International Journal of Grid and Distributed Computing
Vol. 9, No. 3 (2016), pp.221-228
http://dx.doi.org/10.14257/ijgdc.2016.9.3.23

High Exploitation Genetic Algorithm for Job Scheduling on Grid Computing

Walaa AbdElrouf[1] , Adil Yousif[1] and Mohammed Bakri Bashir[2]
University Science & Technology-Sudan[1], Shendi University-Sudan[2]

Abstract

Scheduling jobs on computational grids is identified as NP-hard problem due to the heterogeneity of resources; the resources belong to different administrative domains and apply different management policies. Genetic algorithm which is a metaheuristic search on the basis of the idea of the natural evolution of living organisms generate solutions in order to reach the best solution, using techniques inspired by nature, such as the selection, crossover and mutation. One of the most important processes in the genetic algorithm is the crossover process that combines two chromosomes (parents) to produce a new chromosome (offspring). The parents with the highest fitness functions are selected to participate in the process. The idea behind crossover is that the new chromosome will be better than both parents because it takes the best qualities of both of them. This paper proposed a new job scheduling mechanism based on increasing the crossover rate in genetic algorithm in order to reach the best solution faster to improve the functionality of the genetic algorithm. To evaluate the proposed mechanism this study conducted a simulation using GridSim simulator and different workloads. The results of the simulation process revealed that the increase in the exploitation process decrease the finish time.

Keywords: *Grid Computing, Job Scheduling, Genetic, Crossover, Exploitation*

1. Introduction

Computational grid is a large scale distributed system consists of a huge number of heterogeneous resources belong to different virtual organizations. Scheduling jobs on such environments represents a great challenge and identified as NP-hard problem [1, 2]. Therefore, heuristics and metaheuristics mechanisms have been applied to handle the job scheduling problem on computational grid[3]. However, these mechanisms are commonly able to find good but not necessarily efficient and optimal solutions for the job scheduling problem. Nature-inspired metaheuristics has demonstrated an excellent degree of effectiveness and efficiency for handling combinatorial optimization problems [4]. The remarkable rise in the size of the solution search space motivated researchers to employ nature-inspired metaheuristics mechanisms to solve computational grid scheduling problems.

Grid scheduling involves three main phases: resource discovery, which generates a list of potential resources; information gathering about those resources and selection of a best set; and job execution, which includes file staging and cleanup[5]. Scheduling schemes are centralized, hierarchical and decentralized scheduling schemes[6].Genetic Algorithm is an important mechanism used for job scheduling problem on grid computing. The genetic exploitation is essential process in genetic algorithm as it helps in merging existing solution to enhance the optimization process. Crossover employs exploitation of existing solutions and searching for good solutions in hand. This paper aims to increase exploitation process on genetic algorithm. Because, sometimes the early populations are so near from the best solution than the population created in the late iterations of the genetic process.

This paper contains five sections. Section two describe the related works, section three illustrates the proposed scheduling algorithm, section four simulation and results. We concluded in section five.

2. Related Works

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 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 [7]. 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"[8]. 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[9]. 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[10]. 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[11, 12].

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 [1, 13-15].

3. Genetic Algorithm (GA)

Genetic algorithm is a metaheuristics search method [16]. GA is used to find optimal solutions from among many solutions exists based on the genetic process of organisms[17]. Genetic algorithm is used to explore the search space and the exploitation of existing solutions in order to reach the best solution [18]. Search space contains all possible solutions for the problem under the study. GA generates optimal solutions using techniques inspired by nature, such as crossover and mutation. The genetic algorithm idea came from Darwin's theory of evolution and survival of the fittest [17]. The basic steps involved in Genetic Algorithm are:

- Create initial population.
- Evaluate the fitness of each individual of the initial population.
- Selection: the process of choosing two chromosomes from the population
- Crossover: mates two chromosomes to produce a new offspring.
- Mutation: Take a single chromosome and a change in the genes.
- Termination: Is a process to stop the work of the genetic algorithm.

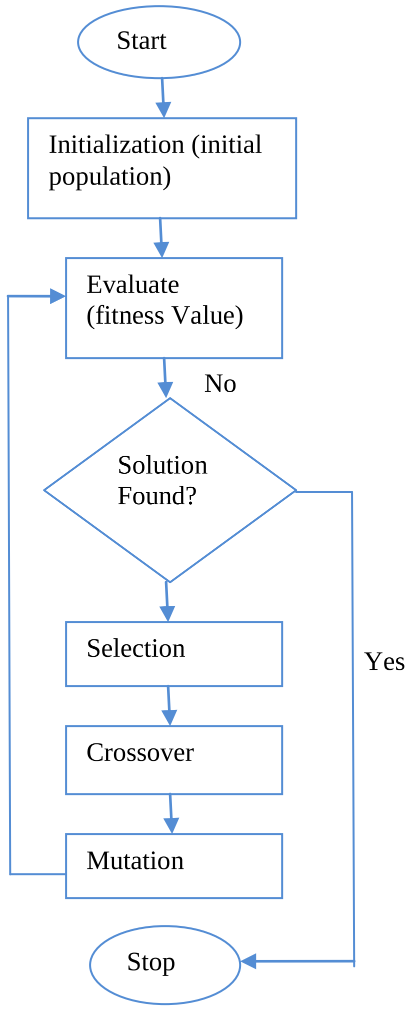

Figure 1. Flowchart of Genetic Algorithm

As shown in Figure 1 genetic algorithm begins by creating the initial population, which is a set of possible solutions to the problem and the creation of the initial population is done randomly. Each individual in the initial population is evaluated using a fitness function [19]. A difficult issue in using GA is often the attempt to find a suitable fitness function which calculates the fitness value, and expresses the problem as well as possible. Each point in the search space must be represented by a valid fitness value[20]. The selection is the choice of a subset of the chromosomes of the initial population according to their fitness and then two of them is chosen to produce the new generation and the selection process is first applied to the initial population. The selected Chromosomes are called parent. Crossover process applied to the parent and through which the exchange of genes parents to produce new individuals [10]. Individuals that are produced from the crossover process will be then sent to the mutation process take a single chromosome and change the genes

[9]. Then all the new individuals that have been produced by the processes of crossover and mutation are evaluated by fitness and individuals who have high fitness function of the population is transferred to the new population. This process continues until we reach the best solution. Individuals with high fitness have the greatest opportunity to be transferred to the new population. Near the best solution solutions are with fitness high unlike the solutions that are far from the best solution.

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[10].

Abraham *et al* in [21] 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 [22] 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[23]. In [24] a suboptimal optimization mechanism for scheduling job on computational grid based on genetic algorithm is 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 [25]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 is developed in [26]. To reduce the repetitions of the generations in GA, Delavar *et al* in [27] 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 [28]. Furthermore, they utilized MCT schedule to initialize the algorithm.

4. The Proposed High Exploitation Genetic Algorithm

Exploitation and exploration are the two main techniques that work to improve the evolutionary algorithms. As shown in Figure 2 exploitation process means the use of the already found solutions to discover new solutions. Exploitation is based on crossover in which the parents are merged to find a better offspring.

Individuals with high fitness are selected for reproduction process while individuals have low fitness are ignored and not selected for reproduction process. The crossover process is often seen as a process of exploitative. Exploitation means the local search is intensifying searched in the vicinity of the best solution and focuses on individuals with high fitness within each region and thus leads to better solution. The main goal of the exploitation and exploration is to control the selection pressure. The main idea of the proposed mechanism is based on better use of existing solutions originally rather than creating new solutions to reach the best solution. This takes advantage of the time spent in the search for the best solution, rather than wasting time in the creation of new solutions so as to obtain the best solution in the shortest possible time. The increase in the exploitation by increasing crossover in a genetic algorithm in order to reach the best solution to this problem as soon as possible so as to improve the functioning of the genetic algorithm.

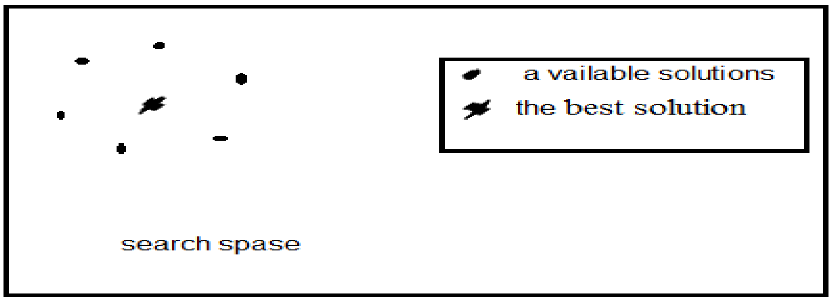

Figure 2. Exploitation in Genetic Algorithm

5. Experimental Results

To evaluate the proposed mechanism, this research has considered different size of workload traces ranging from lightweight loads containing only 1000 jobs to heavy load contains 5000 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. Increasing Crossover when Number of Jobs = 1000

Crossover	50	100	150	200	250
Time	4805	4800	4480	4236	4222
Crossover	350	400	450	500	550
Time	4095	3848	3848	3848	3848

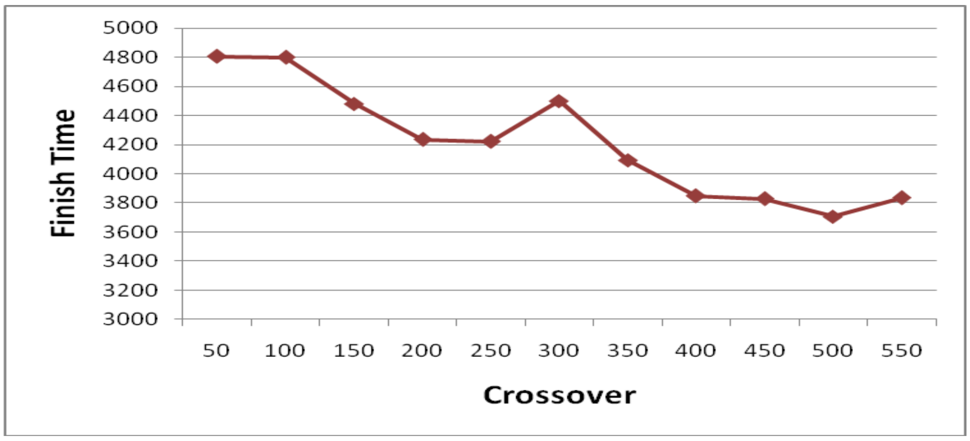

Figure 3. Increasing Crossover when Number of Job = 1000

As shown in Table 1 and Figure 3 that, increasing the exploitation process through increasing crossover rate enhanced the scheduler and decreased the finish time.

When crossover was 100 genes the finish time was 4800 and when the crossover increased to 250 the finish time decreased to 4222. Whenever crossover increased we obtain less finish time. Except for an increase in 300 the exploitation lead to an increase in finish time, this may happens because of random selection of the initial population.

Table 2. Increasing Crossover when Number of Jobs = 5000

Crossover	50	100	150	200	250	300
Time	742800	687194	72695	666543	666954	667090
Crossover	350	400	450	500	550	
Time	683590	683590	683590	683590	597840	

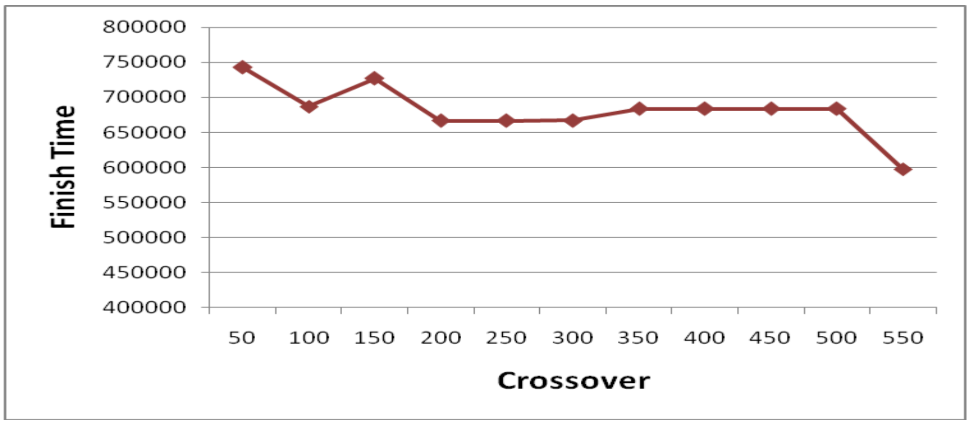

Figure 4. Increasing Crossover when Number of Job = 5000

As shown in Table 2 and Figure 4 that increasing the exploitation process through increasing the crossover rate enhance the scheduler and decrease the finish time. When crossover was 150 genes the finish time was 726950 and when the crossover increased to 300 the finish time decrease to 667090.

Whenever crossover increased we obtain a less time. The best increase of crossover is 550 which get the minimum possible time to reach the best solution. Less than 550 from 350 to 500 when we increase the crossover rate , the finish time is constant. However, from 50 to 200 when we increase the crossover rate the finish time is decreased. Except 250 and 300, when we increase the crossover time increased.

6. Conclusion

One of the most important issues in grid computing is scheduling which is an allocation of jobs on the available network resources in order to accomplish the tasks in the shortest possible time. It is one of the most difficult tasks in the grid computing. This paper proposed a new job scheduling mechanism for grid computing based on increasing exploitation (crossover) in the genetic algorithm in order to reach the best solution in the shortest possible time. This research developed a genetic algorithm with high exploitation for job scheduling on grid computing. The increase exploitation leads to reduce the finish time and thus lead to improved genetic algorithm work. In some cases increasing exploitation leads to an increase in time rather than a decrease. This is because the initial population is randomly generated. Sometimes the random generation of the initial population creates a population that is so for from the best solution. These are the cases where increasing the exploitation rate fails to enhance the efficiency of the grid. Through the exploitation we can explore new areas in the search space that have not explored before. If genetic algorithm runs with minor exploitation process, some areas wil never be visited.

References

[1] 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.

[2] H. Izakian, B. Tork Ladani, K. Zamanifar, and A. Abraham, "A novel particle swarm optimization approach for grid job scheduling," *Information Systems, Technology and Management,* pp. 100-109, 2009.

[3] A. Yousif, S. M. Nor, A. H. Abdualla, and M. B. Bashir, "Job Scheduling Algorithms on Grid Computing: State-of-the Art," *International Journal of Grid and Distributed Computing,* vol. 8, pp. 125-140, 2015.

[4] H. Zang, S. Zhang, and K. Hapeshi, "A review of nature-inspired algorithms," *Journal of Bionic Engineering,* vol. 7, pp. S232-S237, 2010.

[5] J. M. Schopf and A. N. L. Mathematics and Computer Science Division, *TEN ACTIONS WHEN GRID SCHEDULING The User as a Grid Scheduler*.

[6] R. Buyya, , a. M. Murshed, 1, , , , , S. o. C. S. a. S. Eng., C. C. Monash University, V. Melbourne, Australia , rajkumar@buyya.com, G. S. o. C. a. IT, G. C. Monash University, V. Churchill, Australia , and Manzur.Murshed@infotech.monash.edu.au, "GridSim: A Toolkit for the Modeling and Simulation of Distributed

a. Resource Management and Scheduling for Grid Computing ".

[7] 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.

[8] P. Brucker, *Scheduling algorithms*: Springer Verlag, 2007.

[9] 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.

[10] 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.

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

[12] 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.

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

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

[15] 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.

[16] F. X. Javier Carretero, D. d. L. i. S. Inform`atics, U. P. e. d. Catalunya, C. J. G. S.-. Campus Nord - Ed. Omega, S. 08034 Barcelona, fatos@lsi.upc.edu, A. Abraham, S. o. C. S. IITA Professorship Program, Y. University, S.-. Sudaemoon-ku, Republic of Korea, and ajith.abraham@ieee.org, "GENETIC ALGORITHM BASED SCHEDULERS FOR GRID COMPUTING SYSTEMS," *International Journal of Innovative*

a. Computing, Information and *Control ICIC International c°2005 ISSN 1349-4198*

b. *Volume 3, Number 6, December 2007,* 2007.

[17] C. K. J. M. N. S. T. Ian Foster 1, , A. N. L. 1 Mathematics and Computer Science Division, Argonne, IL 60439 , U. o. C. 2 Department of Computer Science, Chicago, IL 60637 , U. o. S. C. 3 Information Sciences Institute, Marina del Rey, CA 90292 , P. 4 IBM Corporation, NY 12601 , , and f. m. a. g. c. i. e. j. u. i. c. tuecke@mcs.anl.gov, "The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration ".

[18] M. T. a. K. Mathew, S. C. Swinburne University of Technology, Jalan Simpang Tiga, 93350, Kuching, Sarawak, Malaysia , and m. s. e. m. a. kmathew@swinburne.edu.my, "A Genetic Algorithm Analysis towards Optimization solutions " *International Journal of Digital Information and Wireless Communications (IJDIWC) 4(1): 124-142 The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2225-658X),* 2014.

[19] P. G. M. a. S. Kumar, "Genetic Algorithms in Intrusion Detection Systems: A Survey " *International Journal of Innovation and Applied Studies* 2014.

[20] H.-H. Sthamer, "The Automatic Generation of Software Test Data Us*ing Genetic Algorithms," Uni*versity of Glamorgan, November 1995.
[21] S. K. Nayak, S. K. Padhy, and S. P. Panigrahi, "A novel algorithm for dynamic task scheduling," *Future Generation Computer Systems,* 2012.
[22] V. Di Martino and M. Mililotti, "Scheduling in a grid computing environment using genetic algorithms," 2002, p. 297.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] A. G. Delavar, M. Nejadkheirallah, and M. Motalleb, "A new scheduling algorithm for dynamic task and fault tolerant in heterogeneous grid systems using Genetic Algorithm," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, 2010, pp. 408-412.
[28] W. Abdulal, A. Jabas, S. Ramachandram, and O. Al Jadaan, "Rank based genetic scheduler for grid computing systems," in *Computational Intelligence and Communication Networks (CICN), 2010 International Conference on,* 2010, pp. 644-649.