A Proposed Meta-Heuristic Approach for Cloudlets Scheduling in Cloud Computing Environment

Aida A. Nasr*, Nirmeen A. El-Bahnasawy*, Gamal Attiya* and Ayman EL-SAYED*

*Dept. of Computer Science and Eng., Faculty of Elect., Eng., Menoufia University.

(Received: 4 Mar. 2018 – Accepted: 21 Jun. 2018)

Abstract

This paper presents a new hybrid approach, called ACOSA, for cloudlets scheduling to enhance the scheduler behavior in Cloud computing (CC) environment and to overcome the results oscillation problem of the existing meta-heuristic scheduling algorithms. The proposed approach combines both the Ant Colony Optimization (ACO) and Simulated Annealing (SA) algorithm to improve both quality of solutions and time complexity of the scheduling algorithm. The proposed approach is evaluated by using the well-known CloudSim, and the results are compared with the ant colony and simulated annealing separately in terms of schedule length, load balancing, and time complexity. It decreases the schedule length by 29.75% with SA and 12.25% with ACO. The ACOSA provides higher load balancing degree. It improves the balancing degree ratio by 36.36% than SA and 12.13% than ACO algorithms.

1. Introduction

Cloud computing platform is a novel kind of shared resources. It provides large pools of resources and let end users to use these resources over the Internet [1-3]. Users can get everything in cloud environment as a service such as Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). Providers, like amazon EC2, permit their clients to assign, access, and manage a group of virtual machines (VMs) that run inside the data centers and only charge them for the period of using the machines. Therefore, management of cloud resources is

critical, especially when several cloudlets (i.e. is the object that refer to tasks) are submitted simultaneously to (CC) environment.

Nowadays, (CC) becomes an efficient paradigm, as it presents high-performance computing resources to solve large-scale scientific and engineering problems. However, one of the important issues that degrade cloud-computing performance is cloudlets scheduling. This problem is characterized by the presence of limited number of VMs on which several cloudlets have to be executed. Therefore, the main goal is to search for good schedule of the cloudlets on the VMs to minimize the execution cost or schedule length.

Several researchers have developed algorithms to solve the cloudlets scheduling issue [4-8]. However, most of the existing algorithms consider schedule length, and disregard several constraints that may affect the scheduling process like memory and processing load constraints. In addition, most of the existing algorithms have a common oscillation problem. That is, for the same cloudlets scheduling problem, the obtained results of most meta-heuristic algorithms change with each running time. In other words, because of random solution and random values that used in meta-hubristic algorithms, results are oscillated with the running times.

This paper presents a new hybrid approach for cloudlets scheduling in (CC) environment to improve the performance of (CC) and solve the oscillation problem. The proposed algorithm, called ACOSA, combines both the Ant Colony Optimization (ACO) and the Simulated Annealing (SA) algorithm. Indeed, the proposed algorithm takes into consideration the requirements of different cloudlets and the availability VMs resources. That is, it takes both the memory and processing load constraints. The proposed approach is evaluated by using the well-known CloudSim [9], and the results are compared with ant colony and simulated annealing algorithms separately in terms of schedule length and time complexity.

The remainder of this paper is organized as follows. Section 2 introduces (CC) and cloudlet scheduling problem. Section 3 illustrates the formulation of scheduling problem as an optimization problem. Section 4 describes the existing scheduling techniques, while Section 5 presents the proposed approach. Section 6 presents the simulation results while Section 7 presents the conclusion of this research work.

2. Cloud Computing and Cloudlet Scheduling Problem

The main components of cloud computing scheduler are: User/client, cloud information system, scheduler, and VMs [10]. The relationship between the scheduler components is shown in Figure 1. Client submits his/her cloudlet(s) to cloud to be executed. Information system is responsible for collecting all information about cloudlets and resources. This component is very important as it provides the necessary information of cloudlets that arrived at (CC) environment for execution purpose. This information includes cloudlet length, arrival time and resources requirements. The information system also detects the resources availability in the cloud-computing environment. Datacenter broker includes the scheduler that responsible for scheduling the cloudlets onto the VMs. It is the backbone of scheduling process. It determines the execution order of each cloudlet. VMs are the main components in (CC) environment that are responsible for execution of cloudlets and return the results.

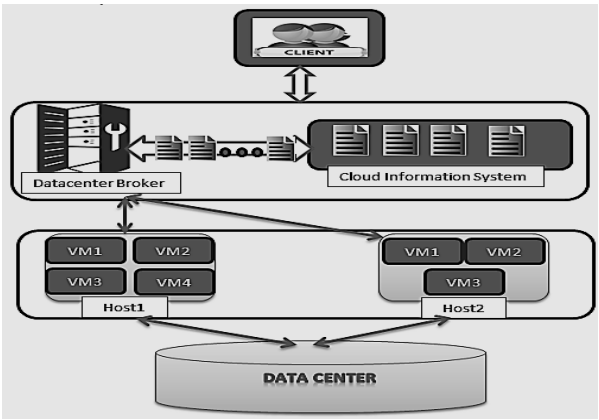

Figure 1: Components of cloud computing environment [10].

In (CC) environment, several cloudlets arrive to the system at the same time. Each cloudlet needs to be assigned into a suitable VM to be executed in a shortest time. However, because the number of available VMs is less than the number of submitted cloudlets, a scheduling algorithm is required to schedule the cloudlets onto the available VMs. This problem is called cloudlets scheduling problem. Briefly, given a set of *n* cloudlets to be executed on *m* VMs in the cloud-computing environment. The cloudlets require certain resources and have computational capacity requirements.

On the other hand, the VMs have limited resources as memory and processing power. Thus, the purpose is to schedule the cloudlets onto the VMs such that the schedule length is minimized, the requirements of cloudlets are met, and the capacities of the VMs resources are not violated. In other words, cloudlets should be scheduled efficiently to reduce the execution cost and time.

3. Problem Formulation

The cloudlet-scheduling problem may be formulated as an optimization problem to be solved by optimization approaches. Designing a mathematical model to the cloudlet-scheduling problem involves two steps; (i) formulate a cost function to represent the objective of the cloudlets scheduling, (ii) formulate set of constraints in terms of the cloudlets requirements and the availability of the VMs resources.

To formulate the scheduling problem, let n be the number of cloudlets, m is number of VMs, e_{iv} be the processing time of cloudlet *i* on machine *v*, and x_{iv} is a binary variable such that $\mathrm{x_{iv}}$ is 1 if the cloudlet i is assigned to machine v and 0 otherwise as shown in eq. (1).

$$\mathrm{x_{iv}} = \begin{cases} 1, & if \text{ cloudlet } i \text{ assigned to machine } v \\ 0, & Otherwise \end{cases} \qquad (1)$$

The cloudlet scheduling problem may be formulated mathematically as:

$$\min \; Z = \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{v=1}}^{\mathrm{m}} \mathrm{e_{iv}\, x_{iv}} \qquad (2)$$

Subject to

$$\sum_{v=1}^{m} x_{iv} = 1, \qquad \forall \, cloudlet\, i \qquad (3)$$

$$\sum_{i=1}^{n} EC_i \, x_{iv} \le L_v \;\; \forall VM \qquad (4)$$

$$\sum_{i=1}^{n} mem_i \, x_{iv} \le Mem_v \qquad \forall VM \qquad (5)$$

The main objective of the mathematical model is formulated, in eq. (2), to reduce the execution time. Indeed, several constraints are formulated to

meet the requirements of cloudlets and not violate the availability of virtual resources. The first constraint, eq. (3), guarantees that each cloudlet i is assigned to exactly one virtual machine $_v$. The second constraint, eq. (4), guarantees that the total requirements of total execution time *EC* of the cloudlets assigned to a *VM* don't exceed the load L_v of that *VM*. The third constraint, eq. (5), guarantees that memory requirements of scheduled cloudlets should not exceed the maximum available memory of running VM.

4. Related Work

Recently, many scheduling methods are developed to address scheduling issue. They may be classified into static and dynamic scheduling [11]. In static scheduling, all cloudlets arrive simultaneously and all cloudlets schedule first before execution. In dynamic scheduling, cloudlets arrive at different time slots, and they schedule are based on the state of VMs. Two kinds of scheduling methods are proposed; heuristic methods and meta-heuristic methods. Heuristic methods use the predictions to achieve a near optimal solution [12-17]. These methods often have low time complexity, but they provide high schedule length. Contrary to heuristic-based, the meta-heuristic methods search the solution space in a direct manner and produce efficient results on the broad domain problems, but these methods have high time complexity. Meta-heuristic algorithms are also called guided-random search-based methods [18-30].

In (CC) environment, the famous heuristic cloudlet-scheduling algorithm is First-Come First-Serves (FCFS) [12]. In FCFS, all cloudlets are queued in a queue and then assigned to computing resources once they become available based on the arrival time. The FCFS is easy to implement but it provides high time complexity. In [13], a heuristic cloudlet-scheduling algorithm called greedy algorithm is presented. This algorithm first orders the arrived cloudlets in descending order according to its lengths. Then, it schedules cloudlets onto the VMs to minimize the finish time. Greedy algorithm provides near optimal solution and has low time complexity. Another heuristic cloudlet-scheduling algorithm, called Min-Min algorithm, is proposed and tuned in [14,15]. The Min-Min algorithm computes minimum completion time of each cloudlet overall VMs. Then, it assigns the cloudlet to VM that achieves minimum completion time. The algorithm iterates until all cloudlets are scheduled. The Min-Min algorithm doesn't consider the system load balancing because it assigns

smaller cloudlets in faster VMs. Another heuristic cloudlet-scheduling algorithm, called Max-Min algorithm, is proposed and tuned in [16,17]. It selects the cloudlet with the longest completion time and assigns it to the VM that gives minimum completion time. Therefore, the Max-Min is more efficient than Min-Min algorithm as it considers the system load balancing [11].

On the other hand, Genetic Algorithm (GA) is most famous technique for guided-random-search-based scheduling techniques [23]. It improves the scheduling results in term of schedule length. However, the time complexity is high. Another metaheuristic cloudlet-scheduling algorithm is Simulated Annealing (SA) [24-26]. It has smaller time complexity than GA. In [26], X. Liu and J. Liu developed new cloudlet scheduling algorithm for (CC) based on SA and greedy. The algorithm uses greedy strategy as initial stage to get near optimal solution and then improve the solution by SA. Other meta-heuristic algorithms developed to solve the scheduling issue are Particle Swarm Optimization (PSO) [27, 28] and Ant Colony Optimization (ACO) [29].

5. Proposed Approach

This section presents a new hybrid approach, called Ant Colony Optimization with Simulated Annealing (ACOSA), for cloudlets scheduling in (CC) environment. The proposed approach combines both the Ant Colony Optimization (ACO) and Simulated Annealing (SA) algorithms. The developed ACOSA approach exploits low time complexity of SA algorithm and the efficient way of the ACO algorithm for searching the near optimal solution. In addition, the ACOSA approach overcomes the drawback of the ACO that occurs in the first stages due to the absence of pheromone by generating an initial solution using the greedy algorithm. Briefly, the new ACOSA approach consists of three stages: initialization stage, ant-colony optimization stage, and simulated annealing stage.

5.1 Initialization Stage

In this stage, the ACOSA generates an initial solution by using the greedy algorithm to improve the efficiency of ACO search algorithm in the second stage. The greedy algorithm sorts the cloudlets according to their

lengths by descending order. Then, it selects the VM that minimizes the finish time of executing the cloudlet.

5.2 Ant Colony Optimization Stage

The ACOSA applies the Ant Colony Optimization (ACO) algorithm to generate new solution. The ACO takes its characteristics from the ability of ant colony to find the shortest path between the food and their nest [29]. The ants interact by laying trails of pheromone. Ants choose their path by using Probability (P) that depends on pheromone trails on the ground. The higher the pheromone trail within a particular direction means higher probability of choosing this direction. The ACO works as shown in Fig. 2.

1. Pheromone initialization

When a cloudlet *i* assigned to a VM_v, a new path is created with pheromone trial τ_{iv}. By using the greedy scheduling solution, the algorithm assigns each ant on a specific VM according to the same order of initial solution. Then, it initializes the pheromone trial for each edge according to eq. (6):

$$\tau_{iv}^{k}(0) = \tau_c + \frac{MIPS(VM_v)}{task_i\_length} \qquad (6)$$

Where, $\tau_{iv}^{k}(0)$ is the pheromone trial value at initial iteration *t=0* for ant k, and τ_c is constant.

2. Virtual machine selection for next cloudlet

Each ant applies the probability *P* in eq. 7 to select a VM for next cloudlet.

$$P^{k}_{iv}(t) = \begin{cases} \dfrac{\tau_{iv}(t) \times \eta_{iv}}{\sum_{s \in allowed_k} \tau_{is}(t) \times \eta_{is}} & \text{if } v \in allowed_k \\ 0 & \text{Otherwise} \end{cases} \qquad (7)$$

Where, τ_{iv} is the pheromone trial of $cloudlet_i$ in VM_v, $allowed_k$ is the available *VMs* of ant_k that are not chosen yet for any cloudlet by the ant. The VMs that are chosen are stored in $tabue_k$. $\eta_{iv} = 1/d_{iv}$ is heuristic information representing the visibility of ant_k at iteration t, where d_{iv} is the expected execution time and transfer time of $cloudlet_i$ at VM_v.

3. Pheromone updating

After each ant creates a path, it updates the local pheromone of this path by eq. 8.

$$\Delta \tau_{iv}^{k}(t) = \begin{cases} \dfrac{Q}{L^{k}(t)} & \text{if } (i,v) \in T^{k}(t) \\ 0 & \text{Otherwise} \end{cases} \qquad (8)$$

Where, $T^{k}(t)$ represents $Tabu_k$ (the collection of VMs that ant_k visited) at iteration t, $L^{k}(t)$ is the expected schedule length of ant_k and Q is a control parameter. After generating new solution, the global pheromone updates by the eq. 9.

$$\tau_{iv}(t+1) = (1-\rho)\tau_{iv}(t) + \Delta \tau_{iv}(t) \qquad (9)$$

Where, $\rho \in [0\text{-}1]$ is the trial volatility coefficient, and $\Delta \tau_{iv}(t)$ is computed by $\Delta \tau_{iv}(t) = \sum_{k=0}^{n} \Delta \tau_{iv}^{k}$.

An Iteration of Ant Colony Optimization (ACO) Algorithm
1. Assign y ants on m VMs according to the initial greedy solution
2. Initialize τ_{iv} for each rout between cloudlet$_i$ and VM$_v$
3. While cloudlet-list is not empty repeat
4. For k=0 to y
5. Ant$_k$ selects a suitable VM$_v$ for the selected cloudlet$_i$ according to P_{iv}^{k} (T)
6. Insert VM$_v$ in Tabu$_k$ and remove it from allowed$_k$
7. Remove the selected_cloudlet from cloudlet_list
8. Update local pheromone
9. End For
10. End while
11. *Update global pheromone*

Figure 2: Ant Colony Optimization Algorithm

5.3 Simulated Annealing Stage

Simulated annealing (SA) is a global optimization technique that attempts to find the lowest point in energy landscape [31]. The idea of this method was derived from how a regular crystalline structure is generated by cooling molten metal slowly. The distinctive feature of the SA algorithm is that it incorporates random jumps to potential new solutions. This

ability is controlled and reduced as the algorithm progresses. Clearly, the SA emulates the physical concepts of temperature and energy to represent and solve the optimization problems.

The objective function of the optimization problem is treated as the energy of a dynamic system while the temperature is introduced to randomize the search for a solution. The state of the dynamic system being simulated is related to the state of the system being optimized. Firstly, the system is started at a high temperature *Temp* and is then slowly cooled through a series of temperature levels. At each level, the algorithm searches for the system equilibrium state through elementary transformations which will be accepted if they reduce the system energy. The probability of a new solution acceptance is $exp(\Delta/Temp)$. It is a function of the temperature and the magnitude of the increasing Δ.

5.4 ACOSA hybrid Approach

The developed ACOSA approach is shown in Figure 3. The algorithm starts by applying the greedy algorithm to obtain an initial solution, and then it computes the energy (schedule length) E_s at the initial solution. After setting the parameters of ACO: the number of ants y, and initial pheromone trial $\tau_{iv,}$ by using the initial solution, the ACOSA algorithm generates a new solution called opt solution by applying the ACO algorithm, and then it calculates energy E_{opt} of the solution. SA algorithm is starting from step 9.

After setting an initial temperature *Temp*, the ACO algorithm is used again to generate a new solution and compute the corresponding energy E_{new}. If the energy E_{new} at the new solution is lower than the current energy E_s, then the new solution is accepted as a current solution. Otherwise, a probability function $exp(-\Delta/T)$ is evaluated to determine whether the new solution may be accepted as a current solution or not, where $\Delta = E_{new} - E_s$. In step 22, the ACOSA algorithm checks if the ACO algorithm works efficiently or not. If E_s of the current solution is less than E_{opt}, the ACO generates a new solution. At this moment, ACOSA algorithm continues by repeating SA algorithm and generating more solutions. Otherwise, the algorithm stops and returns the Opt solution as the best solution. This way decreases the time complexity of the algorithm.

Proposed ACOSA Approach

1. Generate Initial solution by applying the greedy algorithm
2. Set Current solution=initial solution
3. Calculate E_s
4. Initialize number of ants y
5. Initialize t=0
6. Initialize $\tau_{i,v}(t)$ by using current solution
7. Generate Opt_Solution by applying ACO algorithm
8. Calculate E_{opt}
9. Initialize SA parameters, $Temp_0$ and calling rate $\alpha \in [0\text{-}1]$
10. $t=t+1$
11. Generate new solution by applying ACO algorithm
12. Calculate E_{new}
13. Calculate $\Delta = E_{new} - E_s$
14. **If** $\Delta < 0$
15. $E_s = E_{new}$ *and Current Solution=New Solution*
16. **Else**
17. *Generate a random value* $r \in [0\text{-}1]$
18. **If** $(r < exp(-\Delta/Temp))$
19. $E_s = E_{new}$ *and Current Solution=New Solution*
20. **End if**
21. **End if**
22. **If** $E_s < E_{opt}$
23. $E_{opt} = E_s$ *and* Opt_Solution =*Current_Solution*
24. *Set Temp=*$\alpha \times Temp$
25. **If***(Temp >1)*
26. *Go to step (10)*
27. **Else**
28. *Return Current_Solution*
29. **End if**
30. **Else**
31. *Current_Solution*= Opt_Solution
32. *Return Current_Solution*
33. **End if**

Figure 3: Proposed ACOSA Approach for cloudlet Scheduling

5.5 Time Complexity

The time complexity of the proposed ACOSA approach may be calculated as the summation a time complexity of stage 1, 2, and 3. In stage 1, the ACOSA uses the greedy algorithm to generate new solution with time

complexity *O (n log n+ nm).* In stage 2, the ACOSA applies one iteration of the ACO algorithm with time complexity *O(n*×m×y*).* In stage 3, the ACOSA repeats SA to generate new solution by ACO algorithm. This stage has a time complexity O(t_{max} * time complexity of ACO). The overall time complexity of the ACOSA algorithm is *O((n log n+ nm)+ t_{max}* ×*n*×m×y*) = O(n(log n +2m)),* where $t_{max}<10$ *and y=30*.

On the other hand, the time complexity of the ACO algorithm depends on the number of iterations, number of cloudlets, and number of VMs. It has time complexity equal to O(g_{max}×n×m×y), where g_{max} is the maximum number of generations, often $g_{max} \geq 200$. While the time complexity of SA depends on three factors. The first factor is the number of iterations, the second is the number of cloudlets, and the third is number of VMs. It has time complexity *O(itr* ×*n*×*m),* where *itr* is the number of iterations often *itr>600 at temperature 1000 and cooling rate .01*. To obtain a best solution with SA and ACO, *itr* and g_{max} should be very high. However, if the number of iterations increases, time complexity will increase. This makes the scheduling process takes more time to schedule large number of cloudlets.

The developed algorithm achieves low time complexity by avoiding the drawbacks of other algorithms using two strategies:

- It uses the greedy algorithm to generate the initial solution as near optimal solution.
- It stops repeating, when it achieves the optimal solution, so it doesn't depend on the number of iteration. Because the number of iterations is very low on the contrary to SA and ACO algorithms.

5.6 The Oscillation Problem

Because the initial solutions of both the ACO and the SA are generated randomly, these algorithms provide different solutions with different makespans for different running times for the same scheduling problem. This is called the oscillation problem. The oscillation problem causes finding solution with high makespan. No one can guarantee specific solution for the scheduling problem. That means we will find solutions with low makespan and another with high makespan for the same scheduling problem. Because the problems may repeat, the oscillation problem affects on the results of the meta-heuristic algorithms. The proposed ACOSA approach overcomes this problem and provides only

one solution for the same problem at different running times. Figure 4 shows the obtained results of applying three different algorithms 15 times to solve a cloudlet scheduling problem. In this example, the number of cloudlets (n) =100, the number of VMs (m) = 8, and the initial values of the required parameters are: Temp=1000, α=0.01, y=30, ρ=0.5, τc=0.3. we use these initial values because they are excited in [26,29] (the references of related work) . From Figure 4, the new algorithm ACOSA is better than the other algorithms. It has constant makespan =50 Sec, while the makespan of SA algorithm oscillates from 100 Sec to 230 Sec and the makespan of the ACO oscillates from 90 Sec to 176 Sec.

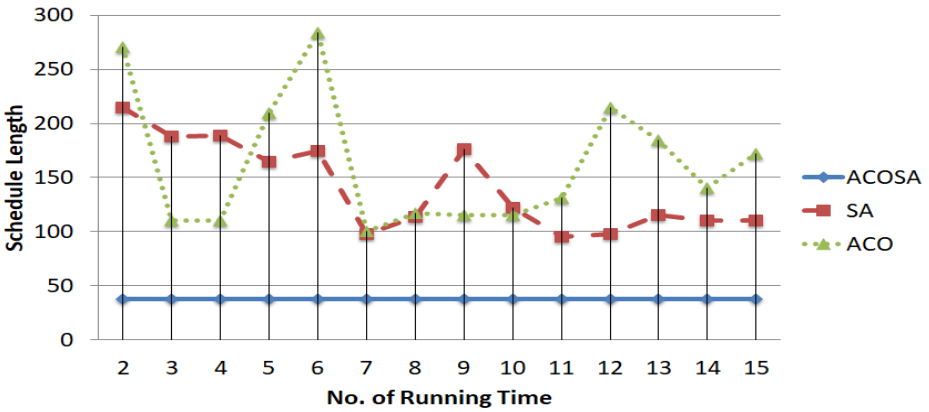

Figure 4: The results oscillation of scheduling 100 cloudlets into 8VMs.

6. Simulation Results

To evaluate the performance of the proposed hybrid cloudlet scheduling approach (ACOSA), the well-known CloudSim is used to simulate the cloud-computing environment. The results of the proposed ACOSA approach are compared with the Simulated Annealing (SA) and the Ant Colony Optimization (ACO) separately in terms of schedule length, load balancing and time complexity.

The simulation environment is a 64-bit windows 7 operating system installed in laptop core i5 with 8 GB RAM. In addition, a list of random independent cloudlets is generated with lengths from 1000 MI to 10,000 MI and a list of VMs is generated with MIPS in the range [100-1000]. The

initial values of the ACOSA parameters are y=30, ρ=0.5, τc=0.3, Temp=1000, α=0.01.

6.1 Schedule length

Schedule length is the execution time at maximum loaded VM. Figures 5, 6, 7 and 8 show schedule length of scheduling different cloudlets by three different algorithms at 2, 4, 8, and 16 VMs respectively. From Figures 5, 6, 7 and 8, the ACOSA is more efficient than SA and ACO algorithms in terms of schedule length. The developed ACOSA has low schedule length than both the SA and the ACO algorithms. It improves schedule length by 29.75% with SA and 12.25% with ACO. This is because; the ACOSA depends on logical steps to achieve the near optimal solution. It does not start with any random solutions like SA and ACO algorithms. In addition, the developed algorithm stops generating new solutions, if the current solution is not improved than the previous solution for 4 iterations.

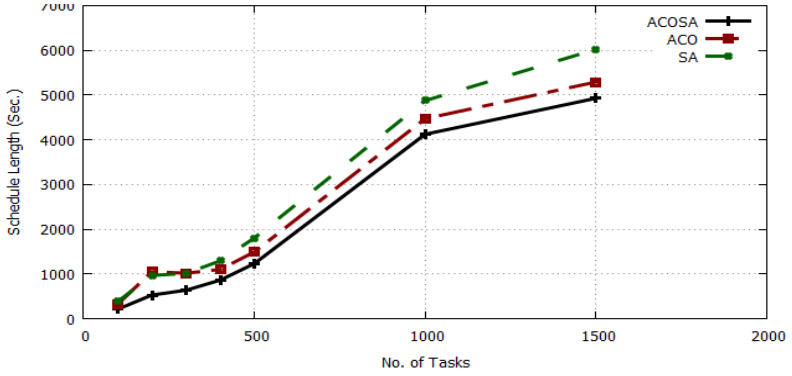

Figure 5: Schedule Length of Different Algorithms vs. Number of Cloudlets for 2 VMs.

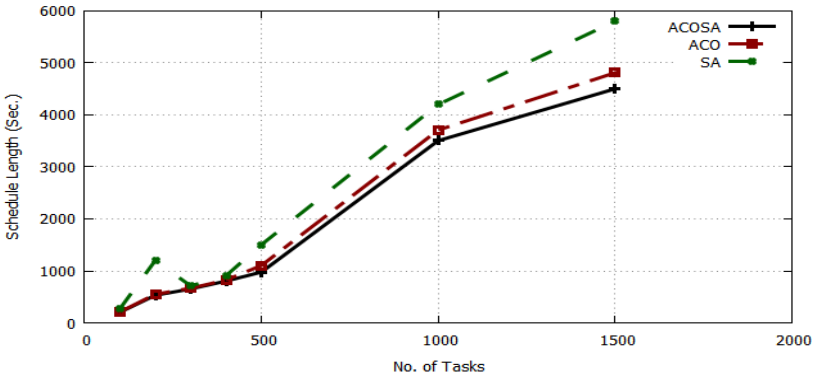

Figure 6: Schedule Length of Different Algorithms vs. Number of Cloudlets for 4 VMs.

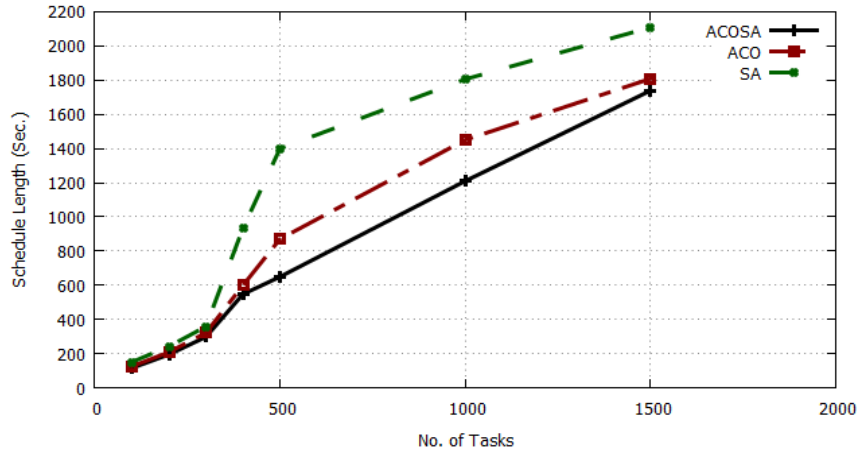

Figure 7: Schedule Length of Different Algorithms vs. Number of Cloudlets for 8 VMs.

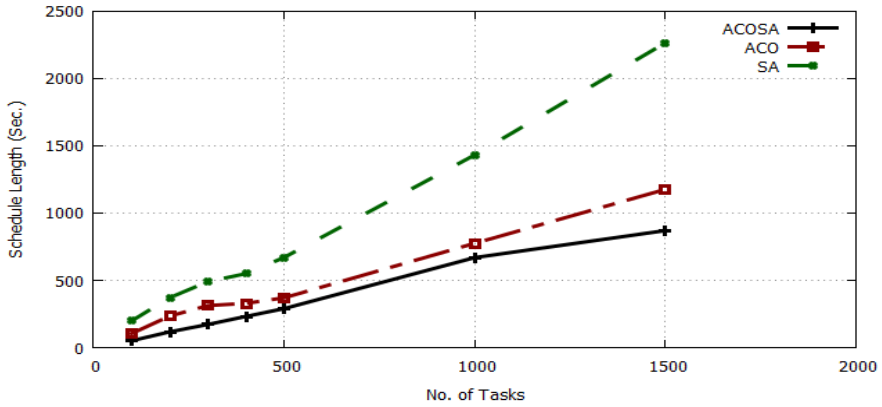

Figure 8: Schedule Length of Different Algorithms vs. Number of Cloudlets for 16 VMs.

6.2 Balancing Degree

Balancing Degree (BD) measures the degree of workload distribution of cloudlets on the available VMs. To calculate this ratio, let define an eclecticism solution by the ideal solution that achieves the lowest schedule length. The system can achieve the lowest schedule length (Best solution) if and only if the next conditions are achieved:

i. VMs execute number of cloudlets that have MI per second exact equal the amount of MIPS or multiples for those VMs.
ii. The finishing times for all VM are equal, after scheduling all cloudlets.

If the system achieves the two conditions, the system will be balanced with 100% balancing degree.

Eclecticism Solution Length (ESL) can be calculated by:

$$ESL(s) = Total\ MI\ /\ Total\ MIPS$$

ESL is less than the schedule length. We can compute BD from the following equation: $BD(s) = ESL\ /\ Makespan$

Figures 9, 10, 11 and 12 show the balancing degree of scheduling different cloudlets by three different algorithms at 2, 4, 8, and 16 VMs respectively. According to the definition of BD, we find that the ACOSA has higher BD ratio than SA and ACO algorithms, because it achieves lower schedule length. The developed ACOSA improves BD ratio with 36.36% than SA and 12.13% than ACO algorithms.

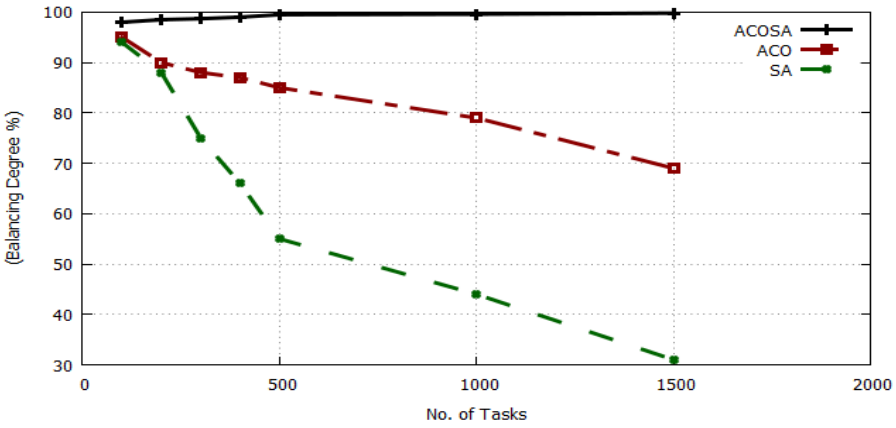

Figure 9: Balancing Degree by Different Algorithms on 2 VMs.

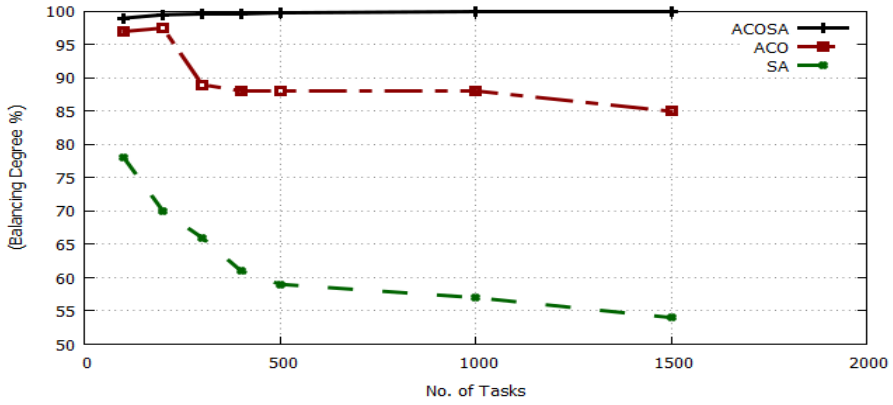

Figure 10: Balancing Degree by Different Algorithms on 4 VMs.

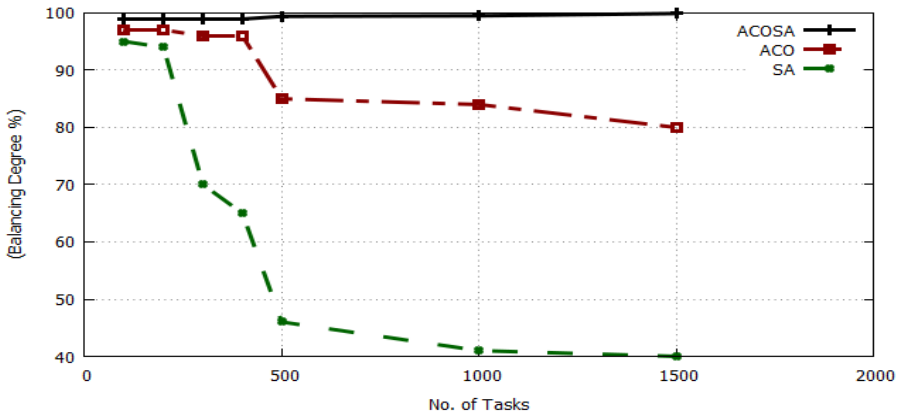

Figure 11: Balancing Degree by Different Algorithms on 8 VMs.

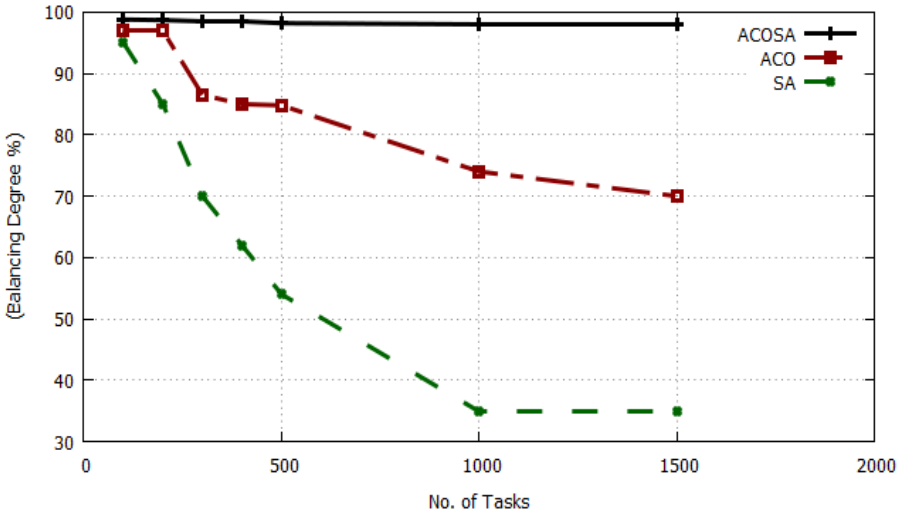

Figure 12: Balancing Degree by Different Algorithms considering 16 VMs.

6.3 Computation Time complexity

Figures 13, 14, 15 and 16 show running time of scheduling different cloudlets by three different algorithms at 2, 4, 8, and 16 VMs respectively. From Figures 4, 5, 6 and 7, the ACOSA has running time than SA and ACO algorithms. The developed ACOSA decreases running time 64.5% than SA and 98.7% than ACO algorithms.

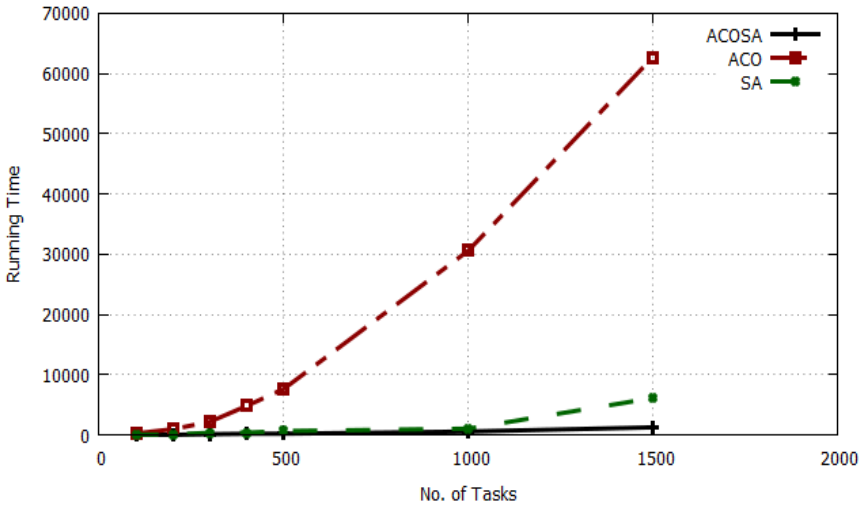

Figure 13: Computation Time of Different Algorithms for 2 VM.

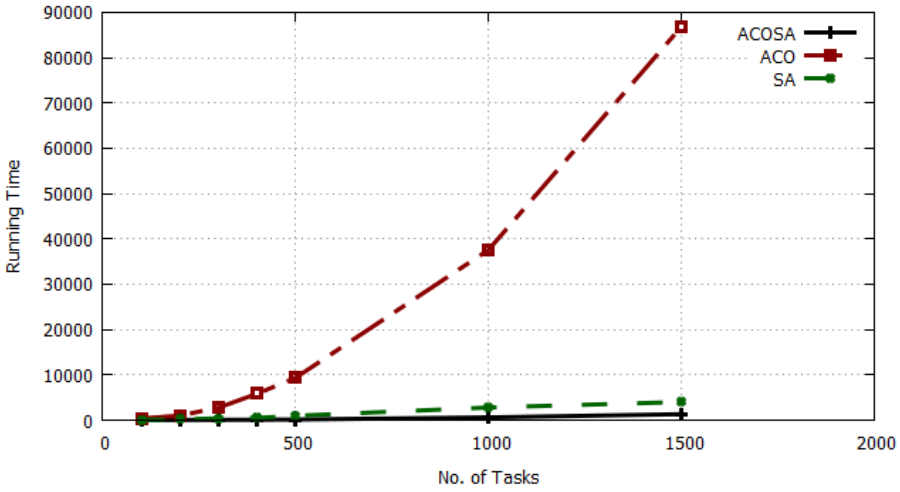

Figure 14: Computation Time of Different Algorithms for 4 VM.

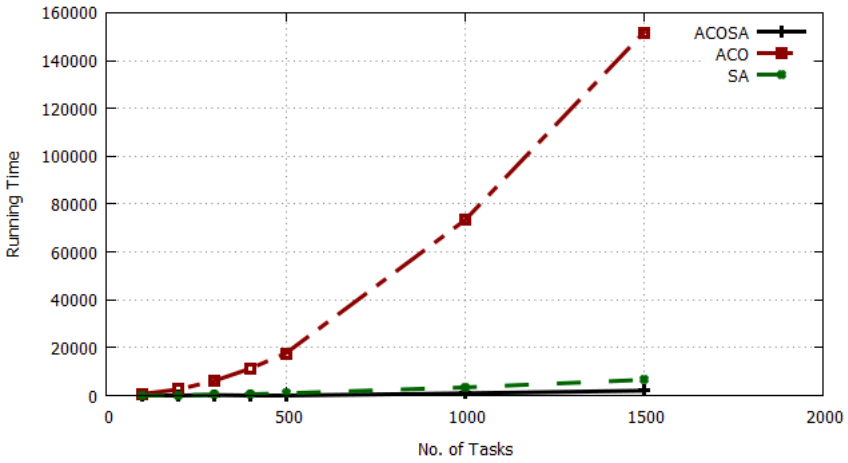

Figure 15: Computation Time of Different Algorithms for 8 VM.

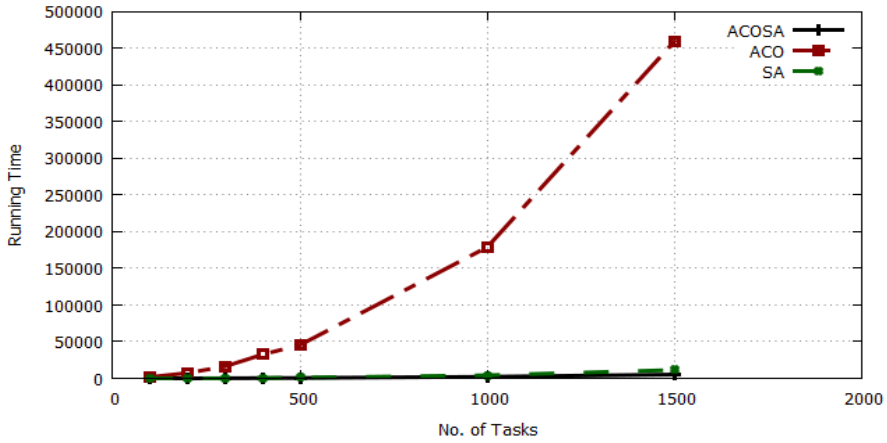

Figure 16: Computation Time of Different Algorithms for 16 VM.

7. Conclusions

In this paper, a new hybrid cloudlet scheduling approach called ACOSA is proposed for (CC) environment considering both the resources availability and cloudlets requirements. The proposed ACOSA approach enhances the overall system performance. It achieves four goals: the first is minimizing schedule length of the cloudlets, the second is keeping the system in high balancing degree, the third is minimizing the time complexity of the scheduling algorithm, and the fourth is solving the results oscillation problem. By comparing the new ACOSA with the Simulated Annealing

(SA) and Ant Colony Optimization (ACO) algorithms, the new approach is more efficient than those algorithms. The experimental results show that the ACOSA achieves lower schedule length than the SA and the ACO algorithms. It decreases the schedule length by 29.75% with SA and 12.25% with ACO. The ACOSA provides higher load balancing degree. It improves the balancing degree ratio by 36.36% than SA and 12.13% than ACO algorithms. In addition, the ACOSA achieves low computation time complexity. It decreases running time 64.5% than SA and 98.7% than ACO algorithms.

References

[1] https://www.ibm.com/cloud-computing/learn-more/what-is-cloud-computing/ accessed at 21 June 2018.

[2] Hamdaqa, Mohammad, and L. Tahvildari. "Cloud computing uncovered: a research landscape." In Advances in Computers, vol. 86, pp. 41-85. Elsevier, 2012.

[3] L. Mei, W.K. Chan, and T.H. Tse, "A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues", Proceedings of the APSCC 2008, pp. 464-469, 2008.

[4] H. Yuan, J. Bi, W. Tan and B. Li, "Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds", IEEE Transactions on Automation Science and Engineering, Vol. 14, pp. 337-348, 2017.

[5] A. Abbasi-Tadi, M. Khayyambashi, and H. Khosravi-Farsani, "Data center task scheduling through Biogeography-Based Optimization model with the aim of reducing makespan", The 6th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 41 - 47, 2016.

[6] A. A. Nasr, N. A. EL-Bahnasawy, and A. El-Sayed, "task scheduling optimization in heterogeneous distributed system", International Journal of Advanced Computer Science and Applications(IJAACSAA), Vol. 7, No. 4, pp. 88-96, 2014.

[7] A. A. Nasr, and S. A. Elbooz. "Scheduling Strategies in Cloud Computing: Methods and Implementations." (2018).

[8] A. A. Nasr, N. A. EL-Bahnasawy, and A. El-Sayed, "Performance Enhancement of Scheduling Algorithm in Heterogeneous Distributed Computing Systems", International Journal of Advanced Computer Science and Applications(IJAACSAA), Vol. 6, No. 5, pp. 88-96, 2015.

[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, and C. A. F. De Rose, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software Practice and Experience, vol. 41, no. 1, pp. 23–50, August 2010.

[10] A. A. Nasr, N. A. EL-Bahnasawy, G. Attiya and A. El-Sayed, "Using the TSP Solution Strategy for Cloudlet Scheduling in Cloud Computing", Journal of Network and Systems Management, pp. 1-22, 2018.

[11] T. Mathew, K. Sekaran, and J. Jose, "Study and Analysis of Various Task Scheduling Algorithms in the Cloud Computing Environment", Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 658-664, 2014.

[12] T. Chatterjee, VK. Ojha, M. Adhikari, S.Banerjee, U. Biswas, and V. Snáše, "Design and Implementation of an Improved Datacenter Broker Policy to Improve the QoS of a Cloud", Proceedings of the 5th International Conference on Innovations in Bio-Inspired Computing and Applications IBICA, pp. 281-290, 2014.

[13] Z. Zhong, K. Chen, X Zhai, and S. Zhou, "Virtual machine-based task scheduling algorithm in a cloud computing environment", Tsinghua and Technology, pp. 660-667, 2016.

[14] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing", National Conference on Parallel Computing Technologies (PARCOMPTECH), October 2013, pp. 1-8.

[15] T. Kokilavani, and GA DI. "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", International Journal of Computer Applications, Vol. 20, No. 2, PP. 43-49, 2011.

[16] K. Etminani, and M. Naghibzadeh, "A Min-Min Max-Min selective algorihtm for grid task scheduling", IEEE/IFIP International Conference in Central Asia on Internet, pp.1-7, Tashkent, Uzbekistan ,September 2007.

[17] S. Devipriya, and C. Ramesh, "Improved Max-min heuristic model for task scheduling in cloud", Proceedings of the International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), IEEE, pp. 883-888, Chennai, India , December 2013.

[18] SH. Adil, K. Raza, U. Ahmed, S.S.A Ali, and M. Hashmani, "Cloud task scheduling using nature inspired meta-heuristic algorithm", International Conference on Open Source Systems & Technologies (ICOSST), pp. 158-164, Lahore, IEEE, Pakistan, Dec 2015.

[19] M.A Tawfeek, A El-Sisi, A. E. Keshk, and F A Torkey, " Cloud task scheduling based on ant colony optimization" In Computer Engineering & Systems (ICCES), Des. 8th International Conference on (pp. 64-69). IEEE, Cairo, Egypt , Nov. 2013.

[20] S Sindhu, S Mukherjee " A genetic algorithm based scheduler for cloud environment" In Computer and Communication Technology (ICCCT), 20 (pp. 23-27). IEEE, Allahabad, India , Sep 2013.

[21] M. Agarwal, and G. M. S. Srivastava, "A genetic algorithm inspired task scheduling in cloud computing", Proceedings of the International

Conference on Communication and Automation (ICCCA),IEEE, Noida, India April 2016.

[22] I. Kar, R.N.R. Parida, and H. Das, "Energy Aware Scheduling using Genetic Algorithm in Cloud Data Centers", Proceedings of the International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE, Chennai, India, 2016.

[23] S. Singh, and M. Kalra, "Scheduling of Independent Tasks in Cloud Computing using Modified Genetic algorithm," Proceedings of the International Conference on Computational Intelligence and Communication Networks (CCIN), IEEE, pp.565-569, Bhopal, India, November 2014.

[24] M. Houshmand, E Soleymanpour, H Salami, M Amerian, and H Deldari, "Efficient Scheduling of Task Graphs to Multiprocessors Using a Combination of Modified Simulated Annealing and List Based Scheduling", Proceedings of the 3rd International Symposium on Intelligent Information Technology and Security Informatics (IITSI), IEEE, Jinggangshan, China, April 2010.

[25] H. Bonan, W. Xia, Y. Zhang, J. Zhang, Q. Zou, F. Yan, and L. Shen. "A task assignment algorithm based on particle swarm optimization and simulated annealing in Ad-hoc mobile cloud." In Wireless Communications and Signal Processing (WCSP), 2017 9th International Conference on, pp. 1-6. IEEE, Nanjing, China, December 2017.

[26] X. Liu, and J. Liu, "A Task Scheduling on Simulated Annealing Algorithm in Cloud Computing", International Journal of Hybrid Information Technology (IJHIT), Vol. 9, No. 6, pp. 403-412, 2016.

[27] K. K. Raja, P. Sengottuvelan, and J. Shanthini. "A hybrid approach of genetic algorithm and multi objective PSO task scheduling in cloud computing." Asian Journal of Research in Social Sciences and Humanities 7, no. 3 : 1260-1271, 2017.

[28] A. Awad, N. EL-Hefnawy, and H. Abdel_Kader, "Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environment", International Conference on Communication, Management and Information Technology (ICCMIT), Elsevier, pp. 920-929, 2015.

[29] Liu, C. Y., Zou, C. M., & Wu, P "A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing" In Distributed Computing and Applications to Business, Engineering and Science (DCABES), IEEE , pp. 68-72, November 2014.

[30] J. Xu, A. Lam, and V. Li "Chemical reaction optimization for the grid scheduling Problem", Proceedings of the International Conference on Communications, ICC, pp. 1–5, South Africa, May 2010.

[31] E. Aarts, J. Korst, Simulated Annealing and Boltzmann Machines, Wiley, New York, 1989.

الملخص باللغة العربية

يقدم البحث خوارزمه مقترحة تسمي ACOSA لحل مشكلة الجدولة وتحسين اداء الحوسبة السحابية والتغلب علي مشكلة التذبذب الموجوده بالخوارميات الاخري . وتتكون الخوارزميه ACOSA من دمج الخوارزميات SA, ACO و ذلك لتحسين جوده الحلول وتقليل وقت التنفيذ. تم استخدام أداة CloudSim لقياس اداء الخوارميه المقترحة مع الخوارزميات الموجودة من خلال قياس الوقت النهائي للتنفيذ و قياس درجة اتزان الأحمال ووقت تشغيل الخوارزوميه. من النتائج نجد ان الخوارميه المقترحة تتفوق علي الخوارميات ACO , SA في الوقت النهائي للتنفيذ و درجه توازن الاحمال و وقت التشغيل. حيث يقلل الوقت النهائي للتنفيذ بنسبه 29.75% بالنسبه SA و 12% بالنسبه ACO ويزيد درجة اتزان الاحمال بنسبة 36% افضل من SA و 12% افضل من ACO.