IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 1, No. 4, Dec 2012, pp. 171~181
ISSN: 2252-8938

Independent Task Scheduling in Grid Computing Based on Queen-Bee Algorithm

Zahra Pooranian*, Mohammad Shojafar, Bahman Javadi*****
* Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
** Electrical and Computer Department, Qazvin Islamic Azad University, Qazvin, Iran
*** School of Computing, Engineering and Mathematics, University of Western Sydney, Sydney, Australia

Article Info

Article history:

Received August 05th, 2012
Revised Oct 1st, 2012
Accepted Oct 16th, 2012

Keyword:

Grid computing
Scheduling
Queen-Bee
PSO
Genetic
SA

ABSTRACT

Grid computing is a new model that uses a network of processors connected together to perform bulk operations allows computations. Since it is possible to run multiple applications simultaneously may require multiple resources but often do not have the resources; so there is a scheduling system to allocate resources is essential. In view of the extent and distribution of resources in the grid computing, task scheduling is one of the major challenges in grid environment. Scheduling algorithms must be designed according to the current challenges in grid environment and they assign tasks to resource to decrease makespan which is generated. Because of the complex issues of scheduling tasks on the grid is deterministic algorithms work best for this offer. In this Paper, the Queen-bee algorithm is presented to solve the problem and the results have been compared to several other meta-heuristic algorithms. Also, it is shown that the proposed algorithm decline calculation time beside decreasing makespan compared to other algorithms.

Corresponding Author:

Mohammad Shojafar,
Electrical and Computer Department,
Qazvin Islamic Azad University,
Barajin, Qazvin, Iran. 34197-416, Tel: +98-281-366-5275 Fax: +98-281-366-5279.
Email: Shojafar@qiau.ac.ir

1. INTRODUCTION

The idea of task distribution has been considered from many centuries ago in order to achieve high speed and in other words, to save the time. In fact, Grid technology makes it possible to make use of the resources and decentralized systems and the interconnection of these systems. When the Grid technology invented for the first time, it just aimed at cooperating the resources of the system and provide a powerful system and generally it was at the disposal of research institutes. But today, there are higher expectations from the Grid and much more importance had been dedicated to it, especially in the e-commerce and the decentralized and distributed commercial systems. The modern Grids could be found in different organizations like scientific research and medicine detection organizations and also in the analysis of financial risks, Weather Forecast, designing, simulating, commercial intelligence and the transaction processing environments all over the world. In fact, Grids make use of the resources of the computers interconnected with the network and they could perform complex calculations using the resultant power of these resources. They do this by segmenting this operation and assigning each segment to a computer in the system.

One of the most important parts in the Grid systems is scheduler. Due to the vastness of grid and the assignation of each task to a specific computer and the necessity for each computer to assign a time to the grid, there is a need to use a scheduler. Scheduling plays the most important role in the improvement of grid's efficiency. Weak scheduling increases the execution time and therefore reduces grid's performance. Grid system performs hundreds or thousands jobs simultaneously and therefore making weak about the execution place could notoriously decrease the efficiency. But, effective scheduling or in other words, good decision – making about the execution place in NP-complete problem that faces with different challenges.

Different scheduling algorithms have been proposed for grid systems; such as, max-min, min-min[1] and because of the complexity of the scheduling problem, it has been shown that using heuristic algorithms is more suitable for this purpose, different algorithms such as GA (Genetic Algorithm)[2], PSO (Particle Swarm Optimization) [3], PSO-SA (Simulated Annealing) [4], TS (Tabu Search) [5], GA-TS [6], GA-SA [7] have been proposed in this area.

In GGA [8], A combination of two genetic and GELS Algorithms has been proposed to solve the problem of independent task scheduling. Since genetic algorithm works weakly in local searches, combining it with GELS Algorithm has improved this problem. In this algorithm, two factors including time and the number of missed tasks have been considered. On the other hand, GGA takes much time to analyze Tasks and if the tasks are increased enormously, it is unable to perform good results in task deadlines.

MSA [9] algorithm is a mutated SA algorithm to perform scheduling in the grid. This Algorithm has been applied to schedule the independent tasks and acts statistically. The difference between static and dynamic scheduling is that in static scheduling, all the data necessary for the tasks, processors, execution times and the number of processors are specified a priori. The change in this algorithm in comparison with standard SA is that in search neighboring step per temperature range more than one neighboring solution is provided. In other words, the mutated stimulation provides more than one solution and then selects one of them on the basis of its fitness function. This change provides for better solutions and increases the change for finding out the global optimization. Its weakness is searching in global problems because, it able to work in local search for tasks and most of grid tasks are assigned globally to the resources. Global searching means that algorithm is able to search whole space problem generally, but local searching could have one solution that searches part of space problem.

GSA [10] is a combination of genetic algorithm and SA for solving the problem of independent task scheduling. The main function of this algorithm is finding out a solution with the minimum execution time. Since the genetic algorithm searches the problem space globally and acts weakly in local searches, by mixing it with SA which is a local searching algorithm. It is tried to resolve this deficiency and this way a mixture of the advantages of these two algorithms have been used. In a contrary, although GSA is an combined algorithm which is formed by local and global search, It able to search problem suitable, but, its searching duration raised dramatically, and it is less beneficial for grid scheduling which is dynamic and needs to schedule before task deadlines.

In this Study, the Queen-bee [11] algorithm has been used for scheduling independent tasks in computing grids. In addition to this, also, four other heuristic algorithms have been used including GA, SA, GSA, PSO-SA and the results of the simulation of these algorithms have been shown.

The aim of this paper is to investigate a new biologic algorithm inspire of bee insects for resolving scheduling problem with minimized mean makespan and rune-time. The remainder of this paper is organized as follows. In Section 2, the problem studied in this research is described in detail. In Section 3, the structure of the proposed algorithms is explained. Then numerical tests are established to solve the problems in section 4. This is followed by a demonstration of the simulation results. Finally Section 5 presents a summary of the research with concluding remarks and recommendations for further research.

2. PROBLEM DESCRIPTION

The scheduling problem of independent tasks is a NP-hard problem that consists of N tasks and M machines. Each task should be processed by each M machine, as the Makespan is minimized. In other word, we have introduced a Deadline (D) for every task as each task should end its implementation before ending D. Each task can be just implemented on a resource and it is not stopped before finalizing its execution. We use the expected time to compute (ETC) matrix model. Since the proposed scheduling algorithm is as static, we have supposed that expected implementation time for each task, i, on each resources j, was determined before and was set on ETC matrix, ETC[i, j]. In this paper, we propose five meta-heuristic algorithms for the above problem. The framework of these algorithms is described in the next section.

3. THE PROPOSED ALGORITHMS

In this section, we have offered 5 Meta-heuristic algorithms to solve the scheduling problem of the grid.

3.1. Queen-Bee Algorithm

Human being is hard wired to achieve the best, so, the problem of optimization theory have been raised since a long time ago, but since in most cases recognizing and defining all the dominant conditions is impossible, instead of the best solution or the absolute optimal solution a satisfactory solution suffices. Therefore, due to human disability of optimization the improvement receives a specific value. In most cases the improvement is what is done for optimization. Optimization seeks to improve the performance in order to reach the optimal point or points. Algorithms which have been obtained inspiring the physical and biological rules are called heuristic algorithms. These algorithms are able to find out solution approximate with optimal solutions for the problems that there are or are not optimal solutions for them will the sound calculation time.

Genetic algorithms are the oldest type of Evolutionary algorithms has been widely used to solve the optimization problems. It was first proposed by John Holland in 1975[12], but Queen Bee [13] algorithm has been proposed in 2003 for the first time. Queen bee algorithm has got common concepts with genetic algorithms like, gens, chromosomes. Population crossover and mutation operators – This algorithm has two major differences with genetic algorithm. First in the normal genetic algorithm, a cost function is calculated for the initial population and the population is arranged in proportion with increase in this function.

Then some of the worst members of the population are cross out and the rest are selected which have lower cost for generating descendants. This means that equal numbers of parents are selected and using crossover algorithm they reproduce some offspring equal to the number of population disposed previously. So, the new offspring are replaced for the worst population.

In Queen Bee Algorithm, the stages of selecting the initial population and classifying them are on the basis of the cost function and disposing the worst similar genetic members. But in this algorithm just one mother is selected which is the Queen Bee and the queen produces a number of offspring by mixing with male counter parts using crossover operator. Therefore the number of marriages in Queen Bee is less than this number in genetic algorithm this leads to much speed rate in this algorithm in comparison with the genetic algorithm. But the high speed rate of convergence results in the emergence of this premature phenomenon. In this phenomenon, Instead of finding out the optimum result, the algorithm converges to a local minimum. One of them mutates with a normal mutation rate and the other one does so will strong mutation by Pm probability rate which is normally higher than Pm; i.e., P'm is higher than Pm. Therefore a variety is emerged in the offspring will be more and the premature convergence is avoided. The proportion of these two probabilities is defined equal to the four parameters. You could see the Pseudo-code of the algorithm in Figure 1.

```
// t: time //
// n: population size //
// p: populations //
// σ: normal mutation rate //
// Pm: normal mutation probability //
// P'm: strong mutation probability //
// Iq: a queen-bee //
// Im: selected bees //
01 t← 0
02 initialize P (t)
03 evaluate P (t)
04 while (not termination-condition)
05 do
06 t← t+1
07 select p (t) from P (t-1) (*)
08 P (t) = {Iq (t-1), Im (t-1)}
09 recombine P (t)
10 do crossover
11 do mutation (*)
12   for i = 1 to n
13    if i ≤ (σ x n)
14      do mutation with Pm
15    else
16      do mutation with P'm
17    end if
18   end for
19 evaluate P (t)
20 end
```

Figure 1. Queen-Bee Pseudo-code

3.2. Genetic Algorithm(GA)

After offering the genetic algorithm by Holland [12], this method was completed in 1989 by Goldberg [14]. In order to use genetic algorithm to solve task scheduling problem, a number of parameters must be defined. That is described in the following parts.

- **Showing Chromosomes**

Here, a simple method has been used to present chromosomes, so that real numbers are used to coding the chromosomes. So that the numbers of gens are random numbers between 1 to k. k is the resources N.O. and the length of chromosomes is considered equal to the size of the number of input tasks. Figure 2 show a sample of chromosomes arrangement which is assigned T_1 to the resource 2, R_2, in $chromosome_2$, and then the primary population of chromosomes is made randomly.

	T_1	T_2	T_3	T_4
$Chromosome_1$	R_1	R_2	R_3	R_4
$Chromosome_2$	R_2	R_3	R_4	R_2

Figure 2. The sample of chromosomes illustration

- **Fitness function**

The main purpose that is considered in all proposed algorithms in this paper for scheduling problem is to make it possible to minimize makespan. This time is considered equal to the maximum completion time for each task, i, per resource, j, in addition to the Time – Finish [j] that is calculated as equations (1) and (2) as follow:

$$\text{Fitness}(\text{chromosome}_i)= \frac{1}{\text{makespan}(\text{chromosome}_i)} \quad (1)$$

$$\text{Makespan}=\text{Max}(\text{Time\_Finish}[j]+ \text{ETC}[i,j]) \quad 1\leq i\leq N , 1\leq j\leq M \quad (2)$$

- **Selection Operator**

Before using mutation and crossover operators, it is the selection step. Here tournament operator has been used of chromosome selection.

- **Crossover Operator**

Two points crossover operator has been used in the proposed algorithm. Therefore, two random points over two selected chromosomes have been chosen by previous level and the gen between these two points are transferred in chromosome (Figure 3).

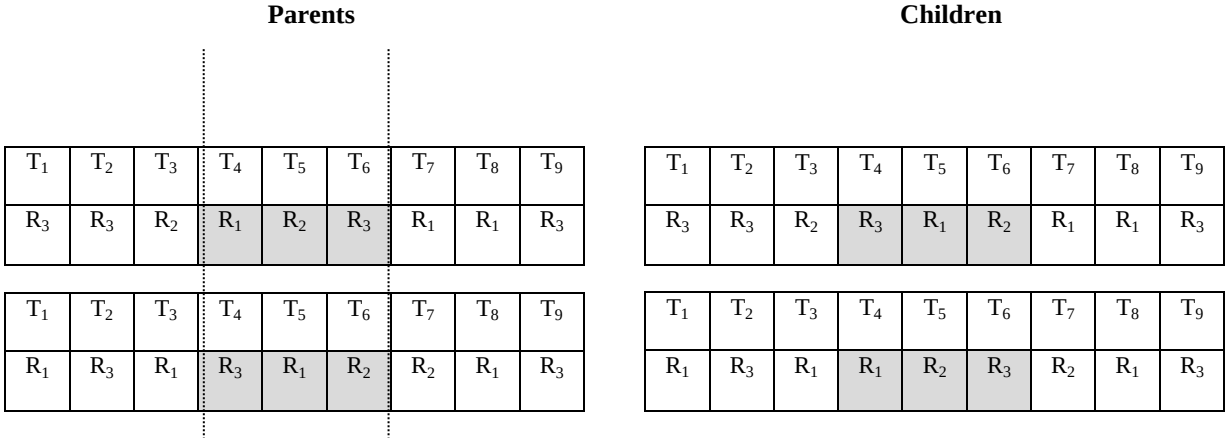

Figure 3. A sample of operation for two-point crossover

- **Mutation Operator**

In the previous level, one point is selected randomly over each chromosome and one random number in the range of 1 to M is created. Figure 4 shows a sample of this operation.

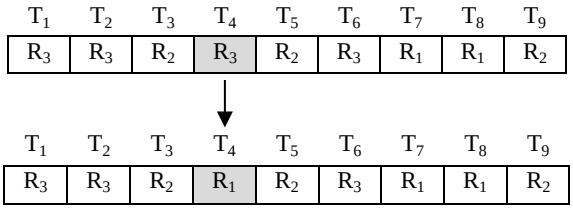

Figure 4. A sample of Mutation Operating

- **Termination Condition**

Algorithm finishes when the iteration time of the algorithm reaches its maximum. The best chromosome is selected as a solution for scheduling issue.

3.3. Simulated Annealing (SA)

SA method is one of the possible algorithm methods that have been offered for solving optimization problem by Kirkpatrick [15] in 1983 which has a big searching space. SA method has been originated from cooling metals. In this method, if each searching space is supposed to be s. Each s' state from searching space is a reply to the problem. The problem begins with a primary state and moves gradually towards the optimum solution by transferring from one state to another in searching space. In each iteration, SA algorithm of s' state is selected as a neighbor and moves towards the next state from the existing possibility to the existing state or remains in the existing state. This trend continues till the relatively optimum solution is found out or it continues ill the maximum iteration time of the algorithm is resulted. Accepting the neighboring state as a reply to this problem is based on a probability. Accepting the neighboring state is considered as a solution for the problem based on a probability. If the cost of neighboring state is better than that of the existing state, the neighboring state will be accepted as a solution. The cost of this algorithm is the criteria and parameter that is used in searching a solution for the problem. So, it could go beyond the scope of the local optimum solution. In SA method, the temperature parameter (T) is used for achieving the acceptance probability of the neighboring state, so that first the maximum number of neighbors is selected as a solution and this temperature gradually reduced by increasing the number of iterations, so that before the finishing time of maximum iterations, the algorithm execution equals to zero. Therefore, increasing the number of iterations leads to a decrease in the possibility of accepting the neighboring state that has no better cost saving advantage for the problem. How T parameter reduces in each step and how to reach the neighboring state in this algorithm and also determining the primary amount for the temperature parameter (T) and the maximum number of iterations of max algorithm is considerably important. The Pseudo-code of SA algorithm has been shown in Figure 5. Showing the primary solution in the algorithm is similar to Figure 1. Also the cost algorithm is like Equation 1.

```
t= T0
Initial Solution= S0
Best solution= S0
OF0= Object Function(S0)
OF Best Solution= OF0
For i=1 to Max do {
    S1= Generate neighbor(S0)
    OF1= Object Function(S1)
    If OF1 ≥ OF0 then {
        S0= S1
        OF0 = OF1
        If OF1 ≥ OF Best Solution then {
            Best Solution= S1
            OF Best Solution= OF1    }
                }
        Else {
            Accept= e^-((OF0-OF1)/t)
            r= Random(0,1)
            if ( r > Accept) then {
                S0= S1
                OF0 = OF1        }
            }
    T= 0.95 *t    }// end For
Return (Best Solution)
```

Figure 5. Simulated Annealing Pseudo-code

3.4. GSA

In this part, hybrid GSA algorithm has been formed by mixing genetic and SA algorithms. Since the genetic algorithm searches the space of the problem globally and searching local space it hasn't any ideal performance, mixing it with SA algorithm which is a local one, it has been tried to improve this weak point and this way a mixture of the advantages of these two algorithms has been utilized. The illustration form of chromosomes is like Figure 1; also, fitness function is like Equation 1. The hybrid schema has been shown in Figure 6. As it is specified in the figure instead of assigning each number of populations to SA, after finishing the genetic algorithm, the best solution is delivered to SA algorithm so that a solution is made for it. This could reduce the calculation cost.

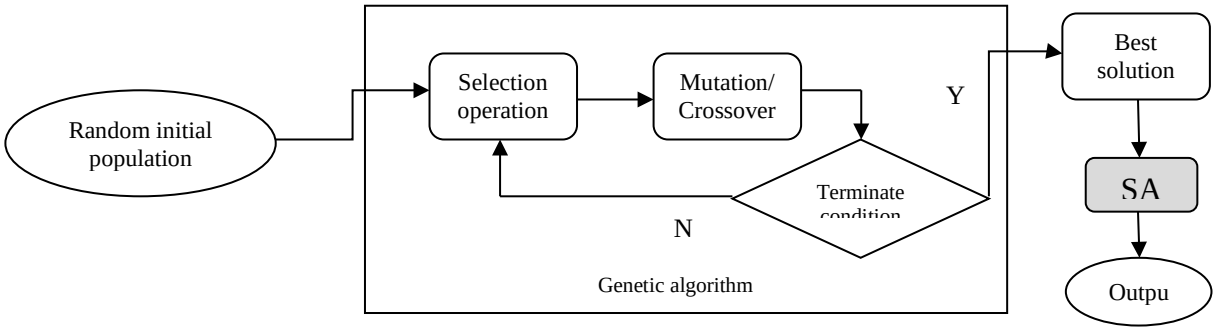

Figure 6. Hybrid proposed for Resolving Scheduling Problem Methodology

3.5. PSO-SA

PSO method has been offered by Abrhat and kennedy [16] in 1995. This method has been inspired by behavior of bird flocking and schooling fish. In this algorithm, we have a swarm of the particles; the structure of this algorithm is as follow:

- **Initialize Swarm**

The first issue in using PSO to solve optimization problem is to create a correspondence between the problem and particle vector.

The dimensions of the problem are equal to the number of input tasks. So the length of each particle and speed vector is considered equal to the number of tasks – the amounts in each particle has been considered as an integer random number between 1 to K. You see a population of particles in Figure 7. So that T_4 is executed on resource, R_4, in particle$_1$. Then, the initial population is produced randomly and then for each particle an initial velocity vector is produced in which the amounts in the speed vector interval have been defined:

	T_1	T_2	T_3	T_4
Particle 1	R_3	R_1	R_3	R_4
Particle 2	R_4	R_1	R_1	R_2
Particle 3	R_2	R_1	R_4	R_2

…

Figure 7. Particle Presentation

- **Fitness Function**

In order to estimate each particle defined fitness function in equation 1 has been used. The fitness amount is calculated for each particle and if the fitness amount of each particle is less than that of pbest of each particle; new coordinate is set in pbest. It is clear that in the first moment, the coordinate of each particle is considered as pbest. The best pbest is considered as gbest.

- **Update of the Particles Position**

After producing the initial particle population as X^i, a position vector has its own velocity and fitness amount. In each iteration, the algorithm of position ad velocity amounts changes by Equations (3) and (4).

$$V_{i+1} = \omega V_i + C_1 rand_1(pbest_i - X_i) + C_2 rand_2(gbest_i - X_i) \quad (3)$$

$$X_{i+1} = X_i + V_{i+1} \quad (4)$$

In the above mentioned equation, ω is a factor of inertia weight that is obtained by Equation (5), pbest is the optimum position of the before particle and gbest is the best position of the previous position of all particles in all previous steps, V_i is position and the velocity of i^{th} particle, $rand_1$, $rand_2$ are two random numbers, and C_1& C_2 are two constant indices.

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}}.iter \quad (5)$$

Where, ω max: Initial value of weighting coefficient;
ω min: Final value of weighting coefficient;
itermax: Maximum of iteration;
iter: Current iteration;

After producing the new population, it is possible that the amounts in the position vector are decimal position amounts which are invalid amounts for resources number. So, in the offered algorithm, the nearest obtained decimal amount is rounded to the nearest integer number. This trend keeps on going so that the algorithm reaches its maximum iteration number.

- **Update gbest by SA**

After finishing PSO algorithm, gbest is obtained from it is considered as the best solution to this problem. Since PSO algorithm works weakly in the local search, in order to keep away from falling into a local optimum, achieved gbest from PSO is given to SA in order to produce a neighboring solution.

4. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, the result achieved from proposed algorithm is described and it is compared with other algorithms. Here, performance measurement is considered makespan minimizing. All algorithms are simulated in java environment on a PC with CPU 2.66 GHZ and RAM 4GB. Table 1 illustrates initialized parameters in proposed algorithm. The simulated results are shown in Table 2.

Here, achieved results are surveyed for 100 and 300 iterations. Also, task numbers are changed in iteration between 50 to 500 and resource numbers between 10 to 30 resources. As a result, Queen-bee algorithm provides better result compared SA, GA, and GSA. Moreover, it produces less makespan compared to combined algorithm (PSO-SA) in three parameters.

Table 1. Tuned Values of the Parameters of the Algorithms

Algorithms	Parameter	Value
PSO	V_{max}	number of resources
	C_1,C_2	1
	Initial Velocity	[1, W_{max}]
	ω_{max}	0.9
	ω_{min}	0.1
Genetic	P-Crossover	0.85
	P-Mutation	0.02
Queen-Bee	P_m	0.01
	P'_m	0.6

Table 2. Comparison of makespan produced by different algorithms

iteration	(task, resource)	SA	GA	GSA	PSO-SA	Queen-bee
100	(50,10)	136.742	99.198	95.562	89.586	93.5
	(50,20)	98.944	62.496	60.968	53.266	60.85
	(50,30)	71.442	50.53	49.476	40.45	45.0
	(100,10)	307.738	183.49	190.353	167.33	177.83
	(100,20)	190.862	111.1742	111.646	100.714	105.62
	(100,30)	138.632	99.25	89.822	73.318	73.71
	(300,10)	973.728	638.082	597.8	511.532	521.0
	(300,20)	585.848	352.698	337.648	288.962	315.25
	(300,30)	384.928	262.166	256.664	216.402	248.66
	(500,10)	1837.662	1105.56	1072.362	887.195	1018.33
	(500,20)	833.996	602.174	571.796	504.33	556.33
	(500,30)	721.596	449.73	446.646	352.978	419.75
300	(50,10)	131.12	89.486	86.98	87.694	84.6
	(50,20)	74.832	60.87	57.304	53.77	52.75
	(50,30)	62.055	47.637	42.932	41.208	45.14
	(100,10)	233.2	172.628	179.062	167.16	167.57
	(100,20)	173.116	111.946	105.314	100.098	94.6
	(100,30)	120.452	90.716	87.846	70.572	79.0
	(300,10)	911.68	570.466	532.968	526.71	541.5
	(300,20)	523.33	327.522	337.428	296.168	343.33
	(300,30)	408.714	253.132	246.48	205.496	246.33
	(500,10)	1492.616	1071.014	1037.942	896.586	959.16
	(500,20)	893.262	602.134	593.116	493.9	557.0
	(500,30)	626.162	430.32	412.7152	339.331	400.25

In Table 3 illustrates implementation time for proposed algorithms for 100 and 300 iterations. In fact, *Queen-bee algorithm* consumes less time rather than others. Hence, although it is important to run tasks in allotted resources in least possible time and based on grid environment is dynamic, proposed algorithm is more suitable compared to others.

Table 3 . Algorithms Compare in Terms of Runtime

iteration	(task, resource)	SA	GA	GSA	Queen-bee
100	(50,10)	0	2.6	3	2
	(50,20)	0	3.4	3.4	2
	(50,30)	0	3.4	3.6	2
	(100,10)	0.2	8.8	5.6	5
	(100,20)	0.8	5.4	6.6	5
	(100,30)	1.4	6	7.4	4
	(300,10)	1	13.4	16.2	12
	(300,20)	1.6	15.4	18.4	12
	(300,30)	2.4	16	20	13
	(500,10)	1.4	21.2	25.8	19
	(500,20)	2.8	22.2	30	20
	(500,30)	3.4	26	31.8	22
300	(50,10)	0.4	28.2	10.6	6
	(50,20)	1.4	29.2	12.4	8
	(50,30)	1	26.6	12.8	7
	(100,10)	1	28.2	17	10
	(100,20)	2	44.6	20	12
	(100,30)	1.8	48.8	21.4	13
	(300,10)	2	105.4	48.2	32
	(300,20)	3.8	99	55	38
	(300,30)	3.8	108.6	58.2	39
	(500,10)	3.2	90.8	78.6	53
	(500,20)	5.4	111	90.4	65
	(500,30)	6.2	176.6	93.2	65

Figure 8 provides comparison of makespan between algorithms for 300 iterations in 50 independent tasks with {10, 20, 30} resources.

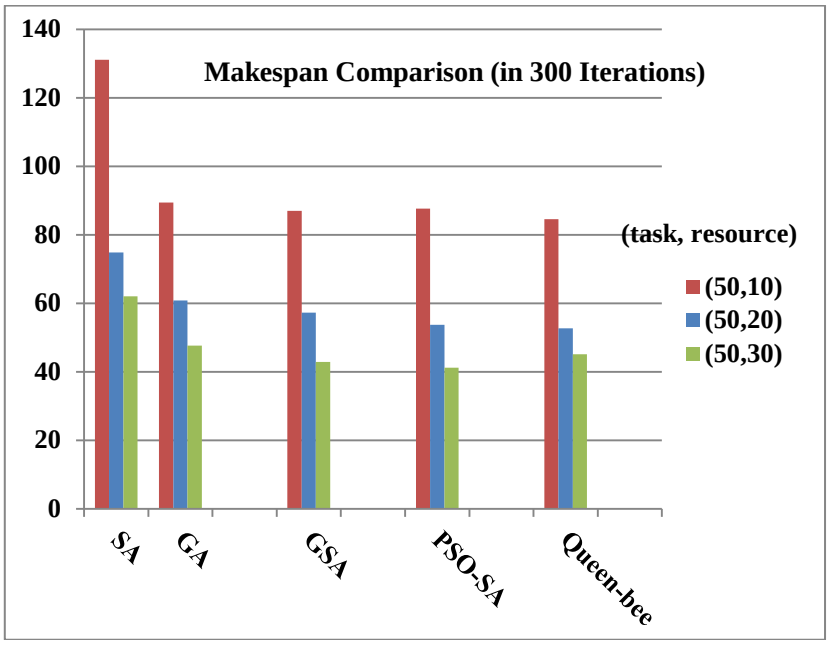

Figure 8: Makespan Comparison among algorithms for 300 iterations

In Figure 8, PSO-SA is the best (lowest makespan) for (50, 10) state and SA is the worst one. Besides, while resources increase and reached to its double one, Queen-bee (proposed method) is the best one, although SA is fallen dramatically but it still the worst one. But, while resources are reached to 30, PSO-SA will be the best, because, if we have sufficient resources PSO will be free to use them and allocate tasks to these 30 resources in less waiting time for each task in the queue of each resources. Moreover, while we have tested 50 tasks in 10 resources (50, 10) in SA algorithm, we have seen makespan has just upper than 130 units but rest of them has less than this amount. When resources increase, Queen-Bee algorithm declines makespan more than half of its amount compared to other algorithms (just upper than 40 units).

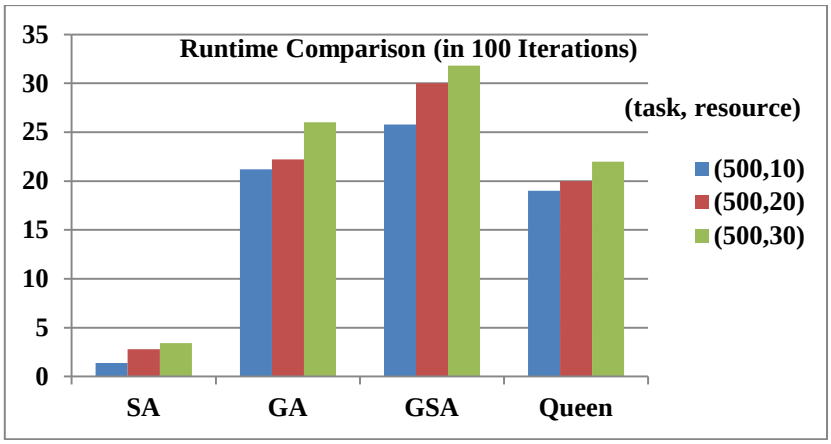

Figure 9: Runtime Comparison among algorithms for 100 iterations

Figure 9 illustrates runtime of GA, SA, GSA, and Queen Algorithms for 500 tasks in different resources in 100 iterations. In Figure 9, GSA is the worst time consumer and SA is the least time consumer. Although, SA has the best solution among these algorithms but it produce worst makespan, because, it would locally in problem space and it could not search space of the problem clearly, hence, it could not produce an optimum result for makespan, so, we do not consider this method among others. As a result, Queen is the best method among others in runtime. As is shown, while free resources are grown, all algorithms takes more time to search and allocate tasks among the queue of resources. The rate of increment in Queen is less than others (approximately 4 seconds). For example, in (500, 20) state, runtime in GSA is 30 seconds; in GA just upper than 20 seconds, and in Queen is just lower than 20 seconds.

5. CONCLUSION

Nowadays, increase the complexity and the necessity to improve the calculating power is considered as a characteristic of scientific issues. Therefore, creating and optimizing the computing grid systems have been considered very much. Scheduling in the grid is one of the most important problems in determining the efficacy. Considering that the grid scheduling is a nondeterministic problem, hence, deterministic algorithms are not suitable. There are many heuristic methods to improve scheduling in the grid that could be utilized.

In this paper, Queen-Bee algorithm was used for scheduling the independent tasks and was compared with four other offered algorithms. The calculation results showed that the proposed algorithm produces less makespan then GA, SA, GSA algorithms and it is also more suitable regarding the execution time.

REFERENCES

[1] M. Shojafar, *et al.*, "A new Method on Resource Scheduling in grid systems based on Hierarchical Stochastic Petri net, " in *3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)*, 2010, pp. 175-180.
[2] K., Jammu, "Reliability-Aware Genetic Scheduling Algorithm in Grid Environment," in *2011 International Conference on Communication Systems and Network Technologies*, 2011.
[3] Q. Tao, *et al.*, "A rotary chaotic PSO algorithm for trustworthy scheduling of a grid workflow," *Journal Computers and Operations Research, Elsevier*, vol. 38, Issue. 5, pp.824-836, 2011.
[4] R. Chen, *et al.*, "Combined discrete particle swarm optimization and simulated annealing for grid computing scheduling problem," in *ICIC'09 Proceedings of the Intelligent computing 5th international conference on Emerging intelligent computing technology and applications*, 2009, pp. 242-251.
[5] J.M. Garibaldi, *et al.*, "Fuzzy Grid Scheduling Using Tabu Search," in *Fuzzy Systems Conference, FUZZ-IEEE 2007*, 2007, pp.1-6.
[6] F. Xhafa, *et al.*, "A GA (TS) Hybrid Algorithm for Scheduling in Computational Grids," in *HAIS '09 Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, 2009, pp. 285 – 292.
[7] B. Singh, *et al.*, "HybridS GSA: SexualGA and Simulated Annealing based Hybrid Algorithm for Grid Scheduling," *Global Journal of Computer Science and Technology*, vol. 10, Issue 9, pp. 78-81, 2010.
[8] Z. Pooranian, *et al.*, " New Hybrid Algorithm for Task Scheduling in Grid Computing to Decrease missed Task," *World Academy of Science, Engineering and Technology*, vol. 79, pp. 924-928, 2011.
[9] A. Kazem, *et al.*, "A Modified Simulated Annealing Algorithm for Static Scheduling in Grid Computing," in *International Conference on Computer Science and Information Technology, ICCSIT 2008*, 2008, pp. 623-627.
[10] A. Tamilarasi, *et al.*,"An Enhanced Genetic Algorithm with Simulated Annealing for Job-Shop Scheduling," *International Journal of Engineering, Science and Technology*, vol. 2, No. 1, pp. 144- 151, 2010.
[11] S.L.D. Qin, *et al.*, "A Queen– Bee Evolution Based on Genetic Algorithm for Economic Power Dispatch," in *39th International Universities Power Engineering Conference, UPEC 2004*, 2004, pp. 453– 456.
[12] J. Holland, Adaptation in Natural and Artificial Systems, *MIT Press Cambridge*, ISBN: 0262581116, 1975, pp. 228.
[13] S.H. Jung , "Queen-bee evolution for genetic algorithms," *Electronics Letters*, vol. 39 , Issue 6 pp. 575 – 576, 2003.
[14] D. E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, *Addison-Wesley*, ISBN: 0201157675, 1989, pp. 432.
[15] S. Kirkpatric, *et al.*, "Optimization by simulated annealing", *Science*, vol. 220, No. 4598, 1983.
[16] R. Eberhart, *et al.*, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Sym. Micro-machine and Human Science*, 1995, pp. 39-43.

BIBLIOGRAPHY OF AUTHORS

Zahra pooranian received her Msc in Computer Architecture degree as honor student in Dezful Islamic Azad University since 2011. She is an instructor in Sama University in Dezful and Ahvaz since 2009. Her research interest in Grid computing specially in resource allocation and scheduling. She has worked on several papers in decreasing time and makespan in grid computing by using several AI methods such as GA, GELS, PSO, and ICA. She has published more than 5 papers especially in grid scheduling and resource allocation in various conferences, such as WASET 2010-11, ICCCIT 2011, and ICEEE 2011.

Mohammad Shojafar Received his Msc in Software Engineering in Qazvin Islamic Azad University, Qazvin, Iran in 2010.Also, he Received His Bsc in Computer Engineering-Software major in Iran University Science and Technology, Tehran, Iran in 2006. Mohammad is Specialist in Network Programming in Sensor field and Specialist in Distributed and cluster computing (Grid Computing and P2P Computing) and AI algorithms (PSO, LA, GA). He Published two Journals in IEEE, 4 papers in IEEE, 2 papers in WASET, and two papers in WORLDCOMP Conferences Series held in USA till now. Now, Mohammad is a Faculty of Somesara Islamic Azad University in Iran. Moreover, He is a system analyzer in FFSDB project in Exploration Directorate in N.I.O.C.

Dr Bahman Javadi is a Lecturer in Networking and Cloud Computing at the University of Western Sydney. Prior to this appointment, he was a Research Fellow at the University of Melbourne, Australia. From 2008 to 2010, he was a Postdoctoral Fellow at the INRIA Rhone-Alpes, France. He received his MS and PhD degrees in Computer Engineering from the Amirkabir University of Technology in 2001 and 2007 respectively. He has been a Research Scholar at the School of Engineering and Information Technology, Deakin University, Australia during his PhD course. He has served on program committees for multiple international conferences and workshops and was co-guest editor of a special issue of the Journal of Future Generation Computer Systems on Desktop Grids.