algorithm based on Ant Colony Scheme", *International Journal of Engineering and Technology*, Vol.7, No.4, pp.1163-1172, 2015.

[11] H. Cui, Y. Li, X. Liu, N. Ansari, and Y. Liu, "Cloud service reliability modelling and optimal task scheduling", *IET Communications*, Vol.11, No.2, pp.161-167, 2016.

[12] H. He, G. Xu, S. Pang, and Z. Zhao, "AMTS: Adaptive multi-objective task scheduling strategy in cloud computing", *China Communications*, Vol.13, No.4, pp.162-171, 2016.

[13] C.W. Tsai, W.C. Huang, M.H. Chiang, M.C. Chiang, and C.S. Yang, "A hyper-heuristic scheduling algorithm for cloud", *IEEE Transactions on Cloud Computing*, Vol.2, No.2, pp.236-250, 2014.

[14] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing", *IEEE Access*, Vol.3, pp.2687-2699, 2015.

[15] S.K. Panda and P.K. Jana, "Normalization-Based Task Scheduling Algorithms for Heterogeneous Multi-Cloud Environment", *Information Systems Frontiers*, pp.1-27, 2016.

[16] A.B.A. Muthu and S. Enoch, "Optimized Scheduling and Resource Allocation Using Evolutionary Algorithms in Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol.10, No.5, pp.125-133, 2017.

[17] D. Nagaraju and V. Saritha, "An Evolutionary Multi-Objective Approach for Resource Scheduling in Mobile Cloud Computing", *International Journal of Intelligent Engineering and System*, Vol.10, No.1, pp.12-21, 2017.
[18] S. Loganathan, R. Saravanan, and S. Mukherjee, "Energy aware resource management and job scheduling in cloud datacentre", *International Journal of Intelligent Engineering and Systems*, Vol.10, No.4, pp.175-184, 2017.
[19] M. Goyal and M. Aggarwal, "Optimize workflow scheduling using hybrid ant colony optimization (ACO) & particle swarm optimization (PSO) algorithm in cloud environment", *International Journal of Advance Research, Ideas and Innovations in Technology*, Vol.3, No.2, pp.181-189, 2017.