ISSN No. 0976-5697

Volume 6, No. 6, July - August 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A New Scheduling Method for Workflows on Cloud Computing

Seyed Ebrahim Dashti
Department Of Computer, science and research Branch,
Islamic Azad University, Tehran, Iran

Amir masoud Rahmani
Department Of Computer, science and research Branch,
Islamic Azad University, Tehran, Iran

Abstract: Cloud computing has recently become a very popular topic. Nowadays, with the increasing demand for process automation in the cloud, the investigation on cloud workflow scheduling strategies is becoming a significant issue. Majority of existing workflow scheduling algorithms consider only compute resources that usually cannot be provisioned on demand size of workflows or not released to the environment until the workflow execution completes. That is why the performance of these algorithms is being decreased and time and cost of them is being increased. In this paper, we present a new workflow scheduling method based on Ant Colony Optimization algorithm in order to reduce this scheduling overhead with considering the above problems in our dynamic environment. Furthermore, we do consider these problems and various type VMs during the execution dynamically based on Amazon EC2. Also in comparison with state of the art in large-scale scheduling method, our datasets are based on real workflow applications with maximum 100 nodes. The results show that performance of our algorithm is significantly better than Greedy Randomized Adaptive Search Procedure (GRASP) and scalable for increasing nodes of workflow.

Keyword: workflow scheduling, cloud computing, VM allocation, Dynamic VM, user constraint, ACO (Ant Colony Optimization).

1. INTRODUCTION

Cloud computing is emerging as the latest distributed computing paradigm and attracts increasing interests of researchers in the area of Distributed and Parallel Computing, Service Oriented Computing, and Software Engineering. Though there is yet no consensus on what Cloud is, some of its distinctive aspects as proposed by Ian Foster in [1] can be borrowed for an insight: “Cloud computing is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.” Compared with the definitions of conventional computing paradigms such as cluster, grid, and peer-to-peer (p2p), “economies” is a noticeable keyword in cloud computing which has been neglected by others.

Utility and cloud computing have emerged as new service provisioning models and are capable of supporting diverse computing services such as servers, storage, network and applications for e-Business and e-Science over a global network. In cloud these services introduced as Infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [2]. In cloud environments, users consume the services when they need to, and pay only for what they use. With economy incentive, these technologies encourage organizations to offer their specialized applications and other computing utilities as services so that other individuals/organizations can access these resources remotely. Therefore, it facilitates individuals/ organizations to develop their own core activities without maintaining and developing fundamental infrastructure.

Nowadays, with the increasing demand for process automation, especially for large-scale collaborative, e-science applications, and distributed e-business [2, 3], the investigation on cloud workflow scheduling strategies is becoming a significant issue. In the following paragraph, we define the workflow and the importance of workflow in cloud environment.

Workflows can be modeled as Directed Acyclic Graphs (DAGs). In this model, each node in the DAG represents an executable task. Each directed edge represents a precedence constraint between two tasks (data or control dependence). A DAG represents a model that help build a schedule of the tasks onto resources in a way that precedence constraints are respected and the schedule is optimized. The majority of proposed workflows [4, 5] aim at minimizing execution time of the schedule; However, other important parameters such as budget and money should be taken into consideration in cloud. In this paper, we do solve the workflow scheduling problems in cloud.

There are several reasons that we encourage to investigate the workflow in cloud as follow a: Initially, workflows were being implemented in grids. Due to the above mentioned causes and reduced performance faced in grids, now there is a need to implement workflows in cloud. The primary benefit of moving to clouds is application scalability. Unlike Grids, scalability of cloud resources allows real-time provisioning of resources to meet application requirements. This enables workflow management systems to readily meet Quality of Service (QoS) requirements of applications, as opposed to the traditional approach that required advance reservation of resources in global multi-user Grid environments. Cloud services like compute, storage and bandwidth resources are available at substantially lower costs. Workflow applications often require very complex execution environments. These environments are difficult to create on grid resources. In addition, each grid site has a different configuration, which results in extra effort each time an application needs to be ported to a new site. Virtual machines allow the application developer to create a fully customized, portable execution environment configured specifically for their application [6].

Main contributions of this paper are as follows.

1. Present architecture for workflow scheduling.
2. Present a new workflow scheduler as a subdomain of workflow manager which would be defined as a new service in PaaS level.
3. Define a new problem in cloud workflow by scheduling it in various type of VM.
4. Define a new aspect in scheduling cloud workflow problem by assign task to various type of VM.
5. Adapt ant colony optimization method to solve workflow scheduling in cloud computing in order to improve makespan in budget constraint from user perspective.
6. New formulate the ACO pheromone and heuristic by embedding cost, time and budget constraint to solve our problem.
7. Design refined algorithm to reduce bought VM cost in full hour billing model.
8. An extensive simulation-based evaluation and performance analysis of the proposed algorithm.

The rest of the paper is organized as follow. In Section 2 we discuss the related work. We present our proposed architecture and Framework in Sections 3, at last of this section, we define some of cloud characteristics assumed in our environment. In section 4, description of our problem is continued with design heuristic by formulate pheromone in ACO to solve workflow scheduling. In section 5 we design a refined sub algorithm to improve ACO results. In section 6, we evaluate our approach and analysis of experimental result. We discuss conclude the paper in secion7.

2. LITERATURE REVIEW

Many heuristics have been investigated by several projects for scheduling workflows on Grids. The heuristics can be classified as either task level or workflow level. Task level heuristics make scheduling decisions based only on the information about a task or a set of independent tasks, while workflow level heuristics take into account the information of the entire workflow. Min-Min, Max-Min and Sufferage [7] are three major task level heuristics employed for scheduling workflows on Grids which are being used by Mandal et al [8] to schedule EMAN bio-imaging applications. Blythe et al. [9] proposes a workflow level scheduling algorithm based on Greedy Randomized Adaptive Search Procedure (GRASP) [10] which is also compared with Min-Min in compute-intensive and data-intensive scenarios. Another two workflow level heuristics have been employed by the ASKALON project [11, 12]. One is based on Genetic Algorithms and the other is a Heterogeneous-Earliest-Finish-Time (HEFT) algorithm. Sakellariou and Zhao [13] developed a low-cost rescheduling policy in order to reduce the overhead produced by rescheduling by conducting rescheduling only when the delay of a task execution impacts on the entire workflow execution.

However, these works only attempt to minimize workflow execution time and do not consider users' budget constraints in Grid. Several works have been proposed to address scheduling problems based on users' budget constraints. Nimrod-G [14] schedules independent tasks for parameter-sweep applications to meet users' budget. A market-based workflow management system [15] locates an optimal bid based on the budget of the current task in the workflow. Tsiakkouri et al [16] developed scheduling approaches, LOSS and GAIN, to adjust a schedule which is generated by a time optimized heuristic and a cost optimized heuristic to meet users' budget constraints respectively. A time optimized heuristic attempts to minimize execution time while a cost optimization attempts to minimize execution cost. More recently, [17] schedule bag of tasks to VMs in cloud using integer programing method (without any order). Some researchers assign workflow tasks to services in user's QoS(Quality of Service), it means they investigate in higher layers of cloud (SaaS and PaaS)[18,19]. In contrast of [17], we have been ordering of tasks in workflow and in comparison with [18, 19] we consider VMs and type of them in cloud and discuss it in lower layers (PaaS and IaaS).

A number of Grid workflow management systems [20, 21, 22] with scheduling algorithms have been developed. They facilitate the execution of workflow applications and minimize their execution time on Grids. However, to impose a workflow paradigm on cloud, execution cost must also be considered when scheduling tasks on resources. The price of VMs is mainly determined by its QoS level such as the processing speed.

Users may not always need to complete workflows earlier than they require. They sometimes may prefer to use cheaper VM that is sufficient to meet their requirements. [23]. There are few work examining issues related to budget constraints in a Grid context. The most relevant work is available in [23, 24], where it is demonstrated, through simulation. Although, the constraint budget and deadline parameters were considered in the grid environment [25] but not in cloud, which minimizes workflow execution cost within a certain deadline and other QoS. In [23] a genetic algorithm based scheduling heuristic is developed to minimize execution time while meeting user's budget constraint,[25] develop an ACO algorithm in Grid for different QoS. Several researchers use ACO algorithm in cloud to minimize the makespan by balancing the entire system load. Two different load balancing scheduling algorithms based on ant colony are proposed in [26] and [27]. Another ant colony based algorithm aims to minimize job completion time based on pheromone is proposed in [28]. Some researchers consider the cost parameter in grid/cloud for service or VM. To the best of our knowledge, all of those methods get and free VM statically that once VM provision, power on, until end of scheduling. But in cloud computing, dynamic scalability becomes more attractive and practical because of the unlimited resource pool. Some articles are based on the best effort method and optimize workflow execution time and scalability in multiple and single workflow [29, 30] without considering cost parameter that plays important role in cloud. Some researchers consider specific workflow in cloud such as transaction-intensive. In [31], the objective of optimization is to maximize the throughput. [32] Solve the workflow optimization in cost constraint parameter and [33] improves it using comprise cost and time constraint. Large-scale distributed systems are considered by Look-Ahead Genetic Algorithm (LAGA) in [34]which optimize both makespan and reliability of a workflow application, some other improve performance and reliably of their workflow middleware in distributed environments, such as grids and clouds [35].

In [36], authors discuss load-balancing technologies for SaaS infrastructure to execute workflow processes, they propose a proactive load balancing algorithm, by which requests from different tenants are scheduled concurrently by a single service instance over shared hosting resources and the predicted cost of executing the process instances. Some other solve the problem by using virtual clusters as cloud in IaaS level, they propose the SHEFT algorithm to schedule a workflow elastically on a Cloud computing environment [29]. In another paper researchers present a new workflow scheduling method based on iterative ordinal optimization (IOO) and they try to reduce makespan and improve throughput [37]. In [30], researchers also propose virtual cluster and try to find a solution that meets all users preferred QoS constraints while improve CPU utilization. Schedule workflow in two levels of service-level and task-level from SaaS to IaaS is represented in[38], in [39,40] researchers presented scheduling heuristic methods based on Particle Swarm Optimization (PSO) to minimize data transmission cost of data intensive workflows on Cloud.

In this paper, we consider workflow applications that are modeled as DAGs. Moreover, the above mentioned articles focusing on either service flow scheduling with various QoS requirements [18], or makespan optimization or throughput. We also consider that a budget constraint and scalability of VMs need to be satisfied. Each job, when running on a machine, consumes some money. Thus, the overall aim is to find the schedule that gives the shortest makespan for a given DAG and a given set of resources without exceeding the budget available. To achieve these aims, we develop an Ant colony Optimization algorithm for scheduling workflow.

3. THE PROPOSED ARCHITECTURE

Many computation-intensive applications in science and business can be described as workflows. According to [38], we can imagine the general cloud architecture as four basic layers from top to bottom: application layer (user applications), platform layer (middleware cloud services to facilitate the development/deployment of user applications),unified resource layer(abstracted/encapsulated resources by virtualisation), and fabric layer (physical hardware resources).Here, we present the lifecycle of a workflow application to illustrate the system architecture in Figure 1. Note that here we focus on the system architecture.

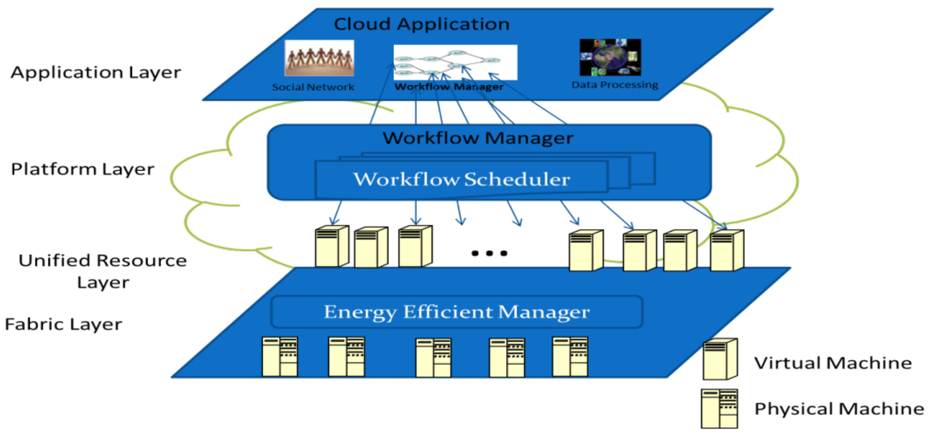


Figure 1: Cloud workflow system architecture

We design a framework for workflow management scheduling as it's components can be seen in Figure 2. We define each component briefly in the following:

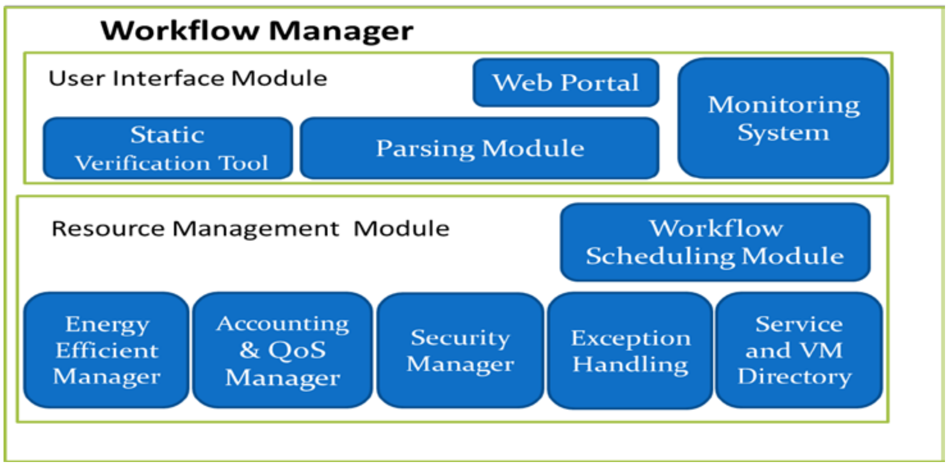


Figure 2: Framework of workflow management in our cloud

User Interface Module contains four components as follows:

Web portal: this module defines a user friendly interface to communicate with customer and other users. The user can define a new workflow with her parameters and QoS or use original template workflow and manipulate them for her specific usage or parameters.

Parsing Module: this module transforms graphical workflows that enter by the user to XML standard form that understandable for workflow manager; Moreover, it adds metadata like size of data transfer between nodes. Clustering can add to this module to merge small tasks to enhance performance.

Static Verification Tool: this module tests connectivity between nodes in workflow and verify data user in enter parameters in order to be in a valid range. Also this module tests XML format of workflow that the user enter.

Monitoring System: this module communicates with all other components and collects information. Customer and designer are able to monitor related module in an online fashion.

Resource Management module contains six components as follows:

Service and VM Directory: this module index all existing services and VMs type and price. It also index services and VM images.

Exception Handling: this component manages fault tolerance, for example if fault occurs in VM during run time execution; this component reschedules the task in order to prevent any delay for other nodes of the workflow.

Security Manager: this component maintains privacy of other components and VMs and prevents malicious access.

Accounting and QoS Manager: this component monitors task execution QoS and amount of credit that spend. If any inconvenience occurs in QoS, it reports to Exception Handling to manage them. This module also calculate amount of credit in user‘s account according to QoS.

Workflow Scheduling Module: this component schedules a task to a suitable VM that satisfy the user QoS and other constraints like order in workflow.

Energy Efficient Manager: this component manages energy efficiency of datacenter and manages load of host and merge under load host and switch off the free host.

A user may first submit a workflow application that specifies task definitions, the execution order of tasks (process structures), and QoS constraints through a web portal [19]. Once workflow specifications are created, workflow manager can verify structure errors and QoS constraints statically. Our workflow scheduler specifies the number and type of VM in user constraint that schedule tasks in order to be mapped to VM in unify Resource Layer so that a concrete workflow is generated.

3.1. CLOUD ARCHITECTURE CHARACTERISTICS

As a computing platform, clouds own distinct characteristics compared to utility computing and grid computing. We have identified the following characteristics which get from popular cloud computing like amazon EC2 that can largely affect the way people use cloud platforms, especially in cloud scaling activities.

Unlimited resources limited budget: Clouds offer users unlimited computing power and storage capacity. Unlimited resource enables applications to scale to extremely large size. On the other hand, these unlimited resources are not free. Every cycle used and byte transferred are going to appear on the bill. Budget cap is a necessary constraint for users to consider where they deploy applications in clouds. Therefore, a cloud auto-scaling mechanism should explicitly consider user budget constraints when acquiring resources.

Full hour billing model: The pay-as-you-go billing model is attractive, because it saves money when users shut down machines. However, VM instances are always billed by hours. Fraction consumption of an instance-hour is counted as a full hour. In other words, 10 minutes and 60 minutes usage are both billed as 1 hour usage and if an instance is started and shut down twice in an hour, users will be charged for two instance hours. The shutting down time can greatly affect cloud cost. Therefore, a reasonable policy is that whenever an instance is started, it is better to be shut down when approaching full hour operation.

4. PROBLEM DESCRIPTION

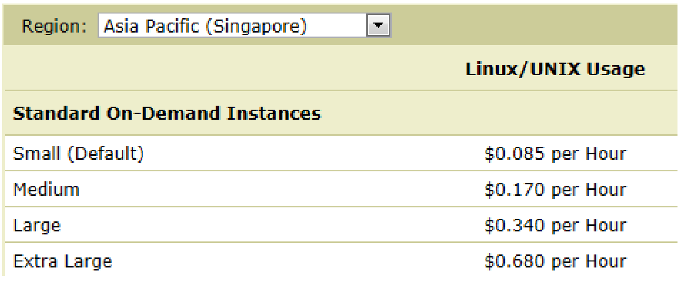
We model a workflow application as a Directed Acyclic Graph (DAG). Let Γ be the finite set of tasks Ti $1\le i\le n$. Let Λ be the set of directed arcs of the form (Ti,Tj) where Ti is called a parent task of Tj , on the other hand Tj is the child task of Ti. We assume that a child task cannot be executed until all of its parent tasks are completed. Let B be the cost constraint (budget) specified by the users for workflow execution. Then, the workflow application can be described as a Directed Acyclic Graph (DAG) G=(Γ,Λ, B).

Suppose that we have unlimited Virtual Machines (VMs) with four variant types like speed and RAM. Virtual machines have varied processing capability delivered at different prices. We estimate the max number of type groups that ants can be selected by Equation 1. In this formula one VM add to number of groups in order to execute a critical path. Total number of VM instances that is able to select can be calculated according to the Equation 2.

Equation 1: Number of group= {Number_of_total_task_in workflow – length of longest critical path +1}

Equation 2: Total VM Instance=Number of group* Number of VMs Type

Suppose only one VM can be assigned for the execution of a task; each task must be allocated to one available time. The cost of virtual machine type is shown in Figure 3 that each type has specific speed and RAM (Random Access Memory). We got costs and types from Amazon VM in Asia, where the VM is charged at least for one hour. The scheduling problem is to map every T_i into a suitable virtual machine to minimize the execution time of the workflow and complete it within the budget B.


Region: Asia Pacific (Singapore)

	Linux/UNIX Usage
Standard On-Demand Instances	
Small (Default)	$0.085 per Hour
Medium	$0.170 per Hour
Large	$0.340 per Hour
Extra Large	$0.680 per Hour

Figure 3: cost of amazon VM [41]

4.2. ACO METHOD FOR SCHEDULING ALGORITHM

The general idea of Ant Colony Optimization (ACO) is to simulate the foraging behavior of ant colonies. When a group of ants set out from their nest to search for food source, they use a special kind of chemical to communicate with each other. The chemical is referred to as the pheromone. Once ants discover a path to food source, they deposit pheromone on the path. By sensing pheromone on the ground, ants can follow the path to food source discovered by other ants. As this process continues, most of the ants tend to choose the shortest path to food as there have been a huge amount of pheromones accumulated on this path. This collective pheromone depositing and pheromone following behavior of ants becomes the inspiring source of ACO. The flowchart of the high level algorithm is given by Figure 4. These procedures are described in detail below.

The ant colony optimization has been implemented on several engineering domain [42]. Our problem is similar to travelling salesman problem, ACO usually is use to produce near-optimal solutions for it [42]. ACO also, has an advantage over simulated annealing and genetic algorithm approaches of similar problems when the graph may change dynamically. In this paper, we apply ACO algorithms to tackle the workflow scheduling problem in cloud computing. ACO has several advantages; one of them is avoiding the convergence to a locally optimal solution (because of Pheromone evaporation). Despite the distributed computation avoids premature convergence, positive feedback leads to rapid discovery of good solutions. It is possible to prove that it is convergent i.e., it is able to find the global optimum in finite time [43]. Also, the ACO algorithm is scalable for number of nodes and its performance and results are acceptable. In other words, by increasing the workflow nodes, overall performance remains in an acceptable level. Informally, the algorithm can be viewed as the interplay of the following procedures:

1) Initialization of the algorithm: All pheromone values and parameters are initialized.
2) Initialization of ants: Assume that a group of M ants are used in the algorithm. At the beginning of the iteration, all ants are set to the initial state.

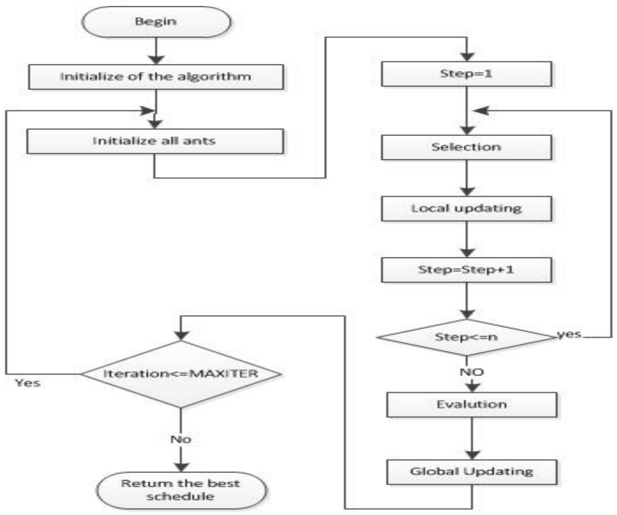


Figure 4: Flowchart of the ACO algorithm

3) Solution construction: M ants set out to build M solutions to the problem. The construction procedure includes n steps where n is the number of Tasks in the workflow. In each step, each ant picks up the next tasks using pheromone and heuristic information and maps it to the most suitable Virtual Machine. The algorithm also estimates the earliest start time and the earliest end time of tasks and the free duration of last one hour of the VM in terms of the information of partial solution built by each ant. This information is helpful to guide the search behavior of ants. Each ant chooses duration via greedy method with regard the constraint Budget heuristic. Based on the heuristic, each ant builds its tackling sequence of tasks
4) Local updating: Soon after an ant selects the possible T_i and maps it to VM, the corresponding pheromone value is updated by a local pheromone updating rule.
5) Global updating: After all ants have completed their constructions, global pheromone updating is applied to the best-so-far solution. The cost and makespan of all solutions are evaluated. The pheromone values related to the best-so-far solution is significantly increased. Moreover, some parameters of the algorithm are adaptively adjusted in this procedure.
6) Terminal test: If the test is passed, the algorithm is end. Otherwise, go to step (2) to begin a new iteration.

4.3. CONVERT WORKFLOW SCHEDULING TO ACO PROBLEM

The workflow should be converted to the ACO problem. The problem which is modelled with the ACO must be a graph. The workflow as defined at the beginning of this section is essentially an acyclic graph, the tasks represent the nodes of graph, and the edges between them represent the dependencies between workflow nodes. Ants are only allowed to flow this directions, in other words cost of other direction is infinity, in term of ACO, we define wall in forbidden directions. Ants travel workflow graph to optimize order of traversal for assign optimal nodes to the VMs, the traverse stops when the ants are a stopping criterion specified or the number of repetitions completed. Figure 5 illustrates selection and assignment of nodes (tasks) to the VM by an ant.

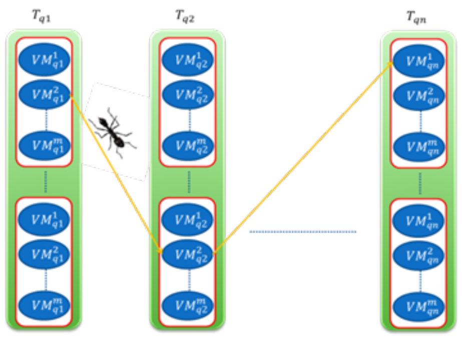


Figure 5: A sample Ant build process scheduling

Ant select the next node and VM by quantities obtained from the probability by Roulette wheel function that is calculated from the consistency of nodes and the costs ,i.e. execution and communication costs, and the running cost of VM purchased in the previous period in order to utilize the unused period of it. Also the historical of ant pheromone affects the quantity of probability that illustrated in the next section.

4.4. DEFINITION OF PHEROMONE AND HEURISTIC

Pheromone and heuristic information are the most important factors of an ACO algorithm. In general, pheromone is used to record the historical searching experiences and bias the ants' searching behaviour in future. On the other hand, heuristic information is some problem based values to guide the search direction of ants. As the scheduling problem is mainly to map all tasks to VMs, we denote the pheromone value of mapping VM_j to task T_i as fitness(i,j) and then calculate the cost of them.

In Equation 3, we estimate distance of current slot selection in VM_j to suggested budget for T_i . We use the first term of this equation in numerator and denominator of fraction to normalize the fitness. We calculate the cost of T_i in Equation 4,free time in this equation encourage ant to select VM_j that product minimum slot time that is not used for T_i ,or T_i can fill previous free slot time in VM_j.

Equation 3:

$$fitness(i,j) = \frac{Max\{|MaxVMcost_i - SB_i|, |SB_i - minVMcost_i|\} - |new\ allocated\ slot_{Cost} - SB_i|}{max\{|maxVMcost_i - SB_i|,\ |SB_i - minVMcost_i|\}}$$

Equation 4: $Cost(i) = \frac{1}{fitness(i)} + freetime\ of\ vmj$

Suggest Budget (SB): the SB heuristic biases the artificial ants to select the VMs within the budget cost. To achieve this objective, for each task T_i, we assign a suggested budget SB_i based on the user-defined budget of the application [25].

By mapping all tasks to the VMs with the lowest cost, we can obtain the minimum cost of the whole workflow application (denoted as min_Cost). That is,

Equation 5: $\mathbf{minVMCost}_i = \sum_{j=1}^{n} \mathbf{min\ cost\ j}$

The suggested budget SB_i for T_i is evaluated by enlarging the value of min $cost_i$ on a scale of Budget:

Equation 6: $\mathbf{SB}_i = \mathbf{min\ VMcost}_i * \frac{Budget}{mincost}$

Suppose an ant's heuristic type is the SB heuristic, and then the heuristic value of mapping VM_j to T_i is set to Equation 3. According to Equation 3, a VM instance, the cost of which is closer to SB_i, will be associated with a higher heuristic value and fitness(i,j) ∈ (0, 1].

Global Pheromone Updating: Global updating takes place after all ants have built their solutions. The algorithm first compares all solutions in that iteration. The quality of a solution schedule I can be evaluated by the following equations Equation 7. In the case of makespan optimization in constraint budget, the score of schedule I is given by:

Equation 7:

$$length(I) = \begin{cases} \frac{cost(I)}{Budget} + \frac{Max\_Makespan}{min\_Makespan} & if\ cost(I) > Budget \\ \frac{makespan(I)}{min\_Makespan} & if\ cost(I) \leq Budget \end{cases}$$

The length of the schedule I in terms of (5) is composed of two parts: penalties of QoS constraints and quality of the user-preferred QoS parameter. If I satisfies all QoS constraints, its length for QoS constraints will be set to zero, and the length for the user-preferred QoS parameter will be set according to the makespan of I. Better makespan will contribute to a lower length of tour. On the other hand, if I fails to satisfy all QoS constraints, its score for QoS constraints will be set according to the degree of satisfaction, and the length for the user-preferred QoS parameter will be set to a maximum value. The smaller the length is, the better the schedule will be.

5. REFINE SCHEDULING BY MERGE HOMOGENISE VM

Scheduling multitask workflows is NP-hard problem, therefor one approach to solve them in an acceptable time is meta-heuristic methods such as ACO algorithm. Because of exploring more search space, Equation 1 (number of VMs selected by ACO=Number of group*4(Figure 3)) is designed very optimistically. But in practice, we experimentally understood in scheduling result some homogenise VMs could merge due to reduced cost in pre-specified makespan. Figure 6 and Figure 7 illustrate the one instance scheduling that our algorithm generated for e-protein workflow in budget 5$ before and after refine the scheduling. We illustrate the pseudo code of the refined algorithm in Algorithm 1.

Algorithm 1:Refine algorithm

```
For i=1 to Total VM Instance step Number of VMs Type
For j=i to i-1+ Number of VMs Type
If VM is not empty
If Tasks duration in homogenous VM not conflict
            Merge VMs
```

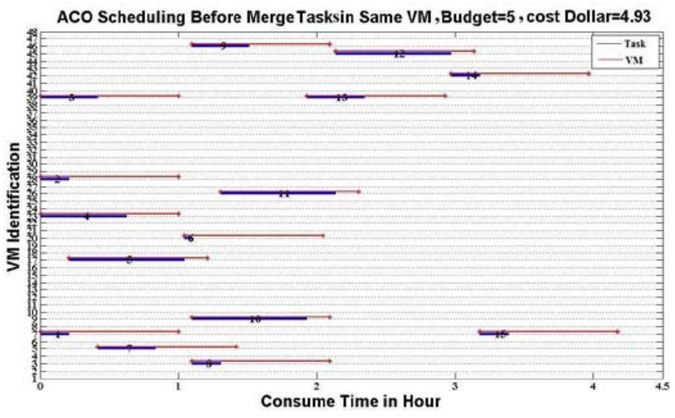


Figure 6: ACO scheduling before merge tasks

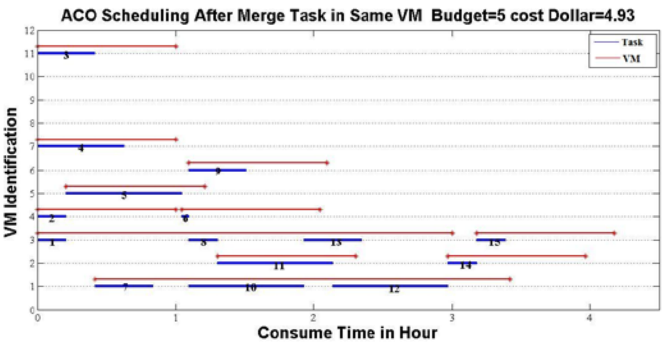


Figure 7: ACO scheduling after merge tasks

6. EXPERIMENTAL RESULT

We simulate our experiments at high level in Matlab and at low level in cloudsim environment; there are mainly five parameters in the algorithm: ant number, α, β, ρ, and maximum iteration. α and β consequently determine the impact of before and current best value in ant new selection path. ρ is the global refresh effect in founded tour. The maximum iteration is a variable that is specified based on our experience which is responsible to prevent the algorithm not to be in an infinity loop because of unexpected user's entry. If the ant fails to satisfy the user's constraints in the maximum iteration, the algorithm exit and print the best result that found in the maximum iteration. The best empirical values of those parameters for our problem consequently are 50, 7, 1, 0.65, and 50.

We test the ACS algorithm in four workflow applications. The basic information of these workflow applications is shown in Table 1. The first two workflows, including the neuroscience application [functional MRI (fMRI)] with 15 tasks, and the e-protein workflow with 15 tasks [23], are derived from real life applications.

Table 1: Test Instances

Instance Name	**Number of Tasks**	**Network**
fMRI	15	Figure 8
e-Protein	15	Figure 9
J50_0	50	J50_0 (PSPLIB)
J100_0	100	J100_0 (PSPLIB)

The DAGs of these applications are shown in Figure 8 and Figure 9. The other two workflows are generated based on the networks in the project scheduling problem library (PSPLIB) [24], which is a library for project scheduling problems. These networks include j50_0 with 50 tasks, and j100_0 with 100. We estimate maximum number of type groups (VM instances) that ants can be select according to Equation 1.Total number of VM instances that ant can be selected calculate from Equation 2. The QoS parameters (execution time and cost) of all VMs instances are given from amazon, but they follow the rule that for the same task, a VM instance with shorter execution time may cost more money and vice versa. The range of task's length in workflow is between one million to ten million instructions.

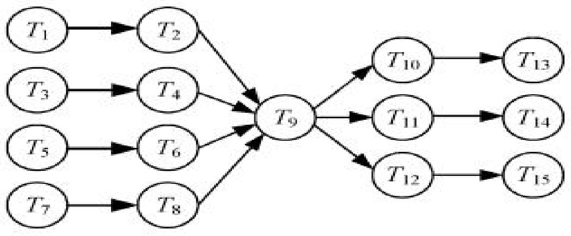


Figure 8: a neuro-science workflow: fMRI (parallel app)

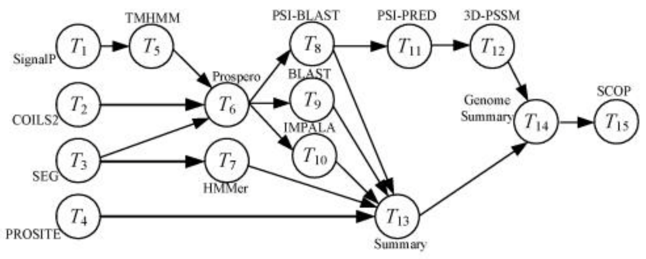


Figure 9: e-protein workflow (Hybrid structure)

As has been mentioned before, few methods solve workflow scheduling problem with different user-defined QoS constraints in grid, but we consider this problem in cloud by dynamic get and free of a VM and try to reduce makespan in user's budget.

We compare our ACO approach with Greedy Randomized Adaptive Search Procedure (GRASP) that is one kind of greedy algorithm [27] and other researchers usually compare their proposed evolutionary algorithms with this method. Algorithm 2 shows pseudo code of this algorithm. The greedy approach assigns a VM to each task in the workflow based on the free slot time of VM randomly

Algorithm2:GRASP (Greedy Randomized Adaptive Search Procedure)algorithm

```
1:While stopping criterion not satisfied do
2:      schedule ← createSchedule(work flow)
3:      if   schedule is better than bestSchedule   then
4:             bestSchedule ← schedule
5:      end if
6: end while
7: PROCEDURE:createSchedule(workflow)
8: solution ← constructSolution(workflow)
9:nSolution ← localSearch(solution)
10:if nSolution is better than solution then
11: return nSolution
12:end if
13: return solution
14: END createSchedue
15:PROCEDURE: constructSolution(workflow)
16:while schedule is not completed do
17:      T ← get all unmapped ready tasks
18:        make a RCL for each tεT
19:      subSolution ← select a resource randomly for each t   εT
from its RCL
20:      solution ← solution     ∪  subSolution
21:      update information for further RCL making
22: end while
23:return solution
24:END constructSolution
25:PROCEDURE:localSearch(solution)
26:nSolution ←find an optimal local solution
27:return nSolution
28: END localSearch
```

We simulate two common workflow structures in scientific workflow applications and two other workflows from PSPLIB data set for our experiments: parallel and hybrid and random (many nodes). A parallel application (see Figure 8) requires multiple pipelines to be executed in parallel. A pipeline executes a number of tasks in a single sequential order. For example, in Figure 8, there are 4 pipelines (1-2, 3-4, 5-6 and 7-8) before task 9. A hybrid structure application (see Figure 9) is a complex combination of parallel and sequential execution. In our experiments, we used a neuro-science workflow for our parallel application and a protein annotation workflow developed by London e-Science Centre for our hybrid workflow structure application [23]. Moreover, in order to check the scalability of our approach, our simulation is applied for two workflow instances with 50 and 100 nodes from PSPLIB dataset.

Figure 10 to Figure 17 compare the execution costs and time (makespan) of using the ACO and GRASP for scheduling parallel, hybrid structure applications, 50 and 100 tasks workflow from PSPLIB dataset with suitable budget. It can be seen that the GRASP takes much higher execution cost. That is because the decision making of the GRASP is based only on the information of the current task. It may produce the best schedule for the current task but it could consequently reduce the entire workflow performance. However, as the user's budget increases, the results of the two approaches are closer. Moreover as shown in the result of 50 and 100 nodes of the workflow, we understand when the number of task increase performance of GRASP decreases but performance of ACO does not changes significantly.

In our optimization problem time and cost are opposite and when cost is increased the time decreases, as shown in Figure 10 to Figure 17 the ACO result is near to budget constraint in compare of GRASP, due to remaining the best path of the ant in each iteration and use it in other iteration, while in GRASP each iteration is searched randomly without considering the best path obtained in pervious iterations, in other word, its search is limited to compare with total result. For example, Figure 12 and Figure 13 show our algorithm for fMRI (parallel workflow) in execution cost and time is significantly better than GRASP. Our method in all sample constraints produce better results except in budget 7 which our algorithm and GRASP has similar cost. However as shown in Figure 16 our method in budget 7 is outperformed in term of time.

As the results show; in some budget the exact budget constraint may not be satisfied and it may be little lower or higher than budget, these variances is because of VM hourly cost and the length of tasks do not fitted exactly in duration hourly although our algorithm produce nearest cost to budget . On average, the ACO algorithm is better in cost or both in makespan and cost, while there are differences in various benchmark workflow. For example, the greatest performance in compare of GRASP is in parallel and large scale workflow. Thus, we can see there are comparisons to be made in cost and makespan in results of algorithms by changes in number of nods and structure of workflow.

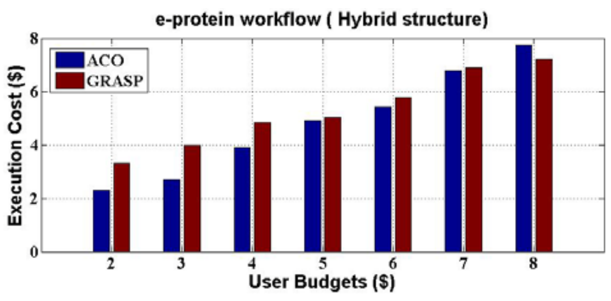


Figure 10: Comparison of execution cost in various budget in two approaches for e-protein workflow

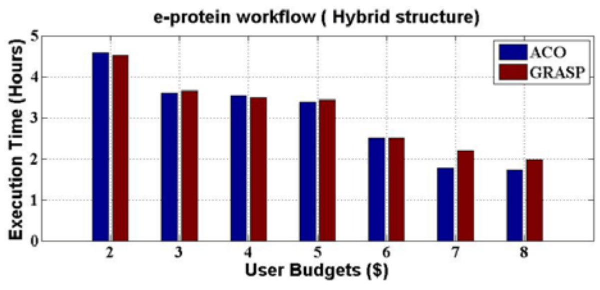


Figure 11: Comparison of makespan in different budget in two algorithms for e-protein workflow

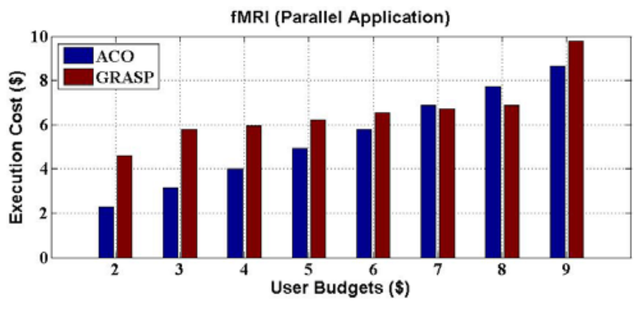


Figure 12: Comparison of execution cost in various budget in two approaches for fMRI workflow

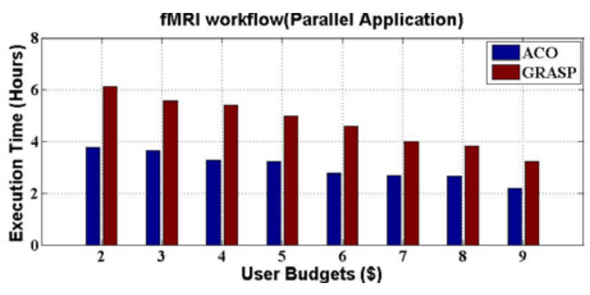


Figure 13: Comparison of makespan in different budget in two algorithms for fMRI workflow

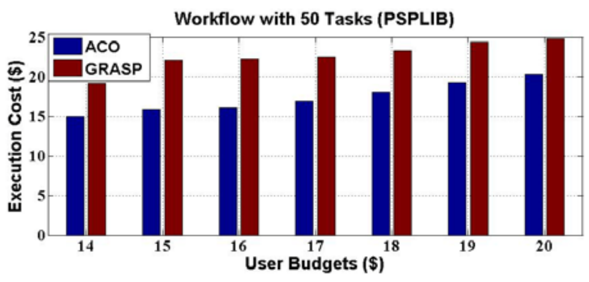


Figure 14: Comparison of execution cost in various budget in two approaches for 50 tasks workflow

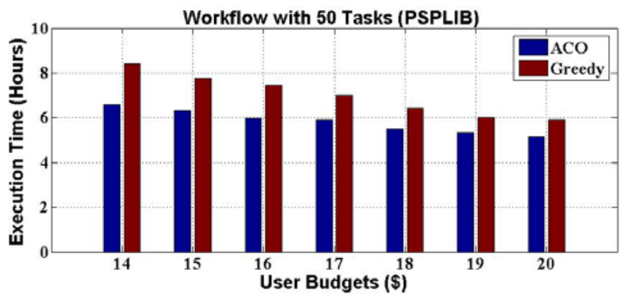


Figure 15: Comparison of makespan in different budget in two algorithms for 50 tasks workflow

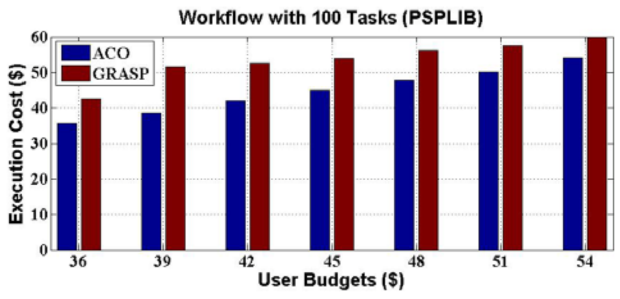


Figure 16: Comparison of execution cost in various budget in two approaches for 100 tasks workflow

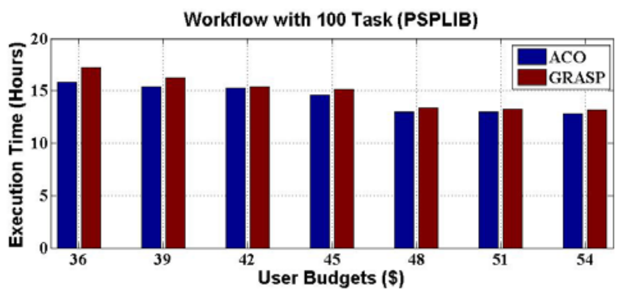


Figure 17: Comparison of makespan in different budget in two algorithms for 100 tasks workflow

7. CONCLUSION

This paper investigates about workflow scheduling in cloud in the levels of PaaS and IaaS based on the ACO optimization algorithm. Money is an important parameter in cloud that is ignored in many researches because of workflow scheduling methods, often centralize in grid. In this study, we propose the ACO approach to acquire the best scheduling in user's constraint budget. Also, instead of provisioning all VMs in the beginning of the scheduling like previous researches, we consider VM dynamically and according to the Amazon cloud the least get duration is on hour. We evaluate our method in real and PSPLIB workflow benchmark, the result shows our approach has significantly better performance in comparison to similar works.

REFERENCES

[1]Foster I, Yong Z, Raicu I, Lu S (2008) "Cloud computing and grid computing 360-degree compared" .In: Proc 2008 grid computing environments workshop, pp 1–10

[2]Verma,A. ,and Kaushal,S. (2011) "Cloud Computing Security Issues and Challenges: A Survey". In proceeding of springer International Conference on Advances in Computing and Communication, Kochi,India, 445-454.

[3]Deelman E, Gannon D, Shields M, Taylor I (2008) "Workflows and e-science: an overview of workflow system features and capabilities". Future Gener Comput Syst Volume 25, Issue 6,528–540

[4]Xiaofeng Wang , Chee Shin Yeo, Rajkumar Buyya, Jinshu Su ,(2011)"Optimizing Makespan and Reliability for Workflow Applications with Reputation and Look-ahead Genetic Algorithm". International Journal of Future Generation Computer Systems Volume 27, Issue 8, Pages 1124–1134.

[5]Cui Lin, Shiyong Lu,(2011)," Scheduling Scientific Workflows Elastically for Cloud Computing" in IEEE 4th International Conference on Cloud Computing.

[6]Anju Bala, Inderveer Chana,(2011)," A Survey of Various Workflow Scheduling Algorithms in Cloud Environment ", 2nd National Conference on Information and Communication Technology (NCICT) , International Journal of Computer Applications (IJCA).

[7]Yu J, Buyya R, (2008) "Workflow scheduling algorithms for grid computing". In: Xhafa F,Abraham A (eds) Metaheuristics for scheduling in distributed computing environments. ISBN:978-3-540-69260-7. Springer, Berlin.

[8]A. Mandal et al.,(2005) "Scheduling Strategies for Mapping Application Workflows onto the Grid",IEEE International Symposium on High Performance Distributed Computing (HPDC 2005).

[9]J. Blythe et al.,(2005) "Task Scheduling Strategies for Workflow-based Applications in Grids", In IEEE International Symposium on Cluster Computing and Grid (CCGrid).

[10]Resende, M. and Ribeiro, C.,(2002) "Greedy Randomized Adaptive Search Procedures, State-of-the-art Handbook in Metaheuristics",Glover and Kochenberger, eds., Kluwer Academic Publishers.

[11]R. Prodan and T. Fahringer,(2005) "Dynamic Scheduling of Scientific Workflow Applications on the Grid using a Modular Optimisation Tool: A Case Study", In 20 th Symposium of Applied Computing (SAC 2005), Santa Fe, New Mexico, USA, ACM Press.

[12]M. Wieczorek, R. Prodan and T. Fahringer,(2005),"Scheduling of Scientific Workflows in the ASKALON Grid Environment", Special Issues on scientific workflows, ACM SIDMOD Record, 34(3):56-62, ACM Press.

[13]R. Sakellariou and H. Zhao.(2004), "A Low-Cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems". Scientific Programming, Volume 12, Issue 4, pages 253-262.

[14]R. Buyya, J. Giddy, and D. Abramson,(2000) " An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications", In 2nd Workshop on Active Middleware Services (AMS 2000), Kluwer Academic Press, Pittsburgh, USA.

[15]A.Geppert, M.Kradolfer, and D.Tombros.(1998) "Market-based Workflow Management", International Journal of Cooperative Information Systems, World Scientific Publishing Co., NJ, USA.

[16]E.Tsiakkouri et al.,(2005) "Scheduling Workflows with Budget Constraints", In the CoreGRID Workshop on Integrated research in Grid Computing,S. Gorlatch and M.Danelutto (Eds.), Technical Report TR-05-22, University of Pisa, Dipartimento Di Informatica, Pisa, Italy, pages 347-357 .

[17]Ming Mao,Jie Li,Marty Humphrey ,(2010) "Cloud Auto-scaling with Deadline and Budget" , 11th IEEE/ACM International Conference on Grid Computing (GRID), Pages: 41 - 48.

[18]H. liu, D. Xu, H. Miao,(2011) "Ant Colony Optimization Based Service flow Scheduling with Various QoS Requirements in Cloud Computing", First ACIS International Symposium on Software and Network Engineering, Pages: 53-58.

[19]A. Verma, S. Kaushal,(2012) "Deadline and Budget Distribution based Cost- Time Optimization Workflow Scheduling Algorithm for Cloud". International Journal of Computer Applications (IJCA).

[20]T. Fahringer et al,(2005), "ASKALON: a tool set for cluster and Grid computing", Concurrency and Computation: Practice and Experience, Volume 17,Pages:143-169, Wiley InterScience.

[21]B.Ludäscher et al.,(2005) "Scientific Workflow Management and the KEPLER System", Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows.

[22]J. Yu and R. Buyya,(2005) “A Taxonomy of Workflow Management Systems for Grid Computing”, Journal of Grid Computing, Springer, Volume 3, Issue 3,4, Pages: 171-200, Spring Science+Business Media B.V., New York, USA.

[23]Jia Yu and Rajkumar Buyya ,(2006) “A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms”, Journal of Scientific Programming - Scientific Workflows archive,Volume 14 Issue 3,4, Pages: 217-230.

[24]R. Kolisch and A. Sprecher,(1997) “PSPLIB—A project scheduling problem library: OR Software—ORSEP operations research software exchange program,” Eur. J. Oper. Res., Volume 96, Issue 1, Pages: 205–216.

[25]Wei-Neng Chen, Jun Zhang,(2009) "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements",IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART C: APPLICATIONS AND REVIEWS, Volume 39, Issue 1 Pages:29-43.

[26]Xin Lu, Zilong Gu,(2011) “A load-adapative cloud resource scheduling model based on ant colony algorithm”, IEEE.

[27]Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, Wang, D.,(2011) “Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization”, IEEE.

[28]Xiangqian Song, Lin Gao, Jieping Wang,(2011) “Job scheduling based on ant colony optimization in cloud computing”, IEEE.

[29]Cui Lin, Shiyong Lu,(2011) ” Scheduling Scientific Workflows Elastically for Cloud Computing”, In IEEE 4th International Conference on Cloud Computing.

[30]P.Varalakshmi, Aravindh Ramaswamy, Aswath Balasubramanian, and Palaniappan Vijaykumar, (2011) ”An Optimal Workflow Based Scheduling and Resource Allocation in Cloud”, Department of Information Technology, Anna University Chennai, India, Pages: 411–420.

[31]Ke Liu, Jinjun Chen, Yun Yang and Hai Jin,(2008) “A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G”, Concurrency and Computation:Practice and Experience, Wiley, Volume 20, Issue15, Pages:1807-1820.

[32]Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan and H. Jin,(2008) “An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows”, Proc. of 4th IEEE International Conference on e-Science, Indianapolis, USA,Pages:374-375.

[33]K. Liu, Y. Yang, J. Chen, X. Liu, D. Yuan and H. Jin,(2010) “A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-intensive Cost-ConstrainedWorkflows on Cloud Computing Platform”, International Journal of High Performance Computing Applications, Volume 24, Issue 4, Pages:445-456.

[34]Xiaofeng Wang , Chee Shin Yeo, Rajkumar Buyya, Jinshu Su ,(2011) ”Optimizing Makespan and Reliability for Workflow Applications with Reputation and Look-ahead Genetic Algorithm“,International Journal of Future Generation Computer Systems Volume 27, Issue 8,Pages 1124–1134.

[35]Ewa Deelman,(2010) ”Grids and Clouds: Making Workflow Applications Work in Heterogeneous Distributed Environments”,International Journal of High Performance Computing Applications ,International Journal of High August , Volume 24, Issue3, Pages:284-298.

[36]Liyong Zhang , Yan Wen , Yanbo Han,(2010) “ A Proactive Approach to Load Balancing of Workflow Execution in a SaaS Environment”,Fifth IEEE International Symposium on Service Oriented System Engineering.

[37]Fan Zhang, Junwei Cao, Kai Hwang, and Cheng Wu,(2011) “Ordinal Optimized Scheduling of Scientific Workflows in Elastic Compute Clouds”, submitted to IEEE Transactions on Computers.

[38]Zhangjun Wu, Xiao Liu, Zhiwei Ni, Dong Yuan and Yun Yang,(2011) ”A market-oriented hierarchical scheduling strategy in cloud workflow systems”, Springer, The Journal of Supercomputing ,Science+Business Media.

[39]S. Pandey, Wu. Linlin, Guru,Buyya,(2010) ”A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments”, 24th IEEE International Conference on Advanced Information Networking and Applications (AINA),Pages: 400 – 407.

[40]Z.Wu, Z.Ni, L. Gu,(2010) “A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling”. Computational Intelligence and Security (CIS), Pages: 184-188.

[41] Amazon,Amazon EC2 Pricing, Available from: http://aws.amazon.com/ec2/pricing/,2012.

[42] B. Chandra Mohan , R.B.,(2012) “A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. Expert Systems with Applications”, Elsevier. Volume 39, Issues 4, Pages: 4618–4627.

[43]Dorigo M., Blum C.,(2005) “Ant colony optimization theory: A survey” , Elsevier, Theoretical Computer Science, Volume 344, Issues 2-3.