IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

An Experimental Algorithmic Result Of Cloud Computing For Resource Optimization Based On Virtual Machines

Ms. Rachna Pandey[1] Prof.(Dr.) R.K.Bathla[2]

Ph.D Research Scholar, Department of CSA, Desh Bhagat University,Mandi Gobindgarh, Punjab, India[1]
Professor, Department of CSA, Desh Bhagat University,Mandi Gobindgarh,, Punjab, India[2]

Abstract

In this paper, presentation of different static & stochastic meta-heuristic algorithms is compared with the proposed algorithm. Static algorithms are easy to implement but they fail to provide even acceptable solutions. Ant based algorithms are very popular for task scheduling related problems in cloud computing. Autonomous agent-based load balancing algorithm (A2LB) is a dynamic agent based resource scheduling algorithm that providesscalability and reliability by offering better resource utilization, and minimum response time,but it results in high degree of migration. Particle Swarm Optimization (PSO) is a swarm- based meta-heuristic algorithm influenced by the social behaviour of animals such as bird orfish. PSO has fewer primitive mathematical operators than other metaheuristic algorithms which results in lesser convergence time and is applied to continuous value problems. Ant Colony Optimization (ACO) is also a swarm based meta-heuristic algorithm inspired by the behaviour of real ants looking for the shortest path between their colonies and a source of food. Ant Colony Optimization algorithm is suited for solving discrete problems and can beused in solving the cloud computing resource management and job scheduling. Algorithm proposed in thesis tries to overcome the limitations of ACO & PSO.

Keywords:- ACO, PSO, Autonomous agent-based load balancing algorithm (A2LB)

1. INTRODUCTION

Optimization problems are prevalent in all domains and involve finding optimal or near-optimal solutions. Large problems often require a sequential approach, focusing on objective functions. Researchers define steps for problem recognition, model construction, and solution evaluation [17]. Cloud task scheduling is an optimization problem involving efficient resource allocation to achieve a desired objective. The objective function determines the best alternative, and even slight changes can lead to different solutions. Developers face numerous optimization problems, requiring them to choose the optimal or near-optimal option based on evaluation criteria. These problems typically aim to maximize or minimize the evaluation function.

Optimization problems have the following characteristics:

- Numbers of decision alternatives are available from which one alternative is selected.
- Selection of the alternate is subject to constraints that limit the number of available alternatives.
- Evaluation criteria is directly affected by choice of decision alternate.
- An evaluation function defined on the decision alternatives helps to describe the effect of the different decision alternatives.
- Decision alternate for an optimization problem should be based on all available constraintsand the one that maximizes/minimizes the evaluation function.

1.2 Ant Colony Optimization

Ant Colony Optimization (ACO) comes under the category of met heuristic algorithms. Thealgorithm is based on real ants and how they search for food. The ants travel from their colonyto the food source. The ants leave pheromones as they walk. Initially random ants select random paths. The pheromone they leave also evaporates but at lesser intensity. So, the shortest path after some time is the one with highest pheromone intensity which leads all otherants to follow that path. After a certain period, all ants choose that path and which happens tobe the shortest path [45].

Following is the pseudo code for implementation of ACO. [63]

Pseudo Code 1: ACO algorithm

//Input

Input: List of Cloudlet (Tasks) and List of VMs

//Output

Output: The best solution for tasks allocation on VMs Steps:

//Pseudo Code

1. Initialize:

   Set Current_iteration_t=1.

   Set Current_optimal_solution=null.

   Set Initial value τij(t)=c for each path between tasks and VMs

//random assignment of ants on VM

2. Place m ants on the starting VMs randomly.

// selection of VM for each task

3. For k:=1 to m do

   Place the starting VM of the k-th ant in tabuk.Do ants_trip while all ants don't end their trips Every ant chooses the VM for the next task.

   Insert the selected VM to tabuk.

   End Do

// updating the optimal solution

4. For k:=1 to m do

   Compute the length Lk of the tour described by the k-th ant according to EquationUpdate the current_optimal_solution with the best founded solution.

5. For every edge (i, j), apply the local pheromone.
6. Apply global pheromone update according to Equation 7.
7. Increment Current_iteration_t by one.

// check if maximum iterations done

8. If (Current_iteration_t < tmax)Empty all tabu lists.

   Goto step 2Else

   Print current_optimal_solution.

   End If

7. Return

Pseudo code 2: Scheduling based ACO algorithm

//Input

Input: Incoming Cloudlets and VMs List

//Output

Output: Print “scheduling completed and waiting for more Cloudlets”Steps:

//Pseudo Code

1. Set Cloudlet List=null and temp_List_of_Cloudlet=null

2. Put any incoming Cloudlets in Cloudlet List in order of their arriving time.
3. do ACO_P while Cloudlet List not empty or there are more incoming CloudletsSet n= size of VMs list

// if the no. of cloudlets left more than VMs

if (size of Cloudlet List greater than n)

Transfer the first arrived n Cloudlets.from Cloudlet List and put them on temp_List_of_Cloudlet

// if the no. of cloudlets left less than VMs

Else

Transfer all Cloudlets.from Cloudlet List and put them on temp_List_of_Cloudlet
end If

Execute ACO procedure with input temp_List_of_Cloudlet and nend Do

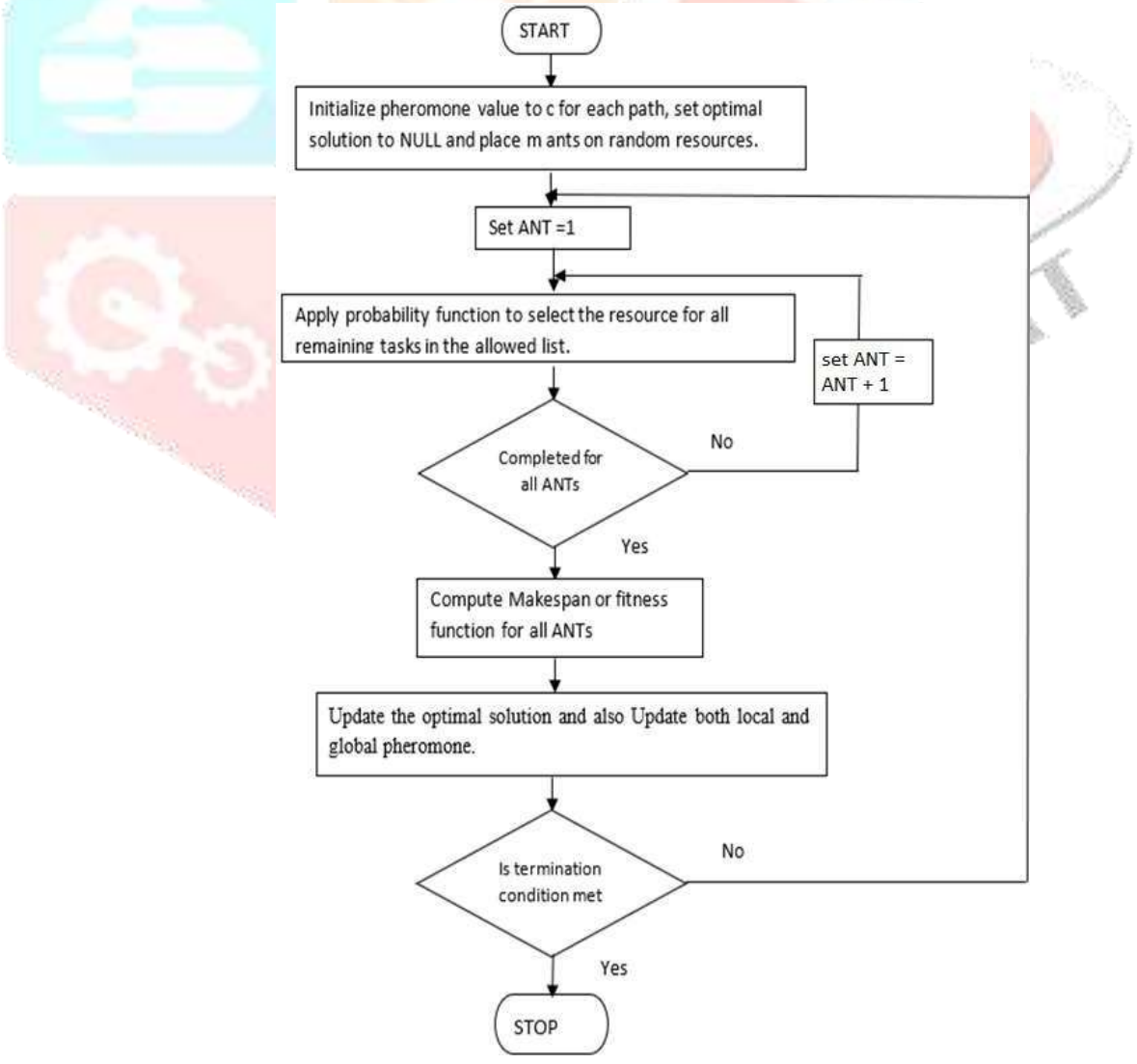


Fig. 1: Flow Diagram of Ant Colony Optimization Scheduling Algorithm

2. LITERATURE SURVEY

OLB (Opportunistic Load Balancing) , begins by assigning the tasks randomly or in free order to available resources on the cloud. It does so by assigning workload to nodes in free order. The implementation is very easy as no computation is required and it does not consider any constraints while assigning tasks to resources. For instance, it does not considerthe expected execution time of task on different resources. Idea is to ensure that all resourcesor machines get work.

MET (Minimum Execution Time) is also simple and easy to execute. The tasks are assigned to the machines considering that it should take minimum time to execute the task. Itseems very valid. But it does not take into consideration the current load and availability of the machine. It simply assigns the task to the best machine. This strategy can result in poor load balance across various machines. Load balance Min-Min (LBMM) [70] is an example of Minimum Execution Time task scheduling algorithm.

MCT (Minimum Completion Time) is a strategy that works differently than the Minimum Execution Time. Rather than considering the Execution Time, it works on Completion Time of the task. It may take more time to execute, but the task is guaranteed to complete in minimum time as compared to other machines. Each task is assigned arbitrarily to the machines which possesses the minimum completion time to complete this task. However, the strategy fails to ensure that task takes minimum execution time.

MOMCT (Modified Ordered Minimum Completion Time) proposes an algorithm using which it is possible to identify MCT (Minimum Completion Time) that allocates tasks in a random order to the minimum completion time machine. It suggests an ordered approach to the MCT heuristic, which order tasks in accordance to the mean difference of thecompletion time on each machine and the minimum completion time machine.

Min-min is based on Completion Time of all tasks that are still waiting for the resource allocation. Idea is to compute the matrix for minimum completion time of every task which is still waiting for resource allocation. Task with minimum completion time is scheduled to the respective machine on which its completion time is minimum. The task is then removed fromthe list of tasks that are waiting for resource allocation and the same procedure is followed forall the remaining tasks in the list.

Min-max is also based on Completion Time of tasks and is quite similar to Min-min heuristic on the basis of its implementation. Only difference between min-min and max-min is the selection of corresponding machine where it should execute. It also has a set of all unscheduled tasks. Again, we compute the matrix for minimum completion time of every task which is still waiting for resource allocation. But, rather than selecting the task with overall minimum time, here the task with overall maximum completion time is scheduled the respective

machine on which its completion time is maximum. The task is then removed from the list of tasks that are waiting for resource allocation and the same procedure is followed forall the remaining tasks in the list.

GA (Genetic Algorithm) is another popular heuristic strategy used to find near-optimal solution for complex problems It is a population-based heuristic. First step of GA is to randomly initialize the population of chromosomes. Objective function is then designed basedon one or two parameters. One of the most widely used QOS parameter is Makespan time. After getting the initial population, all chromosomes in the population are evaluated on the basis on their respective fitness value (Makespan time). Next step is to perform a crossover operation that selects a random pair of chromosomes of a task and picks a random point in first chromosome. Allocation of resources is also exchanged between particular corresponding tasks. Last step is to perform mutation operation. It randomly selects a chromosome and taskwithin the chromosome and the task is then re-assigned or re-allocated to the selected resource. The same process is repeated for number of iterations till the stopping criteria is met,which it the objective function.

In paper Load balancing strategy has been implemented using Genetic Algorithm.Generally, the problem of task scheduling in cloud computing is dynamic in nature, still at some points you have a certain set of tasks to be assigned to available resources. In this paper,two vectors were used to represent the current load of the VM's at any given time and information related to the job submitted to the cloud. The focus of this paper was to optimize the cost function. Simulation results shows that performance of load balancer using GA is much better than other static algorithms.

In paper a cloud task scheduling policy based on Ant Colony Optimization (ACO) [38] algorithm has been implemented and its performance is compared with different scheduling algorithms like First Come First Served (FCFS) and Round-Robin (RR). In this paper, a probabilistic function on the basis of expected time to compute for each task has been proposed. The probabilistic function takes into consideration the pheromone concentration, transfer time of task, expected time to compute, length of each task, processing capabilities of each Virtual machine including band width are considered. Then the paper also suggests a function for updating the pheromone value. The pheromone value is updated after each tour by ant. Computed length after each tour by an ant refers to the Makespan value. The function also considers the trail decay, i.e. decay of pheromone concentration at each path. The main goal of these algorithms is minimizing the Makespan of a given tasks set. Experimental resultsshowed that cloud task scheduling based on ACO outperformed FCFS and RR algorithms. Tasks varying in size from 100 to 100 are then scheduled using the Ant Colony Optimization algorithm. Performance of the metaheuristic ACO is much better as compared to static algorithms like FCFS and Roubd-Robin. But paper does not suggest any measure to improve the imbalance factor beyond the implementation of evolutionary ACO algorithm. The algorithm does not consider parameters other than Makespan time and also the probabilistic function used in this paper can be extended to consider other issues. The probability functioncan be designed by considering different QoS parameters.

3. Objective of the study

- To study and analyze various meta heuristic load balancing algorithms.
- To design efficient algorithm for providing system load balancing using Novel Hybrid(ACO & PSO) based technique.
- To design efficient algorithm for providing the system load balancing using Novel GAtechnique.
- To compare the results of A2LB Algorithm with the proposed Hybrid (ACO & PSO)based technique and Novel GA based technique for load balancing on the basis of:
  - Over all response time
  - Data Center Service Time
  - Transfer Cost.
  - To minimize the execution time for improving the resource utilization of the balanced machines by using Novel Resource Aware Scheduling Algorithm.
  - To compare the results of existing resource aware algorithm with the proposed Novel Resource Aware Scheduling algorithm on the basis of:
    - Over all response time
    - Data Center Service Time
    - Transfer Cost.

4. RESULTS AND DISCUSSIONS

4.1 Performance comparison of First Come First Serve, Shortest Job First & AntColony Optimization

Parameters Setting of CloudSim

The experiments are implemented with 10 data centers with 40VMs and 600-1000 tasks under the simulation platform. The length of the task is from 20000 Million Instructions (MI) to 40000 MI. The parameters setting of cloud simulator are shown in table 8**.**

Table 1: Parameters Setting of Cloudsim for performance comparison of First Come First Serve, Shortest Job First & Ant Colony Optimization

Entity Type	Parameters	Values
Task(Cloudlet)	Length of Task	20000-400000
	Total Number of Task	600-1000
Virtual Machine	Total Number of VMs	20-40
	MIPS	256-512
	VM Memory (RAM)	1024
	Bandwidth	500
	Cloudlet Scheduler	Time_shared and Space_shared
	Number of PEs Requirement	1-2
Data Centre	Number of Datacenter	10
	Number of Host	2
	VM Scheduler	Time_shared and Space_shared

The implementation has been done using two approaches. In the first approach, size of VMhas been kept fixed and the size of cloudlets is changed. In the second approach, size of cloudlets is fixed and the size of VM is changed. The experiment result for the two approachesis given below:

Keeping the VM's Fixed

First Come First Serve, Shortest Job First and Ant Colony Optimization have been implemented keeping the VM's fixed to 40. Experiment was conducted by assigning cloudletsin range of 600 to 1000.

First come first serve has been implemented by assigning the cloudlets to the VM which is idle. It is implemented by maintaining a queue which is automatically implemented by CloudSim. By default, the allocation done by CloudSim is using FCFS.

Table2: Makespan Time (in seconds) using First Come First Serve with 40 VM's

Scheduling Algorithm	No. of Cloudlets				
	600	700	800	900	1000
First Come First Serve	1730	2021	2233	2541	2702

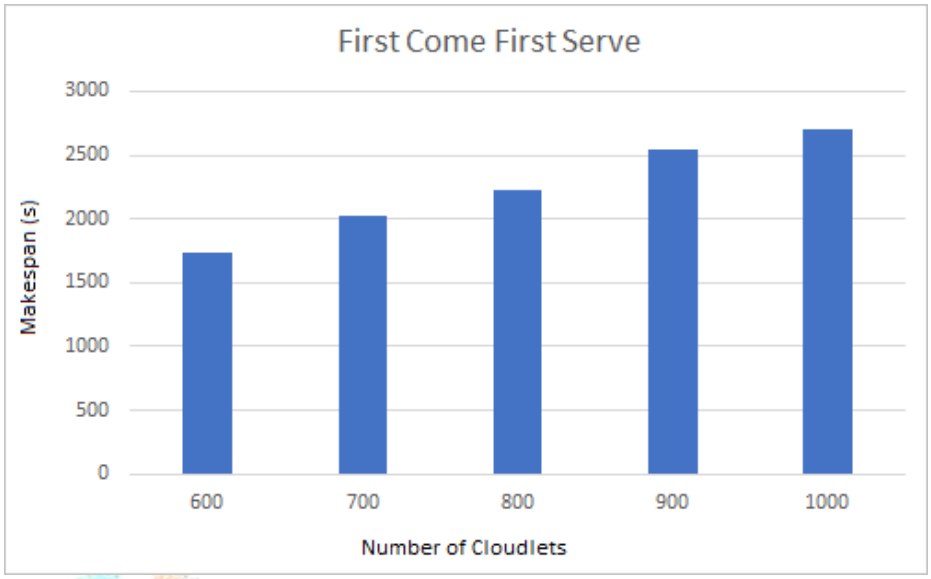


Fig.2: Makespan using First Come first serve with 40 VM's.

Shortest Job First selects the cloudlet with the smallest length from the cloudlets submitted to broker. There is no relation between the cloudlets size and the VM's configuration. The task is assigned to the idle VM on the basis of shortest length. It can be implemented as round robin in which the tasks are assigned on the basis of length to the VM's in round robin fashion.

Table 3: Makespan Time (in seconds) using Shortest Job First with 40 VM's

Scheduling Algorithm	No. of Cloudlets				
	600	700	800	900	1000
Shortest Job First	1437	1687	1914	2151	2337

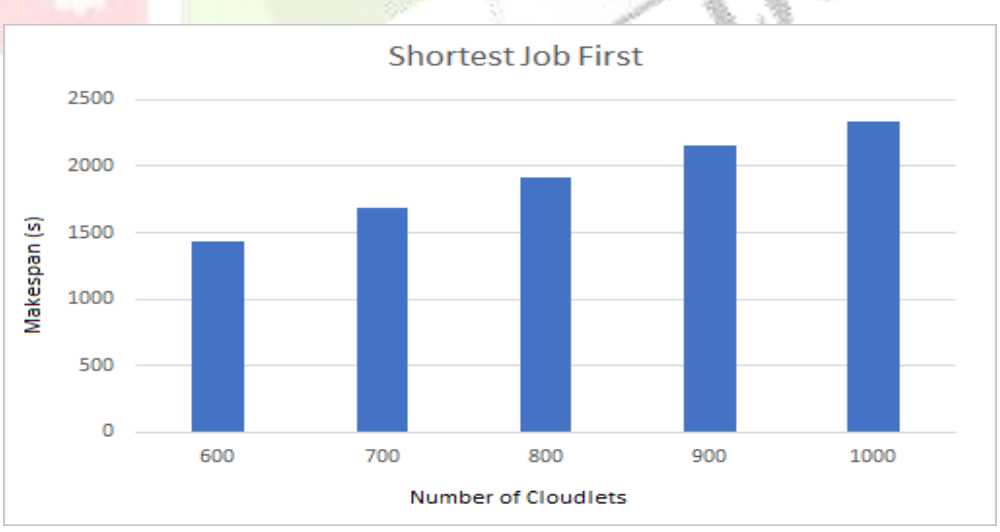


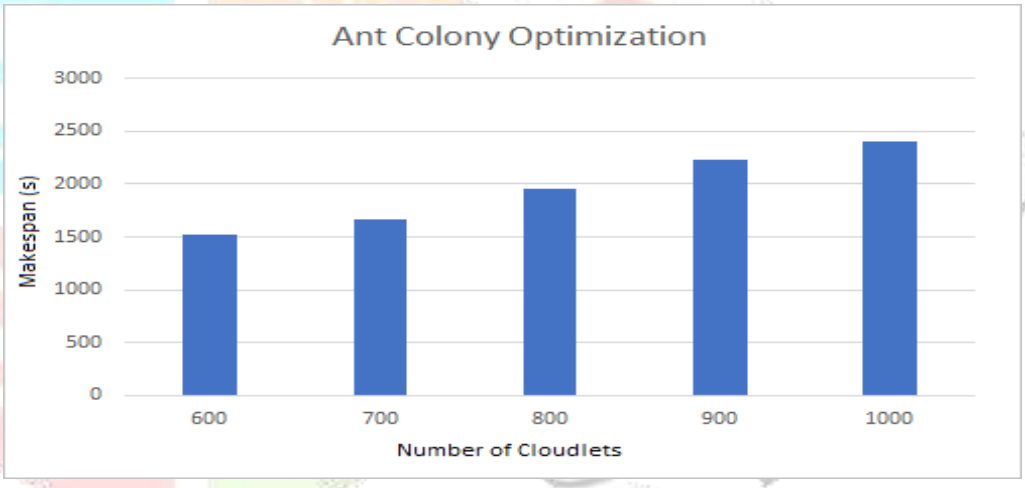
Fig.3: Makespan using Shortest Job First with 40 VM's.

Third Scheduling algorithm implemented is Ant Colony Optimization. The cloudlets are assigned to VM's on the basis of probabilistic function. There is a direct relation between the expected time of execution for each cloudlet corresponding to every VM. Ant Colony Optimization (ACO) comes under the category of metaheuristic algorithms. The algorithm is based on real ants and how they search for food. The ants travel from their colony to the foodsource. The ants leave pheromones as they walk. Initially random ants select random paths. The pheromone they leave also evaporates but at lesser intensity. So, the shortest path after some time is the one with highest pheromone intensity which leads all other ants to follow that path. After a certain period, all ants choose that path and which happens to be the shortest path [45]. Ants during trips select the best VM for each cloudlet. At the end of maximum number of iterations, the global best is selected as final allocation of cloudlets to VM. All antsperform a trip and compute a solution. In the end you have multiple solutions available.

Fig.4: Makespan using Ant Colony Optimization with 40 VM's.

Table 4: Makespan Time (in seconds) using Ant Colony Optimization with 40 VM's

Scheduling Algorithm	No. of Cloudlets				
	600	700	800	900	1000
Ant Colony Optimization	1530	1672	1963	2233	2398



If you observe the table above, Ant colony optimization fails to reduce the Makespan time ascompared to shortest job first. The main reason can be that shortest job first algorithm for static implementation guaranteed to provide the optimized result. Even though the shortest job first algorithm results in better Makespan time, the same cannot be used in practical scenario. Comparison in the performance of 3 algorithms keeping the number of VM's fixed can be analysed from the table & chart below:

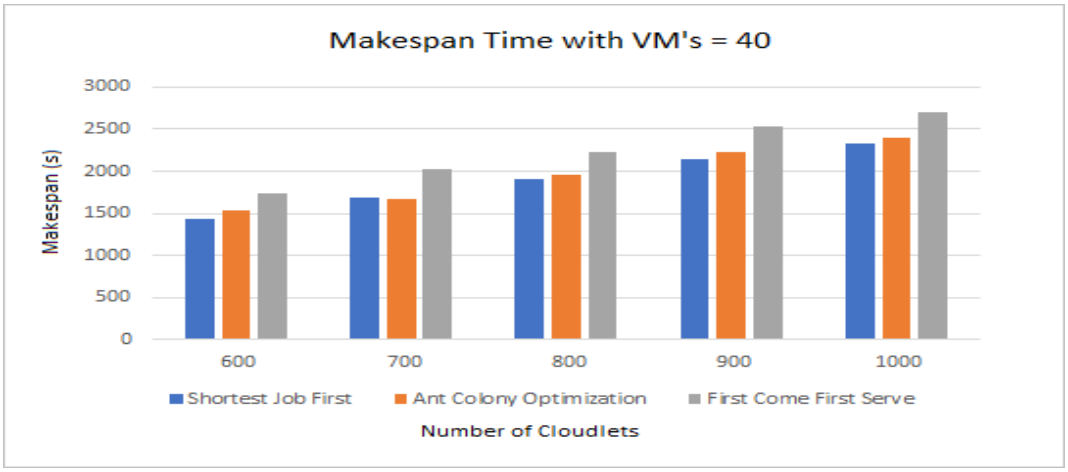


Figure 5: Performance comparison between First Come first serve, Shortest Job First andAnt Colony Optimization in terms of Makespan Time with 40 VM's.

Table 5: Performance comparison in terms of Makespan Time (in seconds) between First Come First Serve, Shortest Job First & Ant Colony Optimization with40 VM's

Scheduling Algorithm	No. of Cloudlets				
	600	700	800	900	1000
First Come First Serve	1730	2021	2233	2541	2702
Shortest Job First	1437	1687	1914	2151	2337
Ant Colony Optimization	1530	1672	1963	2233	2398

Keeping the Cloudlets Fixed

In the second phase, number of cloudlets were fixed to 1000. The experiments were conducted by changing the number of VM's from 20 to 40.

Following is the chart for Makespan time using First Comer First Serve scheduling algorithmkeeping the number of cloudlets fixed to 1000:

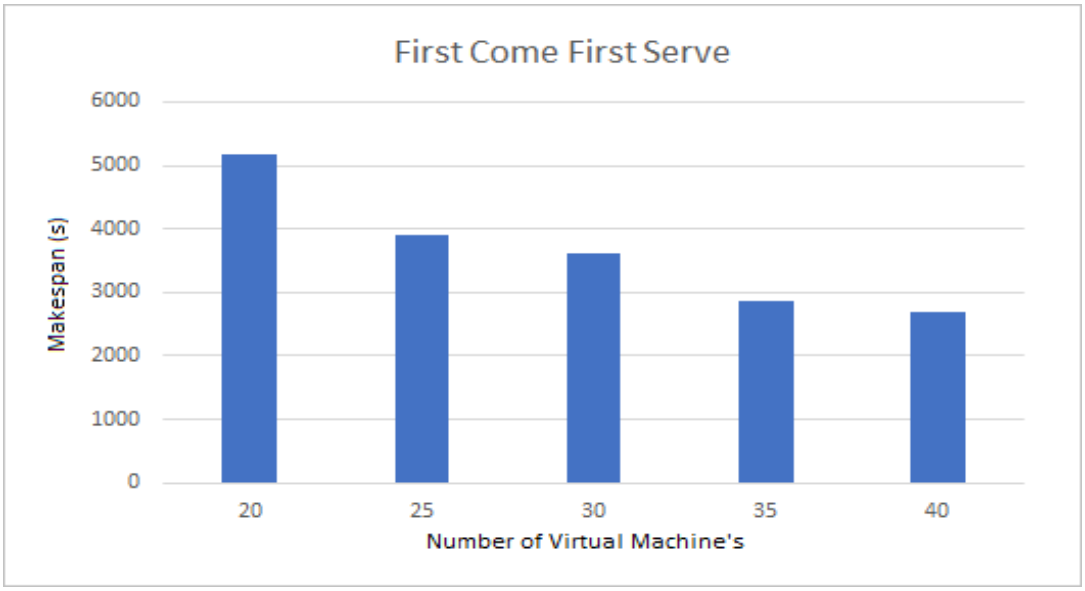


Fig.6: Makespan using First Come first serve with 1000 Cloudlets.

Table 6: Makespan Time (in seconds) using First Come First Serve with 1000 cloudlets

Scheduling Algorithm	No. of VM's				
	20	25	30	35	40
First Come First Serve	5177	3895	3624	2876	2703

Shortest job first is based on selection of task with minimum instructions. Following is the chart for Makespan time using Shortest Job First scheduling algorithm keeping the number ofcloudlets fixed to 1000:

Table 7: Makespan Time (in seconds) using Shortest Job First with 1000 Cloudlets

Scheduling Algorithm	No. of VM's				
	20	25	30	35	40
Shortest Job First	4661	3712	3097	2686	2326

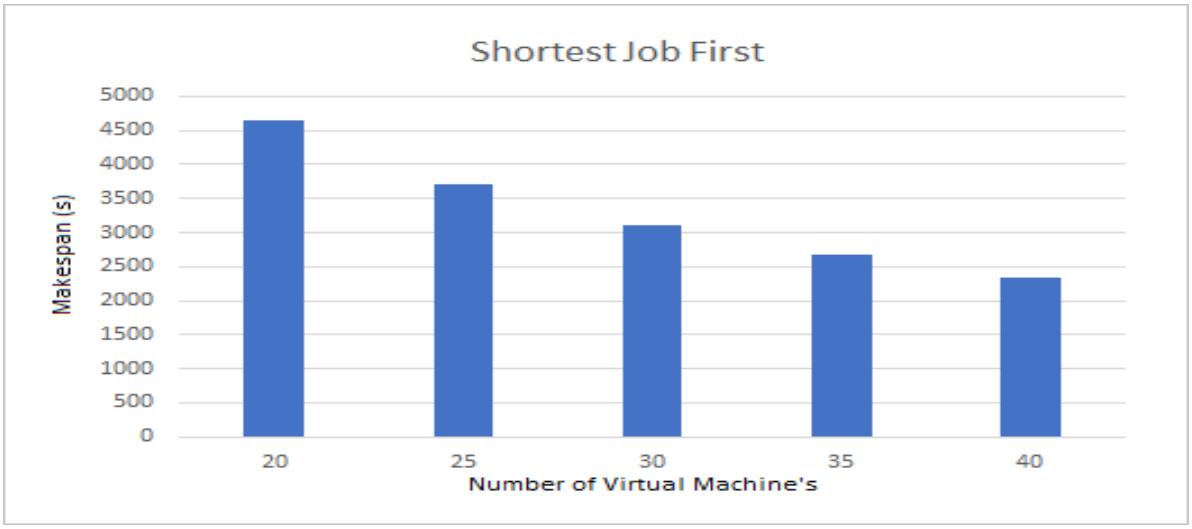


Fig. 6: Makespan using Shortest Job First with 1000 cloudlets.

Following is the chart for Makespan time using Ant Colony Optimization algorithm keeping the number of cloudlets fixed to 1000:

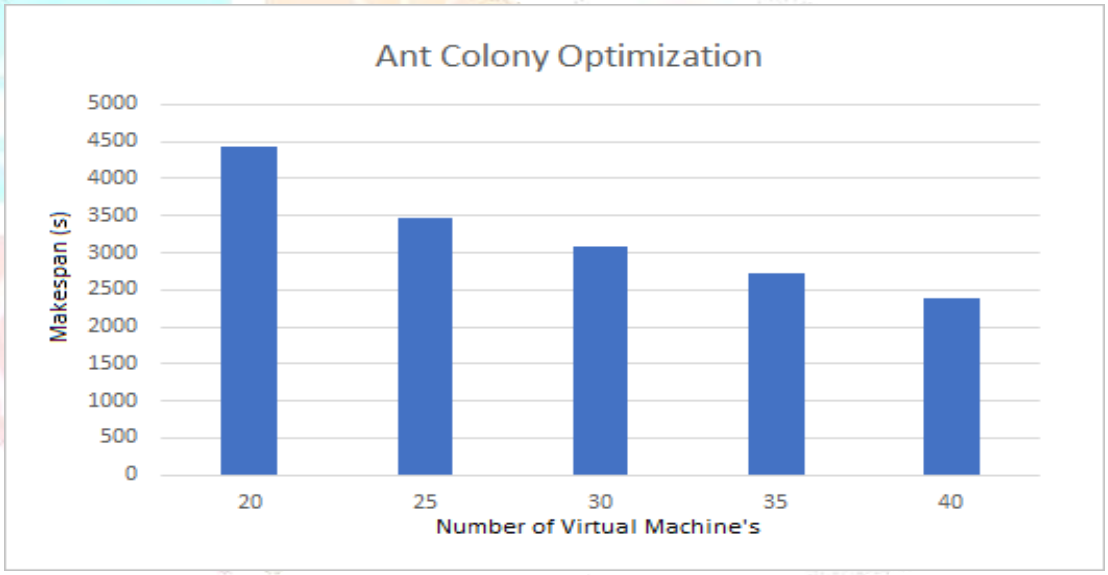


Fig.7: Makespan using Ant Colony Optimization with 1000 cloudlets.

Table 8: Makespan Time (in seconds) using Ant Colony Optimization with 1000Cloudlets

Scheduling Algorithm	No. of VM's				
	20	25	30	35	40
Ant Colony Optimization	4430	3473	3095	2735	2387

Comparison in the performance of 3 algorithms keeping the number of cloudlets fixed can beanalysed from the table & chart below:

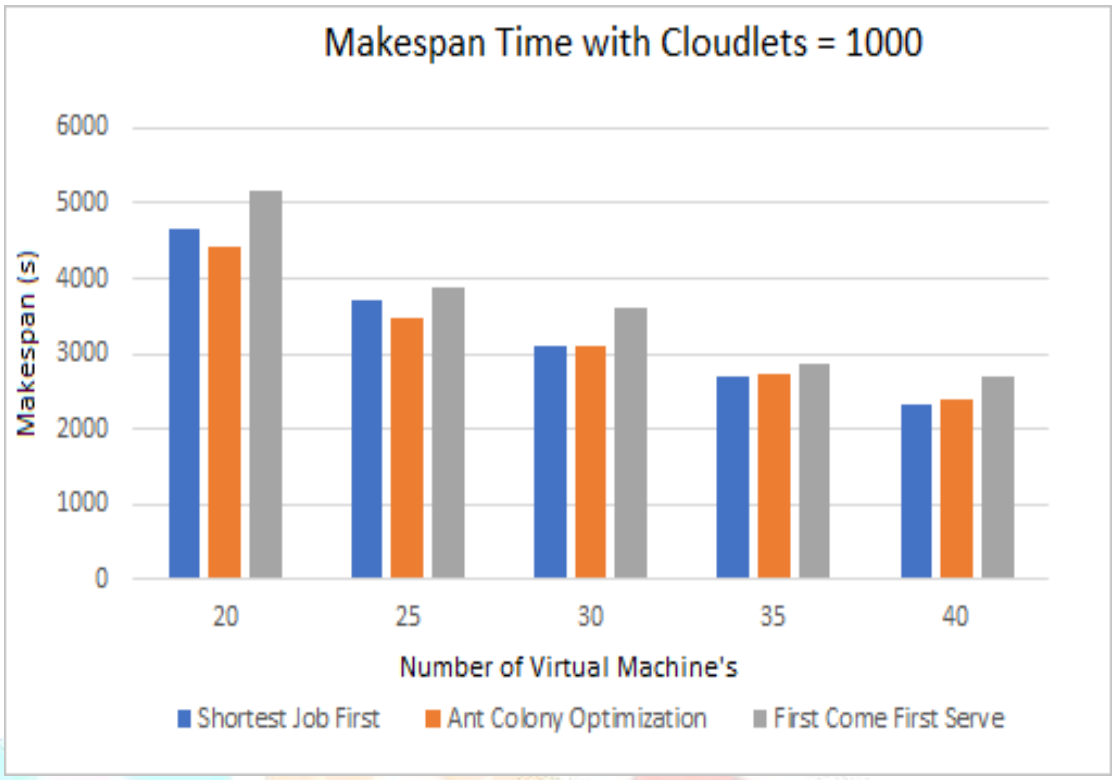


Fig.8: Performance comparison between First Come first serve, Shortest Job First andAnt Colony Optimization in terms of Makespan Time (in seconds) with 1000 cloudlets.

The Makespan time of shortest job first and ant colony optimization is almost same. There isvery little to choose between the two algorithms. The only difference is in terms of their applicability. The Ant colony optimization can be used in real life scenario with complex problems where shortest job first can be used in offline scenario for small problems.

Table 9: Performance comparison in terms of Makespan Time (in seconds) between First Come First Serve, Shortest Job First & Ant Colony Optimization with 1000 cloudlets

Scheduling Algorithm	No. of VM's				
	20	25	30	35	40
First ComeFirst Serve	5177	3895	3624	2876	2703
Shortest JobFirst	4661	3712	3097	2686	2326
Ant Colony Optimization	4430	3473	3095	2735	2387

It can be analyzed that SJF performs as good as ACO in terms of Makespan time. But SJF is practically not implementable in real life dynamic environment. SJF generally results in starvation of jobs with higher expected time to compute. There" is a need to look beyond algorithms like SJF when it comes to implementing task scheduling in real time environment. There is not much of a gap in the performance of ACO and SJF.

But, this gap in the Makespan widens as you increase the number of cloudlets keeping the VMs constant or when you decrease the VMs keeping the cloudlets" constant. ACO tends to produce better results when the size of the cloudlets become larger and larger. Implementation of ACO depends greatly on the probability function. In this implementation, the probability function only focused on the Makespan time. It is possible to design the probability function of ACO by considering VM parameters like bandwidth, RAM, storage.

Shortest job first can also be used, but in situations when the information related to all incoming tasks is known in prior and the number of cloudlets to be scheduled is small in number. The algorithm does not guarantee best results as capability of the VM is not considered while scheduling tasks. In further experiments, performance comparison of different metaheuristic scheduling algorithms has been done. These scheduling algorithms are Ant Colony Optimization and Particle Swarm Optimization.

Performance comparison of Ant Based Scheduling Algorithms (Autonomous Agent Based Load Balancing Algorithm & Ant Colony Optimization Algorithm)

Autonomous Agent based Load Balancing (A2LB) algorithm was proposed by A. Singh et al. in 2015. Autonomous agent-based load balancing algorithm (A2LB) tries to address the issues like optimizing resource utilization, improving throughput, minimizing response time, dynamic resource scheduling with scalability and reliability. A2LB works by distributing the resources in such a manner that the available resources are utilized in a proper manner and load at all the virtual machines remain balanced. A2LB mechanism comprises of three agents: Load agent, Channel gent and Migration Agent. Load and channel agents are static agents whereas migration agent is an ant. The reason behind deploying ants is their ability to choose shortest/best path to their destination. Ant agents are motivated from biological ants which seek a path from their colonies to the food source

Basic difference between the two ant-based algorithms is that the ACO works on more constraints and looks to find the best virtual machine for a particular cloudlet. Lots of computations are performed. Whereas in case of A2LB, cloudlet is initially assigned randomly to any virtual machine and only in case if it becomes unbalanced then the cloudlet is migrated to some other machine. It may seem easy and not much computation is done, but once the virtual machines are loaded to the fullest, the cloudlets are either always migrated from one machine to other or they keep waiting for a virtual machine to be assigned to them.\Does A2LB perform better than ACO? To find answer to this question, a comparative analysis of two algorithms is done. Difference between A2LB and ACO.

Both are Ant Based Scheduling Algorithms The experiments are implemented with 3 Data Centers with 2 hosts each. 12VMs are created with 4 VMs per DC. Experiment was done with 200-500 tasks under the simulation platform. The length of the task is from 20000 Million Instructions (MI) to 400000 MI. The parameters setting of cloud simulator are shown in Table 17.

The Experiment is conducted by keeping the number of VM's fixed. The arrival time for each task is considered to be 0 for the computation of response time. Hence the Execution Start Time of each cloudlet is its response time. Average response time is computer in the end by dividing the overall response time with the total number of tasks. Makespan is computed as the Finish Time of the last cloudlet.

Table 10: Parameters Setting of Cloudsim for performance comparison of Ant Colony Optimization algorithm & Autonomous Agent Based Load Balancing Algorithm

Following is the outcome of the experiment.

Entity Type	Parameters	Values
Task(Cloudlet)	Length of Task	20000-400000
	Total Number of Task	200-500
Virtual Machine	Total Number of VMs	12
	MIPS	256-512
	VM Memory (RAM)	1024
	Bandwidth	500
	Cloudlet Scheduler	Time_shared and Space_shared
	Number of PEs Requirement	1-2
Data Centre	Number of Datacenter	3
	Number of Host	2
	VM Scheduler	Time_shared and Space_shared

Response Time keeping the VM's fixed to 12

Table: 11.Response Time (in seconds) using Ant Colony Optimization with 12 VM's

Scheduling Algorithm	No. of Cloudlets			
	200	300	400	500
Ant Colony Optimization	156.5	166.1	172	200.6

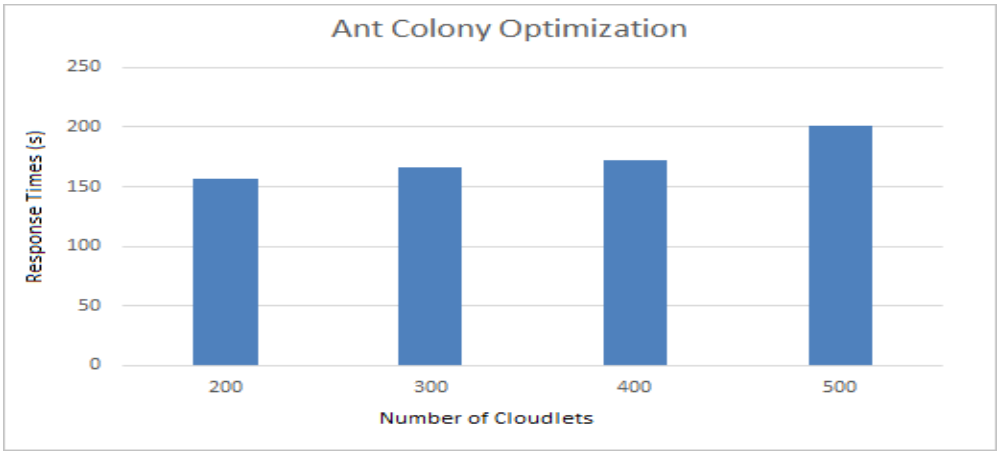


Fig.9: Response time using Ant Colony Optimization with 12 VM's.

Table 12: Response Time (in seconds) using Autonomous Agent Based Load Balancingalgorithm with 12 VM's

Scheduling Algorithm	No. of Cloudlets			
	200	300	400	500
Autonomous Agent Based Load Balancing	157.5	165.2	192.9	191.28

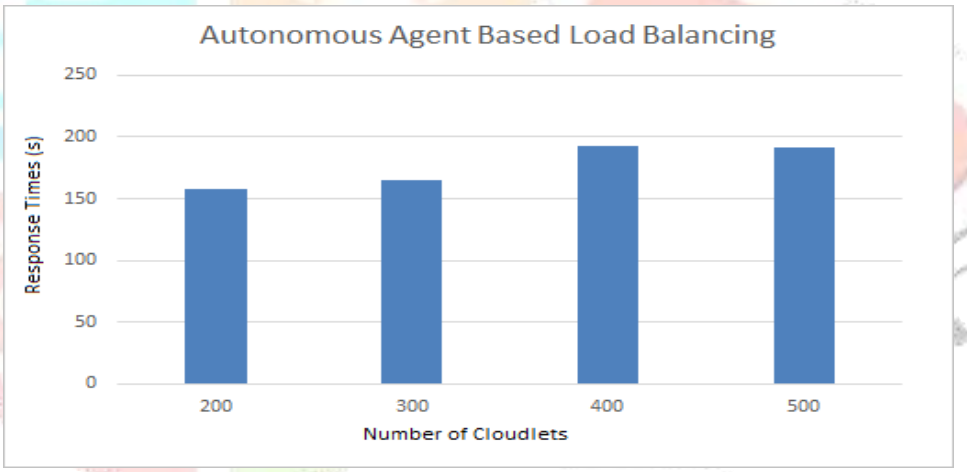


Fig.10: Response Time using Autonomous Agent Based Load Balancing algorithmwith 12 VM's

Comparison in the performance of 2 algorithms in terms of response time keeping the numberof VM's fixed can be analysed from the table & chart below:

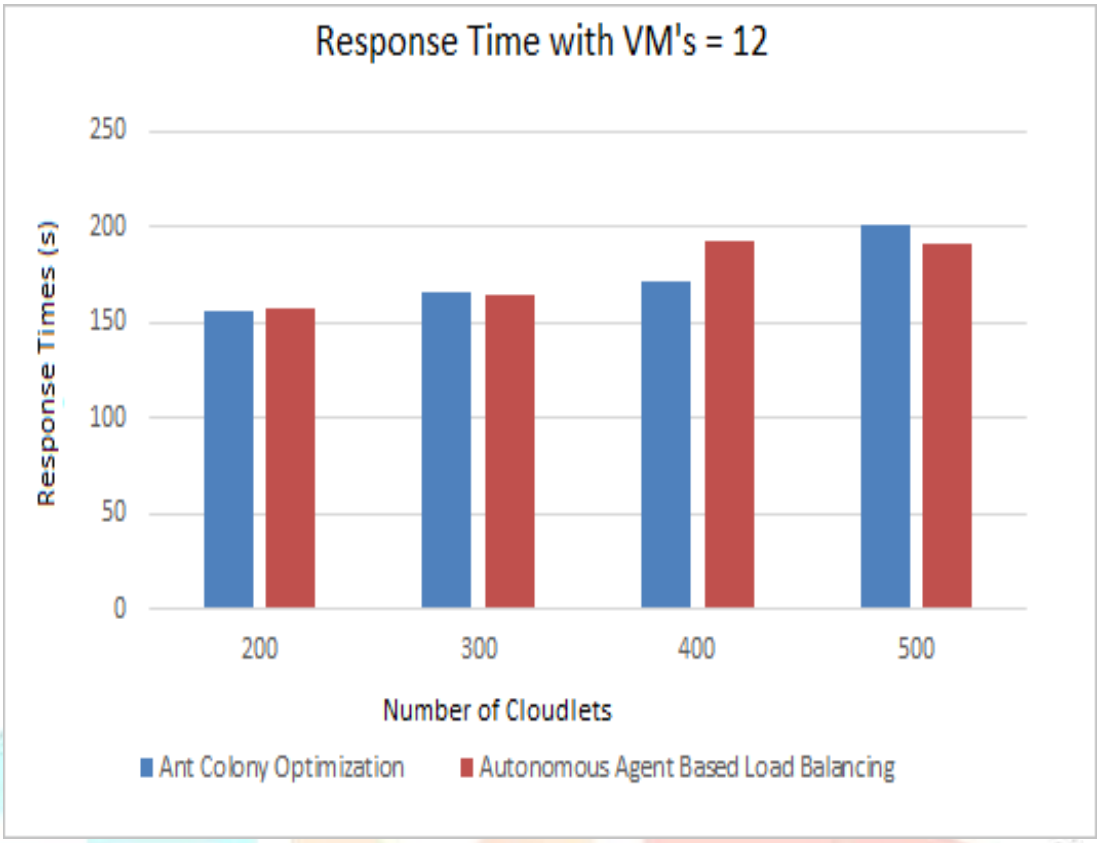


Fig.11: Performance comparison in terms of Response Time between Ant Colony Optimization & Autonomous Agent Based Load Balancing algorithm with 12VM's

Table 13: Performance comparison in terms of Response Time (in seconds) between Ant Colony Optimization & Autonomous Agent Based Load Balancing algorithm with 12 VM's

Scheduling Algorithm	No. of Cloudlets			
	200	300	400	500
Ant Colony Optimization	156.5	166.1	172	200.6
AutonomousAgent Based Load Balancing	157.5	165.2	192.9	191.28

As you can see in Figure 31, the Response Time for both the algorithms is almost same. As the number of cloudlets increase, there is some variation in the results. It was also observed that the performance of A2LB varied quite a lot as the performance greatly depends on the initial random allocation of resources to the tasks.

Makespan Time keeping the VM's fixed to 12

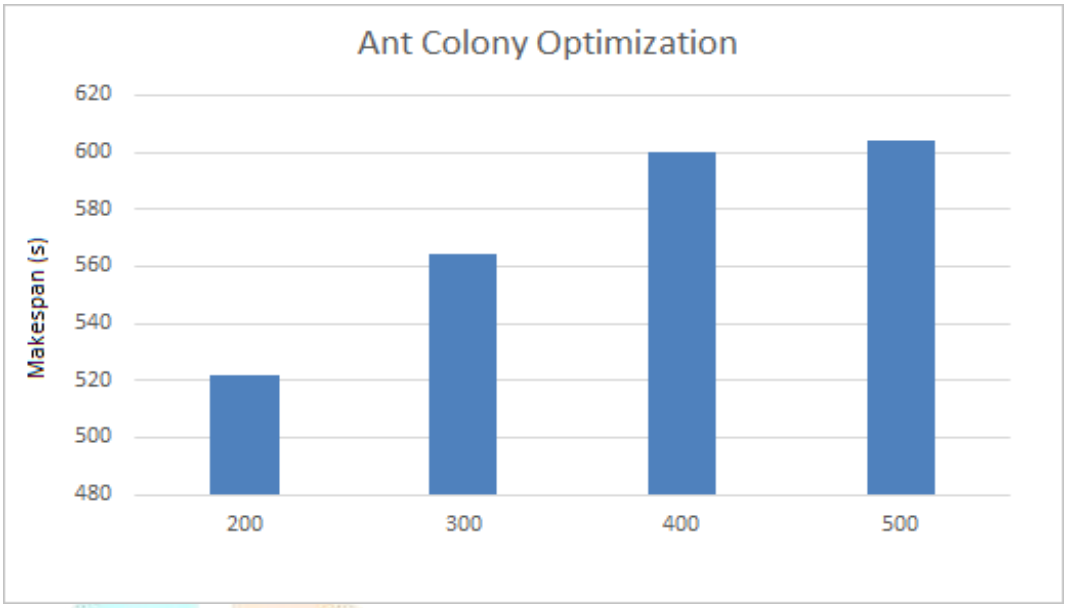


Fig.12: Makespan time using Ant Colony Optimization with 12 VM's.

Table 14: Makespan Time (in seconds) using Ant Colony Optimization with 12 VM's

Scheduling Algorithm	No. of Cloudlets			
	200	300	400	500
Ant Colony Optimization	522	564.1	600	604.4

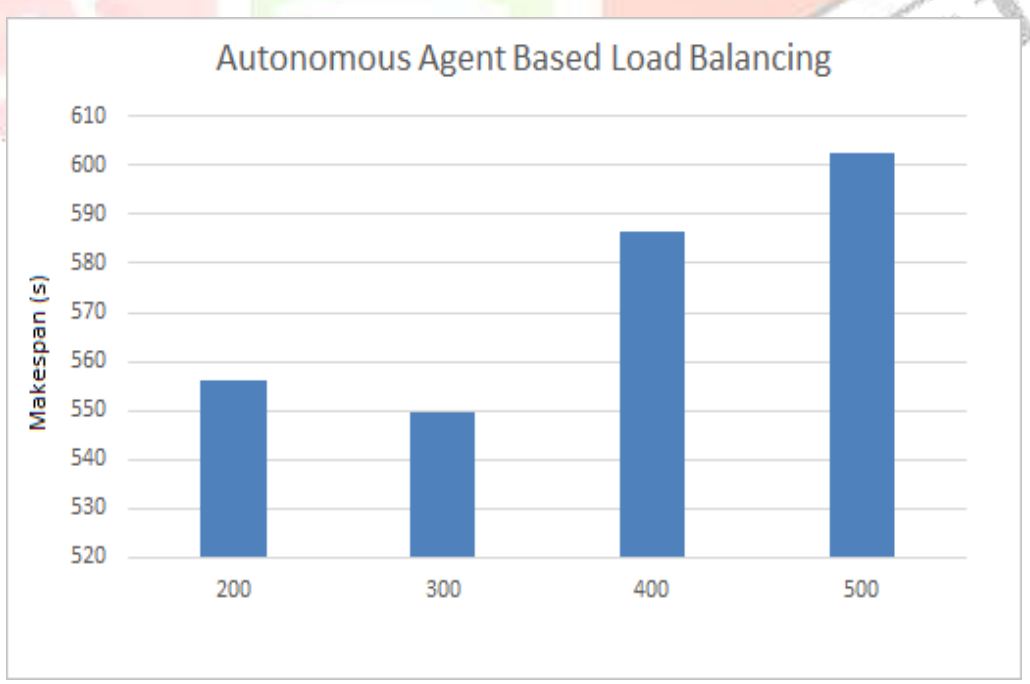


Fig.13: Makespan Time using Autonomous Agent Based Load Balancing algorithmwith 12 VM's

Table 15 Makespan Time (in seconds) using Autonomous Agent Based LoadBalancing algorithm with 12 VM's

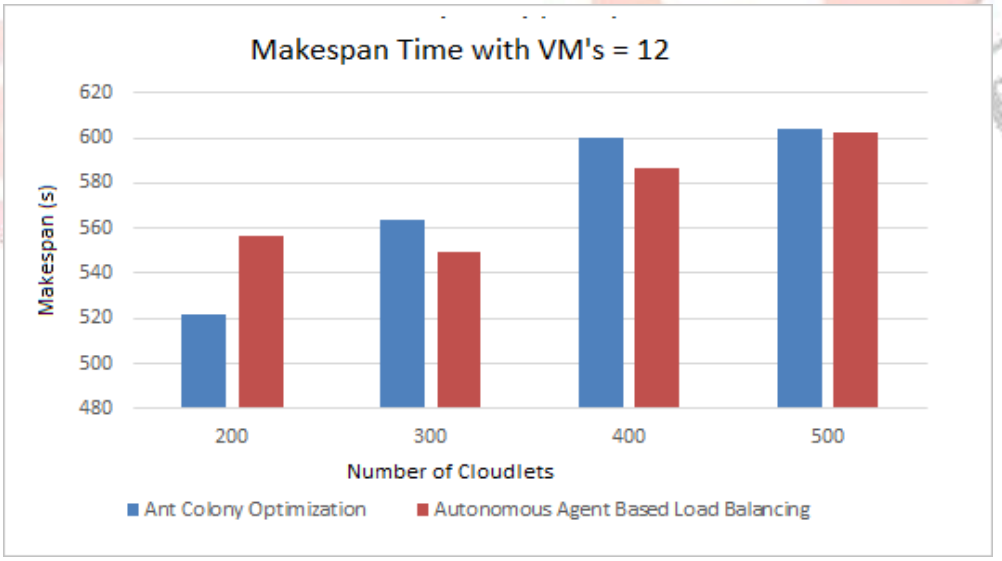
Scheduling Algorithm	No. of Cloudlets			
	200	300	400	500
Autonomous Agent Based Load Balancing	556.3	549.5	586.7	602.5

Comparison in the performance of 2 algorithms in terms of Makespan time keeping thenumber of VM's fixed can be analyzed from the table & chart below:

Table:15 Performance comparison in terms of Makespan Time (in seconds) betweenAnt Colony Optimization & Autonomous Agent Based Load Balancing algorithm with 12 VM's

Scheduling Algorithm	No. of Cloudlets			
	200	300	400	500
Ant Colony Optimization	522	564.1	600	604.4
Autonomous Agent Based Load Balancing	556.3	549.5	586.7	602.5

Fig.14: Performance comparison in terms of Makespan Time (in seconds) betweenAnt Colony Optimization &



Autonomous Agent-Based Load Balancing algorithm with 12 VM's

As you can see in Figure 34, the performance of ACO is quite steady, whereas the performance of A2LB is unpredictable. Performance of both the algorithms is almost same even as your increase the cloudlets to 500.

Performance of both the ant-based algorithms in term of response time and Makespan time is quite similar. But the performance of A2LB greatly depends on how the initial allocation of resources to tasks is done. Although A2LB seems to perform slightly better, but it is more time-consuming

algorithm and it involves migration of tasks. ACO in general does not perform any migration, whereas A2LB migrates the tasks from the overloaded VM to other VM. Also, the A2LB is a resource aware scheduling algorithm, whereas ACO only considersthe Expected Time to Compute (ETC) for allocation the resources to tasks. Performance of ACO can be further improved by adding resource utilization parameters along with the ETC in its probability function.

Performance comparison of Ant Colony Optimization Scheduling Algorithm, Particle Swarm Optimization Scheduling Algorithm & Genetic Algorithm

Table 16: Parameters Setting of Cloudsim for performance comparison of Ant Colony Optimization, Particle Swarm Optimization & Genetic Algorithm

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

The 3 algorithms are different from each other in terms of their applicability, benefits and challenges. GA belongs to the class of evolutionary algorithms, whereas ACO & PSO belong to the class of swarm optimization algorithms. Experiments were conducted to compare the performance of 3 algorithms in terms of makespan time. Following are the results of the experiments.

Table 17: Makespan Time (in seconds) using Ant Colony Optimization with 8 VM's

Scheduling Algorithm	No. of Cloudlets				
	150	175	200	225	250
Ant Colony Optimization (ACO)	270	309	325	355	403

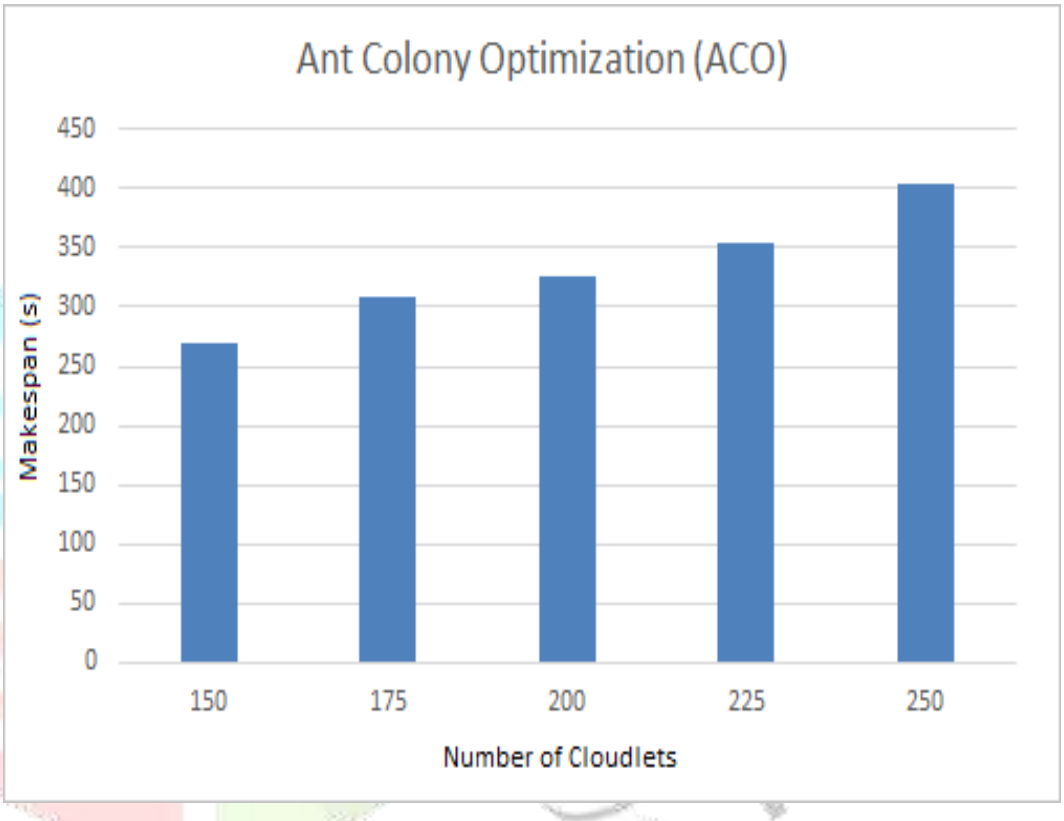


Fig.15: Makespan time using Ant Colony Optimization with 8 VM's.

CONCLUSION & FUTURE WORK

Cloud computing offers efficient and affordable on-demand services and resources, utilizing virtualized resources. It's an extension of existing technologies like parallel and grid computing, providing services over the internet through an infrastructure and policies. Cloud task scheduling is a crucial research area, aiming to reduce resource underutilization and ensure timely task completion. Load balancing assigns equal work to machines based on their capabilities. For example, minimize cost, maximize profit, minimize execution time, maximize throughput. Metaheuristic algorithms are used to optimize task scheduling on cloud resources, but their results are only near-optimal in time, indicating potential for improvement.

In this Thesis 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 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 total execution time of each task on a resource individually on the basis of two parameters. 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 proposedin this thesis work only. 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 this thesis work only. The proposedalgorithm 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 this thesis work only.

REFERENCES

[1] Joly, M. M., Verstraete, T., & Paniagua, G. (2014). Integrated multifidelity, multidisciplinary evolutionary design optimization of counterrotating compressors, *Integrated Computer Aided Engineering*, 21(3), 249-261.

[2] Kociecki, M., & Adeli, H. (2014). Two-phase genetic algorithm for topology optimization of freeform steel space-frame roof structures with complex curvatures, *Engineering Applications of Artificial Intelligence*, 32(1), 218–27.

[3] Siddique, N., & Adeli, H. (2013). *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing*. West Sussex, Chichester: Wiley.

[4] Lopez-Rubio, E., Palomo, E. J., & Dominguez E. (2014). Bregman Divergences for Growing Hierarchical Self-Organizing Networks, *International Journal of Neural Networks*, 24(4), 1-20.

[5] Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self organization in biological systems*, New Jersey, NJ: Princeton University Press.

[6] Balusamy, B., Sridhar, J., Damodaran, D., & Krishna P. V. (2015). Bio-inspired algorithms for cloud computing: A Review. *International Journal of Innovative Computing and Applications*, 6(3), 181-202.

[7] Pinto, T. A., Runkler, T. A., & Sousa, J. M. (2005). Wasp swarm optimization of logistic systems.

Adaptive and Natural Computing Algorithms, 264–267.

[8] Goss, S., Aron, S., Deneubourg, J. L., & Pasteels, J. M. (1989). Self-organized shortcuts in the Argentine ant, *Naturwissenschaften (The Science of Nature)*, 76(12),579–581.

[9] Corne, D., Dorigo, M., & Glover, F. (1999). *New Ideas in Optimization*. UK, Maidenhead: McGraw-Hill Ltd.

[10] Dorigo, M., & Gambardella, L. (1997). Ant colony system: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1), 53–66.

[11] Bilchev, G., & Parmee, I. C. (1995). The ant colony metaphor for searching continuous design spaces, Evolutionary Computing (Springer), 993, 25–39.

[12] Kennedy, J., & Eberhart R. (1995). Particle Swarm Optimization*, presented at IEEE International Conference on Neural Networks*, Perth, New York: IEEE.

[13] Dorigo, M. (1992). *Optimization, learning and natural algorithms*, University of Politecnico di Milano, Italy.

[14] Karger, D., Stein, C., & Wein, J. (2010). Scheduling Algorithms. In Atallah, M. J. & Blanton M. (Eds.) *Algorithms and Theory of Computation Handbook: Special Topics and Techniques.* (pp. 20-20), Boca Rotan, BC: Chapman & Hall/CRC.

[15] Himani, Sidhu, H. S. (2014). Comparative Analysis of Scheduling Algorithms of CloudSim in Cloud Computing. *International Journal of Computer Applications*, 97(16), 29-33.

[16] Braun, & Tracy, D. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, *Journal of Parallel and Distributed computing*, 61(6), 810 – 837.

[17] Yu J., Buyya R., Ramamohanarao K. (2008) Workflow Scheduling Algorithms for Grid Computing. In: Xhafa F., Abraham A. (eds) *Metaheuristics for Scheduling in Distributed Computing Environments*. (pp. 173-221), Brelin,Berlin: Springer.

[18] Kalra, M., & Singh, S. (2015). A Review of Metaheuristic Scheduling Technique in Cloud Computing, *Egyptian Informatics Journal (Cairo University)*, 16(30), 275- 295.

[19] Xhafa, F., and Abraham, A. (2010). Computational Models and Heuristic Methods for Grid Scheduling Problems. *Future Generation Computer System*, 26(1), 608-621.

[20] Dorigo, M., Stutzle, T. (2004). *Ant colony optimization*. Cambridge, Cambridge: MIT Press.

[21] Singh, G., & Kaur, A. (2015). Bio Inspired Algorithms: An Efficient Approach for Resource Scheduling in Cloud Computing, *International Journal of Computer Applications*, 116(10), 16-21.

[22] Kousalya, K. (2009). To improve ant algorithm's grid scheduling using local search. *International Journal Cognitive Computation*, 7(1), 47–57.

[23] Wielenga, G. (2014, August 5), *The top 10 NetBeans features according to its users* Retrievd from https://jaxenter.com/netbeans/the-top-10-netbeans-features- according-to-its-users.

[24] Selvaraj, S., & Jaquline, J. (2016), Ant Colony Optimization Algorithm for Scheduling Cloud Tasks, *International Journal of Computer Technology & Applications*, 7(3), 491-494.

[25] Guo, L., Zhao, S., Shen, S., & Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic Algorithm, *Journal of Networks*, 7(1), 547–553.

[26] Zhang, L., Chen, Y., & Sun, R. (2008). A Task Scheduling Algorithm based on PSO for Grid Computing, *International Journal of Computational Intelligence Research,*4(1), 37–43.

[27] Pandey, S., Wu, L., & Buyya, R. (2010). A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, *presented at 24th IEEE International Conference on Advanced Information Networking and Applications*, 20-23 April 2010, Perth, IEEE.

[28] Wu, Z., Ni, Z., Gu, L., & Liu, X. (2010). A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling, *presented at International Conference on Computational Intelligence and Security*, 11-14 Dec. 2010, China, IEEE.

[29] Sossa, M. R., & Buyya, R. (2014). Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2), 222– 235.

[30] Xue, S., & Wu, W. (2012). Scheduling workflow in cloud computing based on hybrid particle swarm algorithm. *Indonesian Journal of Electrical Engineering*, 10(7),1560-1566.

[31] Pooranian, Z., Shojafar, M., Abawajy, J. H., & Abraham, A. (2015). An efficient meta-heuristic algorithm for grid computing. *Journal of Combinatorial Optimization- Springer*, 30(3), 413-434.

[32] Gomathi, B., & Krishnasamy, K. (2013). Task Scheduling Algorithm Based on Hybrid Particle Swarm Optimization in Cloud Computing Environment. *Journal of Theoretical and Applied Information Technology*, 55(1), 33–38.

[33] Izakian H., Tork Ladani B., Zamanifar K., Abraham A. (2009) A Novel Particle Swarm Optimization Approach for Grid Job Scheduling. In: Prasad S.K., Routray S., Khurana R., Sahni S. (eds) Information Systems, Technology and Management. ICISTM 2009. Communications in Computer and Information Science, vol 31. Springer, Berlin, Heidelberg.

[34] Abdi, S., Motamedi, S. A., & Sharifian, S. (2014). Task Scheduling Using Modified iPSO Algorithm In Cloud Computing Environment. *presented at International Conference on Machine Learning, Electrical and Mechanical Engineering*, January8-9, UAE, IEEE.

[35] Clark, T. (2017, December 18). *Search Algorithm Series: (PSO) Particle Swarm Optimization*. Retrieved from https://medium.com/@iamterryclark/swarm-intelli- eb5e46eda0c3.

[36] Eberhart, & Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *presented at Congress on Evolutionary Computation,* May 27-30, South Korea, IEEE.

[37] Singh, A., Juneja, D., & Malhotra, M. (2015). Autonomous Agent Based Load Balancing Algorithm in Cloud Computing. Procedia Computer Science, 45(1), 832 – 841.

[38] Tawfeek, M., El-Sisi, A., Keshk, A., & Torkey, F. (2015). Cloud Task Scheduling Based on Ant Colony Optimization. *The International Arab Journal of Information Technology*, 12(2), 129-137.

[39] Ghribi, C., Hadji, M., & Zeghlache D. (2013). *Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms*, *presented at 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, May 13-16, Delft, IEEE.

[40] Developing Applications with NetBeans IDE. (2013, October 18). Retrieved from https://docs.oracle.com/cd/E40938_01/doc.74/e40142/gs_nbeans.htm.

[41] Thaman, J., & Singh, M. (2016). Current Perspective in Task Scheduling Techniquesin cloud Computing: A Review, *International Journal in Foundations of Computer Science & Technology*, 6(1), 65-85.

[42] Arian, Y., & Levy, Y. (1992). Algorithms for Generalized Round Robin Routing. *Operations Research Letters*, 12(5), 313-319.

[43] Kamalakar, M., & Moulika, T. (2015). A Priority Based Job Scheduling Algorithm in Cloud Computing. *International Journal of Innovative Technologies*, 3(1), 19-21.

[44] Selvarani, S., & Sadhasivam, G. S. (2010). Improved Cost-Based Algorithm for Task Scheduling In Cloud Computing. *presented at IEEE International Conference on Computational Intelligence and Computing Research,* December 28-29*,* Coimbatore,IEEE.

[45] Wang, S. C., Yan, K. Q., Liao, W. P., & Wang, S. S. (2010). Towards a load balancing in a three-level cloud computing network, *presented at 3rd International Conference on Computer Science and Information Technology (ICCSIT),* July 9-11,Chengdu, IEEE.