A Synthetic Heuristic Algorithm for Independent Task Scheduling in Cloud Systems

Arash Ghorbannia Delavar, Yalda Aryan

Department of Computer Engineering, Payam Noor University, PO BOX 19395-3697 Tehran, IRAN

Abstract

In this paper, we present a synthetic method based on genetic algorithm, for independent task scheduling in cloud computing systems. Task scheduling is a major issue in large-scale distributed systems that impresses on system performance. For some reasons such as heterogeneous and dynamic features in cloud environment, task scheduling has appeared as a NP-complete problem. Our proposed algorithm (SHIS), by some goal oriented operations such as, making an optimize initial population, dual step evaluation, and also, running the tasks by a special ordering considering resource load balancing and quality of service, achieves the optimize makespan. It also decreases the probability of task failure rate on running, based on the resource failure frequency rate, and also decreases the task starvation problem. It supports the scheduling for new entered tasks in system by a dynamic method. The experimental results show that the proposed method solution is better than the other studied algorithms.

Keywords: *Cloud computing, task scheduling, resource allocation, Independent task scheduling, non-preemptive scheduling.*

1. Introduction

Cloud computing, is a paradigm for distributed computing with cluster and grid and also its characteristics [1]. It includes many resources and requests, in an effort to share the resources as services on the Internet that typically divided into three levels of service offerings: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These levels support virtualization and management of differing levels of the solution stack [2]. In the cloud environment, the processes are allocated on dynamic and virtual resources pool by online and on-demand forming.

Cloud computing service providers, make the large-scale network servers form the pool of large-scale virtual resources. IaaS level is the delivery of hardware (server, storage and network), and associated software (operating systems virtualization technology, file system), and provides a large amount of computational capacities to service remote users in a flexible and efficient way. In this level, resources have provisioned in the form of Virtual Machines (VMs) deployed within the cloud equipments consisting of: Data-center, physical resources and etc, for fulfilling the requests.

Task scheduling is a key process for IaaS; it means to assign the requests to resources in an efficient way, considering cloud characteristics. It takes VMs as scheduling units for mapping physical heterogeneous resources on tasks. Each VM is an abstract unit of computing and storage capacities in cloud. Although the task scheduling in heterogeneous environment such as cloud computing systems with dynamic characteristics is a NP-complete problem, so it has to be done immediately and automatically.

For this problem, they have been presented various methods such as: Greedy (First-fit) [3], [4], [5] and Round-Robin (used by some cloud systems such as Eucalyptus [4]), queuing system (used in Open Nebula), advanced reservation and preemption scheduling [6], max-min [3] and etc. Some of them do not consider parameters such as maximum usage of physical resources, or resources' load balancing and quick response.

To solve NP-complete problems, it almost is used evolutionary and heuristic algorithms, toward scheduling problem in distributed systems used some algorithms such as: Partial swarm optimization (PSO), Simulated Annealing, Tabu Search, Genetic Algorithm (GA) and etc. Among them GA is good to gain an optimized response in parallel search [7], [8].

In this work, we suggest, SHIS algorithm based on GA, consisting optimal characteristics of Min-Min, MMC, MXC and Round-Robin algorithms with efficient parameters in cloud environment, to produce a proper scheduling with immediate response for cloud.

2. Related work

Since task scheduling in dynamic environments such as cloud computing is a NP-complete problem, so obtaining the proper solution with minimum execution time of batch of tasks and load balancing in resources, is too hard, because:

- Amount of entered requests are more than the amount of resources, and they are entered each time and never stop.

- The request characteristics are always variable and indeterminate, such as: arrival time, execution time, necessary memory and etc.
- The cloud environment is a collection of heterogeneous resources.
- The resources have dynamic hardware and software characteristics such as: average of workload on node, CPU usage, memory usage, and network features such as current traffic and bandwidth.

Therefore, we have to consent an optimized solution near to main solution.

The different presented algorithms have been taken into consideration some parameters such as: estimated task execution time, task computational requirement (MFLOP) or communication cost. For example, in heuristic min-min (MM) and max-min (MX) algorithms [3], [8], [9], [10], that are begun by batch of ready and unmapped tasks, compute the execution time of each task on each processor and add to ETC(i,j) matrix with M*N cells (M is amount of tasks, N is amount of processors) . The ETC matrix is directly equivalent to the makespan. Then for each task the processor which will compute it with minimum amount of time will be select, and add to set. In MX, the tasks are assigned to the processor queue by the largest to smallest completion time. This process is repeated until all tasks are mapped to processors. The MM scheduler is similar to MX except that, after the set of minimum completion time is found, tasks are assigned to the queue in ascending completion time order. In MXC, MMC, LLXC and LLMC algorithms, is considered communication cost too. The makespan is set of ETC(i,j) + C(i,j), where C(i,j) is the estimated communication overhead associated with executing task i on processor j. Each method is suitable for different situations. MX is good when there are more large tasks than small tasks, and MM is opposite of it.

The algorithms which estimate communication cost are used by high communication overhead systems such as distributed systems. But these methods aren’t suitable for cloud with mentioned reasons and we should use less responding and running time methods. In some policies, First-fit [4], [5], or Round-Robin (Eucalyptus uses them) are being used for cloud system, so that, the requests will be run on all of resources and implemented a little load balancing, but they don’t support optimal usage of resources and a good load balancing. Also, some algorithms, have used evolution method such as GA, and changed its operations such as initial population [11].

The SHIS proposed algorithm, in addition to mentioned parameters such as resource access cost and load balancing, it considers order of tasks according to workload, wasted time, priority and deadline, and gains an optimal solution using genetic algorithm by decreasing the task failure rate on running, task starvation rate and optimizes the completion time of tasks.

3. Proposed Algorithm

In this approach, we suggest a dynamic scheduling method, using genetic algorithm, because the GA is, simple and authentic in searching problems and can be used in parallel methods among other experiments [7].

The scheduling in distributed systems such as cloud depends on many parameters, for heterogeneous and dynamic features and also more workload of these systems than local networks. But some parameters are more important. Some of them are hidden and some are obvious [11]. In this proposal, we focus on some parameters which are effective in a good scheduling.

The SHIS proposed algorithm, in addition to mentioned parameters such as resource access rate, quality of service and load balancing, it considers order of tasks accordance with workload, wasted time, priority and deadline, and gains an optimal solution by genetic algorithm. This proposed method tries to obtain an optimal tasks mapping on resources by minimizing completion time of tasks or, completion time of the last task (makespan). Furthermore, it decreases the task failure rate on running based on the resource failure frequency rate, and also increases the quality of responding using a special task ordering to run and to decrease the task starvation rate by proper parameters in it.

The features in this approach are:

1) Tasks are entered in system, each time.
2) All tasks are independent.
3) Each resource can process more than one request.

Generally, the proposed algorithm is as follow:

Input: Set of available resources and unmapped tasks.
Output: An optimized generated dynamic schedule.

```
1.  Calculate the available resources characteristics
    and network workload.
2.  Repeat
3.    for $\forall t_i \in T^{ready}$ do
4.      The tasks are ordered by workload in ascending.
5.      Make initial population.
6.      Evaluate population.
7.      While the stop conditions are met.
8.        Cycle crossover operation.
9.        Goal oriented mutation operation.
10.       Select the best chromosomes.
11.     End while.
12.     Save the best solution.
13.     Sort tasks, according to the candidate resource,
        in descending by Eq. (9).
14.   End for.
```

```
15.  Dispatch all mapped tasks on candidate
     resources in obtained solution.
16.  Update the ready task list.
17.  Update the virtual resource list regarding current
     characteristics from data-center.
18.  Until there are unscheduled tasks.
```

The SHIS algorithm

In the cloud computing environment, there are data-center system that is assumed to collect and save the information. The collectors in data-center in the each work-node are responsible for collecting the static and dynamic information of resources and tasks. Some key static information such as: physical memory storage space, virtual memory storage space, disk storage space, and etc are collected, and some dynamic information such as: the load average of the node itself, the number of the running tasks, the current running tasks' number of threads, and the status of these tasks, CPU usage and etc, are captured periodically or based on the polling strategy or others, and are sent to the Data Receiver of the master node through the communication component. These data are updated frequently, and in real-time form [12].
We can use some static and dynamic information for scheduling. Since a lot of tasks are received in the cloud system in each time, so the workload and other dynamic information such as current network-traffic could impress to select the good candidate resource for task.
In this algorithm, first, by sorting the tasks using workload in ascending (like the first step in Min-Min), and making a virtual list from available resources with updated characteristic (by data-center), the algorithm (SHIS) starts. Then the SHIS scheduling algorithm uses the GA method. It makes a goal oriented initial population using combining the MMC and Round-Robin algorithm, and some good tests, according to description. Also it performs the cycle crossover [9] and targeting mutation. Finally, after the best solution (candidate resources) is obtained by GA, the algorithm, sorts the tasks by special ordering using efficient parameters in descending to run, and the tasks are dispatched on candidate resources.
Since in cloud system, the new tasks are entering each time, so the algorithm supports them by updating the available resources and their characteristics list from data-center to resumption.

3.1 Encoding

In GA method, every solution is encoded as a chromosome. Each chromosome has N genes, as chromosome length. A schedule has appeared in form of a solution or in other word, a chromosome. The fig 1, depicts a chromosome in SHIS method.

T17	T2	T8	T0	T12	…	T24
R2	R9	R3	R1	R8	…	R5

Fig. 1 A sample encoding of a schedule as a chromosome in SHIS.

Here, the tasks are ordered by workload and then file-size in ascending, and the tasks should map to suitable resources from set of available resources. In our algorithm, to make a chromosome, each task mapped to a selected resource from a virtual list of available resources, according to the data-center information. The virtual list will be update in some operations such as initial population.

3.2 Initial population

Let us say that, a set of multiple possible solutions (chromosomes) are referred to a population. The initial population is made randomly in normal genetic algorithm.
We mean to make good and goal oriented initial population, that leads to find the response quickly, that be as close as possible the best solution.
For this purpose, in making initial population, after the tasks are sorted by workload in ascending, they will be placed in chromosome, and for each task, a suitable resource will be selected with minimum running time from virtual resource list by these equations:

$$T^{ct} = (T^{len} / R^{w}) + R^{r} \qquad (1)$$

$$ECCT = T^{ct} + (T^{f} + T^{out}) / N^{bw} \qquad (2)$$

And these conditions:

$$\begin{cases} T^{ct} / T^{dl} < 1 & \text{: if Task Priority is high.} \\ T^{ct} < T^{dl} + R^{d} & \text{: else} \end{cases} \qquad (3)$$

$$T^{f} + T^{out} < R^{s} \qquad (4)$$

$$T^{ct} / R^{ff} < Trsh \qquad (5)$$

Where:

T^{ct}: Task completion time
N^{bw}: Bandwidth between the scheduler and computing node
T^{len}: Task Length
T^{dl}: Task Deadline
Trsh: Threshold
R^{w}: Current Resource workload
R^{r}: Resource readiness time
T^{f}: File size of task
T^{out}: Output size of task
R^{s}: Current Resource storage size
R^{d}: Resource delay
R^{ff}: Resource failure frequency rate

In fact, the best suitable resource with completion time of task and communication cost between all resources, will be candidate for each task by Eq. (1) to Eq. (2). If a resource is not suitable according to Eq. (3) to Eq. (5), the next resource (The Kth best) will be checked. This checking method causes to consider the waiting time of tasks by mentioned parameters such as deadline as much possible, and also decreases the probability of task failure when it will be run because it pay attention to the failure frequency rate of resource [13]. We also assume the N^{bw} parameter is the bandwidth between the scheduler and computing node [14]. This test is the first evaluation step for selecting the suitable resources.

The threshold is assigned within system policy. This process is repeated for all genes as Best fit. To make initial population, for first chromosome, we select the fittest resource from the first resource in virtual list, but for the second chromosome, the fittest candidate resource, will be searched from the second resource in virtual list, and so on like Round-Robin method but for resource selection. This process, continues to make a population. But in making each chromosome, if the algorithm could not find a good resource, go backs to first resource in virtual list.

This method makes us sure, that all resources, will be selected for population. Thus all possible solutions can almost be made, and also attended the balance the load on resources. After each available resources are selected as candidate, the virtual list will be updated based on the rest processing capacity on current resources' workload. Because the tasks are more than resources, so each resource processes several tasks.

3.3 Fitness function

To recognize the value of a solution, we should evaluate it by a fitness function with efficient parameters in quality of solution. In GA, the fitness function is applied on all solutions and computes their value, and then a solution with the best value, based on parameters placement policy, is minimum or maximum as the fittest solution. In this paper we consider some parameters to minimize the makespan as follow:

N^{cc}: Network communication cost
n: Chromosome Length
F_i: Fitness value of i-th Gene
Fitness $_{total}$: Fitness value of Chromosome

Fitness value for each gene is computed in this way:

$$F_i = (T^{len} / R^w) + N^{cc} \quad , N^{cc} = (T^f + T^{out}) / N^{bw} \qquad (6)$$

The network communication cost is mentioned in DSQGG algorithm [15]. Total fitness value for a chromosome is computed by sum of fitness values of all genes:

$$\text{Fitness}_{total} = \sum_{i=0}^{n-1} F_i \qquad (7)$$

And the chromosome with minimum value is the best solution among the others.

$$\text{Target is: Minimum (Fitness}_{total}) \qquad (8)$$

The algorithm tries to find the solution by minimizing the fitness value as much possible by crossover and mutation operations while running.

3.4 Crossover operation

As each resource can process several tasks, so adjusting a proper mapping the tasks on resources, impresses to balance the load on resources and minimize the running time of tasks. Therefore crossover operation can increase probability of gene displacing for obtaining the load balancing.

Here, we use cycle crossover that were used in [9]. So two parents chromosomes are selected randomly from the population, as A and B. One gene in A chromosome, is selected randomly too. We named it A_I. Then the same gene in B is marked too as a visited. B_I value is noted. This value will be searched in A chromosome and the gene of this value is denoted as J. Then A_J and B_J are marked as visited, and the value in B_J is searched for in A. This process continues until a gene in A is visited twice. Now a cycle has been found. All visited genes are replaced with each other to produce two new child strings. The authors assert that: "This method ensures that, the child chromosomes generated are valid. Since both parents contain the exact same character, just in a different order, a cycle will always be found".

3.5 Mutation operation

A chromosome and one of its genes will be chosen randomly and a resource will be selected randomly from virtual list of resources. If the new fitness of chromosome is better than previous, the new selected resource will be replaced. The selecting and checking new candidate resource for a better result will be repeated in a limited case. The mutation operation causes, GA does not stop in the local minimum, but this method in mutation causes, the good solution will be found quickly.

3.6 Termination conditions

The algorithm will be finished until one of termination conditions will be met. The conditions to stop the process are:

- Number of generations, will reach to a maximum bound.
- The makespan of the best solution will not be changed, after the number of generations.
- All chromosomes converge to the same mapping.

3.7 Task ordering by preferences

In variant systems according to their policy, the order of executing of requests is different. For example, some preferences consider some parameters such as: Arrival time of task, workload or task length, or completion time.
In cloud system with mentioned features, we use: sensitivity of task execution as Task Priority, workload, deadline, and wasted time. Thus, after obtaining the best solution by GA, the tasks are ordered, considering candidate resources, in a queue by following Eq.

$$f_{sort} = (2^{1/Tp} / (T^{dl} - wt)) * T^{w}$$
$$, wt = T^{at} - ct \qquad (9)$$

Where: T^p is task priority, T^{dl} is deadline, T^w is workload, wt is the wasted time of task, T^{at} is task arrival time, and ct is current time.
Our aim of paying attention to these parameters is the effect of preference in running tasks, and also decreasing the task starvation problem for tasks with low priority. Task priority (T^p) is assigned by system policy. We suppose the priority parameter is set by descending values, from less value for higher priority, to more value for lower priority of tasks. This ordering will be applied to run the tasks on resources, after the scheduling algorithm has produced a solution for mapping. In fact the tasks will be run in sequence on this ordered list.

4. Performance evaluation

To evaluate this algorithm, we compare it with well-known algorithms like First-fit, Round-Robin and Standard GA, and PN algorithm [9].
The used environment to simulation is Cloudsim tool. The used parameters are represented in table as follow:

Table 1: The used parameters for simulating in the first experiment

Task lengths	20000 ~ 40000 (number of machine instructions)
Node Processing capacities	100 ~ 500 (machine instructions per second)
Node memory size	256 ~ 512 (MB)
Node storage size	500 ~ 10000 (MB)
Task file and output size	400 ~ 800 (MB)
Band width	100 ~ 1000 (Mbps)

The obtained results of comparison by charts are:

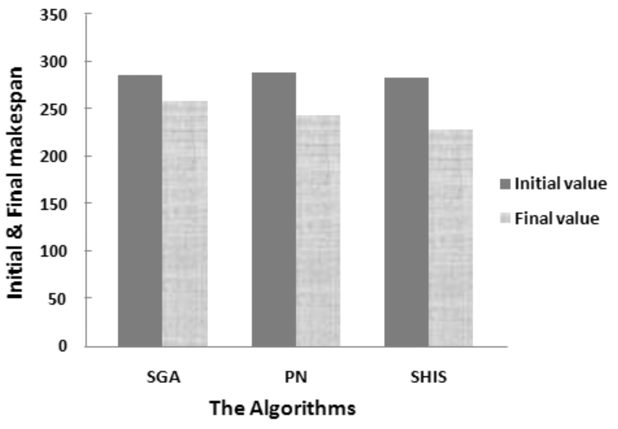

Fig. 2 Simulating the results of proposed algorithm in contrast with the others.

Fig. 2 represents the initial and final makespan for SGA, PN and SHIS. We perceive in SGA, the time that system responds to tasks (makespan) is more and in SHIS is less than others. It shows that quality of final makespan is optimized in contrast with SGA about 11.87% and PN about 6.25%. Whereas the initial makespan in SHIS is more optimize than the others.

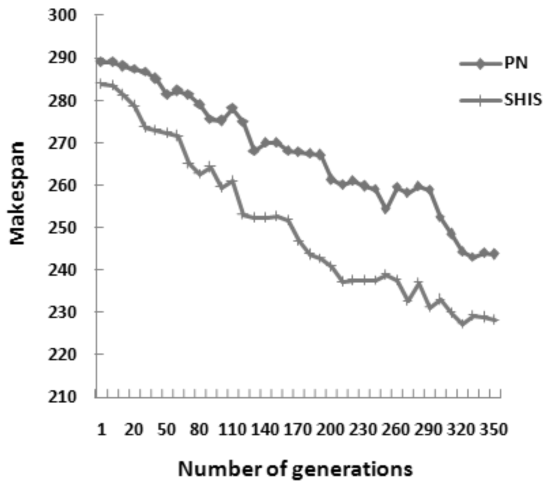

Fig. 3 Results of proposed algorithm in contrast with PN in several iteration of generations.

We also compare SHIS with PN algorithm, Fig. 3 shows the results, by incrementing the number of generation. SHIS produces the solutions with more optimal makespan, because of the optimized initial population in the beginning of algorithm, and so the leaded operation such as mutation.

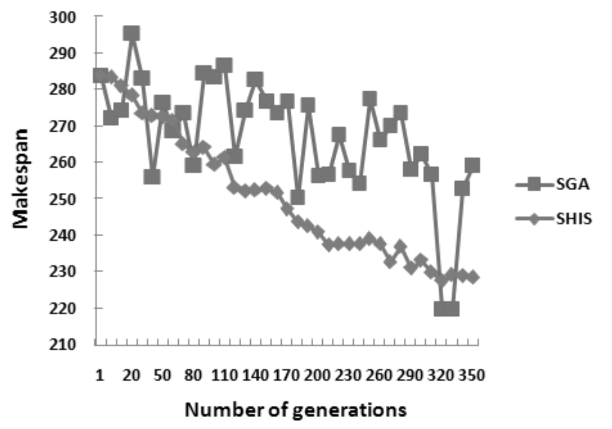

Fig. 4 Results of proposed algorithm in contrast with SGA in several iteration of generations.

In Fig. 4, we perceive better gradient rate of crump in SHIS in contrast with SGA. The random essence of SGA causes, the quality of produced solution will be variable, whereas each operation in SHIS is controlled. This subject demonstrates reliability of produced solution of SHIS is more than SGA.

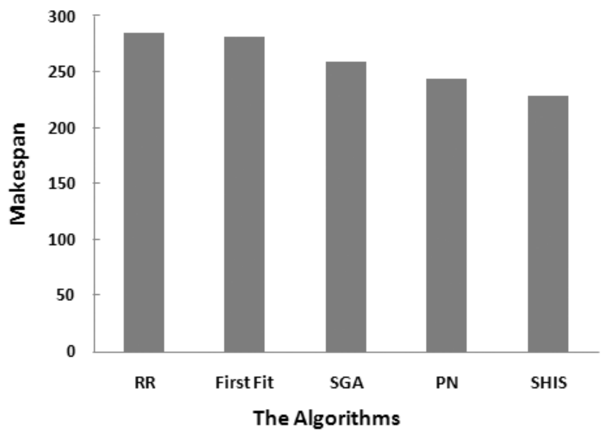

Fig. 5 Results of proposed algorithm in contrast with other algorithms.

But in Fig. 5, other algorithms are compared with proposed one, too. The RR and First-Fit are used on some cloud systems such as Eucalyptus [4]. This chart illustrates the results and better makespan for SHIS in contrast with RR about 19.82%, First-Fit about 18.9%, SGA about 11.87%, and PN about 6.25%.

5. Conclusions

In this paper, we suggest a dynamic task scheduling method in cloud computing environment based on GA, that uses the optimize characteristics Min-Min, MMC, MXC and Round-Robin algorithms.
This method makes initial population by the virtual list of resources and their updated properties, and an ascending ordered list of tasks by workload that makes to find the good solution quickly. The algorithm performs the goal oriented operations such as initialize population, and mutation. For making the initial population, it uses the suitable evaluation in two stages, a test stage and computing a fitness function stage, that lead the process to select good resources for mapping on tasks and consequently a good solution that is as close as the best possibility. The test makes us sure to obtain a reliable answer, decreases probability of task failure rate by resource failure frequency rate, and also considers deadline, which the waiting time of user (request) is regarded as much possible. When the scheduling algorithm gains the suitable resource for each task, the tasks will be run, based on the sorted list according to workload, priority, deadline and wasted time, to support the preferences the running of tasks and so, it prevents task starvation problem for low priority tasks. The SHIS results are compared with First-fit, Round-Robin, SGA and PN algorithms. We have perceived, the produced solution by SHIS, decreases the running time of tasks, in comparison with Round-Robin about 19.82%, First-Fit about 18.9%, SGA about 11.87%, and PN about 6.25%, whereas it supports the load balancing and regards the quality of service with the important parameters. Because of the operations are leading in our method, in various iteration of generations, the produced makespan using SHIS are more uniform and reliable than SGA, and this issue demonstrates reliability of produced solution by SHIS.

References

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility “,Future Generation Computer Systems, ELSEVIER (2008).
[2] Steve Bennett, Mans Bhuller, Robert Covington ,”Architectural Strategies for Cloud Computing”, August (2009).
[3] Hesam Izakian, Ajith Abraham, Senior Member, IEEE, Václav Snášel, “Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments”.
[4] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. So-man, L. Youseff, and D. Zagorodnov, “ The Eucalyptus open-source cloud-computing system”, IEEE International Symposium on Cluster Computing and the Grid (CCGrid ’09), 2009.
[5] R. P. BRENT, “Efficient Implementation of the First-Fit Strategy for Dynamic Storage Allocation “,Australian National University, Pages 388-403, ACM Transactions on

Programming Languages and Systems, Vol. 11, No. 3, July (1989).

[6] Hai Zhong, Kun Tao, Xuejie Zhang, "An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems ",The Fifth Annual ChinaGrid Conference - IEEE (2010).

[7] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing" , (2009) IEEE.

[8] Muthucumaru Maheswaran, Shoukat Ali and Howard Jay Siegel, Debra Hensgen, Richard F. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems "Journal of Parallel and Distributed Computing 59, 107_131 (1999).

[9] Andrew J. Page, Thomas M. Keane, Thomas J. Naughton, "Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system ",Journal of Parallel and Distributed Computing 70, 758_766 ELSEVIER (2010).

[10] Jia Yu and Rajkumar Buyya, " Workflow Schdeduling Algorithms for Grid Computing ",Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Australia.

[11] Myungryun Yoo, " Real-time task scheduling by multi objective genetic algorithm" The Journal of Systems and Software 82, ELSEVIER (2009).

[12] Junwei Ge, Bo Zhang, and Yiqiu Fang, "Research on the Resource Monitoring Model Under Cloud Computing Environment ",Verlag Berlin Heidelberg - Springer (2010).

[13] Maria Chtepen, Filip H.A. Claeys, Bart Dhoedt, Member, IEEE, Filip De Turck, Member, IEEE, Piet Demeester, Senior Member, IEEE, and Peter A. Vanrolleghem, "Adaptive Task Checkpointing and Replication: Toward Efficient Fault-Tolerant Grids", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 20, NO. 2, FEBRUARY 2009.

[14] Arash Ghorbannia Delavar, Vahe Aghazarian, Sanaz Litkouhi and Mohsen Khajeh naeini, "A Scheduling Algorithm for Increasing the Quality of the Distributed Systems by using Genetic Algorithm ",International Journal of Information and Education Technology, Vol. 1, No. 1, April (2011).

[15] Delavar A.G., Rahmany M., Halaakouie A., Sookhtsaraei R., "DSQGG: An optimized Genetic-based algorithm for Scheduling in Distributed Grid ",2nd International Conference, Computer Technology and Development (ICCTD), (2010).

Arash Ghorbannia Delavar received his M.Sc. and Ph.D. degrees in computer engineering from Sciences and Research University, Tehran, IRAN, in 2002 and 2007. He obtained the top student award in Ph.D. course. He is currently an assistant professor in the Department of Computer Science, Payam Noor University, Tehran, IRAN. He is also the Director of Virtual University and Multimedia Training Department of Payam Noor University in IRAN. Dr. Arash Ghorbannia Delavar is currently editor of many computer science journals in IRAN. His research interests are in the areas of computer networks, microprocessors, data mining, Information Technology, and E-Learning.

Yalda Aryan received her B.Sc. in computer engineering from Azad University, Arak, IRAN, in 2000, and is a M.Sc. student in computer engineering in Payam Noor University. Her research interests include computational intelligence, Grid computing and cloud computing.