Communication Papers of the Federated Conference on Computer Science and Information Systems pp. 189–195
DOI: 10.15439/2018F180
ISSN 2300-5963 ACSIS, Vol. 17

A new task scheduling approach based on Spacing Multi-Objective Genetic algorithm in cloud

Ali Belgacem
DCS Laboratory
EMP
bordj el bahri, Alger
al.BLgacem@gmail.com

Kadda Beghdad-Bey
DCS Laboratory
EMP
bordj el bahri, Alger
k.beghdadbey@gmail.com

Hassina Nacer
MOVEP Laboratory
USTHB
bab ezzouar, Alger
sino_nacer@yahoo.fr

***Abstract*—The dazzling progress in information and communication technologies, contributed significantly to the emergence of cloud computing paradigm, where it promotes prosperity in all fields of human activity, especially in business. Furthermore, manage the resources and use in ways that sharing with large number of users, consider as one of the challenges facing cloud computing environment today. Because cloud processes a huge tasks, which require the employment of scheduling techniques to handle and monitor the resources in an optimal, flexible and dynamic manner. In this paper, we review a new approach called Spacing-MOGA based on spacing distance to rank no-dominate solutions. It aims mainly to minimize both the makespan and cost of execution tasks on virtual machines (VMs). As well, we study its impact on the availability of resources. Experimental results show that S-MOGA is better than Max-min, PSO and MOGA methods, especially as it minimizes the number of active VMs.**

***Index Terms*—Cloud computing. Resource allocation. Scheduling. Multi-Objective genetic algorithm. Spacing distance**

I. Introduction

The emergence of Cloud computing is considered as a critical turning point in the world of computer, it made the computing power rentable. This announced the beginning of the fifth generation of computing after mainframe, personal computer, web and grid computing. In recent years, cloud has become very popular in different fields, especially for companies to increase economic efficiency and competitiveness. To meet every changing business, the companies need to invest time and budget to up their IT (Information Technology) infrastructure such as hardware, software and services. However, with on-premises IT infrastructure the scaling process is slow and the company is frequently unable to achieve efficient utilization of resources.

That is why, cloud computing is a paradigm shift that provide computing over Internet with an outstanding performance. It consists of set optimized virtual datacenters that provide various software, information and services (servers, storage, databases, networking, software, analytics and more). For use resources needed, companies can simply connected to cloud and use available resources on pay per use basis. This helps companies avoid capital expenditure on additional on-premises infrastructure resources and scale up or scale down according to business requirement.

Cloud computing environment is characterized by four types of accessing (public, private, hybrid and community), and offers three types of services (Software as a service (Saas), Platform as a service (Paas) and Infrastructure as a service (IaaS)). Besides, the virtualization technology is used to allocate the data center resources dynamically according to the application demands. Furthermore, live migration technology make it possible to assign each virtual machine to the physical machines while tasks are executing, which allow efficient utilization of resources. Other related technology that characterize cloud computing environment is the VM consolidation technique, it allows function many VMs on the same server in order to increase the number of unused servers.

On the other hand, resources are an entities where tasks are allocated. Each resource has its own characteristics (computing power (CPU), memory size, etc.). In addition, there are different types of resources: storage resources, power resources, networking resources and compute resources. Since, the scheduling mechanism is an important issue that improves the use of resources and also makes a better performance of this computing environment, many approaches are used to find optimal solutions and to achieve the preferable results.

Most of the studies about the techniques used in the area of task scheduling in cloud, focused on the application of heuristic and meta-heuristic mechanisms [1]. They are a nature inspired algorithm based on the biological or physique phenomena. For example, the work [8] discussed the techniques used for task scheduling founded on Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Genetic Algorithm (GA), and two novels League Championship Algorithm (LCA) and BAT algorithm. In addition to this, the authors in [9] illustrated an analysis about the workflow scheduling approaches based Simulated Annealing (SA), and Cat Swarm Optimization (CSO).

The remainder of the paper is organized as follows: section II illustrates the existing approaches of task scheduling problems in cloud. The description of proposed algorithm is given in section III. Section IV presents the formulation of the studied problem. The proposed algorithm steps are explained in section V. The section VI describes the simulation methodology used to evaluate our approach. Finally section VII presents the conclusion of the paper.

II. Related Works

There are several approaches applied to examine the task scheduling in cloud, with a view to solve various resources allocation problems [1]. Bey, K.B., et al. presented a new scheduling strategy based on load balancing (LBE) approach for an independent tasks which gave a good results in terms of execution time, makespan and resource utilization [2]. The reference [4] introduced a pareto-based multi-objective workflow scheduling algorithm for allocating different on-demand instances with optimal makespan and various prices. But, authors didn't consider the security issues, reliability of spot instances or energy consumption of the system. Correspondingly, the paper [7] aimed to minimize the makespan and ensure a better load balancing of system. The proposed approach assigns the tasks without deadlines or priorities which provided better results than NSGA-II algorithm.

In the work [11] Portaluri, Giuseppe et al. applied MOGA algorithm to reduce the power consumption in data center. The authors evaluate the proposed algorithm by combining between different numbers of auxiliary objectives which affected negatively on quality of results. Moreover, they did not consider the traffic exchanged between VMs. Instead, Zhang, Fan et al. [12] presented a multi-objective scheduling (MOS) scheme for the multitasking workflow application over different virtual clusters. It allows mainly to reduce scheduling overhead time and yet a close to optimal performance. However, this method applied just for a small number of nodes.

In order to optimize both makespan and cost Zhu, Zhaomeng et al. proposed Evolutionary Multi-objective Optimization (EMO)-based algorithm that contain novel encoding scheme to represent the genetic operators. This algorithm gave more stability on the workflow scheduling problem, without conceder more than one pricing schemes, instance type groups or even multi-clouds in a single schedule [13]. Other works based on ant colony algorithm, as the researche [14], the authors take into account the makespan, cost, deadline violation rate, and resource utilization as constraints to achieve a multi-objective optimization of both makespan and cost. The applied method is better than other similar methods results in terms of makespan.

Our major contributions in this work is to impose an integrated solution that covering several aspects of resource allocation in cloud, as following:

- minimize makespan
- minimize cost

III. Problem Description and Definitions

Before solving our studied problem, it is important to define the main actors and the architecture of the task scheduling system in cloud, as shown in the following:

A. Definition

This problem considers the following difinitions:
`Task (T)`: It reflects a set of independent requests $T = \{t_1, t_2, ...t_m\}$ that describe the client's requirement, each task is characterised by identifier id_t and *resource requirment* $(T_{FileSize_i}, T_{CPUi})$.
`Virtual Machine (VM)`: Is a VM image hosted on cloud infrastructure (exactly on servers). It may contain an OS, data files, and applications. Each VM instance is represented by its identifier id_v and *resources available* (V_{Mj}, V_{CPUj}).
`Server Manager`: It provides a centralized platform for managing the set of VMs in data center, allowing to create and deploy a VMs on physical servers quickly and easily. Also, it contains the scheduling mechanisms.
`Host`: It is a server for hosting the VMs.

B. The System Model

As shown in figure 1, users send the requests to cloud provider to express their needs. The server manager analyse the request in order to extract the resources requirment, then assigns the request to the available resources. So that, the provider must serve the customer in an optimal way that meets his requirements. In other words, the main steps of the stadied problem are:

- The client transmits a request to determine its requirements for resources, via a user interface (figure 1).
- Examining and revising the client request for analysing and evaluating how the required service could be provided at a lower cost and in short time (occurs at the level of Cloud Broker).
- The scheduler assigns the tasks to apropriet VMs.

This study deals with the mapping between a set of independent tasks and set of VMs. The VMs are hosted on physical machines (PM). So that, the problem is modeled as follows:
Input:

- Set of $VM_s = \{V_{m1}, V_{m2}, ..., V_{mn}\}$ with different configurations such as CPU type and memory size. Each machine represented by their (id) and millions of instructions per second (mips).
- Set of independent tasks $T = \{t_1, t_2, ..., t_m\}$ with different sizes.

Output: The best mapping of T_i to VM_j (T_i, VM_j), in manner to reduce both makespan and cost .

IV. Problem Formulation

A. The resource cost model

Cost: Our concern here is the cost of resources reservation. In the problem addressed, the cost is expressed as follows:

$$C_i(t_{run_i}) = c_j \times t_{run_i} + C_{tr_{ij}} \tag{1}$$

$$C_{tr_{ij}} = \frac{T_{FileSize_i}}{mips_j \times 32} \times \varepsilon, \qquad \varepsilon = 0.001\$ \tag{2}$$

$$Total\ \ cost = \sum_{i=0}^{m} C_i \tag{3}$$

Where C_i is the cost associated with the execution of task i on the resource j, c_j is the price of using resource j, t_{run_i} is the duration time of running the task i on resource j, $C_{tr_{ij}}$ is the transfer rate cost for a bus of 32 bits wide with a clock

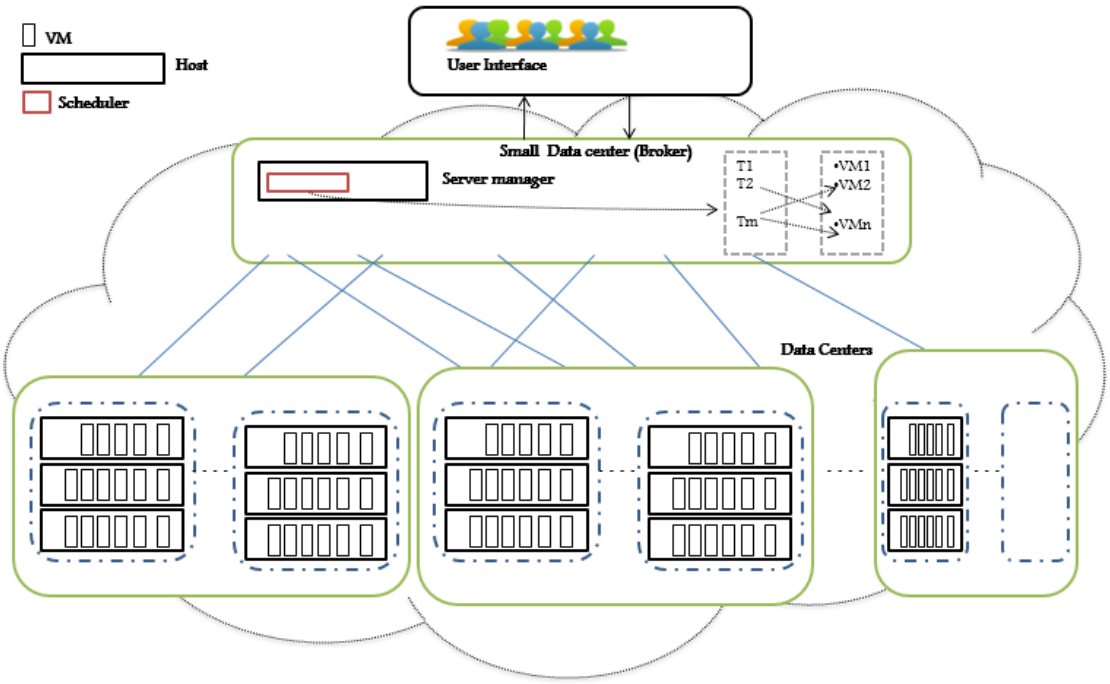

Fig. 1. System architecture

speed of $mips_j$, ε is the transfer price and m is the number of all tasks.

B. The Compulation time (makespan) model

Makespan: It is the total execution time of all the tasks. In this work, the expression of maskespan is given as follows:

$$ET_{ij} = \frac{T_{length_i}}{mips_j} \tag{4}$$

$$Makespan = \sum_{i=0}^{m} ET_i \tag{5}$$

Where ET_{ij} is the execution time of task i on resource j.

C. The function objectives

$$Minimize\ \ \theta(x) = \varphi(x), \phi(x) \tag{6}$$

Subject to :

$$V_{Mj} \geq T_{FileSize_i} \tag{7}$$

$$V_{CPUj} \geq T_{CPUi} \tag{8}$$

$$x \geq 0 \tag{9}$$

Here x is a feasible solution (execution of task i on VM instance j), $\varphi(x)$ is a function of the performance objectives that refer to makespan. $\phi(x)$ is the objective function of the user budget costs. The equation 7 and 8 means that the CPU and memory configuration of VMs must be greater than or equal to the user request requirement.

V. Proposed Algorithm

In order to review our multi-objective method, firstly the fundamentals of the proposed algorithm should be discussing. Then explaining the S-MOGA algorithm steps, as follows:

1) Pareto Dominance: When solving a problem of multi-objective optimization, a multitude of solutions be obtained. Only a limited number of these solutions will interest us. For a solution to be interesting, there must be a relation of `dominance` between the considered solution and the other solutions. More precisely, the vector $\vec{x} = (x_1, x_2, ..., x_n)$ dominates the vector $\vec{y} = (y_1, y_2, ..., y_n)$ if:

- $\vec{x}$ is at least as good as $\vec{y}$ in all objectives,
- $\vec{x}$ is strictly better than $\vec{y}$ in at least one objective.

The solutions that dominate, but do not dominate each other are called `no-dominated` solutions.

2) Genetic Algorithm: It is a search algorithm based on directed random searches to locate optimal solutions. It is meta-heuristic based on the iterative application of stochastic operators on a population of candidate solutions [10].

3) Multiple Objective Genetic Algorithm (M.O.G.A): This method is based on pareto dominance. The "rank" of an individual (order number which ranks an individual in relation to others) is given by the number of individuals who dominate it, in each iteration.

4) S-MOGA: Our proposed approach based on MOGA, but it applies a new way of individual ranking based on the Spacing Distance [5]. The Spacing-MOGA process explained in the following steps :

a) Stop criteria: To determine the stop criteria, Ω defined as stabilizing factor, it increases when the value of maksepan

in the current iteration is the same in precedent iteration, and it take $\Omega = 0$ in otherwise.

b) Initialisation: The first step is beginning with a set of individuals which is called a population. Each individual is a solution to the problem. In the studied problem the genes consider as tasks and the positions of a genes as positions of VMs. This phase aims to dispatch the selected task to a randomly selected available VMs.

c) Crosver: In this phase of our proposed algorithm, for each pair of chromosome to be mated, a crossover point is chosen at middle of chromosome from within the genes.

d) Mutation: In new offspring formed after crossover phase, some of the individuals be flipped randomly in the population string.

e) Mixed population: The initiale population and sorte population are mixed, to form a *mixed population* after the generation of new solutions. Then, the individuals in the mixed population are hierarchically categorised into the dominant and no dominat subsets based on the concepts of dominance mentioned above.

f) Rank: The method inspired by the Spacing Distance used to determine the relationship between two no-dominance solutions, where the following formula is applied:

- Calculate the ω_s (the average distance for each φ_s):

$$\omega_s^{nd} = \frac{\varphi_s^{max} - \varphi_s^{min}}{\sigma - 1} \tag{10}$$

where ω_s^{nd} is the average distance for the no-dominat set, σ is the number of non dominant individuals. The following equation shows how calculate the distance per individual from its neighbors:

$$d^{nei}(i) = \frac{\left|\varphi^i - \varphi^{i-1}\right| + \left|\varphi^{i+1} - \varphi^i\right|}{2} \tag{11}$$

- Measure the new fitness of each individual:

$$\varphi(i) = d^{nei}(i) - \omega_s^{nd} \tag{12}$$

g) Filter Solutions: This step updates our population set, through changing just the solutions that have a worst fitness in the enhanced population pool compared to those achieved by the above steps, in the same iteration.

Figure 2 shows the Organogram of S-MOGA algorithm. The pseudo code of S-MOGA is presented as algorithm 1.

VI. Implementation

A. Simulation Environment

The platform used to execute the experiments was an Intel I5 3320M 2.60 GHz equipped with 4GB RAM with OS Windows 7 Professional. The experiments programmed with Cloudsim toolkit in Eclipse development environment, for modeling and simulation of cloud computing infrastructures and resource allocation [3]. It is a simulator tool founded on Java application, which is an object-oriented computer programming language and a portable computing execution environment [6].

Algorithm 1 S-MOGA

Input : $T_1, T_2, ..., T_m$, $VM_1, VM_2, ..., VM_n$
Result: An optimized generated schedule
Ω:=0;
while $\Omega \leq 10$ **do**
  function INITIALISATION( )
    function CROSSOVER( )
      function MUTATION( )
        function MIXED POPULATION( )
          function RANK ( )
            function FILTER SOLUTIONS ( )
end
Return the best mapping (Tasks, VMs) as the best found solutions.

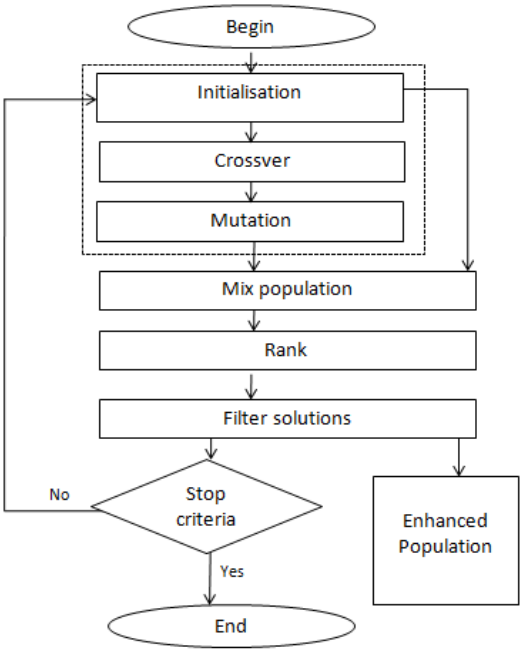

Fig. 2. Organogram of the S-MOGA

B. Experimentation Results

The experiments, mainly focus to evaluate the makespan of our proposed algorithm. Also, study the variation of budget costs for resource utilization. Besides, compare the proposed algorithm of this paper with the original Particle Swarm Optimization (PSO) algorithm, the classical heuristic algorithm Max-Min and MOGA scheduling.

The test is done in two cases: in the first case assumes that the number of VMs is 16789. In the second case considers that the number of VMs is 34568. The MIPS of each VM is between [2000, 2050]. The length of tasks is between [100, 1070]. The configuration of image size, VM memory and VM bandwidth is illustrated in Table I. Furthermore, the number of tasks varying as 1000, 1678, 2874 and 3456. In addition, the test considers the reservation cost of various VMs 0.02 $/hour.

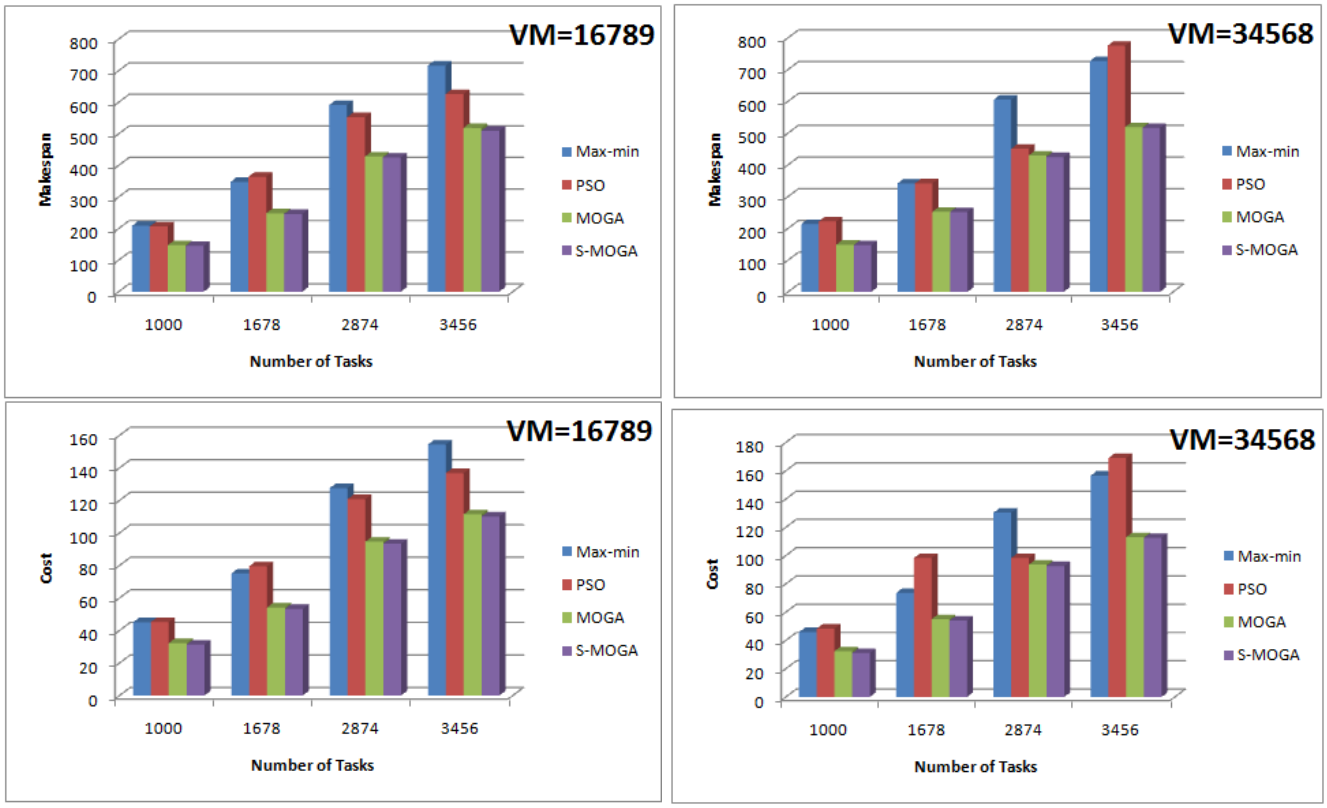

Fig. 3. Comparisons of proposed scheduling algorithm with Max-min, PSO and MOGA algorithm for Makespan and Cost

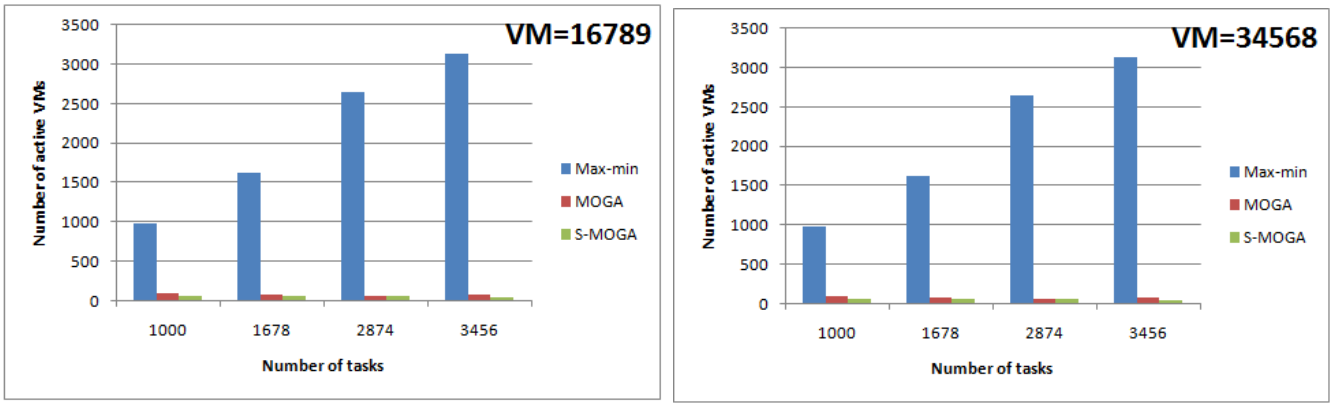

Fig. 4. Comparisons of proposed scheduling algorithm with Max-min and MOGA algorithm for Availability of Resources

TABLE I
THE PARAMETER SETUP OF VMS

size	ram	mips	bw
10000 MB	[500, 512]	[2000, 2050]	1000

In order to evaluate the availability of resources aspect of our proposed algorithm, the next test keeps the same range of MIPS and task length variation. Also, defines the Availability of Resources (AR) of the system as a number of *inactive VM* in each allocation. Hence, considers that the AR is increased only if the number of the VM active in the system is minimized, which means a good availability of the resources for next allocation. So that, the AR of the system is calculated as shown in the following equation:

$$AR = m - \sum_{j=0}^{k} Av_j \tag{13}$$

Where Av_j is an active VM_j and k is the total number of

TABLE II
SIMULATION RESULTS OF MAKESPAN AND COST FOR MAX-MIN, PSO, MOGA AND PROPOSED ALGORITHM

				Max-min		PSO		MOGA		S-MOGA	
Tasks	**Lengthe**	**VM**	**MIPS**	**Makespan**	**Cost**	**Makespan**	**Cost**	**Makespan**	**Cost**	**Makespan**	**Cost**
1000	[200,300]	9087	[2340,3450]	89.130	20.247	82.11	18.968	61.142	14.126	60.699	14.018
2345				215.897	48.931	191.323	44.283	142.760	32.983	140.558	32.548
3456				316.229	71.758	277.063	64.028	210.747	48.707	208.346	48.188
4000				369.269	83.677	306.841	70.8797	244.300	56.4797	241.308	55.789
1000	[900,2345]	10056	[12340, 45670]	71.815	14.689	58.477	12.002	35.798	7.3503	35.706	7.325
2345				165.3199	33.805	131.782	27.055	84.677	17.3792	83.792	17.200
3456				246.355	50.373	178.085	36.551	123.990	25.453	122.844	25.217
4000				281.602	57.597	190.499	39.096	142.753	29.308	142.723	29.295
1000	[9000,22556]	30974	[56720, 345960]	116.048	23.264	86.391	17.326	50.760	10.1795	50.9299	10.213
2345				268.527	53.829	168.676	33.828	119.206	23.905	116.483	23.3598
3456				403.235	80.833	356.743	71.5409	174.846	35.064	173.772	34.848
4000				464.007	93.017	305.234	61.211	202.784	40.666	200.098	40.129

TABLE III
SIMULATION RESULTS OF AVAILABILITY OF RESOURCES FOR MAX-MIN, MOGA AND PROPOSED ALGORITHM

				Max-min	MOGA	S-MOGA
Tasks	**Lengthe**	**VM**	**MIPS**	**VM active**	**VM active**	**VM active**
2354	[100, 500]	1000	[234, 567]	896	27	4
3785				972	34	8
4000				983	38	6
2354	[1950, 2700]	3789	[7567, 8464]	1747	2829	20
3785				2403	2818	17
4000				2450	2819	19

Av, m is the total number of VMs.

1) Makespan and Cost: As it is shown in figure 3, the S-MOGA approach has a lower values of makespan and cost compared to the other approaches Max-min, PSO and MOGA.

The experiment is repeated with varying: the MIPS value of VMs at different intervals, for various numbers of tasks and VMs. So that, the simulation takes 1000, 2345, 3456 and 4000 numbers of tasks run on 9087, 10056 and 30974 heterogeneous machines successively in a cloud (Table II).

The results illustrated in the table II indicate clearly that the proposed scheduling algorithm performs well compared with other algorithms, it gave significant improvements in terms of makespan and cost of resource reservation.

2) Availability of resources: To examine the impact of our proposed algorithm on the AR, the experiment is reoccurred by using a new configuration of VMs: different intervals of mips, various numbers of tasks and VMs. The test done for 2345, 3785 and 4000 numbers of tasks distributed over 1000 and 3789 heterogeneous VMs successively (Table III).

The figures 4 and the results of table III show that the proposed algorithm allows using a minimum of resources by consolidate the execution of tasks over few number of VMs and make off others which ensure the availability of resources.

VII. CONCLUSION

This paper presents an improvement of Multi objectives Genetic Algorithm based on spacing distance (S-MOGA) to enhance the Quality of Service (QoS) requirements in Cloud by minimizing the total task execution time and cost. We considered various independent tasks with different lengths, which are corresponding to user request and various VMs to mimic resource allocation in cloud. The experimental results shown that the proposed algorithm generated results better to those of the MOGA, PSO and Max-min in terms of makespan and cost. In addition, we studied the performance of the Spacing-MOGA from the viewpoint of their feasibility to meet the needs of users. Where the results confirmed that the proposed algorithm offering a good availability of resources compared with other methods.

In future work, we will consider other scheduling algorithm that can be used to solve a resource allocation problem in a Cloud computing environment. As well, we will study its ability to manage the resources in a dynamic manner.

References

[1] Ali Belgacem, Kadda Beghdad-Bey, and Hassina Nacer. Task scheduling in cloud computing environment: A comprehensive analysis. In International Conference on Computer Science and its Applications, pages 14–26, Algiers, Algeria, 24-25 April 2018. Springer.

[2] Kadda Beghdad Bey, Farid Benhammadi, Mohamed El Yazid Boudaren, and Salim Khamadja. Load balancing heuristic for tasks scheduling in cloud environment. In Proceedings of the 19th International Conference on Enterprise Information Systems – Volume 1: ICEIS,, pages 489–495, April 26-29, in Porto, Portugal, 2017. INSTICC, SciTePress.

[3] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience, 41(1):23–50, 2011.

[4] Juan J Durillo and Radu Prodan. Multi-objective workflow scheduling in amazon ec2. Cluster computing, 17(2):169–189, 2014.

[5] L Falahiazar and H Shah-Hosseini. Optimisation of engineering system using a novel search algorithm: the spacing multi-objective genetic algorithm. Connection Science, pages 1–17, 2018.

[6] Tarun Goyal, Ajit Singh, and Aakanksha Agrawal. Cloudsim: simulator for cloud computing infrastructure and modeling. Procedia Engineering, 38:3566–3572, 2012.

[7] Ashish Gupta and Ritu Garg. Load balancing based task scheduling with aco in cloud computing. In Computer and Applications (ICCA), 2017 International Conference on, pages 174–179, Doha, United Arab Emirates, 6-7 Sept 2017. IEEE.

[8] Mala Kalra and Sarbjeet Singh. A review of metaheuristic scheduling techniques in cloud computing. Egyptian informatics journal, 16(3):275–295, 2015.

[9] Mohammad Masdari, Sima ValiKardan, Zahra Shahi, and Sonay Imani Azar. Towards workflow scheduling in cloud computing: a comprehensive analysis. Journal of Network and Computer Applications, 66:64–82, 2016.

[10] Mohand Mezmaz, Nouredine Melab, Yacine Kessaci, Young Choon Lee, E-G Talbi, Albert Y Zomaya, and Daniel Tuyttens. A parallel biobjective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing, 71(11):1497–1508, 2011.

[11] Giuseppe Portaluri and Stefano Giordano. Multi objective virtual machine allocation in cloud data centers. In Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on, pages 107–112, Pisa, Italy, 3-5 Oct 2016. IEEE.

[12] Fan Zhang, Junwei Cao, Keqin Li, Samee U Khan, and Kai Hwang. Multi-objective scheduling of many tasks in cloud platforms. Future Generation Computer Systems, 37:309–320, 2014.

[13] Zhaomeng Zhu, Gongxuan Zhang, Miqing Li, and Xiaohui Liu. Evolutionary multi-objective workflow scheduling in cloud. IEEE Transactions on parallel and distributed Systems, 27(5):1344–1357, 2016.

[14] Liyun Zuo, LEI Shu, Shoubin Dong, Chunsheng Zhu, and Takahiro Hara. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access, 3:2687–2699, 2015.