HYBRID ACO-IWD OPTIMIZATION ALGORITHM FOR MINIMIZING WEIGHTED FLOWTIME IN CLOUD-BASED PARAMETER SWEEP EXPERIMENTS

R. Angel Preethima[1], Margret Johnson[2]

[1]*Student, Computer Science and Engineering, Karunya University, Tamilnadu, India*
[2]*Assistant Professor, Computer Science and Engineering, Karunya University, Tamilnadu, India*

Abstract

Scientists and engineers conduct several experiments by executing the same coding against the various input data, which is achieved by the Parameter Sweep Experiments (PSEs). This may finally results in too many jobs with high computational requirements. Therefore the distributed environments, particularly clouds, are used in-order to fulfill these demands. Since it is an NP-complete problem the job scheduling is much changeling. Now the proposed work is determined by the Cloud scheduler based on the bio-inspired techniques, since it works well in approximating problems with little input. But in existing proposals the job priority is ignored; which in turn it is the important aspect in PSEs because it accelerates the result of the PSE and visualization of scientific clouds. The weighted flow time is minimized with the help of the cloud scheduler based on Ant Colony Optimization (ACO). All matching recourses of the job requirements and the routing information are defined by the Intelligent Water Drops (IWDs) in order to reach the recourses. Among all matching resources of the job the Ant colony optimization is determined as the best resources. The main aim of this approach is to converge to the optimal scheduler faster, minimize the make span of the job, improve load balancing.

Keywords: *Ant Colony Optimization, Intelligent Water Drops, Parameter Sweep Experiments, Weighted Flowtime.*

----------***----------

1. INTRODUCTION

As said by Wang et al., a computing cloud is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing platforms on demand, which could be accessed in a simple and pervasive way *[12]*.

The optimal resource for the incoming jobs is obtained by job scheduling and then execute is the respective resource. Following are the three main phases of the job scheduling. 1. Resource discovery, in this phase the set of resources is selected. 2. System selection in this a single resource is selected from a group of resource which meet the job requirements. 3. Job execution, this is the phase were the job is executed in the selected resource. High performance is achieved by job scheduling in cloud computing system as the resource are distributed sometimes the jobs are executed remotely.

Ant colony optimization uses an evolutionary process in order to converge the solution, hence it is known as the biologically inspired population based search method. ACO follows the real ant colony principle, which is how the ant reaches the food from the nest. In the beginning all the ants from the nest moves out in search of the food. Each and every ant moves in their own path in search of the food. Finally the path which is discovered by the ants is converged and in that the shortest path is determined and also the pheromone content is also high. Normally the ants take a long time in order to find the shortest path. In the same way the artificial ants are used in order to find the optimal resource for the incoming jobs. This is achieved by the pheromone trial and pheromone evaporation factors.

The large computational problems like grid scheduling can be solved by ACO and it is a swam-intelligence based heuristic algorithm the optimal solution is obtained by the help of artificial stigmergy. ACO is selected for job scheduling because it is capable of solving both static and dynamic problems.

Basically in nature the water drops in the river make their path towards the lake, seas or ocean; while travelling they prefer the low soil on its bed to the high soil based on this the Intelligent water drops(IWD) algorithm is derived. Depends on the path the velocity of the water changes while its movement. IWD is defined as the best swam intelligence approaches compared to all other existing approaches based on the amount of soil in the path being traversed.

The optimal solution or the nearly the optimal solution is found for the problems like TSP, n-queen and knapsack problems by the IWD algorithm. It can adapt the changes in

the dynamic environment. Among the entire existing algorithm the time to convergence is better in IWD.

This paper is organized as follows. Section 2 includes a discussion about job scheduling techniques. Section 3 describes ACOIWD algorithm. Section 4 analyses the experimental result and Section 5 gives the conclusion of the paper.

2. RELATED WORKS

Various versions of the ACO algorithm are Ant algorithm, Meta heuristics algorithm, Hybrid ant algorithm, Enhanced ant algorithm and ant colony optimization inspired algorithm. Fidanova S et al [4] used this ant algorithm for task scheduling. The main purpose is to maximize the system utilization. With the available computer resources the high through put is to be achieved hence the task scheduling is performed. The final result obtained is that the good load balancing.

Izakian H et al [8] was the one who proposed the Meta heuristic algorithm. His main aim is that in heterogeneous distributed computing system the make span and the flowtime should be reduced. This proposed heuristic is considered to obtain the best minimized make span.

Sathish K et al [11] developed the enhanced ant algorithm for task scheduling. The main objective of this algorithm is to get the better through put with the controlled cost. This algorithm is also used for the reducing the processing cost and the processing time on executing the job. Hybrid ant algorithm was proposed by Ritchie et al [10] in heterogeneous computing environments. This algorithm was proposed to find the efficient scheduling of the independent computational jobs. The results are obtained by the combination of the ACO algorithm with local and tabu search.

Cristian Mateos et al [3] were the one who used the ant colony optimization inspired algorithm in cloud environment. This usage is based on minimizing the make span and the weighted flowtime. In this there are two levels they are Cloud level or Data center level and VM level. In the case of cloud level the VM is allocated to host. But in the case of the VM level the jobs are scheduled to VM using job priority policy. In parameter sweep environment this algorithm gives the minimum make span.

Hamed Shah-Hosseini [5] [7] applied intelligent water drops algorithm for solving multiple knapsack problem and n-queen problem and optimal or near-optimal solutions were obtained.

3. PROPOSED ACOIWD ALGORITHM

The ACOIWD algorithm is proposed to minimize the weighted flowtime of jobs that must be executed by available resources in cloud server. IWD algorithm will choose a set of resources for each job by matching its requirements with the available resources in virtual machines. These set of resources are given as input to ACO algorithm for finding the best resource for each job. The steps in ACOIWD algorithm are as follows

3.1 VM Allocation using ACO Algorithm

In [3], to solve the problem of load balancing, the AntZ algorithm is implemented in the cloud level logic of the scheduler. The combined idea of how the ants cluster object with the capability of leaving the pheromone trails on their path in order to guide the other ants to accompany them by the other ants. Each ant works independently which in turn represent the VM in order to allocate in the best host. The ants start working only after the VM is created. Each load is initialized and the information's are loaded in the master table. The ant is taken from the pool of the ants when the algorithm is executed already exist only if the ants is associated to the VM. The new ant is created if the VM does not exist in the ant pool. The VM is obtained first by allocating the list of all the suitable hosts. The host is suitable for wandering VM only when it has the amount of processing power, memory and the band width greater or equal.

3.2 ACO-Specific Logic

The ACO- specific logic [3] starts to operate when the working ant is added to the working pool. The ant visit each and every host while it's each iteration and adds that information collected from the hosts to the private load history. Each host maintains a table and the ant updates the information in the information table. The table holds the information of the own load of the individual ants and the information of the other hosts and these are loaded when the other ants visits the hosts. The CPU utilization in the host is denoted by the load.

There are two choices when the ant moves from one host to another host.

- Using constant probability or the mutation rate one choice is moved to the random hosts.
- The other choice is using the table information of the current host.

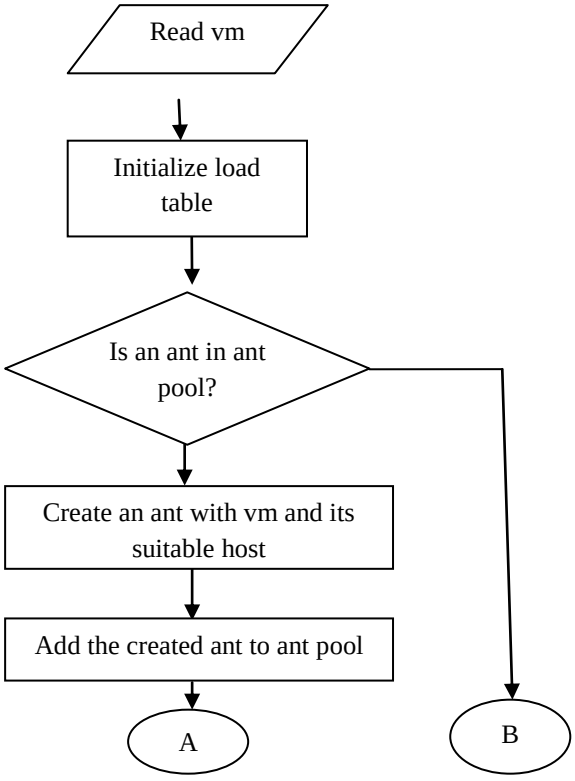
As time passes the mutation rate decreases with the decay rate factor, hence the ant will be more concentrated on the load information than the random choice. These steps are repeated until the finishing criteria are achieved. The completion criteria is give as a predefined number of steps means max steps or the number of iteration. Finally the task is completed. When every steps are completed the ants delivers its VM to current host and finishes its task. When a single ant completes its task it cannot allocate its associated VM because it rotates its VM with all the other ants in order to complete the task. When all the ants cannot complete its task then the first module is repeated until the entire ant completes its task and

achieves the final state. The ant moves from one host to another host and during this process the ant updates the host loaded information into the host where it travels. Not only it updates the information of the table it also carries load table information with it; hence it may be useful for all the other ants to guide in a better path rather than wandering in the cloud. The VM to the host can be allocated based on the load information. The load information has the details of the host. These details are updated by means of the ants when the ant reaches the host while it is travelling on its path.

3.3 ACO-Specific Logic: Choose Next Light Loaded Host

In [3], the ant chooses the lightest load in the host by reading the information of the load table in the each host. That is the current load of the host visited is compared and executed with the entry of the load information. When the visited host load is smaller than the other host in the table, then the ant chooses the lightest load host, if there are no such possibilities the ant chooses the one with the equal probability.

The load is calculated by receiving the number of the jobs that are executing in the recourses. But in case of the proposed algorithm the load is calculated by means of calculating the CPU utilization made by the VM which are executing in the each host. Finally The VM is allocated to the selected host when the best host is selected when the entry load is less than the current load.



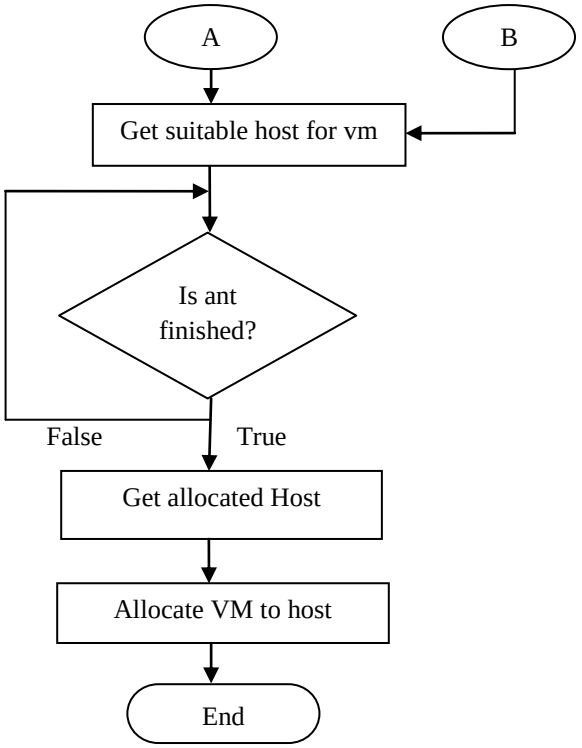


Fig -1: Flow Chart for VM allocation to hosts

Fig-1 shows the allocation of VM to host. Initially, an ant is created for each VM if ant is not in the ant pool. Each ant allocates the VM to host based on the host's load information. The process is repeated, until every ant finishes their work.

3.4 ACOIWD Algorithm

Intelligent Water Drops (IWD) [6] is a technique performs the action as the water flows into the river, ocean and seas by finding their path to flow. This IWD algorithm will be executed on the bases of finding the path with low soils on its bed to the path with high soils. The velocity of the water drops varies according to the path it flows i.e. the soil in the path is inversely proportional to the velocity. In this the soil from the path is taken and deposited in the path where the velocity of the water is low. Therefore both the deposition and the removal of soil take place. All these steps are required in order to select the shortest path being traversed to the destination from the many possible paths. The IWD has the better convergence speed than the any other existing swam algorithm. As mentioned earlier the amount of velocity depends on the path the water travel to reach the host.

The main aim of this algorithm is that to minimize the time take for each job to be executed and processed with the available recourses in the grid computing. Depending on the resources the algorithm will select matching job requirement with resources. In order to find the best resources for each incoming jobs the ACO is applied.

There may be same job requirements for one or more jobs and thus matching with same MDS. In that situation the priority is given to the job which has the high priority or the job which arrived earlier. The grid system is represented by the graph which comprises of the MDS and the grid nodes within each MDS. The IWD checks for any matching MDS with the job requirements while it travels from the source node. And then it travels to the destination. IWD works based on the following steps[11].

1. Initialize static parameters and dynamic parameters:
Static parameters:

For velocity updating, parameters are a_v =1, b_v = 0.01, c_v = 1

For soil updating, parameters are a_s = 1, b_s = 0.01, c_s =1. InitSoil = 10000 and InitVel = 200.

Dynamic parameters:

Visited node list of each IWD, v_c(IWD) is set as empty initially.

2. Repeat steps 2.1 to 2.4 for each IWD.
2.1 IWD chooses the next node based on probability,

$$p(i,j) = \frac{f(soil(i,j))}{\sum_{k \notin v_c} f(soil(i,j))} \quad (1)$$

Such that,

$$f\big(soil(i,j)\big) = \frac{1}{\epsilon_s + g(soil(i,j))}$$

And

$$g\big(soil(i,j)\big) = \begin{cases} soil(i,j) & \text{if } \min\big(soil(i,j)\big) > 0 \\ soil(i,j) - \min\big(soil(i,j)\big) & elsewhere 1 \notin v_c(IWD) \end{cases}$$

2.2 Update the velocity of IWD after it moves from node *i* to *j*

$$vel(t+1) = vel(t) + \frac{a_v}{b_v + c_v \,.\, soil(i,j)} \quad (2)$$

2.3 Compute the soil that IWD taken from the path it travels, Δsoil (i,j)

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s time^2(i,j;vel(t+1))} \quad (3)$$

Such that, $time\big(i,j;vel(t+1)\big) = \frac{HUD(j)}{vel(t+1)}$ where HUD (*j*) is the distance from the current node to the next node

2.4 Update the soil in the path where the IWD is traversed using,

$$soil(i,j) = (1-\rho_n).soil(i,j) - \rho_n.\Delta soil(i,j) \quad (4)$$

$$soil^{IWD} = soil^{IWD} + \Delta soil\ (i,j)$$

3. Find the iteration best solution among all the solutions.
4. Update the soils on the paths where the iteration best solution is found by,

$$soil(i,j) = (1-\rho_{IWD}).soil(i,j) - \rho_{IWD}.\frac{1}{(N_{IB}-1)}.soil_{IB}^{IWD} \quad (5)$$

Where, N_{IB} is the number of nodes in the solution.

5. Increment the iteration and go to step 2 until the maximum number of iteration is reached.

ACO is used in order to find the optimal solution from all the matched resources.

IWD takes number of jobs as input and finds the matching resources based on job requirements. The matching resources are given to ACO for choosing best resource for executing each job and it is shown in Fig-2. Hence, it reduces processing element wastage because of minimum flowtime.

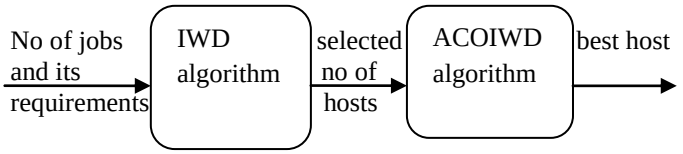


Fig -2: Architecture for proposed ACOIWD algorithm

4. EXPERIMENTAL RESULTS

Using the cloudsim toolkit the proposed ACOIWD approach is simulated. The ACOIWD scheduler process the job which is submitted in the cloud environment and gives the best resources for the job.

Table -1: Comparison Of Weighted Flowtime Between Existing ACO Technique And Proposed ACOIWD Technique

S.No	Number Of Jobs	Existing ACO Algorithm(msec)	Proposed ACOIWD Algorithm(msec)
1	90	872699.854	205589.014
2	100	1248409.37	584415.790
3	110	1636978.056	677156.205
4	120	1644036.189	863805.621

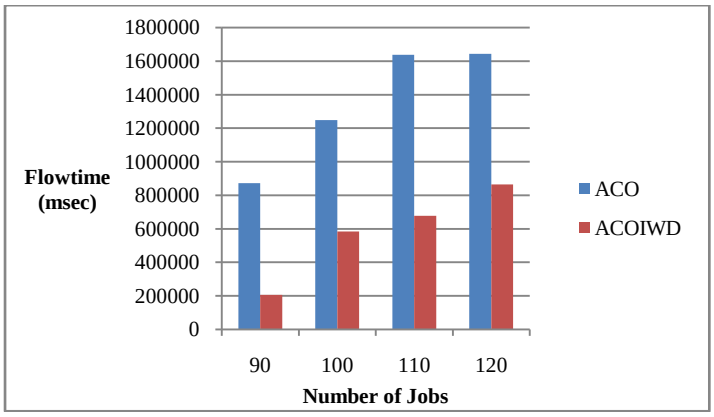


Chart -1: Graph shows the difference between the proposed ACOIWD techniques and the existing ACO techniques.

Table-I show that the flowtime values of jobs when executing existing ACO and proposed ACOIWD algorithm in cloudsim. These values are plotted in graph which is shown in Chart-1.From chart we can understand that the proposed ACOIWD algorithm gives minimum flowtime of jobs when compared to existing ACO algorithm. By obtaining minimum flowtime, we can reduce processing element wastage.

5. CONCLUSIONS

This paper describes the new technique of using intelligent water drop with ant colony optimization technique in order to solve the problem of scheduling in the cloud environment. This paper is proposed to find the optimal schedule decision for the jobs faster. The main aim is also to reduce the makespan of the executing job, maximizing the load balancing of the job among the resources in a distributed manner.

ACKNOWLEDGEMENT

The authors would like to acknowledge the School of Computer Science and Technology and the Department of Computer Science for providing us with the opportunity to prepare this report.

REFERENCES

[1]. Buyya R, Yeo C, Venugopal S, Broberg J, Brandic I. (2009) "computing Cloud, et al. and reality for delivering computing as the 5th utility". Future Generation Computing System; 25(6):599–616.

[2]. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. (2011) "CloudSim: a toolkit for modeling and simulation of Cloud Computing environments and evaluation of resource provisioning algorithms". Software: Pract Exp; 41(1):23–50.

[3]. Cristian Mateos , Elina Pacini, Carlos García Garino C. (2013) "An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments". Advances in Engineering Software 56 (2013) 38–50

[4]. Fidanova S, Durchova M. (2005) "Ant algorithm for Grid scheduling problem". In: 5th International conference on large-scale scientific computing". Springer; p. 405–12.

[5]. Hamed Shah-Hosseini. (2008) "Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem". International Journal of Intelligent Computing and Cybernetics, Vol. 1, No. 2, pp. 193-212.

[6]. Hamed Shah-Hosseini. (2009) "Optimization with the Nature-Inspired Intelligent Water Drops Algorithm" InTechOpen, pp.297-320

[7]. Hamed Shah-Hosseini. (2009) "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm". International Journal of Bio-Inspired computation, Vol. 1, Nos. 1/2, pp. 71-79

[8]. Izakian H, Abraham A, Snasel V. (2009) "Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments". International joint conference on computational sciences and optimization", vol.1 Washington, DC, USA: IEEE Computer Society; 2009. p. 8–12.

[9]. P. Mathiyalagan, S. N. Sivanandam and K. S. Saranya(2013) " Hybridization Of Modified Ant Colony Optimization And Intelligent Water Drops Algorithm For Job Scheduling In Computational Grid". ICTACT Journal on Soft Computing, October 2013, Volume: 04, Issue: 01

[10]. Ritchie G, Levine J. (2004) "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments". In: Proceedings of the 23rdworkshop of the UK planning and scheduling special interest group.

[11]. Sathish K, Reddy. (2008) "ARMS Enhanced ant algorithm based load balanced task scheduling in Grid computing". IJCSNS International Journal Computer Science Network Security; 8(10):219–23.

[12]. Wang L, Tao J, Kunze M, Castellanos AC, Kramer D, Karl W. (2008) "Scientific cloud computing: early definition and experience". In: 10th IEEE international conference on high performance computing and communications. Washington, DC, USA: IEEE Computer Society; p. 825–30