CHAPTER 5

OPTIMAL HEURISTIC RULE BASED SCHEDULING ALGORITHM

5.1 INTRODUCTION

Grid computing which is an extension of distributed computing is a highly dynamic topology in which the resources join and leave the grid in an unpredictable manner. The grid is operated in P2P mode where the resources join or leave without any pre-arranged schedule.

The true computational capability is also hard to obtain at any instance of time as these computational resources may go offline regardless of the job state allocated. The Resource Management System (RMS) of the grid has to select the appropriate resource and to provide uninterrupted service. Foster defines the responsibilities of the RMS on the grid with regard to the discovery of available resources to the application, mapping the resources to the application subject to some performance goals and scheduling policies and loading the application to the resource in accordance with the best available schedule.

The task of mapping jobs to the available computing nodes or scheduling of the jobs on the grid is a decision making problem. The schedule strategies can have a significant impact on the performance characteristics. The optimization problem is often solved using some heuristics techniques. The heuristics are intended to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision. The simple

heuristics methods may not find an optimal solution since they perform moves that lead to a final solution for a local optimum.

The proposed heuristics rule are developed to be used in general optimization methods, which can be applied to any optimization problem easily. They are operated on the search space of heuristics instead of candidate solutions. Thus unlike other heuristics, the Authors Heuristic mechanism deploys a set of simple heuristics and uses only non problem-specific data, such as, fitness change or heuristic execution time. The heuristic rule approach is used to solve the various scheduling problems such as travelling salesman problems and timely bounded problems. This algorithm is more adaptive to the grid scenarios where both resources and applications are highly diverse and dynamic in nature.

5.2 RELATED WORK

Various types of research works have been carried out to devise efficient algorithms for scheduling resources optimality in decentralized work environment.

- Jin Xu et al (2011) have formulated the task scheduling problem as a linear programming problem and proposed permutation based representation to schedule workflow jobs on the grid.
- Vazquez et al, (2008) compares eleven replication heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems.
- (Rafael 2004) and Lanier Watkins et al (2011) addresses the issues which a resource broker has to encounter such as

resource discovery and selection, job scheduling, job monitoring and migration.

- (Lingyun Yang et al 2003) has proposed a conservative scheduling policy that uses information about expected future variance in resource capabilities to produce more efficient data mapping decisions.

- Ajith Abraham et al (2000) have hybridized nature's heuristics namely GA, TS and SA and used the hybrid heuristics for job scheduling in the grid.

This chapter proposes an optimization scheduling method which uses three heuristic mechanisms to schedule independent jobs in the computational grid. The heuristic uses a set of parameters such as fitness value and job confirmation for decision making processes in the contexts of the Job heterogeneity, resource heterogeneity and the objective functions makespan, flow time and the resource utilization.

5.3 BASIC REQUIREMENTS IN OPTIMISTIC HEURISTIC RULE

The execution of a job in a dynamic environment like grid often calls for efficient algorithms to schedule the resources required for successful execution of the jobs as proposed by Yulai Yuan et al (2008). These resources may themselves be dynamic and may enter or leave the system at any point of time or fail and can be idle. So a scheduling strategy is required to generate schedules, which seek to minimize the total execution time of jobs and also adapt to the heterogeneity and the dynamism of the environment. Allocation of a resource to a job, involves three basic steps.

- Resource Discovery: Identifying the resources that are currently free (without being allocated to any job) in the system.
- Resource Selection: Choosing one of the free resources based on some underlying algorithm to schedule it to a job on the ready queue.
- Job Execution: Allocating the chosen resource to a job and executing the job.

Some of other important key terminologies used in this context are,

- Confirmed jobs: It defines the ready factor = 1, that is the job has no dependency and the execution token is received from the resource broker.
- Non confirmed jobs: If the ready factor is less than 1, then the jobs are said to be unconfirmed whether the token is generated from the resource broker or not.
- Scheduling Policies: The scheduling of grid jobs can be done using on time characteristics where the resource broker has to make decision based on resource management policies or in the presence of time characteristics derived from the prediction mechanisms.

In this present work, it is considered that the time characteristics are available. The resource owners provide some offers based on their local policies and the grid resource broker is responsible for resource discovery, deciding allocation of a job to a particular resource, binding of user applications, initiating computations to adapt to changes in grid resources and present the grid to the user as a single unified resource.

5.4 OPTIMISTIC HEURISTIC RULE ALGORITHM

The each job which is to be scheduled for processing has a unique id and an associated processing resource requirement like the number of processing cycles required. The jobs are indivisible and independent of all other jobs. The arrivals of jobs are random and are placed in a queue of unscheduled jobs. The available processing resources vary over time and an exponential smoothing function A(i) is repeated sequentially, where v is the control parameter chosen between 0 and 1 and the sequential arrival of the process is a(i).Then the exponential smoothing is given by:

A (i) = (1-v)A(i-1) + va(i-1)

The smoothing is done by allowing the recent values to exert more influence than the older values. The batches of jobs from the queue are scheduled on processors during each invocation of scheduler. At any instance, when the number of jobs that have to be scheduled are greater than the number of resources available, then a suitable mapping of jobs to resources have to be devised. The following allocation mechanism is adopted, in such a case. If R resources and J jobs are available at a instant (J>R), then a balanced randomized initial population is generated using the Most-Into-Least list scheduling heuristics which has been successfully used in Heuristic Rule schedulers.

A random number of tasks are assigned to resources on a priority basis. The remaining tasks are then sorted out and the job requiring the maximum number of cycles (Longest Job) to complete is chosen and is allocated to the resource with the greatest speed (Fastest Resource). The fastest resource is chosen from the list of resources that become available first. This scheduling is implemented with three rules,

1. Heuristic Local Search.
2. Heuristic Simulated Annealing

3. Heuristic Tabu Search

5.4.1 Heuristic Local Search

The Heuristic Local Search (HLS) performs mutations on the entire job queue, accepting mutated solutions only if they have a better fitness value than the current queue that has been mutated as shown in Figure 5.1. This process is repeated for a random number of times, or until a termination condition is met, creating a generation per execution of the process. The population in each generation is sorted out according to fitness function and this generation is used as input for the next.

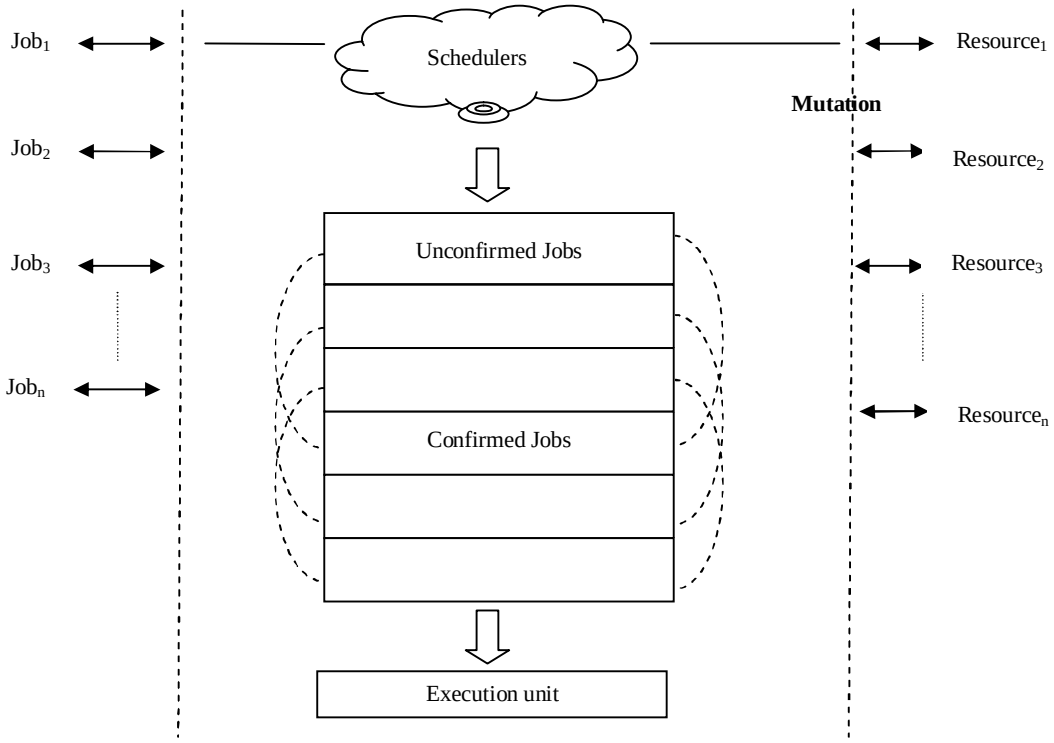

Figure 5.1 Heuristic Local Search Framework

5.4.2 Heuristic Simulated Annealing

In Heuristic Simulated Annealing (HSA), for every mutation of the given jobs in the input queue, a parameter 'density' is decreased by a random

value as shown in Figure 5.2. If the mutated solution is better than the current solution, then it is accepted with a probability 1. If the mutated solution is inferior to the current solution, it is accepted with a probability of $e^{-Cost/density}$. When the density becomes 0(threshold), it is re-initialized to 1. The acceptance of solutions creates a new input queue. The job in each generation is sorted according to fitness function and this generation is used as input for the next scheduler.

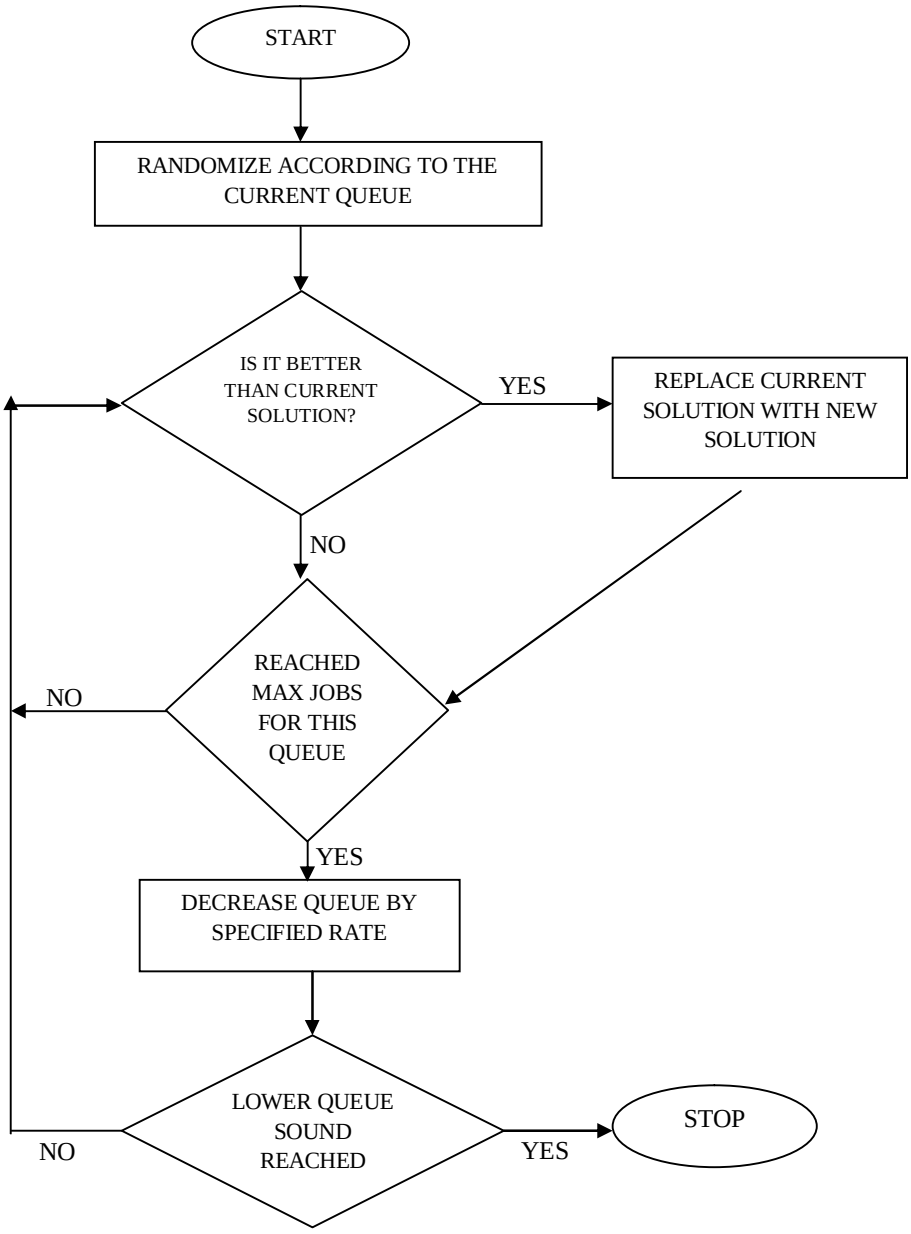

Figure 5.2 Heuristic Simulated Annealing Formulation

5.4.3 Heuristic Tabu Search

The Heuristic Tabu Search (HTS) performs mutations on the entire job queue, accepting mutated solutions only if they have a better fitness value than the current queue that has been mutated. This process is repeated for a random number of times, or until a termination condition is met (Figure 5.3), creating a generation per execution of the process. If the unconfirmed job repeatly presents then HTS hybrid maintains a Tabu list. The unconfirmed jobs will be sent to the tabo list when the queue is full. The tabo list mutates with respect to delay timer. If delay of the unconfirmed job continues it is rejected else it redirects to the scheduling queue when it is free.

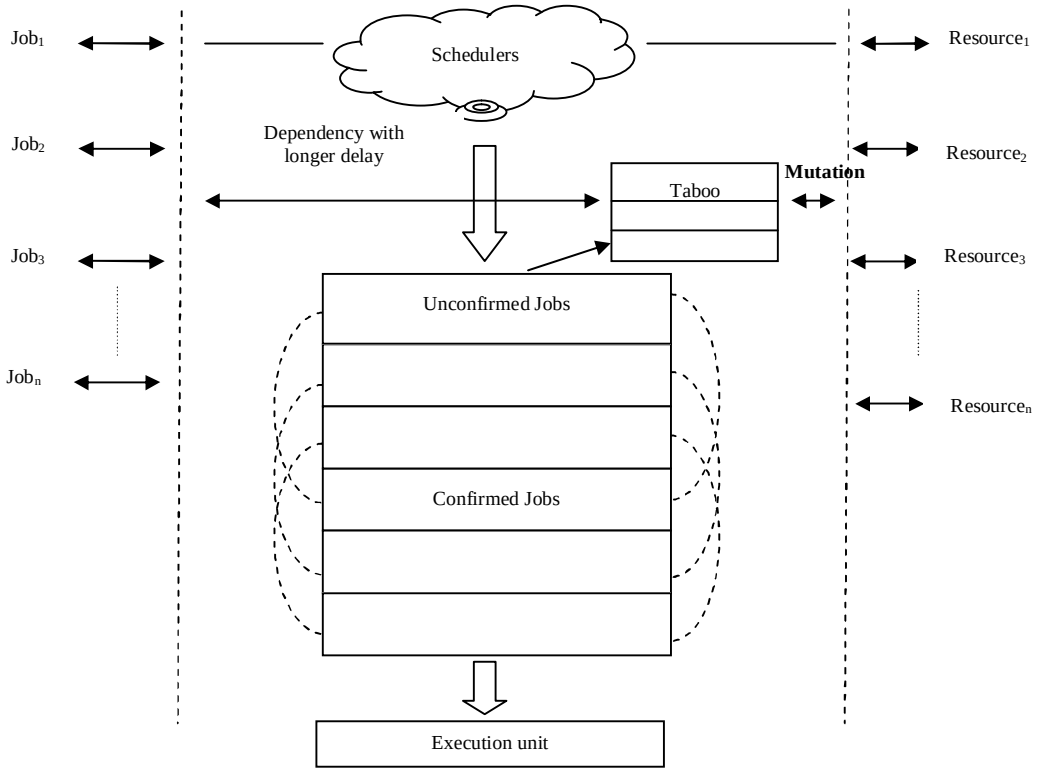

Figure 5.3 Heuristic Tabu Search Frame Work

5.5 OPTIMISTIC HEURISTIC RULE IMPLEMENTATION

Once the job dependency occurs, providers must take some measures to minimize the loss. What a provider can do is to arrange the execution order of jobs in its job queue. It is called Optimistic Heuristic Rule (OHR) scheduling. The relative order of the jobs in the origin queue is unchanged. Every time a provider is offered a job that is not kept in the job queue, it starts the Heuristic Rule scheduling algorithm.

1. Identify resources and their properties and then select resources capable of executing user jobs.

2. Select the resources that fit user requirements using scheduling Heuristic algorithm and map jobs to them.

3. The jobs are separated as confirmed and unconfirmed jobs. The mutation is performed to choose the required resource in the HLS algorithm

4. The job nature is determined whether the jobs can be completed through current solution or randomized solution.

5. If the unconfirmed jobs repeatedly persist for long time in the scheduling queue then the unconfirmed jobs is sent to the tabu list.

6. Deploy jobs on execution unit to ensure confirmed jobs, jobs pressure and independent jobs. This algorithm is proposed by combining the HLS, HSA and HTS.

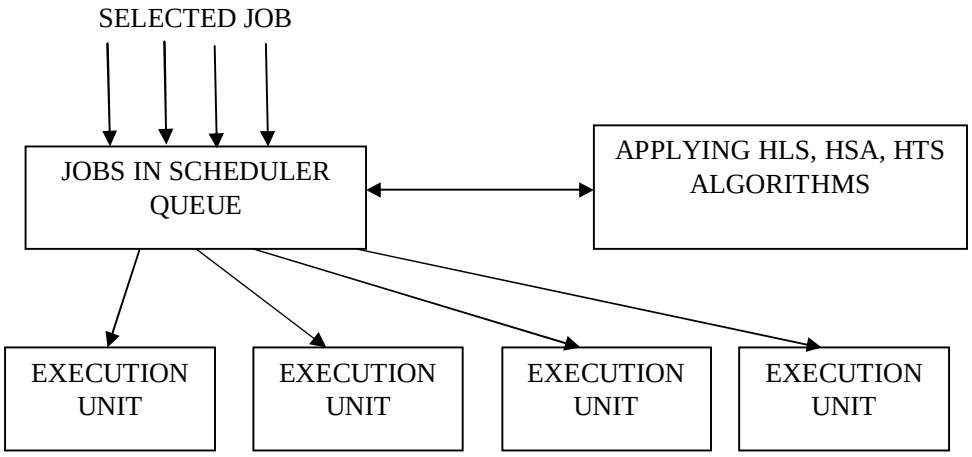

Figure 5.4 General Framework of Heuristic Rule Approach

The frame work represents the sequence of jobs and resources. This method of allocation is represented by a 'scheduler' as shown in Figure 5.4. Each 'sequence' represents a mini-allocation table, allocating R jobs to R resources. The Expected Time to Compute (ETC) matrix is formed from the predictable time characteristics of the jobs and stored as the ETC matrix of size J×R. The fitness function is based on the makespan of the schedule. It is calculated as min-max {Ci , i = 1,...n},where Ci is the finishing time of the latest job.

5.6 EXPERIMENTAL RESULTS

The scheduling experiment has been performed for several test cases and the results obtained are graphically represented. For each test case, the experiment has been executed at an average of 50 times to measure the performance of the heuristics. The graphs in Figures 5.5 and 5.6 show the comparison of heuristic and heuristic rule for 50 runs of the sample test cases.

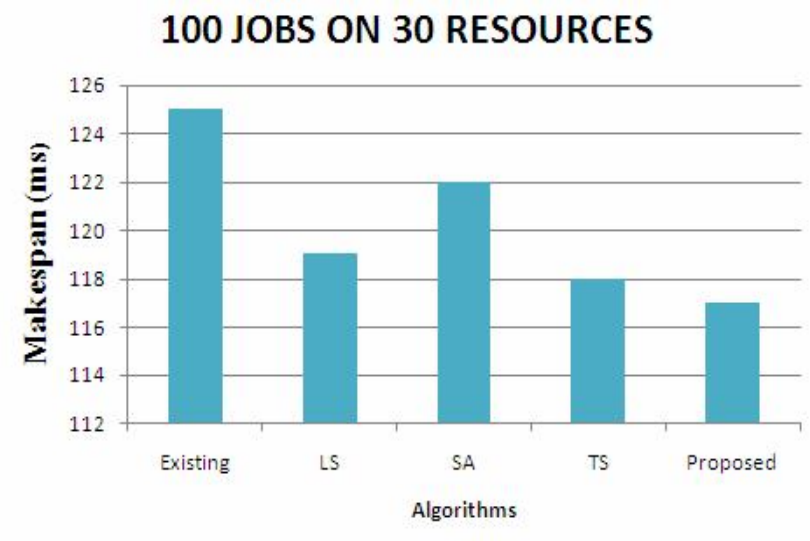

Figure 5.5 Makespan Comparison for Scheduling 100 Jobs on 30 Resources

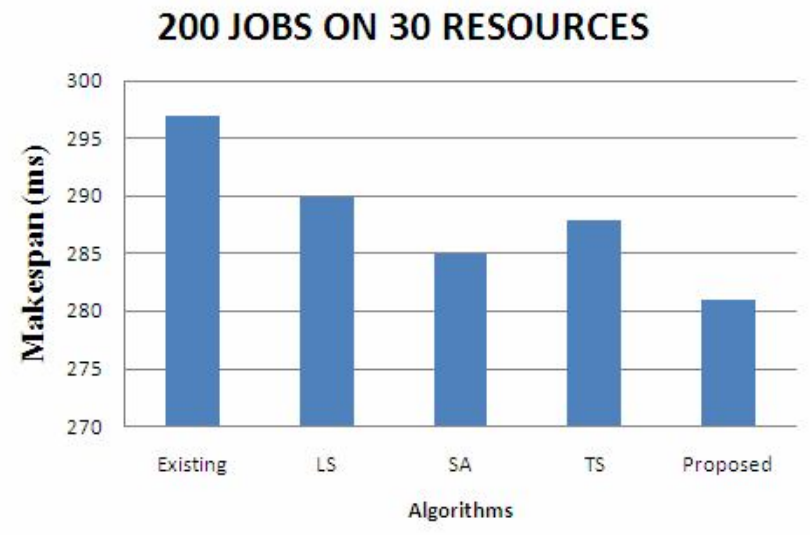

Figure 5.6 Makespan Comparison for Scheduling 200 Jobs on 30 Resources

5.7 CONCLUSION

An attempt has been made to combine the heuristics rules such as HLS, HSA and HTS for scheduling jobs in the grid environment. The heuristic rule on the top of heuristics offers better results than the individual heuristics in all the test cases. The performance of the system gets considerably increased due to reduction of unconfirmed jobs and compilation time. The performance of the scheduler can be improved by using parallel integration.

The proposed optimistic heuristics rule scheduling algorithm is capable of discovering the resources to execute the user's job. The jobs are separated as confirmed and unconfirmed jobs. The job nature is determined whether the jobs can be completed through current solution or randomized solution. If the unconfirmed jobs repeatly persist for long time in the scheduling queue then the unconfirmed jobs are sent to the tabu list. The proposed integrated heuristic rule algorithm effectively deploys jobs into execution unit after the jobs are confirmed.