Provide a Method of Scheduling In Computational Grid Using Imperialist Competitive Algorithm

Mostafa Pahlevanzadeh [1] and Ali HarounAbadi [2],

[1]Department Of Computer Engineering, Electronic Branch, Islamic Azad University,Tehran,Iran
[2]Department Of Computer Engineering, Central Tehran Branch- Islamic Azad University, Tehran,Iran

Summary
The computational grids are a kind of distributed computations in which the resources of different computers that are distributed geographically, are shared to solve a particular problem. Scheduler is used to schedule user tasks appropriately according to the resources available in the Grid environment. To achieve this objective, efficient scheduling is an important part of a grid. In this paper, we present a new method using a combination of Imperialist Competitive Algorithm and Gravitational emulation local search algorithm. Findings of the experiments show that the proposed algorithm can reduce the duration of finishing tasks and the rate of missed tasks.

Keywords:
Computational Grid, scheduling of workflow, Imperialist Competitive Algorithm, Gravitational emulation local search algorithm

1. Introduction

The grid is actually a type of parallel and distributed system that enables the sharing, selection and integration of distributed resources across the field and various organizations on the basis of availability, capability, efficiency, cost and quality required by users [1]. Computational Grids have been developed as a new approach to solving large-scale problems in science, engineering and business. They be able to create Virtual Enterprises in order to share and integrate geographically millions of sources in organizations and managerial areas [2]. One of the important components in Grid systems is scheduler. This scheduler can be very simple, but most schedulers should be able to prioritize tasks and control system. In a Grid system, the user must be aware of the available and accessible resources in the system. Workload management system can easily do it [3]. Services like GIS and MDS tell the system what source or sources are provided [4]. The scheduler is for scheduling systems and heterogeneous distributed computing systems including NP-Complete and so far, several models and algorithms are provided to optimize the scheduling problem on heterogeneous systems.

2. Work Platform

Society-based algorithms such as Imperialist Competitive Algorithm searching the problem space as a whole and as a result, its integration is toward the global optimal and it has nothing to do with the local optimal. This disadvantage can be overcome using a combination of local search algorithms. Gravitational emulation local search algorithm is one of the local search algorithms that mimics the gravitational force and prevents the particles trapped in local optima.

2.1 Imperialist Competitive Algorithm

ICA algorithm is a method in the field of evolutionary computation that is able to find the optimal solution for optimization problems. With the mathematical model of the process of social-political development, this algorithm offers an algorithm for solving optimization mathematical problems. The algorithm's main pillars consist of assimilation, imperialistic competition and revolution.In fact, this algorithm considers the answer of optimization problem according to countries and tries to improve these solutions in an iterative process and ultimately lead to the optimal solution [5].

2.2 Gravitational emulation local search algorithm

This algorithm is based on the principle of gravitational force that causes objects in nature to be absorbed towards each other, so that an object which has more mass, would has a higher gravitational force. In this algorithm, each answer has different neighbors. Neighbors obtained in each group called the neighbors in its dimension [8]. For each dimension of answer, an initial speed would be defined. However, any dimension which has more initial speed, would be a more suitable answer to the required problem. In this algorithm, the gravitational force between two objects is calculated using (1).

$$F=\frac{G(CU-CA)}{R2} \quad (1)$$

Manuscript received June 5, 2016
Manuscript revised June 20, 2016

3. Related Work

Regarding extension and dynamics of grid space, deterministic algorithms cannot be efficient for solving the problem of scheduling. This has prompted researchers to experience the meta-heuristic algorithm for this problem, because the major share of experiences to solve this problem belong to the society-based algorithms such as genetics, population of particles and imperialist competition.

In [7], the imperialist competitive algorithm is designed solely for scheduling computational grid. The duration parameter is compared to methods such as GAA, GGA and GSA. The result of this comparison is superiority of Imperialist Competitive Algorithm in relation to the other methods of scheduling in computational grid.

In [9], a hybrid algorithm called RHGGSA that combines GA and Glass algorithms have been proposed to solve the scheduling problem and take into consideration the time and cost of completing the implementation simultaneously. The results of this algorithm has been compared with Min-Min, GA and GA-VNS algorithms, which indicates the efficiency of the proposed algorithm.

In[10], with a combination of genetic and gravitational emulation local search, a new algorithm is proposed to reduce the execution time and the number of new tasks that their deadline has passed. Compare the performance of the proposed method with similar methods showed that this method obtains better computational time. In this paper, in addition to Makespan, the measure of load balancing on resources is also investigated.

In [12], the combination of PSO and GELS algorithm is used to schedule tasks. This algorithm aims to minimize the execution time. Compare the results of this algorithm and PSO, GA and SA algorithms, indicates the efficiency of the proposed algorithm.

4. The proposed method

In this section, a method is proposed to schedule tasks of computational grid by optimizing the time and the number of the missed tasks. Suppose that there is a set of n tasks and a set of m source. Each of the n tasks must be processed in some way by each of the m source, so finally when performing tasks minimized and also minimize the number of missed tasks.

4.1 Fitness function

The first objective is to minimize Makespan or finish the longest execution time among all processors in the system. Suppose that Li and SPj to represent the size of the task i and j are the source of processing speed. So the execution time of task i on the source j can be achieved through the (2).

$$T(exe)=\frac{Li}{SPj} \quad (2)$$

On the other hand, the execution time of task i on the source j can be achieved through the (3).

$$Tcompelete(I,j)=Texe(i,j)+wait(i,j) \quad (3)$$

As a result, the (4) can obtain Makespan.

$$T\ complete = \frac{\sum Tk}{spj} \quad (4)$$

$$Makespan(\alpha)=max(T\ complete)$$

As noted above, the most important goal of scheduler is to be able to minimize the Makespan. Note that this time should be less than or equal to the maximum deadline (MD) in the midst of all tasks. In the proposed method for solving tasks scheduling with the help of imperialist competitive algorithm, a colony is appropriate that in addition to minimizing the Makespan, has a lower number of missed tasks. (5) shows how to calculate the fitness function for each colony.

$$fitness\ (country) = \frac{1}{makespan\ (country)} + \frac{1}{MissTask * Md} \quad (5)$$

4.2 Production of initial population

The initial population (Ncountry) randomly generated that each solution (country) is a vector to the dimensions of 1*n and n is the number of tasks. The value of each solution is numbers from 1 to the number of available resources. For example, Figure 1 shows an example of a solution with 9 tasks and 3 sources. The fitness of each solution is calculated based on the (5). The number of Nimp solution of the members of this population (countries with the best fitness function value) picked as the empire. Ncol or the rest of the countries constitute some colonies and each belong to an empire. For the partition of colonies, a number of colonies to be assigned to each empire commensurate with its power.

T1	T2	T3	T4	T5	T6	T7	T8	T9
R3	R3	R2	R1	R2	R3	R1	R1	R3

Fig. 1. Example of a solution with 9 tasks and 3 sources

In (6), Ci is the value of fitness in any empire and Min (Ci) is the lowest fitness function in an empire.

$$ci = Ci - Min\ (Ci) \quad (6)$$

Each imperial power is calculated according to (7), which means that by the fitness of each empire to the total amount of fitness in all empires.

$$Pi = \left| \frac{ci}{\sum_{j=1}^{Nimp} ci} \right| \quad (7)$$

Finally, the number of colonies for each empire, according to the (8) is equal to:

$$N.Ci = Round(Pi * Ncol) \quad (8)$$

4.3 Policy of assimilation

In line with the policy of assimilation, some features of empire will be applied to each colony. First, a number of empires' cells randomly selected based on a absorption rate and the properties of these cells applied in the colony. This operation is shown in Figure 2.

	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_3
New-Colony	R_3	R_3	R_3	R_1	R_2	R_1	R_2	R_1	R_3

Fig. 2. Movement of colonies toward the empire

4.4 Revolution operator

In Imperialist Competitive Algorithm, the revolution be modeled, when a colonized country move accidentally toward a new random situation. In this section, the number of cell of colonies were randomly selected based on the rate of revolution and their value changes [7]. If the new colony is better than colony previous, it will replace the previous colony. This operation is shown in figure 3.

	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_3	R_2	R_1	R_2	R_1	R_3

Fig. 4. The revolution operation in task Scheduling

4.5 Changing the locations of colony and empire

At the time of movement of colony to the imperialist country, some colonies may have better conditions than the empire. In this case, the empire is replaced by the imperialist country and algorithm continues with its imperialist and imperialist country in the new position. This time, it is considered as a new empire tries to impose a uniform policy on the colonies.

4.6 Calculate the total power in an empire

The power of an empire is defined as the power of the imperialist state, plus a percentage of the total power of its colonies. ζ is a number between 0 and 1 as is sent input to the algorithm. The power of an empire is calculated according to the (9).

T.ci=Fit(imperialist)+ζ Mean(Fit(colonies of Empire)) (9)

4.7 The competition between empires

During the competition between empires, the weak empires that during the execution of the algorithm gradually lose their colonies, would drop and their colonies fell into the hands of powerful empires. In each iteration of the algorithm implementation, one or more colonies will get from the weakest empires and would give to the empires with the most power.

To model the competition between empires, the power of each empire was obtained using (10) and then normalized using (11):

N.T.Ci= T.ci- Min (T.ci) (10)

$$Ppi=\left|\frac{N.T.Ci}{\sum_{j=1}^{Nimp} N.T.Ci}\right| \quad (11)$$

To do this, the roulette wheel method can be used, but due to the exorbitant cost, we use another method described in [4]; thus, first of all, vector P should be made that contains normalized values of empires' power (12).

Then we use a vector of length 1 * Nimp that its amount randomly selected from the uniform distribution in the range of 0 and 1 (13). Finally, we subtract this random vector of the vector P. The colonies gave to an empire related to an index in vector D that contains the greatest value.

P = [Pp1, Pp2, . . . , PpNimp] (12)

R = [r1, r2, . . . , rNimp] (13)

D = P − R = [D1,D2, . . . , DNimp] =[Pp1 − r1, Pp2 − r2, . . . , PpNimp − rNimp]

In the proposed algorithm, an empire be considered deleted when it has lost its colonies.

4.8 Gravitational algorithm

The solution is considered the gravity of each cell dimension and neighbors of a solution obtain at any dimension by change the solution in that dimension [9]. For example, in Figure (5) and the first solution, the cell which has the most speed would be selected and a neighbor's solution can be obtained.

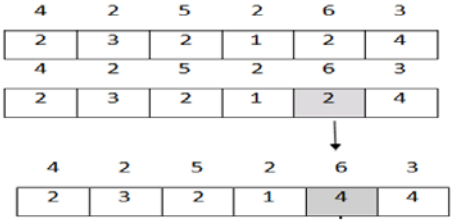

Fig. 5. Showing a neighbor solution

After obtaining the solution adjacent to the current solution, the solution neighboring fit using (10) is calculated, the initial velocity vector to be updated. The amount of gravitational force is added to the initial velocity vector related to the dimension by which the

neighbor solution is obtained in order to update velocity vector [10].

$$\text{Force}=6.672*\left[\frac{\text{Fit(Condidate)}}{\text{R2}}-\frac{\text{Fit(Current)}}{\text{R2}}\right] \quad (14)$$

In (10), Candidate and Current are neighboring current solution, respectively. G is a fixed amount of 6.672 and R is neighborhood radius parameter between two objects in the search space. The algorithm ends when it reaches its maximum number of iterations.

4.9 Work stages

After the competition in empires, because Imperialist Competitive Algorithm does a poor job, some of the best colonies selected and gave to the GELS algorithm with the empire himself in order to produce a better neighbor's solution. Finally, the Candidate colonies derived from the output of the local search algorithm and replace the weakest colonies of the empire.

Step 1. Based on the fitness function, some of the best colony selected as the primary empires and then the remaining colonies assigned in relation to the power of the empires

Step 2. Move colonies toward the empire (matching policy or absorption).

Step 3. Impose the revolution operator on the colonies.

Step 4. If there is a colony in an empire which has more fitness function; the position of colony and empire should be substituted.

Step 5. Calculate the total power of an empire (taking into account the fitness value of an empire and its colonies).

Step 6. Select the weakest colonies of the weakest empires and assign them to the most powerful empires

Step 7. Remove weak empires.

Step 8. GELS algorithm should be imposed on both empire and percentage of best colonies.

8.1.: The current solution is considered as the best solution.

8.2.: In the current solution, the dimension which has the most initial speed would be selected and its value randomly changes (a number between 1 to M). Fitting of the neighboring countries obtained at this stage and calculated according to the (14).

8.3.: If an empire or colony derived from previous empire or a colony is better in terms of fitness function, it is considered as the best solution.

8.4: The gravitational force between the neighbor empire and the current empire is calculated using (10).

8.5.: The gravitational force obtained in the previous step added to the dimension in which the neighbor solution obtained. Therefore, the initial velocity vector be updated.

8.6.: If all elements of the vector of initial velocity are zero, or the number of iteration reach its maximum, the algorithm ends, otherwise go to step b.

8.7.: The colonies derived from the GELS algorithm should be replaced by the weakest colonies of any empire in terms of fitness function.

Step 9. If the algorithm has reached the maximum number of iterations, stop now or otherwise go to 2.

Step 10. Choose the best empires among the empires as a response.

5. Evaluation

In this section, the results of the proposed algorithm for scheduling independent tasks on a computational grid network in comparison with other methods is provided. All experiments were carried out using Matlab software on a system with 2.6 GHz CPU, 8 GB memory and Windows 10.

Different parameters are shown in Table 1.

TABLE I. values of input parameters for the ICA-GELS algorithms

Value	Parameter	
%10	Revolution operator rate	ICA
%30	Absorption Rate	
0.1	ζ	
1	Neighborhood radius (R)	GELS
6.672	Gravitational constant	
Between 1 and maximum speed	Range of velocity vector values	
Number of tasks	Maximum speed	

TABLE 2 shows the results of ICIA-GELS algorithm and other algorithms for MakeSpan. As can be seen, ICA-GELS algorithm obtained better results than other algorithms.

TABLE II: Comparison ICA-GELS algorithm results with other Method

Iter.	(pop,task, source)	SA	GA	GSA	GGA	ICA	ICA-GELS
100	(50,50,10)	124.32	81.81	80.99	82.42	81.79	79.60
100	(100,50,20)	97.63	50.94	48.5	49.73	46.27	44.43
100	(200,50,30)	69.97	35.05	34.33	36.45	35.32	31.20
100	(50,100,10)	293.22	162.16	158.67	164.88	157.83	155.42
100	(100,100,20)	155.53	90.71	87.24	90.75	86.22	81.73
100	(200,100,30)	115.97	66.72	V.L.	68.22	64.71	57.24
100	(50.200,10)	583.55	329.36	316.66	317.78	312.90	305.52
300	(50,50,10)	111.91	81.18	81.37	81.27	81.48	79.04
300	(100,50,20)	73.75	45.76	45.09	45.75	45.30	44.16
300	(200,50,30)	69.78	34.44	V.L.	34.16	34.11	30.91
300	(50.200,10)	566.16	313.21	310.19	313.5	309.16	304.38

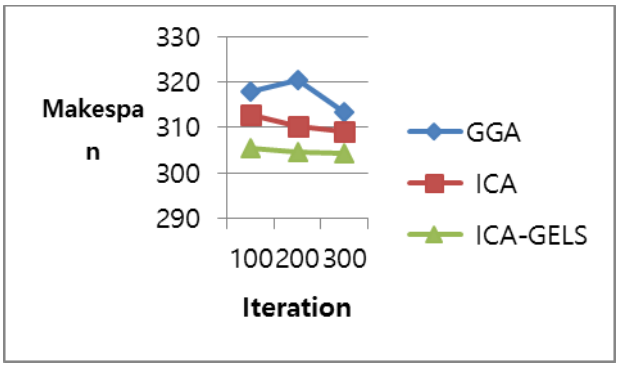

Fig. 6.: Comparison of Makespan for 50 task for different iteration

Figure 6 shows the Makespan for ICA GGA, GSA,, GA ICA-GELS algorithms where 200 tasks, 10 sources in repetitions of 100 to 300 on the resources scheduled using specific algorithms. Among the scheduling algorithms, it can be seen that the proposed algorithm has less Makespan compared to other algorithms [7].

TABLE III: shows the results of ICA and ICA-GELS algorithms for Miss Rate.

Iteration	pop	task	Res	ICA	ICA-GELS
100	50	50	10	0.08	0.06
200	50	50	10	0.08	0.06
300	50	50	10	0.08	0.06

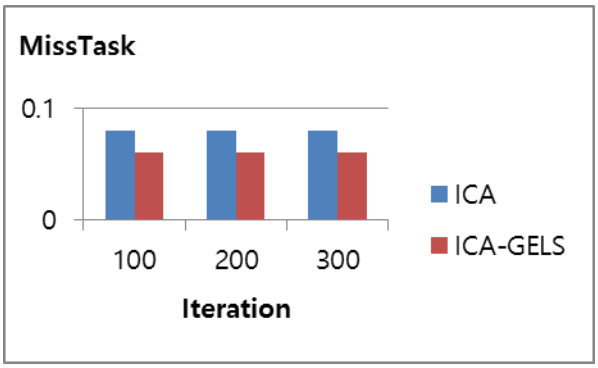

Fig.7.: Comparison of Miss Rate for 50 tasks for different iterations with 10 sources

6. Conclusions and future work

The imperialist competitive algorithm be used in the Grid scheduling problems and its superiority over other meta-heuristic algorithms such as genetic algorithms and aggregation of particles is proven[7].

In the proposed algorithm (ICA-GELS), the global search ability of Imperialist Competitive Algorithm combined with Gravitational Emulation Local Search, so that the proposed algorithm can be more efficient than other algorithms. The proposed algorithm was compared with ICA algorithm and the simulation results show that the proposed hybrid algorithm compared to ICA algorithm, produced less Makespan and also reduces the number of the missed tasks. In future research, other parameters related to quality of service such as cost, efficiency and fault tolerance should be examined.

References

[1] S.Dipti and M.pradeep, "Job Scheduling Algorithms For Computational Grid In Grid Computational Environment.", International Journal Of Advance Research In Computer Science And Software Engineer.Vol. 3 Issue. 5,2013,pp.735-744

[2] Y.Jia, B.Rajkumar and R.Kotagiri, " Workflow Scheduling Algorithms for Grid Computing.", Grid Computing and Distributed Systems (GRIDS) Laboratory. ISBN: 978-3-540-69260-7.2008,pp. 110-150

[3] B.Elwyn, "Survey On Heuristics Based Resource Scheduling In Grid Computing.". IJCSE Vol. 5 No.1,2014,pp.9-14

[4] A.Yousif, and M.Sulaiman, ,"Job Scheduling Algorithms on Grid Computing.", International Journal of Grid Distribution Computing Vol. 8, No.6, 2015,pp.125-140.

[5] E.Atashpaz, and C.Lucas, " An algorithm for optimization inspired by imperialist competive algorithm.", IEEE congress on evolution.2007, pp. 4661-4666.

[6] S.Attar, M.Mohammadi and R.Tavakoli, "A Novel Imperialist Competitive Algorithm to SolveFlexible Flow Shop Scheduling Problem in Order to Minimize Maximum Completion Time.", International Journal of Computer Applications. Vol.28, No.10,2011, pp. 37-32

[7] Z.Pooranian, M.Shojaeefar and N.Behrouzian,"Using imperialist competition algorithm for independent task scheduling in grid computing." Journal of Intelligent & Fuzzy Systems.2013, DOI:10.3233/IFS-130988.

[8] A.Jula and N.Naseri, " A Hybrid Genetic Algorithm-Gravitational Attraction Search algorithm to Solve Grid Task Scheduling Problem." ICSCA, 2012, pp. 158-162.

[9] V. GhaedRrahmati and S.Alavi, "A Reliable and Hybrid Scheduling Algorithm based on Cost andTime Balancing for Computational Grid." ACSIJ, Vol. 3, Issue 3, No.9,2014,pp22-31.

[10] Z.Pooranian, " A Hybrid Metaheuristic Algorithm for Job Scheduling on Computational Grid". Informatica,Vol. 37,NO. 2, 2014, pp. 157-164.

[11] R.vijaylakshmi, V.vasudevan, "Static Batch Mode Heuristic Algorithm for mapping independent task in computational grid.", International journal of Computer Science, Vol. 11,2015,pp224-229

[12] Z.Pooranian, J.Abawajy," An efficient meta-heuristic algorithm for grid computing." , Springer, Journal of Combinatorial Optimization (JOCO). Vol. 30, Issue. 3,2015,pp 413-435.

Mostafa Pahlevanzadeh graduated in Computer Applications from the South branch of azad university in Tehran,2003. He was born in 1972 in Ardebil. He was a teacher in Web designing and computer science from 2003 to 2015. He has been working as a researcher in the fields of algorithms and resource allocation. several internal articles has published by this person.

Ali HarounAbadi graduated from computer Applications in Phd degree and working as a university teacher in central branch of azad university in Tehran .He is interested in following research topics:Grid,Software engineering,Computer Systems Modeling and evaluation,etc.