International Journal of Computer Applications (0975 – 8887)
Volume 93 – No 16, May 2014

# Workload Analysis in a Grid Computing Environment: A Genetic Approach

P.K.Yadav
CSIR- Central Building Research Institute, Roorkee

Anuradha Aggarwal
Department of Mathematics and Statistics, Faculty of Science, Gurukul Kangri University, Haridwar (UK), India

M.P.Singh
Department of Mathematics and Statistics, Faculty of Science, Gurukul Kangri University, Haridwar (UK), India

## ABSTRACT
Grid computing is the collection of computer resources from multiple locations to reach a common goal. The grid is a special type of distributed system with non-interactive workloads that involve a large number of files. Partitioning of the application program/ software into a number of small groups of modules among dissimilar processors is an important parameter to determine the efficient utilization of available resources in a grid computing environment. It also enhances the computation speed. The task partitioning and task allocation activities influence the distributed program/ software properties such as IPC. This paper presents a metaheuristic model, that performs static allocation of a set of "m" modules of distributed tasks/program considering the two conflicting objectives i.e. minimizing the makespan time and balanced utilization of a set of "n" available resources of a grid computing. Experimental results using genetic algorithm indicates that the proposed algorithm achieved these two objectives as well as improve the dynamic heuristics presented in literature.

## Keywords
Grid computing, Task Allocation, Makespan, Execution Cost, Inter Task (module) Communication Cost

## 1. INTRODUCTION
A Grid is a dynamic heterogeneous environment agglomerating geographically distributed resources and is defined as a process of taking scheduling decisions concerning resources spread over various administrative domains [1]. Users can contribute to grid resources by submitting computing tasks to grid system. The contribution of resources can be active or inactive within the grid. Hence it is impossible for anyone to assign tasks to computing resources by hand in grids. Therefore grid job scheduling [2, 3, 4] is one of the challenging issues in grid computing. Grid scheduling system selects the resources and allocates the user submitted tasks to appropriate resources in such a way that the user and application requirements are met [5]. With more applications looking for faster performance, makespan and balanced utilization of resources are most important objectives that scheduling algorithms challenge to optimize. Makespan is the resource utilization time between the beginning of the first task and the completion of the last task assigned to that resource.

Several heuristic algorithms [6, 7, 8, 9] have been proposed for grid job scheduling. These algorithms include genetic algorithms [10], particle swarm optimization [11], simulated annealing [12], ant colony optimization [13], tabu search [14], gravitational emulation local search [15] and Firefly algorithm [16]. Combinations of these algorithms known as hybrid heuristics have also been reported in the literature [17, 18].

Genetic algorithm was first proposed in 1975 by John Holland et al. [19] at Michigan University. Genetic algorithms (GAs) have also been adopted for solving the problem and obtained promising results. Hamed [20] proposed GA for task allocation in heterogeneous distributed computing system. In [21], a new evaluation algorithm, GLOA is used to solve the problem of scheduling independent task in a grid computing system. Simulation results show that this algorithm produces shorter makespan.

In the previous paper [22], A GA based task allocation problem has been discussed for multiple task allocation without considering the task's size. This paper considers task allocation problem for a single task having some modules of different sizes with the goal of minimizing the makespan time. The experimental results reveal that the proposed algorithm produces better module allocation than other algorithms.

## 2. NOTATIONS

| | |
|---|---|
| m | Total number of modules |
| n | Total number of processors |
| $e_{ij}$ | Execution cost of $i^{th}$ module on processor j |
| $c_{ik}$ | Inter module communication cost of communicating modules i and k |
| $er_j$ | Execution rate of processor j |
| PER | Processor's Execution Rate Vector |
| MS | Module Size vector |
| IMCCM (,) | Inter Module Communication Cost Matrix |
| ECM (,) | Execution Cost Matrix |
| TS | Task Size |
| PS | Population Size |
| MaxIter | Maximum number of Iterations |
| $s_i$ | Size of module i |

| | |
|---|---|
| $p_m$ | Mutation Probability |
| $p_c$ | Crossover probability |

## 3. PROBLEM DESCRIPTION

This paper considers a grid with an arriving task to GA for scheduling. The task's size has been partitioned randomly into the small modules of different sizes. The proposed algorithm should be efficient in finding a solution that produces the minimum makespan. Thus, the problem is to assign these modules to the processors with the minimum makespan.

Let m be the total number of modules to be scheduled and $s_i$ , where i= 1,2,.......,m, be the size of each module in number of bytes and it denoted by MS = [$s_1$,..., $s_m$], a module size vector. $c_{ik}$ is the inter module communication cost of communicating modules due to the exchange of data units between them, during the process of execution. This communication cost is given in the form of Inter Module Communication Cost Matrix, IMCCM () = $[c_{ik}]_{m*m}$. Let n be the total number of processors and $er_j$, where j= 1, 2, ....., n, is the execution rate of each processor in number of bytes per unit time and it is denoted by PER = [ $er_1$,....., $er_n$], processor's execution rate vector.

The execution cost of each module depends on the resource, to which it is assigned and the work to be performed by each of modules of that resource. The execution cost of each module on all the processors is given in the form of Execution Cost Matrix, ECM () = $[e_{ij}]_{m*n}$, where $e_{ij}$ is the execution cost of module i on processor j. All $e_{ij}$ can be computed as the ratio of the coordinates of MS and PER vectors that is to say:

$$e_{ij} = \frac{s_i}{er_j} \tag{1}$$

The execution and communication costs of the processor j can be computed as below:

$$EXE_j = \sum_{i=1}^{m} \sum_{j=1}^{n} e_{ij} \cdot X_{ij} \tag{2}$$

$$COMM_j = \sum_{i=1}^{m} \sum_{\substack{k=1 \\ k \neq i}}^{m} \sum_{\substack{l=1 \\ l \neq j}}^{n} c_{ik} \cdot X_{ij} \cdot X_{kl} \tag{3}$$

where

$$X_{ij} = \begin{cases} 1, \text{if } i^{th} \text{ module is assigned to the} \\ j^{th} \text{ processor} \\ 0, \text{otherwise} \end{cases}$$

$$X_{kl} = \begin{cases} 1, \text{if } k^{th} \text{ module is assigned to the} \\ l^{th} \text{ processor} \\ 0, \text{otherwise} \end{cases}$$

The total cost of the processor j is the sum of execution and communication costs and can be computed as below:

$$T\cos t_j = EXE_j + COMM_j \tag{4}$$

$$\text{Makespan} = \text{Max}\,\{ T\cos t_j \} \tag{5}$$

### 3.1 Constraints

In order to determine the proper allocation, initially the average load of each processor must be determined.

If the execution cost of module i on processor j is $e_{ij}$, the average load on each processor is as shown in equation (6)

$$L_{avg}(j) = \frac{\sum_{i=1}^{m} e_{ij}}{n} \tag{6}$$

The second constraint module allocation considers balanced utilization of a set of "n" available processors.
The number of modules to be assigned to a processor is given by

$$\left\lceil \frac{m}{n} \right\rceil$$

### 3.2 Assumptions

- In this problem cost has been taken as time.
- If more than one module is assigned to the same processor then IMCC between them is zero.
- Allocated modules to processors should not > $\lceil m/n \rceil$.

## 4. THE PROPOSED GENETIC ALGORITHM

The representation of chromosomes is necessary for solving the problem using genetic algorithm.

### 4.1 Encoding Method

Natural numbers are used for encoding the chromosomes. The chromosomes lengths are assumed to be module numbers. Every gene in the chromosome represents the processor. Figure 1 gives an allocation of m modules on n processors.

| $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | ... | ... | ... | ... | $m_m$ |
|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | ... | ... | ... | ... | $P_n$ |

**Fig 1: The module allocation in the form of chromosome**

### 4.2 Initial Population

The initial population is created randomly.

### 4.3 Fitness Calculation

Fitness value is the measure based on which one can decide the fitness of solution. In this paper the fitness of solution has been measured in the form of makespan of that solution. The solution with minimum makespan is the fittest solution.

### 4.4 Genetic Operations

Before the mutation and crossover operations apply, the selection phase is first executed. The selection technique for reproduction used in this paper is based on the roulette wheel method.

#### *4.4.1 Crossover Operation*

The proposed algorithm uses a one- point crossover. The crossover operation will perform if the crossover ratio (Pc >= 0.8) is verified. One point is selected randomly. Then combine the genes of the first chromosome from first gene to the cut point and the genes of second chromosome from cut point to the last gene.

### *4.4.2 Mutation Operation*
A point on each chromosome from the previous phase is randomly selected and then changed to a random number between 1 and m. In the proposed approach, the mutation operation will perform if the mutation ratio (Pm<=0.1) is verified.

## 4.5 Terminating Condition
The algorithm terminates when the maximum number of iterations have been reached.

The mapping of the modules to processors takes place according to the following algorithm:

1) Input: Set the parameters m, n, PER, IMCCM (,), TS, PS, MaxIter, $p_m$, $p_c$.
2) Partitioning the task into modules and obtain MS.
3) Calculate ECM (,) by according to the equation (1).
4) Determine $L_{avg}$ (j) on processor j according to the equation (6).
5) Determine the number of modules to be assigned to the processor according to $\left\lceil \frac{m}{n} \right\rceil$ .
6) Generate the initial population of random individuals as shown in fig.1.
7) Calculate the makespan of each individual using the equation (2), (3), (4) and (5).
8) Iter=1
9) While (Iter <= MaxIter)
10) P=1
11) While (P <= PS)
12) Genetic operations
    - Reproduction
    - Apply crossover according to $p_c$ ( $p_c$ >= 0.8)
    - Mutate the individual according to $p_m$ ( $p_m$ <= 0.1)
13) Calculate the makespan of modified individual.
14) P=P+1.
15) End while.
16) Set Iter=Iter+1.
17) End while.
18) Select only those Childs which satisfy the constraint.

## 5. RESULTS AND COMPARISON
The effectiveness of the proposed scheduling method is assessed and evaluated using makespan. Table 1 lists the parameters used in the performance study of proposed algorithm. The experiment was conducted using three processors and a task having nine modules.

**Table 1: System parameters and the corresponding value**

| Parameter | Value |
|---|---|
| TS | 2268 |
| PS | 1000 |
| $P_c$ | 0.8 |
| $P_m$ | 0.1 |
| MaxIter | 200 |

After partitioning the task's size (TS) randomly into nine different parts, the obtained MS is:

$$MS = \begin{array}{c|c|} m_1 & 240 \\ m_2 & 300 \\ m_3 & 190 \\ m_4 & 301 \\ m_5 & 225 \\ m_6 & 255 \\ m_7 & 232 \\ m_8 & 245 \\ m_9 & 280 \end{array}$$

The experiment was run 10 times with different initial random population. Each run had a fixed number of 200 iterations and makespan value was recorded after each run.

The most suitable chromosome is 213321231 i.e. the modules are allocated as given below:

$m_2, m_6, m_9 \rightarrow p_1$

$m_1, m_5, m_7 \rightarrow p_2$

$m_3, m_4, m_8 \rightarrow p_3$

The execution and the communication costs obtained using the proposed algorithm are given in the table below:

**TABLE 2: Results**

| processor | Execution cost | Communication cost | Total cost |
|---|---|---|---|
| P1 | 488.48 | 663 | 1151.48 |
| P2 | 526.93 | 651 | 1177.93 |
| P3 | 499.75 | 680 | 1179.75 |

It is clear from the above table, that the makespan obtained by the genetic algorithm is 1179.75.

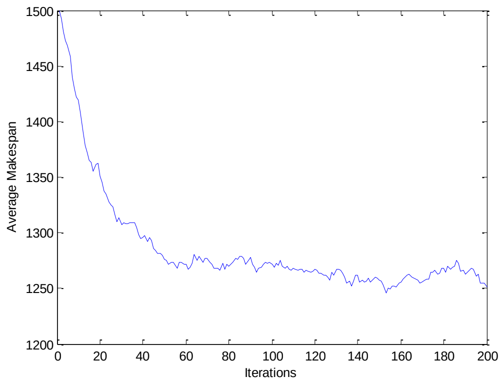


**Fig 2: Average makespan vs. Number of Iterations**

The average makespan of the newly generated chromosomes has been calculated after each iteration and it is observed from the Figure 2 that with the proposed genetic algorithm, optimal makespan is achieved.

In this study, the performance of the proposed algorithm is evaluated in comparison with Yadav et.al. [23]. The comparison of makespan is shown in the table below:

**Table 3: Comparison of makespan time obtained by proposed GA and Yadav et al. [23]**

| | Proposed GA | Yadav et al. [23] |
|---|---|---|
| Module allocation | 213321231 | 223121133 |
| Makespan | 1179.75 | 1268.34 |

# 6. CONCLUSION

In this paper, a genetic algorithm has been presented which not only minimizes the makespan time but also achieves balanced utilization of available resources by executing successfully a task consisting of several modules. The algorithm was studied by conducting more number of runs and the reason was found to be the size of population. As the population size increases, makespan converges to optimal makespan rapidly. Table 3 shows that the results obtained by genetic algorithm are much better than Yadav et.al. [23]. The present model will be validated in future through developing the simulator for real environment.

# 7. REFERENCES

[1] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[2] A. J. S. Santiago, A. J. Yuste, J. E. M. Exposito, S. G. Galan and R. P. D. Prado," A Multi-Criteria Meta-Fuzzy-Scheduler for Independent Tasks in Grid Computing",Computing and Informatics vol. 30:1201-1223 (2011).

[3] K. Vivekanandan and D. Ramyachitra,"A Study on Scheduling in Grid Environment", International Journal on Computer Science and Engineering, vol. 3, No. 2, Feb 2011.

[4] H. M. Lee, J.S. Su and C.H. Chung,"Resource Allocation Analysis Model Based on Grid Environment", International Journal of Innovative Computing, Information and Control, vol. 7, No. 5(A), May 2011.

[5] S. Selvi, D. Manimegalai and A. Suruliandi,"Efficient Job Scheduling on Computational Grid with Differential Evolution Algorithm", International Journal of Computer Theory and Engineering, vol. 3, No. 2, April 2011.

[6]S. Roy and A. Rana,"Comparitive Study of Heuristics Techniques for Resource Allocation in Grid Computing Environment", International Journal of Technology vol.2, No. 2, 2012.

[7] D. Thilagavathi, and A. S. Thanamani,"A Survey on Dynamic Job Scheduling in Grid Environment Based on Heuristic Algorithms", International Journal of Computer Trends and Technology vol. 3, Issue 4, 2012.

[8] S. F. El-Zoghdy, M. Nofal, M. A. Shohla and A. El-sawy,"An Efficient Algorithm Resource Allocation in Parallel and Distributed Computing Systems", International Journal of Advanced Computer Science and Applications, vol. 4, No. 2, 2013.

[9] S. Mandloi and H. Gupta," A Review of Resource Allocation and Task Scheduling for Computational Grids based on Meta-Heuristic Function", International Journal of Research in Computer and Communication Technology, vol. 2, Issue 3, March-2013.

[10] J. Kolodziej and S. U. Khan," Multi-Level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment", Information Sciences: An International Journal, vol. 214, pp. 1-19, Dec. 2012.

[11] P.Y. Yin, S.S. Yu, P.P. Wang and Y.T. Wang," Multi-objective task allocation in distributed computing systems by hybrid particle swarm optimization", Applied Mathematics and Computation vol. 184, page no. 407-420.

[12] S. Y. Rashida and H. Navidi, " A Bargaining based scheduling for resources advanced reservation using simulated annealing into grid system", Internation Journal of Computer Science, vol. 6, no. 9, issue 6, No. 1, Nov. 2012.

[13] R.S. Chang, J.S. Chang, and P.S. Lin (2009),"An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems, 25, 1, pp. 20-27.

[14] C. Fayad, J. M. Garibaldi and D. Ouelhadj," Fuzzy Grid Scheduling Using Tabu Search", IEEE, 2007

[15] Z. Pooranian, M. Shojafar, R. Tavoli, M. Singhal and A. Abraham," A Hybrid Metaheuristic Algorithm for Job Scheduling on Computational Grids", Informatica, vol.37, no.2, 2013 June, p.157(8)

[16] A. Yousif, A. H. Abdullah, S.M. Nor and A.A. Arbdelaziz," Scheduling Jobs on Grid Computing Using Firefly Algorithm", Journal of Theoretical and Applied Information Technology, vol. 33, No. 2, November 2011.

[17] Radha and V. Sumathy," A Hybrid Genetic Algorithm with Elitist Ant System in Grid Scheduling", Life Science Journal, 2013; 10(7s): 510-515.

[18] W. Abdulal, A. Jabas, S. Ramachandram and Omar Al Jadaan," Task Scheduling in Grid Environment Using Simulated Annealing and Genetic Algorithm", in book Grid Computing-Technology and Applications, Widespread Coverage and New Horizons edited by Soha Maad, ISBN 978-953-51-0604-3, Published: May 16, 2012 , chapter 5.

[19] J. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, Ann Arbor, ISBN: 0-262-58111-6, 1975.

[20] Y. Hamed,"Task Allocation for Maximizing Reliability of Distributed Computing Systems Using Genetic Algorithms", International Journal of Computer Networks and Wireless Communications vol. 2, No. 5, 2012.

[21] Z. Pooranian, M. Shojafar, J. H. Abawajy, and M. Singhal,"GLOA: A New Job Scheduling Algorithm for Grid Computing", International Journal of Artificial Intelligence and Interactive Multimedia, Vol.2, No. 1, pp. 59-64, 2013.

[22] M. P. Singh, P. K. Yadav and A. Aggarwal,"Response time optimization of a grid computing system using genetic approach", in conference proceeding, Dhanbad, Jharkhand, 2013, pp. 171-179.

[23] .K. Yadav, Preet Pal Singh and P. Pradhan,"A Tasks Allocation Algorithm for Optimum Utilization of Processor's in Heterogeneous Distributing Computing Systems", vol.2, Issue 1, 2013.