An Efficient Scheduling Technique for Cloud Computing Systems Based on Particle Swarm Optimization and Genetic Algorithm

Ebrahim Behrouzian Nejad[†]

Department of Computer, Shoushtar Branch, Islamic Azad University, Shoushtar, Iran

Summary
Cloud computing is a new processing method for delivering information technology services based on computer networks. One of the main challenges in cloud based system is task scheduling algorithm. Performance of cloud computing systems largely depend upon underlying scheduling algorithms which can affect other performance parameters such as makespan, load balancing, cost and energy consumption. So, in this paper a scheduling algorithm is presented called HGPA which is based on Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). This algorithm attempts to improve load balancing and make span parameters in cloud computing systems. Evaluation and simulation results show that this proposed scheduling algorithm is able to reduce makespan and improve load balancing in cloud computing systems in comparison with other evaluated techniques.

Key words:
Cloud computing systems, Scheduling Algorithm, Particle Swarm Optimization and Genetic Algorithm (GA).

1. Introduction

In recent years, Cloud Computing has emerged as an important solution for cost effective and reliable delivery of information technology services. According the formal definition of the National Institute of Standards and Technology (NIST), "cloud computing is a model for enabling ubiquitous, convenient, on- demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. Efficiency of cloud computing system is depend upon several parameters. The main popular and known services of cloud computing are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In fact, cloud computing can be regarded as the next stage in evolution of the network computing. Due to importance of cloud computing in developing and providing user requirements as web-based service, many of researchers and specialists are attracted to this area of information technology.

.

Manuscript received August 5, 2026
Manuscript revised August 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.8.14

Achieving high performance in a cloud system requires effective task scheduling and also load balancing. So, selecting an efficient task scheduling technique is one of the main challenges is in the cloud computing systems. The heterogeneous and dynamic nature of the cloud based systems, as well as the differing demands of applications run on the cloud, makes cloud scheduling complicated, so, the deterministic algorithms will not necessarily be efficient to solve these kinds of problems.

Scheduling is a NP-hard problem and with a given number of jobs and the machines, it is difficult to find a definitive solution to this problem. So, Scheduling problem is more complicated to be solved by deterministic algorithms with polynomial complexity [2][3]. Hence, various researches have focused on using heuristic algorithms to solve task scheduling problem in cloud-based systems. A good and efficient task scheduling algorithm may be result to better performance in cloud system, on other hand, a non-efficient one may lead to lower performance, higher cost and non-suitable load balancing in cloud computing systems.

In this paper we try to introduce an efficient task scheduling algorithm to achieve better load balancing and minimized makespan. To achieve these goals, a hybrid task scheduling method is presented which is based on Particle Swarm Optimization Algorithm (PSO) and Genetic Algorithm (GA). This technique is a hybrid algorithm, which is called Hybrid Genetic-PSO Algorithm (HGPA). Evaluation and simulation results show that the proposed method is better than the current methods in the task scheduling problem.

The simplicity and parallel nature of genetic algorithm and also searching the problem environment in different ways leads to using it for resolving several optimization problems. But since genetic algorithm is an inherent algorithm which searches the problem space globally and does not have the efficiency required for local searching, combining it with local search algorithms can compensate for this shortcomings [2][4]. Therefore, we combined GA with other meta-heuristic methods such as PSO to resolve the local search problem. In this way we

can achieve better performance for The main goal of this paper is to present an effective scheduling technique to reduce makespan, which is the required time to finish all the running tasks. Reduction of the makespan will lead to increasing performance and better usage of cloud resources. Other goal is achieving better load balancing. Load balancing make it possible to have higher resource utilization and even better response time in cloud computing systems. Load balancing helps to distribute all tasks between all the cloud resources. It is also possible to ensure that every computing resource is distributed efficiently and fairly via load balancing techniques. Load balancing can remove bottlenecks of the system which may occur due to load imbalance [5].

This paper is organized as follows: In section 2 we review Genetic Algorithm and Particle Swarm Optimization to provide an appropriate background to present our new method in next sections. Also we explain the concept of task scheduling in this section. Related works are reviewed in Section 3. Section 4 explains our proposed scheduling algorithm in detail. Evaluations and simulation results are presented in section 5. Finally, section 6 gives the conclusion of the paper and future works.

2. Backgrounds review

In this section we review Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), because these our presented scheduling algorithm is based on these two algorithms.

2.1. Genetic Algorithm

Genetic Algorithm (GA) is one of the old Meta heuristic algorithms which was first proposed in 1975 by John Holland et al [6]. The simple and parallel features of this algorithm make it appropriate and applicable to several optimization problems, especially in computer world and computer networks related problem it is used to solve optimization problems. In brief, A GA selects the most suitable strings from organized stochastic information that is searched and gathered by humans. In each generation, a new set of strings is produced based on artificial strings with the help of the most suitable bits and elements among the old elements. The new set is tested stochastically, and its strength or fitness level is evaluated [2][6]. The general form of a GA is as follows:

A. **Start:** Generate random population of *n* chromosomes
B. **Fitness:** Evaluate the fitness *f(x)* of each chromosome *x* in the population
C. **New population:** Create a new population by repeating following steps until the new population is complete
    1. **Selection:** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **Crossover:** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
    3. **Mutation:** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **Accepting:** Place new offspring in the new population
D. **Replace:** Use new generated population for a further run of the algorithm
E. **Test:** If the end condition is satisfied, **stop**, and return the best solution in current population
F. **Loop:** Go to step **2**

2.2. Particle Swarm Optimization Algorithm (PSO)

Particle swarm optimization (PSO) is another population based stochastic optimization algorithm which is proposed by Kennedy and Eberhart in 1995 [7]. In this algorithm behavior of flock of birds or group of fishes in order to get to desired destination is simulated. Similar to GA, in PSO system is initialized with a population of random solutions and searches for optima by updating generations. In PSO each particle in the population regarded as a potential solution. PSO is initialized with a group of random particles (solutions. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbours, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equation (1) and (2).

$$V_i^{k+1}=W_k \times V_i^k+C_1 rand_1 \ (Pbest_i^k-X_i^k)+C_2 rand_2 \ (Gbest_i^k-X_i^k) \quad (1)$$

$$X_i^{k+1}=X_i^k+V_i^{k+1} \quad (2)$$

Where:

V_i^k is velocity of particle *i* at iteration *k*
V_i^{k+1} is velocity of particle *i* at iteration *k+1*
W is inertia weight
$Pbest_i^k$ is best position of particle *i*
$Gbest_i^k$ is position of best particle in a population
C_j is acceleration coefficients*; j=1,2*
$rand_i$ is a random number between 0,1; *i=1,2*
X_i^k is current position of particle *i* at iteration *k*
X_i^{k+1} is position of particle *i* at iteration *k+1*

c1, c2 are learning factors. usually c1 = c2 = 2.

The pseudo code of the procedure is as follows:

For each particle
Initialize particle
END

Do
For each particle
Calculate fitness value
If the fitness value is better than the best fitness value (pBest) in history
set current value as the new pBest
End

Choose the particle with the best fitness value of all the particles as the gBest
For each particle
Calculate particle velocity according equation (1)
Update particle position according equation (2)
End
While maximum iterations or minimum error criteria is not attained.

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax[7] [8].

2.3 Scheduling Concept

Scheduling is the one of the most important topics discussed in cloud computing. An efficient task scheduling algorithm can improve the performance of cloud system, on other hand, a non-efficient one may lead to lower performance, higher cost and non-suitable load balancing in cloud computing systems. Now, we describe an example of scheduling and load balancing.

Suppose there are two machines (Resource) and three tasks to run on these machines. We assume that both machines have the same performance and the time required to perform each task as in Table 1.

Table 1: Information of three tasks

Task1	Task2	Task3
2	4	7

Some possible solutions for scheduling these three tasks are shown in Table 2, Table 3 and Table 4.

Table 2: Solution 1

Machine#	Task #		Times	Makespan
Machine1	2	7	9	9
Machine2	4		4	

Table 3: Solution 2

Machine#	Task #		Times	Makespan
Machine1	4	7	11	11
Machine2	2		2	

Table 4: Solution 3

Machine#	Task #		Times	Makespan
Machine1	2	4	6	
Machine2	7		7	7

Makespan is maximum amount of time required to complete all the tasks. Solution 3 has less makespan as a scheduling approach and is more efficient [4]. With a little attention, we can also find that this will happen when load balancing is satisfied. In this paper HGPA algorithm is proposed for task scheduling. It tries to minimize makespan and improve load balancing.

3. Related Works

Scheduling algorithms have a key role in cloud computing systems, So that they can make effect on various aspect of cloud based systems. An efficient scheduling algorithms can improve one or more of different performance parameters, such as makespan, load balancing, energy consumption, cost and so on. So, several efforts have been done attempting to improve scheduling algorithms with different goals and approaches. In this

section we review some presented scheduling algorithms, predominantly which use different Meta heuristic and hybrid algorithms to solve scheduling problem in grid and cloud computing systems.

In [4], a new meta heuristic scheduling technique is presented which is called which is called CSO-GA. This algorithm is a hybrid optimization algorithm which use a combination of Cat Swarm Optimization (CSO) and Genetic Algorithm (GA). This algorithm has two phase. The first phase is based on CSO algorithm. In the second phase, the outputs of CSO algorithm was introduced to GA. It is don, order to benefit from the advantages of Both CSO and GA algorithms. It is proved that this new hybrid algorithm is more efficient, and its outputs offer better load balancing in cloud computing systems.

In [5] a meta heuristic scheduling technique is presented which is called ACO-LB. this algorithm is a load balancing optimization algorithm based on ant colony algorithm which algorithm to solve the load imbalance of virtual machine in the process of task scheduling .The ACO-LB algorithm can adapt to the dynamic cloud environment. It will not only shorten the makespan of task scheduling, but also maintain the load balance of virtual machines in the data center.

By using of genetic algorithm and fuzzy theory, a hybrid job scheduling approach in [9], which considers the load balancing of the system and reduces total execution time and execution cost. In this paper the standard Genetic algorithm is modified and also the fuzzy theory is used to its optimization. The main goal of this research is to assign the jobs to the resources with considering the VM MIPS and length of jobs. In this paper the jobs are assigned to the resources with considering the job length and resources capacities.

Also, multiple scheduling algorithms for grid computing and cloud computing systems have been presented in another litretures, which are based on meta heuristic optimization algorithms [10] [11] [12][13] [14] [15] [16] [17].

4. Proposed Method

In this section we present our new scheduling algorithm which is based on GA and PSO algorithms and called HGPA. The aim of this algorithm is minimize makespan and improve load balancing in cloud computing systems. Before presenting our proposed method, it is necessary to determine the description and modelling of scheduling problem.

4.1 Modelling Task Scheduling Problem

To describe our new scheduling method, we use a popular model which is more similar to [2],[4] and [13]. In this model, scheduling problem space consists of N tasks and M machines. In our desired model, each task should be considered to be processed by each of the M machines, so that be able to improve load balancing and also makespan is minimized.

Each task can be executed on only one resource and is not stopped before its execution is complete. So that our proposed scheduling algorithms is static, we can assume that the expected execution time for each task i on each resource j have already been determined. We use the expected time to compute the ETC matrix model described in [18]. ETC matrix is a 2D matrix. Each element in ETC matrix, ETC [i,j], show expected execution time for each task i on each resource j [2][18].

As shown in Fig.1, our proposed algorithm is composed of two phases. At first phase genetic algorithm is performed for task scheduling. At the end of this phase, the best chromosomes are calculated as the best solution for task scheduling process. Then, in second phase, the outputs of genetic algorithm are given to PSO algorithm as input values. In this phase, PSO will find the optimum solution as final and the best solution for scheduling problem.

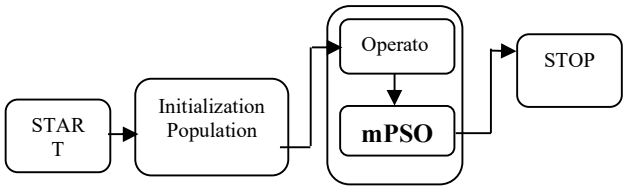

Figure. 1 Proposed method steps

4.2 First phase: Genetic Algorithm

In Genetic Algorithm the chromosomes and their representation method is very important, so that underlying representation method for chromosomes can affects the efficiency of this algorithm. Similar to other works [2][4][13], In the proposed scheduling algorithms, a simple method has been used to represent chromosomes. That natural numbers are used to encode the chromosomes. In this algorithm, a solution to the problem is represented by a chromosome. Length of chromosomes is considered equal to size of the number of input independent tasks. The values inside the genes are random numbers between 1 to M. M is the resource numbers. Algorithm finished when the iteration time of the algorithm reaches its maximum. Fig 2 shows an example of the chromosome representation. For example, in this figure, task T1 executes on resource R4, task T2 executes on resource R3 and so on.

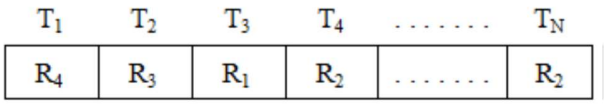

Figure. 2 Representation of a chromosome

If there is a possible solution to the scheduling problem, so that in this solution task *i* is allocated to resource *j*, then this solution can be shown as (3)

solution i = j (3)

One of the major goals in our work is minimizing makespan. The makespan is equal to the maximum complete time Completion_Time [*i*,*j*]. Makespan can be calculated as (4)

makespan=Max(Completion_Time[*i*,*j*]) (4)
{1≤*i*≤N,1≤*j*≤M}

Completion_Time [*i*,*j*] is equal to the time that task *i* is ended on resource j and is calculated as (5) .

Completion_Time[*i*,*j*]=Ready[*j*]+ETC[*i*,*j*] (5)

Where ETC[*i*,*j*] can calculates as (6)

ETC[*i*,*j*] = JobCycle *i* / Processor Speed *j* (6)
{1≤*i*≤N,1≤*j*≤M}

In (6), JobCycle*i* parameter indicates the length of task *i* (number of instructions) and Processor Speed*j*, indicates processor speed of the resource *j*. The speed of each resource is expressed in the form of MIPS (Million Instructions Per Second), and the length of each job in the number of instructions.
In each iteration of the population update, the generated solutions are evaluated. Based on this assessment, fitness functions are calculated.

Creating initial population

In this step, the initial solutions must be generated. The initial population is created randomly. A random number between 1 to M is generated. This randomly generated number determine the number of a machines which is selected to run considered task. In this way, a machine is selected randomly to execute desired task executed on it.

GA Fitness Functions

The major goal of our proposed scheduling algorithm is to minimizing makespan, which is the total time required until all of the tasks can complete their execution. Inspired by [2], In our proposed method we use fitness function as (7) to minimizing makespan.

Fitness(Solution *i*) = 1/ makespan *i* (7)

So, Solution *i* is more appropriate if it has less makespan.

In addition to minimizing makespan, we have another goal which is improve load balancing. Sometimes several chromosomes may be found that have similar makespans but don't all balance the load among their resources. So, Inspired by [2], in our proposed method we use another fitness function to improve load balancing.

If the execution time for resource R*j* is EXT[R*j*], the mean execution time (MXT) for all resources is as shown in Equation (8):

$$MXT = \sum_{i=1}^{NR} EXT[R_j]/NR \quad (8)$$

Where NR is the number of resources.
The load balance for resource *i*, Cpu_LB*i*, can then be calculated with Equation (9):

Cpu_LB *i* = makespan / MXT (9)

The fitness function to improve load balancing is calculated as (10)

Fitness (Solution *i*) = 1/ Cpu_LB*i* (10)

Selection operator

In this step, before using mutation and crossover operators, it is the selection step. Here we use elitism operator for our proposed algorithm.in this way the best solution will be selected and then crossover and mutation will be performed on them.

Crossover

The proposed algorithms, the uniform crossover operator is used. In uniform crossover, a value of the first parent's gene is assigned to the first offspring and the value of the second parent's gene is assigned to the second offspring. An example is shown in Fig 3.

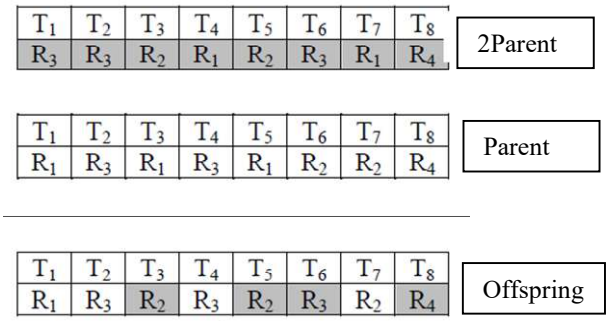

Figure. 3 An example of uniform crossover

Mutation

On each selected chromosome from the previous phase, one points is randomly selected and then it changed to a random number between 1 and M. An example is shown in Fig .4.

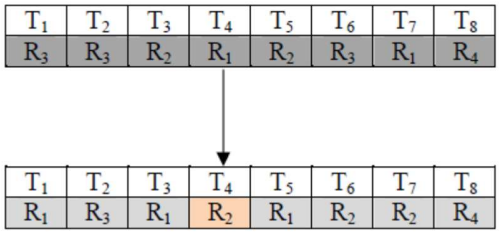

Figure. 4 An example of mutation

4.3 Phase 2: PSO Algorithm

In this phase we use PSO algorithm to achieve optimized solution for scheduling problem. In PSO, the system is initialized with a population of random solutions. This solutions are called particles.

In the proposed methods each particle represents a candidate solution for cloud scheduling problem in a vector form. Here a simple method has been used to represent particles. Each particle can be a vector. The length of each particle can be a number between 1 to N. N is total number of input tasks in the system. Each element of a particle can be a number between 1 to M, where M is total number of resources (machines) in system. Fig.5 shows an example of particle representation. For example, in the figure, task T2 in particle2 executed on resource M3, and task T3 in particlr2 executed on resource M4 and so on.

Particle\Task	T_1	T_2	T_3	T_4
Particle 1	M_1	M_2	M_3	M_4
Particle 2	M_1	M_3	M_4	M_2
Particle 3	M_3	M_4	M_1	M_2

Figure. 5. An example of particle representation.

PSO algorithms receive its inputs (initial population) from previous phase. To put it simply, PSO algorithm receive the output of GA as its inputs and try to find the best solutions as optimized final solution for scheduling problem.

PSO Fitness Functions

As mentioned in previous sections, our goal is to minimize makespan and also improve load balancing. So, we use fitness function (11) as fitness function to minimizing makespan.

$$\text{Fitness (Particle i)} = 1/\text{ makespan i} \quad (11)$$

So, Particle i is more appropriate if it has less makespan.
In addition to minimizing makespan, we have another goal which is improve load balancing. Sometimes several particles may be found that have similar makespans but don't all balance the load among their resources. Once again we use fitness function (12) to improve load balancing.

$$\text{Fitness (Particle i)} = 1/\text{ Cpu\_LB}i \quad (12)$$

Update Particles

At the next step, the velocity and new position of each particle are calculated according equation (13) and (14) . in this way a new population of particles is generated.

$$V_{id}^{n+1}=W_n \times V_{id}^{n}+C_1 r_1^n (Pbest_{id}^{n}-X_{id}^{n})+C_2 r_{id}^{n} (Gbest_{id}^{n}-X_{id}^{n}) \quad (13)$$

$$X_{id}^{n+1}=X_{id}^{n}+V_{id}^{n+1} \quad (14)$$

W is Inertia Weight and In our hybrid algorithm, HGPA, w is calculated by equation (15)

$$W_n=W_{max}-\frac{(W_{max}-W_{min})\times n}{iter_{max}} \quad (15)$$

After generation new population, some values in the position vectors may be floating point type. So we convert this value to integer values. In this way, the continues optimization problem can be converted to discrete optimization problem. Figure (6) shows the details flowchart of our proposed hybrid scheduling algorithm.

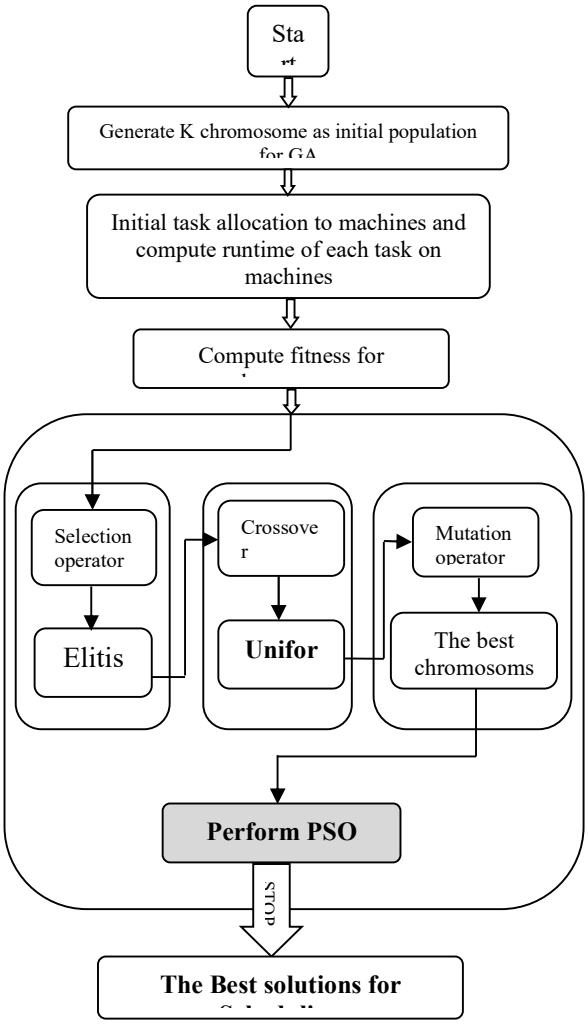

Figure 6. Flowchart of proposed algorithm.

5. Performance Evaluation

To evaluate the performance our proposed scheduling algorithm, HGPA, we used java programming. In these experiments we evaluate the performance of the HGPA and also compare HGPA with other heuristic based scheduling algorithms such as GA, PSO, CSO and CSO-GA. All the simulation are conducted on a computer system under windows 7 operating system, Intel Core i7 2.9 GHZ CPU with 4 GB RAM.

Simulation results are calculated with different conditions such as different number of iteration numbers, different number of tasks and different number of machines (resources).

In our proposed algorithm, we Assumed that the crossover rate CR = 0.50 and the mutation rate MR = 0.01 for GA algorithm. Also we assumed that the C1=2 and C2=2. For CSO algorithm we assumed all the parameters similar to [4].

Fig. 7 shows the performance of the five scheduling algorithms (HGPA, GA, PSO, CSO and CSO-GA) with different iteration numbers. In this diagrams we compare our proposed scheduling algorithm, HGPA, with other scheduling algorithms. In this experiment the number of iteration is variable from 100 to 2500 by an interval of 400. We used an ETC matrix which is computed with random value of computer processing power with values between 30 to 3000. Also the numbers of machines is 7 and the numbers of tasks is 60.

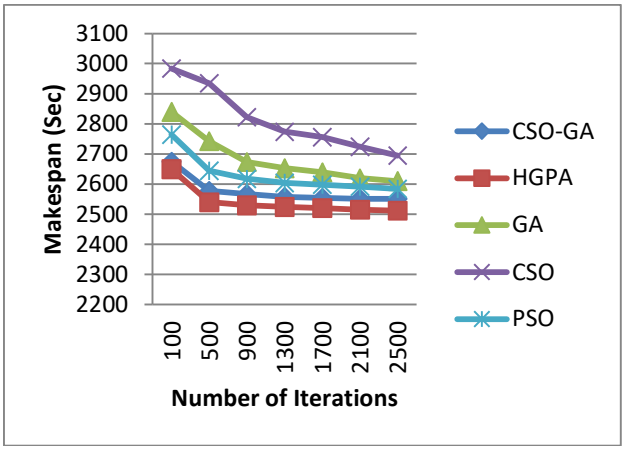

Figure. 7 Comparison of make spans with different iteration numbers

As shown in Fig. 7, our proposed algorithm, HGPA, which is a hybrid scheduling algorithm, perform better than other algorithm.

Fig. 8 shows the performance of the five scheduling algorithms with different numbers of machines (resources). In this experiment the number of machines is variable from 7 to 25.

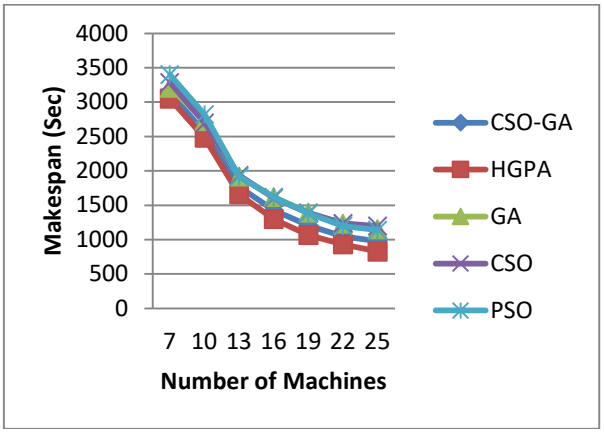

Figure. 8 Comparison of make spans with different numbers of machines (resources).

As shown in Fig. 8, when the number of machines increases, the makespan will decreases. This manner is observable in the result of all algorithms. This figure shows that HGPA has better performance and can produces a smaller makespan than the other algorithms.

Fig. 9 shows the performance of the five scheduling algorithms with different numbers of tasks. In this experiment the number of tasks is variable from 50 to 100.

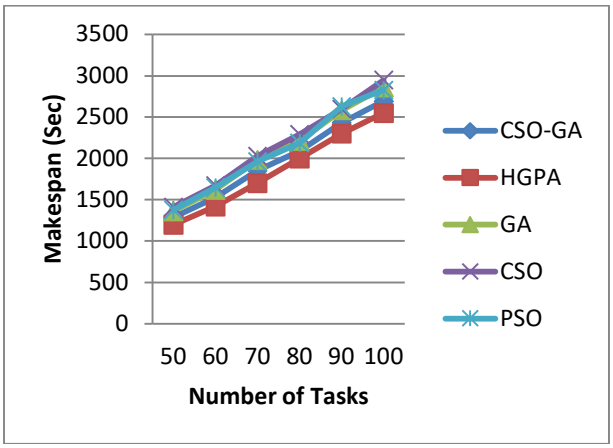

Figure. 9 Comparison of makespans with different numbers of tasks

As shown in Fig. 9, when the number of tasks increases, the makespan will increases. This manner is observable in the result of all algorithms. This figure shows that HGPA has better performance and can produces a smaller makespan than the other algorithms in such situations.

6. Conclusion

In this paper an efficient hybrid scheduling algorithm is presented called HGPA. HGPA is based on combination of Particle Swarm Optimization Algorithm (PSO) and Genetic Algorithm (GA). HGPA tries to benefit from advantages of both GA and PSO algorithms to achieve better performance, better load balancing and minimized makespan.

We evaluated our proposed method and compared it with other scheduling algorithms. Simulation results were calculated with different conditions such as different number of iteration numbers, different number of tasks and different number of machines (resources). The simulation results showed that HGPA can achieve smaller makespans than the other scheduling algorithms in different conditions.

Acknowledgement
The author gratefully acknowledges the financial support for this work that was provided by University of Islamic Azad University-Shoushtar branch.

References

[1] P. Mell and T. The NIST Definition of Cloud Computing (Technical report). National Institute of Standards and Technology: U.S. Department of Commerce. Grance (September 2011), doi:10.6028/NIST.SP.800-145. Special publication 800-145.

[2] Z Pooranian, M Shojafar, R Tavoli, JH Abawajy, M Singhal, A Hybrid Metaheuristic Algorithm for Job Scheduling on Computational Grids, Informatica 37 (2013) 157–164.

[3] J. D. Ullman, “Np-complete scheduling problems.” J. Comput. Syst. Sci, 10(3), 1975.

[4] S. Rouhi and E. Behrouzian Nejad, " CSO-GA: A New Scheduling Technique for Cloud Computing Systems Based on Cat Swarm Optimization and Genetic Algorithm", cumhuriyet Science Journal , Vol 36, No 4 , 2015.

[5]S, Xue , M. Li , X. Xu , J. Chen, An ACO-LB Algorithm for task scheduling inthe cloud environment, journal of software, Journal of Software, 9 (2), 466-473, 2014.

[6] J. Holland, “Adaptation in Natural and Artificial Systems,” University of Michigan Press, Ann Arbor, ISBN: 0-262-58111-6, 1975.

[7] J. Kennedy, R.C. Eberhart, “Particle swarm optimization”, Proc, IEEE Conf. Neural Netw., vol. IV, IEEE, Piscataway, NJ, 1995,pp.1942-1948.

[8] http://www.swarmintelligence.org/tutorials.php

[9] S. Javanmardi, M.Shojafar, D. Amendola, N. Cordeschi, H. Liu, A. Abraham, Hybrid Job Scheduling Algorithm for Cloud Computing Environment, Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014 pp 43-52

[10] Z. Pooranian, M. Shojafar, J. H. Abawajy, A. Abraham, "An Efficient Meta-heuristic Algorithm for Grid Computing", Springer, Journal of Combinatorial Optimization (JOCO), ISSN: 1382-6905, Impact Factor: 0.939, Vol. 30, Iss. 3, pp. 413-434, October 2015.

[11] Z. pooranian, A. Harounabadi, M. Shojafar, N. Hedayat, "New Hybrid Algorithm for Task Scheduling in Grid Computing to Decrease missed Task", WASET Journal, Vol. 55, pp. 5-9, July 2011.

[12] Z. Pooranian, M. Shojafar, J. H. Abawajy, M. Singhal, "GLOA: A New Job Scheduling Algorithm for Grid Computing, International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI), Vol. 2, No. 1, pp. 59-64, March 2013.

[13] H. Parvan, E. Behrouzian Nejad, S. E. Alavi, " New Hybrid Algorithms for Task Scheduling in Computational Grids to Decrease Makespan", International Journal of Computer Science and Network Solutions, Vol. 2, No.4, 2014.

[14] R.G. Babukarthik, R. Raju, and P. Dhavachelvan, “Hybrid Algorithm for Job Scheduling: Combining the Benefits of ACO and Cuckoo Search,” in Advances in Computing and Information Technology. Springer Berlin Heidelberg, pp. 479-490, 2013. 472 JOURNAL OF SOFTWARE, VOL. 9, NO. 2, FEBRUARY 2014
© 2014 ACADEMY PUBLISHER

[15] S.B. Zhan, H.Y. Huo, “Improved PSO-based Task Scheduling Algorithm in Cloud Computing,” in Journal of Information & Computational Science 9: 13,(2012), pp. 3821–3829,2012.

[16] S.Kaur, A.Verma, “An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment” in I.J. Information Technology and Computer Science, pp. 74-79, 2012.

[17] M. Rahman, X.R. Li, H. Palit, “Hybrid Heuristic for Scheduling Data Analytics Workflow Applications in Hybrid Cloud Environment,” in Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW),2011, IEEE International Symposium on IEEE, pp. 966-974, 2011.

[18] T.D. Braun., H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.L. Reuther, J.P. Robertson, M.D. Theys, B. Yao, D. Hensgen and R.F. Freund, “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems,”*Journal of Parallel and distributed Computing*, 61(6), pp. 680- 1983, 2001.

Ebrahim Behrouzian Nejad received his B.S., M.S. and Ph.D. degrees in Computer Engineering from Islamic Azad University in 2004, 2006 and 2011 respectively. Since 2005 up to now, he stayed in IAU-Shoushtar branch as a faculty member of Computer Engineering Department. His research interests are: Distributed systems, Cloud and Grid computing systems, Computer Network, Wireless Sensor Networks, Network Security and On-Chip Networks.