An Efficient Ant Colony Optimization Algorithm for Resource Provisioning in Cloud

M. Aliyu, M. Murali, A. Y. Gital, S. Boukari

Abstract: ***Paper Several Ant Colony Optimization (ACO) techniques for Cloud resources management are considered by many researchers. ACO techniques in existence still need some improvements for effective resource management and planning with the heterogeneous and voluminous services offered. Hence, an optimized hybrid scheme that combined deterministic characteristics for exploiting ACO search process is proposed. Spanning Tree (ST) algorithm was chosen in the hybridization that obtained a faster convergence speed, minimized makespan time and throughput that ensured resource utilization in least time and cost. Extensive experiments were conducted in cloudsim simulator provided an efficient result compared to other ACO techniques as it significantly improves performance.***

Keywords: Cloud computing, Deterministic, Metaheusristics Optimization.

I. INTRODUCTION

ACO is a member of the metaheuristic algorithm that follows real ant colony foraging behavior principle. Ant's behavioral means of communication is by through pheromone (chemical substance) that enables them to find shortest path from their nest to food source. Secreted pheromone at the cause of searching their food is used for exchanging information while trailing for the shortest path. Path with the highest concentration of the substance is regarded as the shortest path. ACO has flourished in areas for solving job shop scheduling problem [1], multidimensional knapsack problem [2], Traveling Salesman Problem (TSP) [3], scheduling of tasks in grid and Cloud [4], quadratic assignment problem [5] and many more. The algorithm performs exigently in solving problems having to do with discrete optimization that needs to discover short routes to goals. Despite the plethora adoptions of ACO in such fields, the algorithm still needs some improvements in handling transition loops that leads to longer convergence time, initial population generation and pheromone evaporation. Best performing techniques make intensive use of its optional local search phase and proper initial solutions generation [6].

Researchers have strongly suggests that a solid theoretical framework is required for the development of ACO [6] [7] [8]. Early investigation regarding simplified framework was conducted by Merkle and Middendorf [9] and that of convergence proof by Gutjahr, [10]. Neumann and Witt [11] [12] and Gutjahr [13] independently conducted the first rigorous probing on ACO regarding convergence using simple pseudo-Boolean functions optimization. The algorithm presented by Neumann and Witt [12] memorizes the best solution if found and was termed '1-ANT'. Iterations are performed to obtain pheromone result. Pheromone evaporation parameter is measured according to a scale $0 < P < 1$, where P is pheromone concentration for new-best-solution. Iteration with the least p value is recorded as the best and feasible solution. If the evaporation factor of 1-Ant is too small, the algorithm stagnates on simple problems [14]. To mitigate such occurrence, Stutzle and Hoos [15] reinforce the best solution found in every iteration and termed it as MAX-MIN Ant System (MMAS). Best-so-far update strategy was used to avoid the stagnation resulting to efficient convergence for various problems by Orlin et al. [16]. Sudholt [17], also make use of the reinforcement learning for the best solution created in the current iteration and was termed as 'iteration-best update'.

The concept of hybridizing ACO with local search was first seen in Neumann et al. [18]. The authors demonstrated the use of local search for artificially constructed problems in turning to polynomial using exponential run time with high probability. With convergence problems in ACO, Neumann and Witt [12] showcased an analysis for minimum SPT. The authors also provided a guideline for the construction process on the impact of heuristic information. To our knowledge, this is the first application of ACO and SPT for eliminating loops in order to obtain faster convergence. Numerous researchers recently proposed several hybrid ACO algorithms [19] [20] [21] [22] in cloud task scheduling. Some of these techniques scale well in terms of convergence, throughput, makespan, execution time, and waiting time. These algorithms do still necessitate some betterment in order to cope with the actual resources allocation in Cloud. The aim of this research work is to design a framework for an improved ACO principle with ST (ACOST) in task scheduling. A loop free network is targeted to be achieved in determining the shortest path for faster convergence. Presence of transition loops is vividly clear to cause longer convergence time which makes it a perfect theoretical investigation.

Revised Manuscript Received on November 15, 2019

M. Aliyu, Mathematical Sciences Department, Abubakar Tafawa Balewa University Bauchi, Nigeria. Email: maliyudeba@gmail.com

M. Murali, Department of Computer Science and Engineering, SRM IST, Tamilnadu, India. Email: muralim@srmist.edu.in

A. Y. Gital, Mathematical Sciences Department, Abubakar Tafawa Balewa University Bauchi, Nigeria.Email: asgital@yahoo.com

S. Boukari, Mathematical Sciences Department, Abubakar Tafawa Balewa University Bauchi, Nigeria. Email: bsouley2001@yahoo.com

II. RELATED WORK

Improving the quality of solution and convergence speed is nowadays the most fruitful ACO research in cloud computing. These issues have been explored by researchers through the modification of transition operator, hybridization in metaheuristic or preprocessing the input population [4]. In [23], integrating two population based metaheuristics with similar characteristics might likely delay the convergence time. The authors further argued that there is higher chance of obtaining better solution result by hybridizing ACO with different search strategies. Based on this analysis of attaining efficient convergence speed, integrating a deterministic with ACO might improve its performance because they have different characteristics which can leverage the strengths of each other. This section discus several related work reviews conducted by some authors on different ACO approaches to resource provisioning and further shows the analogy of summary.

Tawfeek et al.[4] adopted the ACO algorithm for resources allocation in cloud. The authors adopted makespan minimization as their objective function. The authors looked into the relative strengths and weaknesses of the algorithm by experimentation where incoming batch jobs are assigned to VM's based on a simple, short-term memory using constraint satisfaction rule. Graph theorem where PMs are represented with vertex (node) and edge defines VM migration from one PM to another was used to model the system [24] [25] [26]. The rule failed to address the issue of transition loops which causes delay in convergence and longer time for couplings to be assigned (task and VM). Same makespan minimization study by Mainak and Trachand [27] targets to balance load for IaaS cloud. The authors purported the Heuristic Based load Balancing Algorithm (HBLBA) to strategize tasks based on their sizes and incoming number to configure servers in order to suitably find VMs for assignment. Other makespan minimization by ACO technique studied can be seen in [19] [21] [22] [28] [29] [30] [22].

Chaharsooghi et al. [31] proposed a modified ACO algorithm with the aim to obtain a set of Pareto solution. The authors inculcated an approximate non-deterministic tree-search procedure based on the ACO. This leads to simplifying the probability calculation and also updating pheromone rule causing increase in ants learning ability. Qiang [32] purported Multi- Objective ACO (MO-ACO) algorithm that considered cost, load balancing and makespan minimization to be their objective function. The rule uses small number of tasks in their experiment and failed to address dependency between tasks for determining customer satisfaction. Existing setbacks in ACO which include easy falling into local optimal solution, poor convergence accuracy and slow solving speed at the primary stage was identified by Weihua et al. [33]. The authors decided to increase the convergence speed of the ACO and address the initial pheromone shortage through rapid searching ability of GA. In [34], the authors used the general concept of clonal selection and ACO algorithm to propose task scheduling solution. The CLONALG used in pattern recognition was used based on the independence of memory cells and antigen populations.

Table 1: Comparison of related work table

S/n	Author(s)	Methodology	Problem with the proposed solution
1	Zne-Jung and Chou-Yuan [26]; Weihua et al. [33], Senthil and Venkatesan [25] Jianbiao et al. [34].	Heuristics hybrid search algorithm method to ameliorate performance through faster search in resource allocation.	Hybridized 2 population based algorithms having similar search characteristics. They might end up slowing down the convergence speed. A deterministic technique having stringent tight procedure, where paths and values of both the functions and design variables are repeatable should be used to obtain a fixed topology structure
2	Chaharsooghi et al. [31] Tawfeek et al. [4], Zhe [24], Guo [19], Anupama et al . [20].	A simple, short-term memory using constraint satisfaction rule to avoid multiple visits to a VM in one ACO procedure.	The rule failed to address the issue of transition loops which causes delay in convergence and longer time of the assigned couplings. Hybridization to avoid loops and promote faster convergence.
3	Ren and Juebo [29]	Mmax-min rules and forward-backward ant mechanism to obtain a balanced distribution and coordination of tasks.	It did not strategize a rule for pheromone updates which can strategically reduce candidate nodes searching time. A rule for pheromone updates should be strategized and also promote faster convergence.
4	Shabeera et al. [22]	Heuristic method that reduced bandwidth usage and cross network traffic by placing required number of data in PMs and VMs which are physically closer.	It considered homogeneous VMs as cloud resources are heterogeneous which causes the resource utilization not being maximized; deteriorating the allocated VMs job completion time. It should be improved to consider heterogeneous VMs and also a method that promotes faster convergence thereby saving additional bandwidth cost.
5	Harshitha and Beena, [29]	Uses shortest path method on TSP method. By providing a global search technique. As such, minimizes delay in resources allocation.	Initial takeoff delay and presence of transition loops were not addressed. As such, took longer time for the system to converge. Hybridization with spanning tree to avoid loops.

6	Qiang [32]	Pseudo - random transition probability rule was used to improve pheromone update rule and basic ant colony algorithm heuristic function.	Failed to address tasks dependency, small number of tasks in their experiment and consider customer satisfaction. A mechanism for proper scheduling for faster convergence for customer satisfaction
7	Rajalakshmi et al. [21]	Optimal VM placement algorithm that periodically monitors the response time of each Cloud Service Providers (CSP) so as to reduce delay in resource provisioning to users.	The VM placement used in this instance is not tested on real time environment; hence its efficiency and convergence speed cannot be justified. The system should be implemented in real time environment
8	Mainak and Trachand [27].	Queing model strategized incoming tasks and their sizes to configure servers leading to makespan minimization and tasks completion.	They only considered task-VM mapping and server configuration so as to minimize makespan and balance load. The authors failed to address the issue of heterogeneous resources and transition loops that can take longer time for the process to converge.
9	Mohit et al. [30]	Threshold based trigger to provide elasticity and design an approach that is capable of handling maximum user request before jobs deadline.	Their results prove to be effective in comparison with traditional algorithms (FCFS, SJF, and MinMin) not heuristics optimization algorithms. It should be bench marked with optimization algorithms to show its efficiency and also a scalable means for adding more users.

Based on the review conducted and summarized in table1, it can clearly be seen that the issue of transition loops causes delay in convergence speed, resulting to longer time for resources provisioning. As such, it is an overhead that leads to wastage of resources (like time, CPU cycles, bandwidth, and memory), hence degrades system performance which is a very serious issue that needs to be addressed [4] [22] [35] [36] [37].

III. METHODOLOGY

The constraint satisfaction rule for makespan minimization adopted in [22] failed to address the issue of transition loops which causes delay in convergence and longer time of the assigned couplings. The proposed research will focus on improving the rule by injecting the mechanism of loop free SPT algorithm to improve on convergence, makespan time and cost in a multi-dimensional spaces.

A. Problem Formulation

For sufficient scheduling of resources where tasks can be executed by cloud users, each task requires sufficient hardware configuration (processor, RAM, and bandwidth). For m heterogeneous VMs; vm_1, vm_2, $vm_{3..}$ .vm_m running in the environment of cloud, here, scheduler tries to map T_i task to vm_j VM. The value of Decision Variable DV_{ij} if task T_i is matched with VM, vm_j =1, otherwise=0.

$$DV_{ij} \begin{cases} 1; \text{ if Ti is assigned for vmi} \\ 0; \text{otherwise} \end{cases} \quad \dots (1)$$

When user demands for cloud services in form of task request, scheduler assigns a VM for necessary actions. Decision variable as can be seen from equation (1) is 1 for successful assignment, otherwise is 0.

$$ET(i) = \frac{\text{MLi}}{\text{Ps} * \text{Rs} * \text{Qn} * \text{Bs}} \quad \dots (2)$$

Where;
ET (i) is Execution Time,
MLi is Million Lines of instruction,
Ps is Processor speed,
Rs is capacity of RAM,
Qn is number of processor and RAM,
Bs is bandwidth in mbps (megabit per second).

Equation 2 illustrates execution time at an instant *i* for a specific VM using time shared mechanism. VM receives user job for processing in millions of instruction length. Processing rate here is based on the quality of hardware resource configurations. The mechanism for space shared here; execution of task will be carried out one after the other implying CPU executes one task per CPU/core. DV_{ij} will be applicable in waiting queue for remaining incoming tasks for VM assignment. Remaining time is the overhead or time interval between when a user job is submitted for assignment and the delay experienced before execution starts. When a user job is submitted, it has to be put in ready queue for it to be scheduled for VM assignment.

$$RT_i = WT_i - ET_i \quad \dots (3)$$

Equation 3 defines the resource provisioning remaining time. Here, WT_i is the workload submission time for jobs to be assigned to a VM and ET_i is the execution start time for submitted jobs after leaving the ready queue. The WT_i as shown in equation 4 must be greater than or equal to the ET_i for task execution to avoid resource contention.

$$WT_i >= ET_j \quad \dots (4)$$

Where task are executing without delay; VM execution time is given in equation 5 as

$$ET_i = \max \sum_{i=1}^{n} \text{Execution time of task (Ti)} * \text{DVij} \quad \dots (5)$$

Speeding up the execution of tasks seeks to determine system performance through pipelining. Sufficient pipelined system also produce better system throughput (million number of task request completed per unit time). With an efficient technique in cloud platform, incorporating the smallest amount of extra resources as possible increases its scalability leading to better performance [38]. Scalable techniques in Cloud also burst traffic and heavy workloads to cope with the voluminous services demanded. Optimization metric used in this research is based on makespan time minimization.

Makespan is the most populous optimization metric used in cloud computing [23] as clients look for the fastest application while CSPs tries to service more users to earn more profit [39]. Makespan covers the earliest start time on the first VM to the last finish time of the last VM.

Makespan (t) = min $\sum_{j=1}^{m}$ Execution time of VM … (6)

As can be seen from equation 6, makespan time covers task execution time length elapsed for VM_1, VM_2... VMm. Multi-objective resource provisioning constrained problem seeks to get the shortest logical time span, by efficiently using available resources as possible to achieve the minimum makespan.

B. System Architecture

The proposed system architecture comprises of three major components; improved ACO module, improved SPT module and hybridized ACOST. Cloud users' send and receive their jobs request through the edge path (see figure 1). Edge path sends and receives N number of independent task for Million instructions Length (MLi) to and from the VMs. VMs query (communicate) each other in form of broadcast for information exchange.

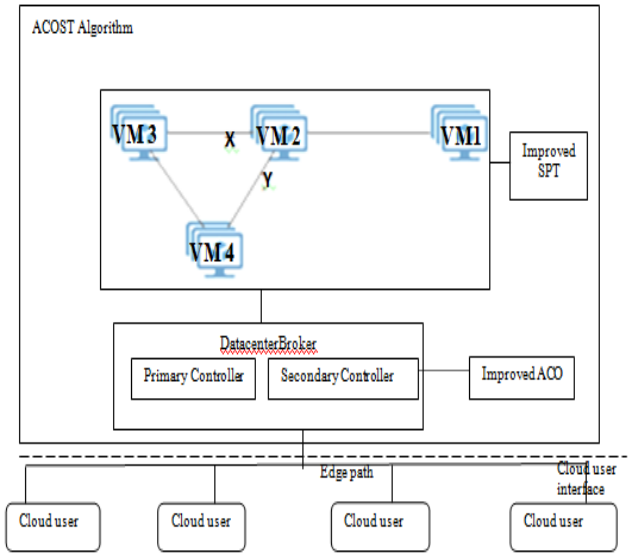


Fig 1: Proposed ACOST Architecture

From figure 1, VM1 communicates to VM2. VM2 reach VM3 at 'x' and VM4 at 'y'. The redundant link between VM3 and VM4 will also send message y to VM3 and message x to VM4. As such, there is high tendency of occurrence of transition loops which can cause convergence delay amounting to CPU, memory and bandwidth wastage. Datacenterbroker is the core infrastructural service level model composing of primary and secondary controllers. The primary controller is responsible for defining paths to transverse for both artificial ants and heterogeneous/homogeneous tasks set. Artificial ants are used here in determining how the transition is going to take place for information exchange to avoid loop occurrence. In case of any failure by the primary controller, the secondary controller will act as as backup for subsequent path determination.

1. Improved ACO Module

With improved ACO algorithm, S_i stage representation is adopted for each ant, where each s_k has S_j states represented using connected graph $G = (S_i, S_j)$. A complete and feasible multi-objective resource allocation solution is formulated when the shortest path is formed in G. Ants conduct transversal in order to look for optimal paths in reaching all VMs through a probabilistic function. Optimal elaboration will be performed by each ant in its current state for the remaining trails. The trail and stage are the two activities for the foraging behavior. Slight modification is conducted on the initial population for the ACO.

Initial population

A good initial population generation approach provides a promising advantage for optimization in solving dynamic problems. It provides a finite initial state space the chances for designing new metaheuristics [6]. Ants here construct a probability parameter for iteration through all the given VMs. Beginning from a random VM; an ant chooses succeeding VM using pheromone and heuristic information. Each transition can be represented as (S_i, S_j), i, j ∈ (1 ... n) of VMs where S_i and S_j denote stage and state respectively. Pheromone information will be denoted as $\xi_{i,j}$ and heuristic information denoted by $\Re_{i,j}$.

$$\Re_{i,j} = \frac{1}{L(i,j)} \quad \ldots (7)$$

Equation 7 denotes the Heuristic value, where $L_{i,j}$ is the length between VM_i and VM_n. The heuristic function $\Re_{i,j}$ will be used as the algorithms' probability parameter iterating at a range $0 \leq \Re_{i,j} < 1$. Ants make use of the parameter for choosing vm_i to vm_j from the set G of VMs which have not been transverse before. All outputs obtained will get closer and closer to 1 in achieving convergence. Entire system convergence here is based on probability distribution $(T_{i,j})$ over G as shown from equation 8.

$$T_{i,j} = \frac{\Re(i,j) \cdot \xi(i,j)}{\sum_{i,j=1}^{n} G \cdot \xi(i,j) \cdot \Re(i,j)^{\sigma}} \quad \ldots (8)$$

The overall ACOST objective here is to optimize $\Re_{i,j}$and $\xi_{i,j}$. σ is the relative influence constant to determine paths transversal for ants in making decisions.

2. Improved SPT Module

A graph's spanning tree is its subset having all the nodes connected with least possible number of arcs. Furthermore, spanning trees have no loops or cycles and cannot be disconnected [40]. Improved SPT here will come up with a transition operator as a relative influence to eliminate loops in the set *G* = (*N*, *M*). When cloud users request for job(s), spanning tree stops all redundant paths that could cause loop resulting in convergence delay, ensuring the existence of only one logical path between all destinations in the cloud. If users' request is intentionally prevented from entering or leaving a path, that path will be considered as blocked path. That won't include the artificial ants by ACOST for determining topology. To provide redundancy, the physical paths will still exist. But these paths will be in disable state to mitigate the occurrence of transition loops. If the path is ever required to indemnify for a path disruption, SPT will allow the redundant path to become active by recalculating the topology and unblocks the necessary path. Five path states will be adopted for this operation as shown on figure 2.

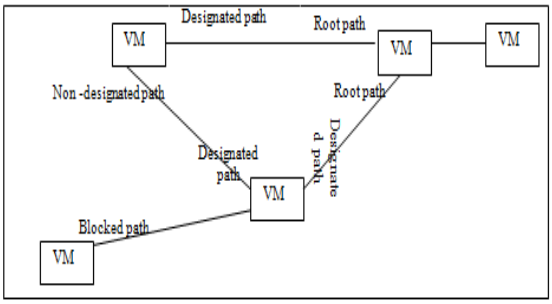


Fig 2: Five VM path states

- Designated - a VM path will be considered designated if it can ingrerss or egress out tasks requests to other VMs.
- Non-designated- a VM path is to be considered non-designated if tasks request are not allowed but allows the transversal of artificial ants to determine best path.
- Root- a VM path will be considered a root path if it connects directly to the root bridge. Root-bridge here is the VM serving as the focal point for determining best paths to reach other VMs.
- Blocked – a path or edge will be considered blocked if for security or access control, a path is permanently blocked to prevent all passage. A blocked path will remain blocked except if activated by the administrator.
- Edge – a path will be considered as edge path if it can ingress and egress out user task requests.

The 5 path definition will be used to control activities from the DataceterBroker for establishing negotiations between VMs in the cloud environment. It will also ensure QoS through VM path definition in providing policy for the allocation of VMs to users.

3. Hybridized ACOST

Hybridized ACOST module is responsible for scheduling resources that have no similar computation performance and resource costs. Entire scheduling here is expected to accept varying number of tasks, average machine instruction (processing capacity of each resource in millions instructions per second) of tasks, deviation percentage of machine instruction granularity size, and processing overhead for all the tasks. Path assignment will also be carried out in this module in accordance to VM path states.

ACOST Algorithm
Step1: Determine Root Bridge (RB) and Backup bridge
Step2: Determine all distance from RB
Step 3: Switch (Path) {
Case 1: Root path always forward and receives;
Break;
Case 2: Designated path always receives and might send;
Break;
Case 3: Non-designated path might send or might receive;
Break;
Case 4: Root path always forward and receives;
Break;
Default: Blocked path don't send or receive;
Break;
Step 4: Compute $T_{i,j}$; $\forall G(N, M)$; $\xi_{i,j}$, $\Re_{ij} = vm_k$; i,j=1,2,3,..,n
Step5: Compute new topology for all tasks available
Step6: Assign relative influence
Step7: Set $\sum T_i=0$ (where T_i is the distribution probability)
Step8: Set an index i to 1 and resource ID j to 1
Step9: Get new heuristic value
*Step10: Ti=Heuristic * MIPS of resource*
Step11: Repeat steps 8 and 9 until $T_i=1$ for all received tasks
Step12: End for
Step 13: End for

Fig 3: ACOST Algorithm

The algorithm here determines ant task movement in accordance to path definition. Parameter chosen here for the topology formation is based on the VMs, probability distribution, pheromone information and heuristic information to determine paths assignment. From the algorithm (figure3), the first 2 steps determine the root bridge that will serve as the focal point for topology determination. All paths will be determined from the root bridge location in reaching all VMs. The first VM automatically becomes the root bridge while the second VM becomes the backup which becomes active in case the root bridge fails. Pheromone and heuristic information that determines ant movement defines paths from the root bridge ensuring one and only one vm_j transversal for the VM_i. Relative influence is also attached to the probability distribution function for task assignment. It is believed after proper path and state definition; probability distribution will ensure loop free transition. After successful definition of path states, transition loops will be avoided and the system will converge in least time. As such, effective tasks scheduling will be achieved optimizing computation/communication ratio thereby making the system more scalable. With the increase number of cloud users, scalable systems like ACOST will handle a growing amount of task requests making it more robust to accommodate that growth.

C. Convergence Rate

Consider;

$$\lim_{t\to\infty} \text{Makespan}(t) = \lim_{t\to\infty} \sum_{j=1}^{m} ET(Ti) \quad \ldots \quad (9)$$

Taking the limit (from equation 6) for the makespan time covers task execution time length needed to achieve faster convergence rate. Convergence condition can be demonstrated according to solution history based on reduction as

$$\lim_{t=1\to m} T = \begin{cases} 0, if, max||\Re i,j||) < 1, \\ ||\Re i,j|| = 1. \end{cases} \quad \ldots (10)$$

The algorithm convergence condition is based on the probability distribution. The probability distribution is a function between 0 and 1. It covers time to transverse VM_1 to VM_m where total heuristic function iterates when less than 1 and terminates at an absolute value of 1.

IV. Experimental Setup and Result

We assume that there is no precedence constraint between tasks i.e, tasks are mutually independent. Tasks also cannot be interrupted of moved to another processor during execution, i.e no preemption. Simulation scenario used in this study includes a data center with 10 PMs, 40VMs and also a range of 10-120 tasks length ranging from 20000-500000 ML length. Four Dell PMs used run at 3.0 GHz [CO(TM) i7-3540M Intel (R)], 16GB primary memory, and 2TB secondary storage. The remaining six are Linovo with 2.7 GHz [CO(TM) i5-2540M Intel (R)], 8GB primary memory, and 1TB secondary storage. Each VM runs at 128MB with a storage capacity of 4GB that randomly selects PM processor.

Cloudsim version 3.0 toolkit was used which combined various features such as containers, modeling of Web applications on multi-clouds and VM extensions with performance monitoring. Based on different number of task with random length cloudlet (tasks), the simulation was run for more than 3 hours (150 times approximately) and the result was calculated using the policy of space shared in cloudsim. In performance analysis to test the superiority and efficiency of ACOST, same configuration was used to execute and compare results with CLONALG [34], ACO [4] and MO-ACO [32].

V. Analysis and Comparison of Experimental Result

After an in-depth computation and analysis of experimental results, a set of tasks ranging from 10-30 were executed on 10VMs, 20-50 on 20VMs, 40-100 on 30VMs and 80-120 on 40 VMs

A. Makespan Time

Table 1 records each algorithm's makespan time based on the number of tasks and VMs used to achieve required result.

Table 1: Comparison of makespan

No. of VMs	No. of Tasks	CLONALG Time in seconds	ACO Time in seconds	MO-ACO Time in seconds	ACOST Time in seconds
10	10	657	644	623	**621**
10	20	934	762	782	**692**
10	30	1202	1111	1222	**731**
20	20	333	313	318	**303**
20	30	513	497	521	**423**
20	40	717	681	723	**512**
20	50	945	913	1021	**512**
30	40	538	491	633	**420**
30	60	681	611	735	**540**
30	80	791	689	941	**613**
30	100	1118	972	1222	**623**
40	80	512	417	584	**381**
40	100	811	766	878	**491**
40	110	838	799	1022	**515**
40	120	1108	1009	1334	**515**

As can be seen from the table, ACOST records the least time as compared to the benchmarked algorithms.

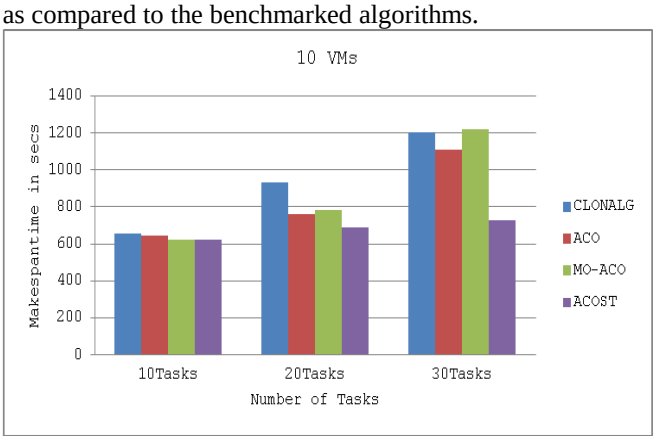


Fig 4: Makespan time of CLONALG, ACO, MO-ACO and ACOST for 10VMs

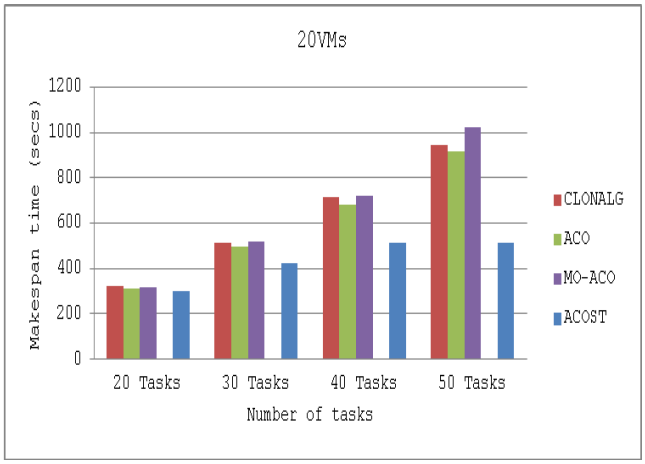


Fig5: Makespan time of CLONALG, ACO, MO-ACO and ACOST for 20VMs

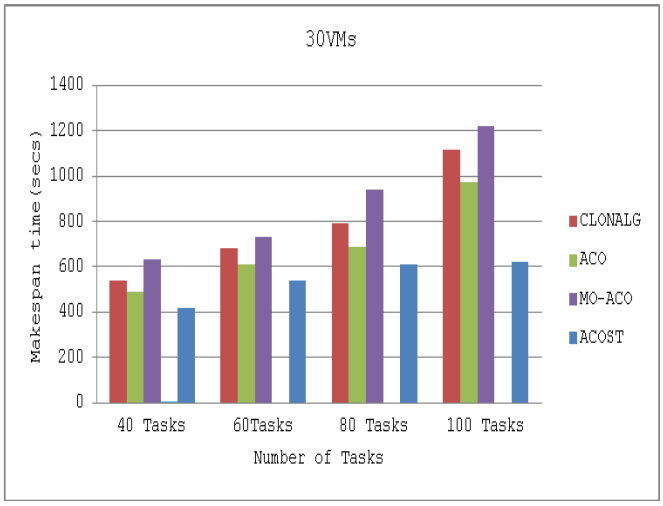


Fig 6: Makespan time of CLONALG, ACO, MO-ACO and ACOST for 30VMs

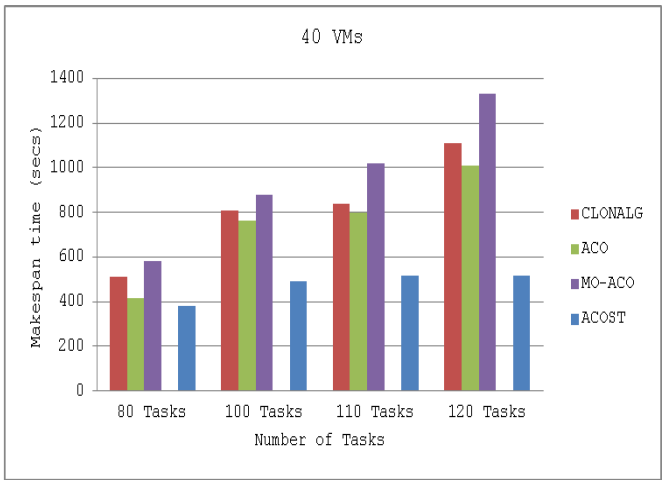


Fig 7: Makespan time of CLONALG, ACO, MO-ACO and ACOST for 40VMs

From figure 4 to 7, makespan times are shown in seconds (cloudsim relative time unit) from the y-axis with the total number of VMs on the x-axis. With 10VMs running 10 tasks (million instruction lenght); ACOST takes a time of 620 seconds as compared to CLONALG that takes 656 seconds, ACO 643 seconds and MO-ACO 622 seconds.

With increase in number of tasks, makespan time also increases except for ACOST as it normalizes due to the convergence rate. At the last lap of the simulation with the highest number of tasks (120) with 40 VMs, ACOST also takes the least time of 516 seconds as compared to the benchmarked CLONALG with 1109, ACO 1008 and MO-ACO 1330. This clearly shows that ACOST elimination of loops results to faster convergence, reduction in makespan time, hence normalizing task processing.

B. Throughput

Throughput *is* the number of processes completed per unit time. The algorithms performance was measured in form of throughput based on the measure of imbalance degree (Δ) as can be seen from equation 11.

$$\Delta = \frac{\Omega+\sigma}{\partial} \quad \ldots \quad (11)$$

Where;
Ω is the maximum execution time of task,
σ is the minimum execution time of task,
∂ is the average execution time of task.
From figures 8-12, throughput time is shown on the y-axis while task range on the x-axis.

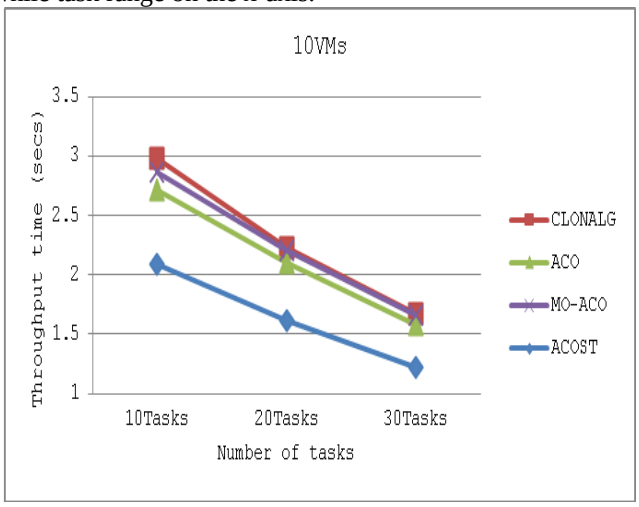


Fig 8: Throughput of CLONALG, ACO, MO-ACO and ACOST for 10vms

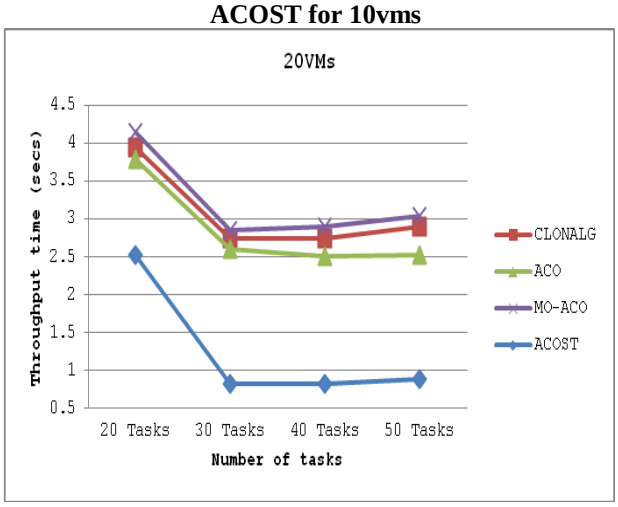


Fig 10: Throughput of CLONALG, ACO, MO-ACO and ACOST for 20vms

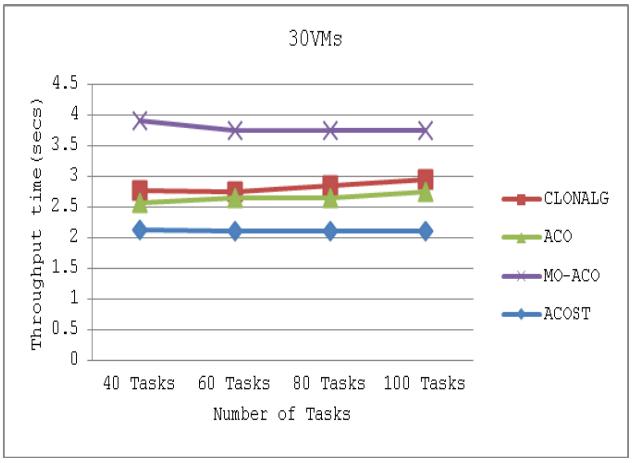


Fig 11: Throughput of CLONALG, ACO, MO-ACO and ACOST for 30vms

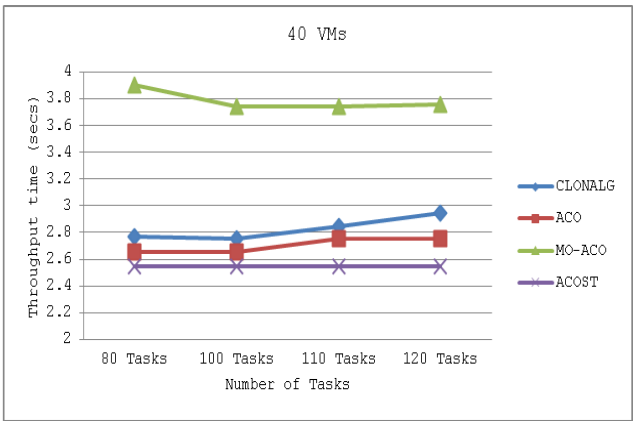


Fig 12: Throughput of CLONALG, ACO, MO-ACO and ACOST for 40VMs

ACOST achieves better system throughput to have finally settled on a time of 2.1 seconds as compared to CLONALG, ACO and MO-ACO algorithms that kept fluctuating in their convergence. An optimal time performance of full utilization of resources ensures a well balanced load distribution which suffices bandwidth consumption and further cost [41].

VI. CONCLUSION

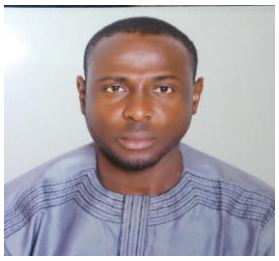
To achieve a well balanced load, minimize bandwidth consumption and minimize makespan time; ACOST algorithm was developed which eliminated loops for faster algorithm convergence. The system architecture component involves the datacenter broker which contains primary and secondary controllers that determines how user jobs are moved in and out of the defined 5 VM path states. The relative influence attached to the ACO probability distribution function defines ant movement thereby ensuring loop transition. A total of 10 to 40 VMs receiving a range of 10-120 user tasks varying of length from 20000-500000MI was evaluated. ACOST recorded least makespan time based on simulated results with a better system throughput that outperforms CLONALG, ACO and MO-ACO. As such, it is more scalable to incorporate extra resources with a well balanced load distribution. Cloud security is an essential aspect for CSPs as their trustworthiness strictly depends on it. Future work will look at improving using the smearing technique.

REFERENCES

1. P. HADI, AND M, REZA, SOLVING A MULTI-OBJECTIVE OPEN SHOP SCHEDULING PROBLEM BY A NOVEL HYBRID ANT COLONY OPTIMIZATION, A JOURNAL ON EXPERT SYSTEMS WITH APPLICATIONS, VOL 38, ISSUE 3 (2011) PPG 2817-2822.
2. A. Ines, S. Christine and G. Khaled, Ant Algorithm for the Multi-Dimensional Knapsack Problem, International Conference on Bioinspired Optimization Methods and their Applications (Bioma 2004, (2004).
3. L. Wu, S.K. Garg & R. Buyya, SLA-Based Admission Control for a Software-as-a Service Provider in Cloud Computing Environments, Journal of Computer and System Sciences, 78(5) (2012) pp.1280-1299.
4. M.A.Tawfeek, A. El-Sisi, A.E. Keshk & F.A. Torkey, Cloud Task Scheduling Based on Ant Colony Optimization. 8th IEEE int conf comput eng syst, (2013) pp. 64–69.
5. E. Rainer, Quadratic Assignment Problems, Handbook of Combinatorial Optimization ppg 2741- 2814 (2013).
6. M. Dorigo, M. Birattari, & T. Stutzle, Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique, IEEE Computational Intelligence Magazine, (2006).
7. W.J. Gutjahr, (2007). Mathematical Runtime A nalysis of ACO Algorithms: Survey on an emerging issue. Swarm Intelligence © Springer Science + Business Media, LLC ppg 59–79 DOI 10.1007/s11721-007-0001-1.
8. I. Wegener, Towards a Theory of Randomized Search Heuristics, in: Mathematical Foundations of Computer Science, LNCS Journal, vol. 2747 (2003) pp. 125–141 Springer.
9. D. Merkle, and M. Middendorf, Modelling the Dynamics of Ant Colony Optimization Algorithms, A Journal on Evolutionary Computation vol 10 (3) (2002) ppg 235–262.
10. W.J. Gutjahr, A Generalized Convergence Result for the Graph-Based Ant system, A Journal of Metaheuristic, Probability in the Engineering and Informational Sciences (2003) ppg 545–569.
11. F. Neumann and C. Witt, Runtime Analysis of a Simple Ant Colony Optimization Algorithm, 17th International Conference Proceedings of theand Symposium on Algorithms Computation (ISAAC '06), in: LNCS, vol. 4288 (2006) ppg 618–627 Springer.
12. [12] F. Neumann and C. Witt, Runtime Analysis of a Simple Ant Colony Optimization Algorithm, A journal of Algorithmica vol 54 (2) (2009) pg 243–255.
13. W.J. Gutjahr, First Steps to the Runtime Complexity Analysis of Ant Colony Optimization, A journal on Computers and Operations Research vol 35 (9) (2008) ppg 2711–2727.
14. B. Doerr, F. Neumann, D. Sudholt and C. Witt, Runtime Analysis of the 1-ANT Ant Colony Optimizer, A journal on Theoretical Computer Science vol (17) (2011) ppg 1629–1644.
15. T. Stützle, and H. Hoos, MAX–MIN Ant System. A Journal of Future Generation Computer Systems vol 16 (2011) ppg 889–914.
16. J. B. Orlin, K. Madduri, K. Subramani and M. Williamson, A Faster Algorithm for the Single Source Shortest Path Problem with Few Distinct Positive Lengths, A journal of Discrete Algorithms vol8 (2) (2011) ppg 189–198.
17. D. Sudholt, Using Markov-Chain Mixing Time Estimates for the Analysis of Ant Colony Optimization, Proceedings of the 11th Workshop on Foundations of Genetic Algorithms (FOGA 2011) (2011) pp. 139–150 ACM Press.
18. F. Neumann, D. Sudholt and C. Witt, Rigorous Analyses for the Combination of Ant Colony Optimization and Local Search, In: Proceedings of the 6th International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS '08), in: LNCS, vol. 5217 (2011) ppg 132–143 Springer.
19. X. Guo, Ant Colony Optimization Computing Resource Allocation Algorithm Based on Cloud Computing Environment. International Conference on Education, Management, Computer and Society (2016) pp 1039-1042, Atlantis press
20. T. Anupama, P. Richhariya & P. Satyaranjan, Ant Colony Based Cloud VM Allocation and Placement Approach for Resource Provisioning in Cloud, IEEE International Journal of Computer Applications, Volume 158 – No 4 (2017) pp 8-12.
21. S. Rajalakshmi, U. Fareentaj & T.K. Divya, 2017. A.n Effective Mechanism for Virtual Machine Placement using ACO in IAAS Cloud. IOP Conf. Series: Materials Science and Engineering pp225, doi:10.1088/1757-899X/225/1/012227.
22. T.P. Shabeera, K. M. Madhu, S. Sameera & Murali K., 2016. Optimizing VM allocation and Data Placement for Data-Intensive Applications in Cloud Using ACO Metaheuristic Algorithm. International Journal on Engineering, Science and Technology, pp 616-628, http://dx.doi.org/10.1016/j.jestch.2016.11.006
23. K. Mala & S. Sarbjeet, 2015. A Review of Metaheuristic Scheduling Techniques in Cloud Computing. Egyptian International Journal, pp 275-295, University of Cairo, http://dx.doi.org/10.1016/j.eij.2015.07.001, available at Elsevier.com, scidirect.com.
24. G. Zhe, The Allocation of Cloud Computing Resource Based on The Improved Ant colony Algorithm, IEEE 6th international conference on Intelligent Human-Machine Systems and Cybernetics (2014) pp 334-337.
25. A. M. Senthil & M. Venkatesan, An Efficient Multiple Object Resource Allocation Using Hybrid GA-ACO Algorithm, Australian Journal of Basic and Applied Sciences Journal (2015) pp 53-59 home page: www.ajbasweb.com.
26. L. Zne-Jung & L. Chou-Yuan, A Hybrid Search Algorithm with Heuristics for Resource Allocation Problem, Information Sciences journal, Volume 173, Issues 1–3 (2006)Pages 155-167.
27. A. Mainak & A. Tarachand, Heuristic-Based Load-Balancing Algorithm for IaaS Cloud, Future Generation computer systems, vol 8 (2017) pp 156-165.
28. H. D. Harshitha, & B. M. Beena, Ant Colony Optimization for Efficient Resource Allocation in Cloud Computing, International Journal on Recent and Innovation Trends in Computing and Communication, vol5, issue 6 (2017) Pp 1232-1235.
29. G. Ren & W. Juebo, 2015. Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization. A Journal of Future Internet ISSN 1999-5903, pp 465-483, doi:10.3390/fi7040465 available at: www.mdpi.com/journal/futureinternet.
30. K. Mohit, D. Kalka & S. C. Sharma, Elastic and Flexible Deadline Constraint Load balancing Algorithm for Cloud computing. 6th International Conference on Smart computing and Communications, Vol 125 (2018) Pages 717-724 Kurushetra India.
31. S. K. Chaharsooghi, H. Amir & K. Meimand, An Effective Ant Colony Optimization Algorithm (ACO) for Multi-Objective Resource Allocation Problem (MORAP), Applied mathematics and Computation Journal, Volume 200 (2008) Pg 167-177.
32. G. Qiang, 2017. Task Scheduling Based on Ant Colony Optimization in Cloud Environment. 5th International Conference on Computer-Aided Design, Manufacturing, Modeling and Simulation (CDMMS) AIP Conf. Proc. doi: 10.1063/1.4981635, Published by AIP Publishing.
33. H. U. Weihua, X. U. Junjun, & B. A. Qian, Cloud-Computing-Based Resource Allocation Research on the Perspective of Improved Ant Colony Algorithm, IEEE International Conference on Computer Science and Mechanical Automation, (2015) pp76-80.
34. L. Jianbiao, Y. Zhong, L. Xiaowei, H. Lin & Q. Zeng, Hybrid Ant Colony Algorithm Clonal Selection in the Application of the Cloud's Resource Scheduling, A journal on Distributed, parallel and clustering computing (2014) Pp67-72, Cornell University journal.
35. A. Al-Maamari & F. Omara, Task Scheduling using Hybrid Algorithm in Cloud Computing Environments, IOSR Journal of Computer Engineering, p- ISSN: 2278-8727, vol 17 (2015) pp 96- 106.
36. S. M. Mousavi & G. Fazekas, A Novel Algorithm for Load balancing and Using HBA and ACO in Cloud Computing Environments, International Journal of computer science and information security (2016) pp 48-52.
37. N. C. Brintha, S. Benedict & W. T. Jappes, 2017. A Bio-Inspired Hybrid Computation for Managing and Scheduling Virtual Resources using Cloud Concepts. International Journal in Applied Mathematics & Information Sciences, pp 565-572 DOI: 10.18576/amis/110228.
38. M. Luis, C. Juan and M. Daniel, 2019. The Challenge of Service Level Scalability for the Cloud. International Journal of Cloud Applications and Computing (IJCAC) vol1(1), PPg 11. DOI: 10.4018/ijcac.2011010103
39. J. Sujata, B. Sanjay and R. Kiran, 2017. Customer experience and associated customer behaviour in end user devices and technologies (smartphones, mobile internet, mobile financial services). International Journal of High Performance Computing and Networking Vol 10, issue 5, DOI: 10.1504/IJHPCN.2017.083209.

40. N. Frank & W. Carsten, Ant Colony Optimization and the Minimum Spanning Tree Problem. Electronic Colloquium on Computational Complexity, (2006) Report No. 143.
41. A. Layla, 2017, Comparative Study for Different Provisioning Policies for Load balancing in Cloudsim. IJCAC Vol 7 (3) ppg 11 DOI: 10.4018/IJCAC.2017070104.

AUTHORS PROFILE



Muhammad Aliyu, A PhD scholar of Computer Science at the Abubakar Tafawa Balewa University (A.T.B.U) Bauchi, Nigeria. Acquired B.Tech and Msc in Computer Science in A.T.B.U, Bauchi. Attended quite a number of local and international conferences in Nigeria and India. A member and certified instructor CISCO professional, CPN Nigeria and NCS Nigeria. Area of interest includes Networking, Cloud environment, Machine learning, big data and Pattern recognition.

Dr M. Murali is an assistant professor in the Department of Computer Science and Engineering SRM IST Kattankulathur campus Tamilnadu India. Have both Msc and PhD in Computer Science. He is currently involved in teaching, researching and supervision of students both undergraduates and Postgraduates in Computer Science. Research interest includes Mobile and Data Management, Database, Wireless network, and Machine Learning.

Dr. Abdulsalam Y Gital, is a Doctor of Computer Science from Abubakar Tafawa Balewa University, Bauchi Nigeria where He is currently involved in teaching, research and supervision of students both undergraduates and Postgraduates in Computer Science. He has a B-Tech, M.Sc. and PhD in Computer Science. His current research areas are Modeling and Simulation, Cloud Computing, Collaborative Virtual Environment and Computer Communication and Network.

Dr. **Souley Boukari**, is a Professor of Computer Science from Abubakar Tafawa Balewa University, ATBU, Bauchi Nigeria where He is currently involved in teaching, research and supervision of students both undergraduates and Postgraduates in Computer Science. He has a B-Tech, M.Sc. and PhD in Computer Science from ATBU. His current research areas are Data Mining, Machine Learning, Artificial Intelligence, Software Engineering, cyber security and Digital Forensic.