Analysis of Particle Swarm Optimization and Genetic Algorithm based on Task Scheduling in Cloud Computing Environment

Frederic Nzanywayingoma
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China

Prof. Yang Yang
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China

***Abstract*—Since the beginning of cloud computing technology, task scheduling problem has never been an easy work. Because of its NP-complete problem nature, a large number of task scheduling techniques have been suggested by different researchers to solve this complicated optimization problem. It is found worth to employ heuristics methods to get optimal or to arrive at near-optimal solutions. In this work, a combination of two heuristics algorithms was proposed: particle swarm optimization (PSO) and genetic algorithm (GA). Firstly, we list pros and cons of each algorithm and express its best interest to maximize the resource utilization. Secondly, we conduct a performance comparison approach based on two most critical objective functions of task scheduling problems which are execution time and computation cost of tasks in cloud computing. Thirdly, we compare our results with other existing heuristics algorithms from the literatures. The experimental results was examined with benchmark functions and results showed that the particle swarm optimization (PSO) performs better than genetic algorithm (GA) but they both present a similarity because of their population based search methods. The results also showed that the proposed hybrid models outperform the standard PSO and reduces dramatically the execution time and lower the processing cost on the computing resources.**

Keywords—Execution Time; Task Scheduling Algorithms; Particle Swarms (PSO); Genetic Algorithm (GA); Virtual Machines (VMs)

I. Introduction

Cloud computing[1] is the delivery of computer services and resources including networks, data storage space, computer processing power, specialized corporate and user applications over the internet. Cloud computing models allow cloud users to use software and hardware that are managed by cloud providers without knowing which servers are in use to deliver service or knowing their exact physical locations where their data are stored. The cloud providers provide services that can be grouped into three models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

Service is a very important concept in cloud computing environments. Service is used to illustrate the details of a resource within the cloud. Cloud services and resources are registered within one or more cloud Information Servers. The cloud users send the requests to the scheduling task manager. Then after the scheduling task manager receives the service requests from the users tracks the available active resources to assign the services. The services are executed depending on the task scheduling strategies.

A service requests may be any online file storage, online business applications, social media sites, any software access and execution or any data processing. We define a task as a request for a task of the contracted application that may require a defined amount of resources and the creation of a virtual machine to support the application. Scheduling is the matter of assigning tasks to machine to achieve their work. It is used to decide which of the outstanding requests is to be allocated resources. A task scheduling is defined as a set of rules that decide the tasks to be executed at a particular time[2].

Scheduling is a challenging problem in cloud computing environment. As mentioned in [3, 4] task scheduling is NP-complete problem that requires heuristic methods. The work[5] presents a particle swarm optimization (PSO) based heuristic method to schedule tasks in Cloud resources that takes into consideration both execution time and computing cost. Other three existing basic heuristic methods inspired from nature for cloud computing such as Genetic Algorithm(GA), Simulated Annealing(SA) and Tabu Search(TS) heuristics for cloud task scheduling were presented in several works[6, 7],[8, 9], and[10]. PSO works well in solving global optimal problems and it has a good ability of global searching and was applied in other areas like neural network, system analysis, design, robotics, and so on. This work uses a comparison approach between two nature inspired heuristic methods, PSO and GAs algorithms applied in task scheduling to minimize the two parameters mentioned above simultaneously. Another notable advantage of PSO and GA is that they perform better in problems for which the searching space is complex - those where the objective function is discontinuous, changes over time, or has many local optima[11]. PSO and GA have both characteristics of exploring simultaneously different parts of the solution space, area less prone to converge to these local optima. GA and PSO are flexible to handle constraints which may be implemented more easily, when comparing to the `standard' optimization techniques. PSO is a population consisting of various particles, with each particle representing a solution.

This research was sponsored by the Student Financing Agency of Rwanda and National Science Foundation of China (Grant Nos. 61202508, 61272432, 61370132, 61472033, and 61370092, Fundamental Research Funds for the Central Universities [FRF-TP-14-045A2]).

A Genetic Algorithm is a search technique to find solutions to optimization and search problems. One of the first references to it was made by Holland (1975). It uses concepts inspired from biological evolution such as inheritance, selection, crossover and mutation. The comparison between GA and PSO shows that PSO presents more focused search ability than GA. PSO takes more emphasis on exploitation than exploration. PSO concentrates the search around a promising area in order to refine a candidate solution and explores different region of the search space to locate a good optimum. Both PSO and GA depend on good initial positioning of the particles in the solution space[12]. With their exploitation and exploration, the particles fly through the problem space and get two reasoning capabilities: the memory of the best position (pBest) and memory of the neighborhood's best position (gBest)[13, 14]. The same as in cloud systems, each task runs on virtual machine where the resources are distributed virtually like the way particle swarm moves through problem space maintain useful information of their local position and global position. The position of particle depends on the velocity and should be updated each time the particle moves from one point to the next position. Assuming that the tasks are totally different and are dependent as particles move in swarm and all tasks need to use resources such as CPU, memory, bandwidth, to be accomplished and they must be measured in terms of cost. The more accurate costs, the more the profits are[15].

Our main aim in this study is to minimize the execution time and computation cost. Since the traditional approaches used in optimization provided can't be applicable in cloud computing or present weaknesses, modern heuristic based algorithms were developed and have been proven to be suitable for task scheduling.

This paper involves various sections describing genetic algorithm(GA) and particle swarm optimization(PSO) and it is organized as follows: In section I, we introduced PSO and GA algorithms and listed their pros and cons; in section II, we cited the related work, in section III, we conducted a comparison method to compare two based heuristic algorithms: PSO, GA, and we proposed PSO-GA; in section IV, we discussed and modeled task scheduling problem by a task graph; in section V, we outlined the experimental set up, parameter settings, and benchmark functions used to measure the performance between PSO and GA; finally, section VI contains the conclusion of the paper.

II. Related Work

Since cloud resources are heterogeneous, dependent, and present a lot of capabilities, task scheduling problem becomes NP-complete problem. We define NP-complete problems as computational problems which are normally hard to be solved in real world such as vertex cover, knapsack, or traveling salesman problems and which have the property that they can be solved in polynomial time if and only if all other NP-complete problems can also be solved in polynomial time by maximizing or minimizing some values[16]. NP-hard problems are indispensable in practical applications to develop heuristic method to provide ways to measure, analyze, compare and increase the system performance[17]. As purpose of task scheduling algorithm in cloud system is to get optimal task-processor assignment and minimize application completion time and the total cost, it is our viewpoint that we explore how PSO and GA work and how they can be applied to task scheduling problems from the individual particle's point of view to the chromosome in all the searching space.

PSO approach can solve the task scheduling problems. Therefore, we list other approaches to solve scheduling problems[5] such as GA [18], Simulated annealing[9], tabu search[10], and ant colony [19]. The work[20]studied the comparison of particle swarm optimization and the genetic algorithm in the improvement of power system and stability. L. Zhang et al.[21] has compared GA and PSO in times of minimum completion time. Other comparison of particle swarm optimization and the genetic algorithm can be found in [22]. It was found that PSO is comparable to the Genetic Algorithm (GA) so that these two heuristics are population-based search methods[22]. A comparative study of DE, PSO was also introduced in[5] with objective of examining which algorithm outperform better among all others on a large and diverse set of problems.

III. Particle Swarm Optimization (PSO) Versus Genetic Algorithm (GA)

A. *Basic principles and implementation of Particle Swarm Optimization*

PSO was firstly introduced by J. Kennedy through simulation of a simplified social model to the optimizer. PSO has found widespread application in two main component methodologies: one in artificial life and another one based to bird flocking, fishes schooling, and swarm theory. As mentioned in [23], the advantages of using PSO in task scheduling are as the following: a PSO algorithm can maintain useful information about characteristics of the environment; PSO as characterized by its fast convergence behavior, has an in-built ability to adjust to a dynamic environment; PSO is effective for locating and tracking optima in both static and dynamic environments. The particle swarm optimizer has been found to be fast in solving nonlinear, non-differentiable, multi-modal problems[24]. PSO introduces a method for optimization of continuous non-linear functions. Other advantages of PSO with optimization algorithms are that PSO present a simple mathematical operation with less parameters, and is inexpensive in terms of both memory and speed requirements. PSO have no overlapping and mutation calculation[12]. The disadvantages of PSO algorithms are cited in[23]as the following: (1)The method suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction. (2)The method cannot work out the problems of scattering and optimization. (3)The method cannot work out the problems of non-coordinate system, such as the solution to the energy field and the moving rules of the particles in the energy field. Every single solution is a bird in the searching space called a "particle" and all particles possess positions and velocities. The particles fly through the problem space by following the current optimum particles. Each time a particle moves from one bin to another. In the whole searching space, all particles depend on the value of the chosen optimization function and have the following information: position and the speed. The Fig.1 below represents the traditional Particle Swarm Optimizer in multiprocessor environment.

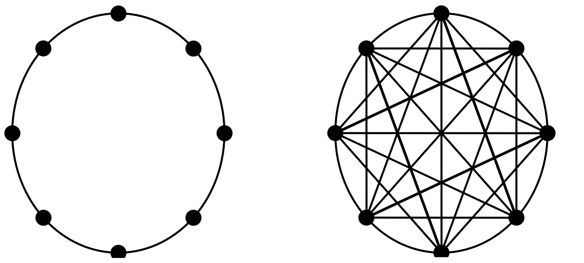

Fig. 1. Two traditional neighbourhood topologies

In order to achieve good optimization, each particle in the searching space moves with two information: position and velocity. We have two kinds of traditional topologies in figure1: (1) Ring topology to represent local best position and full mesh topology to represent the global best position. All particles have positions and velocities. The i^{th} particle is represented with the following elements: x_i^k the current particle positions; v_i^k the velocities vector, the current best position $pBest_i$ and global positions $gBest_i$. c_1 and c_2 are the acceleration coefficients, r_1 and r_2 are two random vectors which can take any value between 0 and 1. The initialization process is given in the following formula.

$$x_i^0 = x_{\min} + rand(x_{\max} - x_{\min})$$

$$v_i^0 = \frac{x_{\min} + rand(x_{\max} - x_{\min})}{\Delta t}$$

At initial position particles position will be x_i^0 then all particles move towards the optimal point with the velocity. At the time k+1, there must be an update of all particles with particle objective or fitness value for the next iteration. PSO is described by the below equations:

$$v_i^{k+1} = \omega v_i^k + c_1 rand_1 \times (pbest_i - x_i^k) + c_2 rand_2 \times (gbest_i - x_i^k)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}\Delta t$$

Where v_i^k is the velocity of the i^{th} particle at the k^{th} iteration; ω is the inertia factor; c_1 and c_2 are the acceleration constants (cognitive and social); $rand_1$ and $rand_2$ are the random numbers between 0 and 1; for $i = 1,2$; x_i^k is the current position of the i^{th} particle at the k^{th} iteration; $pbest_i$ is the best position for the i^{th} particle and $gbest_i$ represents the particle position or global position. To achieve a high performance, we set the inertia weight as

$$\omega = \omega_{end} + (\omega_{start} - \omega_{end})e^{-\frac{ya}{y_{\max}}}$$

ω_{start} and ω_{end} are the starting and ending inertia values. We set their values to 0.65 and 0.2 respectively. y and $y_{\max}$ represent the current and maximum iteration number which we set to 100 and a is an integer constant number.

B. *Basic principles and implementation of task scheduling based on Genetic Algorithms*

A GA is among the evolutionary algorithms which mimic the process of natural selection used to solve optimal and search problems[25]. It generates solutions to optimization problems using natural evolution methods. We present different genetic algorithm operators as follows:

1) *Encoding and initialization*

In genetic algorithm task scheduling-based, the initial population of candidate solutions is randomly generated. The chromosome sequence represents a variety of tasks. Every task is considered as a gene. The chromosome is encoded using permutation encoding. The length of chromosome is the same as the length of the input tasks.

To start, the initial population is generated randomly using random generator function of chromosomes. Some resource information such as CPU, number of tasks, the size of population is needed to create the initial population.

TABLE I. A SAMPLE CHROMOSOME OF 5 TASKS

2	3	1	5	4

Table 1 shows a sample chromosome of 5 tasks with their task allocations: tasks $\{2,3\}$ are assigned to resource 1, task $\{1\}$ is assigned to resource 2, and tasks $\{5,4\}$ are assigned to resource 3.

2) *Fitness function*

The fitness function is the evaluation function to guide the search space. For task scheduling based on genetic algorithm in cloud computing, the fitness function is based on execution time, computation cost and measures the quality of the solution and determines if the genetic material will be transmitted from parent to offspring. It helps to transform the objective function value in a measure of relative fitness[26].

$$F(x) = g(f(x))$$

The objective function f and g are two functions which result to relative fitness. f is used to measure how the individuals have performed in the problem domain and g transforms the value of the objective function f to a negative number. $F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)}$, where N_{ind} represent the population size and x_i is the phenotypic value of individual i .

3) *Crossover*

Crossover operator is used to vary the programming of the chromosomes from one generation to the next.

4) Mutation

The mutation operation expands the search space by decreasing the execution time based on mutation probability and generates the offspring with different assignment. P_m is the probability of mutation. It is not greater in nature and during our matlab simulation of results; the probabilities of mutation are randomly given by computer.

C. Comparison of genetic algorithms and particle swarm optimization

In this section, we compare PSO and GA. As both algorithms introduce the basics of evolutionary computing,

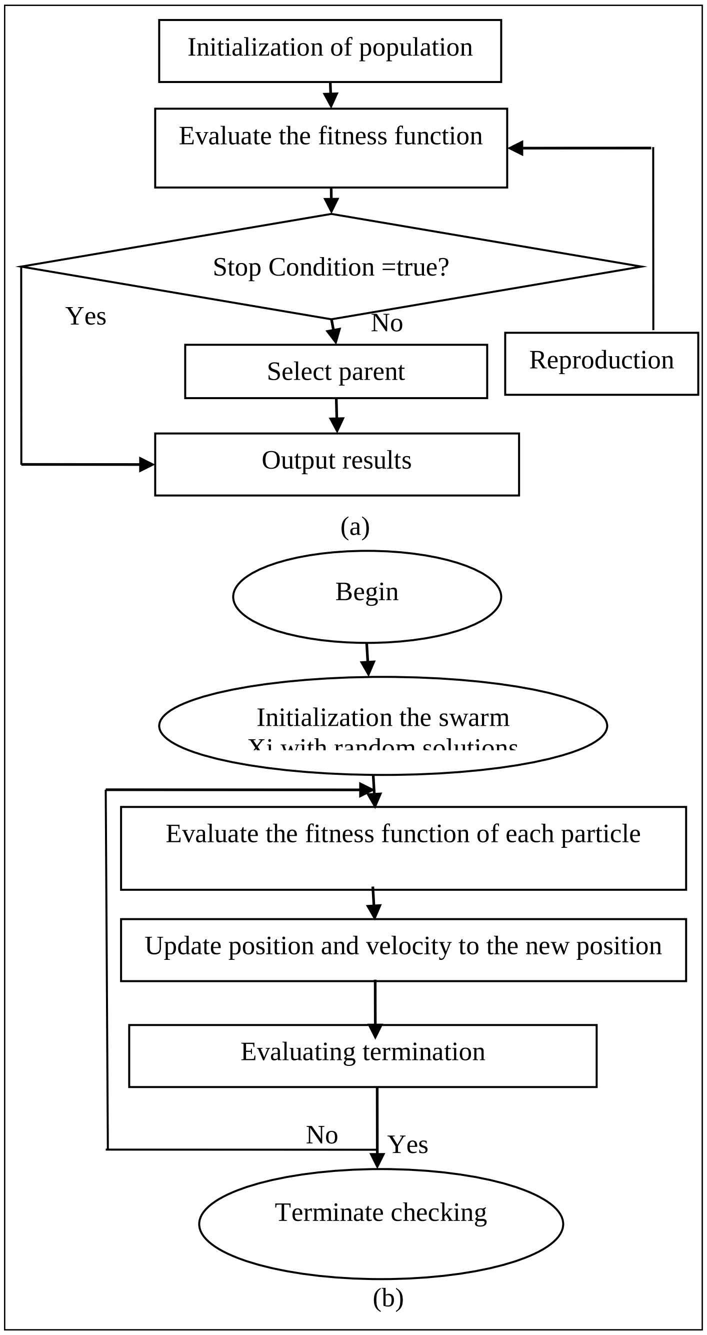
Initialization of population
Evaluate the fitness function
Stop Condition =true?
Yes
No
Select parent
Reproduction
Output results
(a)
Begin
Initialization the swarm Xi with random solutions
Evaluate the fitness function of each particle
Update position and velocity to the new position
Evaluating termination
No
Yes
Terminate checking
(b)

Fig. 2. Flow chart of genetic algorithm (a) and PSO algorithm (b)

PSO shares many similarities techniques with GAs in particular [27]. GA and PSO are both heuristic algorithms and are used in optimization problems to find solution to a given objective function by using different techniques and computational effort. Fig.2 represents the flow chart of GA (a) and PSO algorithm (b). GA begins with a population of random chromosomes to present a better solution to the problem. At each step, the GA takes individuals from the current population to be parents and uses them to produce the children for the next generation. GA uses operators such as crossover and mutation. GAs and PSO can both be applied in pattern discovery, signal processing, neural networks, cloud computing, manufacturing, power Electronics to control power System such as scheduling power flow, providing voltage support, limiting short-circuit, etc[27, 28].

```
//The pseudocode of the proposed PSO&GA algorithm
Set the particle dimensional according to the ready tasks
Initialize the particle swarm position $X_i$ and velocity $V_i$
randomly,
Repeat
        for each particle $i=1,2,...,P$ do
                if $f(X_i)>f(pest_i)$ then //Calculate the fitness value
                    $pbest_i = X_i$;
                    end
                    if (f(pbesti)>f(gbesti) then
                    $gbest_i = pbest_i$;
                    end
                  end
                  for each particle i=1,2,...,P do
                       update the velocity matrix //update the
velocity of each particle
                      update the position matrix //update the
position of each particle
            end
            Until stopping condition is true//
```

GA vs PSO Scheduling algorithms

```
*The pseudocode of the average computation cost for all resources*
Calculate average computation cost of all tasks in all compute resources
Calculate average communication cost between resources
Set task node weight $\omega_{kj}$ as average computation cost
Set edge weight $e_{k1,k2}$ to the size of the transferred between tasks
Compute $PSO(\{t_i\})$ //a set of all tasks
Repeat
    for allready tasks do
    Assign tasks $\{t_i\}$ to available resources $p_j$ according to PSO's solution
   end for
       Dispatch all the mapped tasks
      Wait for polling_time
      Update the ready task list
      Update the average cost of communication between
          resources
    Compute $PSO(\{t_i\})$
    Until there are unscheduled tasks
```

IV. Task Scheduling in Cloud Computing using Hybrid GA-PSO Model

The task scheduling aim [23] is to assign incoming tasks to the available resources. According to the scheduling strategies used, the task scheduling algorithms can significantly affect the efficiency of the whole system. In this paper, we are using hybrid PSO and GA models to solve a task scheduling problem in cloud computing. As a result, the first task which is the most useful is to know how to model the problem as a set of individuals. In order to model the task scheduling problem, suppose that the number of swarm particles correspond with a set of task numbers. Then we denote n as the number of tasks and m the number of available heterogeneous computing resources. The objective to model the scheduling problem is to find the best resource utilization. Here the fitness of a particle is measured with execution time and communication cost to all tasks. In this paper, task scheduling problem is modeled by a task graph. Firstly, using task graph model, tasks are represented by nodes and edges represent the dependencies between tasks. Let $G=(V,E)$ be a graph with $V=\{t_1,t_2,...,t_n\}$ as a set of tasks nodes/vertices, and E is a set of directed edges between two tasks t_i and t_k. The graph in Figure 3 starts with root node and ends with end node. The node with no parent is called an entry node or root node and a node with no child is called an exit node or end node. A task t_1 is called the entry task and t_n the exit task of the graph. We calculate the communication cost according to the amount of data to be transmitted between resources and the available bandwidth between the resources. If we suppose that n tasks are submitted from the task schedule manager to the available resources, and we suppose that those tasks are dependent to each other with inter-task data dependencies and they are non-preemptive; and also if we assume that the number of the tasks is less than the number of available resources, we will rely on the first come-first-served rule. Otherwise we will adopt other scheduling schemes where the number of tasks is greater than the number of resources. From Fig.3 below, task 5 cannot start its execution until task 2 and task 4 complete their executions.

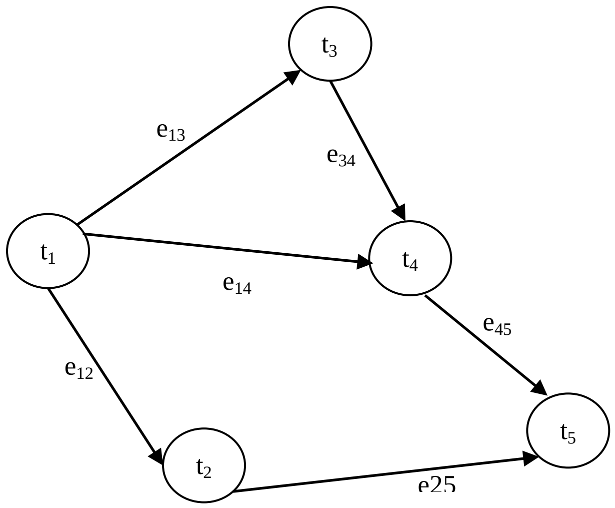

Fig. 3. Task graph with 5 tasks

Secondly, mapping the set of tasks to the available heterogeneous resources, we can compute the completion time of the tasks. To map a set of resources, consider *m* number of available heterogeneous computing resources, and b_{ij} the bandwidth between resources as it is shown in Figure 3. Then calculate the available bandwidth $B=(b_{ij})_{NxN}$ for the available resource. Fig.4. shows that a task can be executed randomly by the available resources after finding that there are a finite number of possible mappings from a collection $T=\{t_1,t_2,....,t_n\}$ to a collection $M=\{r_1,r_2,...,r_m\}$ and a large number of pair of task and resource.

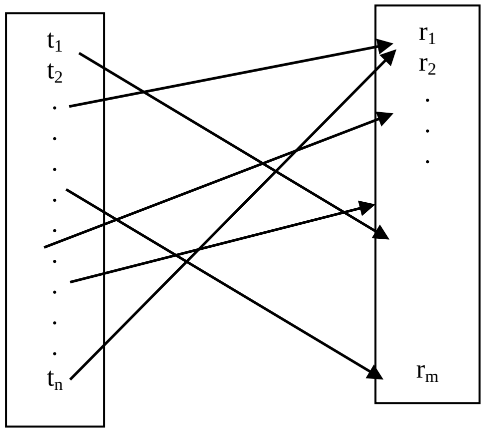

Fig. 4. Mapping of the task to available resources

We consider a discrete-time model with a collection M of machines indexed from $1,2,...,m$. Tasks come in with a tagged random mapping number and each task is associated with m number of available resources and they are flocked together according to their indices in an increasingly order into a vector

$$\vec{V} \in \{(r_1, r_2, \ldots, r_m) \in \{1,2,\ldots,M\}^m \; r_1 < r_2 < \ldots < r_m\}$$

and execution time equals to the ration of the workload and computation ability of the resource r_i

$$ET_{ij} = \frac{\sum_{i\geq 1}^{n} t_k}{\sum_{j\geq 1}^{m} r_k}$$

$T = \{t_i \;\; 1 \leq i \leq n \;\;\}$represents a set of n tasks

$R = \{r_j \;\; 1 \leq j \leq m \;\;\}$ represents a set of m resources

$E = \{e_{i,j} \;\; 1 \leq j \leq m \;,\; 1 \leq j \leq m \;\;\}$ represents a matrix of communication times of task on resource t_i number of resources r_j

The communication cost of edges is defined as

$$CT_{ij} = \begin{cases} \frac{e_{nm}}{b_{ij}} \text{ if } t_i \text{ is a predecessor of } t_j \text{ and } i \neq j \\ \qquad\qquad\qquad\qquad \textit{otherwise} \\ 0 \end{cases}$$

e_{nm} represents the quantity of data to be transmitted between two resources and b_{ij} is representing the link communication speed between two resources. If $e_{nm} = 0$, that means that both tasks t_i and t_j are assigned on the same resource.

V. Experimental Results and Statistical Analysis

A. Simulation environment

To compare the performance of PSO and GA algorithms, we take into consideration various parameters such as number of tasks, number of processors, swarm size, population size, number of chromosomes, and number of iteration. The algorithms are simulated with java language running and in matlab on Intel(R) dual-Core(TM)i5-4590 CPU@3.30GHz, 4.00GB installed memory on windows 7 Ultimate service park1 and NetBeans IDE 8.0.2.

Table2 gives a summary of PSO&GA parameters. Firstly, genetic algorithms will run with the following parameters: the population size, crossover probability, mutation probability, and maximum number of iteration. Secondly, the particle swarm optimization will run with the following parameters: number of particle (Swarm size), maximum velocity $V_{\max}$, the neighborhoods best found solutions $c_1 = c_2 = 2.0$, number of iterations=$[20 \times n]$with n stands for the number of nodes, and inertia weight. The inertia weight will decrease linearly over time up to 0.1.

TABLE II. Summary of (1) PSO Parameters (2) GA Parameters

1	GA parameters	Population size	60
		Crossover probability	0.7
		Mutation probability	0.01
		Number of iterations	100
2	PSO parameters	Population size	60
		ω	0.65
		C1	2
		C2	2
		Number of iterations	100

B. Simulation Result and analysis

In this work, hybrid PSO-GA algorithms are used to solve task scheduling problem in cloud computing, and a comprehensive performance based on benchmark functions has been conducted. We applied Schaffer and Ackley benchmark functions showed in Table III below to assess the performance of the algorithms. We chose the ranges of their searching space and their dimensions. We ran 100 test computations randomly on a couple of test functions. The combined PSO-GA algorithm performs well for all test functions as it is represented in Fig.5 and Fig. 6 and it can easily find the global minima in 100 runs better than PSO or GA.

TABLE III. Benchmark Functions

Names	Functions
Schaffer	$0.5 + \frac{\sin\sqrt{x^2+y^2} - 0.5}{(1.0 + 0.001(x^2+y^2))^2}$, $-100 \leq x_i \leq 100$
Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{d=k}^{D} x_d^2}\right) - \exp\left(\frac{1}{D}\sum_{d=1}^{D}\cos(2\Pi x_d)\right) + 20 + e$

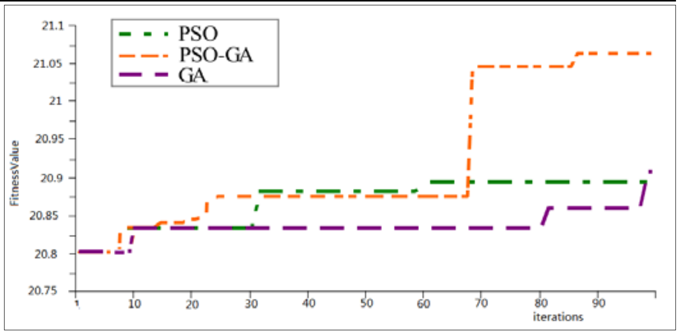

Fig. 5. Ackley function

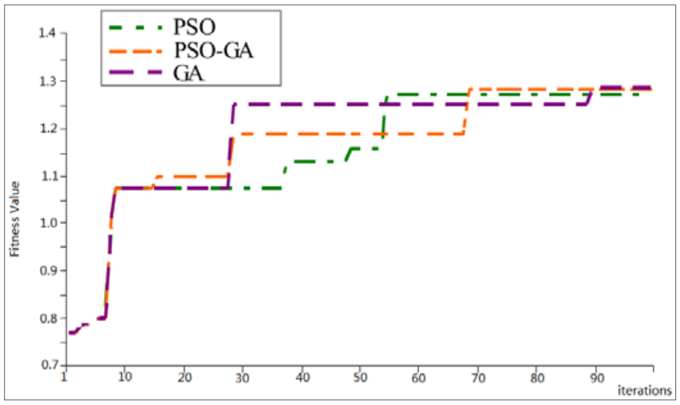

Fig. 6. Schaffer function

VI. Conclusion

In this study, heuristic algorithms were compared based on task scheduling problems and based on two QoS (quality of service) parameters. The main purpose of the work is to use comparison approach to determine the efficiency of GA and PSO. The study found that PSO and GA are similar in finding the global optimal solution because they all utilize the fitness value to evaluate the population and also they all update the population. The criterions considered to major the performance are execution time and processing cost. In this study, we explored how PSO/GA work and apply them to solve NP-complete problems of task scheduling in cloud computing based on execution time and processing cost. Using these two algorithms, the results show that the genetic algorithm (GA) presents high global searching ability but has poor computation efficiency, and poor optimization speed compared to its counterpart. PSO presents good advantages in convergence speed, in finding global optimal, and in simplicity ability. Therefore, we conclude by saying that while using PSO algorithms the cloud computing resources can easily notice resources discovery, resources matching, and task execution. The results show that the combination of these two algorithms can reduce dramatically the task execution time, and reduce the computation cost on the available resources. In the future work, better results will be provided by improving our solution using PSO combined with other meta-heuristic techniques(i.e Simulated Annealing(SA), Tabu Search(TS), etc.).

References

[1] Bakshi, T., et al. A New Meta-heuristic PSO Algorithm for Resource Constraint Project Scheduling Problem. in Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012). 2013. Springer.

[2] Dubey, S. and S. Agrawal, QoS driven task scheduling in cloud computing. International Journal of Computer Applications Technology and Research, 2013. 2(5): p. 595>< meta name=.

[3] Chen, Z.-G., et al. Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. in 2015 IEEE Congress on Evolutionary Computation (CEC). 2015. IEEE.

[4] Wickboldt, J.A., et al., Resource management in IaaS cloud platforms made flexible through programmability. Computer Networks, 2014. 68: p. 54-70.

[5] Lu, X., J. Zhou, and D. Liu, A Method of Cloud Resource Load Balancing Scheduling Based on Improved Adaptive Genetic Algorithm. Journal of Information and Computational Science, 2012. 9(16): p. 4801-4809.

[6] Pradhan, S.R., et al., A Comparative Study on Dynamic Scheduling of Real-Time Tasks in Multiprocessor System using Genetic Algorithms. International Journal of Computer Applications, 2015. 120(20).

[7] Wan, B., A Hybrid genetic scheduling strategy. International Journal of Hybrid Information Technology, 2008. 1(1): p. 73-80.

[8] Sahoo, B., S. Mohapatra, and S.K. Jena, A genetic algorithm based dynamic load balancing scheme for heterogeneous distributed systems. 2008.

[9] Van Laarhoven, P.J. and E.H. Aarts, Simulated annealing: theory and applications. Vol. 37. 1987: Springer Science & Business Media.

[10] Glover, F., Tabu search-part I. ORSA Journal on computing, 1989. 1(3): p. 190-206.

[11] Bajpai, P. and M. Kumar, Genetic algorithm–an approach to solve global optimization problems. Indian Journal of computer science and engineering, 2010. 1(3): p. 199-206.

[12] Chapman, B., When clouds become green: the green open cloud architecture. Parallel Computing: From Multicores and GPU's to Petascale, 2010. 19: p. 228.

[13] Kennedy, J., Particle swarm optimization, in Encyclopedia of Machine Learning. 2010, Springer. p. 760-766.

[14] Sedighizadeh, M., et al., Parameter optimization for a PEMFC model with particle swarm optimization. Int J Eng Appl Sci, 2011. 3: p. 102-108.

[15] Liu, C.-Y., C.-M. Zou, and P. Wu. A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. in Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on. 2014. IEEE.

[16] Wang, N., et al. A task scheduling algorithm based on qos and complexity-aware optimization in cloud computing. in Information and Communications Technology 2013, National Doctoral Academic Forum on. 2013. IET.

[17] Raghavendra, P., Approximating np-hard problems efficient algorithms and their limits, 2009, University of Washington.

[18] John, H., Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, 1992, MIT Press, Cambridge, MA.

[19] Colorni, A., M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. in Proceedings of the first European conference on artificial life. 1991. Paris, France.

[20] Peyvandi, M., M. Zafarani, and E. Nasr, Comparison of Particle Swarm Optimization and the genetic algorithm in the improvement of power system stability by an SSSC-based controller. Journal of Electrical Engineering and Technology, 2011. 6(2): p. 182-191.

[21] Pico, C.G. and R.L. Wainwright. Dynamic scheduling of computer tasks using genetic algorithms. in Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. 1994. IEEE.

[22] Hassan, R., et al. A comparison of particle swarm optimization and the genetic algorithm. in Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference. 2005.

[23] Rostami, A. and M. Lashkari, Extended PSO algorithm for improvement problems K-Means clustering algorithm. International Journal of Managing Information Technology, 2014. 6(3): p. 17.

[24] Zhan, S. and H. Huo, Improved PSO-based task scheduling algorithm in cloud computing. Journal of Information & Computational Science, 2012. 9(13): p. 3821-3829.

[25] Mitchell, M., An introduction to genetic algorithms. 1998: MIT press.

[26] Chipperfield, A. and P. Fleming. The MATLAB genetic algorithm toolbox. in Applied control techniques using MATLAB, IEE Colloquium on. 1995. IET.

[27] Jones, K.O. Comparison of genetic algorithm and particle swarm optimization. in Proc. Int. Conf. Computer Systems and Technologies. 2005.

[28] Panda, S. and N.P. Padhy, Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design. Applied soft computing, 2008. 8(4): p. 1418-1427.