Improved Virus Optimization algorithm for two-objective tasks Scheduling in Cloud Environment

Kadda Beghdad Bey[1], Sofiane Bouznad[1], Farid Benhammadi[1] and Hassina Nacer[2]

[1]*Laboratory of Distributed and Complexes Systems, Ecole Militaire Polytechnique, BP 17 Bordj el Bahri, Algiers, Algeria (k.beghdadbey, bouznad.sofiane and fbenhammadi2008)@gmail.com*

[2]*MOVEP Laboratory. Computer Science Department University of Science and Technology, USTHB, Algiers, Algeria sino nacer@yahoo.fr*

***Abstract* — *Cloud computing* is increasingly recognized as a new way to use on-demand, computing, storage and network services in a transparent and efficient way. The development of applications in cloud environments is faced with the need to efficiently schedule a large number of tasks and resources. However, in the most of the time, the resources in cloud are not efficiently utilized due to inadequate scheduling task algorithm in virtual machines. Therefore, task scheduling is one of the most challenging issues in cloud computing. In this paper, we propose two-objective virus optimization algorithm of the makespan and the cost, for mapping tasks to virtual machines in order to meet the needs of cloud service quality and proper assignment of resources. Thus, based on genetic algorithm some parameters of Virus optimization algorithm are redefined to strengthen sorting ability between virus infection strategies. Our combined methods aims to improve the performance of scheduling algorithms. It outperforms some existing approaches for task scheduling in Cloud computing.**

Keywords: *Independent task scheduling, Cloud computing, Virus optimization algorithm, Genetic algorithm, Two-objective optimization, CloudSim simulator.*

I. INTRODUCTION

Within few years, IT industries start using Cloud Computing (CC) by serving on demand requests of the users with self-managed virtual infrastructure and with efficient resources utilization. Growth of cloud computing slower down the efficiency, throughput and resource utilization for which cloud computing need to be evolve. Task scheduling is considered as one way to enhance the efficient resource utilization in cloud environments. In fact, independent computational tasks are supposed to be executing in parallel when they are executed concurrently on different virtual machines. The scheduling strategy defines the instants when the algorithm is called to produce a schedule based on the resources performances forecasting and independent computational tasks to be executed. The task scheduling strategies can be classified on two different types: static and dynamic. Static strategies define a schedule at compile or at launch time based on the knowledge of the processors availability and tasks to execute. Dynamic strategies, are applied when the tasks arrival time is not known a priori and therefore the system needs to schedule tasks as they arrive [1].

Plethora of multi-objective optimization techniques for independent tasks scheduling has been proposed for assignment of tasks to machine in Cloud Computing systems [2-11]. The Min-Min heuristic is the well-known standard scheduling algorithm for its performance, simplicity and practicability. The Min-Min heuristic [3] gives the highest priority for dynamic tasks scheduling in heterogeneous computing systems. The principle of Min-Min heuristic is to finish each task as early as possible and it schedules the tasks with the selection criterion of minimum earliest completion time.

In this paper, we are interested in two-objective optimization for scheduling independent tasks in cloud computing environment. The optimization process consists to minimize the makespan value and the operational cost in order to ensure the performance and quality of service in the cloud. The proposed optimization approach uses a virus optimization algorithm [10]. Thus to provide an improvement of this algorithm for task scheduling purpose, we propose GA operators to select strong and weak virus for exploitation and exploration in space search. In addition, the initialization step is based on standard scheduling algorithms such as Min-min, Max-min and Tabou. Experimental results show that our proposed approach improves the performance of several existing approaches in literature such as the Min-Min (Max-Min) heuristic, Genetic heuristic and Virus optimization algorithms. Moreover, we present in this paper a comparative study in which the proposed scheduling approach is evaluated with improved Particle Swarm Optimization (PSO) approach [12] using a set of different types of Expected Time to Compute (ETC) matrices up to randomly heterogeneous machines and heterogeneous tasks using simulated and real data.

The remainder of this paper is organized as follows. Section 2 describes the relevant multi-objective optimization scheduling problems in Cloud environment. Section 3 formulates the two-objective optimization problem for independent tasks scheduling. Section 4 details the proposed standard combination, Virus optimization and genetic algorithms used for resolving this problem. Experimental results and performance evaluations of the proposed combination are reported in section 5. Finally, section 6 presents conclusions and directions for future works.

II. Related Works

Various classical scheduling tasks algorithms have been proposed and deployed until date in CC environments, such as: First Come First Serve, Min-Min, Min-Max based scheduling. These classical schemes posed drawbacks of resulting into more execution time and reduced throughput. Recently, several researches works focus on multi-objective optimization tasks scheduling methods in CC and proposes several techniques. Most of them are based on evolutionary algorithms to minimize the operational cost and ensure the performance and quality of service in CC. A vast literature exists on bio inspired approaches for optimized scheduling tasks in CC. Ahmad et al. [11] proposed a hybrid genetic algorithm for solving the workflow-scheduling problem and optimizing the load balance for maximum resource utilization. The Multi-objective particle swarm optimization is another class of task scheduling problem that has been addressed in CC environment [12, 13]. Generally, the algorithms based on this optimization achieved best performances compared to the classical scheduling methods. In recent times, others methods use the combination of several evolutionary strategies. This combination of evolutionary type optimization algorithms has provided the best ways to solve multi-objective optimization problems, because of their efficiency, robustness and quick convergence compared to strategies using only single evolutionary optimization method. In 2015, Shu, and al. [14] proposed an improved clonal selection algorithm for meeting the service level agreement requested by the users. The experimental results show that the proposed algorithm performs better than other two algorithms with minimum execution time and increased throughput of the cloud computing system. In [15], authors proposed an algorithm based on the combination of genetic algorithm along with fuzzy optimization theory. Another multi-objective optimized tasks scheduling algorithm using genetic algorithms with greedy approach is proposed in [16]. This algorithm not only performs task scheduling but also perform others load balancing methods in Cloud environment. Chu [17] used the combination of genetic algorithm and support vector machines for two-objective optimized tasks scheduling algorithm. Based on the cross operation of genetic algorithm and components selection of partial regression, the proposed work gives a high effective scheduling and service cost may be reduced in cloud computing environment using both completion time and cost of tasks. Similarly, Kim et al. [18] have developed biogeography-based optimization for tasks scheduling. This algorithm performs more satisfactorily than other optimization algorithms, such as genetic algorithm and particle swarm optimization, in large size problems. Lakra et al. [19] proposed a Two-objective tasks scheduling algorithm for mapping tasks to a VMs improving the throughput of the datacenter and reducing the cost without compromising the service level agreement in cloud environment. The proposed scheduling algorithm in [20] is involving Non-dominated sorting which targets the two-objective issues considering completion time and cost minimizations. The exposed results by the authors outperform the preceding algorithms and this method represents better convergence performance and resource scheduling capability in the different number of resources. These combination strategies perform satisfactorily than other multi-objective optimization problems using only genetic algorithm or particle swarm optimization.

III. Problem Definition

In cloud computing, many datacenters consist of several servers where each server runs a number of virtual machines that have different capacity to execute tasks with different QoS parameter. Scheduling refers to the mapping or assigning a task to a specific virtual machine, such that resource utilization increases.The main problem is to bind set of tasks received by the broker to the virtual machines with the respect of optimized QoS. Based on the makespan and the cost, this problem can be modelled using two-objective optimization for tasks scheduling problem. This optimization model includes two objective functions to be minimized simultaneously:

$$Find\ X = \{x_1, x_2, \ldots\ldots\ldots\ldots x_n\}$$

Which minimize two objective functions $f_1, f_2 : Min(f_1(x), f_2(x))$, where X is the feasible solutions set.

Our two-objective model for task scheduling optimization in CC environment can be described by a triplet (*T, VM, F*). $T = \{t_1, t_2, \ldots, t_n\}$ is a consumer tasks set including *n tasks*, $VM = \{vm_1, vm_2, \ldots, vm_l\}$ is a virtual machine resource set including *l* virtual machine and *F={Makespan, Cost}* is a set of the considered cloud resource scheduling and optimization functions. The proposed model is defined as follows:

The *Makespan* is the maximum completion time of all tasks in all virtual machines. It can be obtained using the Expected Time to Compute (*ECT*) matrix by the following equation:

$$Makespan = \underset{1 \le j \le m}{Max} \sum_{i=1}^{n} \left(ETC_{ij} \times x_{ij} \right)$$

Where :

$$x_{ij} = \begin{cases} 1 & \text{if a task } t_i \text{ will run on } vm_j \\ 0 & \text{otherwise} \end{cases}$$

- ETC_{ij} is the estimated execution time of task t_j on machine vm_i and the values of ETC matrix are calculated by the following equation:

$$ETC_{ij} = \frac{L_i}{mips_j}$$

L_i indicates the number of instructions required by the task T_i (implementation time) and $Mips_j$ indicates the frequency of cloud computing virtual machine vm_j.

However, the *Cost function* in our tasks scheduling problem is based on the cost C_j of each virtual machine vm_j. The total execution cost for all tasks scheduling can be defined according to the following formula:

$$C = \sum_{j=1}^{l} \sum_{i=1}^{n} ETC_{ij} \times x_{ij} \times C_j$$

Above all, we need to find the most reasonable tasks scheduling which minimizes the makespan and the cost in cloud system. Thus, the two-objective tasks scheduling problem can be modeled by the aggregation (Weighted Sum Method) of objective optimization functions defined as:

$$Z(x) = Min(\lambda \times Makespan + (1-\lambda) \times C)$$

under the constraints:

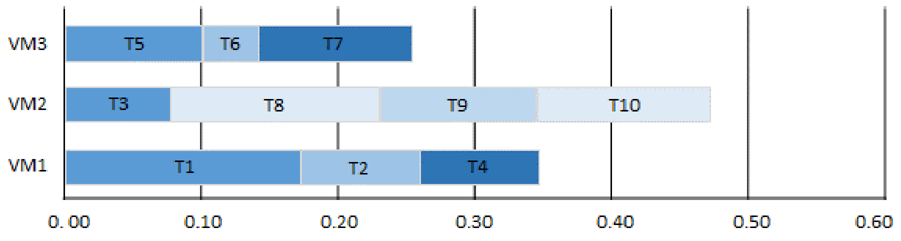
$$\begin{cases} \sum_{j=1}^{m} x_{ij} = 1 & \forall i \in T \\ x_{ij} \in \{0,1\} \\ 0 \le \lambda \le 1 \end{cases}$$

Where λ and (1-λ) represents the weights of the makespan and the cost in our two-objective function.

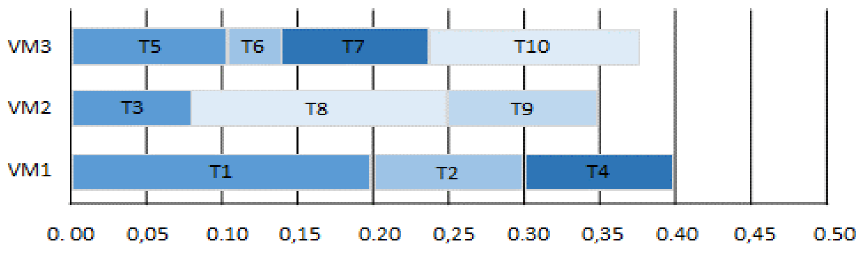
In two-objective optimization, a sufficient solution that minimizes these two objective functions at the same time can considers their linear combination (aggregation called Weighted Sum Method) or Pareto optimal solutions. These solutions cannot be enhanced for any objective without degrading minimum of the other objective. For example, consider the following case where a scenario of *ETC* matrix defined by ten tasks and three virtual machines as given in Table 1. According to the two task assignments (Figure 1), we obtain two solutions (Makespan=0.68, Cost=80.13$) and (makespan=0.40, Cost=82.8$). Therefore, these solutions show that the makespan objective can degrade the cost objective in the solution selection. Thus, we can select one solution based on the λ value according to the weights of these two objectives.

TABLE I. AN EXAMPLE OF ETC VALUES

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	*Cost($)*	*Mips*
Vm_1	0.10	0.20	0.05	0.10	0.15	0.06	0.15	0.10	0.06	0.20	60	1000
Vm_2	0.17	0.33	0.08	0.17	0.25	0.10	0.25	0.17	0.10	0.33	40	600
Vm_3	0.07	0.13	0.03	0.07	0.10	0.04	0.10	0.07	0.04	0.13	120	1500



(a) Scenario 1



(b) Scenario 2

Fig 1. Two scenarios of tasks scheduling in CC.

IV. PROPOSED META-HEURISTIC APPROACH: MOVOA

The development of applications for CC environments is being challenged by the need of scheduling a large number of tasks, datasets and resources efficiently [21]. The general problem of optimally mapping tasks to machines has been shown to be NP-complete. In this work, we propose a combined evolutionary heuristics for solving independent task scheduling problems in CC environment. This combination uses the two-objective optimization methods based on adapted virus optimization algorithm and genetic operators. This combination allows finding the task assignment that minimizes the makespan and the cost. In this section, we briefly give a description of the original virus optimization algorithm [10] and thereafter we introduced our tasks scheduling method based on improvement of this algorithm.

A. Virus Optimization Algorithm

The Virus Optimization algorithm is a meta-heuristic technique, population-based solution used to solve many optimization problems with single objective like continuous domain problems [10]. This algorithm imitates the behavior of the viruses attacking a living cell by infection. Once they are entered, they will start replicating and alter the genetic material of the host cell. More viruses will be produced and ultimately host cell will die. In this algorithm, solution space is taken as cell itself and global optima can be found inside the cell. Many viruses can coexist within a host cell and each such virus represents a solution in the solution space. The viruses are divided into two categories: Strong and Weak, which corresponds to the exploration and exploitation capability of the virus optimization algorithm. Strong viruses will have high objective function value compared to weak viruses and they will replicate faster than them. The algorithm does mainly three types of phases: initialization, replication, updating and maintenance. An advantage of this algorithm is that it can be easily parallelized and therefore easily implemented.

The initialization step occurs according to the fitness value and consist to create the starting population of the possible solutions (virus) fulfilling the constraints. These solutions are sorted based on the objective function evaluation to select strong and weak virus [10]. In the replication step, VOA generates new virus where are ranked in strong and weak ones according to some objective function values. For example, if the required number of strong viruses is three, then the first three virus of the created list are considered strong while the remaining are considered weak. In the last step, once the new viruses have been generated using the replication process, VOA checks and maintain the new population size using the corresponding objective function values. As a result, only the strong viruses are kept ordered according to their objective function values. The checking process verifies the convergence and VOA determines whether the exploitation has to be intensified by creating new members closer to the stronger viruses [10]. However, the Maintain process (antivirus) is activated by interaction between the viruses and the host cell; the antivirus is triggered at each replication, killing a given number of viruses according to some fixed parameters such that the number of strong virus, weak virus and population size [10]. For example, if the needed number of strong viruses is three, then the first three viruses in the created virus list are considered strong while the remaining viruses are considered weak. In addition, this process eliminates some virus from the population if the total number of viruses inside the host cell exceeds 1000 virus. Finally, the stopping condition of the VOA algorithm can be the maximum number of the iterations or the maximum number of the replications.

B. Task scheduling based on improved VOA

Based on the original VOA, we adapted modifications in order to enhance the performances of initialization and replication steps. This following section presents the description to accomplish to the VOA adaptation for two-objective optimization based tasks scheduling in CC, called MOVAO. The population initialization step uses standard algorithms such as Min-Min, Max-Min and Tabou to generate the initial population. In the modification step, a combination of the genetic operators in the virus replications of the classical VOA is applied. Fig.2 shows the visual representation for the main steps of the proposed two-objective optimization process based on VOA and genetic algorithm.

The presented diagram consists of four steps: Initialization, classification, replication and antivirus (maintenance). The proposed two-objective tasks scheduling algorithm is described in Algorithm 1. First, we defined the control parameters of the proposed MOVOA as follows:

. ***Pop_{size}***: *Size of the initial population;*
. ***Pop_{virus}***: *Maximum number of viruses in a cell;*
. *N_{task}*: *Number of tasks to be performed;*
. *N_{vms}*: *Number of virtual machines;*
. *$P_{virusStrong}$*: *The proportion of strong viruses in a population;*
. *$N_{virusStrong}$*: *Number of strong viruses;*
. *N_{virus}*: *Number of viruses to be removed in the antivirus process;*
. *$N_{repStrong}$*: *Number of viruses to generate from a strong virus;*
. *$N_{repWeak}$*: *Number of viruses to generate from a weak virus;*
. ***ETC***: *Matrix of task execution times in virtual machines;*
. ***C***: *Vector of execution cost on each virtual machine;*
. *S_{best}*: *Vector representing the best solution found.*

In our proposed MOVOA, the population initialization is based on some standard scheduling algorithms such as Min-Min, Max-Min and Tabou. According to the logic of the VOA, this first step is to create the starting population of the possible solutions fulfilling the constraints of these standards algorithms.

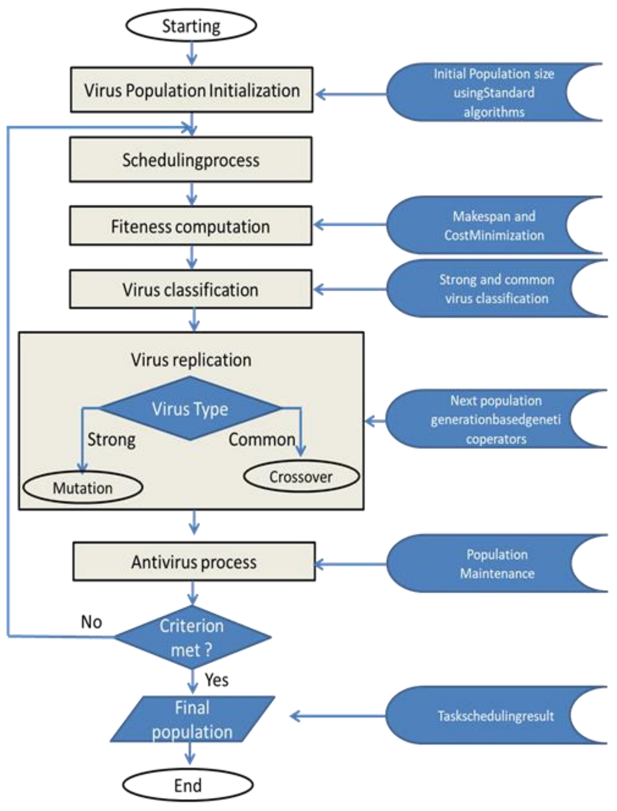


Fig.2 MOVOA Flow chart for task scheduling in CC.

```
Algorithm 1: MOVOA Algorithm
1 : Input : $Pop_{size}$, $N_{tasks}$, $N_{VM}$, $P_{VirusStrong}$, $ETC$, $C$, $Pop_{Virale}$;
2 : Output : $S_{best}$;
3 : $Pop[Pop_{size}]$ = Initialze_Population($Pop_{size}$, $N_{tasks}$, $N_{vm}$) ;
4 : $Moy$ = Evaluate_Fitness($Pop$, $ETC$) ;
5 : $i = 1$ // Iterations number;
6 : while ($i$ <= Iteration-max) do
7 :         classification($Pop$, $P_{VirusStrong}$) ;
8 : [$N_{rep\text{-}strong}$, $N_{rep\text{-}low}$] = random(1,10); // Avec $N_{rep\text{-}strong} > N_{rep\text{-}low}$
9 : $Pop_{new}$= replication ($Pop$, $N_{rep\text{-}strong}$, $N_{rep\text{-}low}$) ;
10 : Calculate the viruses number to be eliminated according
     to the equation:
     $N_{virus}$= rand (0, Population~~size~~− $Strong_{members}$);
11 : antivirus ($Pop_{new}$, $N_{virus}$) ;
12 : $Average_{new}$= Evaluate_Fitness($Pop_{new}$, $ETC$) ;
13 :if ($Average_{new} > Average$) then
14 : Increment intensity by1;
15 : endif
16 : if (size($Pop_{new}$) >$Pop_{virale}$) then
17 : reduction ($Po_{pnew}$) ;
18 : endif
19 : $Pop = Pop_{new}$;
20 : $Average = Average_{new}$;
21 : $i = i + 1$ ;
22 : endwhile
23 : return the best solution of $Pop$ : $S_{best}$;
```

Once the starting population has been initialized, each virus of the population will must be classified as strong or weak virus according to their objective function values and some predefined parameters (number of strong viruses). As shown in algorithm 2, this classification process is based on the fitness function (two-objective function $Z(x)$) and the strong virus number. For example, if we fix the strong viruses number to three, so the first three viruses in the created list are then considered strong while the remaining viruses are considered weak.

```
Algorithm 2: Population classification into strong and
             weak viruses
1 : Input :$Pop$, $P_{virusStrong}$;
2 : Trier($Pop$) according to the fitness function;
3 : $N_{virusStrong}$= size($Pop$) ×$P_{virusStrong}$;
4 : for $i = 0$ to size($Pop$) do
5 :     if ($i < N_{virusStrong}$) then
6 :         setType($Pop[i]$, strong) ;
7 :     else
8 :         setType($Pop[i]$, weak) ;
9 : endif
10 : endfor
```

After classifying the viruses, each ranked virus in the population will reproduce by creating new viruses. Thus, two strategies for this reproduction are adopted. For the strong virus, four viruses and only three new viruses for weak are generated. The reproduction process based on improved replication function is designated in Algorithm 3.

The improvement uses two genetic operators according to virus types. In order to avoid the local convergence for strong virus reproduction (local solution), the mutation operator is used for these types of viruses. However, one-point and two-point crossover operators are used only for weak viruses. For example, for the two-point crossover operator, we generate two new viruses from two weak viruses, which are randomly selected from the population, as illustrated in figure Fig.3.

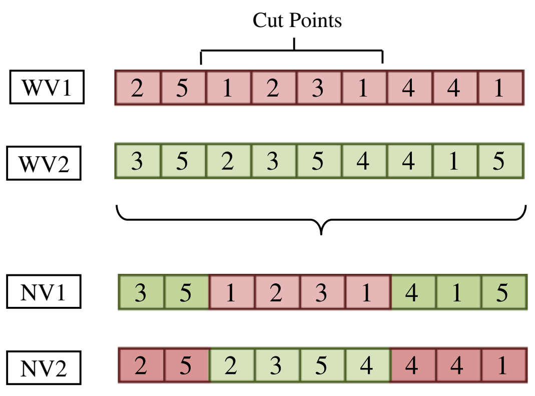


Fig.3 An example for crossover operation for two weak viruses

Algorithm 3: Improved replication with genetic operators

```
1 : Input :$Pop$, $N_{repStrong}$, $N_{repWeak}$;
2 : Output :$Pop_{new}$;
3 : $Pop_{new}$= $Pop$ ;
4 : for $i$ = 0 to size($Pop$) do
5 : if (Type($Pop[i]$) == $strong$)then
6 : Insert ($listVirusStrong$, $Pop[i]$) ;
7 : else
8 : Insert ($listVirusWeak$, $Pop[i]$) ;
9 : endif
10 : endfor
11 : while ( $listVirusWeak$! = $Empty$ ) do
12 : for $i$ = 0 to $N_{repWeak}$ do
13 : // Remove two viruses from the list of weak viruses
14 : [$virus1$, $virus2$] =Insert ($listVirusWeak$) ;
15 : [$fils1$, $fils2$] =Crossing ($virus1$, $virus2$) ;
16 : Insert ($Pop_{new}$, [$fils1$, $fils2$]) ;
17 : endfor
18 : endwhile
19 : while ( listVirusForts! = $V ide$ ) do
20 : for $i$ = 0 to $N_{repStrong}$ do
21 : // Remove one viruses from the list of strong viruses
22 : $virus1$ = Remove ($listVirusStrong$) ;
23 : $fils1$ = Mutation($virus1$) ;
24 : Insert ($Pop_{new}$, $fils1$) ;
25 : endfor
26 : endwhile
```

After the replication process, the virus maintenance process execution (Antivirus algorithm) killed a given number of viruses according to the original mechanism [10] among the weak viruses set, because the weak viruses performing worse. In addition, if the total number of viruses inside the host cell exceeds 1000, generally the maintenance process will reduce the population size to the amount set initially defined (1000 viruses in our case). This extermination is based on fitness function values as a criterion performance of the virus population. The proposed antivirus process is described in Algorithm 4. Finally, if population performances did not improve after some replication process, the algorithm stops.

Algorithm 4: Principe of antivirus function ().

```
1 : Input :Pop, $N_{virus}$ ;
2 : Output :$Pop_{new}$;
3 : $Pop_{new}$ = Pop ;
4 : Trier($Pop_{new}$) according to the fitness function;
5 : for i = 0 to $N_{virus}$ do
6 :        Eliminate the last virus of $Pop_{new}$ ;
7 : endfor
```

V. RESULT AND EXPERIMENTS

In this section, an evaluation and comparison to validate the proposed strategy for task scheduling based MOVOA in Cloud Computing is presented. The conducted experiments use an Intel®Core™ i7-2600 machine. Moreover, we ran a set of experiments to compare it with the other heuristic-based approaches for independent task scheduling using the Cloudsim. The proposed approach has been applied to simulated and real data, with 12 different types of ETC matrix up to 16, 32, 64, 128 and 256 heterogeneous machines, and up to 512, 1024, 2048, 4096 and 8182 randomly generated heterogeneous tasks used in [2]. Thus, we evaluate the proposed MOVOA algorithm in order to minimize two objectives that are the makespan and the cost. These different types of ETC matrix are generated based on the following properties [2]:

Task heterogeneity – represents the amount of variance among the execution times of tasks for a given machine. The task heterogeneity is defined as: lo: low and hi: high.

Machine heterogeneity – represents the variation among the execution times for a given task across all the machines. The machine heterogeneity is defined as: *lo: low and hi: high.*

Consistency – an ETC matrix is said to be consistent whenever a machine m executes all tasks faster than another machine and the inconsistency if the machine m may be faster than another machine for some tasks and slower for others. The consistency type is defined as: *c: consistency; s: semi-consistency and i: inconsistency.*

For real data, we have created many VMs and tasks with different task size using log file introduced in [24]. Task size ranges from 100 to 10000 and the virtual machines from 3 to 48. The VMs have been created with the same processing power MIPS range.

First, we give the results of the improved initialization process adopted in our MOVOA. For simulated data, the obtained results in Table 2 indicate that the standard algorithms (Min-max, Min-Min and Tabou) in initialization process outperform the original VOA. Thus, it is important to note that our improved initialization process in VOA permits to give this algorithm competitive with standard algorithms presented in the literature.

For the setting of the three MOVOA parameters such as growth rate of weak viruses (B) and growth rate of strong viruses (B) and strong virus rate in the population (C), Pareto-optimal front is adopted, where each parameter and their combination are tested. Fig.4 illustrates the influence results of these parameters. As can be seen, the growth rate of strong viruses is the more important parameter for our MOVOA.

Fig.5 shows the influence of the strong virus rate on MOVOA convergence. Experimental results from this figure show that the strong virus rate in the population (C) has better convergence speed and search ability in solving the task scheduling problem. Our simulation parameters setting give the final results adopted for our MOVOA approach. Table 4 summarizes the values of these parameters.

TABLE II. RESULTS OF IMPROVED VOA IN INITIALIZATION PROCESS

Benchmarks		VOA	VOA-MaxMin	VOA-MinMin	VOA-TS
512×16	A.u_c_hihi	33515841,2075974	**9478748,54705373**	11641581,3124835	40734849,2597272
	A.u_c_hilo	3737479,59860668	**139640,463361260**	890323,882382025	717846,168886722
	A.u_c_lohi	9895642,40071556	**325969,379074195**	3548603,69777885	1806764,43897363
	A.u_c_lolo	118682,944348105	**4741,88496593059**	24576,3120032209	186154,388470771
	A.u_i_hihi	54581417,6358823	**5744281,35952104**	46888394,9179914	54177713,8001841
1024×32	A.u_c_hihi	4383867358,88497	**81168110,2585890**	597798515,574730	6242746503,06729
	B.u_c_hihi	1246601703,33311	**24296362,6739020**	191214363,403808	1736816774,16309
	B.u_i_lolo	536609,894367258	**1825,32098972682**	39076,0630305110	553247,143843531
	B.u_s_hi	1620976031,0024	**14406243,8979040**	207133649,666086	1631866310,84382
	B.u_s_hilo	16045273,0005123	**161710,828037789**	2246707,62102429	16504173,9119531
2048×64	A.u_c_hihi	10540814353,3380	**68382087,7694320**	4066453996,05607	21023179983,7550
	B.u_c_hihi	1020053393,50667	**6809804,53715252**	429375587,707337	1961912055,42496
	B.u_s_hilo	34185834,4073316	**156508,279151177**	2833080,10334692	35777433,2609869
	B.u_i_lolo	1150676,97244939	**1062,26755656421**	40530,6137005020	1160879,21708258
	B.u_s_hihi	3326676485,82300	**14012103,8195210**	286662462,690487	3523788387,30358
4096×128	A.u_c_hihi	23996285871,9060	**6624387,60572100**	399220512,895295	24316863996,2840
	B.u_c_hihi	2411310541,92293	**559791,014140813**	57468211,0218529	2449266772,19776
	B.u_i_lolo	2411851,97259703	**553,207997194039**	41703,9413734879	2458020,21421099
	B.u_s_hilo	71071822,1568924	**132058,746687681**	4053296,88793036	73666932,4830240
	B.u_s_hihi	7163570672,51973	**12989954,2801580**	370826776,222161	7359095019,85810
8192×256	A.u_c_hihi	50062019882,6850	**2855306,64712900**	410602822,406195	50514086017,2900
	B.u_c_hihi	146893067,056463	**936711,747171347**	123165194,094830	14813093403,2000
	B.u_i_lohi	4946876,98939152	**336,242668651566**	41518,2627413454	5005558,14873186
	B.u_s_hilo	149517372,311974	**109082,878871015**	5591057,69139399	15091076305,0000
	B.u_s_hihi	26761574493,4720	**3612402,92343436**	118162914,600465	28141746047,1400

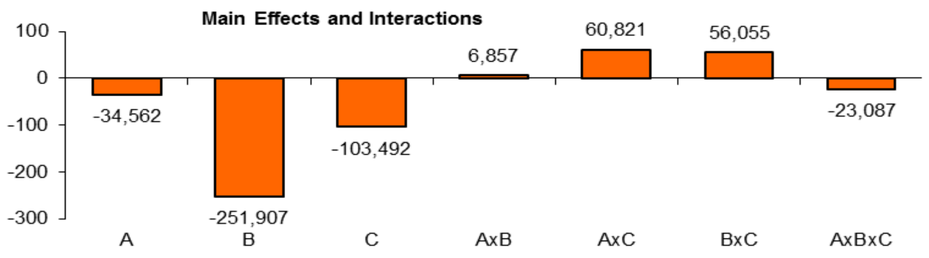


Fig.4 The setting of the three MOVOA parameters

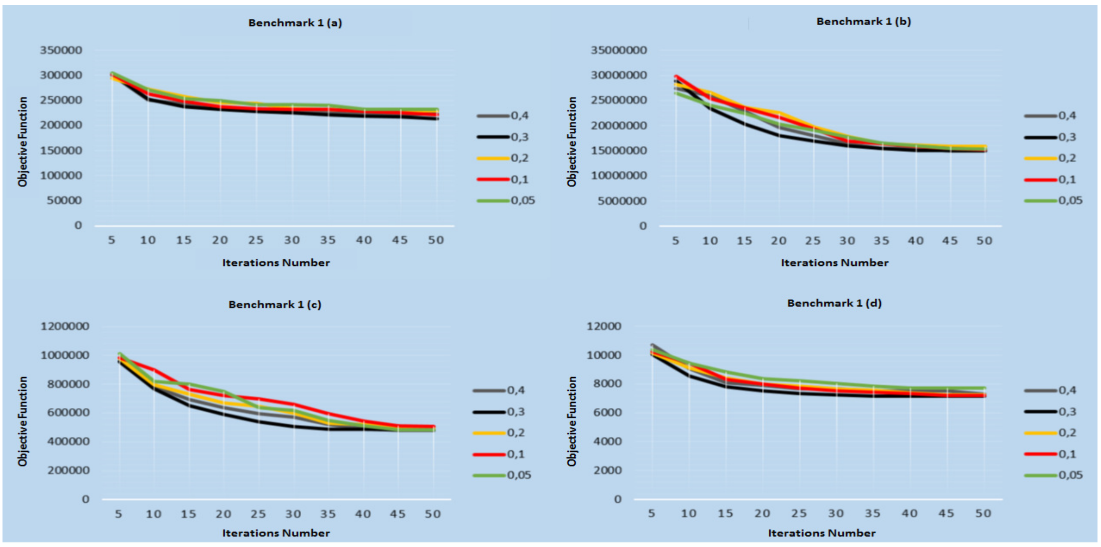
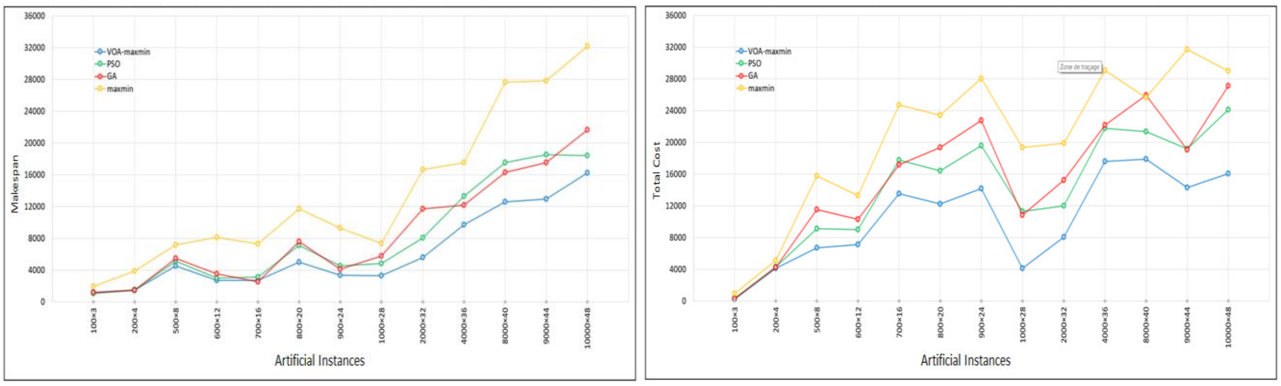


Fig.5 Influence of strong virus rate in the population.

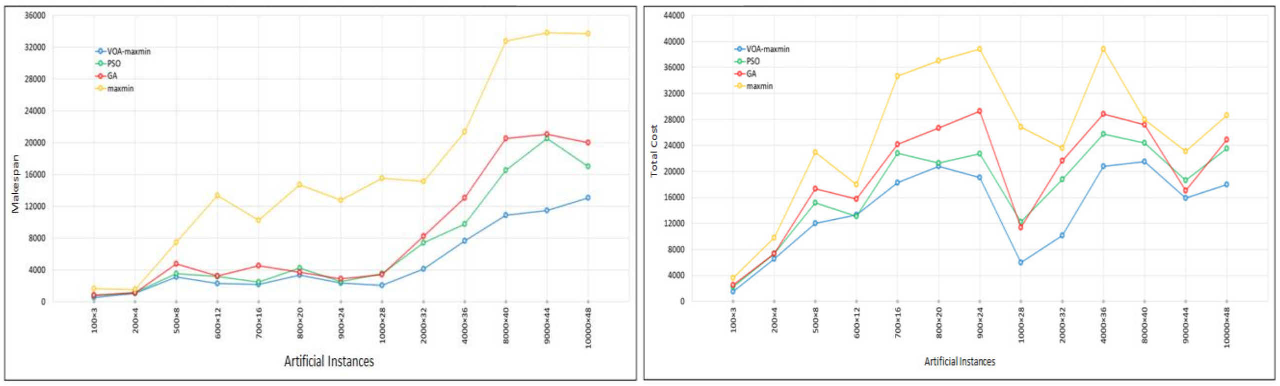
Table 3 Parameter setting values

Parameters	Value
$\boldsymbol{P}_{virusStrong}$	33.33% of classified population
$\boldsymbol{Pop}_{size}$	50 virus
$\boldsymbol{Pop}_{virus}$	Intul
$Iterations_{Max}$	30 Iterations

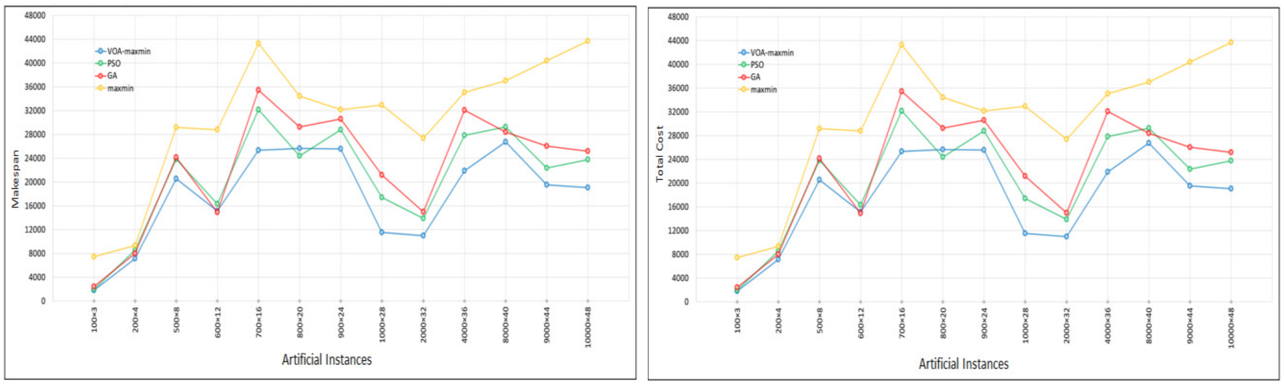
After tuning the parameters, we compare our MOVOA with some evolutionary algorithms with different values of the parameter λ and using a real instances. There are many prior works on multi-objective optimization problems for independent task scheduling using evolutionary techniques. The most popular of meta-heuristic algorithms are Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The comparison results with three competing algorithms (standard Max-Min algorithm and 2 evolutionary algorithms) are shown in Fig. 6. The results show that the MOVOA algorithm outperforms all the algorithms and still provides stable performance when the benchmark functions are subject to a variation of the two-objection linear combination parameter λ. Once again, MOVOA with Min-max initialization not only outperforms these evolutionary algorithms, but also shows outstanding results for all instances when λ=0.75 (Fig. 10(c)). Note that MOVOA performs the best in all of the thirteen instances except for 800×24.



λ=0.25



λ=0.50



λ=0.75

Fig. 6 Comparisons of MOVOA algorithm with Max-Min, GA and PSO algorithms.

VI. CONCLUSION

Scheduling of tasks is one of the most challenging problems in cloud computing environment. The proposed two-objective optimization for tasks scheduling algorithm in CC environment is based on the combination of virus optimization and genetic algorithms. This last algorithm gives a best replication for strong and weak virus in space search. Moreover, Max-Min algorithm reinforces the population initialization process for the proposed approach. The proposed algorithm has been simulated and the results are compared with standard and evolutionary algorithms previously implemented multi objective tasks scheduling algorithm for CC environment. Based on Weighted Sum Method and Pareto-optimal front, the proposed MOVOA showed competitive performance in terms of average two-objective function value, achieving the best results in most of the tested instances and outperforms some standard and evolutionary algorithms. The proposed algorithm can be generalized by taking consideration of some other QoS parameters of the CC environment.

REFERENCES

[1] J. Barbosa and B. Moreira, “Dynamic job scheduling on heterogeneous clusters”, Eighth International Symposium on Parallel and Distributed Computing, 2009.

[2] O.H. Ibarra, C.E. Kim, Heuristic algorithms for scheduling independent tasks on non-identical processors, Journal of the ACM 24 (2) (1977), pp 280–289.

[3] H. Izakian, A. Abraham, V. Snasel, “Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments”, In: IWHGA ’09: Proceedings of the IEEE International Workshop.

[4] S. Tareghian, Z. Bornaee, “Algorithm to improve job scheduling problem in cloud computing environment”, 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI). IEEE, 2015, pp 684-688.

[5] C.Y. Liu, C.M. Zou and P. Wu, “A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing”, International Symposium on Distributed Computing and Applications To Business, Engineering and Science. 2014, pp 68-72.

[6] F. Tao, Y. Feng, L. Zhang, et al., “CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling”. Journal of Applied Soft Computing, 2014, 19(6), pp 264–279.

[7] M. Zhang, Y. Yang, Z. Mi, et al., “An Improved Genetic-Based Approach to Task Scheduling in Inter-cloud Environment[, Ubiquitous Intelligence and Computing”, IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATCScalCom), 2015: 997-1003.

[8] B. Keshanchi, A. Souri, N.J. Navimipour, “An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing”, Journal of Systems and Software, 2017, 124, pp 1-21.

[9] K. Beghdad Bey, F. Benhammadi, M. Y Boudaren and S. Khamadja, “Load Balancing Heuristic for Tasks Scheduling in Cloud Environment”, 19th International Conference on Enterprise Information Systems (ICEIS2017), 26-29 April 2017.

[10] Y.C. Liang and J. R. C. Juarez, “A novel metaheuristic for continuous optimization problems: Virus optimization algorithm”. In: Engineering Optimization 48.1 (2016), pp.73–93.

[11] S. G. Ahmad, C. S. Liew, E. U. Munir, T. F. Ang, S. U. Khan, “A Hybrid Genetic Algorithm for Optimization of Scheduling Workflow”, Applications in Heterogeneous Computing Systems, Vol. 87, January 2016, pp. 80-90.

[12] L. Guo, G. Shao, and S. Zhao, “Multi-objective Task Assignment in cloud computing by Particle Swarm Optimization”. In Proceedings of 8th Int. Conf. on Wireless Communications, Networking and Mobile computing, 2012, pp 1-4.

[13] S. Pandey, L. Wu, S. Guru and R. Buyya, “A particle swarm optimization based heuristic for scheduling workflow applications in cloud computing environments”. 24th IEEE Int’l Conference on Advanced Information Networking and Applications (AINA), Perth, Australia, 2010, pp. 400-407.

[14] W. Shu, W. Wang, Y. Wang, “A Novel Energy-Efficient Resource Allocation Algorithm Based on Immune Clonal Optimization for Green Cloud Computing”. EURASIP Journal on Wireless Communications and Networking, Vol. 64, December 2014.

[15] S. Tayal, “Task Scheduling optimization for the Cloud Computing Systems”, International journal of advanced engineering sciences and technologies, Vol No. 5, Issue No. 2, 201, pp. 111-115.

[16] T. Wang, Z. Liu , Y. Chen, Y. Xu, X. Dai, “Load Balancing Task Scheduling based on Genetic Algorithm in Cloud Computing”, 12th International Conference on Dependable, Autonomic and Secure Computing, IEEE 2014.

[17] H. Chu, “Service Cost of Resource Scheduling in Cloud Computing based on an Improved Algorithm Combining Support Vector Machine with Genetic Algorithm”, International Journal of Grid and Distributed Computing Vol. 9, No. 6 (2016), pp.51-62.

[18] S. Kim, J. Byeon, H. Yu and H. Liu, “Biogeography-Based Optimization for Optimal Job Scheduling in Cloud Computing”, Applied Mathematics and Computation, Elsevier, Vol.247, pp. 266-280, 2014.

[19] A.V Lakra, and D. K Yadav, “Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization”, International Conference on Intelligent Computing, Communication & Convergence, 2015.

[20] A. Narwal and S. Dhingra, “Task Scheduling Algorithm Using Multi-Objective Functions for Cloud Computing Environment”, International journal of control theory and applications, Vol. 10(14), pp. 227-238, 2017.

[21] N. Bansal, A. Maurya, T. Kumar, et al., “Cost performance of QoS Driven task scheduling in cloud computing”. Procedia Computer Science, 57, 2015, pp. 126-130.

[22] M. Abdullahi, M. A. Ngadi, S. M. Abdulhamid, “Symbiotic Organism Search Optimization Based Task Scheduling in Cloud Computing Environment”. Future Generation Computer Systems, Vol. 56, March 2016, pp. 640-650.

[23] Y. Sun, J. White, S. Eade, D. C. Schmidt, “ROAR: AQoS-Oriented Modeling Framework for Automated Cloud Resource Allocation and Optimization”, Journal of Systems and Software, 2015.

[24] D. G. Feitelson and B. Nitzberg, “Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860”, In: workshop on job scheduling strategies for parallel processing, Springer, 1995, pp. 337–360.