Different Task Scheduling Algorithms in Cloud Computing: A Review

Akshay Kumar[*1], Khushi[2], Neha Bhagat[3], Niranjan Singh[4].

[1]*Department of Computer Science and Engineering, Konark Institute of Science & Technology, Khorda, Bhubaneswar, 752050, India.*

[2]*Department of Computer Science and Engineering, Konark Institute of Science & Technology, Khorda, Bhubaneswar, 752050, India.*

[3]*Department of Computer Science and Engineering, Konark Institute of Science & Technology, Khorda, Bhubaneswar, 752050, India.*

[4]*Department of Computer Science and Engineering, Konark Institute of Science & Technology, Khorda, Bhubaneswar, 752050, India.*

Corresponding Author: Akshay Kumar. Email: akshaykumarsinha0326@gmail.com

Abstract: **Cloud computing is a type of parallel and distributed system consisting of a collection of interconnected and virtual computers. 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 The objective of this paper is the analysis of various task scheduling algorithms in Cloud Computing. This paper reviews the already available task scheduling algorithms in Cloud Computing. There are various scheduling algorithms; some are static scheduling algorithms that are considered suitable for small or medium-scale cloud computing; and dynamic scheduling algorithms that are considered suitable for large-scale cloud computing environments. In this research, we attempt to show the most popular static and dynamic task scheduling algorithm performances, there are: first come first service (FCFS), short job first scheduling (SJF), MAX-MIN, Genetic Based Algorithm, Ant Colony Optimization, etc**.

Keywords: Task scheduling, Scheduling algorithms, Resource allocation, Scheduling strategy, Scheduling metrics, Ant colony optimization, Genetic Based Algorithms, Static task Scheduling

Introduction: What is Cloud Computing?

Cloud Computing is a general term for anything that involves delivering hosted services over the Internet.

These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [1].

Cloud Computing is an emerging web-based technology where information can be accessed by anyone anywhere. Cloud Computing attracts users by minimizing infrastructure investments and resource management costs while presenting a flexible and elastic service.

Cloud Computing makes possible the efficient utilization of resources by allocating and deallocating the hardware and software resources to the customers on demand. So, the customers need not worry about the arrangement of computing resources. They can use these resources without making any costly investment by purchasing them. The customer uses this resource on demand and when these are not needed, the customers can give their resource back to the resource's provider.

Customers will be charged only based on usage.

What is Unique About Cloud Computing?

A cloud service has three distinct characteristics that differentiate it from traditional web hosting.

- It is sold on demand, typically by the minute or the hour.
- It is elastic -- a user can have as much or as little of a service as they want at any given time;
- The service is fully managed by the provider (the consumer needs nothing but a personal computer

  and Internet access).

What Is Unique About Cloud Computing?

A cloud service has three distinct characteristics that differentiate it from traditional web hosting.

- It is sold on demand, typically by the minute or the hour;
- It is elastic -- a user can have as much or as little of a service as they want at any given time;
- The service is fully managed by the provider (the consumer needs nothing but a personal computer

  and Internet access).

CLASSIFICATION OF CLOUD COMPUTING:

Cloud Computing is classified base on the service provided and the kind of deployment:

SOFTWARE AS A SERVICE (SaaS):

Software resources are offered by SAAS providers on request. The customers need not bother about software purchasing costs and other maintenance tasks and related costs. Cloud applications allow the cloud to be leveraged for software architecture, reducing the burdens of maintenance, support, and operations by having the application run on computers belonging to the vendor. Gmail and Salesforce are among the example of SaaS running as clouds, but not all SaaS is on cloud Computing.

INFRASTRUCTURE AS A SERVICE (IaaS):

IaaS gives businesses access to vital web architecture, such as storage space, servers, and connections, without the business need of purchasing and managing this internet infrastructure themselves. IaaS allows an internet business a way to develop and grow on demand. Choosing to use an IaaS cloud demands a willingness to put up with complexity, but with that complexity comes flexibility. Amazon EC2 and Rack space Cloud are examples of IaaS.

INFRASTRUCTURE AS A SERVICE (IaaS):

IaaS, gives businesses access to vital web architecture, such as storage space, servers, and connections, without the business need of purchasing and managing this internet infrastructure themselves. IaaS allows an internet business a way to develop and grow on demand. Choosing to use an IaaS cloud demands a willingness to put up with complexity, but with that complexity comes flexibility. Amazon EC2 and Rack space Cloud are examples of IaaS.

PLATFORM AS A SERVICE (PaaS):

Clouds are created, many times inside IaaS Clouds by specialists to render the scalability and deployment of any application trivial and to help make your expenses scalable and predictable. The chief benefit of a service like this is that for as little as no money you can initiate your application with no stress more than basic development and maybe a little porting if you are dealing with an existing app. Some examples of a PaaS system include: Mosso, Google App Engine, and Force.com.

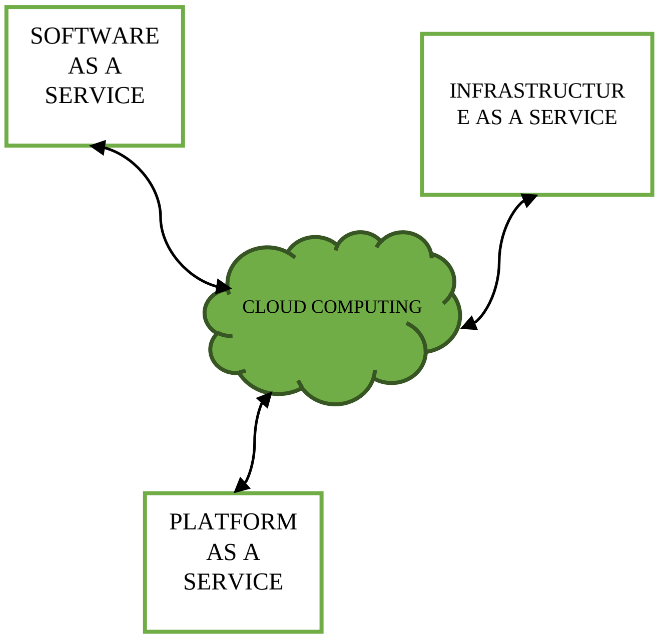


FIGURE 1: DIAGRAM OF CLOUD COMPUTING SERVICE.

Based on deployment models, Cloud is of four types:

PUBLIC CLOUD: A public cloud sells services to anyone on the Internet. (Currently, Amazon Web Services is the largest public cloud provider.) In the public cloud model, a third-party cloud service provider delivers the cloud service over the internet. Public cloud services are sold on demand, typically by the minute or hour, though long-term commitments are available for many services. Customers only pay for the CPU cycles, storage or bandwidth they consume. Leading public cloud service providers include Amazon Web Services (AWS), Microsoft Azure, IBM, and Google Cloud Platform.

PRIVATE CLOUD: A private cloud is a proprietary network or a data center that supplies hosted services to a limited number of people. Private or public, the goal of cloud computing is to provide easy, scalable access to computing resources and IT services. services are delivered from a business's data center to internal users. This model offers the versatility and convenience of the cloud while preserving the management, control, and security common to local data centers. Internal users may or may not be billed for services through IT chargeback. Common private cloud technologies and vendors include VMware and OpenStack.

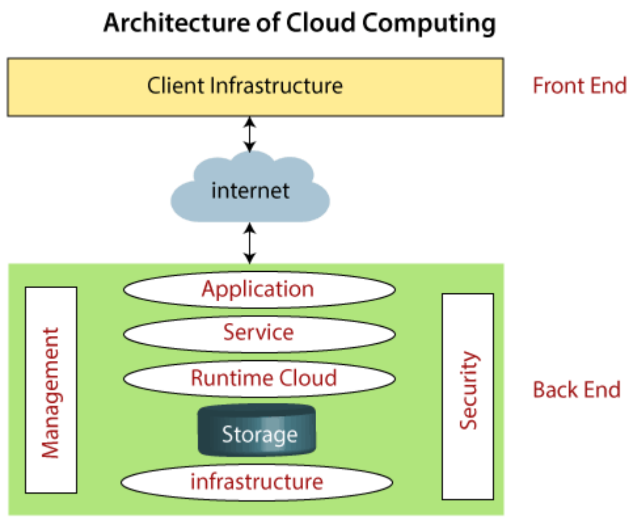
HYBRID CLOUD: A hybrid cloud is a combination of public cloud services and an on-premises private cloud, with orchestration and automation between the two. Companies can run mission-critical workloads or sensitive applications on the private cloud and use the public cloud to handle workload bursts or spikes in demand. The goal of a hybrid cloud is to create a unified, automated, scalable environment that takes advantage of all that a public cloud infrastructure can provide, while still maintaining control over mission-critical data.

COMMUNITY CLOUD: Where two or more organizations with common interests jointly share a common cloud.

CLOUD ARCHITECTURE:
Cloud Architecture is built with two vital parts i.e., its front-end and the back-end.

- Front End – It is that part of the cloud which is seen by the end-user or client. All the applications that the client is using over the cloud are accessed using the internet and over a web browser.
- Back-End – It is the cloud service provider's Data Centre where all the infrastructure devices like Servers, Switches, Routers, Storage, Firewalls, etc are placed.

Below figure 2 shows cloud computing architecture:



CLOUDSIM:
Simulation is a technique where a program models the behaviour of the system (CPU, networ ,etc.,) by calculating the interaction between its different entities using mathematical formulas, or actually capturing and playing back observations from a production system. 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].

CLOUDSIM CHARACTERISTICS:
Cloudsim can be used to model datacentres, host, service brokers, and scheduling and allocation policies of a large scaled cloud platform. Hence, the researcher has used cloudsim to model datacentres, hosts, VMs for experimenting in simulated environments [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].

All the VMs in a data centre not necessarily 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.

CLOUDSIM DATA FLOW:

Each data center 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 the application's quality of service, and hardware and software requirements. In thecase, match occurs the broker deploys the application with the cloud that was suggested by the CIS [3].

THE CLOUDSIM PLATFORM:

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

- **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].

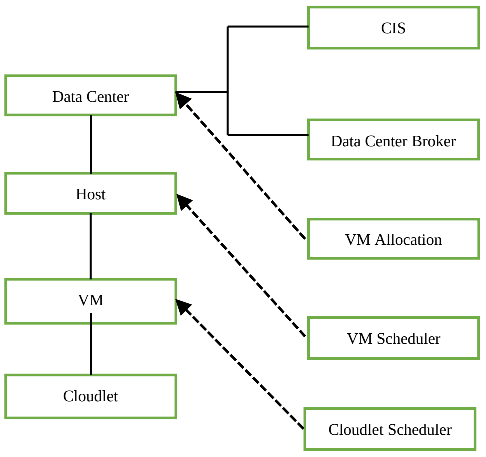


Figure 3: Main Parts of Cloudsim related to our experiment.

TASK SCHEDULING ALGORITHM OVERVIEW:

Cloud Service Provider's Data Centre has a large number of Servers and other computing infrastructure. Thousands of Virtual Machines run inside a data centre to utilize the resources in the best possible manner. Task Scheduling can be done to efficiently utilize the resources by allocating specific tasks to specific resources. It automatically improves the quality of service and performance. These task scheduling algorithms are categorized into two different categories, one is the Batch Mode Heuristic algorithm finds the task with the least of execution time and then assigns the resource to that task that produces the least execution time. If multiple resources provide the same amount of execution time, then the resource is selected on a random basis. Various BMHA examples are FCFS Algorithm, Round-Robin Algorithm, Min-Min Algorithm, and Max-Min Algorithm. Online Mode Heuristic Algorithms working a way that tasks are scheduled as they arrive the system.
Some Important Terms that are used in these scheduling algorithms-
A. T= Arrival Time
B. T= Burst Time
C.T= Completion Time
T.T = Turnaround Time = Completion Time - Arrival Time = C.T - A.T
W.T = Waiting Time = Turnaround Time - Burst Time = T.T - B.T

SCHEDULING-LEVELS:

In the cloud computing environment, there are two levels of scheduling algorithms:

- First level: in host level where a set of policies to distribute VMs in the host.
- Second level: in VM level where a set of policies to distribute tasks to VM.

In this research we focus on VM level to scheduling tasks, we selected task scheduling algorithms as a research field because it is the biggest challenge in cloud computing and the main factor that controls the performance criteria such as (execution time, response time, waiting time, network, bandwidth, services cost) for all tasks and controlling other factors that can affect performance such as power consumption, availability, scalability, storage capacity, buffer capacity, disk capacity, and the number of users.

TASKS SCHEDULING ALGORITHMS DEFINITION AND ADVANTAGES:

Tasks scheduling algorithms are defined as a set of rules and policies used to assign tasks to the suitable resources (CPU, memory, and bandwidth) to get the highest level possible of performance and resource utilization.

Task scheduling algorithm's advantages:

1. Manage cloud computing performance and QoS.
2. Manage the memory and CPU.
3. Good scheduling algorithms maximize resource utilization while minimizing the total task execution time.
4. Improving fairness for all tasks.
5. Increasing the number of successfully completed tasks.
6. Scheduling tasks on a real-time system.
7. Achieving a high system throughput.
8. Improving load balance

TASK SCHEDULING ALGORITHM CLASSIFICATION:

The Task Scheduling algorithm is classified in **Figure 4.**

TASKS SCHEDULING ALGORITHMS CAN BE CLASSIFIED AS FOLLOWS:

• Immediate scheduling: when new tasks arrive, they are scheduled to VMs directly.

• Batch scheduling: tasks are grouped into a batch before being sent; this type is also called mapping events.

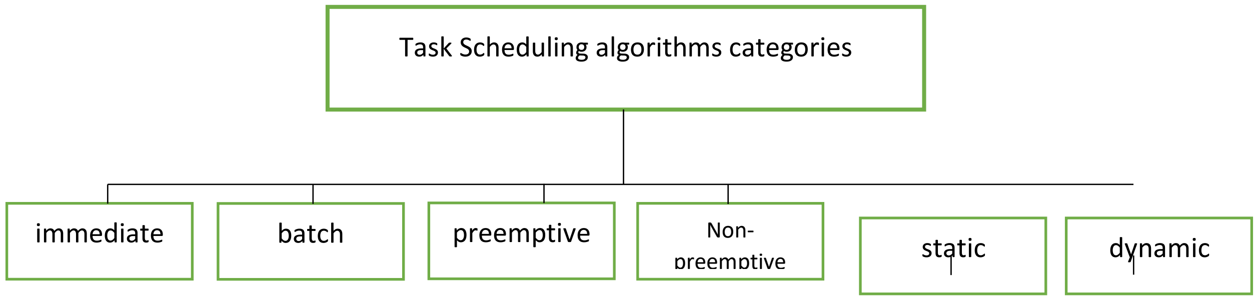
• Static scheduling: is considered very simple compared to dynamic scheduling; it is based on prior information of the global state of the system. It does not take into account the current state of VMs and then divides all traffic equivalently among all VMs in a similar manner such as round robin (RR) and random scheduling algorithms.

• Dynamic scheduling: takes into account the current state of VMs and does not require prior information of the global state of the system and distributes tasks according to the capacity of all available VMs [4–6].

• Pre-emptive scheduling: each task is interrupted during execution and can be moved to another resource to complete execution [6].

• Non-pre-emptive scheduling: VMs are not re-allocated to new tasks until finishing executing the scheduled task [6].

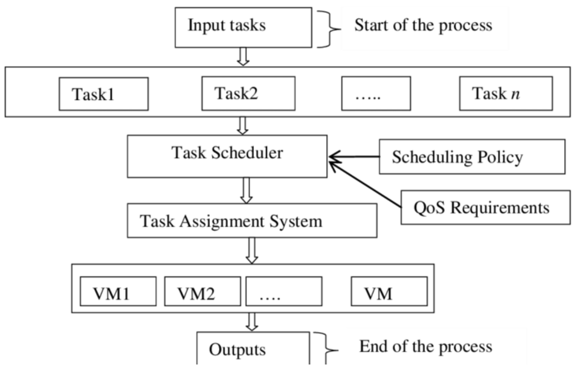
In this research, we focus on static and dynamic scheduling algorithms. Static scheduling algorithms as first come first serve (FCFS), shortest job first (SJF), and MAX-MAX scheduling algorithms in complexity and cost within a small or medium scale. And, as dynamic scheduling algorithms such as Genetic and Ant Colony Optimization.



TASK SCHEDULING SYSTEM IN CLOUD COMPUTING:

The task scheduling system in cloud computing passes through three levels [7].

1. The first task level: is a set of tasks (Cloudlets) that are sent by cloud users, which are required for execution.
2. The second scheduling level: is responsible for mapping tasks to suitable resources to get the highest resource utilization with minimum makespan. The makespan is the overall completion time for all tasks from the beginning to the end [7].
3. The third VMs level: is a set of (VMs) that are used to execute the tasks as in **Figure 5**.



This level passes through two steps:

• The first step is discovering and filtering all the VMs that are presented in the system and collecting status information related to them by using a datacentre broker [8].
• In the second step a suitable VM is selected based on task properties [8].

TASKS SCHEDULING ALGORITHMS IN CLOUD COMPUTING ENVIRONMENT:

FCFS- FCFS: the order of tasks in the task list is based on their arrival time and then assigned to VMs [3].

Advantages:

1. Most popular and simplest scheduling algorithm.
2. Fairer than other simple scheduling algorithms.
3. Depend on the FIFO rule in scheduling tasks.
4. Less complexity than other scheduling algorithms.

Disadvantages:

1. Tasks have a high waiting time.
2. Not give any priority to tasks. That means when we have large tasks in the beginning tasks list, all tasks must wait a long time until large tasks to finish.
3. Resources are not consumed in an optimal manner.
4. In order to measure the performance achieved by this method, we will be testing them and then measuring their impact on (fairness, ET, TWT, and TFT).

Assumptions

Some of the assumptions must be taken into account when scheduling tasks to VMs in the cloud computing environment.

- Number of tasks should be more than the number of VMs, which means that each VM must execute more than one task.
- Each task is assigned to only one VM resource.
- Lengths of tasks vary from small, medium, and large.
- Tasks are not interrupted once their executions start.
- VMs are independent in terms of resources and control.
- The available VMs are of exclusive usage and cannot be shared among different tasks. It means that the VMs cannot consider other tasks until the completion of the current tasks is in progress.

Tasks lengths: assume we have 15 tasks with their lengths as in **Table 1:**

Task	**Length**
t1	100000
t2	70000
t3	5000
t4	1000
t5	3000
t6	10000
t7	90000
t8	100000
t9	15000
t10	1000
t11	2000
t12	4000
t13	20000

t14	25000
t15	80000

VM PROPERTIES:

Assume we have six VMs with different properties based on tasks size:

VM list = {VM1, VM2, VM3, VM4, VM5, VM6}.

MIPS of VM list = {500, 500, 1500, 1500, 2500, 2500}.

We selected a set of VMs with different properties to make each category have VMs with the appropriate ability to serve a specific class of tasks, to improve the load balance. Because when we use VMs with the same properties with all categories it leads to load imbalance, where each class is different from other classes in terms of task lengths.

When applying FCFS, the working mechanism will be as follows Figure shows the FCFS tasks scheduling algorithm working mechanism and how tasks are executed based on their arrival time.

- Dot arrows refer to the first set of task scheduling based on their arrival time.
- Dash arrows refer to the second set of task scheduling based on their arrival time.
- Solid arrows refer to the third set of task scheduling based on their arrival time. And here it is clear to us that t1 is too large compared with t7 and t12. However, t7 and t12 must wait for t1, which leads to an increase in the TWT, ET, TFT, and a decrease in fairness.

VM1 = {t1 → t7 → t12}.

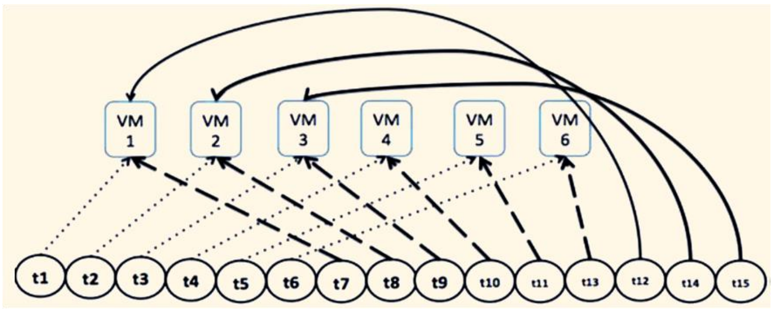
VM2 = {t2→ t8 → t14}.

VM3 = {t3→t9→t15}.

VM4 = {t4→t10}.

VM5 = {t5→t11}.

VM6 = {t6→13}.



Task	ET	Waiting Time		
t1	200	VM1		
t2	140	VM2		
t3	3.33	VM3		
t4	0.66	VM4		
t5	1.2	VM5		
t6	4	VM6		
t7	180	Wait (200)	VM1	
t8	200	Wait (140)	VM2	
t9	10	Wait (3.33)	VM3	
t10	0.66	Wait (0.66)	VM4	
t11	0.8	Wait (1.2)	VM5	
t12	1.6	Wait (4)	VM6	
t13	40	Wait (380)		VM1
t14	50	Wait (340)		VM2
t15	53.33	Wait (13.33)		VM3

Table 2: Waiting time of tasks in FCFS

Shortest Job First (SJF):

Tasks are sorted based on their priority. Priority is given to tasks based on task lengths and begins from.

(Smallest task = highest priority).

Advantages

• Wait time is lower than FCFS.

• SJF has a minimum average waiting time among all task scheduling algorithms

Disadvantages:

• Unfairness to some tasks when tasks are assigned to VM, due to the long tasks tending to be left waiting in the task list while small tasks are assigned to VM.

• Taking long execution time and TFT.

SJF work mechanism:

When applying SJF, the working mechanism will be as follows:

Assume we have 15 tasks as in **Table 1** above. We will be sorting tasks in the task list, as in **Table 3**. Tasks are sorted from smallest task to largest task based on their lengths as in **Table 3**, then assigned to VMs list sequential.

Execute tasks will be

VM1 = {t4 ⟶ t6 ⟶ t7}.

VM2 = {t10 ⟶ t9 ⟶ t1}.

VM3 = {t11 ⟶ t13 ⟶ t8}.

VM4 = {t5 ⟶ t14}.

VM5 = {t12 ⟶ t2}.

VM6 = {t3 ⟶ t15}.

Task	**Length**
t1	100000
t2	70000
t3	5000
t4	1000
t5	3000
t6	10000
t7	90000
t8	100000
t9	15000
t10	1000
t11	2000
t12	4000
t13	20000
t14	25000
t15	80000

Task	**Length**
t4	1000
t10	1000
t11	2000
t5	3000
t12	4000
t3	5000
t6	10000
t9	15000
t13	20000
t14	25000
t2	70000
t15	80000
t7	90000
t1	100000
t8	100000

Table 3: A set of tasks sorted based on SJF Algorithm.

Task	**ET**	**Waiting Time**		
t4	2	VM1		
t10	2	VM2		
t11	1.33	VM3		
t5	2	VM4		
t12	1.6	VM5		
t3	2	VM6		
t6	20	Wait (2)	VM1	
t9	30	Wait (2)	VM2	
t13	13.33	Wait (1.33)	VM3	
t114	16.66	Wait (2)	VM4	
t12	28	Wait (1.6)	VM5	
t115	32	Wait (2)	VM6	
t7	180	Wait (22)		VM1
t1	200	Wait (32)		VM2
t8	66.66	Wait (14.66)		VM3

Table 4: Waiting time of tasks in SJF.

MAX-MIN:

In MAX-MIN tasks are sorted based on the completion time of tasks; long tasks that take more completion time have the highest priority. Then assigned to the VM with minimum overall execution time in VMs list.

Advantages

• Working to exploit the available resources in an efficient manner.

• This algorithm has better performance than the FCFS, SJF, and MIN-MIN algorithms.

Disadvantages

• Increase waiting time for small and medium tasks; if we have six long tasks, in the MAX-MIN scheduling algorithm they will take priority in six VMs in the VM list, and short tasks must wait until the large tasks finish.

Unfairness to some or most small and medium tasks when tasks are assigned to VM.

• When applying MAX-MIN, the Work Mechanism will be as follows.

Task	**Length**
t1	100000
t2	70000
t3	5000
t4	1000
t5	3000
t6	10000
t7	90000
t8	100000
t9	15000
t10	1000
t11	2000
t12	4000
t13	20000
t14	25000
t15	80000

Task	**Length**
t1	100000
t8	100000
t7	90000
t15	80000
t2	70000
t14	25000
t13	20000
t9	15000
t6	10000
t3	5000
t12	4000
t5	3000
t11	2000
t10	1000
t4	1000

Table 5: A set of tasks sorted based on the MAX-MIN scheduling algorithm.

Assume we have 15 tasks as in **Table 1** above. We will be sorting tasks in the task list as in **Table 5**. Tasks are sorted from largest task to smallest task based on the highest completion time. They are then assigned to the VMs with minimum overall execution time in the VMs list.

Execute tasks will be:

VM6 = {t1 ⟶ t13 ⟶ t11}.

VM5 = {t8 ⟶ t9 ⟶ t10}.

VM4 = {t7 ⟶ t6 ⟶ t4}.

VM3 = {t15 ⟶ t3}.

VM2 = {t2 ⟶ t12}.

VM1 = {t14 ⟶ t5}.

Tables 6 shows that the small and medium tasks must be waiting in the task list until the large tasks finish execution. **Figure 4** shows the TWT and TFT for the three tasks scheduling algorithms FCFS, SJF, and MAX-MIN. SJF tasks scheduling algorithm is the best in terms of TWT and TFT.

Task	ET	Waiting Time		
t1	40	VM6		
t8	40	VM5		
t7	60	VM4		
t15	53.33	VM3		
t2	140	VM2		
t14	50	VM1		
t13	8	Wait (40)	VM6	
t9	6	Wait (40)	VM5	
t6	6.66	Wait (60)	VM4	
t3	3.33	Wait (53.33)	VM3	
t12	8	Wait (140)	VM2	
t5	6	Wait (50)	VM1	
t11	0.8	Wait (48)		VM6
t4	0.4	Wait (46)		VM5
t10	0.67	Wait (66.67)		VM4

Table 6: Waiting time of tasks in MIX-MIN scheduling algorithm.

Let us compare all the static scheduling algorithms together.

	FCFS	SJF	MAX-MIN
TWT	739.19	79.59	404
TFT	1969.69	678.69	968.698

Table 7: Comparison between FCFS task scheduling algorithm, SJF, and MAX-MIN in terms of TWT and TFT.

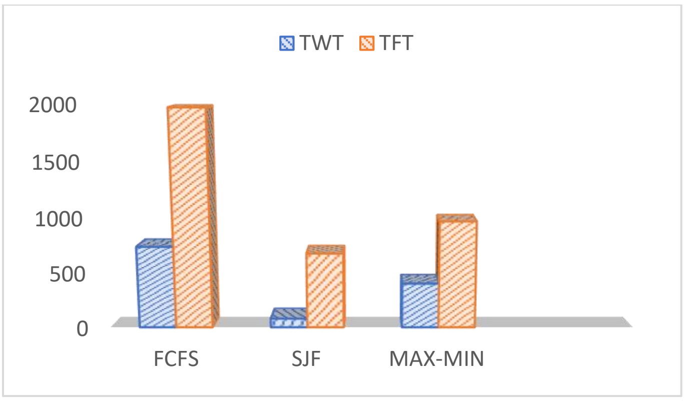


Figure 6: Comparison between FCFS task scheduling algorithm, SJF, and MAX-MIN in terms of TWT and TFT.

Now, we will look at dynamic task scheduling algorithms, and execute and compare some of their methods.

So, in dynamic task scheduling algorithms, we'll see two methods:

1. Ant Colony Optimization (ACO).
2. Genetic-based task scheduling algorithm.

ANT COLONY OPTIMIZATION (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 number of pheromones accumulated on this path. The advantages of the algorithm are the use of the positive feedback mechanism, inner parallelism and, extensible. The disadvantages are overhead and the stagnation of the 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. It is clear that an ACO algorithm can be applied to any of the combinatorial problems 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 that forces the construction of feasible solutions.
4. A pheromone updating rule specifies how to modify the pheromone trail τ on the edges of the graph.
5. A probabilistic transition rule of the heuristic desirability and of pheromone trail.

In this section, a cloud task scheduling-based ACO algorithm will be proposed. Decreasing the makespan of tasks is the basic idea of 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. 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.

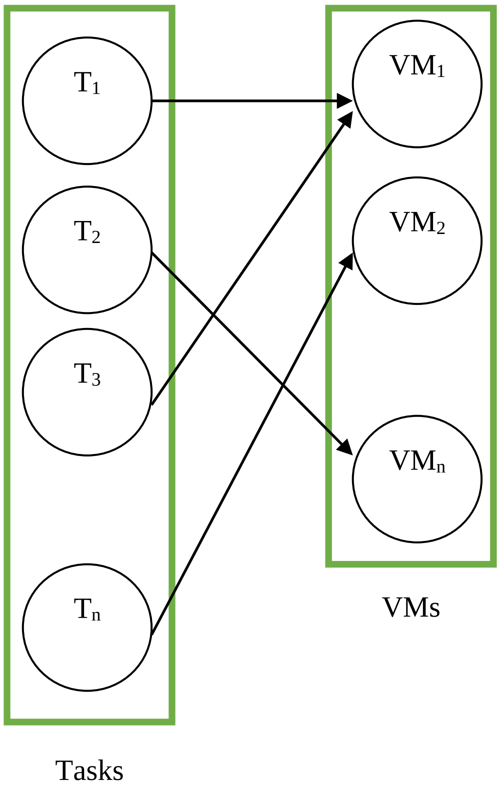


Figure 7: Problem representation of task scheduling-based ACO.

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 type of the 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 the next task, and pheromone updating as following:

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 $T_{ij}(t)=c$ 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 tabuk.
      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 < tmax)
        Empty all tabu lists.
                Goto step 2
        Else
                Print current_optimal_solution.
        End If
9. Return
```

Initializing Pheromone:

The amount of virtual pheromone trail $T_{ij}(t)$ on the edge connects task i to VM_j. The initial amount of pheromone on edges is assumed to be a small positive constant τ0 (homogeneous distribution of pheromone at time t=0).

VM Choosing Rule for Next Task:

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

$\boldsymbol{P}_{ij}{}^{k}(t) = \ [T_{ij}(t)^{\alpha} * [\eta_{ij}]^{\beta} / \sum s€allowed_k \ [T_{is}(t)]^{\alpha} * [\eta_{ij}]^{\beta}$

If j € allowed $_k$ *(1), otherwise*

Where, $T_{ij}(t)$ shows the pheromone concentration at the t time on the path between task i and VMj, allowed k={0,1,…,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 ηij=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 VMj can be computed with Equation 2.

$d_{ij} = TL\_Task_i / Pe\_num_j * Pe\_mips_j + InputFileSize / VM_j$

Where, TL_Taski is the total length of the task that has been submitted to VMj, Pe_numj is the number of VMj processors, Pe_mipsj is the MIPS of each processor of VMj, InputFileSize is the length of the task before execution and VM_bwj is the communication bandwidth ability of the VMj. 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.

Pheromone Updating:

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

$\boldsymbol{\Delta_{ij}{}^{k}(t) - \{Q/ L^{k}(t)}$

if (i,j)€ $\boldsymbol{T^{k}(t)}$ ***(3), if (i,j)ϵ*** $\boldsymbol{T^{k}(t)}$***otherwise,***

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

$\boldsymbol{L^{k}(t) - argmax_{j€j} \{sum_{icIJ}(d_{ij}) \} \qquad (4)}$

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

$\boldsymbol{T_{ij}(t) - (1-p)T_{ij}(t) + \Delta T_{ij}(t) \qquad (5)}$

Where, ρ is the trail decay, 0< ρ< 1 and Tij(t) is computed by Equation 6.

$\boldsymbol{\Delta T_{ij}(t) - \sum_{k-1}{}^{m} \Delta^{k}{}_{Tij}(t) \qquad (6)}$

When all ants complete a traverse, an elitist is an ant that 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.

$\boldsymbol{T_{ij}(t) \ - \ T_{ij}(t) \ + Q/L^{+} \ if \ (i,j) € T^{+} \qquad (7)}$

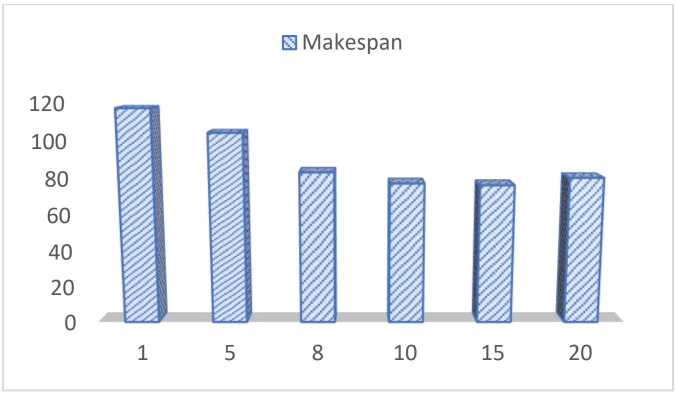
PARAMETERS SETTING OF CLOUDSIM:

The experiments are implemented with 5 Datacentres with 6 VMs and 15 tasks under the simulation platform. The length of the task is from 100000 Instructions (MI). The parameters setting of the cloud simulator are shown in Table.

ACO Parameters Evaluation and Setting:

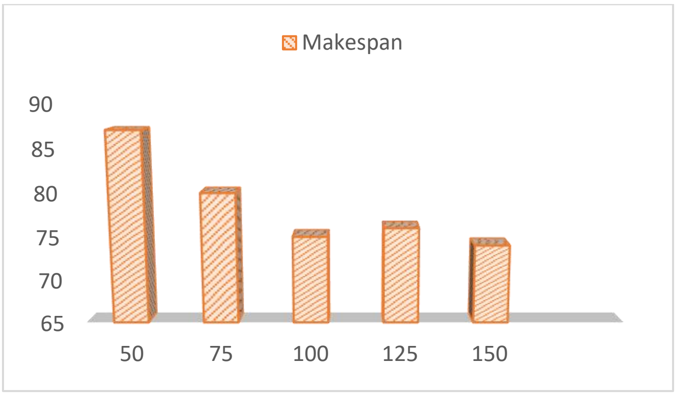
We implemented the ACO algorithm and investigated their relative strengths and weaknesses by experimentation. The parameters (α, β, p, tmax, m the number of ants and Q) considered here are those that affect directly or indirectly the computation of the algorithm. We tested several values for each parameter while all the others were held constant on 100 tasks.

The default value of the parameters was α=1, β=1, ρ=0.5, Q=100, tmax=150 and m=8. In each experiment only one of the values was changed, the values tested were: α ∈ {0, 0.1, 0.2, 0.3, 0.4, 0.5}, β ∈ {0, 0.5, 1.5, 2, 2.5, 3}, ρ ∈ {0, 0.1, 0.2, 0.3, 0.4, 0.5}, Q ∈ {1, 100, 500,1000}, tmax ∈ {50, 75, 100, 150} and m ∈ {1, 5, 8, 10,15, 20}. We also use the time in the cloudSim torecord the makespan. The ACO performance for different values of parameters (α, β, p, tmax, m the number of ants and Q) has been evaluated. The ACO performance for different values of parameters (m: The number of ants, tmax, 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 tmax 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 tmax parameter to reduce the overhead of the ACO algorithm. Table 2 shows the selected best parameters of ACO.



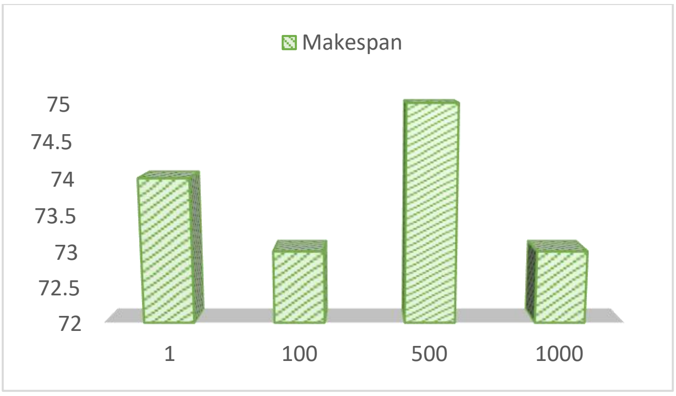
α=1, β=1, r=0.5, Q=100, and t_{max}=200.

Figure 8: ACO performance for different values of ant numbers.



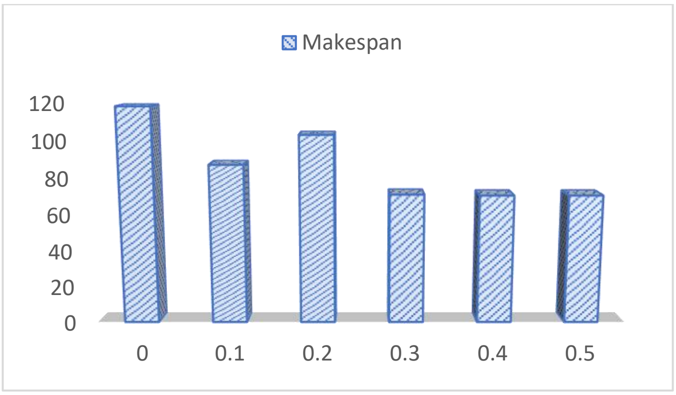
α=1, β=1, r=0.5, Q=100 and m=10

Figure 9: ACO performances on different values of Q.



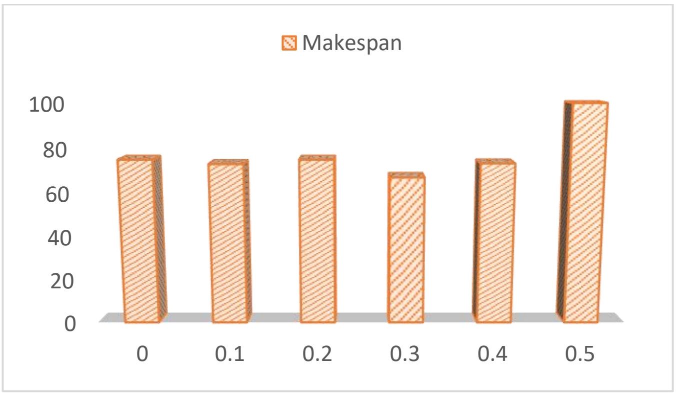
α=1, β=1, r=0.5, tmax=100 and m=10.

Figure 10: ACO Performance for different value of Q.



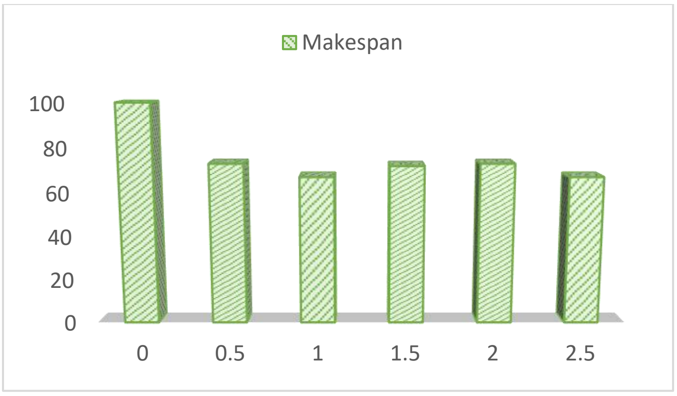
α=1, β=1, Q=100, tmax=100 and m=10.

Figure 11: ACO performance for different value of RHO.



β=1, ρ=0.4, Q=100, tmax=100 and m=10.

Figure 12: ACO performance for different values of alpha.



α=0.3, ρ=0.5, Q=100, tmax=100 and m=10.

Figure 13: ACO performance for different values of beta.

Parameters	α	β	ρ	Q	M	t_{max}
Value	0.3	1	0.4	100	10	100

Table 8: Selected parameters of ACO.

GENETIC ALGORITHM:

The genetic algorithm (GA) is based on the biological concept of generating the population. GA is considered a rapidly growing area of Artificial Intelligence. By Darwin's theory of evolution was inspired the Genetic Algorithms (GAs). According to Darwin's theory, term “Survival of the fittest” is used as the method of scheduling in which the tasks are assigned to resources according to the value of fitness function for each parameter of the task scheduling process [13]. The main principles of the GA are described as follows

1. INITIAL POPULATION: The initial population is the set of all individuals that are used in the GA to find out the optimal solution. Every solution in the population is called as an individual. Every individual is represented as a chromosome for making it suitable for the genetic operations. From the initial population, the individuals are selected, and some operations are applied on them to form the next generation. The mating chromosomes are selected based on some specific criteria.
2. FITNESS FUNCTION: The productivity of any individual depends on the fitness value. It is the measure of the superiority of an individual in the population. The fitness value shows the performance of an individual in the population. Therefore, the individuals survive or die out according to the fitness or function value. Hence, the fitness function is the motivating factor in the GA.
3. SELECTION: The selection mechanism is used to select an intermediate solution for the next generation based on the Darwin's law of survival. This operation is the guiding channel for the GA based on the performance. There are various selection strategies to select the best chromosomes such as roulette wheel, Boltzmann strategy, tournament selection, and selection based on rank.
4. CROSSOVER: Crossover operation can be achieved by selecting two parent individuals and then creating a new individual tree by alternating and reforming the parts of those parents. Hybridization operation is a guiding process in the GA and it boosts the searching mechanism.
5. MUTATION: After crossover, mutation takes place. It is the operator that introduces genetic diversity in the population. The mutation takes place whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators. It occurs during evolution according to a user-defined mutation probability, usually set to fairly low. Mutation alters one or more gene values in the chromosome from its initial state. This can produce the entirely new gene values being added to the gene pool. With this new gene values, the genetic algorithm may be able to produce a better solution than was previously.

GA-ALGORITHM:

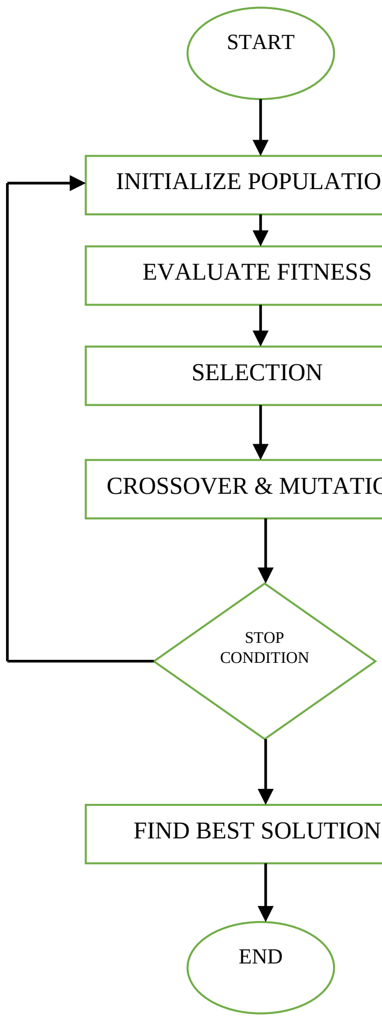


Figure 14: Genetic algorithm.

1. **Initialize Population:**

According to the proposed TS-GA algorithm, the population is randomly generated using encoded binary (0, 1). Therefore, the representation of solutions in task scheduling for each gene or (chromosome) consists of VM ID and ID for each task to be executed on these VM (see Fig. 3). Each VM and the executed tasks on it are encoded into the binary bit (e.g., VM3: - TS4-TS8-TS9 □ [0110 – 0100-1000-1001]).

2. **The Fitness Function Representation**

The main objective of task scheduling in Cloud computing is to reduce the completion time for execution of all tasks on the available resources. Therefore, the completion time of task T_i on VM_j as CT_{ij} is defined using equation 8.

$$\textbf{Completion Time} = \mathbf{CT_{max}[i,j]}$$

$$i \in T,\ i=1, 2, 3, \ldots n$$

$$j \in VM,\ j = 1, 2, 3, \ldots m \qquad (8)$$

Where CTmax denotes maximum time for complete Task i on. n and m denote the number of tasks and virtual machines respectively. Therefore, to reduce the completion time which can be denoted as, the execution time of each task for each virtual machine must be calculated for the scheduling purpose. If the processing speed of the virtual machine is, then the processing time for task Pi can be calculated by equation (9).

$$P_{ij} = C_i / P_{sj} \qquad (9)$$

Where, the processing time for task Pi on VMj and Ci computational complexity of task Pi. The processing time of each task in the virtual machine can be calculated by equation (10).

$$P_j = \sum^{n}_{i=1} P_{ij} \qquad (10)$$

3. **Selection Process:**

Tournament selection is computationally more efficient and more amenable to parallel implementation [16]. Therefore, the developed TS-GA algorithm, Tournament Selection is used to overcome the limitation of the population size. Two individuals are chosen at random from the population. A random number r is then chosen between 0 and 1. If r < k (where k is a parameter, for example, 0.75), the fitter of the two individuals is selected to be a parent; otherwise, the less fit individual is selected. The non-chosen individuals are then returned to the original population and could be selected again.

4. **Crossover:**

In the proposed TS-GA algorithm, the new crossover has been used differently from the used crossover in the original GA. Therefore, two chromosomes that are selected for the crossover process to generate two offspring will be considered as offspring also. So, the proposed crossover produces four children (see Figure 15). After that, the two best children are chosen from these.

5. **Initialize Subpopulation:**

After each iteration, subpopulations (i.e., new populations after crossover) are added into old populations (i.e., parents). This step can enhance the diversity of the population. This idea is introduced by the GA algorithm.

6. Keep the Best Solution

There is a solution that might satisfy a good fitness function, but it is not selected during the crossover process. By the proposed GA algorithm, this solution is not removed from the population, but it is chosen and added to the population when the next iteration is started. This step is considered a good step as some of the iterations can generate the best solution.

Generally, according to the GA algorithm, a set of modifications have been introduced. These modifications are as follows.

- The tournament is used instead of the roulette wheel in the selection process to select the best solution.
- The solutions not chosen in the selection process are considered and added to the new population. This might help in generating the best solution for the next generations.
- The new crossover is introduced by considering parents individuals as a new child (see Fig. 15).

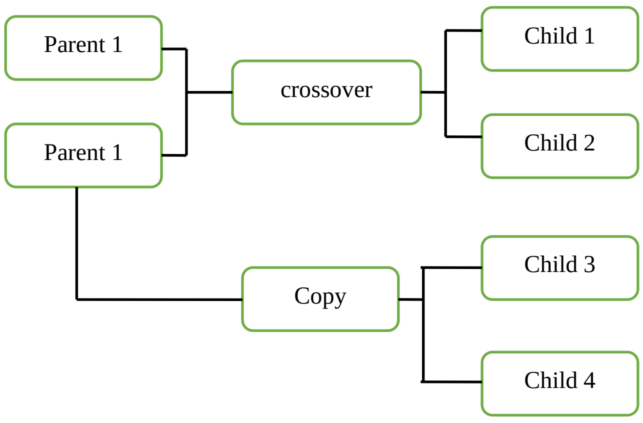


Figure 15: Crossover Process.

- After each iteration, subpopulations (i.e., new populations after crossover) are added into old populations (i.e., parents).

In this section, the experimental evaluation of the proposed GA algorithm, and ACO algorithms is presented, starting by describing the experimental environment.

IMPLEMENTATION RESULTS OF ACO, FCFS, SJF, MIN-MAX, AND GA TASK SCHEDULING ALGORITHM:

In the following experiments, we compared the average makespan with different task sets. The average makespan of the ACO, SJF, MIN-MAX, FCFS, and GA algorithms are shown in Figure 16. It can be seen that, with the increase of the quantity task, ACO takes time less than RR and FCFS algorithms. This indicates that the ACO algorithm is better than any other algorithms.

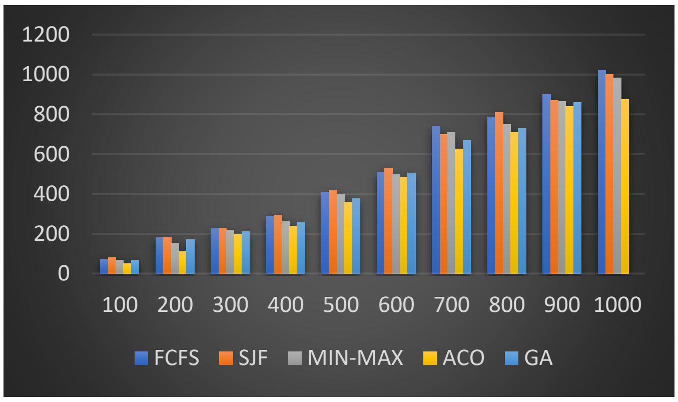
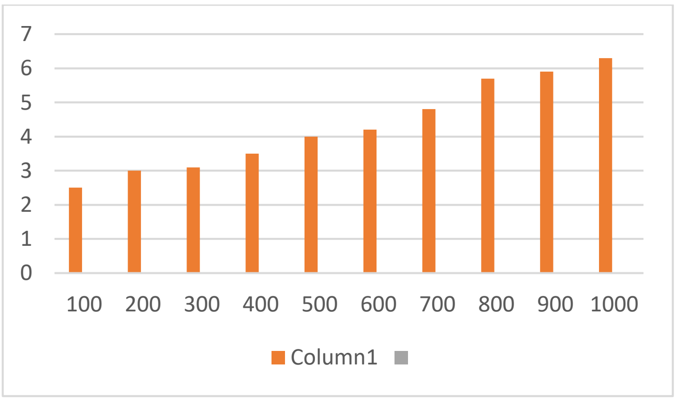


Figure 16: Average makespan of FCFS, SJF, MIN-MAX, ACO, and GA algorithms.

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.



The Degree of Imbalance (DI) measures the imbalance among VMs, which is computed by Equations 11 and 12.

$$T_i = TL\_Tasks / Pe\_num_J * Pe\_mips_j \qquad (11)$$

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

$$DI = T_{max} - T_{min} / \ T_{avg} \qquad (12)$$

Where, Tmax, Tmin and Tavg are the maximum, minimum and average Ti 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.

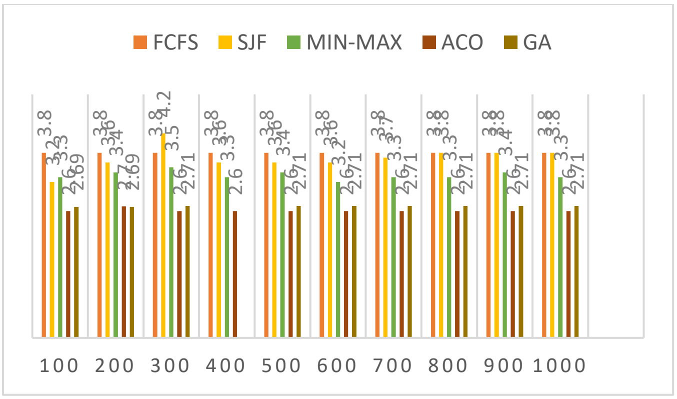


Figure 18: Average DI of FCFS, SJF, MIN-MAX, ACO, GA Task Scheduling Algorithm.

CONCLUSION:

This chapter introduces the meaning of the tasks scheduling algorithms and types of static and dynamic scheduling algorithms in cloud computing environment. This chapter also introduces a comparative study between the static and dynamic task scheduling algorithms in a cloud computing environment such as FCFS, SJF, and MAX-MIN, ACO and GA in terms of TWT, TFT, fairness between tasks, and when becoming suitable to use? Experimentation was executed on CloudSim, which is used for modelling the different tasks scheduling algorithms.

REFERENCES

[1] S. H. Jang, T. Y. Kim, J. K. Kim, and J. S. Lee, "The study of genetic algorithm-based task scheduling for cloud computing," International Journal of Control and Automation, vol. 5, pp. 157-162, 2012.

[2] T. Goyal and A. Agrawal, "Host Scheduling Algorithm Using Genetic Algorithm In Cloud Computing Environment," International Journal of Research in Engineering & Technology (IJRET) Vol, vol. 1, 2013.

[3] B. Furht, "Armando Escalante Handbook of Cloud Computing," ISBN 978-1-4419-6523-3, Springer2010.

[4] F. Etro, "Introducing Cloud Computing," in London Conference on Cloud Computing For the Public Sector, 2010, pp. 01-20.

[5] R. Kaur and S. Kinger, "Enhanced Genetic Algorithm based Task Scheduling in Cloud Computing," International Journal of Computer Applications, vol. 101, 2014.

[6] J. W. Ge and Y. S. Yuan, "Research of cloud computing task scheduling algorithm based on improved genetic algorithm," in Applied Mechanics and Materials, 2013, pp. 2426-2429.

[7] S. Ravichandran and D. E. Naganathan, "Dynamic Scheduling of Data Using Genetic Algorithm in Cloud Computing," International Journal of Computing Algorithm, vol. 2, pp. 127-133, 2013.

[8] V. Vignesh, K. Sendhil Kumar, and N. Jaisankar, "Resource management and scheduling in cloud environment," International Journal of Scientific and Research Publications, vol. 3, p. 1, 2013.

[9] V. V. Kumar and S. Palaniswami, "A Dynamic Resource Allocation Method for Parallel DataProcessing in Cloud Computing," Journal of computer science, vol. 8, p. 780, 2012.

[10] Z. Zheng, R. Wang, H. Zhong, and X. Zhang, "An approach for cloud resource scheduling based on Parallel Genetic Algorithm," in Computer Research and Development (ICCRD), 2011 3rd International Conference on, 2011, pp. 444-447.

[11] K. Thyagarajan, S. Vasu, and S. S. Harsha, "A Model for an Optimal Approach for Job Scheduling in Cloud Computing," in International Journal of Engineering Research and Technology, 2013.

[12] S. Singh and M. Kalra, "Scheduling of Independent Tasks in Cloud Computing Using Modified Genetic Algorithm," in Computational Intelligence and

Communication Networks (CICN), 2014 International Conference on, 2014, pp. 565-569.

[13] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in High Performance Computing & Simulation, 2009. HPCS'09. International Conference on, 2009, pp. 1-11.

[14] J. S. Raj and R. M. Thomas, "Genetic based scheduling in grid systems: A survey," in Computer Communication and Informatics (ICCCI), 2013 International Conference on, 2013, pp. 1-4.

[15] B. Kruekaew and W. Kimpan, "Virtual Machine Scheduling Management on Cloud Computing Using Artificial Bee Colony," in Proceedings of the International MultiConference of Engineers and Computer Scientists, 2014.

[16] M. Mitchell, An introduction to genetic algorithms: MIT press, 1998.

[17] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," arXiv preprint arXiv:0903.2525, 2009.

[18] R. Sahal and F. A. Omara, "Effective virtual machine configuration for cloud environment," in Informatics and Systems (INFOS), 2014 9th International Conference on, 2014, pp. PDC-15-PDC-20.

[19] D. M.Abdelkader, F.Omara," Dynamic task scheduling algorithm with load balancing for heterogeneous computing," system Egyptian Informatics Journal, Vol.13, PP.135–145, 2012.

[20] Zhu W, Lee C (2016) A security protection framework for cloud computing. J Inf Process Syst 12:538–547

[21] Maity S, Park J-H (2016) Powering IoT devices: a novel design and analysis technique. J Converg 7:1–18

[22] Lim J, Jeong YS, Park D-S, Lee H (2016) An efficient distributed mutual exclusion algorithm for intersection traffic control. J Supercomput. doi:10.1007/s11227-016-1799-3.

[24] Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B (Cybern) 26:29–41