Cloud Task Scheduling Based on Ant Colony Optimization

Medhat Tawfeek, Ashraf El-Sisi, Arabi Keshk and Fawzy Torkey
Faculty of Computers and Information, Menoufia University, Egypt

Abstract: *Cloud computing is the development of distributed computing, parallel computing and grid computing, or defined as the commercial implementation of these computer science concepts. One of the fundamental issues in this environment is related to task scheduling. Cloud task scheduling is an NP-hard optimization problem and many meta-heuristic algorithms have been proposed to solve it. A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks. In this paper, a cloud task scheduling policy based on Ant Colony Optimization (ACO) algorithm compared with different scheduling algorithms First Come First Served (FCFS) and Round-Robin (RR), has been presented. The main goal of these algorithms is minimizing the makespan of a given tasks set. ACO is random optimization search approach that will be used for allocating the incoming jobs to the virtual machines. Algorithms have been simulated using cloudsim toolkit package. Experimental results showed that cloud task scheduling based on ACO outperformed FCFS and RR algorithms.*

Keywords: *Cloud computing, task scheduling, makespan, ACO, cloudsim.*

Received July 3, 2013; accepted September 2, 2013; published online April 23, 2014

1. Introduction

Cloud computing is associated with a new paradigm for provisioning different computing resources, usually addressed from three fundamental aspects: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [20]. Due to fast growth of cloud computing in the IT landscape, several definitions have emerged. The cloud computing can be defined as a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers [8]. With the support of virtualization technology cloud platforms enable enterprises to lease computing power in the form of virtual machines to users [6]. Because hundreds of thousands of Virtual Machines (VMs) are used, it is difficult to manually assign tasks to computing resources in clouds [17]. So, we need an efficient algorithm for task scheduling in the cloud environment.

A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks [7]. Therefore, a dynamic task scheduling algorithm, such as Ant Colony Optimization (ACO), is appropriate for clouds. ACO algorithm is a random search algorithm [4]. This algorithm uses a positive feedback mechanism and imitates the behaviour of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths travelled. Many researchers used ACO to solve NP-hard problems such as travelling salesman problem, graph colouring problem, vehicle routing problem and scheduling problem [5, 8]. In this paper, we use ACO algorithm to find the optimal resource allocation for tasks in the dynamic cloud system to minimize the makespan of tasks on the entire system. Then, this scheduling strategy was simulated using the cloudsim toolkit package. Experimental results compared to First Come First Served (FCFS) and Round Robin (RR) showed the ACO algorithm satisfies expectation. The organization of paper is as following: Section 2 introduces background and scans the related work. Cloudsim toolkit is presented in section 3. Section 4 covers the basic ACO and the details of cloud scheduling based ACO algorithm. The implementation and simulation results are seen in section 5. Finally, section 6 concludes this paper.

2. Background and Related Work

2.1. Cloud Computing Environment

Cloud computing is a virtual pool of resources which are provided to users. It gives users virtually unlimited pay-per-use computing resources without the burden of managing the underlying infrastructure. The goal of cloud computing service providers is to use the resources efficiently and gain maximum profit [16]. This leads to task scheduling as a core and challenging issue in cloud computing. Cloud has an extra layer called virtualization layer. This layer acts as a creation, execution, management and hosting environment for application services [11]. The modelled VMs in the above virtual environment are contextually isolated but, still they need to share computing resources-

processing cores, system bus etc., [21]. Hence, the amount of hardware resources available to each VM is constrained by the total processing powers such CPU, the memory and system bandwidth available within the host [21].

2.2. Combinatorial Optimization Problem

In combinatorial optimization problems, we are looking for an object from a finite or possibly countably infinite set. This object is typically an integer number, a subset, a permutation or a graph structure [15]. A combinatorial optimization problem $P=(S, f)$ can be defined by:

- A set of variables $X=\{x_1, x_2, \ldots, x_n\}$.
- Variable domains $D_1, \ldots, D_n$.
- Constraints among variables.
- An objective function f to be minimized where, $f: D_1 * \ldots * D_n \rightarrow R^+$.

The set of all possible feasible assignments is: $S=\{s=\{(x_1,v_1), \ldots, (x_n,v_n)\} | v_i \in D_i, s$ satisfies all the constraints$\}$ S is usually called a search (or solution) space, as each element of the set can be seen as a candidate solution. To solve a combinatorial optimization problem one has to find a solution $s^* \in S$ with minimum objective function value [15]. Examples for, combinatorial optimization problems are the Travelling Salesman Problem (TSP), the Quadratic Assignment Problem (QAP), time tabling and scheduling problems. Due to the practical importance of combinatorial optimization problems, many algorithms to tackle them have been developed. These algorithms can be classified as either complete or approximate algorithms. Complete algorithms are guaranteed to find for every finite size instance of a combinatorial optimization problem an optimal solution in bounded time. In approximate methods we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time especially for combinatorial optimization problems that are NP-hard [18]. Among the basic approximate methods we usually distinguish between constructive methods and local search methods. Constructive algorithms generate solutions from scratch by adding components to an initially empty partial solution until a solution is complete. Local search algorithms start from some initial solution and iteratively try to replace the current solution by a better solution in an appropriately defined neighbourhood of the current solution [15]. In past, four decades, a new kind of approximate algorithm has emerged which basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. This class of algorithms includes ACO, simulated annealing, tabu search and others [18]. A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions [18].

2.3. Related Work

Millions of user share cloud resources by submitting their computing task to the cloud system. Scheduling these millions of task is a challenge to cloud computing environment. Optimal resource allocation or task scheduling in the cloud should decide optimal number of systems required in the cloud so that the total cost is minimized. Cloud service scheduling is categorized at user level and system level [6]. At user level scheduling deals with problems raised by service provision between providers and customers [17, 21]. The system level scheduling handles resource management within data centers [8, 11, 16]. A novel approach of heuristic-based request scheduling at each server, in each of the geographically distributed data centers, to globally minimize the penalty charged to the cloud computing system is proposed in [1]. A new fault tolerant scheduling algorithm MaxRe is proposed in [23]. This algorithm incorporates the reliability analysis into the active replication schema and exploits a dynamic number of replicas for different tasks. Scheduling based genetic algorithm is proposed in [12, 14, 22]. This algorithms optimizes the energy consumption, carbon dioxide emissions and the generated profit of a geographically distributed cloud computing infrastructure. The QoS Min-Min scheduling algorithm is proposed in [10]. An optimized algorithm for VM placement in cloud computing scheduling based on multi-objective ant colony system algorithm in cloud computing is proposed in [8]. Scheduling in grid environment based ACO algorithms are proposed in [13, 14, 19]. The existing scheduling techniques in clouds, consider parameter or various parameters like performance, makespan, cost, scalability, throughput, resource utilization, fault tolerance, migration time or associated overhead. In this paper, cloud task scheduling based ACO approach has been presented for allocation of incoming jobs to VMs considering in our account only makespan to help in utilizing the available resources optimally, minimize the resource consumption and achieve a high user satisfaction.

3. Cloudsim

Simulation is a technique where a program models the behaviour of the system (CPU, network etc.,) by calculating the interaction between its different entities using mathematical formulas, or actually capturing and playing back observations from a production system [3]. Cloudsim is a framework developed by the GRIDS laboratory of university of Melbourne which enables seamless modelling, simulation and

experimenting on designing cloud computing infrastructures [3].

3.1. Cloudsim Characteristics

Cloudsim can be used to model datacenters, host, service brokers, scheduling and allocation policies of a large scaled cloud platform. Hence, the researcher has used cloudsim to model datacenters, hosts, VMs for experimenting in simulated cloud environment [9]. Cloud supports VM provisioning at two levels:

1. At the host level: It is possible to specify how much of the overall processing power of each core will be assigned to each VM known as VM policy Allocation.
2. At the VM level: The VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine known as VM Scheduling [9].

In this paper, the ACO algorithm will be used for allocation of incoming batch jobs to VMs at the VM level (VM Scheduling). All the VMs in a data center not necessary have a fixed amount of processing power but, it can vary with different computing nodes, and then to these VMs of different processing powers, the tasks/ requests (application services) are assigned or allocated to the most powerful VM and then to the lowest and so on. Hence, the performance parameter such as overall makespan time is optimized (increasing resource utilization ratio) and the cost will be decreased.

3.2. Cloudsim Data Flow

Each datacenter entity registers with the Cloud Information Service registry (CIS). CIS provides database level match-making services; it maps user requests to suitable cloud providers. The data center broker consults the CIS service to obtain the list of cloud providers who can offer infrastructure services that match application's quality of service, hardware and software requirements. In the case match occurs the broker deploys the application with the cloud that was suggested by the CIS [3].

3.3. The Cloudsim Platform

The main parts of cloudsim that are related to our experiments in this paper and the relationship between them are shown in Figure 1.

- *CIS*: It is an entity that registers data center entity and discovers the resource.
- *Data Center*: It models the core infrastructure-level services (hardware), which is offered by cloud providers. It encapsulates a set of compute hosts that can either be homogeneous or heterogeneous.
- *Data Center Broker*: It models a broker, which is responsible for mediating negotiations between SaaS and cloud providers.
- *VM Allocation*: A provisioning policy which is run in data center level helps to allocate VMs to hosts.
- *VM Scheduler*: This is an abstract class implemented by a host component that models the policies (space-shared, time-shared) required for allocating processor cores to VMs. It is run on every host in data center.
- *Host*: It models a physical server.
- *VM*: It models a VM which is run on cloud host to deal with the cloudlet.
- *Cloudlet*: It models the cloud-based application services.
- *Cloudlet Scheduler*: This abstract class is extended by the implementation of different policies that determine the share (space-shared, time-shared) of processing power among cloudlets in a VM [9].

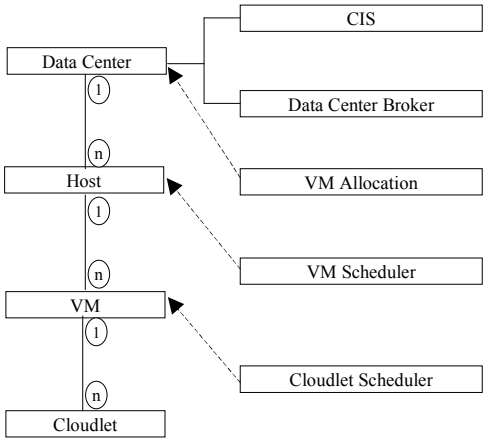


Figure 1. Main parts of cloudsim related to our experiments.

4. Cloud Scheduling Based ACO

The basic idea of ACO is to simulate the foraging behaviour of ant colonies. When an ants group tries to search for the food, they use a special kind of chemical to communicate with each other. That chemical is referred to as pheromone. Initially, ants start search their foods randomly. Once the ants find a path to food source, they leave pheromone on the path. An ant can follow the trails of the other ants to the food source by sensing pheromone on the ground. As this process continues, most of the ants attract to choose the shortest path as there have been a huge amount of pheromones accumulated on this path [4]. The advantages of the algorithm are the use of the positive feedback mechanism, inner parallelism and extensible. The disadvantages are overhead and the stagnation phenomenon, or searching for to a certain extent, all individuals found the same solution exactly, can't further search for the solution space, making the algorithm converge to local optimal solution [4]. It is

clear that an ACO algorithm can be applied to any combinatorial problem as far as it is possible to define:

1. Problem representation which allows ants to incrementally build/ modify solutions.
2. The heuristic desirability η of edges.
3. A constraint satisfaction method which forces the construction of feasible solutions.
4. A pheromone updating rule which specifies how to modify pheromone trail τ on the edges of the graph.
5. A probabilistic transition rule of the heuristic desirability and of pheromone trail [2].

In this section, cloud task scheduling based ACO algorithm will be proposed. Decreasing the makespan of tasks is the basic ideas from the proposed method.

1. *Problem Representation*: The problem is represented as a graph $G=(N, E)$ where the set of nodes N represents the VMs and tasks and the set of edges E the connections between the task and VM as shown in Figure 2. All ants are placed at the starting VMs randomly. During an iteration ants build solutions to the cloud scheduling problem by moving from one VM to another for next task until they complete a tour (all tasks have been allocated). Iterations are indexed by t, $1< t< tmax$, where $tmax$ is the maximum number of iterations allowed.

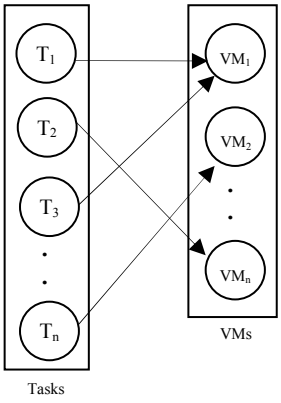


Figure 2. Problem representation of task scheduling based ACO.

2. *Heuristic Desirability*: A very simple heuristic is used the inverse of expected execution time of the task i on VM_j.
3. *Constraint Satisfaction*: The constraint satisfaction method is implemented as a simple, short-term memory of the visited VM, in order to, avoid visiting a VM more than once in one ACO procedure and minimize time of the assigned couplings (task and VM).
4. *Pheromone Updating Rule*: It is the one typical of ant system as shown in Equations 3, 4, 5, 6 and 7. Pheromone evaporates on all edges and new pheromone is deposited by all ants on visited edges; its value is proportional to the quality of the solution built by the ants.
5. *Probabilistic Transition Rule*: The probabilistic transition rule, called random proportional, is the one typical of ant system as shown in Equation 1.

The pseudo code of the proposed ACO algorithm and scheduling based ACO algorithm are shown in Algorithms 1 and 2 respectively.

The main operations of the ACO procedure are initializing pheromone, choosing VM for next task and pheromone updating as following:

Algorithm 1: ACO algorithm

Input: List of Cloudlet (Tasks) and List of VMs
Output: The best solution for tasks allocation on VMs Steps:

1. Initialize:
Set Current_iteration_t=1.
Set Current_optimal_solution=null.
Set Initial value $\tau_{ij}(t)=c$ for each path between tasks and VMs.
2. Place m ants on the starting VMs randomly.
3. For k:=1 to m do
Place the starting VM of the k-th ant in $tabu_k$.
Do ants_trip while all ants don't end their trips
Every ant chooses the VM for the next task according to Equation 1.
Insert the selected VM to $tabu_k$.
End Do
4. For k:=1 to m do
Compute the length L_k of the tour described by the k-th ant according to Equation 4.
Update the current_optimal_solution with the best founded solution.
5. For every edge (i, j), apply the local pheromone according to Equation 5.
6. Apply global pheromone update according to Equation 7.
7. Increment Current_iteration_t by one.
8. If (Current_iteration_t < t_{max})
Empty all tabu lists.
Goto step 2
Else
Print current_optimal_solution.
End If
9. Return

Algorithm 2: Scheduling based ACO algorithm

Input: Incoming Cloudlets and VMs List
Output: Print "scheduling completed and waiting for more Cloudlets"Steps:

1. Set Cloudlet List=null and temp_List_of_Cloudlet=null
2. Put any incoming Cloudlets in Cloudlet List in order of their arriving time
3. Do ACO_P while Cloudlet List not empty or there are more incoming Cloudlets
Set n= size of VMs list
If (size of Cloudlet List greater than n)
Transfer the first arrived n Cloudlets from Cloudlet List and put them on temp_List_of_Cloudlet
Else
Transfer all Cloudlets.from Cloudlet List and put them on temp_List_of_Cloudlet
End If
Execute ACO procedure with input temp_List_of_Cloudlet and n
End Do
4. Print "scheduling completed and waiting for more Cloudlets"
5. Stop

4.1. Initializing Pheromone

The amount of virtual pheromone trail $\tau_{ij}(t)$ on the edge connects task i to VM_j. The initial amount of pheromone on edges is assumed to be a small positive

constant τ_0 (homogeneous distribution of pheromone at time $t=0$).

4.2. VM Choosing Rule for Next Task

During an iteration of the ACO algorithm each ant k, $k=1, \ldots, m$ (m is the number of the ants), builds a tour executing n (n is number of tasks) steps in which a probabilistic transition rule is applied. The k-ant chooses VM_j for next task i with a probability that is computed by Equation 1.

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}*[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}*[\eta_{is}]^{\beta}} & if\ j\in allowed_k\\ 0 & otherwise\end{cases} \quad (1)$$

Where, $\tau_{ij}(t)$ shows the pheromone concentration at the t time on the path between task i and VM_j, allowed $k=\{0,1,\ldots,n-1\}$-$tabu_k$ express the allowed VMs for ant k in next step and $tabu_k$ records the traversed VM by ant k, and $\eta_{ij}=1/d_{ij}$ is the visibility for the t moment, calculated with heuristic algorithm and d_{ij} which expresses the expected execution time and transfer time of the task i on VM_j can be computed with Equation 2.

$$d_{ij}=\frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j}+\frac{InputFileSize}{VM\_bw_j} \quad (2)$$

Where, TL_Task_i is the total length of the task that has been submitted to VM_j, Pe_num_j is the number of VM_j processors, Pe_mips_j is the MIPS of each processor of VM_j, $InputFileSize$ is the length of the task before execution and VM_bw_j is the communication bandwidth ability of the VM_j. Finally, the two parameters α and β in Equation 1 are used to control the relative weight of the pheromone trail and the visibility information respectively.

4.3. Pheromone Updating

After the completion of a tour, each ant k lays a quantity of pheromone $\Delta\tau_{ij}^{k}(t)$ computed by Equation 3 on each edge (i, j) that it has used.

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L^{k}(t)} & if\ (i,j)\in T^{k}(t)\\ 0 & if\ (i,j)\notin T^{k}(t)\end{cases} \quad (3)$$

Where, $T^k(t)$ is the tour done by ant k at iteration t, $L^k(t)$ is its length (the expected makespan of this tour) that is computed by Equation 4 and Q is a adaptive parameter.

$$L^{k}(t)=\arg\max_{j\in J}\{sum_{i\in IJ}(d_{ij})\} \quad (4)$$

Where, ij is the set of tasks that assigned to the VM_j. After each iteration pheromone updating which is applied to all edges is refreshed by Equation 5.

$$\tau_{ij}(t)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \quad (5)$$

Where, ρ is the trail decay, $0<\rho<1$ and $\Delta\tau_{ij}(t)$ is computed by Equation 6.

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta_{\tau_{ij}}^{k}(t) \quad (6)$$

When all ants complete a traverse, an elitist is an ant which reinforces pheromone on the edges belonging to the best tour found from the beginning of the trial (T^+), by a quantity Q/L^+, where L^+ is the length of the best tour (T^+). This reinforcement is called global pheromone update and computed by Equation 7.

$$\tau_{ij}(t)=\tau_{ij}(t)+\frac{Q}{L^{+}}\quad if\ (i,j)\in T^{+} \quad (7)$$

5. Implementation and Experimental Results

We assume that tasks are mutually independent i.e., there is no precedence constraint between tasks and tasks are not preemptive and they cannot be interrupted or moved to another processor during their execution.

5.1. Parameters Setting of Cloudsim

The experiments are implemented with 10 Datacenters with 50 VMs and 100-1000 tasks under the simulation platform. The length of the task is from 1000 Million Instructions (MI) to 20000 MI. The parameters setting of cloud simulator are shown in Table 1.

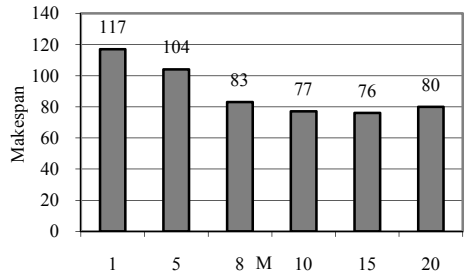
Table 1. Parameters setting of cloudsim.

Entity Type	Parameters	Value
Task (cloudlet)	Length of Task	1000-20000
	Total Number of Task	100-1000
Virtual Machine	Total Number of VMs	50
	MIPS	500-2000
	VM Memory(RAM)	256-2048
	Bandwidth	500-1000
	Cloudlet Scheduler	Space_shared and Time_shared
	Number of PEs Requirement	1-4
Data Center	Number of Datacenter	10
	Number of Host	2-6
	VM Scheduler	Space_shared and Time_shared

5.2. ACO Parameters Evaluation and Setting

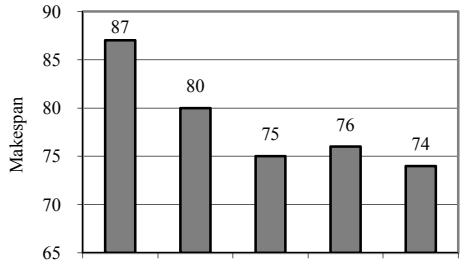
We implemented the ACO algorithm and investigated their relative strengths and weaknesses by experimentation. The parameters (α, β, p, t_{max}, m the number of ants and Q) considered here are those that affect directly or indirectly the computation of the algorithm. We tested several values for each parameter while all the others were held constant on 100 tasks. The default value of the parameters was $\alpha=1$, $\beta=1$, $\rho=0.5$, $Q=100$, $t_{max}=150$ and $m=8$. In each experiment only one of the values was changed, The values tested were: $\alpha\in\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, $\beta\in\{0, 0.5, 1.5, 2, 2.5, 3\}$, $\rho\in\{0,0.1,0.2,0.3,0.4,0.5\}$, $Q\in\{1,100,500, 1000\}$, $t_{max}\in\{50, 75, 100, 150\}$ and $m\in\{1, 5, 8, 10, 15, 20\}$. We also use the time in the cloudSim to record the makespan. The ACO performance for

different values of parameters (α, β, p, t_{max}, m the number of ants and Q) has been evaluated. The ACO performance for different values of parameters (m: The number of ants, t_{max}, Q, ρ, α and β) are shown from Figures 3 to 8. It can be seen that the best value of α is 0.3, the best value of β is 1, the best value of ρ is 0.4, the best value of Q is 100, the best value of t_{max} is 150 and the best values of m is 10. In the following experiments we select the best value for α, β, ρ, Q and m parameters but, the value 100 is selected for the t_{max} parameter to reduce the overhead of the ACO algorithm. Table 2 shows the selected best parameters of ACO.



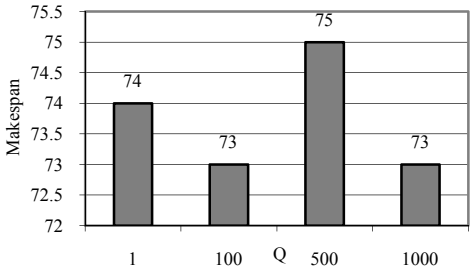
α=1, β=1, r=0.5, Q=100, and t_{max}=200.

Figure 3. ACO performance for different values of ant numbers.



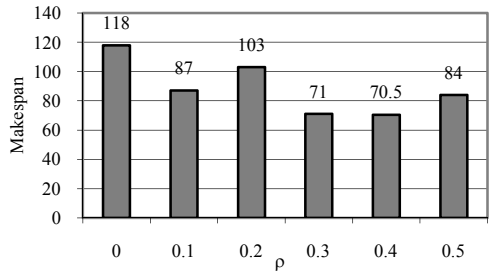
α=1, β=1, r=0.5, Q=100 and m=10.

Figure 4. ACO performance for different values of tmax.



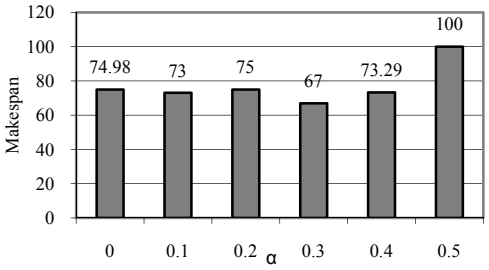
α=1, β=1, r=0.5, t_{max}=100 and m=10.

Figure 5. ACO performance for different values of Q.



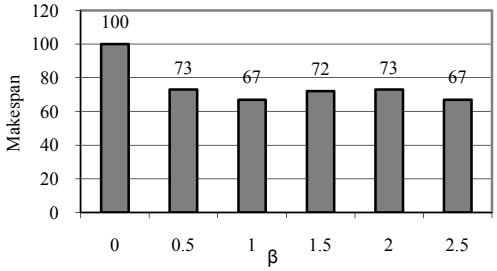
α=1, β=1, Q=100, t_{max}=100 and m=10.

Figure 6. ACO performance for different values of RHO.



β=1, ρ=0.4, Q=100, t_{max}=100 and m=10.

Figure 7. ACO performance for different values of alpha.



α=0.3, ρ=0.5, Q=100, t_{max}=100 and m=10.

Figure 8. ACO performance for different values of beta.

Table 2. Selected parameters Of ACO.

Parameter	α	β	ρ	Q	m	tmax
Value	0. 3	1	0.4	100	10	100

5.3. Implementation Results of ACO, FCFS and RR

The following experiments, we compared the average makespan with different tasks set. The average makespan of the ACO, RR and FCFS algorithms are shown in Figure 9. It can be seen that, with the increase of the quantity task, ACO takes the time less than RR and FCFS algorithms. This indicates that ACO algorithm is better than RR and FCFS algorithms.

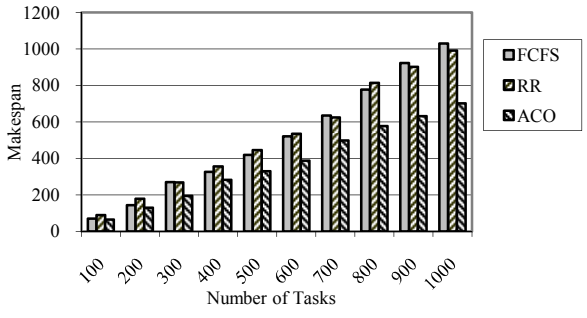


Figure 9. Average makespan of FCFS, RR and ACO.

In statistics and probability theory, standard deviation (σ) shows how much variation or dispersion exists from the average (mean), or expected value. A low standard deviation indicates that the data points tend to be very close to the mean; high standard deviation indicates that the data points are spread out over a large range of values (solving stagnation problem). Since, the standard deviation of never drops

to zero, we are assured that the algorithm actively searches solutions which differ from the best-so-far found, which gives it the possibility of finding better ones. Figure 10 shows the evolution of the standard deviation of the ACO over 10 runs.

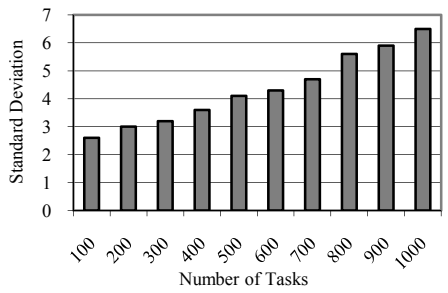


Figure 10. Standard deviation of ACO over 10 runs.

The Degree of Imbalance (DI) measures the imbalance among VMs, which is computed by Equations 8 and 9.

$$T_i = \frac{TL\_Tasks}{Pe\_num_j * Pe\_mips_j} \quad (8)$$

Where, *TL_Tasks* is the total length of tasks which are submitted to the VM_i.

$$DI = \frac{T_{\max} - T_{\min}}{T_{avg}} \quad (9)$$

Where, T_{max}, T_{min} and T_{avg} are the maximum, minimum and average T_i respectively among all VMs. The average DI of each algorithm with the number of tasks varying from 100 to 1000 is shown in Figure 11. It can be seen that the ACO can achieve better system load balance than RR and FCFS algorithms.

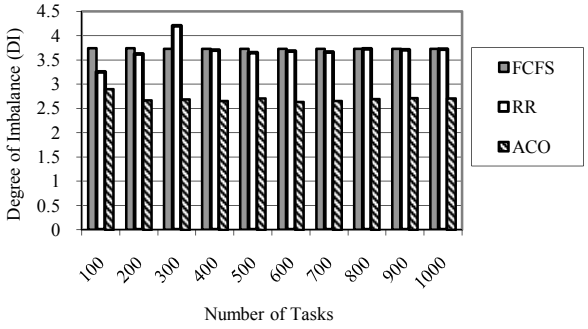


Figure 11. Average DI of FCFS, RR and ACO.

6. Conclusions and Future Work

In this paper, ACO algorithm for achieving cloud computing tasks scheduling has been presented. Firstly, the best values of parameters for ACO algorithm, experimentally determined. Then, the ACO algorithm in applications with the number of tasks varying from 100 to 1000 evaluated. Simulation results demonstrate that ACO algorithm outperforms FCFS and RR algorithms. In future work the effect of precedence between tasks and load balancing will be considered. Also, the comparison between our approach and other metaheuristics approaches will be performed.

References

[1] Boloor K., Chirkova R., Salo T., and Viniotis Y., "Heuristic-Based Request Scheduling Subject to a Percentile Response Time SLA in a Distributed Cloud," *in Proceedings of the IEEE International Conference on Global Telecommunications Conference*, Florida, USA, pp. 1-6, 2010.

[2] Bonabeau E., Dorigo M., and theraulaz G., *Swarm Intelligence: From Natural to Artificial Intelligence*, Oxford University Press, New York, USA, 1999.

[3] Buyya R., Ranjan R., and Calheiros N., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities," *in Proceedings of the 7th High Performance Computing and Simulation Conference*, Leipzig, Germany, pp. 1-11, 2009.

[4] Dorigo M. and Blum C., "Ant Colony Optimization Theory: A Survey," *in Theoretical Computer Science*, vol. 344, no. 2, pp. 243-278, 2005.

[5] Dorigo M., Birattari M., and Stutzel T., "Ant Colony Optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28-39, 2006.

[6] Fangzhe C., Ren J., and Viswanathan R., "Optimal Resource Allocation in Clouds," *in Proceedings of the 3rd International Conference on Cloud Computing*, Florida, USA, pp. 418-425, 2010.

[7] Gao K., Wang Q., and Xi L., "Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment," *the International Arab Journal of Information Technology*, vol. 11, no. 3, pp. 268-275, 2014.

[8] Gao Y., Guan H., Qi Z., Hou Y., and Liu L., "A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230-1242, 2013.

[9] Ghalem B., Tayeb F., and Zaoui W., "Approaches to Improve the Resources Management in the Simulator Cloudsim," *in Proceedings of the Conference on Interaction and Confidence Building Measures in Asia*, *Lecture Notes in Computer Science*, Istanbul, Turkey, pp. 189-196, 2010.

[10] Hsu C. and Chen T., "Adaptive Scheduling Based on Quality of Service in Heterogeneous Environments," *in Proceedings of the IEEE International Conference on Multimedia and*

Ubiquitous Engineering, California, USA, pp. 1-6, 2010.

[11] Ijaz S., Munir E., Anwar W., and Nasir W., "Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment," *the International Arab Journal of Information Technology*, vol 10, no. 5, pp. 486-492, 2013.

[12] Kessaci Y., Melab N., and Talbi E., "A Pareto-Based GA for Scheduling HPC Applications on Distributed Cloud Infrastructures," *in Proceedings of the IEEE International Conference on High Performance Computing and Simulation*, Istanbul, Turkey, pp. 456-462, 2011.

[13] Lorpunmanee S., Sap M., Abdul A., and Chompoo C., "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment," *in Proceedings of World Academy of Science, English and Technology*, 2007.

[14] Mahamud K. and Nasir H., "Ant Colony Algorithm for Job Scheduling in Grid Computing," *in Proceedings of the 4th Asia International Conference on Mathematical/ Analytical Modelling and Computer Simulation*, Kota Kinabalu, Malaysia, pp. 40-45, 2010.

[15] Nemhauser G. and Wolsey A., *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, USA, 1988.

[16] Paul M. and Sanyal G., "Survey and Analysis of Optimal Scheduling Strategies in Cloud Environment," *in Proceedings of the IEEE International Conference on Information and Communication Technologies*, Georgia, USA, pp. 789-792, 2012.

[17] Qiyi H. and Tinglei H., "An Optimistic Job Scheduling Strategy based on QoS for Cloud Computing," *in Proceedings of the IEEE International Conference on Intelligent Computing and Integrated Systems*, Guilin, China, pp. 673-675, 2010.

[18] Reeves C., *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, Oxford, England, 1993.

[19] Singh M., "GRAAA: Grid Resource Allocation Based on Ant Algorithm," *the Journal of Advances in Information Technology*, vol. 1, no. 3, pp.133-135, 2010.

[20] Weiss A., "Computing in the Clouds," *Net Worker on Cloud computing: PC functions move onto the web*, vol. 11, no. 4, pp. 16-25, 2007.

[21] Xu M., Cui L., Wang H., and Bi Y., "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing," *in Proceedings of the IEEE International Conference on Parallel and Distributed Processing with Applications*, Chendu and JiuZhai Valley, China, pp. 629-634, 2009.

[22] Zhao C., Zhang S., Liu Q., Xie J., and Hu J., "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing," *in Proceedings of the IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, China, pp. 1-4, 2009.

[23] Zhao L., Ren Y., Xiang Y., and Sakurai K., "Fault-Tolerant Scheduling with Dynamic Number of Replicas in Heterogeneous Systems," *in Proceedings of the IEEE International Conference on High Performance Computing and Communications*, Washington, USA, pp. 434-441, 2010.

[24] Zhu K., Song H., Liu L., Gao J., and Cheng G., "Hybrid Genetic Algorithm for Cloud Computing Applications," *in Proceedings of the IEEE International Conference on Asia-Pacific Services Computing Conference*, Jeju, Korea, pp. 182-187, 2011.

Medhat Tawfeek received the BSc and MSc degrees in computers and information from Menofia University, Faculty of Computers and Information in 2005 and 2010, respectively. Currently, hold PhD degree student in Faculty of Computers and information, Menofia University. His research interest includes cloud computing, smart card security, distributed system, fault tolerance.

Ashraf El-Sisi received the BSc and MSc degrees in electronic engineering and computer science engineering from Menofyia University, Faculty of Electronic in 1989 and 1995, respectively and received his PhD degree in computer engineering and control from Zagazig University, Faculty of Engineering in 2001. His research interest includes cloud computing, privacy preserving data mining, AI approaches in software testing, intelligent agent, testing biometric security algorithms and devices, and intelligent systems.

Arabi Keshk received the BSc degree in electronic engineering and MSc degree in computer science and engineering from Menofia University, Faculty of Electronic Engineering in 1987 and 1995, respectively and received his PhD degree in electronic engineering from Osaka University, Japan in 2001. His research interest includes software testing, distributed system, data mining and bioinformatics.

Fawzy Torkey received the BSc degree in industrial electronics in 1974 from the Faculty of Electronic Engineering, Menoufia University, Egypt. He received the MSc degree in electrical engineering and electronics, in 1980 from the Faculty of Engineering, Cairo University, Egypt. He received the PhD degree in computer engineering, Liverpool University, England, in 1985. He was the Faculty Dean in the period from 2001 to 2006, Faculty of Computers and Information, Menoufia University, Egypt. He also was the president of Kafrelsheikh University, Egypt, in the period from 2006 to 2011. He is presently a professor in the Department of Computer Science, Faculty of Computers and Information, Menoufia University, Egypt. His research interests include computer architecture, parallel processing, database and distributed systems.