Cloudlet Scheduling with Population Based Metaheuristics

Hussein S. Al-Olimat*, Robert C. Green II†, and Mansoor Alam*
*EECS Department, College of Engineering, University of Toledo, Toledo, OH, 2801 W. Bancroft St., Toledo, OH 43606
{Hussein.AlOlimat, Mansoor.Alam2}@utoledo.edu
†Department of Computer Science, Bowling Green State University, Bowling Green, OH 43403,
greenr@bgsu.edu

***Abstract*—Cloud computing is a particularly interesting and truly complex concept of providing services over networks on demand. Many simulation tools have previously been created to simulate the work of the clouds, such as CloudSim. The main use of these tools is evaluation and testing of architectures and services before deployment on network centers and hosts in order to avoid any potential failures or inconveniences. The benefits of using the pay-per-use clouds may be affected by underutilization of the already reserved resources, so the maximization of system utilization while simultaneously minimizing makespan is of great interest to cloud providers wishing to reduce costs. To minimize makespan and increase resource utilization, a hybrid of particle swarm optimization and simulated annealing is implemented inside of CloudSim to advance the work of the already implemented simple broker. The new method maximizes the resource utilization and minimizes the makespan demonstrating improvements upwards of 53%.**

***Keywords*-cloud computing, particle swarm optimization, random inertia weight, task scheduling, makespan, utilization, CloudSim**

I. Introduction

Cloud computing is a term used to describe the on-demand, elastic, and scalable services offered over a network. The two main types of clouds are private and public clouds which are served over private (internal) and public networks, respectively. Hybrid clouds are also available and are a combination of the two previous kinds of clouds, where public clouds are used to increase and supplement the capabilities of the on-premise private clouds. Cloud providers provide cloud resources such as network, servers, storage or applications for users as services where they pay per resource unit used. Companies like Google, Amazon, and Microsoft provide convenient public cloud services over the Internet on a pay-per-use basis.

One of the major focuses of IT professionals has been reducing costs, reducing data center footprint, and improving the amount of computational power available via the cloud. Thus, network resources in cloud computing should be provided to users while satisfying the aforementioned characteristics of the cloud. Further, one of the essential characteristics of the clouds is to give the consumer the illusion that network capabilities and computing power are unlimited, and can be requested at any time and in any quantity as defined by NIST [1].

When considering the cloud, the widely used saying "time is money" applies. When a user requests cloud resources, the cloud should be able to serve the user's request as soon as possible and in a cost-effective manner. In order to satisfy the "unlimited" and "elastic" characteristics of the cloud, consumer requests are often handled by way of the First-Come-First-Serve (FCFS), where the customer request should be the driving factor for workload scheduling [2].

In order to reduce the costs of using on-demand cloud resources, cloud computing systems always attempt to maximize the utilization value of the available resources. To maximize the utilization value, smart and adaptive scheduling algorithms may be used [3]. Designing smart scheduling algorithms is important to overcome problems and constraints when delivering cloud services over the network. One of the more interesting problems arises when many users request many cloud resources at the same time. Such problems can be solved by scheduling cloudlets to the available resources properly or other oversubscription methods. Execution sequences can be found after satisfying a set of objectives, like minimum execution time or minimal cost, or overcoming a set of constraints, such as bandwidth limitation and location of network resources. In order to minimize the cost of using cloud resources, the time of executing all the tasks assigned to various compute resources should be minimized.

Accordingly, this study presents a solution for improving the makespan scheduled tasks and the utilization of cloud resources through the use of a hybrid variant of popular population-based metaheuristic (PBM) algorithm, the particle swarm optimization (PSO). This paper is considered as the first step towards a complete multi-objective optimization of the cloud network. The paper also use a proposed simulated annealing algorithm to enhance the performance of the binary PSO for scheduling. Using the hybrid metaheuristic method was able to enhance the performance of the simulated cloud in CloudSim tool. The test cases in this paper was designed using real world workload, HPC2N [4], and different combinations of cloud resources.

The paper is organized as follows: Section II contains the necessary background regarding scheduling, clouds, and PSO algorithm as used in this study. In section III we discuss the ideas and methods implemented to optimize the makespan time; the results of the implemented methods are

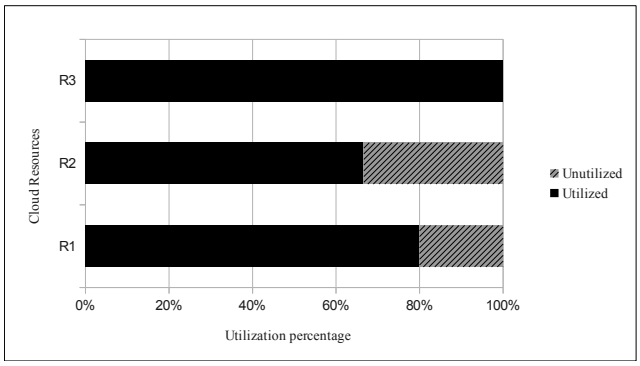

Figure 1. Resource Utilization

examined in section IV, and section V closes the paper.

II. Background

This section will present the necessary background for this study, including coverage of makespan, utilization, and scheduling, CloudSim, Cloudlet Scheduling in CloudSim, PSO, and applications of PSO in cloud scheduling.

A. Makespan, utilization and scheduling

When optimizing scheduling in a system such as a cloud, it is important to consider both makespan and resource utilization. makespan is the total time the network resources take to finish executing all the tasks/jobs. And utilization is the measure of how well the overall capacity of the cloud is used. In order to increase the utilization of the resources at a given time, an effective scheduling algorithm should be used to schedule the tasks among reserved resources. makespan and Utilization have an inverse relationship, i.e. increasing utilization will decrease the makespan. In other words, to achieve higher utilization, tasks should be distributed efficiently among all the reserved cloud resources. As we mentioned before, a more efficient distribution of tasks among the reserved cloud resources will allow us to decrease the makespan, and since we pay per second on the cloud costs of reserving the resources will be decreased by the decrease of the utilization time.

Fig. 1 shows three different resources, with a possibility of having different computing power and different hosts. Consider a makespan time of 500 seconds. During the 500 seconds of executing the tasks on the three resources and according to the current brokering policy, resources R1, R2 and R3 will be utilized during the 500 seconds for around 80%, 66% and 100% respectively.

In order to decrease the makespan of all tasks on the three resources, to be, for example, 450 seconds, rescheduling may be effective in this case, where some of the tasks that were assigned to R3 may be reassigned to either R1 or R2. In this way, the makespan will decrease, and the utilization of the resources will increase for the new duration, and again for the fact that cloud computing is a pay-per-use service, decreasing the makespan and increasing the utilization will certainly decrease the cost of using resources.

B. Cloud computing and CloudSim

1) Network Topology: Cloud simulation tools allow users to test modeled services in a controlled environment with different workloads and scenarios before deploying them on real clouds [5]. CloudSim tool [6] starts by creating a network of nodes as illustrated in Fig. 2. A basic node consists of data centers, hosts, and cloud brokers. Data centers (resource providers in CloudSim) are created first with specifications that define the operating system, memory, bandwidth, storage, etc. One or more hosts are then created on each data center with the proper specification of RAM, storage, bandwidth, processing elements, and the selection of the scheduling algorithm to schedule virtual machines inside the host. Processing elements are known as cores or CPUs, where each processing element is given a defined processing power measured in millions of instructions per second (MIPS). Hosts are managed by data centers where each data center may manage a single or numerous hosts. The cloud broker, "an entity that creates and maintains relationships with multiple cloud service providers" [7], is also created to distribute work among the available data centers or cloud services. A cloud broker is the middleware between a user and the cloud service providers.

After creating all of the network nodes of CloudSim, virtual machines (VM) are created in order to run on the specified hosts. Characteristics of each VM are defined by parameters such as processing power in MIPS, RAM in megabytes, bandwidth, etc. As is illustrated in Fig. 2, a scheduling algorithm should be chosen to schedule cloudlets inside the virtual machine too.

The last step in this process is the generation of tasks/jobs (i.e. cloudlets) either by initializing them through code or from existing workload traces. Cloudlets are defined based on specifications that define the task length in millions of instructions (MI), needed number of processing elements, and a utilization model that states the cloudlet's execution rate through defining the current requested MIPS from the processing elements.

When generating cloudlets from workload traces, the workload format should be checked to make sure that it follows the standard workload format described in [8]. Cloudlets' length should also be converted to MI instead of the standard's execution time in seconds by multiplying the execution time by the execution rate, where the CloudSim authors set a default execution rate value of 1 MIPS. For example, a cloudlet with an execution time of 10 seconds is converted to 10 MI. After creating network nodes, VMs, and cloudlets, the list of available VMs and cloudlets are submitted to cloud brokers.

2) Cloudlet Scheduling: Scheduling in CloudSim is done at the node level, the authors of the tool have already

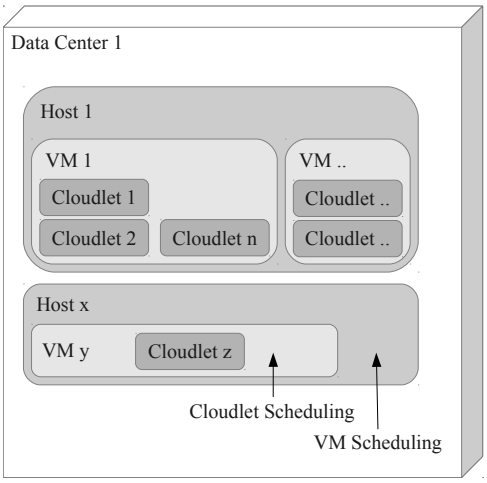

Figure 2. Network Topology in CloudSim Network

implemented two different scheduling algorithms in two different levels of the software: the VM level and the host level as shown in Fig. 2. The provisioning scenarios use space-shared and time-shared policies for VMs and task units [6]. Using the space-shared policy allows one VM or cloudlet to be executed at a given moment of time. On the contrary, using the time-shared policy allows multiple cloudlets to multi-task within a VM and will allow VMs to multi-task and run simultaneously within a host.

By default, VMs will be allocated to the hosts on a FCFS basis as mentioned before and as specified in [6], to give the user the illusion of having unlimited resources even though the Shortest Job First (SJF) policy was found to be faster than FCFS [9] but it wont for sure be in the advantage of the cloud user with larger cloudlets. So FCFS is the ideal policy on the cloud as discussed before. Similarly, cloudlets will be executed on the corresponding VM on FCFS bases, after being allocated to the VMs by the broker.

The simple implemented brokering inside CloudSim will iterate through all cloudlets and assign them to the available VMs one by one. For example if we have 2 VMs and 3 cloudlets, the broker will assign the first VM to run the first cloudlet, the second VM to run the second cloudlet, and then will start again with the first VM by assigning it to run the third cloudlet. Our method works here to allow assigning cloudlets to VMs in a different way, according to their MIPS value to allow some kind of load balancing to decrease the makespan of the whole workload.

C. Particle Swarm Optimization (PSO)

PSO is a population-based search algorithm inspired by bird flocking and fish schooling, where each particle learns from its neighbors and itself during the time it travels in space. There are two versions of PSO. The original PSO was first introduced by Kennedy and Eberhart in 1995 and operates in a continuous space [23]. Later, in 1997 the discrete, binary version of the algorithm was presented also by Kennedy and Eberhart to operate on discrete binary variables [24]. The binary PSO was implemented by [25] to solve the task assignment problem and by [16] and [26] to schedule jobs in grid computing. Both of the methods were successful in finding an optimal solution of the problem.

The PSO algorithm starts by creating a number of particles to form a swarm that travels in the problem space searching for an optimum solution. An objective function should be defined to examine every solution found by each particle throughout the traveling time. A particle in this method is considered as a position in D-dimensional space, where each element can take the binary value of 0 or 1. Additionally, each particle has a D-dimensional velocity, where each element is in the range [0, 1]. Velocities are calculated as probabilities that change during the time particles move in space, where each velocity value changes from one state to another.

The individuals in this algorithm are a group of particles that move through a search space with a given velocity. At each iteration the velocity and position of each particle is stochastically updated by combining the particle's current solution, the particle's personal best solution or $\hat{p_i}$, and the global best solution or $\hat{g}$ over all particles. The required mathematics are listed in (1) and (2) where ω is the inertial constant, c_1 and c_2 represent cognitive and social constants that are usually ~ 2, and r_1 and r_2 are random numbers. In order to update the velocities and positions of each particle, (3) and (4) are used to add nonlinearity where p_i is the i^{th} component of particle p and r is a uniform random number.

$$v_i = \omega v_i + c_1 r_1 \cdot (\hat{p_i} - p_i) + c_2 r_2 \cdot (\hat{g} - p_i) \tag{1}$$

$$p_i = p_i + v_i \tag{2}$$

$$s(p_i) = \frac{1}{1 + \exp(-p_i)} \tag{3}$$

$$p_i = \begin{cases} 0 & s(p_i) \leq r \\ 1 & \text{Otherwise} \end{cases} \tag{4}$$

The four previous equations indicate that the velocity of neighbors and the current velocity of the particle itself will contribute in deciding the next position of the particle. In order for a particle to be part of the swarm and be able to keep up with the other particles during the search of a solution, each particle adapts to the velocity of the swarm as a whole by learning from itself and its neighbor particles.

The inertia weight ω in (1) is one of the most important adjustable parameters in PSO besides the acceleration coefficients and random variables. The inertia weight value can affect the overall performance of the algorithm in finding a potential optimal solution in less computing time. Many techniques are used to choose or modify the value of ω at runtime, such as The fixed inertia weight (FIW) that use a

constant value, linearly decreasing inertia weight (LDIW) implemented by [27] and [28] which changes the value of the inertia weight linearly and per iteration. Also, the adaptive inertia weight (AIW) and random inertia weight (RIW) implemented by [27] and [29], where the inertia weight starts with large value like 0.9 and start decreasing to 0.1.

D. PSO in cloud Scheduling

Workload scheduling is known to be an NP-Complete problem, therefore metaheuristics have been used to solve such problems [10], [11], [12], [13]. The idea behind using metaheuristics is to increase the performance and decrease the computational time to get the job done, in our case metaheuristics are considered the robust solution of finding the right combinations of resources and tasks to minimize the computational expenses, cut costs and provide better services for users. In short, PSO is used to solve the problem shown in Fig. 1.

Abraham et al. [14] discussed the features of GAs, Simulated Annealing(SA), and Tabu Search(TS) as three basic heuristics for grid scheduling. Nevertheless, the most growing and used algorithms in cloud computing are Ant Colony(ACO), Particle Swarm Optimization(PSO) and Genetic algorithms(GAs) [15], therefore other algorithms was not able to grow and compete with them.

According to researchers in [11] and [16], PSO was found to be better than the GA in most of the cases and better than TS in some cases. And according to [17], PSO was found to be faster and more simpler than GA in terms of execution and implementation. PSO is one of the commonly used workflow scheduling algorithms in the cloud [18], where it is mainly used to increase cloud resources utilization which in return will reduce costs and makespans. Zhang et al. in [19] implemented a hierarchical swarms in CloudSim for load balancing and scheduling costs reduction, their method was successful and PSO showed very good results in comparison to the best resource selection algorithm(BRS).

The performance of PSO can be improved with the help of other algorithms or by using an improved version of PSO [20]. For example a hybrid version of PSO and TS was used in CloudSim to maximize the utilization and reduce energy consumption, the method was very successful in reducing the energy consumption by 67.5%. [21]. Less average task execution times also were achieved by a mixed scheduling of PSO and SA than the solution found by the normal PSO, GA, ACO or SA alone [22].

For all the reasons mentioned before and for the fact that PSO implementation is very simple in comparison with other methods, we chose PSO to schedule tasks inside the used simulation tool. A combination of PSO and simulated annealing is used to allow particles to explore more of the problem space and not get stuck in local optima. This paper propose a method to extend the capabilities of the broker, a smarter method than the simple already implemented normal iterative Round-Robin like resource allocation method inside CloudSim.

Table I
COMPLETION-TIME TABLE

	C_1	C_2	C_3	C_4
VM_1	4	6	8	10
VM_2	2.6	4	5.3	6.6
VM_3	1.6	2.4	3.2	4

Table II
EXAMPLE LIST OF CLOUDLETS

Cloudlets	MI
C_1	200
C_2	300
C_3	400
C_4	500
sum	1400

Table III
EXAMPLE LIST OF CLOUD RESOURCES

Resources	MIPS
VM_1	50
VM_2	75
VM_3	125
sum	250

III. IMPLEMENTED METHOD

A. Problem Formulation

Cloud broker assign the cloudlets to the available cloud resources, simply the virtual machines (VMs). The main objective of the proposed solution is to achieve the highest possible utilization of the group of VMs with a minimum makespan. However, the costs of using the resources and the positions of VMs across the system are not considered as parameters in this solution. Additionally, task migration between resources is not allowed which means that every cloudlet can run on one VM until it is done executing.

As an example, Table I shows how many seconds each resource from Table III takes to run each cloudlet from Table II, by dividing the cloudlet length on the processing power of the resource. For example, C_1 will take 4 seconds to run on VM_1, but it takes 2.6 and 1.6 seconds to run on VM_2 and VM_3 respectively.

The objective function of the algorithm used in this study will be to find the shortest time of running the 4 tasks on the available resources, i.e. to minimize the makespan. So the best combination of the pairs of cloudlets on virtual machines should be found. In order to meet the objective of the algorithm and find an optimum solution, (5) is used to calculate the fitness value of each PSO particle. The fitness function in (5) calculates the execution times of all possible cloudlet combinations on every cloud resource and then returns the highest execution time as the fitness value of the particle.

$$Fitness = Max[EXC_{VM_1}(j_1), \dots EXC_{VM_n}(j_m)] \quad (5)$$

```
1: procedure BINARY PSO(nodesList)
2:     CalculateExecTimes();                ▷ as in Table I
3:     initSwarm();
4:     initGlobalBest();
5:     for i ← 0 → numberIterations do
6:         for j ← 0 → numberParticles do
7:             calculateInertiaValue();
8:             calculateNewVelocities();
9:             calculateNewPositions();
10:            calculateFitnessValue();
11:            evaluateSolution();
12:            updateParticleMemory();
13:            updateGlobalBest();
14:        end for
15:    end for
16: end procedure
```

Figure 3. Binary PSO Algorithm

Here, $EXC_{VM_1}(j_1)$ is the execution time of running the set of cloudlets j_1 on VM_1. j_1 is a normal set, e.g. $j_1 = [C_1 + C_2 + ...C_x]$, where x is the number of cloudlets. Also, n and m are the number of VMs and number of possible sets of cloudlets respectively.

B. PSO Algorithm

PSO will initialize a swam of particles where each particle will have a velocity and positions vector, the complete algorithm is shown in Fig. 3. Both of the vectors will look like Table IV, a $3x4$ matrix filled with binary binary values in the case of the positions vector and values in the range of 0 to 1 for the velocity vector. Furthermore, since no task migration is allowed, if we want to run C_1 on VM_2, the second element in the first column of the positions matrix will have the value 1, and the rest of the column values will be 0.

The most optimum solution of the previous example can be found manually due to the small problem space by calculating the fitness value of it using (5) as follows:

$$Fitness = Max[6, 5.4, (1.6 + 4)]$$
$$Fitness = 6$$

The fitness value 6 seconds represents the minimum makespan of running all tasks on all available resources which is the best combination of cloudlets to resources that can be found. So Table IV is actually the positions matrix of the best solution can be found by a particle when solving the previous example using binary PSO. The table clearly defines the relationship between VMs and cloudlets, where C_1, C_2, C_3 and C_4 will be running on VM_1, VM_2 and VM_3 respectively, and each of them will be executed in a FCFS basis as discussed before. On the other hand, if the computing power of resources or the length of the cloudlets were different, a different solution would be found where different VMs will run different cloudlets.

Table IV
CLOUDLET TO RESOURCE MAPPING TABLE

	C_1	C_2	C_3	C_4
VM_1	0	1	0	0
VM_2	0	0	1	0
VM_3	1	0	0	1

C. The Inertia Weight

The inertia wight in 3 is calculated using the proposed RIW method by [30], the method showed an advantage against the other methods mainly in adjusting the balance between particle's local search and global search abilities.

To increase the probability of finding a near-optimal solution in fewer iterations and computing time RIW use a simulated annealing mechanism besides the LDIW method. LDIW was suggested by [31] in 1999, However, it has a disadvantages mainly caused by the low local search ability at the iteration beginning. So even if a particle started at a near point of the global optimization point, the particle will keep moving fast, where sometimes will push the particle away from the point. Similarly, if a particle did not find a close optimal solution and is stuck in one part of the space, and due to the linearly decrease in the inertia weight, the global search ability will decrease, which decreases the chance of finding a better solution. This, in return, makes the iteration forepart more effective in finding the nearest optimal solution. The authors of RIW method combined the linearly decreasing method with a constraint random inertia weight as thoroughly discussed in [30] to overcome the problems of the linear method.

IV. SIMULATIONS AND RESULTS

In the PSO implementation, 100 particles were created where each particle's fitness is evaluated during 1,000 iterations/movements. Additionally, based on a study by Eberhart and Shi [32] that compared inertia weights and constriction factors in particle swarm optimization the values of the acceleration coefficients, C_1 and C_2 in (1), were set to 1.49445 that was found to have better influence on the performance of PSO.

The proposed scheduling method was implemented inside CloudSim as part of the cloud broker. The cloud simulation was implemented as follows: (1) One data center was created with the default characteristics defined by CloudSim authors as in example 6 provided with the source code, (2) two different hosts were created on the data center with: 2 GB ram, 1 TB storage, 10 GB/s bandwidth and time-shared scheduling algorithm was chosen to schedule VMs on hosts. The first host has an Intel Core 2 Extreme X6800 processor

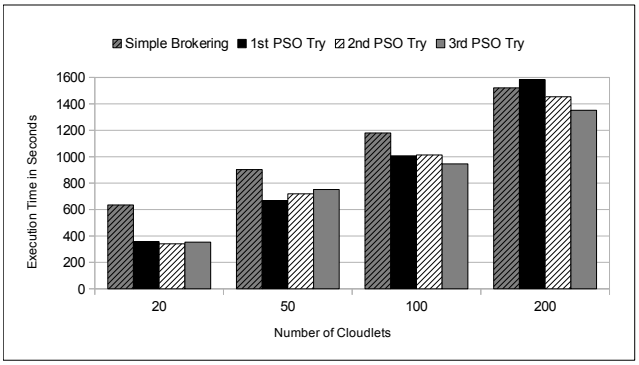

Figure 4. Simulations makespans

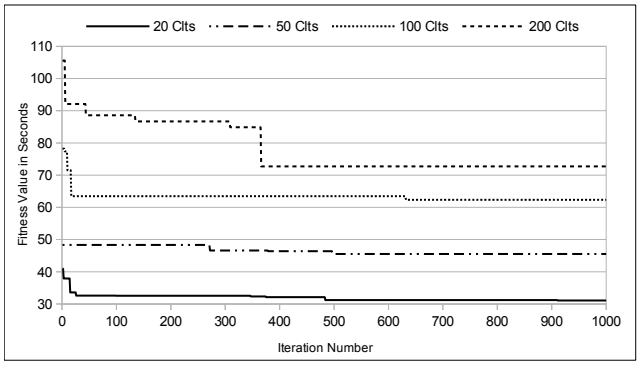

Figure 5. Fitness Value vs. Iteration

Table V
RESULTS OF RUNNING THE PSO METHOD FOR 100 TIMES

Function	20 Clts	50 Clts	100 Clts	200 Clts
AVG	359.15	737.91	1136.88	1670.50
STDEV	22.67	94.58	130.05	163.21
AVG - STDEV	336.48	643.33	1006.83	1507.29
AVG + STDEV	381.82	832.49	1266.93	1833.71

that comes with 2 cores (PEs) and gives a cumulative processing power of 27079 MIPS, as taken from [33]. The second host has an Intel Core i7 Extreme Edition 3960X with 6 cores that gives a cumulative processing power of 177730 MIPS. (3) cloud broker that implements PSO was used, (4) 5 VMs were created, where each has an Intel Pentium 4 Extreme Edition processor with: 10 GB image size, 0.5 GB memory, 1 GB/s bandwidth and 1 processing element that gives 9726 MIPS processing power. Xen virtual machine monitor [34] was also used for all of them, in addition to, using the time-shared scheduling to schedule cloudlets inside the virtual machines, (5) cloudlets were generated from a standard formatted workload of a high performance computing center called HPC2N in Sweden [4]. Each row in the workload represents a cloudlet where we get the id of the cloudlet from the first column, the length of the cloudlet from the fourth column (the runtime value multiplied by the rating which is defined as 1 MI in CloudSim), and finally the number of the requested processing elements from the eighth column.

Fig. 4 shows the time elapsed between submission to completion when executing a group of cloudlets at the same time on the available cloud resources for 4 times, the first time using the simple brokering and the rest using the implemented PSO method. The simulation was developed for 4 different groups of cloudlets, the 1st, 2nd, 3rd and 4th group of cloudlets consisted of 20, 50, 100 and 200 cloudlets respectively. The figure clearly shows that the makespan was minimized when using PSO in most of the cases. The optimization values ranged from 46% to 51% improvement for the 20 cloudlets, 17% to 26% improvement for the 50 cloudlets, 14% to 20% for the 100 cloudlets, and from 11% minimization to -4% maximization of makespan for the 200 cloudlets.

Fig. 5 shows the decrease of the convergence of the global fitness value(the value recorded by the VMs) during the 1,000 iterations of the PSO algorithm. This shows how decreasing the size of the search space and increasing the number of iterations will increase the probability of finding more optimal solutions. Notably, an improvement to the solution for the 20 cloudlets problem was found during the last 100 iterations, with no improvements for approximately the prior 400 iterations. Such findings demonstrate the advantage of using RIW to update the inertia weight value in PSO, which updates the velocity and position of each particle in a way that increased the probability of finding a near-optimal solution even in the later iterations.

Furthermore, the Mann-Whitney test was developed for two different sets, where every value is the makespan of executing a workload of 100 cloudlets. The first numeric set was generated after using the regular CloudSim brokering while the other set was generated after using the metaheuristic method. The two distributions were found to be significantly different with a significance level of $p \leq 0.5$. The regular CloudSim scheduling method was found to have constant fitness values for the same set of cloudlets, in contrast to the metaheuristic method, where fitness values were found to be adaptable and variable along the distribution.

The final test was prepared by running the simulation for 100 times using the PSO method, the average makespans in seconds and standard deviations were then calculated for the same 4 groups of cloudlets the results are listed in Table V. The simple brokering of CloudSim was also tested and the standard deviations were found to be zero as listed in Table VI. The standard deviations were zero due to the fact that the simple broker will always give the same execution sequences and makespans for the same set of cloudlets and cloud resources.

Table V clearly shows that the larger the search space the less reliable the result is, since the result data is widely

Table VI
RESULTS OF RUNNING THE SIMPLE CLOUDSIM BROKERING METHOD

Function	20 Clts	50 Clts	100 Clts	200 Clts
AVG	634.64	902.75	1179.37	1521.06
STDEV	0	0	0	0

spread around the mean. For example, the difference between the best and worst solution found by PSO for the 200 group of cloudlets was approximately 844 seconds. The results mean that the worst solution had approximately 34% more seconds than the best PSO solution. For small populations, like the first three cloudlet groups, the average makespan improvement ranged approximately from 43% to 4%.

V. CONCLUSION

In this study, PSO was used to assign virtual machines to run different cloudlets as part of the broker in CloudSim tool. The results clearly show that PSO was able to minimize the makespan of the workload, and obviously excels at optimizing the simulated scheduling results of CloudSim in a way that will maximize the utilization and minimize the costs of using the on demand cloud services. At the same time PSO, like any other metaheuristic method, does not give any guarantees on finding the most optimal solution. Consequently, and as shown in Fig. 4, whenever the search space expands, the chance of finding an optimal solution becomes harder and harder. However, using the random inertia weight to give the particles the ability to find better solutions during late stages of the search was able to enhance the ability of the PSO method in exploring the problem space.

In some cases the makespan was minimized for about 51% of the original makespan of executing small group of cloudlets when using the simple brokering method. The optimization tend to decrease with the expansion of the search space, but still PSO was able to minimize the makespan by up to 11% for a workload that consists of 200 cloudlets.

In the future, the implementation of a multi-objective solution could further improve the achieved results. The method could, for example, take the costs, bandwidth and locations of data centers into consideration when assigning tasks to cloud resources.

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing-Recommendations of the National Institute of Standards and Technology. NIST," *NIST Special Publication 800-145 The*, pp. 1–3, 2011. [Online]. Available: http://goo.gl/C4OT1

[2] O. Morariu, C. Morariu, and T. Borangiu, "A genetic algorithm for workload scheduling in cloud based e-learning," *Proceedings of the 2nd International Workshop on Cloud Computing Platforms - CloudCP '12*, pp. 1–6, 2012. [Online]. Available: http://goo.gl/7E4SI

[3] S. Tayal, "Tasks scheduling optimization for the cloud computing systems," *INTERNATIONAL JOURNAL OF ADVANCED ENGINEERING SCIENCES AND TECHNOLOGIES*, vol. 5, no. 2, pp. 111–115, 2011. [Online]. Available: http://goo.gl/szKF6

[4] U. University, "The HPC2N Seth log," 2006. [Online]. Available: http://goo.gl/wrxAK

[5] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise Grids and clouds," *2009 10th IEEE/ACM International Conference on Grid Computing*, pp. 50–57, Oct. 2009. [Online]. Available: http://goo.gl/1uzwC

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, and C. A. F. De Rose, "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper. 2011 - Wiley Online Library*, vol. 41, no. August 2010, pp. 23–50, 2010. [Online]. Available: http://goo.gl/Y886a

[7] N. H. Centers, "Getting Familiar with Cloud Terminology Cloud Dictionary," vol. 1, no. 1, pp. 1–7, 2013. [Online]. Available: http://goo.gl/ODFvc

[8] S. Chapin, W. Cirne, and D. Feitelson, "Benchmarks and standards for the evaluation of parallel job schedulers," *... Strategies for Parallel ...*, pp. 1–24, 1999. [Online]. Available: http://goo.gl/CtnDb

[9] Y. Cao, C. Ro, and J. Yin, "Comparison of Job Scheduling Policies in Cloud Computing," *Future Information Communication Technology and Applications*, vol. 235, pp. 81–87, 2013. [Online]. Available: http://goo.gl/Ed6CR

[10] E. Mokoto, "Scheduling to minimize the makespan on identical parallel Machines: an LP-based algorithm," *Investigacion Operative*, pp. 97–107, 1999. [Online]. Available: http://goo.gl/qnlSi

[11] P. Pongchairerks, "Particle swarm optimization algorithm applied to scheduling problems," *ScienceAsia*, vol. 35, pp. 89–94, 2009. [Online]. Available: http://goo.gl/Hl5OT

[12] W.-n. Chen and J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNATICS*, vol. 39, no. 1, pp. 29–43, 2009. [Online]. Available: http://goo.gl/f0P8xR

[13] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A Particle Swarm Optimization (PSO) -based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 400 – 407. [Online]. Available: http://goo.gl/uMztxi

[14] A. Abraham, R. Buyya, and B. Nath, "Nature's Heuristics for Scheduling Jobs on Computational Grids," *IEEE INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND COMMUNICATIONS*, pp. 45–52, 2000. [Online]. Available: http://goo.gl/n0SQu

[15] M. Rana, S. K. KS, and N. Jaisankar, "Comparison of Probabilistic Optimization Algorithms for Resource Scheduling in Cloud Computing Environment," *International Journal of Engineering and ...*, vol. 5, no. 2, pp. 1419–1427, 2013. [Online]. Available: http://goo.gl/B8I1w

[16] L. Zhang, Y. Chen, and R. Sun, "A task scheduling algorithm based on pso for grid computing," *International Journal of Computational Intelligence Research.*, vol. 4, no. 1, pp. 37–43, 2008. [Online]. Available: http://goo.gl/05eD3

[17] S. Mirzayi and V. Rafe, "A survey on heuristic task scheduling on distributed systems," *AWERProcedia Information Technology & Computer Science*, vol. 1, pp. 1498–1501, 2013. [Online]. Available: http://goo.gl/Jak6i

[18] A. Bardsiri and S. Hashemi, "A Review of Workflow Scheduling in Cloud Computing Environment," *International Journal of Computer Science and Management Research*, vol. 1, no. 3, pp. 348–351, 2012. [Online]. Available: http://goo.gl/0eK1Z

[19] H. Zhang, P. Li, Z. Zhou, and X. Yu, "A PSO-Based Hierarchical Resource Scheduling Strategy on Cloud Computing," *Trustworthy Computing and Services*, pp. 325–332, 2013. [Online]. Available: http://goo.gl/HV8yo

[20] Y. Wang, J. Wang, C. Wang, and X. Song, "Research on Resource Scheduling of Cloud Based on Improved Particle Swarm Optimization Algorithm," *Advances in Brain Inspired Cognitive ...*, pp. 118–125, 2013. [Online]. Available: http://goo.gl/yZzz8

[21] Z. Wang, K. Shuang, L. Yang, and F. Yang, "Energy-aware and revenue-enhancing Combinatorial Scheduling in Virtualized of Cloud Datacenter," *Journal of Convergence Information Technology*, vol. 7, no. 1, pp. 62–70, Jan. 2012. [Online]. Available: http://goo.gl/Lq3LV

[22] S. Zhan and H. Huo, "Improved PSO-based Task Scheduling Algorithm in Cloud Computing," *Journal of Information & Computational Science*, vol. 13, pp. 3821–3829, 2012. [Online]. Available: http://goo.gl/mw4JA

[23] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995. [Online]. Available: http://goo.gl/Srnox

[24] J. Kennedy and R. C. Eberhart, "A DISCRETE BINARY VERSION OF THE PARTICLE SWARM ALGORITHM," *IEEE international conference on Systems, Man, and Cybernetics*, pp. 4104 – 4108, 1997. [Online]. Available: http://goo.gl/rwSjO

[25] J. L. Pierobom, M. R. Delgado, and C. A. Kaestner, "Particle swarm optimization for task assignment problem," *10th Brazilian Congress on Computational Intelligence (CBIC2011)*, vol. 10, pp. 1–8, 2011. [Online]. Available: http://goo.gl/7o88a

[26] H. Izakian, B. T. Ladani, K. Zamanifar, and A. Abraham, "A novel particle swarm optimization approach for grid job scheduling," *Information Systems, Technology and Management - Communications in Computer and Information Science*, vol. 31, pp. 100–109, 2009. [Online]. Available: http://goo.gl/gFGCv

[27] K. Deep and Madhuri, "Application of Globally Adaptive Inertia Weight PSO to Lennard-Jones Problem," *... on Soft Computing for Problem Solving (SocProS 2011 ...*, pp. 31–38, 2012. [Online]. Available: http://goo.gl/fviMF

[28] S. Sivanandam and P. Visalakshi, "Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia," *International Journal of Computer Science & Applications*, vol. 4, no. 3, pp. 95–106, 2007. [Online]. Available: http://goo.gl/52UXL

[29] R. Ojha and M. Das, "An Adaptive Approach for Modifying Inertia Weight using Particle Swarm Optimisation," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 105–112, 2012. [Online]. Available: http://goo.gl/IAEnB

[30] L. Chong-min, G. Yue-lin, and D. Yu-hong, "A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy," *Journal of Communication and Computer*, vol. 5, no. 11, pp. 42–48, 2008. [Online]. Available: http://goo.gl/QPtsH

[31] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," *..., 1999. CEC 99. Proceedings of the 1999 ...*, pp. 1945–1950, 1999. [Online]. Available: http://goo.gl/h3oH2

[32] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, no. 7, pp. 84–88, 2000. [Online]. Available: http://goo.gl/YJbz23

[33] the free encyclopedia Wikipedia, "Instructions per second," p. 1, 2013. [Online]. Available: http://goo.gl/YYuN

[34] P. Barham, B. Dragovic, and K. Fraser, "Xen and the art of virtualization," *ACM SIGOPS ...*, 2003. [Online]. Available: http://goo.gl/Yc9p3