Makespan Improvement of PSO-based Dynamic Scheduling in Cloud Environment

Azade Khalili
School of Electrical and Computer Engineering
Kashan University
khalili91@grad.kashanu.ac.ir

Seyed Morteza Babamir
School of Electrical and Computer Engineering
Kashan University
babamir@kashanu.ac.ir

***Abstract*—the latest generation of distributed systems is cloud computing that has been acclaimed scientifically and commercially. CloudSim is one of the simulation tools that enables the evaluation and testing cloud services and infrastructures before development on a real cloud. For optimal use of the cloud's potential power, efficient and effective scheduling algorithms are required which can select the best resources for execution tasks. The matter of mapping and scheduling the tasks is assigning tasks to run on the existing resources in the manner that helps to maximize utilization and minimize makespan. The total time that is needed for all tasks/jobs to be finished is known as makespan. And utilization is the measure of how well the overall capacity of the cloud (network resources) is used. Due to the heterogeneity and dynamic resources, task scheduling is known as NP-complete problem and metaheuristics are needed to find the best scheduling combination. The main objective of this paper is to optimize task scheduling that uses the particle swarm optimization algorithm to minimize the makespan. Different inertia weights have been used. The Linear Descending Inertia Weight (LDIW) with an average 22.7% reduction in makespan shows the best performance.**

Keywords- cloud computing, Particle Swarm Optimization (PSO), inertia weight, task scheduling, utilization, makespan, CloudSim, cloudlet

I. INTRODUCTION

Cloud computing has emerged as a latest generation of distributed systems that provides Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). There are various types of clouds but the popular category has three parts, public clouds, private clouds and hybrid clouds. Although private clouds can be called internal clouds which that offer a series of activities and operations as a service, but these clouds are expanded across an intranet or an enterprise datacenter. Public clouds provide services dynamically on the internet that is available for public. When a cloud provider uses public cloud resources to create a private cloud, the result is a virtual private cloud. Hybrid clouds are a composition of several private and public clouds. According to [1], a cloud service has three distinct characteristics that differentiate it from traditional hosting:

- Services are sold on demand basis as consumers have to pay for minute or hour basis.
- Elasticity is another important feature, as user uses the services as much little as he wants .
- The providers are fully responsible to manage services (the consumer needs nothing but a personal computer and Internet access).

One of the major activities performed in distributed systems is task scheduling that affects reliability and flexibility. The goal of scheduling algorithms is spreading the tasks on resources and approaching to maximize utilization, minimize makespan and minimize execution cost. In this paper, we focus on minimizing makespan and maximizing utilization.

Scheduling algorithms will fall into two main categories: static and dynamic scheduling algorithms. In dynamic scheduling, before starting the program, there is no plan to execute tasks on the resources and all decisions are made at running time. In static scheduling, scheduling is done before starting the program so the resources which tasks must be performed on them, would be predefined [2]. Dynamic scheduling algorithms in cloud computing environment are grouped in two main categories: batch mode heuristic scheduling algorithms (BMHA) and online mode heuristic algorithms. In BMHA, when Jobs arrive in the system, they are queued and collected into a set and after a period of time the scheduling algorithm will start. The main example for BMHA is FCFS algorithm. In On-line mode heuristic scheduling algorithm, when jobs arrive in the system, they are scheduled [3].

In [4], Al-Olimat et al. have used PSO for scheduling cloudlets in CloudSim. They used RIW (Random inertia Weigh) [5] in PSO algorithm and reached an average 13.86% improvement reduction in makespan. This paper has been done based on [4], considering all of its conditions. However five inertia weights within the reasonable computation time more than RIW are used. Because in dynamic environments, resources may change and become unavailable during the calculation, scheduling algorithms should not take too long. Actually our main contribution in this paper is offering different inertia weights to improve an average makespan reduction.

PSO is one of the most popular algorithms in optimization that was presented by Kennedy and Eberhart in 1995 [6]. Mainly PSO was used to solve continuous problems. However in 1997 for discrete problems, the binary version of this algorithm was presented by Kennedy and Eberhart [7].

The rest of this paper is organized as follows. Section II presents the related work on PSO in cloud computing environment. PSO and inertia weight strategies, CloudSim toolkit and cloud modeling are explained in Section III. Section IV gives the description of problem formulation.

978-1-4799-1972-7/15/$31.00 ©2015 IEEE

Section V shows the simulation and experimental results. Finally, section VI addresses the conclusion.

II. Related Work

Task scheduling is known as NP-complete problem and metaheuristics are needed to find the best solution. One of the most popular heuristic algorithms is PSO that is similar to other population-based algorithms like Genetic algorithms but PSO is easy to understand and has simple implement and good convergence rate. Some papers use PSO for load balancing, reducing makespan and increasing utilization [1, 4, 8 and 9] and some papers use it for cost optimization [10, 11 and 12].

Paper [1] offers a new method in CloudSim that is searching cloudlets rather than doing the search in particles. The target is kept constant and by changing in velocity, new particle id can be obtained. In [4], PSO with random inertia weight (RIW) is used to decrease makespan and increase utilization. An average 13.86% improvement reduction in makespan has been obtained. In [8], a dynamic distributed system based on PSO algorithm is designed in cloud computing environments with the aim of balanced distribution, improve the utilization rate of resources and speed up the system. The efficiency of smallest position value (SPV) technique in the mapping task to resources in continuous version of PSO algorithm is investigated in [9] and has resulted in efficient load distribution.

Paper [10] introduces a cost model for the system that can be used to decide purchasing options (on-demand instance or reserved instance) for cloud consumers. The budget can also be estimated for the execution of a large scale scientific workflow. In [11] [12], PSO algorithm is used to schedule applications to cloud resources and minimize the total cost of execution. In experiments both computation cost and data transmission cost are considered.

III. Primitive Definition

A. Particle Swarm Optimization

PSO is a swarm intelligence technique that inspired by the social behavior of a bird flock and a fish school. This algorithm is a population-based approach and started by generating particles randomly. Each particle is a candidate solution that moves in the search space of an optimization problem. Over a number of iterations, particles adjusted their position to reach the position that is closest to the target. In each iteration, particles' velocity and position are updated based on the best position of each particles (pbest or $\hat{p}$), and the best position of all the particles (gbest or $\hat{g}$). The fitness function is required to evaluate the particles, which is problem specific. Each particle is dependent on both the individual and community movement. The combination of these two movements, leads to an efficient model to find the best spot that is an objective of an optimization problem.

In each iteration, formulas (1) and (2) will be used to update the velocity and the position of particles, respectively.

$$v_i = wv_i + c_1 r_1.(\hat{p} - p_i) + c_2 r_2.(\hat{g} p_i) \quad (1)$$

$$p_i = v_i + p_i \quad (2)$$

Where p_i is the i^{th} element of the particle p , w is the inertia weight, *c1* and *c2* represent cognitive and social coefficients that are usually □ 2, *r1* and *r2* are random numbers between 0 and 1. Furthermore, formulas (3) and (4) are used to add nonlinearity in order to update the velocity and position of each particle.

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

$$p_i = \begin{cases} 0, & s(p_i) \le r \\ 1, & otherwise \end{cases} \quad (4)$$

Where *r* is a random number between 0 and 1.

1) Inertia Weight

Inertia weight is the most important control parameters of PSO algorithm, because it can adjust effectively global and local search capabilities algorithm. Many researches have been done into an inertia weight selection strategy to achieve the best balance between global and local search.

In [13] the strategy of random inertia weight is expressed by the following formula, called SRIW:

$$w = 0.5 + 0.5 * rand() \quad (5)$$

Where *rand()* is a random number between 0 and 1.

Paper [14] calculates the Linear Descending Inertia Weight (LDIW) by the following formula:

$$w = (w_1 - w_2) * \frac{MAXiter - iter}{MAXiter} + w2 \quad (6)$$

Where w_1 and w_2 are the initial and final values of inertia weight that set to 0.4 and 0.9, respectively.

In [15] according the formulas that have been introduced in [13] and [14], the chaotic inertia weights are introduced as follows:

Below shows the steps are considered for the Chaotic Random Inertia Weight (CRIW):

Step1. Select a random number for z between [0 ,1]

*Step2. Make Logistic mapping: z=4*z*(1−z)*

Step3. $w = 0.5 * z + 0.5 * rand()$ (7)

And below shows the steps are considered for Chaotic Descending Inertia Weight (CDIW):

Step1.Select a random number for z between [0 ,1]

*Step2.Make Logistic mapping z=4*z*(1−z)*

Step3. $w = (w_1 - w_2) * \frac{MAXiter - iter}{MAXiter} + (w2 * z)$ (8)

Where w_1 and w_2 are the initial and final values of inertia weight that set to 0.4 and 0.9.

The strategies of Adaptive Inertia Weight (AIW) monitor search space and use the feedback from one or more parameters to adjust the value of the inertia weight [16]. The following method is used to calculate the inertia weight:

$$S(i,t) = \begin{cases} 1, & f(Pbest_t^i) < f(Pbest_{t-1}^i) \\ 0, & f(Pbest_t^i) \ge f(Pbest_{t-1}^i) \end{cases}$$

$$P_s(t) = \frac{\sum_{i=0}^{n} S(i,t)}{n}$$

$$w = (w1 - w2)P_s(t) + w2 \quad (9)$$

In other word, if the fitness function value of current position ($f(Pbest_t^i)$) is better than the last one ($f(Pbest_{t-1}^i)$), the value of $S(i,t)$ becomes 1 and 0 in otherwise. W_1 set 0.4.

Briefly, Table 1 shows inertia weight strategies which mentioned in Section III, part A. Two values, "2 and 1.49455" are assumed for c1 and c2. Value "2" is the maximum value for coefficients that was introduced in basic PSO algorithm and "1.49455" is the value used in [4]. For each strategy, two values for c1 and c2 are tested and the value that has better performance is reported in Table 1.

B. CloudSim

CloudSim is a framework for modeling and simulation of cloud computing services and infrastructures. The main objective of CloudSim project is to provide a generalized and extensible simulation framework that enables modeling, simulation and testing of emerging cloud computing applications and infrastructure services [17].

Figure 1 shows the multi-layered design of the CloudSim architecture. The highest layer is User code that exposes basic entities for hosts, applications, VMs, the number of users and broker scheduling policies. The second layer is CloudSim layer that provides modeling and simulation cloud-based datacenter and the third layer is simulation engine [18].

1) Cloud Modeling

Cloud infrastructure-level services can be simulated by datacenter entity of CloudSim. A datacenter can manage multiple hosts that in turn they manage multiple VMs during their life cycle. A host is a CloudSim component implements a physical computing server in cloud computing which is assigned to processing power (expressed in Million Instructions per Second -MIPS), memory, storage and allocation policy for assigning processing element to VMs. A

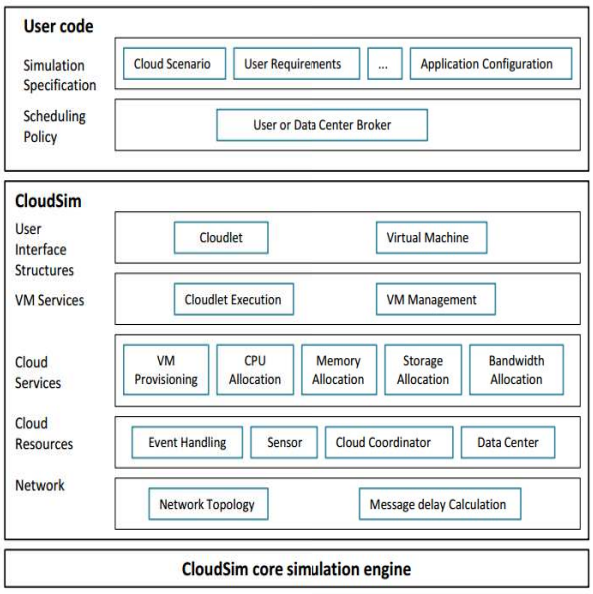

Figure 1 multi-layer design of Cloudsim component[1]

VM is defined by processing power in MIPS, RAM, bandwidth, etc.

A default policy to allocate VMs to the hosts is based on First-Come-First-Serve (FCFS). CloudSim provides possibility to implement new policies to allocate VMs to the hosts in CloudSim layer (Fig. 1). In this layer, a cloud host can be concurrently allocated to a set of VMs which execute applications. For modeling software applications, CloudSim uses cloudlet entity defined by characteristics such as length in Million Instructions (MI), number of processing element required to execution and etc. Cloud broker entity is a middleware between the user and cloud provider that assigns cloudlets to VMs. By default, the assignment is done based on FCFS.

Table 1. SUMMARY OF INERTIA WEIGHT STRATEGIES CONSIDERED IN THIS PAPAER

Inertia weight strategy	Inertia weight formula	coefficients	reference
Simple Random Inertia Weight (SRIW)	$w = 0.5 + 0.5 * rand()$	c1=c2=2	[12]
Linear Descending Inertia Weight (LDIW)	$w = (w_1 - w_2) * \frac{MAXiter - iter}{MAXiter} + w2$	c1=c2=1.49455	[13]
Chaotic Random Inertia Weight (CRIW)	$w = 0.5 * z + 0.5 * rand()$	c1=c2=2	[14]
Chaotic Descending Inertia Weight (CDIW)	$w = (w_1 - w_2) * \frac{MAXiter - iter}{MAXiter} + (w2 * z)$	c1=c2=2	[14]
Adaptive Inertia Weight (AIW)	$w = (w1 - w2)P_s(t) + w2$	c1=c2=1.49455	[15]

IV. PROBLEM FORMULATION

Cloud broker assigns cloudlets to available VMs. The main objective of this paper is using PSO to find the optimal assignment order of cloudlets to VMs that minimize makespan and maximize utilization.

The complete algorithm of PSO is given in Figure 2. In this algorithm, first, the execution time is calculated (step 2). To obtain execution time of each cloudlet on VMs (in seconds), MI (Million Instructions) of cloudlet is divided into MIPS (Million Instructions per Second) of VM. As an example, four cloudlets with their MI value and three VMs with their MIPS value are shown in Table 2 and 3 respectively. Table 4 shows the execution time.

The particles are initialized in step 3. Each particle has a velocity and position vector. Position vector is filled with binary value and velocity vector is filled with values in the range of (0, 1). In fact, value 1 in position vector shows the cloudlet execution on VM. For example, if C_1 is executed on VM_3, the third element of the first column of the position vector will be 1 and other elements of that column will be 0 (cloudlet migration is impossible) and velocity vector represents the possibilities. Entries in each column of the position vector corresponding to the entry with the highest value in column in velocity vector will be 1.

In each iteration, the inertia weight value is calculated (step 6) and the velocity and position vector are updated for each particle (steps 8 and 9) based on formula (1) and (2), respectively. Then the Fitness function introduced in [4], Formula 10, evaluated the solution that is found by particle and updated the particle's pbest (steps 10, 11 and 12). The Fitness function evaluates execution time of all cloudlets on each cloud resources (VMs) and returns highest execution time as a Fitness value of each particle.

$$Fitness=max[EXC_{VM1}(j1)\ldots EXC_{VMn}(jm)] \quad (10)$$

Where $EXC_{\mathrm{VM1}}(\mathrm{j1})$ is execution time of cloudlet set j1 on VM, j_1 is a normal set, e.g. j_1= [C1 + C2+ … +Cx] where x is number of cloudlet, n and m are number of VMs and number of possible cloudlet sets, respectively.

1. **Procedure** PSO(ParticleList)
2. CalculateExecTime();
3. initSwarm();
4. initGlobalBest();
5. **for** i= 0 to number of Iterations **do**
6. calculateInertiaWeight();
7. **for** j=0 to number of Particles **do**
8. calculateNewVelocity();
9. calculateNewPosition();
10. calculateFitnessValue();
11. EvaluateSolution();
12. updateParticleMemory();
13. updateGlobalBest();
14. **end for**
15. **end for**
16. **end procedure**

Figure 2.PSO Algorithm

Table 2. LIST OF CLOUDLETS

cloudlets	MI
C1	200
C2	250
C3	300
C4	350

Table 3. LIST OF VMs

VMs	MIPS
VM1	50
VM2	100
VM3	150

Table 4. EXECUTION TIME BY SECOND

	C1	C2	C3	C4
VM1	4	5	6	7
VM2	2	2.5	3	3.5
VM3	1.33	1.67	2	2.33

Table 5. ASSIGNING CLOUDLETS TO VMs BASED ON FCFS

	C1	C2	C3	C4
VM1	1	0	0	1
VM2	0	1	0	0
VM3	0	0	1	0

Table 6. ASSIGNING CLOUDLETS TO VMs BASED ON PSO

	C1	C2	C3	C4
VM1	1	0	0	0
VM2	0	0	0	1
VM3	0	1	1	0

By default in CloudSim, cloud broker assigns cloudlets to VMs based on FCFS (Table 5) and the makespan is 4+7=11 seconds. Because of concurrency, VM, in a maximum execution time, determines the makespan. In the example, since VM1, VM2 and VM3 take along 11, 2.5 and 2 seconds to finish their cloudlets respectively, VM1 determines the makespan.

Using PSO, we want to find the optimal sequence that minimizes the makespan; Table 6 is a solution obtained by a particle. Based on Formula 10, the Fitness function value is 4 (*Fitness = max* [4, 3.5, 3.67]). At the end of the iteration, the lowest value of the Fitness function is assigned to the best global value (step 13).

V. SIMULATION AND RESULTS

The conditions of this paper are as follows: The Fitness of each particle is calculated over 1000 iterations. The number of particles is 100. One datacenter is created with default properties defined by CloudSim author. Datacenters are the resource providers in CloudSim so at least one of them is

needed to run a CloudSim simulation. Two different hosts with: 2 GB RAM 1 TB storage, 10 GB /s bandwidth and time-shared scheduling algorithm is chosen to schedule VMs on hosts.

In CloudSim there are two policies for execution cloudlets on VMs or VMs on hosts, they are space-shared and time-shared policy. Space-shared policy allows cloudlets on VMs or VMs on hosts to be executed at a time slice. Time-shared policy allows multiple cloudlets within a VM or VMs within a host to be multi-task and run simultaneously.

One of the hosts has 2 cores (PE) and gives a cumulative processing power of 27,079 MIPS and other host has 6-cores and gives a cumulative processing power of 177,730 MIPS. 5 VMs defined with specification: 10 GB image size, 0.5 GB memory, 1 GB/s bandwidth and 1 processing element that gives 9726 MIPS processing power. Cloudlets are generated from a standard formatted workload of a high performance computing center called HPC2N [19]. One broker is created that assigns cloudlets to VMs based on PSO.

The program of PSO-based broker with each inertia weight strategy is run three times. The average of simulation makespan of each strategy and simulation makespan based on FCFS are shown in Figure 3.

The results show that by using five proposed inertia weight strategies, makespan is less than FCFS. Whereas using RIW when the number of cloudlets is equal to 200, makespan is worse than FCFS.

Figure 4 represents the improvement percentage which is obtained by using each strategy. When the number of cloudlets is equal to 20 the RIW with 43.4 % has the best performance. When cloudlets' number is equal to 50, CRIW with 30.45% has the best performance. For a great number of cloudlets, when it is equal to 100 LDIW with 21.58 % and when it is

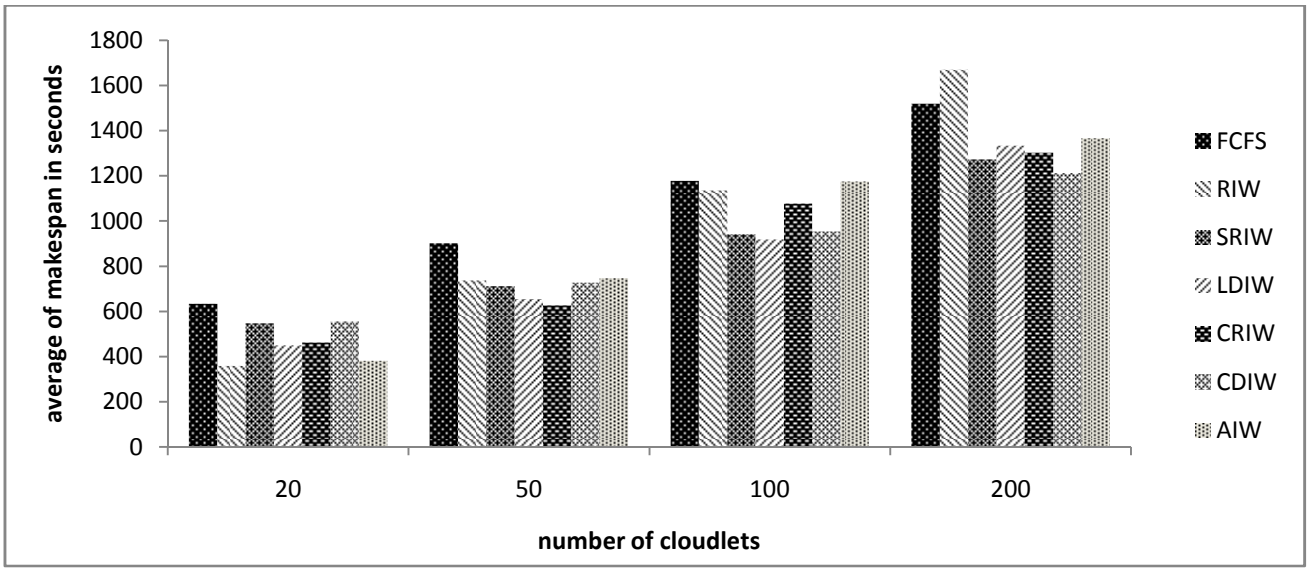

Figure 3. Simulation makespans

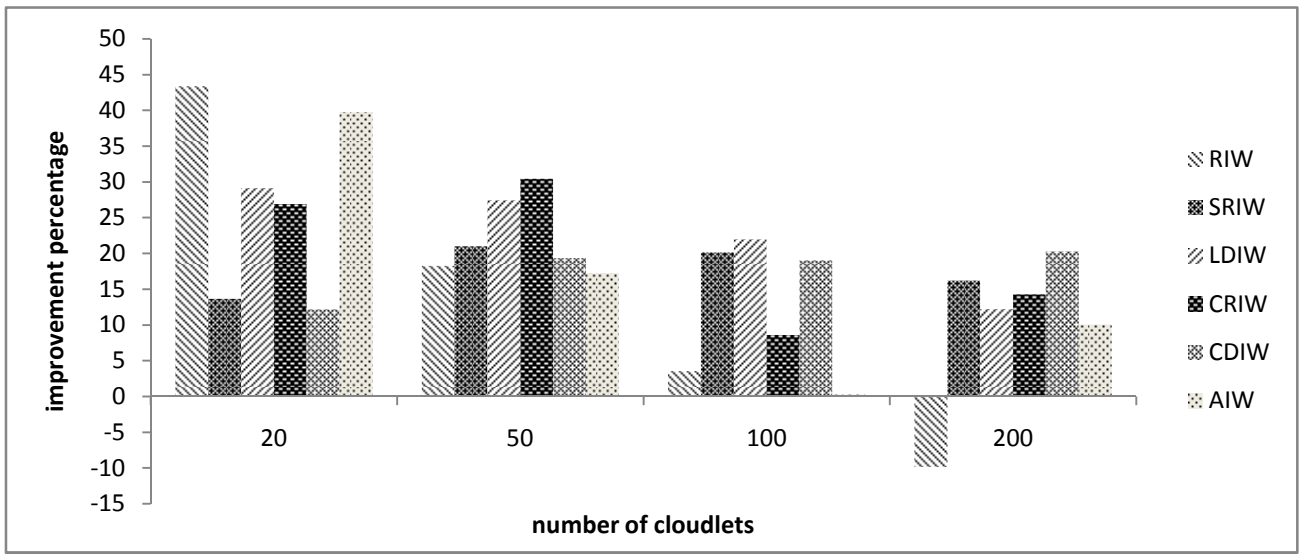

Figure 4. Improvement percentage

equal to 200 CDIW with 20.27% have the best performance.

Table 7 shows the average of improvement percentage, as you can see the best improvement percentage is related to LDIW strategy with 22.7%.

Table 7. AVERAGE OF IMPROVEMENT PERCENTAGE

Inertia Weight Strategy	Average of Improvement Percentage
RIW	13.86
SRIW	17.51
CRIW	20.07
LDIW	22.7
CDIW	17.72
AIW	16.84

VI. CONCLUSION

This paper used PSO for allocating cloudlets to VMs in CloudSim simulation environment. In implementing of the algorithm, five different inertia weight strategies have been used. The experiment shows PSO algorithm with LDIW strategy introduced in [14], has improved makespan with an average of 22.7% compared to FCFS mode.

REFRENCES

[1] Kavita Bhatt, Mahesh Bundele, "CloudSim Estimation of a Simple Particle Swarm Algorithm", *Advanced Research in Computer Science and Software Engineering,* 2013.

[2] Mocanu, E. M., Florea, M., Andreica, M. I., & Tapus, N., " Cloud computing—task scheduling based on genetic algorithms," in *In Systems Conference (SysCon)*, 2012.

[3] Salot, Pinal, "A Survey Of Various Scheduling Algorithm In Cloud Computing Environment," *International Journal of Research in Engineering& Technology (IJRET),* pp. 131-135, 2013.

[4] Hussein S. Al-Olimat, Robert C. Green , and Mansoor Alam, "Cloudlet Scheduling with Population Based Metaheuristics," in *SummerSim*, 2014.

[5] L. Chong-min, G. Yue-lin, and D. Yu-hong, "A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy," *Communication and Computer,* 2008.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, 1995.

[7] Kennedy, J., & Eberhart, R. C., "Particle swarm algorithm," in *International Conference on Systems, Man, and Cybernetics, Orlando, FL*, 1997.

[8] Hongwei Zhao, Wang Chenyu , "A Dynamic Dispatching Method of Resource based on Particle swarm optimization for Cloud Computing Environment," in *Web Information System and Application Conference (WISA)*, 2013.

[9] Sidhu, Manitpal S., Parimala Thulasiraman, and Ruppa K. Thulasiram, "A load-rebalance PSO heuristic for task matching in heterogeneous computing systems," in *Swarm Intelligence (SIS)*, 2013.

[10] Netjinda, Nuttapong, Booncharoen Sirinaovakul, and Tiranee Achalakul, "Cost optimization in cloud provisioning using particle swarm optimization," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2012.

[11] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, Rajkumar Buyya, "A Particle Swarm Optimization based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in *International Conference on Advanced Information Networking and Applications*, 2010.

[12] Zhangjun Wu, Zhiwei Ni, Lichuan Gu, Xiao Liu, "A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling," in *International Conference on Computational Intelligence and Security*, 2010.

[13] Eberhart, Russ C., and Yuhui Shi., "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceeding Congress on Evolutionary Computation*, 2000.

[14] Y. H. Shi, R. C. Eberhart, "Empirical Study of Particle SwarmOptimization," in *Proceeding Congresson EvolutionaryComputation*, 1999.

[15] Yong Feng, Yong-Mei Yao, Ai-Xin Wang, " Comparing with chaotic inertia weights in particle swarm optimization," in *Proceedings ofthe Sixth International Conference on Machine Learning and Cybernetics*, Hong Kong, 2007.

[16] Ahmad Nickabadi, Mohammad Mehdi Ebadzadeh وReza Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," in *Applied Soft Computing*, 2011.

[17] "http://www.cloudbus.org/CloudSim/,"

[18] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R, " CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," in *Software: Practice and Experience*, 2011.

[19] "http://www.cs.huji.ac.il/labs/parallel/workload/,"