AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Coherent Genetic Algorithm for Task Scheduling in Cloud Computing Environment

[1]M. Krishna Sudha and [2]Dr. S. Sukumaran

[1](Ph.D scholar, Department of Computer Science, Erode Arts and Science College, Erode, Tamilnadu, India)
[2](Associate Professor, Department of Computer Science, Erode Arts and Science College, Tamilnadu, India)

ARTICLE INFO

Article history:
Received 25 October 2014
Received in revised form
26 November 2014
Accepted 29 December 2014
Available online 15 January 2015

Keywords:
Cloudlets, Cloud Computing, Genetic Algorithm, Makespan, Task-Scheduling.

ABSTRACT

Cloud Computing offers on-demand and rapidly provisioned resources which is a computing and widely accepted model. For consumption and delivery of IT services cloud computing represents pay as per usage on internet. Constraint for scheduling of the cloud services to the consumers by service providers is Cost and Time. In such a scenario, efficient task scheduling is required such that the execution cost and time can be reduced. In this paper we proposed a Genetic based scheduling Algorithm which minimizes the execution time and execution cost taking into consideration computational cost and computing capacity of the processing elements. Experimental results show that the proposed algorithm exhibits good performance.

To Cite This Article: M. Krishna Sudha and Dr. S. Sukumaran., Coherent Genetic Algorithm for Task Scheduling in Cloud Computing Environment. ***Aust. J. Basic & Appl. Sci.,*** *9(2): 1-8, 2015*

INTRODUCTION

Cloud refers to a distinct IT environment that is designed for the purpose of remotely provisioning scalable and measured IT resources. Cloud consumers consume cloud services provided by the cloud service provider. Mainly three types of services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are provided by the cloud. Task scheduling and resource Provisioning are main problem areas in both Grid as well as in cloud computing. Scheduling is a process that maps and manages the execution of tasks on distributed resources. Proper scheduling can have significant impact on the performance of the system. Scheduling of jobs in a cloud environment is left to the virtual machine layer through the use of resource virtualization (Sawant, Shailesh, 2011). Various types of scheduling algorithms have been applied on various data workloads and measured with different performance metrics to evaluate the performance. Most of the scheduling algorithms are developed to accomplish two aims such as improve the quality of services with expected output on time and to maintain efficiency and fairness for all jobs (Isam Azawi Mohialdeen, 2013).

Makespan is a measure of the throughput of the heterogeneous computing system like the cloud computing environment (Elzeki, O.M., 2012).Genetic algorithm normally contains four different process such as the initial population, fitness function, cross over and mutation. In this proposed work the resources are allocated to cloudlets based on the cost with mean square and used in Smallest Cloudlet to Fastest Processor algorithm. The CGA specification result is applied to genetic algorithm with user defined initial population value instead of randomly selected value. The experimental results produced are used to find out the best performance among the existing and proposed scheduling algorithms

Related Works:

This section gives a brief overview of works done in different task scheduling modes in a cloud environment using genetic algorithm. Most of the scheduling algorithms attempt to achieve two main objectives namely, to run the user task within the deadline (Sujit Tilak, Prof. Dipti Patil, 2012) and to maintain efficiency (load balancing) and fairness for all tasks.

Ke Liu *et al*. (2010) introduces a novel compromised-time-cost scheduling algorithm which uses the batch scheduling algorithm with cost and time factor as an array of workflow instance parameters. The simulation has demonstrated compromised-time-cost algorithm achieves lower cost than others while meeting the user-designated deadline and reduces the mean execution time than others within the user-designated execution cost.

Suraj Pandey *et al*. (2010) presented a particle swarm optimization based heuristic to schedule applications to cloud resources that uses dependency scheduling algorithm mode with resource utilization and time as

Corresponding Author: M. Krishna Sudha, (Ph.D scholar, Department of Computer Science, Erode Arts and Science College, Erode, Tamilnadu, India).

parameters. It is used for workflow application by varying its computation and communication costs. The experimental result shows that the algorithm achieves cost savings and good distribution of workload onto resources.

Saeed Parsa *et al*. (2009) presented a new task scheduling algorithm RASA. It is a combination of two traditional scheduling algorithms Max-min and Min-min. RASA uses the advantages of Max-min and Min-min algorithms and covers their disadvantages. Algorithm works under the batch scheduling mode with parameter as makespan, though the deadline of each task, arriving rate of the tasks, cost of the task execution on each of the resource and cost of the communication are not considered. The experimental results show that RASA outperforms the existing scheduling algorithms in large scale distributed system.

Sindhu *et al*. (2013) proposed two different scheduling algorithms for scheduling tasks in cloud computing such as the LCFP and SCFP. The algorithms are introduced for reducing their computational complexity and the computing capacity of the processing elements. Both of the algorithms are designed in a private cloud environment so the resources are limited. In the first algorithm named LCFP, the scheduling decisions are based on the computational complexity. In the second algorithm SCFP, the smallest cloudlets are mapped to Processing Elements. This process having high computational power so as to reduce flow time while at the same time taking into account that longer jobs are not starved. And also they are not considering the processing time and total cost.

The activity based costing algorithm works on optimized way of resource allocation in cloud computing. The paper focuses on an optimized activity based costing algorithm in order to get an optimized solution for every single user requirement and to achieve profit more than those in traditional ones (Ashutosh Ingole, 2011).

Paul *et al*. discussed the issue of how to utilize cloud computing resources efficiently and to increase maximum profits with the task scheduling system. For this purpose, the proposed credit based scheduling algorithm evaluates the entire group of tasks in the task queue and finds the minimal completion time of all tasks. The proposed scheduling method considers the scheduling problem as an assignment problem in mathematics where the cost matrix gives the cost of a task to be assigned to a resource (Paul, Sanyal, 2014).

Scheduling Algorithms:

Four different task scheduling algorithms Min-Min, Max-Min, LCFP and SCFP were considered with completion time and total cost as parameters and makespan, execution cost and Average resource utilization ratio as the criteria for comparison. The time spent from the beginning of the first task to the end of the last task for a workflow instance is called makespan. Better performance of the task is evaluated based on shorter mean execution time.

The makespan value for each algorithm is calculated using the formula

M_{ij} = total (C_{ij}) **(1)**

Where

M_{ij} – makespan

C_{ij} – Completion time of the resources (j)

for task(i)

For evaluating system performance total execution cost is needed since cloud computing works based on the concept of renting resources. For calculating total cost of the system, Processing time and the amount of data transferred is taken into consideration. Total cost is computed using the formula as given below:

$$Cost_r = \sum_{r=1}^{m} T_r \ X \ ERC_r \quad \textbf{(2)}$$

Average Resource Utilization Ratio (ARUR) is calculated using the relation

ARUR= mean (rt_j)/Makespan*100% **(3)**

Where ARUR is in the range 0 to 1. The best and most efficient load balancing level is achieved if ARUR equals 1. So, scheduling algorithm will have better performance if ARUR is close to 1.

Min-Min algorithm:

The Min-Min algorithm is simple, runs faster and provides efficient result. The Min-Min algorithm begins with the set of all unassigned task and calculates the completion time for task, which is Minimum among all the available resources. From that result, a particular task is assigned to the corresponding machine with minimum completion time. The completion time for all tasks are updated by adding all the completion time value then the task is removed from the scheduled list. This process is repeated until all the tasks are assigned or allocated to the Resources. With a set of n tasks (T_1, T_2, T_3 … T_n) that needs to be scheduled to m available processors (R_1, R_2, R_3 … R_m) the expected completion time for task i (1=i=n) on processors j (1=j=m) is C_{ij}, Where r_j represents the ready time of the processor r_j and E_{ij} represents the execution time of task T_i on processor R_j. The completion time (Ct_{ij}) of task is achieved by adding the task ready time (r_j) and execution time (E_{ij}). In Min-Min

Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 1-8

algorithm the minimum completion time of task is selected for the processor as T_k which is achieved using the equation

$$T_k < \sum_{i=1}^{n} T_{ij} \quad \textbf{(4)}$$

After finding the task T_k the task is allocated to the processor, the task (T_k) is removed from the task set (T_{ij}) and the ready time of each processor is updated. After updating the ready time, the completion time of each processor (C_{ij}) is updated as,

$$C_{ij} = C_{ij} + C_{ij} \quad \textbf{(5)}$$

Where

C_{ij} - is the completion time of already allocated task to processor.

The completion time of task is added to the next minimum task completion time. This process is repeated until end of the process.

Max-Min algorithm:

The Max-Min algorithm almost performs the same like a Min-Min algorithm. The completion time for task is calculated and the task with maximum completion is allocated to the processor with minimum completion time. Remaining task in the list is updated and the allocated or scheduled task is removed from the list. This process is repeated until the remaining tasks are allocated to the processor. By adding execution time (E_{ij}) and ready time (r_j) the completion time (Ct_{ij}) of each task is calculated. Allocating task to the processor is based on the derivative

$$T_k > \sum_{i=1}^{n} T_{ij} \quad \textbf{(6)}$$

The next step is to allocate the task (T_k) to the processor. After allocating the resource T_k to the processor, both ready time (r_j) and completion time (Ct_{ij}) of the tasks are updated. Those processes are repeated until all tasks are allocated to the processor. The final step is to calculate the makespan value of the task. With assumed set of n task (T_1, T_2, T_3 … T_n) and m available resources the tasks are allocated to the processor based on maximum completion time of task i ($1 \leq i \leq n$) on resources j ($1 \leq j \leq m$).

Longest Cloudlet to Fastest Processor Algorithm:

LCFP algorithm with the genetic algorithm generates the initial population based on randomly selected chromosome value. The remaining genetic algorithm steps are calculated and proved that the GA-LCFP produces the efficient result when compared to other algorithms. Before calculating the makespan using genetic algorithm both the task (T_{ij}) and processors (R_j) is sorted in descending order. GA-LCFP scheduling process uses a set of n tasks (T_1, T_2, T_3 … T_n) and the tasks are allocated to m available processors (R_1, R_2, R_3 … R_m), the result is applied to genetic algorithm for calculating the makespan value. In genetic algorithm the first step is initial population (I) selection which has user defined value. After assigning initial value as the user defined value, each individual in the set is evaluated. In evaluation step the fitness function of the individuals are calculated. This process is completed when the final iteration is reached. Finally the initial population value is added with original value for new generation process where the population is evolved in the optimization process mainly by crossover operations.

Shortest Cloudlet to Fastest Processor Algorithm:

In this algorithm for calculating the makespan value SCFP result is used in the GA. The obtained result is considered as an input to genetic algorithm, also the initial population (I) is user defined value. Once the initial population value is found the next step is the fitness function calculation. Based on the fitness function value the mutation and the cross over value will be varying. Algorithm uses a set of n tasks (T_1, T_2, T_3 … T_n) for allocating to m available processors (R_1, R_2, R_3 … R_m) for calculating makespan using genetic algorithm and the tasks are sorted in ascending order and processes in descending order. If the task (T_{ij}) is not empty the result is applied to genetic algorithm. The genetic algorithm performs five different steps to calculate the makespan value. Starting with the initial population generation, the fitness function is calculated based on the initial population value. This process is repeated until the end of iteration process is reached followed by the mutation and cross over functions calculation and the makespan value is found.

Proposed Coherent Genetic Algorithm:

The proposed CGA algorithm is designed for overcoming limitations present in the SCFP-GA scheduling algorithm where the task is assigned to the processor after calculating the cost with mean function of each resource. The cost with mean function is calculated as

$T(i,j) = T_k(i,j)$
$T_k(i,j) = T_c(i,j) = T_s(i.j)$
(or)
$T_k(i,j) = T_s(i,j)$
$T_c(i,j) = T_s(i,j) + p(i,j) / 2$
Where
$T_s(i,j) \rightarrow$ GA-SCFP result
$T_c(i.j) \rightarrow$ Cost with mean result
$T_k(i,j) \rightarrow$ CGA result
$p(i,j) \rightarrow$cost of the task for the processor

For calculating the cost with mean function ($T_k(i,j)$), the GA-SCFP result ($T_s(i,j)$) is added with the cost value of the task for the processor ($P(i,j)$). The obtained result ($T_s(i,j)+p(i,j)$) is divided by 2. The derived value is the cost with mean value ($T_c(i,j)$). The derived result ($T_c(i,j)$) is compared with the GA-SCFP results($T_s(i,j)$). If the calculated cost with mean value is lesser than the GA-SCFP result, the cost with mean value is considered as task completion time otherwise the GA-SCFP task completion time value is taken as the task completion time value for the processor ($T_k(i,j)$).

Using the equation $T_c(i,j) = T_s(i,j) + p(i,j) / 2$ **(7)**

the cost with mean value ($T_c(i,j)$) is calculated. Equation $T(i,j) = T_k(i,j)$ is derived from both the Equations $T_k(i,j) = T_s(i,j)$ and $T_c(i,j) = T_s(i,j) + p(i,j) / 2$. The comparison process is performed with the Equation $T(i,j) = T_k(i,j)$ containing the original result of the cost for the processor.

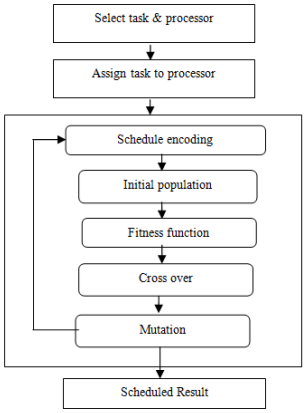

Fig. 1: Flow diagram of CGA algorithm.

In the Proposed algorithm the cost with mean value is found from the GA-SCFP result and the result is compared with the original task completion time of GA-SCFP for overcoming the problems presented in the existing GA-SCFP algorithm and also providing efficient result.

Pseudocode for CGA algorithm:

```
T(i,j) = Tk(i,j)
Do while T(i,j) ← Not NULL
        I ← S
        Initialize ( T(I=S));
       Evaluate (T(I=S));
  While is Not Terminated( )
 do
        Tp(I)←T(I).
        Select Parents();
       Mutate (Tc(I));
      T(I+1)←build next
```

A good scheduling algorithm should lead to better resource utilization and at the same time increase the processing speed and cost. The aim is to find an optimal mapping of tasks to virtual machines and virtual machines to processing elements so as to increase the processing speed and at the same time maximize the

Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 1-8

resource utilization. In a cloud where the resources are limited, makespan and resource utilization play an important role to decide the efficiency of a scheduling algorithm.

CGA-Schedule Encodings:

Schedule Encoding is the first step in the genetic algorithm for which encoding permutation based representation concept is used in the proposed algorithm. Each processor is presented only once in this permutation based representation concept. For the task scheduling problem this representation is obtained in two steps:

1. Create sequence of task(i) and assigned to each processor(j).
2. Concatenate the sequences of task(i).

The resulting sequence of task is a permutation of task (i) assigned to processor (j). This representation requires maintaining additional information on the number of tasks assigned to each processor.

CGA- Initial Population:

Initial population is the set of all the individuals that are used in the genetic algorithm to find out the optimal solution. Every solution in the population is called as an individual and every individual is represented as a chromosome for making it suitable for the genetic operations. The user defined value is used as the initial population of meta heuristic which encode candidate solutions to an optimization problem and evolves better solutions.

CGA-Fitness Function:

A fitness function is used to measure the quality of the individuals in the population according to the given optimization objective. The fitness function (F_{ij}) result is calculated from the initial population value. In initial population stage the user defined value(S) is set as the initial population value(I). After finishing the fitness function calculation of (F_{ij}) the mutation and cross over functions are performed. Based on the fitness function value (F_{ij}) the mutation and crossover value varies. This process is repeated until the end of the process. The fitness function is calculated using the formula

$$(F_{ij}) = T(C_{ij}) / V \quad \textbf{(8)}$$

Where F_{ij} – Fitness function

$T(C_{ij})$ – Completion time of task

V – Process speed

The completion time of task ($T(C_{ij})$) is taken as final completion time for each task as the number of iterations are generated based upon the given initial population value(I). For each time(t) the fitness function value(F_{ij}) is calculated and the final schedule value is considered as the input for generating the fitness function for given input(F_{ij}). Dividing the process speed (V) by the completion time of the task ($T(C_{ij})$) the fitness function(F_{ij}) is achieved. After calculating the fitness function for each individual's the mutation and cross over generation process is completed.

The fitness function for usage cost is defined as:

$$F_c(i) = \frac{1}{Cost_r(i)}, \ 1 \le i \le s \quad \textbf{(9)}$$

where $Cost_r(i)$ is the total usage cost of resource *r*.

CGA-Crossover:

Many crossover operators can be used but the most basic crossover operator is the one-point crossover operator, in this case a crossover point in the string bits of the selected pair is randomly chosen and the bits of the two parents are interchanged at this point. In two-point crossover operation, the two crossover points are selected in the binary strings of the pair under consideration and between these points the bits are swapped. Two-point cross over operator is used in proposed algorithm. The crossover operators are the most important ingredient of any evolutionary-like algorithm. Indeed, by selecting individuals from the parental generation and interchanging their genes, new individuals (descendants) are obtained. The main aim of this work is to obtain descendants of better quality that will feed the next generation and enable the search to explore new regions of solution space not explored yet.

CGA-Mutation:

Mutation process is done by replacing the gene at random position with a new value. Mutation means that the values of some gene locus in the chromosome coding series were replaced by the other gene values in order to generate a new individual. Mutation is that negates the value at the mutate points with regard to binary coded individuals. The genetic algorithm is terminated after reaching the termination condition of user specified value.

Australian Journal of Basic and Applied Sciences, 9(2) February 2015, Pages: 1-8

Implementation And Results:

CloudSim simulator is used for checking the performance of Proposed CGA algorithm with existing Max-Min, Min-Min, LCFP, SCFP algorithms. The CloudSim simulator toolkit supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. A good scheduling algorithm is that which provides the better resource utilization, less average Make-span, less execution cost , maximum resource utilization ratio and better system throughput.

Make-span refers to the total completion time of all cloudlets in the list. Virtual Machines are considered as resources and Cloudlets as tasks/jobs. The cloudlets (R_1, R_2, R_3… R_n) run on processors (P_1, P_2, P_3…..P_n) with an objective to minimize Make-span, the speed of processors is expressed in MIPS (Million Instructions Per Second) and length of the job is expressed as number of instructions to be executed. The proposed CGA algorithm is compared against Min-Min, Max-Min, LCFP and SCFP. From the result it is proved that the proposed algorithm produces efficient result than the existing scheduling algorithms.

In CGA the cost with mean value is calculated after which the result is compared with SCFP result and the respective task for the processor is allocated. Next the scheduling process is initiated. In genetic algorithm the CGA result is used, and the user defined value is set as initial population value. After that the cross over function and mutation calculated respectively. This procedure is repeated till the end of iteration process. Proposed work tests all the algorithms with varying number of cloudlets and varying number of processors.

Table 1: Parameters used in the work.

Cloudlets	10 to 50
Processors	10 to 80
Initial Population Value	User Defined Value
Cross over type	Two-Point Cross Over
Mutation type	Swap
Termination Condition	Iterations

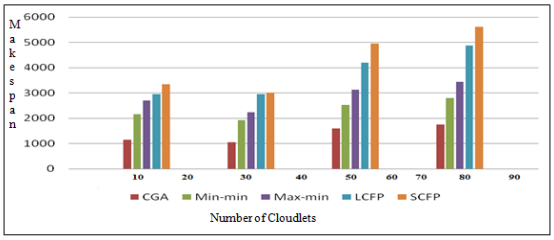

Fig. 2: Shows Average Makespan Vs Number of Cloudlets.

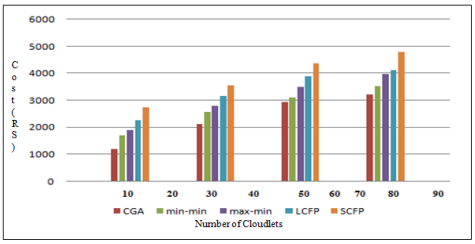

Fig. 3: Shows Average Execution cost Vs Number of Cloudlets.

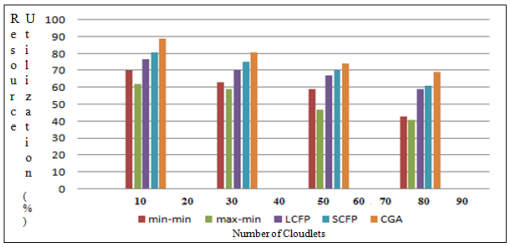

Fig. 4: Shows Average Resource Utilization Vs Number of Cloudlets.

Before allocating the task to the processor in our proposed work we divide each task by the cost because our processed work considered the most important parameter as cost. Because based upon the cost the processor

speed will be varied. We use the following formula to calculate the task for the processor based upon the cost parameter.

$(T_{ij}) = (t_{ij}) / C$ **(10)**

Where

(T_{ij}) →Task for the processor

C → cost

(t_{ij}) → original task

The Equation $(T_{ij}) = (t_{ij}) / C$ can be used to find the task while considering the most important parameter as the cost. The makespan value of the proposed CGA has the least value when compared with the existing scheduling algorithms. Experimental resulting values show that proposed algorithm takes less completion time as compared to existing Min-Min, Max-Min,SCFP and LCFP which is based on the random generation of schedules. CGA with stochastic operators achieved the better results and better resource utilization as cloudlet load is shared equally on all processors. Experimental results show that under heavy loads the coherent Genetic Algorithm exhibits a very good performance. The overall makespan to execute the cloudlets is used as the metric to evaluate the performance of the proposed algorithms.

Conclusion:

Task Scheduling is one of the key issues in the cloud environment. This paper presents an emergence of new task scheduling algorithm called CGA (Coherent Genetic Algorithm). To overcome the limitations of existing task scheduling algorithms two parameters such as the processing time and cost with mean is taken into account along with genetic algorithm for calculating the makespan value, cost of execution and Average Resource utilization of the algorithms from which it is identified that the proposed algorithm (CGA) has the least makespan value, cost of execution and maximum resource utilization. This study is only considered with resources, tasks cost and makespan as fitness criteria. The experimental results show that coherent genetic algorithm minimizes the makespan and utilizes the resources effectively than the existing scheduling algorithms with low cost.

REFERENCES

Kaur, Shaminder, Verma, Amandeep, 2012 "An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment," International Journal of Information Technology & Computer Science, 4(10): 7-15.

Sindhu, S., S. Mukherjee, 2013. "A genetic algorithm based scheduler for cloud environment," 4th International Conference on Computer and Communication Technology, 3(2): 236-239.

Huankai Chen, F., Wang, N. Helian, Akanmu, 2012. "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing" , International Journal of Digital Content Technology, vol 3 issue 7, 99-101.

Ana Madureira, Carlos Ramos, Sílvio do Carmo Silva, Apr, 2000, "Using Genetic Algorithms for Dynamic Scheduling", European journal of operation research, 390-401.

Elzeki, O.M., M.Z. Reshad, M.A. Elsoud, 2012. "Improved Max-Min Algorithm in Cloud Computing", Proceedings of the International Journal of Computer Applications, 30-42.

João Nuno Silva, Luís Veiga, Paulo Ferreira, 2008. "Heuristic for resources allocation on utility computing infrastructures", Journal of Computer Science, 123-151.

Upendra Bhoi, Purvi N. Ramanuj, 2013. "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International Journal of Application or Innovation in Engineering & Management (IJAIEM), 259-264.

Sawant, Shailesh, 2011. "A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment", Journal of computer science and engineering, 176-198.

Pardeep Kumar, Amandeep Verma, 2013. "Scheduling Using Improved Genetic Algorithm in Cloud Computing for Independent Tasks", Proceedings of the International Conference on Advances in Computing, Communications and Informatics, 137-142.

Isam Azawi Mohialdeen, 2013. "Comparative Study of Scheduling Algorithms In Cloud Computing", Journal of Computer Science, 252-263.

Kaustubh Beedkar, Luciano Del Corro, Rainer Gemulla, 2010. "Adaptive Scheduling based on Quality of Service in Heterogeneous Environments", In proceeding of 4th International Conference on Multimedia and Ubiquitous Engineering, 11-13.

Sujit Tilak, Prof. Dipti Patil, 2012. "A Survey of Various Scheduling Algorithms in Cloud Environment", International Journal of Engineering Inventions, 36-39.

Ashutosh Ingole, Sumit Chavan and F. Utkarsh Pawde, 2011. " An Optimized Algorithm for Task Scheduling based on Activity based Costing in Cloud Computing", 2nd National Conference on Information

and Communication Technology , Proceedings published in International Journal of Computer Applications, 34-37.

Sung Ho Jang, Ta e Young Kim, Jae Kwon Kim, Jong Sik Lee, 2012. " The Study of Genetic Algorithm-based Task Scheduling for Cloud Computing", Proceedings of International Journal of Control and Automation, 234-246.

Katyal, Mayanka, Mishra, Atul, 2014. "Application of Selective Algorithm for Effective Resource Provisioning In Cloud Computing Environment", International Journal on Cloud Computing Services & Architecture, 1-10.

Denny Hermawanto, 2013. " Genetic Algorithm for Solving Simple Mathematical Equality Problem", Journal of parallel and Distributed Computing, 45-67.

Rodrigo, N., Calheiros, Rajiv Ranjan, A.F. César De Rose and Rajkumar Buyya, 2009. "CloudSim: A Framework For Modeling And Simulation of Cloud Computing Infrastructures And Services", International symposium on cluster computing and grid, 1-9.

Shalmali Ambike, Dipti Bhansali, Jaee Kshirs sagar, Juhi Bansiwal, 2009. "An Optimistic Differentiated Job Scheduling System for Cloud Computing", Proceedings of the Third International Conference on Multimedia and Ubiquitous Engineering, 295-299.

Saeed Parsa, Reza Entezari-Maleki, 2009. "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications, 91-99.

Salim Bitam, 2012. "Bees Life Algorithm for Job Scheduling in Cloud Computing", The Second International Conference on Communications and Information Technology, 186-191.

Ke Liu1, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan, Yun Yang, 2010. "A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on a Cloud Computing Platform" , The International Journal of High Performance Computing Applications, 445-456.

Pandey, S., Linlin Wu Guru, R. Buyya, 2010. " A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Clou Computing Environments**",** Advanced Information Networking and Applications 24th IEEE International Conference, 400-407.

Paul, Sanyal, 2014. "Credit Based Resource Selection and List Scheduling in Cloud Computing", International Journal of Advances in Engineering & Technology, 2455-2463.