SURVEY ON HEURISTICS BASED RESOURCE SCHEDULING IN GRID COMPUTING

Elwyn Betzar
PG Scholar, Department of CSE, Karunya University
Coimbatore, Tamil Nadu, India

Arul Xavier .V.M
Assistant Professor, Department of CSE, Karunya University
Coimbatore, Tamil Nadu, India

Abstract

The most important goal of the Resource Scheduling in Grid Computing is to efficiently map the jobs to resources. Resource Scheduling in Grid computing is a complex task because of the heterogeneous environment. Grid Scheduling is the core of the Grid resource management systems. Many researchers have proposed various techniques to schedule resources in an efficient manner in Grid computing. This paper describes a survey on Heuristics based Resource scheduling schemes in Grid computing. We have classified these schemes into two main categories namely Meta-heuristics and Hyper-heuristics. We made the comparison of each schemes with different constraints.

***Keywords*:** Grid computing; Resource scheduling; Metaheuristics; Hyperheuristics.

1. Introduction

In the network technology, Grid computing has evolved as major source to provide power to large-scale computational demands. The goal of the grid is to utilize all the idle resources over the network. Various science domains can benefit from the use of grids to solve intensive problems of their specifications [Foster *et al*. (2004)]. As the grid grows many problems arises in the resource management, so a proficient techniques are needed to handle such problems [Aron *et al.* (2012)]. Being an heterogeneous and dynamic type of resources, the mapping of jobs to befitting available resources on the grid is a NP complete problem [Braun *et al*. (2001)]. NP complete problems are preferably executed using heuristic methods. Heuristics in Grid mainly falls into two categories, namely Metaheuristics and Hyper heuristic. Resource availability, load balancing, job scheduling, security oriented issues, information management and so on are the major challenges faced in Grid.

In this paper, we discuss about various heuristic approaches that can be easily be used in Grid scheduling**.** There are two main approaches: Meta heuristics and hyper heuristics. Metaheuristics are a problem specific solution method, which requires knowledge and experience about the problem domain. The hyper heuristics are developed to be general optimization methods, which can be implemented in the problem easily to get the optimal solution [Ajith *et al.* (2000)]. They operate on the search space of heuristic instead of candidate solution. Both the schemes give a good optimal approach of scheduling of resources.

The paper is organized as follows: Section 2 presents the resources scheduling in the Grid computing. Section 3 classifies the heuristic schemes in Grid computing and their performance evaluation is examined in section 4, and the paper is concluded in the section 5.

2. Resource Scheduling

In the Grid Management Systems the Resource Scheduling is the core issue that needs to be efficiently managed. The Grid environment being a heterogeneous environment needs to complete the execution of jobs in an appropriate manner, hence efficient algorithms are needed to schedule the resources in an efficient way. These resources maybe dynamic by itself and at any point of time of failure it may enter or leave the system and can be idle. So a scheduling strategy is needed to generate a schedule that seeks to minimize the total execution time of jobs and also adapt to the heterogeneity and the dynamism of the environment. The mapping of resource to a job, involves the following three basic steps, Resource discovery, Resource Selection and Job execution [Gopalan *et al*. (2008)].

(1) Resource Discovery: Discovering the available resources in the system, resources that are not yet allotted.
(2) Resource Selection: Selecting the optimal resource based on the heuristic algorithm that is applied to schedule the jobs that are in the queue that needs to be allotted.
(3) Job Execution: Allocating the selected resource to the task and executing the task.

Grid resource management systems basically imply mapping jobs to the available ingredient resources. This process includes searching multi administrative domains to use the available resources from the Grid infrastructure in order to satisfy the requirements of the user. Grid scheduling is a two-step process. In step one, the required set of resources is identified as per the user's requests and in the second step, the jobs are mapped on to the actual set of resources, thus further ensuring near optimal satisfaction of QoS parameters [Aron *et al.* (2012)]. The scheduling of Grid jobs can be accomplished using no time characteristics, where the resource broker have to make decision based on resource management policies or in the presence of time characteristics derived from the prediction mechanisms. The resource providers provide offers based on their local policies and the grid resource broker is responsible for discovering the available resources, binding of the user applications, deciding of job allocation to the specific resource, initiating the computational adaptive changes in grid resources and presenting the grid to the user as a bind resource.

3. Metaheuristic Scheduling

The scheduling is core of the Resources management system. The Grid scheduling is a NP complete problem. Various metaheuristic methods are used to sole the scheduling problems in Grid.

3.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is one of the latest evolutionary optimization techniques inspired by nature. It is a robust stochastic optimization technique based on the movement and intelligence of swarms. It has the better ability of global searching and has been successfully applied as the concept of social interaction to problem solving. It also has fewer algorithm parameters than both genetic algorithm and simulated algorithm. Furthermore, PSO algorithm works well on most global optimal problems [Liu *et al.* (2010)]. PSO algorithm is employed to solve the scheduling problem in grid and handles cases in which characters are badly deformed.

The basic idea in PSO is that it uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution. Consider a N-dimensional space which contains particle known as points, that adjusts its movement according to its own moving experience as well as the movement experience of other particles.Each particle keeps track of its coordinates in the solution space which are associated with the best fitness value that has been achieved so far by the particle, and that value is called the personal best, *pbest*. Another best value that is achieved by the heuristic is the best value obtained so far by any particle in the neighborhood of that particle and this value is considered to be the *gbest*. The basic concept of PSO is to accelerating each particle toward its *pbest* and the *gbest* locations, with a randomized weight acceleration at each instance of time steps.

The particles try to modify its position using the following information available such as the current velocities, the current positions, the distance between the current position of the heuristic and its pbest, the distance between the current position of the heuristic and its *gbest* [Lei *et al.* (2008)]. The modification of the particle's position can be mathematically modeled according the following equation:

$$V_i^{k+1} = wV_i^k + c_1\, rand_1(\ldots) \times (pbest_i - s_i^k) + c_2\, rand_2(\ldots) \times (gbest - s_i^k) \ldots\ldots + c_n\, rand_n(\ldots) \times (gbest - s_i^k) \quad (1)$$

where, v_i^k : velocity of agent i at iteration k,
w: weighting function,
c_j : weighting factor,

rand : uniformly distributed random number between 0 and 1,

s_i^k : current position of agent i at iteration k,
$pbest_i$: pbest of agent i,

gbest: gbest of the group.

All particles in PSO are kept as members of the population through the course of the run. Thus PSO helps is getting the optimal solutions without any complexity. Unlike the genetic algorithms and other evolutionary strategies, the PSO has no selection operation. PSO is the only algorithm that does not implement the survival of the fittest.

3.2 Genetic Algorithm

A genetic algorithm (GA) is a search technique used in computing to find true or approximate solutions to optimization and search space problems. Genetic algorithms are known to be as global search heuristics [Alionne Ngom *et al.* (2005)]. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology which involves the steps such as inheriting, mutation, selection of the best chromosomes, and recombination. Genetic algorithm is the most popular and successful among the metaheuristic which is being used frequently in the scheduling problems. Genetic Algorithm is a flexible methods that can be used to solve optimization problems [Vincenzo *et al.* (2004)], based on the genetic course of biological organisms. By imitating the process of Genetic algorithm we can generate solutions to real time problems.

A lot work has been done using Genetic algorithm for the grid scheduling where scheduling generated needs to be optimal [Fatos *et al.* (2002)]. Genetic algorithm uses a direct terminology of natural behavior. The environment in which they work is with a population of individuals each representing a possible solution (partial solution) say a population of 1000 chromosomes to a given problem. Each individual (chromosomes) is assigned a fitness value according to how optimal it is solution to the given problem. The individuals with high fitness values are given opportunities to reproduce by cross-breeding with other individuals in the population. This process is known as mutation and cross-over in technical terms. The following is the general structure of the Genetic algorithm [Patnaik *et al.* (1994)].

```
Initial population generation;
Evaluation;
While (stopping criteria still not met)
{
Selection;
Crossover;
Mutation;
Evaluation;
}
Output optimal solution;
```

The mutation process produces new individuals as offspring which inherits some features of its parents. Which is carried out by mating of individuals of higher fitness values, most of the search space is explored in the environment [Andrew *et al.* (2005)]. The population will get together to get an optimal solution of the problem, if the Genetic algorithm is designed well. The reproduction phase in the algorithm increases the performance of the genetic algorithm. In which, the chromosomes of the selected parents are recombined using mechanisms of mutation and crossover. The crossover is the process in which takes a pair of chromosomes is taken and divided at a random position to form two head portions and two tail portions. The tail portions are then swapped to get a new chromosome. Thus the two off springs get some genes from each of its parent and the mutation process that changes the gene with a randomly selected less probability value [Gopalan *et al.* (2008)]. This makes the search for the optimal solution much simpler.

3.3 Genetic Algorithm-Tabu Search

It is the hybrid of the metaheuristics methods of Genetic Algorithm and Tabu Search. Genetic algorithm and Tabu Search (GA-TS) hybrid maintains the tabu list. It is a metaheuristic strategy based on neighborhood search with overcoming local optimality. It directs the selection of neighbours. Unlike Simulated Annealing this hybrid heuristics works in a deterministic way that tries to model human memory processes. The previous visited solutions are recorded in the memory, using simple but effective data structures, the search is based on a memory list of the neighbour's fitness values. Each time a new chromosome is reproduced by mutating a chromosome from the population [Glover *et al.* (1997)], it is thus checked whether it is present in the tabu list. If it is present, it is rejected, regardless of the quality with respect to the current solution. If it is not present, the new solution is added to the tabu list and then checked for quality. If it is higher quality than the current solution, it is accepted. The hybrid assures that previously visited solutions are not visited, thus saves the run time by avoiding solutions that are already checked. Thus it is adaptive and sequential, thus helps in attaining the optimal solution.

3.4 Hyper-heuristics

Hyper heuristics is to build systems that can handle classes of problems rather than solving just one problem. A hyper-heuristic is a heuristic search methodology that is used to efficiently solve search space problems. There might be multiple heuristics from which one can choose for solving a problem, and each heuristic has its own merits and demerits. The purpose is to simultaneously devise algorithms by combining the strength and incompetence of known heuristics in the search space [Ross (2005)]. In a regular hyperheuristic framework design there is a high-level methodology and a set of low-level heuristics [Burke *et al.* (2003)]. Taking a problem instance, the high-level method chooses the low-level heuristic that should be applied at any given time, depending upon the problem state.

Metaheuristics are problem-specific solution approach, in which they require in depth knowledge and experience about the problem domain and properties and require fine-tuning of parameters. Heuristics are expected to gain the computational performance or theoretical simplicity, probably at the cost of accuracy or precision. Since different metaheuristics in the search space environment have different advantages and disadvantages, it makes sense to see whether they can be combined in some way so that each makes up for the weaknesses of another. A simplistic way of doing this would be as shown below:

if (problem type(p) = p1) then

apply (heuristic1, p);

else if (problem type(p) = p2) then

apply(heuristic2,p);

else if…

One logical extreme of the above approach would be an algorithm containing infinite switch statements enumerating all finite problems and applying the best known heuristics for each certainly is not a better approach. A better approach is to combine each heuristic under the problem condition under which each embellishes and hence apply different heuristics to different phases or parts of the solution process. Hyper-heuristic characterizes a set of strategies that are used to choose a heuristic from a set of low level heuristics. It can be described as a supervisor, which manage control of the choice of which local search neighborhood to choose while constructing a solution/schedule process. A local search neighbour, also known as a low-level heuristic, is a decree or a simple method that commonly yields a small change in the scheduling process. These strategy used can be very simple or metaheuristics and the hyper-heuristic can also be a metaheuristics.

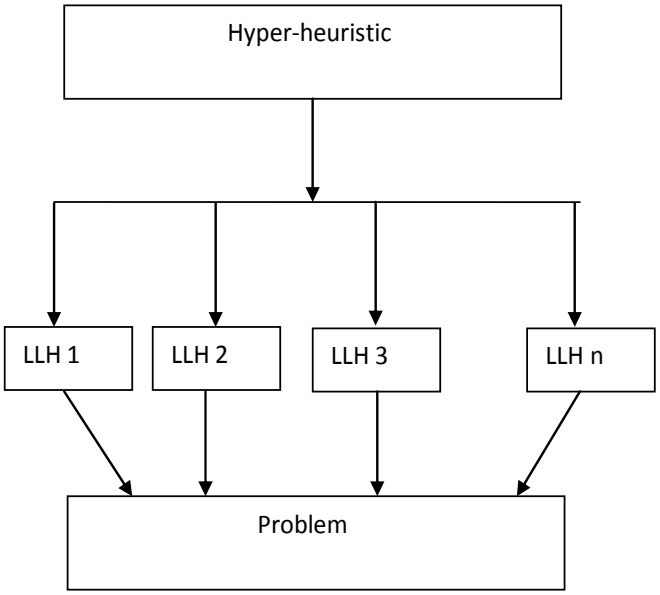

Fig. 1. General Framework of Hyper-heuristic

A single iteration of a hyperheuristic method can be decomposed in two stages, heuristic selection and movement acceptance. The selection methods are simple which randomly select the low level heuristics. Greedy hyperheuristics selects the best performing heuristic at each iterations. Choice function keeps track of previous performance of each heuristic and makes a choice via a choice function. The movement acceptance can be deterministic or nondeterministic. One of the nondeterministic acceptance criteria is the great deluge algorithm. The Hyperheuristic algorithm used is based on the extended Great Deluge algorithm [Gunter (1993)]. The Greedy selection heuristic is used to select the best heuristic. The basic concept of the Extended Great Deluge Hyper-heuristic Algorithm is as shown below.

Step 1: Initialization

1. Initialize population.

2. Set the total iterations N;

3. Evaluate fitness function f(s)

4. Initial level B_0 = f(s)

5. Specify input parameter B

6. Set i = 1;

Step 2: Process i^{th} chromosome

While not stopping condition do

1. Select the candidate solution s^* by applying the selection heuristic

2. Evaluate fitness function for s^*

3. If $f(s^*) \leq B$ then accept s^*; $f(s) = f(s^*)$;

4. $B = B\text{-}\Delta B$;

During the initialization step, the fitness function $f(s_o)$ is set level B. This is slowly decreased by ΔB at each iteration. The pursuance of the algorithm is dependent upon the choice of the ΔB parameter and is dependent on the total number of iterations and initial fitness function.

The vital difference between metaheuristics and hyper-heuristics is that most implementations of metaheuristics search within a search space of problem solutions, whereas hyper-heuristics continues searching within a search space of heuristics. Thus, when using hyper-heuristics, the attempting to find the right method or sequence of heuristics in a given situation rather than trying to solve a problem directly. Moreover, the searching for a generally applicable methodology rather than solving a single problem instance. The hyperheuristic manages the choice of which low-level heuristic method should be applied at any given instance, depending on the characteristics of the region of the solution space currently under exploration. This means the hyperheuristic itself does not search for a better solution to the problem. Instead, it selects at each step of the solution process the most promising simple low-level heuristic, which is potentially able to improve the solution. Thus the Hyperheuristic is a new approach of the high level methodology with a simple process of scheduling.

4. Performance Evaluation

To evaluate the heuristics, the scheduling experiment was performed for several test cases. We have used an average of fifty runs in order to guarantee the statistical correctness. The scheduling experiments performance for several test cases and results obtained are represented graphically. The graph in figure 2 and 3 shows the performance comparison of Particle Swarm Optimization, Genetic algorithm, GA-TS and the Hyper-heuristic for fifty runs of the sample cases.

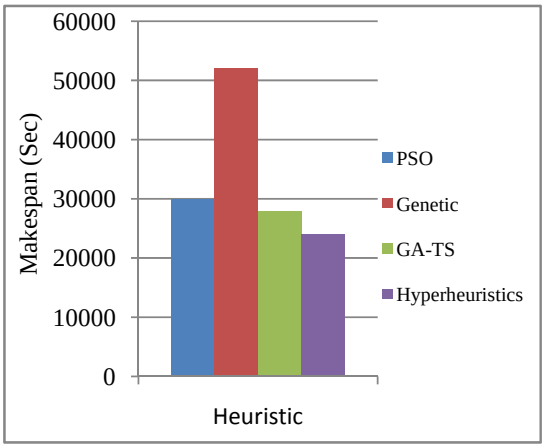

Fig. 2. Makespan comparison of 500 jobs on 20 resources.

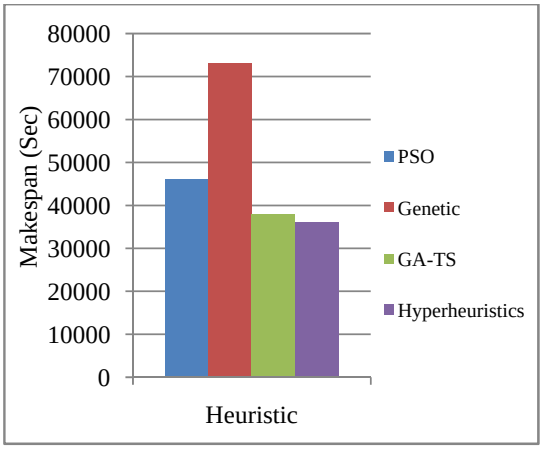

Fig. 3. Makespan comparison of 1000 jobs on 30 resources.

We have presented the graphs by the simulation results of [Ali *et al*. (2000)] simulation model with the Gridsim discrete event simulation so as to test the performance of the heuristics based algorithm. To validate the algorithm, in fig.2. 500 jobs/applications and 20 resources have been considered the results show that the Hyper heuristics outperformance all the other heuristics. The other three heuristics have higher makespan than the hyperheuristics. Considering the other scenario in Fig.3. 1000 jobs/applications and 30 resources is considered and the performance comparison is evaluated and the outcome results such that the hyperheuristic has lesser makespan than the other heuristic approaches, the experiment was executed at an average of 50 times to measure the performance of the heuristics. We have compared the performance of hyperheuristic with well-known scheduling algorithms such as Genetic algorithm, PSO and GA–TS. We have analyzed the performance of the hyperheuristics with variation in both the number of jobs and the number of resources, which are expected to vary in the real Grid environment. We evaluated the performance with respect to makespan. Makespan allows the evaluation of the algorithm which results in better scheduling in the sense of the duration of job execution. The proposed algorithm helps to achieve a high performance and simultaneously it also helps to satisfy the user's requirements. In the experiments conducted, the hyperheuristic algorithm clearly demonstrates its ability to provide better performance with respect to the existing Grid scheduling algorithms.

5. Conclusion

As Grid computing has emerged for solving scientific, engineering and large scale problems, it can be concluded that Grid scheduling is one of the main challenging issues of Grid computing. Meta-heuristic is highly adaptive in the Grid computing environment but it does not provide good solutions for more numbers of jobs in a heterogeneous environment. Considering all these criteria and simulation results, it is found that the hyperheuristic provides a better solution, low resource heterogeneity and near optimal solution for Grid scheduling problems. In this paper, we attempt to use hyper heuristics on top of Meta heuristics built using known heuristics, namely Genetic Algorithm, Particle Swarm Optimization and Hybrid of Genetic and Tabu Search for scheduling jobs in a grid environment. The hyper heuristic that is built on the basis of the hybrid metaheuristics is experimentally shown to give better results than the particular hybrid heuristics in all the test cases. Global optimization algorithms attract considerable computational effort. Hence in a dynamic environment like Grid computing, the main aspect would be to generate schedules in a minimum amount of time, which means that scheduling is to have a minimum makespan. The introduction of the Hyperheuristic in the grid computing has paved the way for an easy and better handling of the scheduling issues in the grid computing. This makes the Grid computing much convenient one.

References

[1] Abdullah .R, Khateeb A.A, Rashid A.N, "Job type approach for deciding job scheduling in Grid computing systems", Journal of Computer Science 5 (10) (2009) 745–750.

[2] Ajith Abraham, Rajkumar Buyya and Baikunth Nath, "Nature's Heuristics for Scheduling jobs on Computational Grids", *International Conference on Advanced Computing and Communications 2000*, 2000.

[3] Ali .S, Siegel .H.J, Maheswaran .M, D. Hensgen, "Representing task and achine heterogeneities for heterogeneous computing systems", Tamkang Journal of Science and Engineering 3(3)(2000)195–207

[4] Alionne Ngom, Mona Aggarwal and Robert D. Kent. "Genetic algorithm based scheduler for Computational Grids", *International Symposium on High performance Computing Systems and Applications(HPCS'05)*, IEEE, 2005.

[5] Andrew J. Page and Thomas J. Naugton, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing",*IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

[6] Aron .R, Chana .I, "Formal QoS policy based Grid resource provisioning framework", Journal of Grid Computing 10 (2) (2012) 249–264.

[7] Braun T.D, Siegel H.J, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing 61 (6) (2001) 810–837.

[8] Burke E.K, Hart .E, Kendall .G, Newall .J, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology", Handbook of Metaheuristics (F. Glover and G. Kochenberger, eds.), Kluwer, 2003, pp. 457–474.

[9] Fatos Xhafa and Javier Carretero. "Use of Genetic Algorithms for Scheduling Jobs in Large scale Grid applications", Okio Technologies IR Ekoonominis Vvstymas, *Technological and Economic development of Economy*, Vol XII, No.1 pp 11-17, 2002.

[10] Foster .I, Kesselman .C, "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, USA, 2004.

[11] Glover .F, Laguna .M (1997). "Tabu Search". *Kluwer Academic Publishers*.

[12] Gopalan N.P and S. Mary Saira Bhanu, "A Hyper-Heuristic Approach for Efficient Resource Scheduling in Grid", Int. J. of Computers, Communications & Control, Vol. III (2008), No. 3, pp. 249-258

[13] Gunter Dueck: "New Optimization Heuristics The Great Deluge Algorithm and the Record-to-Record Travel", Journal of Computational Physics, Volume 104, Issue 1, p. 86-92, 1993

[14] Lei Zhang, Yuehui Chen, Runyuan Sun, Shan Jing and Bo Yang, "A Task Scheduling Algorithm Based on PSO for Grid Computing", International Journal of Computational Intelligence Research. Vol.4, No.1 (2008), pp. 37–43

[15] Liu .H, Abraham .A, Hassanien A.E, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm", Future Generation Computer Systems 26 (8) (2010) 1336–1343.

[16] Patnaik L.P and Srinivas .M, Genetic algorithms: A survey, IEEE Comput. 27, 6 (June 1994),17-26.

[17] Peter Kowling, Graham Kendall and Eric Soubeiga, "Hyper - heuristics : A tool for rapid prototypingin scheduling and optimization", *LNCS 2279, Applications of Evolutionary Computing : Proceedings of EvoCop2002*, Kinsale, Ireland, 2002.

[18] Ross .P, Hyper-heuristics, "Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques" (E. K. Burke and G. Kendall, eds.), Springer, 2005, pp. 529-556.

[19] Vincenzo Di Martino, "SubOptimal Scheduling in a Grid using Genetic Algorithms, *Parallel and nature-inspired computational paradigms and applications"* , Elsevier Science Publishers pp: 553 - 565, 2004.

[20] Vo .S, "Meta-heuristics: the state of the art", in: A. Nareyek (Ed.), Local Search for Planning and Scheduling, in: LNAI, vol. 2148, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 1–23.