# AMERICAN JOURNAL OF SMART TECHNOLOGY AND SOLUTIONS (AJSTS)

VOLUME 2 ISSUE 2 (2023)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Smart Technology and Solutions (AJSTS)

Volume 2 Issue 2, Year 2023
ISSN: 2837-0295 (Online)
DOI: https://doi.org/10.54536/ajsts.v2i2.1804
https://journals.e-palli.com/home/index.php/ajsts

# Genetic Algorithm of Independent Task Meta-Scheduling Centralized in the Cloud Computing

Stéphane Fouakeu Tatieze[1*], Jean Claude Kamgang[2], Marcellin Julius Nkenlifack[3]

**Article Information**

**Received:** June 30, 2023

**Accepted:** July 24, 2023

**Published:** July 30, 2023

**Keywords**

*Cloud Computing, Genetic Algorithm, Meta-Scheduling*

## ABSTRACT

A group of networked, virtualized computers make up the distributed, parallel cloud computing technology. The power for these machines is dynamic, and they are displayed as one or more computing resources. These are compiled based on service level agreements (SLAs) that have been negotiated between the service provider and the customers. Enterprise applications have migrated in large numbers to cloud computing during the past few years. One of the most important challenges of Cloud Computing is the scheduling of tasks; which should satisfy Cloud users in terms of Quality of Service and increase the profit of cloud providers. Bio-inspired algorithms (genetics) represent a heuristic research technique that produces effective solutions. In this article, we propose a genetic meta-scheduling algorithm that optimizes the execution time and makespan of tasks submitted by users. To achieve this, this algorithm is based on the requirements of user requests and the availability of resources (Virtual Machines) of Cloud Computing to obtain a better combination as an optimal solution. This effort makes the meta-scheduling genetic algorithm superior than others in the literature like the Min-Min algorithm and the regular genetic algorithm. Customer satisfaction is higher, and more particularly, the execution time and makespan are better.

## INTRODUCTION

Cloud computing (CC) is a new paradigm for utility virtualized resources, designed for end users in a dynamic computing environment to provide reliable and guaranteed services (Dillon *et al.*, 2010). Cloud Computing has service models and deployment models. As service models, we have Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS). Software as a Service (SaaS) provides users with applications in the form of online services already deployed in the cloud. This layer is managed by the SaaS provider in a way that is transparent to users. Platform as a Service (PaaS) is more oriented to serve application developers. It offers a fully configured and managed plat- form on which the user can develop, test and run their applications. Infrastructure as a Service (IaaS) allows infrastructure resources such as computing capacity, storage, network as utilities. As deployment models, we have private cloud, community cloud, public cloud, hybrid cloud. In the private cloud, all of its resources are made available exclusively to a single company or organization. The private cloud can be managed by the company itself (internal private cloud) or by a third party (external private cloud). In the community cloud, the infrastructure is shared by several independent organizations with com- mon interests. The infrastructure can be managed by the member organizations or by a third party. In the public cloud, the infrastructure is accessible to a wide public and is owned by a service provider. The latter charges users ac- cording to consumption and guarantees the availability of services through SLA contracts and for the hybrid cloud, the infrastructure is a composition of several clouds (private, community or public) (Dillon *et al.*, 2010). Cloud environment allows users to use applications without installation and access their personal files at any computer with Internet access, end users access cloud based applications through a web browser or a light weight desktop (Durga *et al.*, 2016). CC, applies distributed computing techniques to deliver an on-demand access to a shared virtual computing resources (ex. Net- works, Servers, Storage, Applications and Services) over the Internet (Zhang *et al.*, 2010; Singh *et al.*, 2017). Virtualization is an emerging technology for efficient utilization of cloud resources. It is used to split a single physical machine into multiple Virtual Machines (VM) (Malhotra *et al.*, 2010). VM also can provide resource sharing, high utilization of pooled resources, rapid provisioning and workload isolation (Ahmad *et al.*, 2015b). Usually, a Cloud Service Provider (CSP), like Google, presents these facilities using the pay per use model (Arya *et al.*, 2014). By the help of cloud computing technology, users such as the individuals, researchers and large businesses can access their data, applications, on different platforms via the internet without the need for buying costly computing resources. The main goal of Cloud computing is to satisfy cloud users with the agreed QoS and improve profits of cloud providers (Buyya *et al.*, 2009). To provide ensured proficient performance to users, it is necessary that tasks should be mapped efficiently to available resources. Task Scheduling (TS) is one of the core challenges in CC environment. The task Scheduling problem in Cloud, which is known to be NP-hard, is as- signing different tasks to corresponding resource node under the Quality of Services (QoS) constraints (Aarts *et al.*, 2005). TS can be classified into

[1] Department of Electrical Engineering and Industrial Automation, ENSAI, University of Ngaoundéré, Cameroon
[2] Department of Mathematics and Computer, ENSAI, University of Ngaoundéré, Cameroon
[3] Department of Mathematics and Computer, Faculty of Sciences, University of Dschang, Cameroon
[*] Corresponding author's e-mail: fouakeustephane@gmail.com

independent scheduling and de- pendent scheduling. In independent scheduling tasks are independent of each other and can be scheduled in any sequence, however in dependent scheduling, tasks are represented by a Directed Acyclic Graph (DAG) (i.e., workflow scheduling). DAG is a directed graph that comprises group of edges and vertices. Where each vertex signifies the task and every edge signifies the affiliation between two nodes or vertices connected through that edge (Singh *et al.*, 2015). TS can also be classified into static and dynamic task scheduling. In static scheduling, all tasks or VMs are known a priori to scheduling. These tasks are independent of the virtual machine's states and their availability. So, it imposes less runtime overhead. On the other hand, in dynamic scheduling, the information about the tasks is unknown in advance. So, the execution time of task may not be known and the information about VMs is not obtained until it comes into the scheduling stage (Nagadevi *et al.*, 2013). TS is an optimization problem belonging to the class of NP-hard problems. Some traditional task scheduling algorithms have been applied in heterogeneous computing environments such as Min-Min (He *et al.*, 2003), Max-Min (Mao *et al.*, 2014), etc.

So in cloud computing, there are various type of meta heuristic algorithms for scheduling problem in cloud computing such as ant colony optimization (ACO), particle swarm optimization (PSO) genetic algorithm (GA), etc, (Xu *et al.*, 2009) can be applied to achieve near optimal solution. Genetic algorithm, based on natural selection and inheritance theory, has been widely and successfully applied in scheduling problems. In this paper, genetic algorithm is implemented using CloudSim simulator and compared to the traditional heuristic methods to solve the independent static TS problem in CC environment. In static environment, the specifications of the VMs are fixed. When users submit jobs to the resources for execution, meta-scheduler acquire information about resources from CIS (Cloud Information Service) and then divide the job into various tasks or subtasks if needed. Then map the to best resources distributed geographically same according to user's requirements and availability of resources. CIS are responsible for providing information about status of available resources which helps the meta-scheduler for scheduling, monitoring and further communication if required. After execution of all tasks, result is combined and sends back to user via meta-scheduler. The main features of competent TS in this paper are minimizing deadline, and budget. The remainder of the paper is organized as follows. Section II gives an overview of related work on TS in cloud computing. Section III presents the task scheduling problem. Section IV presents the genetic algorithm. In Section V we have the system model. Experimental results and discussions are given in Section VI. Section VII concludes this paper.

**Related Work**

Tasks scheduling is a hot and major research area in the distributed environment like cloud computing. It is a challenging issue in which a lot of research works have been carried out. Many meta-heuristic techniques like Genetic Algorithm (GA) were proposed to solve the Tasks scheduling problems using various strategies: (Jang *et al.*, 2012) proposed task scheduling model where the task scheduler calls the GA scheduling function to make task schedules based on information of tasks and virtual machines. The GA scheduling function creates a population, a set of task schedules, and evaluates the population by using the fitness function considering user satisfaction and virtual machine availability. The function iterates reproducing populations to output the best task schedule. Experimental results show effectiveness and efficiency of the genetic algorithm-based task scheduling model in comparison with existing task scheduling models, which are the round-robin task scheduling model, the load index-based task scheduling model, and the ABC based task scheduling model. (Kaur *et al.*, 2012) have developed a task scheduling algorithm for cloud computing environment. The author used Shortest Cloudlet to Fastest Processor (SCFP) and Longest Cloudlet to Fastest Processor (LCFA) algorithm to initialize the population of GA. Their algorithm takes variable power processors and variable length tasks to represent a real-time scenario but considers single user job. In this research they have proposed a modified genetic algorithm for single user jobs in which the fitness is developed to encourage the formation of solutions to achieve the time minimization and compared it with existing heuristics. Experimental results show that, under the heavy loads, the proposed algorithm exhibits a good performance. In (Dasgupta *et al.*, 2013) proposed GA as a load balancing technique for cloud computing to find a global optimum processor for job in a cloud. They have presented an approach that handles the load on processors as well as reduces the makespan, but takes equal priority tasks. Analysis of the results, indicates that the proposed strategy for load balancing not only outperforms a few existing techniques but also guarantees the QoS requirement of customer job. In (Kaur *et al.*, 2014) genetic algorithm is enhanced using new fitness function based on mean and grand mean values. This optimization can be implemented on both ends, for job scheduling and resource scheduling. It reduces the execution time of all the tasks but considered limited number of tasks. (Atul *et al.*, 2015) propose a multi-objective task scheduling algorithm for mapping tasks to a Vms in or-der to improve the throughput of the datacenter and reduce the cost without violating the SLA (Service Level Agreement) for an application in cloud SaaS environment. The proposed algorithm provides an optimal scheduling method. Most of the algorithms schedule tasks based on single criteria (i.e execution time). But in cloud environment it is required to consider various criteria like execution time, cost, bandwidth of user etc. This algorithm is simulated and the result shows better performance and improved throughput. (Juntao *et al.*, 2016) proposes a novel dynamic task scheduling algorithm based on improved genetic algorithm (IGATS). This paper introduces the concept of load priority. First, select average queue-run

length to as a high-priority load parameter, which reflects the average number of processes running in the queue within the specified time interval; Second, select the CPU utilization and memory utilization as a priority load parameters, which reflects the currently running task size of system resources; experimental results demonstrate that the pro- posed algorithm can effectively improve the throughput of cloud computing systems, and can significantly reduce the execution time of task scheduling. (Amjad *et al.*, 2017) presented an efficient greedy algorithm and a genetic algorithm with adaptive selection of crossover and mutation from a pool of crossover and mutation types to allocate and schedule real-time tasks with precedence constraint on heterogamous virtual machines. The selection of crossover and mutation is based on the previous performance of the operators. The adaptive GA uses population diversity to determine the fitness of each type of crossover while the fitness of mutation is determined in terms of its ability to find a better quality solution (i.e., intensification). (Rasha *et al.*, 2018) proposed HTSCC ( Hybrid Task Scheduling in Cloud Computing) algorithm to improves the local search by using the GA mutation operator and expected to work with the different size of tasks. The proposed HTSCC algorithm makes use of the advantages of the GA and PSO algorithms in order to maximize resource utilization and minimize makespan. These features of the proposed algorithm re- duce the makespan and increase the resources utilization. To optimize large-scale task scheduling problem in Cloud environment with less makespan and computation time, (Kairong *et al.*, 2018) proposed an adaptive incremental Genetic algorithm. Their method based on genetic algorithm which has adaptive probability of mutation rate and crossover rate can provide feasible solutions for the allocation of large numbers of tasks with less computation time. The simulation results show that algorithm outperforms the greedy algorithm and non-adaptive genetic algorithm in terms of solution quality. (Nagwan *et al.*, 2019) have implemented TS using two metaheuristic algorithms (PSO, GA) and compared their performance with two traditional techniques (FCFS, SJF). They generate the chromosomes randomly with a list of tasks and a list of VMs to form the initial population in GA. After, evaluate the performance of each chromo- some using fitness function. It is calculated using a set of metrics such as makespan, flow time, response time, resource utilization, throughput time and degree of imbalance. Based on fitness value retrieved from each metric, chromosomes are selected and then are feed to a crossover and mutation operations. After, they update the population and decode the procuration chromosome (feasible solution) then, the best chromosome is the final solution for tasks allocation on VMS. The algorithms have been implemented as part of the cloud broker in symmetric and asymmetric environment. GA algorithm only fulfilled the optimal degree of imbalance in symmetric environment with real workload traces. Otherwise, it gave sufficient performance in obtaining the optimal response time in asymmetric environment in both of synthetic traces and real workload traces.

**Task Scheduling Problem**

Cloud consists of a number of resources that are different with one other via some means and cost of performing tasks in cloud using resources of cloud is different so scheduling of tasks in cloud is different from the traditional methods of scheduling and so scheduling of tasks in cloud need better attention to be paid because services of cloud depends on them. Task scheduling plays a key role to improve flexibility and reliability of systems in cloud. The main reason behind scheduling tasks to the resources in accordance with the given time bound, which involves finding out a complete and best sequence in which various tasks can be executed to give the best and satisfactory result to the user. In cloud computing, resources in any form i.e. cups, firewall, network are always dynamically allocated according to the sequence and requirements of the task, subtasks. So, this leads task scheduling in cloud to be a dynamic problem means no earlier defined sequence may be useful during processing of task. The reason behind the scheduling to be dynamic is that because flow of task is uncertain, execution paths are also uncertain and at the same time resources available are also uncertain because there is a number of tasks are present that are sharing them simultaneously at the same time (Singh *et al.*, 2014). The scheduling of tasks in cloud means choose the best suitable resource available for execution of tasks or to allocate computer machines to tasks in such a manner that the completion time is minimized as possible. In scheduling algorithms list of tasks is created by giving priority to each and every tasks where setting of priority to different tasks can be based on various parameters. Tasks are then chooses according to their priorities and assigned to available processors and computer ma- chines which satisfy a predefined objective function (Radulescu *et al.*, 2000). Meta-scheduling is defined (Christodoulopoulos *et al.*, 2009) as a software technique for optimizing workloads of grid clusters, by choosing and combining the resources of their different managers into a single aggregated view, so that jobs can be directed batch users to the best execution locations, in a manner transparent to them. For example, meta-scheduling is about assigning user jobs to nodes in a grid, such as clusters, which in turn have their own local schedulers. Cloud computing uses virtualization technique for mapping the resources of cloud to the virtual machine layer, implement the user's task, so the task scheduling of cloud computing environment achieve at the applications layer and the virtual layer of resources. Scheduling is nothing but the mapping of tasks and resources in accordance with some certain principles for achieving the desired goal. Cloud computing paradigm simplifies the mapping of tasks to resources; the required resources together form to be virtual machines (VMs), the process of search the desired resource package is

same as the process of searching the various VMs.

**Genetic Algorithm**

Genetic Algorithm (GA) is based on the biological concept of generating the population. GA is considered a rapidly growing area of Artificial Intelligence (Jang *et al.*, 2012). 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 (Buyya *et al.*, 2009) (see Figure. 1).

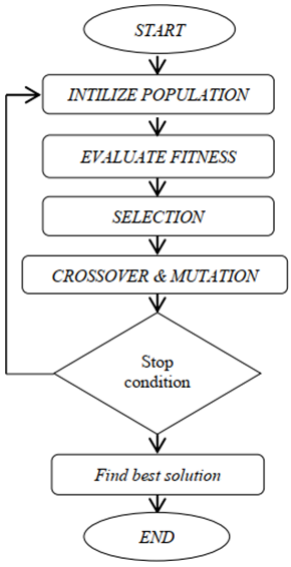



**Figure 1:** Genetic Algorithm (Raj *et al.*, 2013)

The main principles of the GA are described as follows (Jang *et al.*, 2012):

**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.

**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.

**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, selection based on rank and elitist selection.

**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. There are some crossover strategies such as single-point crossover, two-point crossover, uniform crossover and so on. An example of single-point crossover is shown in this figure:

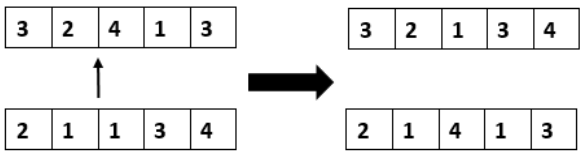



**Figure 2:** Single-point crossover [23]

**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.

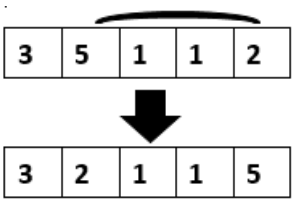



**Figure 3:** Mutation operator [23].

**Keep Best Solution**

There is a solution that might satisfy good fitness function, but it is not selected during the crossover process.

**Overview of Proposed System**

**System Model**

Our work based on the IaaS (infrastructure as a service) model of the system is illustrated in Figure 2: customers submit requests through an interface via their endpoints; cloud service providers provide customers with a virtual machine as a unit of required computing resources. Cloud customers rent these resources and pay the provider based on the amount of resources occupied and the length of

time they occupy them. The IaaS provides the hardware equipment and other basic resources as a service to users or customers in the cloud. The biggest advantage of IaaS is that it allows users to dynamically apply or release nodes, charging based on usage. The number of servers operating in IaaS reaches hundreds of thousands, so the resources that users can request are almost unlimited. At the same time, the IaaS is shared by the public, which allows for greater efficiency in the use of resources.

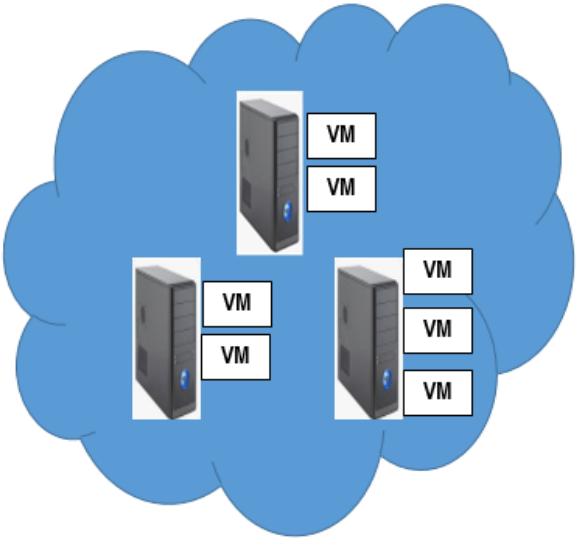



**Figure 4:** The IAAS system model (Song *et al.*, 2014)

Each server is symbolized by a hypervisor in the Cloud. This hypervisor controls and keeps information about the execution of several virtual machines installed on it. Cloud users frequently use these virtual machines to perform their tasks. While performing these tasks, free time slots can be observed on some virtual machines that can be exploited for performing other tasks of cloud clients. The meta-scheduler will have this responsibility to collect the tasks from the cloud clients, redirect them to the best virtualized resource sites (virtual machines) with free time slots through a hypervisor.

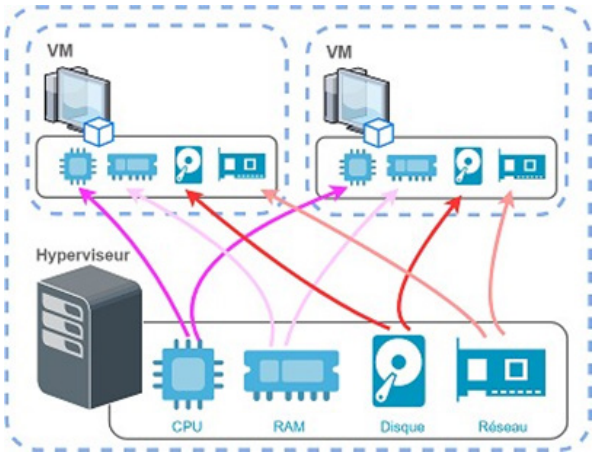



**Figure 5:** Server machine(Song *et al.*, 2014)

### Detailed Architecture

The detailed architecture consists of several entities that describe the different components of the overall system.

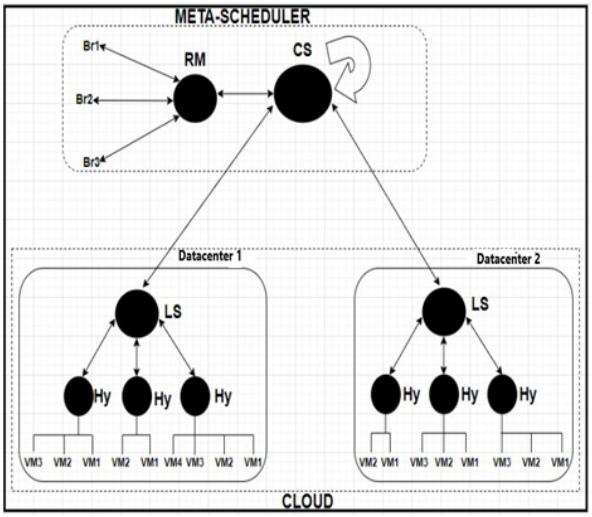



**Figure 6:** Architecture

This architecture works as follows:

- Upon receipt of service requests submitted by the broker, the RM (Request Manager) evaluates them and ranks them in descending order of execution priority in the list of requests and makes them available to the Central Scheduler (CS).
- The Central Scheduler (CS) sends a request for availability of resources or virtual machines (VMs) to the Local Scheduler (LS) in the Cloud.
- The Central Scheduler (CS) will have on one side a list of tasks grouped by order of execution priority and on the other side a list of resources (virtual machines) grouped by order of decreasing execution power.
- The Central Scheduler (CS) then performs the genetic meta-ordering algorithm procedure on these two lists to find an optimal solution.
- The Central Scheduler (CS) transmits the tasks to be executed to the best execution resources (virtual machines) via the appropriate Local Scheduler (LS).
- At the end of execution, the Local Scheduler (LS) receives the processing information via the hypervisors and transmits it to the Central Scheduler (CS).
- The various users will be notified of the end of execution.
- The Central Scheduler (CS) repeats the same process for new tasks that have arrived and tasks that have not been completed.

### Genetic Algorithm of Meta-Scheduling

When the meta-scheduler has at its level a list of user tasks and a list of available virtual machines that can perform the tasks, it determines the priorities of these tasks and then arranges in ascending or descending order. Virtual machines, too, are classified by the power of execution of the tasks in the delay. The initialization of the genetic algorithm can be done under priority constraint, type of virtual machine by task and feasibility (deadline).

### Initial Population

Initial population (Tasks reach to the VM) is generated

randomly. In the task scheduling problem, a chromosome represents a likely solution. This chromosome is encoded by a real number. The length of a chromosome is equal to the number of tasks. The value of the gene represents to which virtual machine the task is allocated. A task is characterized by the waiting time (ati), size or length (li), deadline (di). The priority tasks that have arrived for processing are arranged in a queue through the priority function given by the following formula:

(1/(di-ati-li)*1/(ati+1)) (1)

The available virtual machines are arranged in a queue in a decreasing manner in terms of power. Our coding is as follows:

Each Ti task is assigned to an available VMj virtual machine of the IaaS cloud.

Suppose we have n number of tasks (Tn) and 3 virtual machines available.

An example of a solution is shown in the figure.

| Task | T1 | T2 | T3 | T4 | T5 | ------ | Tn |
|---|---|---|---|---|---|---|---|
| Id VM | VM3 | VM2 | VM1 | VM1 | VM2 | ------ | VM3 |

**Figure 7:** Representation of a solution

A simplified representation of a solution is given in the following figure:

| VM3 | VM2 | VM1 | VM1 | VM2 | ------ | VM3 |
|---|---|---|---|---|---|---|

**Figure 8:** Simplified representation of a solution

This literally translates into: task T1 is assigned to virtual machine 3, T2 is assigned to virtual machine 2, etc.

We form as many chromosomes considered as probable solutions to the problem.

A valid solution must meet the following conditions:

-Each mapping between the task and a virtual machine in the solution must respect: Tij<di

A virtual machine can run only one task at a time, but can manage several tasks over time. Therefore the tasks must be scheduled over an available time frame.

Each task is executed only once on a single virtual machine.

**Evaluation Function**

The quality of a candidate solution is evaluated using the fitness function. This is an index to select the best solution. It will be called makespan. A solution will be considered if and only if the deadline of all tasks assigned to the virtual machines making up the solution is met. When a ti task is assigned to a VMj virtual machine, the execution time of the VMj depends on the length of the task (L) and the frequency (F) of the virtual machine. The execution time of Li on VMj is given by Tij:

Tij=Li/Fj (2)

where i ∈ 1, 2, 3, ....., m and j ∈ 1, 2, 3, , n

It should be noted that the global execution time of a solution is equal to the maximum value of the execution times of all the virtual machines, because the execution of the tasks are done in parallel on VMs. In order to minimize the makespan, a so-called objective function can be presented as follows:

$$\text{Min}(\text{Max}(Tj))=\text{Min}(\text{Max}(\sum_{(i=1)}^{m} Tij)) \quad (3)$$

Tij represents corresponding execution time.

**Selection**

After having evaluated each individual or chromosome of the population, it will be necessary to select those which will undergo the operators of crossing and mutation. Here, we will use the elistic selection. It consists in choosing or retaining the best individuals for the next generation. We will select the best chromosomes with small Makespan.

**Crossover**

In order to produce new chromosomes from the parent chromosomes, we will use the crossover operator and more precisely the single-point crossover.

**Mutation**

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. All the new individuals obtained after the crossover must undergo the operation of mutation. Each genome of the chromosome must be randomly swapped respecting the virtual machine numbers. If we have for example two virtual machines, this mutation operator takes the chosen genome and inverts the bits (i.e. if the genome bit is 1, it is changed to 0 and vice versa).

**Evaluation Function**

After the mutation process, we will reevaluate the new population obtained. It is enough to use the objective function to evaluate the new individuals in order to choose the best ones.

**Termination Condition**

Genetic Algorithm gets terminated after user specified number of generations. We generated 20 evolutions of genetic algorithm to get the better results.

The pseudo code of our genetic algorithm is shown in Algorithm 1.

**Simulation Results and Analysis**

In order to obtain the results of the proposed algorithm, a simulation was performed using the Python3 simulator under Windows 10 OS with Core i3 3.90GHz processor, 500GB hard drive and hard disk and 4GB of RAM. Python is a powerful and easy to learn programming language. It has high-level data structures and allows for a simple but effective approach to object-oriented programming. Because of its elegant syntax, dynamic typing, and interpretability, Python is an ideal language for scripting and rapid application development in many areas and on most platforms.

In this work, we perform the meta-scheduling of independent tasks and the best scheduling algorithm will be the one that optimizes some resource utilization loads. The goal of this algorithm is to satisfy the user in terms

of quality of service such as minimizing the execution time of the tasks and also the makespan. The makespan is the total time needed to process a set of tasks for its complete execution. We will consider in our simulation, a set of user tasks (T1, T2, T3,....,Tn) that will run on the virtual machines (VM1, VM2, VM3,. ,VMn) available in the cloud. The frequency of the virtual machines is expressed in millions of instructions per second (MIPS) and the length of the tasks is expressed in number of instructions. We will calculate the execution time of each task in a list and the makespan. Our simulations will be based on:
table of the parameters of the proposed genetic algorithm in table 1

- variation of the number and frequency of virtual machines
- random variation of task lengths

**Algorithm 1: Algorithm of meta-scheduling**
Input:PQ[], CVM[],Population:P, max iteration
Result: The global best solution
while iteration < max generation do for (i=1; i<=n; i++) do
PQ[]=Task selected and sorted by calling function priorty;
CVM[]= Type of VM selected and sorted by decreasing order of their capacity or frequency;
if length(CVM)==0 then
the tasks cannot be performed;
else
individuali[]=mapping(PQ[], CVM[]);
end
end
for (j=0;j<length(individuali);j++) do if deadline<waiting time then
The individual is considered;
else
not considers the individual;
end end
for (each individual in P) do
Makespan=Fitness function();//Evaluation end
for (i=1; i<=P;P++) do
Sort according fitness; //Selection Return P';
end
for (each two of individual in P') do
Perform crossover;//Crossover Return P'';
end
for (each individual in P'') do
Perform mutation;//Mutation Return P''';
end
for (each individual in P''') do
Perform Fitness function;//Re-Evaluation Return the best fitness value;
end
Update the global best solution; iteration++;
End

**Table 1:** GA Parameters

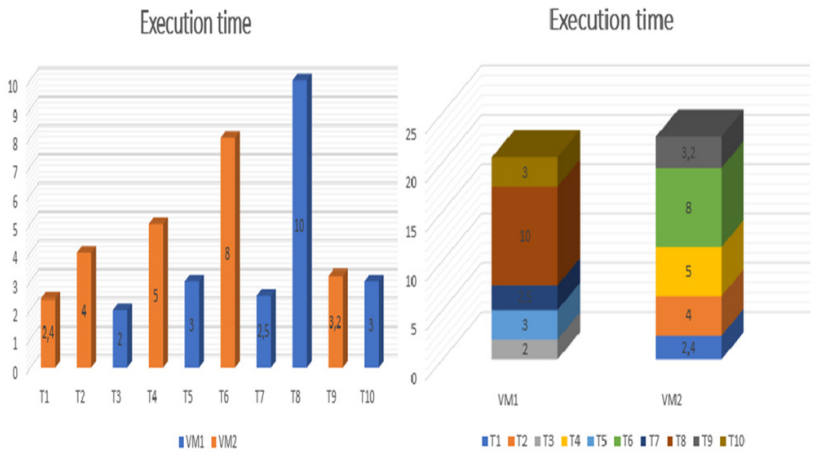
| Parameter | Value |
|---|---|
| Number of Tasks | 10 |
| Number of VM | 2 and 4 |
| Number of Iterations | 20 |
| Selection Type | elistic |
| Crossover Type | Two-Point Crossover |
| Mutation Type | random permutation |
| Termination Condition | Number of Iterations |

**Table 2:** VM types

| No | Instance Type | CU |
|---|---|---|
| 1 | c3:large | 7 |
| 2 | c3:large | 14 |
| 3 | c3:2xlarge | 28 |
| 4 | c3:3xlarge | 55 |

The virtual machine instances(types) used are the Amazon Elastic Compute one from table 2 [31] and the task lengths are randomly generated between 0 and 1000 instructions.
task = ["Taskname", waiting time, length, deadline]
vm = ["Vmname", Freq], Freq =frequency of the processor on which a virtual machine runs.

**10 Tasks with 2 VMs**
tasks = [["t1", 0, 3, 4], ["t2", 0, 5, 6], ["t3", 1, 2, 6], ["t4", 2, 6.25, 11], ["t5", 3, 3, 9], ["t6", 4, 10, 19], ["t7", 5, 2.5, 13], ["t8", 6, 10, 21], ["t9", 7, 4, 23], ["t10", 7, 3, 29]]
vms =[["vm1", 1], ["vm2", 1.25]]



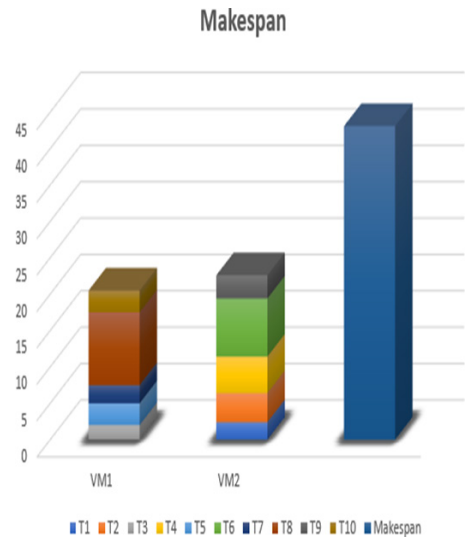
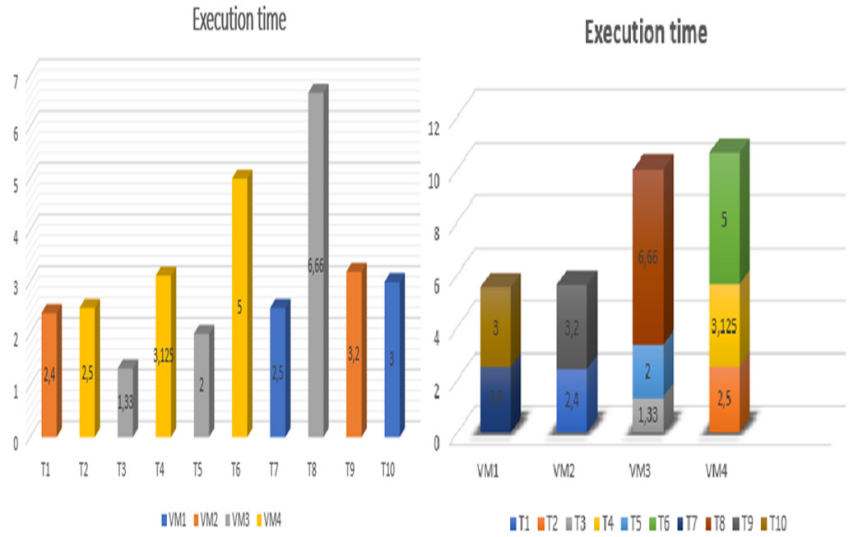



**Figure 9:** 10 Tasks with 2 VMs

**10 Tasks with 4 VMs**

tasks = [[“t1”, 0, 3, 4], [“t2”, 0, 5, 6], [“t3”, 1, 2, 6], [“t4”, 2, 6.25, 11], [“t5”, 3, 3, 9], [“t6”, 4, 10, 19], [“t7”, 5, 2.5, 13], [“t8”, 6, 10, 21], [“t9”, 7, 4, 23], [“t10”, 7, 3, 29]] vms = [[“vm1”, 1], [“vm2”, 1.25], [“vm3”, 1.5], [“vm4”, 2]]



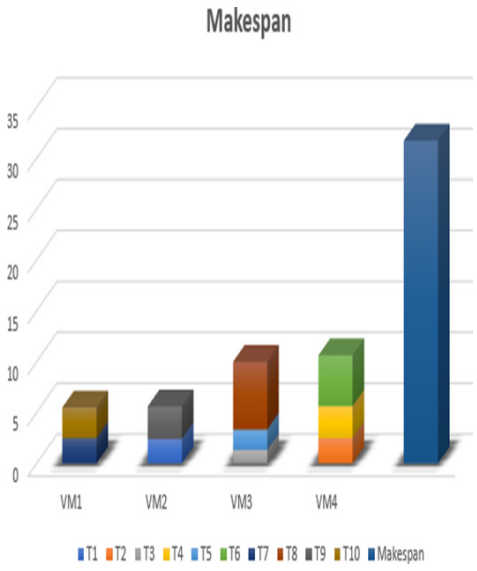




**Figure 10:** 10 Tasks with 4 VMs

**10 Taches avec 4 VMs de types Amazon Elastic Compute**
tasks = [[“t1”, 0, 814, 850], [“t2”, 0, 447, 500], [“t3”, 1, 610, 650], [“t4”, 2, 319, 370], [“t5”, 3, 876, 900], [“t6”, 4, 692, 730], [“t7”, 5, 663, 700], [“t8”, 6, 631, 690], [“t9”, 7, 626, 680], [“t10”, 7, 655, 700]] vms = [[“vm1”, 7], [“vm2”, 14], [“vm3”, 28], [“vm4”, 55]]

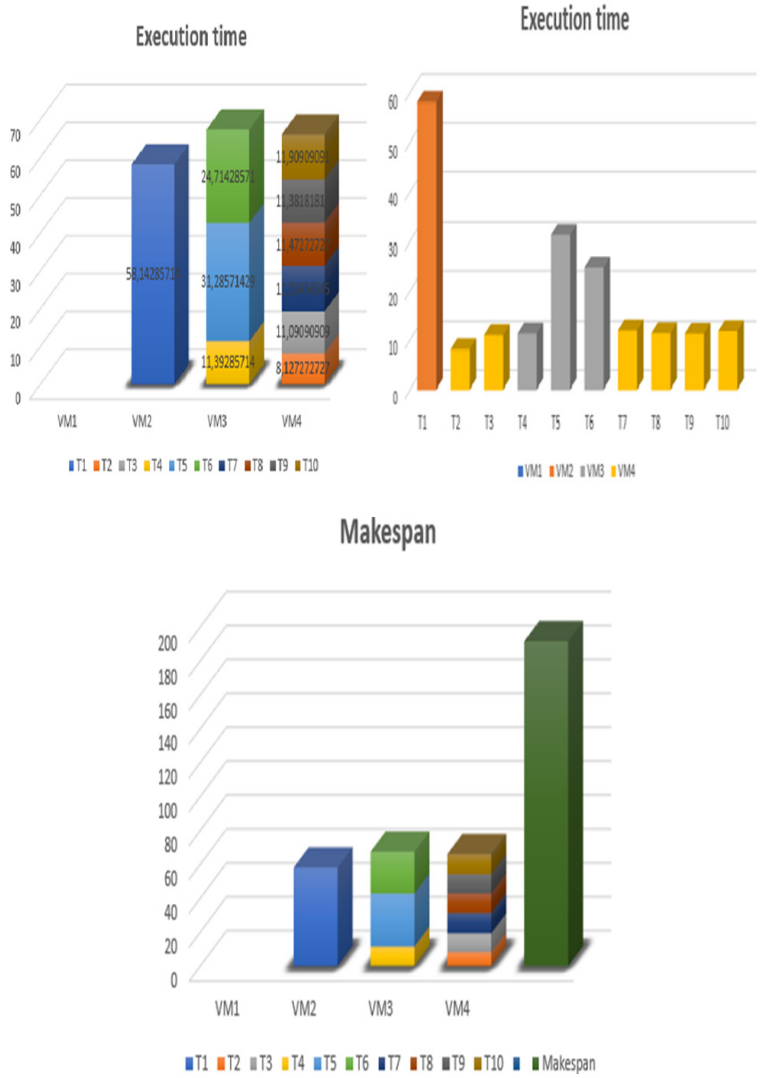



**Figure 11: 10** Taches avec 4 VMs de types Amazon Elastic Compute

In (Shaminder *et al.*, 2012), the authors present the standard genetic algorithm(SGA) and the modified genetic algorithm(MGA). The results that the SGA and MGA present in relation to the makespan and the number of cloudlets are visible on figure 12.

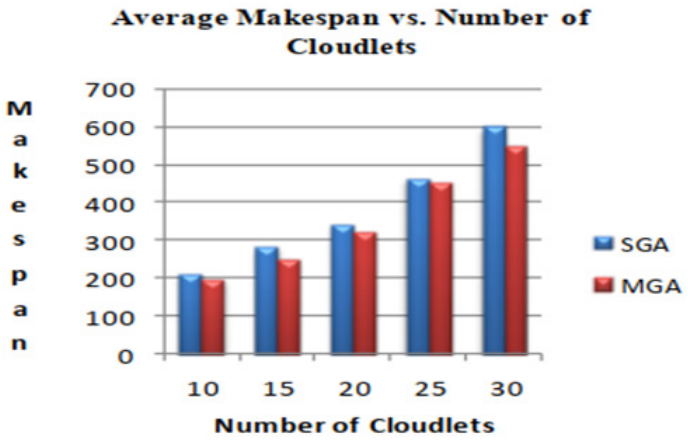



**Figure 12:** Shows Average Makespan vs. Number of Cloudlets (Shaminder *et al.*, 2012)

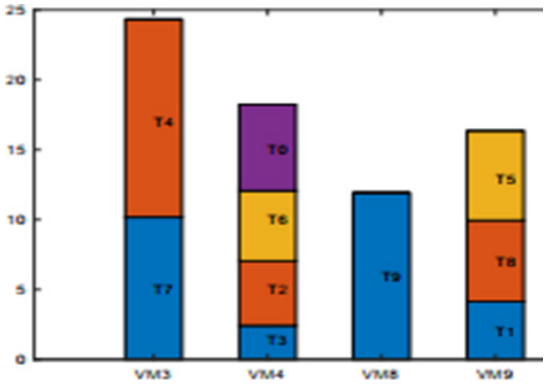



**Figure 13:** Min-Min (He *et al.*, 2003)

In (He *et al.*, 2003), figure 13, we present the results on the simulation of the Min-min algorithm. Here we have 04 types of virtual machines and 10 tasks with length between 0 and 1000 as in our case.

The figures above make the comparison between the execution time and the makespan. Compared to the presented values, we can say that our proposed meta-scheduling genetic algorithm performs well and is better than other algorithms in the literature namely, the standard genetic algorithm and the Min-min algorithm.

## CONCLUSION

The work in this paper presents a genetic algorithm for centralized meta-scheduling in cloud computing. The proposed algorithm optimizes the use of resources taking into account the minimization of the processing time of each user task and the makespan. This algorithm is based on bio-inspired algorithms, namely the genetic algorithm. The objective of this genetic algorithm of meta-scheduling is to achieve a quality of service to the user. Experimental results show that our genetic algorithm performs well compared to others in the literature.

Later, we will implement a genetic algorithm for meta-scheduling scientific workflows in a decentralized and multi-level context in cloud computing.

## REFERENCES


Aarts E, J. Korst, W. Michiels. (2005). Simulated annealing. *In Search Methodologies; Springer: Boston*, MA, USA, 187-210.

Ahmad, R.W., A. Gani, S.H.A. Hamid, M. Shiraz and F. Xia *et al.* (2015b). Virtual machine migration in cloud data centers: A review, taxonomy and open research issues. *J. Supercomput., 71,* 2473-2515.

Amjad Mahmood, Salman A. Khan 2, Rashed A. Bahlool (2017). Hard Real-Time Task Scheduling in Cloud Computing Using an Adaptive Genetic Algorithm. *Journal Computers,* 1-21.

Arya, L.K. and A. Verma. (2014). Workflow scheduling algorithms in cloud environment-A survey. *Proceedings of the Recent Advances in Engineering and Computational Sciences, Mar. 6-8, IEEE Xplore Press,* Chandigarh, India, 1-4.

Atul L, Dharmendra Kumar Y (2015). Multi- Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization. *Procedia Computer Science,* 107-113.

Buyya R, C. S. Yeo, S. Venugopal, J. Broberg and Brandic. (2009). Cloud computing and emerging IT platforms: vision, hype, and reality for deliver- ing computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599-616.

Buyya R, R. Ranjan, and R. N. Calheiros. (2009). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. *in High Performance Com- puting and Simulation, International Conference,* pp. 1- 11.

Christodoulopoulos, Sourlas, Mpakolas, and Varvarigos. (2009). A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in grid networks. *ELSEVIER B.V,* 0140-3664.

Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing. *Procedia Technology, 10*, 340-347.

Dillon T, C. Wu and E. Chang. (2010). Cloud Computing: Issues and Challenges. Proceedings of the IEEE 24th *International Conference Advanced Information Net- working and Applications. Perth, 20,* 31, 27–33.

Durga L, N.Srinivasu. (2016). A dynamic approach to task scheduling in cloud computing using genetic algorithm. *Journal of theoretical and applied information technology, 85*(2), 124-135.

He, X.; Sun, X.; Von Laszewski, G. (2003). QoS guided min-min heuristic for grid task scheduling. *J. Comput. Sci.Tech. 18,* 442-451.

He,Sun, Von Laszewski (2003). QoS guided min-min heuristic for grid task scheduling. *J. Comput. Sci. Tech. 18,* 442–451.

Jang, S. H., Kim, T. Y., Kim, J. K., Lee, J. S. (2012). The study of genetic algorithm-based task scheduling for cloud computing. *International Journal of Control and Automation, 5*(4), 157-162.

Juntao Ma, Weitao Li, Tian Fu, Lili Yan and Guojie Hu (2016). A Novel Dynamic Task Scheduling Algorithm Based on Improved Genetic Algorithm in Cloud Computing. *Wireless Communications, Network- ing and Applications,* 829-835.

Kairong Duan, Simon Fong, Shirley W. I. Siu 1, Wei Song and Steven Sheng-Uei Guan (2018). Adaptive Incremental Genetic Algorithm for Task Scheduling in Cloud Environments. *Journal Symmetry,* 1-13.

Kaur S., Verma A. (2012). An efficient approach to genetic algorithm for task scheduling in cloud computing environment. *International Journal of Information Technology and Computer Science (IJITCS), 4*(10), 74.

Kaur, R., Kinger, S. (2014). Enhanced genetic algorithm based task scheduling in cloud computing. *International Journal of Computer Applications, 101*(14).

Malhotra, L., D. Agarwal and A. Jaiswal. (2010). Virtualization in cloud computing. *J. Inform. Tech. Softw, Eng., 4,* 136-136.

Mao, Y.; Chen, X.; Li, X. (2014). Max-min task scheduling algorithm for load balance in cloud computing. In *Proceedings of the International Conference on Computer Science and Information Technology, Barcelona, Spain, Springer: New Delhi, India,* 457-465.

Nagadevi, S., K. Satyapriya and D. Malathy. (2013). A survey on economic cloud schedulers for optimized task scheduling. *Int. J. Adv.Eng. Tech., 4*, 58-62.

Nagwan M. Abdel Samee, Sara Sayed Ahmed and Rania Ahmed Abdel Azeem Abul Seoud (2019). Meta- heuristic Algorithms for Independent Task Scheduling in Symmetric and Asymmetric Cloud Computing Environment. *Journal of Computer Science,* 594-611.

Previous Generation Instances. Available online: https://aws.amazon.com/ec2/previous-generation/?nc1= hls. (accessed on 1 April 2018)

Radulescu A, A. Gemund, (2000). Fast and effective task scheduling in heterogeneous systems. *Proceedings of the 9th heterogeneous computing workshop,* pp. 229-238.

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

Rasha A. Al-Arasi , Anwar Saif (2018). HTSCC: A Hybrid Task Scheduling Algorithm in Cloud Computing Environment. *Journal: International Journal Of Computers and Technology, 17*.

Shaminder K, Amandeep Verma (2012). An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment *I.J. In- formation Technology and Computer Science , 10,* 74-79.

Singh R, P. Sanchita , A. Kumar (2014). Task Scheduling in Cloud Computing: Review. *International Jour- nal of Computer Science and Information Technologies, 5*, 7940-7944.

Singh, K., M. Alam and S.K. Sharma, (2015). A survey of static scheduling algorithm for distributed computing system. *Int. J. Comput. Applic., 129*, 25-30.

Singh, P., M. Dutta and N. Aggarwal. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowl. Inform, Syst. 52*, 1-51.

Song, W., Xiao, Z., Chen, Q. and Luo, H. (2014). Adaptive resource provisioning for the cloud using online bin packing. *Computers, IEEE Transactions on, 63*(11) pp.2647-2660.

Xu M., Cui L.,Wang H.,Bi Y. (2009). A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. Parallel and Distributed *Processing with Applications, 2009 IEEE International Symposium on. IEEE, 2009*.

Zhang Q, L. Cheng and R. Boutaba. (2010). Cloud Computing: State-of-the-Art and Research Challenges. *Journal of Internet Services and Applications, 1*(1), 7-18.