ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 3, No.9 , May 2014
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

A Reliable and Hybrid Scheduling Algorithm based on Cost and Time Balancing for Computational Grid

Vahid GhaedRahmati[1], Seyed.Enayatallah Alavi[2] and Iman Attarzadeh[3]

[1]Department of Computer Engineering, Khouzestan Science and Research Branch, Islamic Azad University, Ahvaz, Iran
Rahmati@iauahvaz.ac.ir

[2]Department of Computer Engineering, Shahid Chamran university of Ahvaz, Iran
Se-alavi@yahoo.co.uk

[3]Department of Computer Engineering, Dezful Branch, Islamic Azad University Dezful, Iran
Attarzadeh@iaud.ac.ir

Abstract

Grid computing system is different from conventional distributed computing systems by its focus on large-scale resource sharing and open architecture for services. tasks scheduling is a crucial problem in Grid environments. Many of grid scheduling systems optimize completion time and cost separately. In this paper, for solving the scheduling problem of computational grid system used a combination of genetic algorithm and Gravitational Emulation Local Search (GELS) algorithm and a hybrid scheduling algorithm (RHGGSA) which considers both the completion time and execution cost is introduced. The algorithm applies a weighted objective function that takes into account both the completion time and execution cost of the tasks. To show the out performance of the proposed task scheduling algorithm, the obtained results are compared with those of Min-Min, GA and GA-VNS. Simulation results and comparisons based on a set of problem demonstrated the efficiency and effectiveness of our proposed approach.

Keywords: Task Scheduling, Grid Computing, Genetic Algorithm, Gravitational Emulation Local Search, Cost.

1. Introduction

Grid computing is defined as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [1]. Therefore, a grid computing communicates with a wide range of Heterogeneous resources include personal computers, workstations, super computers and clusters that each of these resources has different computational and configuration facilities and organized by various management policies [2]. Thus, task scheduling is a huge challenge in such a system. Providing an appropriate scheduling algorithm is important and at the same time is very difficult. Hence, these systems are faced with a NP-hard problem to scheduling tasks to obtain high-performance, accurate response time and maximal revenue.

Heuristics optimization algorithm is widely used to solve a variety of NP-complete problems. In [3], the author proposed a new method of scheduling in grid based on Heuristic Algorithms. Moreover, Abraham et al [4] presented three basic heuristics implied by nature for grid scheduling, namely Genetic Algorithm (GA) [5], Simulated Annealing (SA) [6] and Tabu Search (TS) [7], and heuristic derived by a combination of their three algorithms. Furthermore, GA works well on most global optimal problems. But, since the ability of local search in GA is weak and also the possibility of becoming trapped in the local optimum is high, in this paper, its combination GELS[8] which is a local search algorithm is used to improve its performance in finding solution. In this paper, a cost and time balancing algorithm which is based on Genetic and GELS algorithm is proposed. a hybrid Algorithm named RHGGSA, is presented for static scheduling of independent tasks within grid environments. The Hybrid scheduling algorithm considers two QoS criteria: makespan and execution cost of user task. The main objective of the proposed algorithm is to reduce the overall cost of task executions without any significant increment in system makespan. The rest of paper is organized as follows. We begin with an overview of related works in section 2. our approach are presented in section 3. Experimental results and discussion are represented in sections 4. Finally the paper is concluded in section 5.

2. Related Work

In the past few years, researchers have proposed scheduling algorithms for grid environments. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, this issue attracts the interests of the large number of researchers. For example, In [9] have offered a model that combines two optimal schemes, genetic algorithm (GA) and simulated annealing (SA), based on the probability distribution. Actually, the algorithm uses the benefits of the genetic algorithms and simulated annealing and provides a parallel genetic-simulated annealing algorithm to solve the task scheduling problem in grid environments.

In [10] combination of local search algorithm were used for scheduling by using of SA and global search by genetic (GSA). In this approach, if in each generation, changed chromosomes by genetic operator don't improve comparing to previous generation they are affected by SA and are likely accepted for the next generation and this work lead to increase search efficiency, further convergence rate and ran of local minimum.

In [11] the Genetic Algorithm is presented in which both QOS parameters including time and cost were regarded simultaneously and because these two parameters are in conflict each other and they can't improve together simultaneously and one improvement leads to efficiency decrease in the other, it gives weight to each parameter as the weighing is done by user as each of the parameters has more value for the user gives more weight and the other gives less weight, or weighting is happened randomly.

In [12] introduced a GA Algorithm which used standard deviation less than (0.1) as stopping criterion to limit the number of iterations of GA. This algorithm has drawbacks such as low quality solutions (almost same as low quality solutions of standard GA), generating initialization population randomly (even though the time consumed by algorithm is small comparing with standard GA), and mutation depends on exchange of every gene in the chromosome. This mutation will destroy the good information in subsequent chromosomes in next generations.

In [13] a group of researchers have developed a novel task scheduling based on hybrid genetic and GELS algorithm. This technique have solved grid scheduling problem and minimize missed tasks. In this approach every chromosome represents visible solution ,and move (pick) solution after GA operation that better than current solution using purpose function (fitness function) and some of advantage of GLES algorithm in random search. This algorithm proved that can decreases the number of missed tasks more than other algorithms.

In [14] considered the minimization of the makespan using GA based on Rank Roullete Wheel Selection (RRWSGA). They use standard deviation of fitness function as a termination condition of the algorithm. The aim of using standard deviation is to shorten the the time consumed by the algorithm with taking into account reasonable performance of Computing resources (97%).

In [15] proposed an algorithm that minimizes makespan, flowtime and time to release as well as it maximizes reliability of grid resources. It takes transmission time and waiting time in resource queue into account. It uses stochastic universal sampling selection and single exchange mutation to outperform other GAs.

In [16] propose HCSGA (Hybrid Clonal Selection GA) and validate performance using GridSim toolkit with GA, Max-min and Min-min. HCSGA first generates a new group of individuals through clone and than crossover/selection independently all the generated individuals, respectively. The results show that HCSGA improves convergence and is superior to other algorithm simultaneously.

Many presented algorithms and methods in scheduling problem within grid environments just consider one of the users' requirements. These approaches mostly consider system-centric factors like throughput and makespan of the system as main objective in scheduling, ignoring the interests and requirements of users. Some users with budget constraints may prefer to execute their own tasks with lower quality of service such as longer execution time for the sake of paying a lower price. In this case, the users' requirements become an important factor in designing schedulers. The algorithm proposed in this paper is mainly different from these studies, Since cost and time are two important factors for users in grid environments, lots of research efforts in scheduling have been focused on presenting methods and algorithms to optimize these two factors [17, 18]. Therefore, the cost is the most important factor considered in this paper. In this paper, a cost and time balancing algorithm which is based on GA and GELS algorithm is proposed.

2.1 Genetic Algorithm

A Genetic Algorithm (GA) is a class of evolutionary, adaptive, stochastic algorithms involving search and optimization. Genetic algorithms were first used by J. H. Holland [5]. The basic idea is to mimic the process of natural evolution, in order to create artificial processes for a "clever" algorithm with the ability to find the solution of complex problems, such as job scheduling in computational Grid. A GA maintains a population of possible solutions to a problem, encoded as chromosomes based on a particular representation scheme [19].

In each iteration the fitness of every individual in population is evaluated. Afterwards, selection and reproduction operators are applied, in order to form a new population, which is again used in the next iteration of the GA. Reproduction is constituted of crossover and mutation

operators. The whole process is repeated a number of times, called generations. The pseudo-code of a typical GA is shown in Figure 1.

```
Create Initial Population;
Evaluation();
While (stopping criteria not met)
{
        Selection ();
        Crossover ();
          Mutation ();
          Evaluation ();
}
Return best solution;
```

Fig.1 Genetic Algorithm pseudo-code

2.2 Gravitational Emulation Local Search (GELS)

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 neighbours which can categorize based on a criteria which is depended on the problem. Obtained neighbours in each group are called neighbours 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 neighbour space of current response and the gravitation force between these two response was calculated. In the second method, the gravitation force among all of the neighbour responses in a neighbour 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 neighbour spaces of current response, The second method is allowed movement to the responses out of local neighbour spaces of current response in addition to allowed neighbour 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.

2.2.1 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.

2.2.2 Gravitational force (GF)

GELS algorithm uses the formula (1) for the gravitational force between the two solutions as

$$F = G\ (\frac{CU-CA}{R^2}) \qquad (1)$$

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. Proposed Algorithm

Genetic Algorithm is weak in local search and it is strong in global search and versus, GELS is a local search algorithm by imitation of gravitational attraction, so it is strong in local search and it is weak in global search. so Considering abilities of genetic algorithms (GA) and Gravitational Emulation Local Search (GELS), a new scheduling algorithm named RHGGSA is presented in this section. The proposed algorithm is a hybrid GA and GELS which aims to reduce the overall cost of the users, while the makespan of the system is taken into account. RHGGSA runs the genetic as the main algorithm and uses the GELS procedure for improving individuals in the population. Each individual in the population is used to generate new offsprings by applying the appropriate genetic operators such as selection, crossover and mutation. In the following subsections, more details about the proposed algorithm are provided to describe the algorithm step by step.

3.1 Encoding Mechanism

In the coding scheme we have developed for our problem, each solution is encoded as a vector of integers. For a

problem with n tasks and m resources, the length of the vector which can be considered as a chromosome is n. As well as, the content of each cell of vector which shows a gene value in chromosome can take a number between 1 and m that representing the resource allocated to that task. An example of a chromosome as a schedule with 6 tasks and 4 resources is shown in Figure 2.

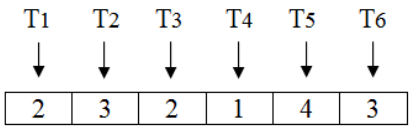

Fig.2 An example of a chromosome in coding Scheme

For generating an initial population with p individuals, a random number between 1 and m is assigned to each cell of the vector the size n for p times.

3.2 Objectives and Fitness Functions

Our main objective here is to get task assignments that will achieve minimum completion time and minimum Price for users. Therefore, our RHGGSA algorithm is a two dimensions optimization. In this problem two objectives Price and Makespan are in conflict with each other naturally so that when Price is reduced then Makespan is increased.

3.2.1 Makespan

The first objective function of our algorithm is the Makespan or latest completion time of the task schedule. Makespan means the longest completion time among all the resources in the system. Consider *Li* and *SPj* denote the size of the task i and processing speed of the resource j, respectively. Then, the execution time of the task i on the resource j can be formulated as follow:

$$Texe(i,j) = \frac{Li}{SPj} \qquad (2)$$

For each resource there will be a completion time for tasks which assigned to it. For example, figure 5 shows completion time in each resource according to figure 2. Suppose there are 6 Tasks with the sizes in figure 3 that are assigned to 4 resources.

Task	T1	T2	T3	T4	T5	T6
Lenght	24	32	28	20	34	18

Fig.3 Example of Tasks and their sizes

And there are 4 resources with following speeds:

Number of Resource	R1	R2	R3	R4
Speed of Resource	3	2	2.5	1.5

Fig.4 Example of Resource and their Processing Speed

Then execution time of each task on allocated resource using (2) based on Figure 2 is:

Texe(1,2) = 24 / 2 = 12
Texe(2,3) = 32 / 2.5 = 12.8
Texe(3,2) = 28 / 2 = 14
Texe(4,1) = 20 / 3 = 6.6
Texe(5,4) = 38 / 1.5 = 25.3
Texe(6,3) = 18 / 2.5 = 7.2

On the other hand, the completion time of the task i on resource j can be defined as formula (3).

$$Tcomplete(i,j) = Texe(i,j) + Wait(i,j) \qquad (3)$$

where Wait(i, j) denotes the start time of the task i on the resource j. Consequently, the system makespan can be computed using formula (5).

$$Tcomplete(j) = \frac{(\sum_{(K \in Aj)} Tk)}{SPj} \quad 1 \leq j \leq m \qquad (4)$$

Where, Aj is the set of tasks indexes which are assigned to resource j.

$$Makespan(\alpha) = Max\ \{Tcomplete(j)\} \quad 1 \leq j \leq m \qquad (5)$$

Therefore, Makespan in Figure 5 is 26. One of goals is to minimize (5), which means that the assigned tasks to resources will be completed in the shortest time.

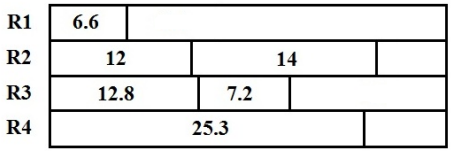

Fig.5 Example of Makespan

For example three tasks T1 and T3 are assigned to resource R2. Therefore, completion time of tasks on R2 will be:

$$Tcomplete(2) = 12+14 = 26$$

3.2.2 Minimum Cost

second objective function is total cost that must be minimized. Suppose Pj denotes to unit price for resource j. Therefore, the execution cost of the task i on the resource j can be computed using formula (6).

$$Cost(i,j) = Texe(i,j) \times Pj \qquad (6)$$

Then, total cost for a chromosome is calculated as follow:

$$Cost(\alpha) = \Sigma\ Cost\ (j) \quad 1 \leq j \leq m \qquad (7)$$

Where Cost(α) denotes the overall cost resulting from a chromosome in population that representing a scheduling.

3.2.3 First Fitness Functions (Cost and Time Balancing)

When introducing tasks, some users are concerned about completion time and others about costs for implementing their jobs. In the proposed method, for this reason, users can measure weights for completion time and task implementation cost (Wt and Wc) while introducing their jobs. These are in the range [0,1] and the sum of the weights is equal to 1. If Wc, for example, is 0.6, then users are concerned about financial costs of task implementation by %60, and about completion times by %40. Users specify weight coefficients during operation. Wt and Wc enable the users to get much more freedom to put their jobs on the Grid. Therefore, given the above definitions, a fitness function of a chromosome can be calculated by Equation (8).

$$Fit1 = Wc \times \frac{Cost(\alpha)}{Budget} + Wt \times \frac{Makespan(\alpha)}{MD} \qquad (8)$$

Where Cost(α) and Budget represent total cost of solutions and costs proposed by the user in the Grid scheduling system to process all tasks, respectively. Makespan(α) is the longest completion time of tasks among all system resources in a chromosome, and MD shows the maximum deadline for completing tasks. Ultimately, whatever a chromosome has low merit, that chromosome has much better merit.

3.2.4 Second Fitness Functions (Reliability)

As some chromosomes may be found where total schedule lengths stay the same, so for the second fitness function, one of the most reliable criteria for achieving the solution will be investigated. In this step to calculate reliability for each chromosome, records of resource fault occurrence are maintained in a fault occurrence history table (FOHT) in the grid information server [23]. At FOHT there is a row with two cells for each resource. One of these cells presents the history of fault occurred in that resource and the other keeps the number of task execution by resource. A part of FOHT in a given time is presented in Figure 6. For example, this figure indicates the number of task execution by R1 is 6700 but at two times of these executions, fault occurred.

	1	2
R1	**10**	**6700**
R2	**5**	**100**
R3	**0**	**123**
R4	**8**	**120**
R5	**3**	**25**
.	.	.
.	.	.
.	.	.

Fig.6 A part of FOHT in a given time

The application of this table allows a chromosome with a low number of fault occurrences in its resources to have the best chance of being selected. In order to evaluate the fitness function of chromosomes, information from the FOHT is used to improve the reliability and chromosomes that are much lower in the total fault occurrences show the most reliability for scheduling [23]. So we can define the second fitness function as below:

$$Fit2 = \sum_{i=1}^{n} \left(\frac{FOHT[i,1]}{FOHT[i,2]} \times 100 \right) \qquad (9)$$

In the proposed algorithm, a tournament operator is used to select chromosomes, while a two-point crossover operator is applied for intersect operation. In the mutation stage, when a chromosome was selected through the previous stage, a gene is randomly selected and the value of its resource field is changed by a random number from 1 to m. The main purpose of this mutation is to replace the task resource parameter so that it can be processed in better resources. Figure 7 shows the application of the mutation operator on a chromosome.

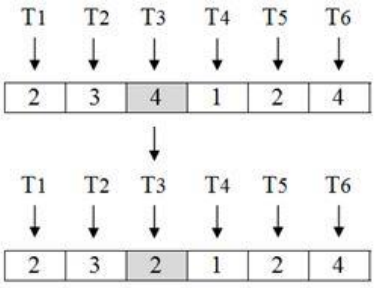

Fig.7 Example of application the mutation operator on a chromosome

In the final stage of running the genetic algorithm, because it shows a poor performance for local searching, solutions with similar total project duration are put on the GELS algorithm so a neighboring solution could be generated for them. Unlike other algorithms for the GELS algorithm, to generate a neighbor solution from the current solution is not completely random. Every current solution has a wide range of neighbors deriving based on one particular change in the current solution. It is called the change of direction to the neighbor solution. All neighbors obtained in this direction are only and only resulted from changes of this type. In proposed method, Each gene of chromosome is considered as a dimension of the problem. In fact the problem's dimensions are just the neighbouring solutions which are obtained by changing the current solution. Initial velocity is given to each of the problem's dimensions that is done randomly and ranges from 1 to the maximum of the initial velocity[22].

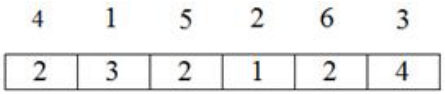

Fig.8 Example of velocity vector in proposed algorithm

A neighbor solution for the current solution is a solution where resource assigned to a particular task changes in the relevant dimension. To do this, a given initial speed is randomly assigned to each dimension of the solution. The value ranges from 1 to the maximum initial speed. A gene which have more velocity was selected and its values was changed randomly with the number 1 to M. Figure 9, for example, shows that for the first solution, the gene with the highest speed is selected for which a neighbor solution is generated.

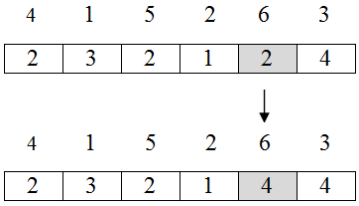

Fig.9 Example of neighbor solution in proposed algorithm

3.3 Calculated Gravitation Force

After that, fitness amount is calculated using Equation 9. If neighborhood solution is improved comparing to current solution, it is substituted to its parent chromosomes in new population. Nevertheless, it is not copied in new population. Then, the amount of the force between neighborhood solution and current solution is calculated using Equation 9 and its amount is added to primary velocity vector related to the dimension which the neighborhood response is obtained from it until its primary velocity vector is updated [22].

$$Force = G \times \frac{Fit(Current\_Solution) - Fit(Neighborhood\_Solution)}{R^2} \quad (10)$$

That in which Current_Solution and neighborhood_Solution are the value of the current solution and the neighborhood solution. G is equal to a constant gravitational force 6.672 too; R is the radius of the distance between two objects. Algorithm is finished when the primary velocity equal to zero for all dimensions or the number of algorithm iteration received to its maximum number. Base on the given explanation the pseudo-code of the search algorithm RHGGSA is shown in figure 10.

Step 1: Initializing parameters

Step 2:
2.1. Generate the P number of random Chromosome with lenght n.
2.2. Dimensioning (n)
2.3 Velocity_Vector[1..n] = Initial velocity for each Dimension()
2.4. Set the parameter (Budget, MaxDeadline, neighbor radius)
2.5. Setting the weights: Weight-Time (Wt) And Weight-Cost (Wc) according to the user's requirement.

Step 3:
Compute Cost(a) and Makespam(a) all chromosomes.

Step 4: Repeat
4.1. Evaluates all individuals in the population using formula 8 and 9

4.2. Select the P/2 members of the combined population based on minimum fitness, to make the population the next generation.
4.3. Crossover
4.4. Mutation

Step 5: Selected chromosomes set as Current_Solutions and make Neighbor_solutions with direction.

Step 6:
6.1. Direction = max(Velocity_Vector[…])
6.2. change the velocity_vector[index] of current_solution with random integer between 1 and Max Velocity.

Step 7: Calculate gravitational force between Current_solution and Neighbor_solutions using formula

$$Force = G \times \frac{Fit(Current\_Solution) - Fit(Neighborhood\_Solution)}{R^2}$$

Step 8: Update Velocity_Vector for each dimension by gravitational force of chromosome.

until a terminating criterion is satisfied.

Fig.10 pseudo-code algorithm RHGGSA

4. Experimental results

In this section, To show the out performance of the proposed task scheduling algorithm, the obtained results are compared with those of Min-Min, GA and GA-VNS. The proposed RHGGSA algorithm for grid task scheduling was implemented in C# programming language on a Intel Pentium(R) 4 CPU 3.00GHz, 3GB machine running under windows XP environment. The performance of the proposed algorithm (RHGGSA) is evaluated through conducting several simulation experiments under different scenarios. Figure 11 summarizes the simulation parameters. To evaluate the proposed algorithm, a set of simulations is considered based on changes in the number of user tasks, budgets payable by customers, and changes in the range of task duration over 10 resources. In the first experiment, the proposed algorithm and three alternative scheduling algorithms are investigated with changes in task numbers and a fixed budget.

	Parameter	Value
GA	Number of generations	100
	Npop	200
	Mutation Rate	0.02
	Crossover Rate 0.8	0.8
	Selection	tournament selection
GELS	Neighborhood Radius (R)	1
	Constant Gravitation (G)	6.672
	Initial Velocity	between 1 and Max Velocity
	Max Velocity	Sized of the input tasks

Fig.11 used parameters to simulate algorithm RHGGSA

4.1 Scenario 1 (Effect of change in number of tasks on makespan)

To improve the precision of the reported results, each experiment is independently repeated 20 times and the obtained results are averaged over these runs. In simulation experiments, the efficiency of the proposed task scheduling method is compared with the above mentioned algorithms in terms of makespan and Cost. Figure 12 shows the average makespan of different algorithms. As it can be seen, the proposed Grid task scheduling algorithm, RHGGSA, significantly outperforms Min-Min, GA and GA-VNS. yet another observable point is that an increase in the number of tasks at the size of the problem results in an increase in makespan for all algorithms.

Table 1: Average makespan of different algorithms under scenario 1

No. Of Task	Average makespan			
	Min-Min	GA	GA-VNS	RHGGSA
50	41	48	36	32
100	85	95	72	67
150	115	121	95	89
200	158	172	133	127

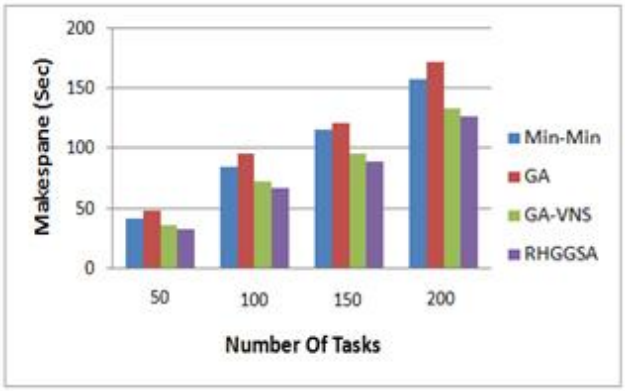

Fig.12 Effect of change in number of tasks on makespan

4.2 Scenario 2 (Effect of change Budget on makespan)

In the second experiment, scheduling algorithms with the time optimizing objective are compared by providing various budgets. It can be seen that in all cases the RHGGSA algorithm shows better performance than others and that budget increases always reduce implementation time. In fact, the higher the budgets given by the user, the less the time needed to perform application tasks by algorithms.

Table 2: Average makespan of different algorithms under scenario 2

Budget	Average makespan			
	Min-Min	GA	GA-VNS	RHGGSA
50000	169	172	133	127
70000	130	143	110	105
90000	117	121	92	87
11000	91	101	76	72
13000	81	87	67	61

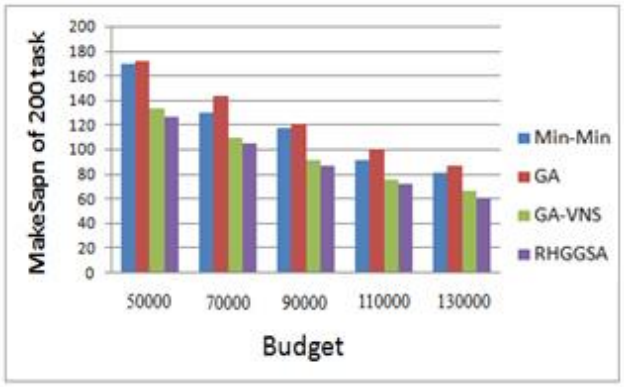

Fig.13 Effect of change Budget on makespan

4.3 Scenario 3 (Effect of heterogeneous tasks on makespan)

In the third experiment, a comparison of scheduling algorithms is done with different heterogeneous tasks. Figure 14 shows the results obtained from scheduling algorithms by the time optimization strategy. As seen, changes in time associated with increased heterogeneity of tasks reveal different trends for each algorithm. And with increased heterogeneity, the RHGGSA algorithm could achieve an acceptable decrease in calculation times.

Table 3: Average makespan of different algorithms under scenario 3

No. Of Task	Average makespan			
	Min-Min	GA	GA-VNS	RHGGSA
[90...120]	268	353	170	185
[70...140]	233	235	155	165
[50...160]	202	218	146	147
[30...180]	185	201	135	130
[10...200]	165	185	128	122

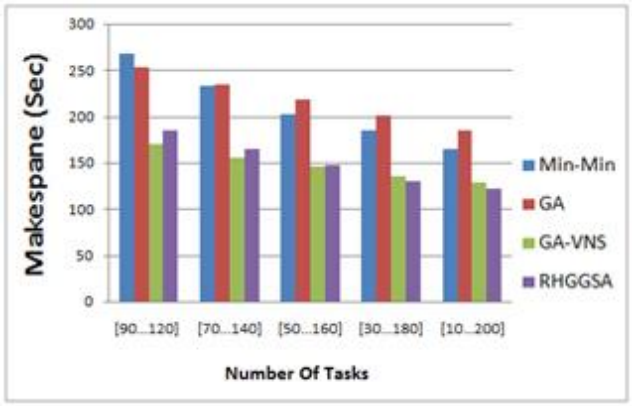

Fig.14 Effect of heterogeneous tasks on makespan

5. Conclusion

The task scheduling problem in the distributed systems is known to be NP-hard. In this paper, in addition to address the concept of scheduling, proposing to combine genetic and Gravitational Emulation Local Search (GELS) Algorithms, benefits of both algorithms is used to solve the scheduling problem. The algorithm minimizes the execution cost and makespan of user tasks by a balancing formula. We compared our algorithm with those of several existing methods and evaluated the performance to the changes in the number of tasks and cost of users. The obtained results showed that the proposed algorithm

significantly outperforms Min-Min, GA and GA-VNS in terms of makespan, cost. The proposed algorithm provides more freedom to determine the importance of task time and costs. And, by taking this into account, the algorithm uses a weighted objective function. The objective function of the algorithm proposed here, indeed, considers both completion time and task implementation costs so the resulting solutions would not exceed the deadline and user's costs.

References

[1] Foster and C. Kesselman, "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, USA, 1999.

[2] R. Zheng, H. Jin, "An Integrated Management and Scheduling Scheme for Computational Grid", Grid and Cooperative Computing, Lecture Notes in Computer Science Volume 3033, 2004, pp 48-56.

[3] M. Shojafar, S. Barzegar, M. R. Meybodi, "A new Method on Resource Scheduling in grid systems based on Hierarchical Stochastic Petri net", First International Conference on Information, Networking and Automation (ICINA 2010), China, 2010 Vol. 9, No 2,pp. V9-175-180.

[4] Abraham, R. Buyya and B. Nath, "Nature's Heuristics for Scheduling Jobs on Computational Grids", The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), Cochin, India, 2000, pp. 45-52.

[5] Y. Gao, H.Q. Rong and J.Z. Huang, "Adaptive grid job scheduling with genetic algorithms", Future Generation Computer Systems, Elsevier, 2005, pp.1510-161.

[6] J.E. Orosz and S.H. Jacobson, "Analysis of static simulated annealing algorithm", Journal of Optimization theory and Applications, Springer, 115 (1), 2002 , pp. 165- 182.

[7] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", J. of Parallel and Distributed Computing, vol.61, No. 6, 2001, pp. 810-837.

[8] B. Barzegar, A.M. Rahmani, K. Zamani far, "Gravitational Emulation Local Search Algorithm for Advanced Reservation and Scheduling in Grid Systems", First Asian Himalayas International Conference on(2009), 2009, pp. 1-5.

[9] S. Zheng, W. Shu, L. Gao, "Task Scheduling Using Parallel Genetic Simulated Annealing Algorithm", In: IEEE International Conference on Service Operations and Logistics, and Informatics, 2006, pp. 46–50.

[10]A. Tamilarasi, T. Anantha kumar, "An enhanced genetic algorithm with simulated annealing for job-shop scheduling", International Journal of Engineering, Science and Technology, Vol. 2, No. 1, 2010, pp. 144- 151.

[11]Q. Tao, H. Chang, Y. Yi, "A Grid Workflow Scheduling Optimization Approach for e-Business Application", International Conference on E-Business and E-Government, 2010, pp. 168- 171. DOI: http://dx.doi.org/10.1109/ICEE.2010.50

[12] Y. Hao, W. Huilin, Zh. Jiliu, "An Improved Genetic Algorithm with Limited Iteration for Grid Scheduling", In: Sixth International Conference on Grid and Cooperative Computing. IEEE Computer Society, pp. (221-227), ISBN 0-7695-2871-6, Washington, DC, USA.

[13] Z. Pooranian, A. Harounabadi, M. Shojafar, N. Hedayat, "New Hybrid Algorithm for Task Scheduling in Grid Computing to Decrease missed Task", World Academy of Science, Engineering and Technology, 2011, pp. 79: 5-9.

[14] W. Abdulal, O.A. Jadaan,A. Ahmad Jabas, S. Ramachandram. "Genetic algorithm for grid scheduling using best rank power", In: IEEE Nature & Biologically Inspired Computing (NaBIC 2009), December 2009, pp. 181-186, IEEE, ISBN 978-1-4244-5053-4, Coimbatore, India.

[15] W. Abdulal, S. Ramachandram, "Reliability-Aware Genetic Scheduling Algorithm in Grid Environment", In: IEEE International Conference on Communication Systems and Network Technologies, June 2011, pp. 673-677, IEEE, ISBN 978-0-7695-4437-3/11, Katra, Jammu, India.

[16] X. Xue,Y. Gu, "Global optimization based on hybrid clonal selection genetic algorithm for task scheduling", 2010, J Comput Inf Syst 6(1):253–261

[17] S. Garg, P. Konugurthi, R. Buyya, "A linear programming driven genetic algorithm for meta-scheduling on utility grids", In: 16th International Conference on Advanced Computing and Communications, Chennai, India, 2008, pp. 19-26.

[18] S. Garg, R. Buyya,H.J. Siegel, " Time and cost trade-off management for scheduling parallel applications on utility grids", Future Generation Computer Systems 2010; 26(8): 1344–1355.

[19] A.Y. Zomaya, R.C. Lee, S. Olariu, "An introduction to genetic-based scheduling in parallel-processor systems". In: Zomaya, A.Y., Ercal, F., Olariu, S. (eds.) Solutions to Parallel and Distributed Computing Problems - Lessons from Biological Science, Wiley, New York, 2001, pp. 111–133.

[20] Voudouris, chris, Edward Tsang, Guided Local Search. Technical Report CSM-247, Department of Computer Science, 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] B. Barzegar, A. Rahmani, K. Zamani far, "Advanced Reservation and Scheduling in Grid Computing Systems by Gravitational Emulation Local Search Algorithm", American Journal of Scientific Research, No. 18, 2011, pp. 62-70.

[23] L. Mohammad Khanli, M. Etminan Far, and A.M. Rahmani, "RFOH: a New Fault Tolerant Job Scheduler in Grid Computing", The 2nd International Conference on Computer Engineering and Applications (ICCEA), Bali Island, Indonesia, March 19-21, 2010.

Vahid, GhaedRahmati received the B.Sc degree in Computer Engineering (Software) from Islamic Azad University, Mahshahr Branch. He is currently M.Sc. student in Computer Engineering (Software) in Islamic Azad University Science and Research Branch of Ahvaz, Iran. His research interests include grid computing, task scheduling algorithms, wireless sensor networks and cloud computing.

Seyed.Enayatallah, Alavi B.S. Computer Eng. From Isfahan University of Technology 1993 Isfahan, Iran, M.S. in Computer Eng. Shiraz University 1996 Shiraz, Iran, Ph. D. in Computer Eng., BNTU 2011Minsk,belarous, assistant Pro. At ShahidChamran Uni. Research interests includes Image processing, and neural network. More than 30 papers.

Iman, Attarzadeh received his M.Sc. in Computer Architecture from Islamic Azad University, Tehran, Iran, in 2005. he was the Head of Department of Computer Science at the University in 2005. His current research interests include software engineering (project management, formal method, algorithm design, and embedded systems), soft computing theories and techniques, image processing, and computer Architecture. More than 30 papers.