International Journal of Computer Applications Technology and Research
Volume 5–Issue 10, 641-647, 2016, ISSN:-2319–8656

Proposing a New Job Scheduling Algorithm in Grid Environment Using a Combination of Imperialist Competition Algorithm (ICA) and Gravitational Emulation Local Search (GELS)

Arman Alizadeh
Department of Computer
Science and Research Branch
Islamic Azad University, Kish, Iran

Ali Harounabadi
Department of Computer
Islamic Azad University
Central Tehran Branch

Mehdi Sadeghzadeh
Department of Computer
Islamic Azad University
Mahshahr,Iran

Abstract: Grid computing is a hardware and software infrastructure and provides affordable, sustainable, and reliable access. Its aim is to create a supercomputer using free resources. One of the challenges to the Grid computing is scheduling problem which is regarded as a tough issue. Since scheduling problem is a non-deterministic issue in the Grid, deterministic algorithms cannot be used to improve scheduling. In this paper, a combination of imperialist competition algorithm (ICA) and gravitational attraction is used for to address the problem of independent task scheduling in a grid environment, with the aim of reducing the makespan and energy. Experimental results compare ICA with other algorithms and illustrate that ICA finds a shorter makespan and energy relative to the others. Moreover, it converges quickly, finding its optimum solution in less time than the other algorithms.

Keywords: Grid computing, scheduling, artificial intelligence algorithm, imperialist competition algorithm (ICA), Local search algorithm following the gravitational attraction

1. INTRODUCTION

Application of a new technology, or scientific evolution, requires scientific proof and practical implementation. Because it is time-consuming to implement practical research and mistakes can arise because of in attention to problems in theoretical subjects, there is a need to use simulations in some contexts instead of real implementations. The computations that are needed to simulate and study all aspects of scientific research projects require a significant amount of computational power that a single computer would take too much time to provide. Superscalar computers, vector processors, and pipeline processing were proposed to address this problem. Although they provide more computational power and greater speed than a single computer, technological limitations related to speed and the high cost of their design and manufacture make them available only to users with no financial limitations. Grid computations, which are based on distributed systems, were proposed to solve such problems. Grid systems have been proposed as a solution overcoming the limitations of hardware availability and computer locations, so that unused computers and their computational power can be exploited [1]. Grid computing systems are well-known for solving complicated large-scale problems in science, engineering, and finance [2], and have provided a wide range of heterogeneous and distributed resources for data-intensive computations [3].

In recent years, grid computing has been the subject of much research and has been used in commercial environments [4]. A resource management system (RMS) is the most important component of grid computing; it has a significant role in controlling and supervising the usage of resources. The most important function of an RMS is to schedule incoming tasks, assigning them to available compatible resources [5]. However, the heterogeneous and dynamic state of resources in grid systems poses difficulties, particularly when combined with complex task scheduling. Deterministic algorithms don’t have the necessary efficiency to solve these scheduling problems, so a considerable amount of research has been devoted to using heuristic algorithms such as genetic algorithms (GAs) [6], simulated annealing (SA)[7], particle swarm optimization (PSO) [8], ant colony optimization (ACO) [9], Queen-Bee Algorithm [10], tabu search (TS) [11], and various combinations of these [12, 13, 14, 15, 16] to produce better results in reasonable time. The heuristic ICA [17], proposed in 2007, was inspired by the sociopolitical evolution of imperial phenomena and has been used for solving many optimization problems in continuous space. This paper proposes a discrete version of ICA for solving the independent task scheduling problem in grid computing systems. The present paper converts ICA from a continuous state algorithm to a discrete state algorithm by changing the assimilation stage. The resulting algorithm is compared with GA and other heuristic algorithms and is shown to produce better results than these. This algorithm simultaneously considers makespan and completion time and energy by using appropriate weights in the mean total cost function.

2. BACKGROUND

2.1 Scheduling Method

The scheduling task issue is considered as a tough challenge which is composed of n tasks and m resources. Each task must be processed by a machine and does not top until the end of the performance. We used ETC matrix model described in [18]. The system assumes that the expected execution time for each task i, on every resource j is predetermined and is located in the matrix ETC, ETC [i, j]. Here, makespan is regarded as the maximum completion time in CompleteT [i, j], calculated in the following equation (1) [19]:

$$\text{Makespan} = \text{Max}\,(\text{completeT}\,[i,j])\ \ 1 \le I \le N,\ 1 \le j \le M \qquad (1)$$

In the above equation, CompleteT[i, j] is equal to the time when the task i on the source j is completed and it is calculated in equation (2):

$$\text{completeT}\,[i,j] = \text{Ready}\,[M] + \text{ETC}\,[i,j] \qquad (2)$$

2.2 Imperialist Competition Algorithm

The imperialist competition algorithm (ICA) proposed by (Atashpaz -Gargari and Lucas 2007) is a mathematical modeling of imperialist competitions. Similar to other evolutionary algorithms, the ICA algorithm begins with a number of initial random populations. Each random population is called a country. The possible solutions for ICA are called countries. A number of the best elements of the population are selected as imperialists, others which are governed by Imperialists are called colonies. By applying an assimilation policy in the direction of various optimization axes, imperialists gain the favor of their colonies. The total power of each empire is modeled as the sum of the imperialist power and a percentage of the mean power of its colonies. After the initial formation of empires, imperialistic competition starts among them. Any empire that has no success in the imperialistic competition with nothing to add to its power is eliminated from the competition. So the survival of an empire depends on its power to assimilate competitor's colonies. As a result, the power of greater empires is gradually increased in imperialistic competitions and weaker empires will be eliminated. Empires have to make improvements in their colonies in order to increase their power. For this reason, colonies will eventually become like empires from the point of view of power, and we will see a kind of convergence. The stopping condition of the algorithm is having a single empire in the world.

2.3 Local Search Algorithms Following Gravitational Attraction

In 1995, Voudouris and his colleagues [20] suggested GLS algorithm for searching in a searching space and NP-hard solution for the first time. In 2004, Vebster [21] presented it as a strong algorithm and called it GELS algorithm. This algorithm is based on gravitational attraction and it imitates the process of nature for searching within a searching space. Each response has different neighbors which can categorize based on a criteria which is depended on the problem. Obtained neighbors in each group are called neighbors in that dimension. For each dimension, a primary velocity was defined which each dimension has much primary velocity has more appropriate response for problem. GELS algorithm calculated gravitation force within responses in a searching space in two ways. In the first method, a response is selected from local neighbor space of current response and the gravitation force between these two responses was calculated. In the second method, the gravitation force among all of the neighbor responses in a neighbor space of current response was calculated and it is not limited to one response. In the movement into searching space, GELS algorithm implements in two methods: the first method is allowed movement from current response to in local neighbor spaces of current response, the second method is allowed movement to the responses out of local neighbor spaces of current response in addition to allowed neighbor responses of current response. Each of these transference methods can be applied with each accounting methods gravitation force, thus, four models are created for GELS algorithm.

2.3.1 Parameters Used in GELS Algorithm

(a) Max velocity: Defines the maximum value that any element within the velocity vector can have used to prevent velocities that became too large to use.

(b) Radius: Sets the radius value in the gravitational force formula; used to determine how quickly the gravitational force can increase (or) decrease.

(c) Iterations: Defines a number of iterations of the algorithm that will be allowed to complete before it is automatically terminated (used to ensure that the algorithm will terminate).

(d) Pointer: It is used to identify the direction of movement of the elements in the vectors.

2.3.2 Gravitational Force (GF)

GELS algorithm uses the formula (3) for the gravitational force between the two solutions as

$$F = G + \frac{(CU - CA)}{R^2} \tag{3}$$

Where

G = 6.672 (Universal constant of gravitation)

CU = objective function value of the current solution

CA = objective function value of the candidate solution

R = value of radius parameter

3. PREVIOUS RESEARCH

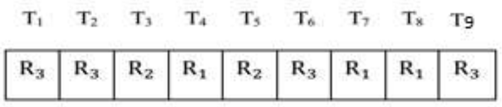
A large number of heuristic algorithms have been proposed for grid scheduling. Most of them try to minimize the maximum completion time of tasks, or makespan. Each task has its own deadline, and we try to decrease the makespan in order to prevent tasks from failing to execute because of their deadlines. That is, decreasing the makespan results in the ability to execute more tasks in the network. It also helps to provide efficient resource allocation and energy utilization.

The hierarchic genetic strategy (HGS) algorithm was proposed in [22] for scheduling independent tasks in a grid system and is implemented in dynamic grid environments in batch mode. This algorithm simultaneously considers optimization of flow time and makespan. The authors generate root nodes based on two other algorithms: longest job to fastest resource and shortest job to fastest resource (LJFR-SJFR) [23] and minimum completion time (MCT) [24], and they generate the rest of the population stochastically. In LJFR-SJFR, initially, the highest workload tasks are assigned to machines that are available. Then the remaining unassigned tasks are assigned to the fastest available machines. In MCT [24], tasks are assigned to machines that will yield the earliest completion time.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by [25]. The research aims to minimize the computation time of job executing in Taiwan UniGrid environment which focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chooses optimal resources to process the submitted jobs by applying the local and global pheromone update technique to balance the system load. Local pheromone update function updates the status of the selected resource after job has been assigned and the job scheduler depends on the newest information of the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after the completion of the jobs. By using these two update techniques, the job scheduler will get the newest information of all resources for the next job submission. From the experimental result, BACO is capable of balancing the entire system load regardless of the size of the jobs. However, BACO was only tested in Taiwan UniGrid environment.

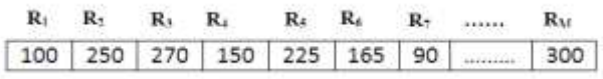
4. THE PROPOSED ALGORITHM

The proposed algorithm for task scheduling is a combination of ICA and GELS which is used for scheduling of independent tasks in computational grid environment. According to ICA's high performance in scheduling problems, a simple method for country representation is taken into consideration firstly. Natural numbers are used to encrypt countries. The value in each country is resource number ranging from 1 to M, where M is the number of all resources. Fig.1 illustrates an example of countries representation where 9 tasks are assigned to 3 resources. As Figure depicts, for example task 4 (T_4) is running on resource 1(R_1).



T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T9
R_3	R_3	R_2	R_1	R_2	R_3	R_1	R_1	R_3

Fig. 1. Country representation in ICA-GELS Algorithm.



R_1	R_2	R_3	R_4	R_5	R_6	R_7	......	R_M
100	250	270	150	225	165	90	....	300

Fig. 2. The amount of consumed energy for each task time unit.

$P_{j,i}$ is considered as a 1*M array where M is the number of resources. The value in each entry represents the amount of consumed energy for executing a task. Fig 2 shows a sample matrix where the consumed energy per each task time unit in second resource (R_2) is 250 J. The initial population of countries in the ICA is generated randomly. A random number between 1 and M is generated which is the resource number and the intended task will be execute on it. The independent task scheduling problem includes M tasks and M machines. Each task should be processed by one of the machines in way that makespan get minimum. Proposed algorithm takes two QoS parameters such as energy and time limitation into consideration. Each task can be execute on one resource and will not stop until execution is completed. Our algorithm applies ETC matrix model. Considering that proposed scheduling algorithm is static, it is assumed that expected execution time is determined for each task on each resource at prior and it is stored in ETC [i, j]. Also, Ready time [M] determines the time that machine M completes the previous assigned task. Makespan is considered as maximum completion time ($\text{completeT}(i,j)$), which is computed using equation (4).

$$\text{Makespan} = \text{Max}\,(\text{completeT}\,[i,j]) \tag{4}$$

completion _time $[i,j]$ is the time when task i completes on resource j and it is computed using equation (5) and TransferC and wait (i, j) are respectively the data transfer time and waiting execution time.

$$\text{completeT}\,(i,j) = \text{ETC}\,[i,j] + \text{TransferC} + \text{wait}\,(i,j) \tag{5}$$

The objective of scheduling in the proposed algorithm is to map each task to each resource in way that makespan and total loosed tasks get minimum. For example, ETC matrix (table (1)) presents 9 tasks and 3 resources.

Table 1. ETC matrix

Task/Resource	**P1**	**P2**	**P3**
T1	2	3	1
T2	2	5	3
T3	1	3	4
T4	4	5	6
T5	8	5	6
T6	3	5	4
T7	4	2	4
T8	4	6	7
T9	2	3	1

4.1 The objective and fitness function in proposed algorithm

The main idea behind task scheduling is to minimize the makespan, which is the total execution time for all tasks. To solve task scheduling by ICA, a country is more suitable that minimize makespan and total consumed energy for all tasks. Equation (6) describes the first fitness function for each country. Also, equation (7) and (8) compute the energy consumption for each solution.

$$\text{Fitness}_1\,(C_i) = \alpha \times \frac{1}{\text{FTime}\,(C_i)} + (1-\alpha) \times \frac{1}{E_{(C_i)}} \tag{6}$$

$$E_{(C_i)} = \sum_{j=1}^{M} \sum_{i=1}^{N} E_{i,j} \tag{7}$$

$$E_{i,j} = (P_{j,i} - I_j)\,\text{ETC}\,[i\,,j] \tag{8}$$

Where E is the total consumed energy to execute all tasks in country i and $P_{j,i}$ is the consumed energy for executing task i on resource j. Also I_j is the consumed energy of resource i when the resource is ideal. It can be concluded from equation (6) that when the consumed enregy and makespan are minimzied, the fitness function is maxmized and shows the more suitable solution to the problem. Also the coeffiction α determines the impact of each parameter on fitness value. For example Fig. 3 illustrates a solution to the scheduling problem. Considering the

ETC matrix in table (1), the makespan equals to 12. Fig. 3 shows the total scheduling length for countries.

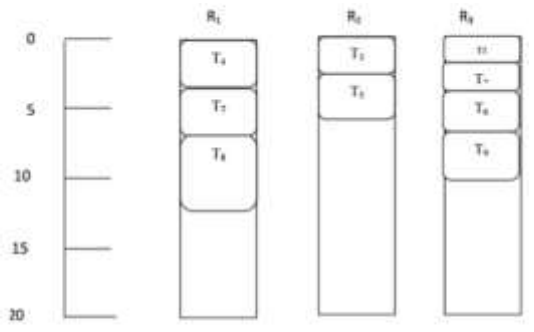


Fig. 3. Total scheduling length for countries

4.2 Second fitness function

As stated before, the main objective of task scheduling is to minimize the makespan. It is possible to have several countries with similar total scheduling time but with different workload on their resources. Therefore, second fitness function takes this factor into consideration. Such that, after achieving solutions with minimum makespan, second fitness function will be applied to them aiming at acquiring balanced solution in terms of workload. To gain maximum balanced workload, the fitness value should be computed for resources. To achieve the value of balanced workload we are have to compute sum of standard deviation. It is clear that we should compute the fitness for resources firstly. Equation (9) shows the second fitness function for balanced workload computing.

$$\mathrm{Fitness}_2(\mathrm{Countery}_i) = \frac{1}{\mathrm{RU}_i} \qquad (9)$$

At fires maximum exaction time is specified using equation (10).

$$\mathrm{E} = \mathrm{Max}\,\{\mathrm{T}_{ij} + \tau_{ij}\} \qquad (10)$$

To compute E's value in highest path, we use T_{ij} and τ_{ij} which are equal to data transmission time RMS and processing time, respectively. If we use u_i to represent fitness value of resource i for execting task j, we have:

$$\mathrm{u}_i = \frac{\mathrm{T}_{ij}+\tau_{ij}}{\mathrm{E}} \qquad (11)$$

To compute fitness value for each resource we use the equation process which is used for computing E value. In other words, the value of E is specified firstly, then using equation (10) the fitness value for each resource is computed. To compute standard deviation, we are have to compute total fitness value at first. Consider the two solutions for task scheduling in Fig. 2 and Fig. 3, the scheduling total time for this solutions is equal to 12.

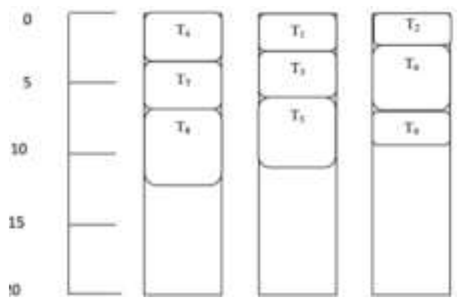
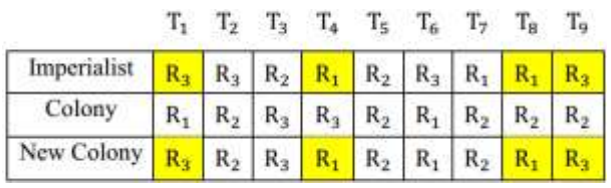


Fig. 4. Total scheduling time with respect to load balancing.

4.3 The assimilation and revolution operations in our proposed algorithm (ICA-GELS)

4.3.1 Assimilation operator

Historically, the assimilation policy was developed with the purpose of moving the culture and social structure of colonies toward the culture of the central government. In the proposed algorithm some cells approximately 40% (Duki, et al. 2010) of the Imperialist array are randomly selected (cells 1, 4, 8, and 9 in the Fig. 5 are imperialist and others are colony).

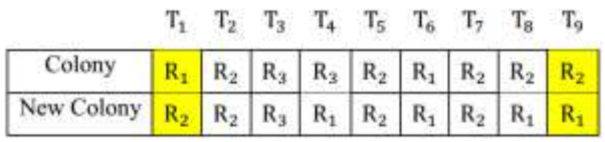


	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Imperialist	R_3	R_3	R_2	R_1	R_2	R_3	R_1	R_1	R_3
Colony	R_1	R_2	R_3	R_3	R_2	R_1	R_2	R_2	R_2
New Colony	R_3	R_2	R_3	R_1	R_2	R_1	R_2	R_1	R_3

Fig. 5. Assimilation operator representation

4.3.2 Revolution operator

To perform this operation, two cells are first selected in the colony in a random manner and their values are exchanged. This stages (random exchange or revolution operation) is repeated based on the percentage of the total number of tasks. If the new colony is better than the old one, it replaces the old colony; otherwise, this procedure is repeated. This operation is illustrated in Fig. 6.



	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Colony	R_1	R_2	R_3	R_3	R_2	R_1	R_2	R_2	R_2
New Colony	R_2	R_2	R_3	R_1	R_2	R_1	R_2	R_1	R_1

Fig. 6. Revolution operator representation

4.3.3 Local search algorithm parameters imitating the gravitational attraction

In the last execution step of ICA, due to the poor performance of the algorithm in local search, the solutions which are similar in scheduling length are given to the GELS algorithm to generate a neighbor solution (response) for them.

4.3.3.1 Solution dimensions (space)

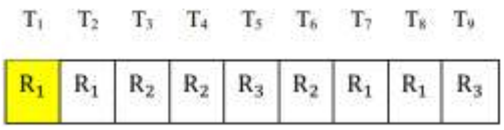
Unlike other algorithms, GELS algorithm does not perform fully randomly to obtain neighbor solution from the current solution. Each solution has many neighbors which are computed by a variable type in the current solution. All achieved neighbors are based on this variable type. The proposed approach assumes that solution dimensions and countries features are equal (solution dimensions are solution neighbors which are achieved by changing the current solution). Solution dimensions are equal to the size of input tasks.

4.3.3.2 Initial velocity vector

The number of initial velocity vectors elements are equal to number of solution dimensions. Initial velocity vector elements will be initialized to random numbers ranging from 1 to maximum initial velocity.

4.3.3.3 Neighborhood

Neighbor solution of the current solution is equivalent to solution which assigned task is changed. The value is between 1 and maximum initial velocity. A feature from country which has a maximum velocity is chosen, then its value changes randomly (from 1 to M). For example Fig. 7 presents a solution in which the property with maximum velocity is chosen and a neighbor solution is generated for it.



T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
R_1	R_1	R_2	R_2	R_3	R_2	R_1	R_1	R_3

Fig. 7. Representing a neighbor solution.

4.3.3.4 Calculating gravitational force in ICA-GELS algorithm

After acquiring neighbor solution for current solution, the fitness for neighbor solution is computed using equation (3). If the neighbor solution is better than the current solution, then it will be replaced its parent country in the new population; otherwise it is not copied in the new population. The amount of force between neighbor solution and current solution is computed using equation (12) and its value will be added to the initial velocity vector from which the neighbor solution is acquired to updated the initial velocity vector (it is possible to have negative gravitational forces). It is worth to noting that if the value of elements in initial velocity vector exceeds the specified ranges; their values is tuned to a number within the range.

$$F = G \times \frac{\mathrm{Fitness}_1\,(\mathrm{Candidate\_C}_i) - \mathrm{Fitness}_2\,(\mathrm{Current\_C}_i)}{R^2} \qquad (12)$$

Candidate_C and Current_C in above equation are neighbor solution and current solution, respectively. The value for G is fixed and it is equal to 6.672 and R is the neighbor radius for to objects in searching space which is fixed as so. The algorithm ends when it reaches the maximum specified threshold. The Pseudo-code of the proposed algorithm is shown below.

```
Step 1 : Initialization
Step 2 : 2.1. Generate the k number of random country with
         length n
          2.2. Speed_vector[1..n] = The initial velocity for
               each dimension
           2.3. Location_vector[1..n] = The value of the
               initial position of each particle
           2.4.Setting the Maximum: Maximum-Time (Mt)
               and Maximum-energy  (Me) according to the
               user's requirement.
Step 3 : 3.1. Create initial empires.
Steo 4 : 4.1. Assimilation & Revolution
        4.2. Assimilation: Colonies move towards
             imperialist.
        4.3. Revolution: Random change occur in the
             characteristics of some countries.
Step 5 : Position exchange between a colony and imperialist
Step 6 : Compute Makespan and energy for all country
Step 7 : 7.1. Imperialistic Competition
        7.2. Eliminate the powerless empires. Weak
             empires lose their power their power gradually
             and they will finally be eliminated.
Step 8 : The best country from the imperialist competitive
         algorithm as a solution to it current GELS will be
         producing a neighboring country
Step 9 :
      9.1. Direction = max (speed_vector [...])
      9.2. change the velocity_vector[index] of
         current_solution with random integer between 1
         and Max Velocity.
      9.3. Fitness of the neighboring country by
equation (3) is calculated and saving the worst fitness
       9.4. if direct country < neighbor country
           Neighbor country is selected as the best solution
Step 10:
      10.1. Calculated the mass of each particle with using
           formula (22)
      10.2. Calculate Acceleration for 'k' factors.
 Step 11 :
      11.1. Update Velocity_Vector for each dimension by
           gravitational force of country.
```

Figure 8. Pseudo-code of the proposed algorithm.

5. DISCUSSION

In this section, aiming at evaluating performance of the proposed work, the simulation results will be presented. Our simulations are conducted on OPNET Modeler simulator (Modeler 2009). We have conducted several simulations to verify the effectiveness of our proposed algorithms. Our experiments are conducted in a system equipped with 4 GB of RAM and 1600 MHz CPU which is run Windows Xp. The performance of the proposed algorithm is compared with several task scheduling algorithms considering the simulation parameters demonstrated in table (2). The iteration parameter shows that to achieving the application execution time using existing algorithms, 100 iteration are performed and the average of the values are computed. Also the initial value for parameters which are used in ICA and GELS and GA algorithms are shown in table (2), respectively.

Table (2). Initial values for parameters for algorithms

ICA	Assimilation rate	2
	Revolution rate	0.1
	α rate	0.8
ICA-GELS	Assimilation rate	2
	Revolution rate	0.1
	α rate	0.8
	Initial velocity	Between 1 and Max Velocity
	Constant Gravitation (G)	6.672
GA	P-Crossover	0.85
	P-Mutation	0.02

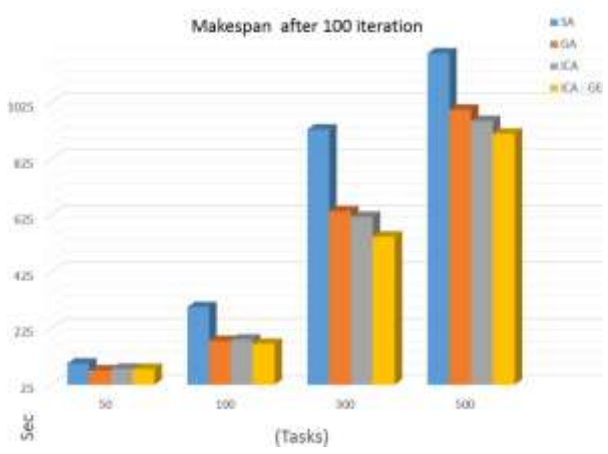


Figure 9. Comparison of makespan of algorithms

Fig.10 compares the algorithms in terms of consumed energy. As results show the consumed energy increases as tasks passes. The consumed energy for our algorithm is lower than other algorithms.

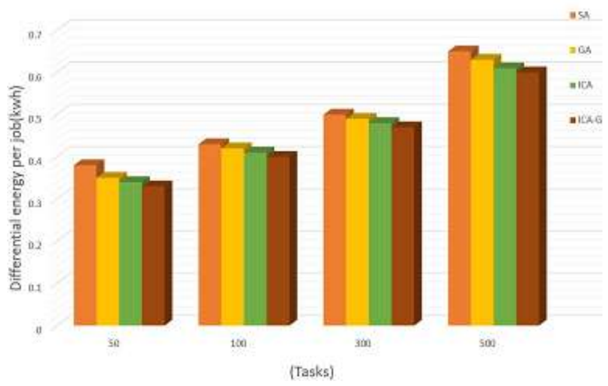


Fig. 10. Comparing consumed energy for task execution

5. CONCLUSIONS

In the proposed method, the researchers have combined imperialist competitive algorithm with gravitational emulation local search algorithm which has been used for scheduling in grid environment. The proposed algorithm, in which a weighted objective function is used considering the degree of importance of time and energy of user's projects, gives more freedom for specifying the time and energy of users' projects. In the use of this algorithm, similar to the objective function, the time and the energy, along with their weight, are considered based on the user's perspective. The purposes of the proposed scheduling are to minimize completion time and energy of implementation of tasks for different tasks of users simultaneously.

The proposed scheduling is compared with two algorithms of ICA and GA with regard to the parameters of time and energy. The results are investigated based on cost and energy requests in different charts. They show that if we have limited number of resources as well as high number of duties, the proposed scheduler has the best performance compared to the other three schedulers in reducing the overall time and energy of scheduling. The results of experiments show that the hybrid imperialist competitive algorithm and the gravitational attraction can reach a high performance regarding creation of a balance between energy and tasks implementation scheduling. Further research can be conducted with regard to the practice of resource allocation to works in the proposed algorithm through using an approach based on fuzzy logic and using of fuzzy inference system.

6. REFERENCES

[1] L.Y. Tseng, Y.H. Chin and S.C. Wang, The anatomy study of high performance task scheduling algorithm for grid computing system, Computer Standards & Interfaces, Elsevier 31(4) (2009), 713–722.

[2]J.Kolodziej and F.Xhada, Meeting security and user behavior requirements in grid scheduling, Simulation Modeling Proactice and Theory 19 (2011), 213-226.

[3] S.K. Garg, R. Buyya and H.J. Siegel, Time and cost tradeoff management for scheduling parallel applications on utility grids, Future Generation Computer Systems 26(8) (2010),1344–1355.

[4] B.T.B. Khoo and B. Veeravalli, Pro-active failure handling mechanisms for scheduling in grid computing environments, Journal of Parallel Distributed Computing 70 (2010),189-200.

[5] F.Xhafa and A.Abraham, Computational models and heuristic methods for Grid Scheduling problems, Future Generation Computer System 70(3)(2010),608-621.

[6] J. Kołodziej and F. Xhafa, Integration of task abortion and security requirements in GA-based meta-heuristics for inde pendent batch grid scheduling, Computers &Mathematics withApplications 63(2) (2012), 350–364.

[7]A.Kazem, A.M. Rahmani and H.H. Aghdam, A modified simulated annealing algorithm for static scheduling in grid computing, Proceedings of the 8th International Conference on

Computer Science and Information Technology (2008),623-627.

[8] Garcia-Galan, R.P. Prado and J.E.M. Exposito, Fuzzy scheduling with swarm intelligence-based knowledge acquisition for grid computing, Engineering Applications of Artificial Intelligence 25(2) (2012), 359–375.

[9] L. Wei, X. Zhang, Y. Li and Yu Li, An improved ant algorithm for grid task scheduling strategy, Physics Procedia, Elsevier 24 (2012), 1974–1981.

[10] Z. Pooranian, M. Shojafar, J.H. Abawajy and A. Abraham,An efficient meta-heuristic algorithm for grid computing,Journal of Combinatorial Optimization (JOCO) (2013),doi:10.1007/s10878-013-9644-6.

[11] Z. Pooranian, A. Harounabadi, M. Shojafar and J. Mirabedini, Hybrid PSO for independent task scheduling in grid computing to decrease makespan, International Conference on Future Information Technology IPCSIT 13,Singapore(2011), 435–439.

[12] S. Benedict and V. Vasudevan, Improving scheduling of scientific workflows using Tabu search for computational grids, Information Technology Journal 7(1) (2008), 91–97.

[13] R. Chen, D. Shiau and S. Lo, Combined discrete particle swarm optimization and simulated annealing for grid computing scheduling problem, Lecture Notes in Computer Science, Springer 57 (2009), 242–251.

[14] M. Cruz-Chavez, A. Rodriguez-Leon, E. Avila-Melgar, F. Juarez-Perez, M. Cruz-Rosales and R. Rivera-Lopez, Genetic-annealing algorithm in grid environment for scheduling problems, Security-Enriched Urban Computing and Smart Grid Communications in Computer and Information Science, Springer 78 (2010), 1–9.

[15] Z. Pooranian, A. Harounabadi, M. Shojafar and N. Hedayat, New hybrid algorithm for task scheduling in grid computing to decrease missed task, world academy of science, Engineering and Technology 79 (2011), 924–928.

[16] F. Xhafa, J. Gonzalez, K. Dahal and A. Abraham, A GA(TS) hybrid algorithm for scheduling in computational grids, Hybrid Artificial Intelligence Systems Lecture Notes in Computer Sci- ence, Springer 5572 (2009), 285–292.

[17] E. Atashpaz-Gargari and C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialist competitive, IEEE Congress on Evolutionary computation, Singapore (2007), 4661–4667.

[18] Braun, T. D., Siegel, H. J , A taxonomy for describing matching and scheduling heuristics for mixed machine heterogeneous computing systems, Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, pp. 330335, 1998.

[19] Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.,"BGSA: binary gravitational search algorithm". Nat. Comput. 9(3), 727–745, 2010.

[20] Voudouris, chris, Edward Tsang, Guided Local Search.Technical Report CSM-247, Department of ComputerScience, University of Essex, UK, August 1995.

[21] Barry Lynn Webster, "Solving Combinatorial Optimization Problems Using a New Algorithm Based on Gravitational Attraction", Ph.D. Thesis, Florida Institute of Technology Melbourne, FL, USA, May 2004.DOI: http://dx.doi.org/10.1109/T-C.1973.223690.

[22] J.Kolodziej and F.Xhafa, Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population, Future Generation Computer Systems,Elsevier 27(8)(2011),1035-1046

[23] A.Abraham,R.Buyya and B.Nath,Nature's heurustics for scheduling jobs on computational grids, Proceeding of the 8th IEEE International Conference on Advanced Computing and Communications, India(2008), 1-8.

[24] C.Weng and X.Lu, Heuristic scheduling for bag-of-tasks applications in combination with QOS in the computational grid, Future Generation Computer Systems 21(2)(2005),271-280.

[25] Chang, R. S., Chang, J. S., Lin, P. S., "Balanced Job Assignment Based on Ant Algorithm for Computing Grids", The 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291295,2007.