Computer Engineering and Applications Vol. 5, No. 3, October 2016

Exploration Based Genetic Algorithm for Job Scheduling in Grid Computing

Hanaa Abdelrahman[1], Adil Yousif[1] , Mohammed Bakri Bashir[3]

[1]*Faculty of Computer Science - University of Science and Technology- Sudan*
[2]*Faculty of Science and Technology -Shendi University-Sudan*

ABSTRACT

Grid computing presents a new trend to distribute and Internet computing to coordinate large scale heterogeneous resources providing sharing and problem solving in dynamic, multi- institutional virtual organizations. Scheduling is one of the most important problems in computational grid to increase the performance. Genetic Algorithm is adaptive method that can be used to solve optimization problems, based on the genetic process of biological organisms. The objective of this research is to develop a job scheduling algorithm using genetic algorithm with high exploration processes. To evaluate the proposed scheduling algorithm this study conducted a simulation using GridSim Simulator and a number of different workload. The research found that genetic algorithm get best results when increasing the mutation and these result directly proportional with the increase in the number of job. The paper concluded that, the mutation and exploration process has a good effect on the final execution time when we have large number of jobs. However, in small number of job mutation has no effects.

Keywords*:* Grid Computing, Genetic Algorithm, Crossover, Mutation, Exploration

1. INTRODUCTION

The term “the Grid” was coined in the mid1990s to denote a proposed distributed computing infrastructure for advanced science and engineering [1]. Computing resources are geographically distributed under different ownerships each having their own access policy, cost and various constraints [2]. Grid computing displays a new direction for the distribution of the Internet and computing to coordinate the sharing of resources is homogeneous on a large scale problem solving in dynamic, multi-institutional virtual organizations. Grid computing has high heterogeneous computing resources, ranging from computers and one workstation, a group of workstations to super computers. With network technologies, it is possible to build large-scale applications on the network environments [3]. Grid computing has emerged as a distributed methodology that coordinates the resources that are spread in the heterogeneous distributed environment, Grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distributed across multiple administrative domains based on their availability, capability, performance, cost and user’s quality-of-service requirements . A Grid is loosely coupled, geographically distributed and heterogeneous computers in the grid

donate printers, application software, disk storage, CPU power etc, Grid uses a middleware layer to communicate with heterogeneous hardware and datasets [4].

A computational Grid is a hardware and software infrastructure that provides Dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities .It is a shared environment implemented via the deployment of a persistent , standards-based service infrastructure that supports the creation of, and resource sharing with in, distributed communities[5]

Scheduling is one of the most important problems in computer systems, such as the network. To increase performance, there is a need to schedule the network performance and efficiency. Unfortunately, the dynamic nature of the network and also the demands of different users, due to the complexity of the problem is the schedule of the network. This means that the efficiency of resources and the network is always changing dynamic nature. Possible for the user and other financial cost of the process is important. The goal of the network schedule is to find the optimal allocation of resources and to function, and to overcome the heterogeneous resources and maximize overall system performance. The network is known as the tabulation in the decision-making process of scheduling resources listed on multiple administrative domains. And this process can include searching multiple administrative domains to use one device or schedule one job to use multiple resources in a single location or multiple locations. Define the function to be anything needs to be a resource request of bandwidth, to application, for a range of applications. We use the term resources to mean anything can be scheduled: machine, disk space, network quality of service, and so on [6]. From the point of view of scheduling systems, a higher level abstraction for the Grid can be applied by ignoring some infrastructure components such as authentication, authorization, resource discovery and access control. Thus, in this paper, the following definition for the term Grid adopted: "A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements" [7]. Each job consists of a set of tasks . The task has a number of processing requirements. In the exceptional case when the job consists of only one task we recognize with and the processing requirements with. Each task is associated with a number of resources and possibly be processed on any of the resources of [8] . Job scheduling on grid computing represents a great challenge. Genetic algorithm is one of the widely used algorithms for scheduling on grid computing. The process of exploring new solution in the genetic search space is an important issue. Because sometimes the initial population so far from the best solution because initial population is random.

This paper contains six sections. Section two describes the related works. Section three gives details about genetic algorithm. Section four illustrates the proposed scheduling algorithm. Section five describes simulation environment and the results. We concluded in section six.

2. RELATED WORK

Natural metaheuristics such as hill climbing, genetic algorithm and particle swarm optimization are inspired by the natural phenomena to solve complex combinatorial real world problems. Hill Climbing (HC) is a local search optimization mechanism. It is an iterative technique that begins with a random

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

solution in the search space, and then tries to discover optimized solutions by continuously modifying a single element of the current solution. If the modification generates a better candidate solution, the modification is considered, otherwise the modification is discarded [9]. Similar to the HC mechanism is Tabu Search (TS) which can be defined as "a strategy for solving combinatorial optimization problems whose applications range from graph theory and matroid settings to general pure and mixed integer programming problems"[10]. Tabu Search has superiority over Hill Climbing as it has a memory facilitates in keeping on exploration even if the improving movement is absence. Moreover, the memory prevents the TS scheduling mechanism form trapping in local optimum that has been visited previously. However, TS uses single search path of solutions and not population search or tree search. In single search path techniques for each candidate solution in this path, the technique assesses a set of moves through the solution search space and chooses the best toward the next solution.

The main challenge for optimization mechanisms is to increase the possibility of finding the global optimal solutions. Greedy optimization mechanism such as HC and TS strive to improve each single step; they can find the solution fast. However, greedy optimization mechanisms are often traps in local optimal solutions[11]. Evolutionary optimization mechanisms, such as differential evolution and genetic algorithm have a limited range of movements; which reduces the likelihood of trapping in local or sub optimal solutions. However, they are slower in finding optimal solutions as a result of the complexity in managing the population movements[12]. TS, HC and GA are used to schedule jobs on computational grid; these metaheuristics scheduling mechanisms outperform the grid basic scheduling mechanisms in most cases[13, 14].

Swarm intelligence (SI) is a new class of nature inspired metaheuristics based on population optimizations. The population elements are particles that aim to find the global optimal candidate solution by communicating with other particles and with the environment. In SI such as particle swarm optimization (PSO), ant colony optimization (ACO) and firefly algorithm (FA) particles do not die, and rather they move throughout the search space themselves. PSO and ACO have been used as scheduling mechanisms to map the jobs to resources on computational grid in several researches [8, 15-17].

3. GENETIC ALGORITHM

Genetic algorithm (GA) is an optimization metaheuristic that imitates the process of natural evolution. GA generates a random initial population of feasible candidate solutions, that is a set of integer random numbers, and each solution represents a chromosome. Each chromosome is a vector indexed with a number from 1 to NP, where NP is the population size. After the initial population is generated, the population chromosomes are refined using crossover and mutation operations. GA has a limited range of movements; and this reduces the likelihood of trapping in local optimum solutions. Nevertheless, they are slower in discovering optimum solutions as a result of the complexity in managing the population movements [12]. Figure 1 describes the follow of genetic algorithm.

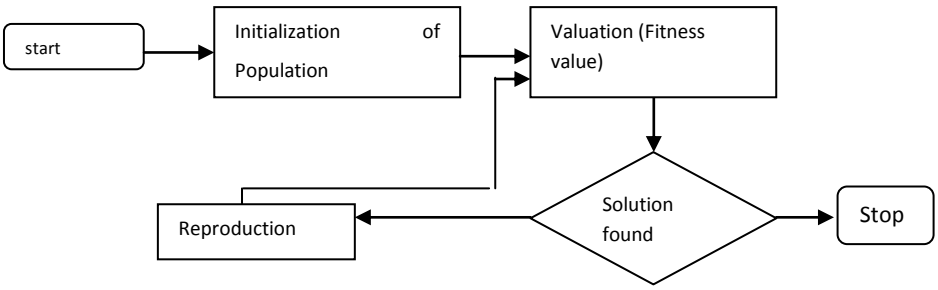

FIGURE 1. Flow Chart of Genetic Algorithm.

Abraham et al (2000) presented a hybrid of three of the nature's mechanisms GA, SA and TS for job scheduling on computational grids. The hybrid mechanism showed a better convergence and enhanced the search process of GA. A simple version of GA is utilized in [18] to find an optimal or suboptimal schedules for the grid job scheduling problem. A Hierarchical mechanism for scheduling jobs using Genetic Algorithms is proposed for computational grid to increase the scalability of the scheduling process [19]. In [20] a suboptimal optimization mechanism for scheduling job on computational grid based on genetic algorithm was developed. The Suboptimal mechanism is capable to converge in simple problems. However, in complex scheduling problems the mechanism cannot converges the search space. Two models, single service and multiple services, are presented by [21]to estimate the completion time for jobs on computational grid using GA. To enhance GA further, an integration between job clustering using fuzzy C-Mean and a scheduling mechanism using GA was developed in [22]. To reduce the repetitions of the generations in GA, Delavar et al (2010) introduced a new scheduling mechanism to achieve a higher speed and to decrease the communication costs. To speeds up convergence and to minimize the search time, a rank based genetic scheduler is proposed by Abdulal et al. (2010). Furthermore, they utilized MCT schedule to initialize the algorithm.

Genetic Algorithm (GA) GAs are adaptive methods that can be used to solve optimization problems, based on the genetic process of biological organisms [2]. The Standard Genetic Algorithm follows the method of haploid sexual reproduction. In the Standard Genetic Algorithm, the population is a set of individual binary integers such as 1001011. Each individual represents the chromosome of a life form. There is some function that determines how fit each individual is, and another function that selects individuals from the population to reproduce. The two selected chromosomes crossover and split again. Next, the two new individuals mutate. The process is then repeated a certain number of times. Let's consider each of the highlighted notions in more detail. Fitness is a measure of the goodness of a chromosome, that is, a measure of how well the chromosome fits the search space, or solves the problem at hand. For the standard genetic algorithm, the fitness f is a function from the set of possible chromosomes to the positive reals. Selection is a process for choosing a pair of organisms to reproduce. The selection function can be any increasing function, but we will concentrate on fitness-proportionate selection, whose selection function is the probability function, crossover is a process of exchanging the genes between the two individuals that are reproducing. There are several such processes, but we will consider only one-point crossover, a process that is both standard and simple. A random integer i is selected uniformly between 1 and n. This is the place in the chromosome at which, with probability p c , crossover will occur. If crossover does occur, then the chunks up to i of the two chromosomes are swapped[23].

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

We describe the seven steps in the Standard Genetic Algorithm as described in Figure 3. Start with a population of n random individuals each with l-bit chromosomes. Calculate the fitness f(x) of each individual. Choose, based on fitness, two individuals and call them parents. Remove the parents from the population. Use a random process to determine whether to perform crossover. If so, refer to the output of the crossover as the children. If not, simply refer to the parents as the children. Mutate the children with probability p m of mutation for each bit. Insert the two children into an empty set called the new generation. Return to Step 2 until the new generation contains n individuals. Delete one child at random if n is odd. Then replace the old population with the new generation. Return to Step 1. Figure 2 illustrates pseudo code genetic algorithm.

```
Simple Genetic Algorithm ( )
begin
initialize population;
evaluate population;
While termination criterion not
reached
{
select solutions for next
population;
perform crossover and
mutation;
evaluate population;
}
```

FIGURE 2. Genetic Algorithm Pseudo Code

4. EXPLORATION BASED GENETIC ALGORITHM

Crossover operation may produce degenerated population. In order to avoid this, mutation operation is performed. Mutation operation can be bit flipping, interchanging, inversion, insertion, reciprocal exchange or others. In case of insertion a node is inserted at random position in the string. This is because a node along the optimal path may be eliminated through crossover. Using insertion, it can be brought back as shown in Figure 3. Once mutation is completed, the offspring generated by mutation have to be validated with the same process used in crossover.

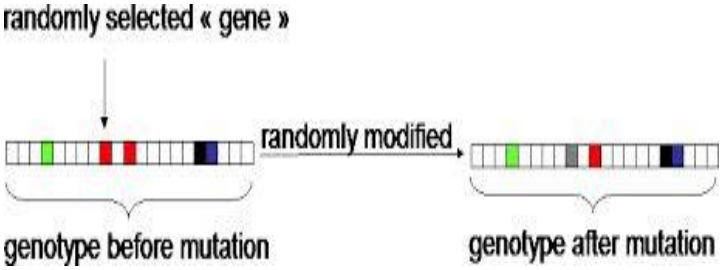

FIGURE 3: Mutation process

Exploration process occurs during the mutation stage assuming the best solutions are far from the available solutions. Genetic Algorithm may traps in local optimal as the algorithm continue searching the same area and regenerate same solutions. Sometimes the optimal solutions are far from the already discovered solutions, as shown in Figure 4. So, it is better if the search process explored new areas.

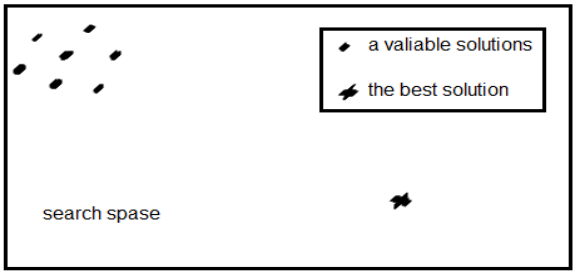

FIGURE 4: Exploration in Genetic Algorithm

In this paper, we have increased the mutation in genetic algorithm in order to get better results, and this increase applied as increase the mutation rate of the genetic algorithm. The research increased the proportion of mutation at a constant rate in every time and registers the results with observations. Furthermore, the research, changes the number of tasks in each time to find out to what extent we need to increase mutations rate in the genetic algorithm to obtain efficient results.

5. EXPERIMENTAL RESULTS

In this section, we'll show the findings through an increase in the mutation on genetic algorithm and process of exploration. To evaluate the proposed mechanism, this research has considered different size of workload traces ranging from lightweight loads containing only 500 jobs to heavy load contains 7000 jobs. Each experiment was repeated several times with different random seeds, and the averages finish times were calculated until the results were saturated. In our experience we have high exploitation by increasing crossover rates in genetic algorithm in each scenario.

TABLE 1.
The effects of increasing the mutation when the number of tasks = 500

Mutation	50	100	150	200	250
Execution Time	2800	2800	2800	2800	2800
Mutation	300	350	400	450	500
Execution Time	2800	2800	2800	2800	2800

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

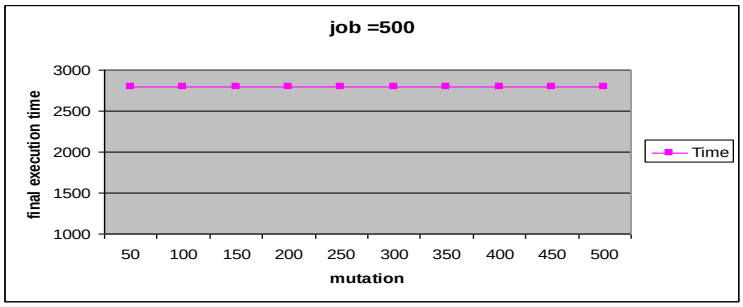

FIGURE 5: The relationship between the mutation and the increase in the time when the number of jobs = 500

As shown in Table 1 and Figure 5 when the mutation rate is increased and the job = 500 the execution time remained constant. This result indicate that the increase in mutation have no effect when the number of jobs is small.

TABLE 2.
The effects of increasing the mutation when the number of tasks = 4000

Mutation	50	100	150	200	250
Execution Time	731400	724650	721400	666954.5	663136.4
Mutation	300	350	400	450	500
Execution Time	583320	581640	579720	569523	539720

a.

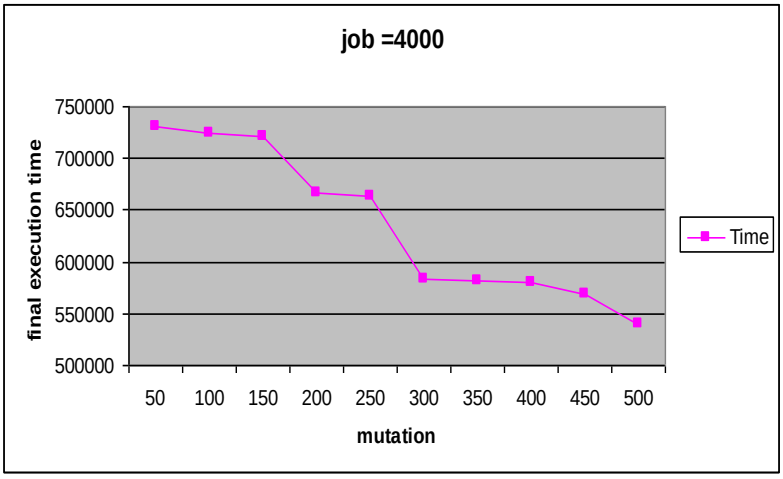

FIGURE 6: The relationship between the mutation and the increase in the time when the number of jobs = 4000

From Table 2 and Figure 6 we can observe that when we increased the number of mutation and the job = 4000 the final execution time of algorithm decreases

whenever we increased the number of mutation. We can state that the increase in mutation affected the final execution time.

TABLE 3.
The effects of increasing the mutation when the number of tasks = 7000

Mutation	50	100	150	200	250
Execution Time	936801	936000	923100	815166	810166.7
Mutation	300	350	400	450	500
Execution Time	771900	752400	747550	743600	741550

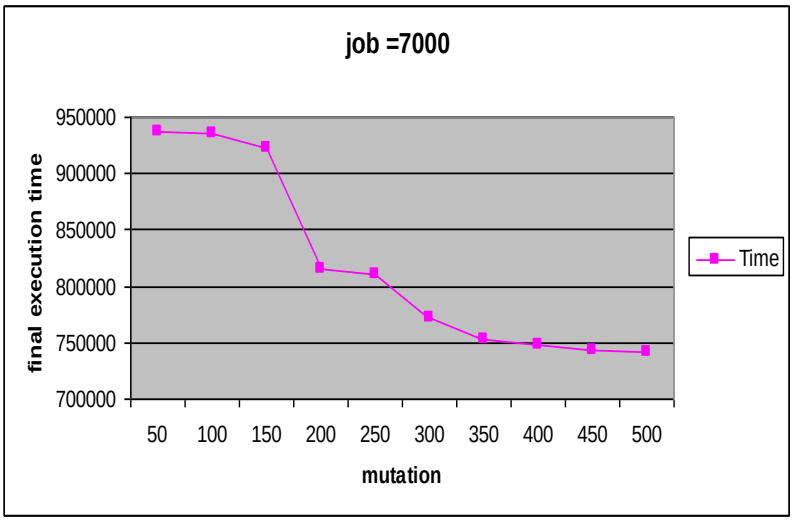

Fig 7: The relationship between the mutation and the increase in the time when the number of jobs = 7000

As shown in Figure 7 and Table 3 when we increased the number of mutation and the job = 7000 the final execution time of algorithm decreases sharply whenever we increased the number of mutation. We can state that, the increase in mutation affected the final execution time significantly.

6. CONCLUSIONS

This research aims to obtain better scheduling results by modifying the implementations of genetic algorithm. The study proposed increasing the mutation of genetic algorithm higher than the typical rates. The study recorded the results with a different number of jobs in each time and the observations that occur in final execution time in each time while increasing mutation. The research found that genetic algorithm get best results with the best time when increasing the mutation and these result directly proportional with the increase in the number of jobs. The mutation and exploration process has a good effect on the final execution time when we have large number of jobs. However, in small number of jobs mutation has no effects.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

REFERENCES

[1] C. K. Ian Foster , Steven Tuecke •, "The Anatomy of the Grid Enabling Scalable Virtual Organizations," {foster, tuecke}@mcs.anl.gov, carl@isi.edu.

[2] R. B. a. B. N. Ajith Abraham, "Nature's Heuristics for Scheduling Jobs on Computational Grids."

[3] H. JIN, X. S. † a) , †, W. Q. , †, a. D. Z. , †, and N. , "DRIC: Dependable Grid Computing Framework," IEICE TRANS. INF. & SYST.,, vol. .E89–D, FEBRUARY 2006.

[4] M. P. M. Dipti Sharma "Job Scheduling Algorithm for Computational Grid in Grid Computing Environment," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, May 2013.

[5] F. D. a. S. G. Ak, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems " January 2006

[6] H. S. Maryam Rabiee "Job Scheduling in Grid Computing with Cuckoo Optimization Algorithm " International Journal of Computer Applications vol. 62, January 2013

[7] F. D. a. S. G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems " 2006.

[8] A. Yousif, S. M. Nor, A. H. Abdullah, and M. B. Bashir, "A Discrete Firefly Algorithm for Scheduling Jobs on Computational Grid," in Cuckoo Search and Firefly Algorithm, ed: Springer, 2014, pp. 271-290.

[9] A. Yousif, A. H. Abdullah, S. M. Nor, and A. Abdelaziz, "Intelligent Task Scheduling for Computational Grid," presented at the 1st Taibah University International Conference on Computing and Information Technology, 2012. P. Brucker, Scheduling algorithms: Springer Verlag, 2007.

[10] A. Abraham, R. Buyya, and B. Nath, "Nature's heuristics for scheduling jobs on computational grids," in The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), 2000, pp. 45–52.

[11] S. Li, Y. Li, Y. Liu, and Y. Xu, "A GA-based NN approach for makespan estimation," Applied Mathematics and Computation, vol. 185, pp. 1003-1014, 2007.

[12] A. Abraham, R. Buyya, and B. Nath, "Nature's heuristics for scheduling jobs on computational grids," 2000, pp. 45-52.

[13] R. Entezari-Maleki and A. Movaghar, "A genetic algorithm to increase the throughput of the computational grids," International Journal of Grid and Distributed Computing, vol. 4, 2011.

[14] H. Liu, A. Abraham, and A. E. Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm," Future Generation Computer Systems, vol. 26, pp. 1336-1343, 2010.

[15] M. Dorigo and T. Stützle, Ant colony optimization: the MIT Press, 2004.

[16] A. Abraham, H. Liu, W. Zhang, and T. G. Chang, "Scheduling jobs on computational grids using fuzzy particle swarm algorithm," 2006, pp. 500-507.

[17] V. Di Martino and M. Mililotti, "Scheduling in a grid computing environment using genetic algorithms," 2002, p. 297.

[18] S. Sanyal, A. Jain, S. K. Das, and R. Biswas, "A hierarchical and distributed approach for mapping large applications to heterogeneous grids using genetic algorithms," in Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on, 2003, pp. 496-499.

[19] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of job-scheduling strategies for grid computing," Grid Computing—GRID 2000, pp. 191-202, 2000.

[20] Y. Gao, H. Rong, and J. Z. Huang, "Adaptive grid job scheduling with genetic algorithms," Future Generation Computer Systems, vol. 21, pp. 151-161, 2005.

[21] S. Lorpunmanee, M. Sap, M. Noor, and A. H. Abdullah, "Fuzzy C-Mean And Genetic Algorithms Based Scheduling For Independent Jobs In Computational Grid," Jurnal Teknologi Maklumat, vol. 18, pp. 1-13, 2006.

[22] G. Singh, 1, a. M. S. Manna, and 2, "Genetic Algorithms: An Unbiased Optimization Technique for Image Segmentation."

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)