Grid Scheduling Strategy using GA (GSSGA)

Dr.D.I.George Amalarethinam  
Director-MCA & Associate Professor of Computer Science, Jamal Mohamed College Tiruchirappalli, Tamilnadu, India.  
di_george@jmc.edu

F.Kurus Malai Selvi  
Assistant Professor of Computer Science  
Government College for Women, Kumbakonam, India.  
fkmsjoe@gmail.com

Abstract

Efficient scheduling algorithm is required to use the grid resources properly. List scheduling heuristic algorithms are to schedule the application programs and produces sub optimal solutions. An evolutionary based scheduling algorithm will produce an optimal solution. The proposed algorithm, namely, “Grid Scheduling Strategy using GA” (GSSGA) adopts the genetic algorithm to solve the dependent task based applications. The new fitness function has been developed in GSSGA and the conventional genetic operators like selection, crossover and mutation are used appropriately. After conducting numerous experiments, the proposed GSSGA outperforms the existing algorithms by minimizing the schedule length of the task graph.

Keywords: Grid Computing, Parallel Processing, GA, Task Scheduling, DAG, Fitness Function.

1. Introduction

The purpose of grid computing is to utilize computational power of idle resources which are located in different areas. The usefulness of a grid system largely depends on the efficiency of the system regarding the allocation of tasks to grid resources. The task scheduling in grid environments strive to maximize the overall performance of the grid. This can be achieved by minimizing the makespan and communication cost while maximizing resource utilization. This paper proposes the Grid Scheduling Strategy using GA (GSSGA) to schedule a group of dependent tasks for grid environments. Genetic algorithms are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. The proposed algorithm uses the genetic algorithm (GA) as search technique to find an efficient schedule in grid computing and adapts new fitness function to find the suitability of the schedule. Furthermore, the single point crossover and mutation operations within the algorithm can move the solution away from the local-optimal solution towards a near-optimal solution. A simulation study with randomly generated Directed Acyclic Graph (DAG) is to compare the performance by four other scheduling algorithms. The results show that the proposed GSSGA outperformed all other scheduling algorithms across a range of scenarios. This paper is organized as: Section 2 discusses the related work of GA. Section 3 contains the proposed GSSGA algorithm. Section 4 discusses the results and discussion and Section 5 concludes the paper.

2. Related Work

Genetic Algorithms are a family of computational models inspired by evolution. GA is the procedure used to find approximate solutions to search problems through application of the principles of evolutionary biology. Genetic algorithms use biologically inspired techniques such as genetic inheritance, natural selection, mutation and reproduction (recombination, or crossover).

The genetic algorithm differs from other search methods as it searches among population of points and works with a coding of parameter set, rather than the parameter values. It also uses objective function information without any gradient information.

GA is a technique to find optimal or nearly optimal solutions of search problems. In 1960 John Holland had an assertive thought and worked on GA. He published the first paper “Adaptation in Natural and Artificial System” in the year 1975 [1]. He invented GA as meta-heuristic search based on “Survival for the fittest”- a common ideology of biology. He introduced mutation and reproduction methodology of biology into the artificial system. From that time the term Gene,
Chromosome, Individual, Population, Crossover and Mutation are used in this search technique.

Majority of GA follows only makespan as its fitness function. Some genetic algorithms are using more than one objective function as the fitness function. Albert.Y.Zomaya and Yee-Hwei Teh [2] have used three objective functions, namely, minimizing the execution time, maximizing the processor utilization and a well balanced load to the resources. Kun-Ming Yu, Cheng-Kwan Chen [3] proposed the fitness function includes the file size $T(i)_{file}$ in Mega Byte of $T(i)$, the output size $T(i)_{result}$ its length $T(i)_{length}$, bandwidth $B(j)$ and processing capacity $P(j)$ of resource j. Rachhpal Singh [4] proposed a fitness function which contains the makespan and communication time. Vikas Gaba and Anshu Prashar [5] proposed a fitness function as average waiting time.

3. The Proposed Grid Scheduling Strategy using GA (GSSGA)

Genetic Algorithm of this study is to optimize the dependent tasks represented by Directed Acyclic Graph (DAG). Most of the GA from the literature is using the independent tasks. The dependent task in the grid environment is very difficult when compare to independent task. The objective function focuses on minimizing the makespan and communication cost while maximizing the utilization of resources.

Steps of Genetic Algorithm

The major steps of GA can be understood easily. GA starts with a set of solutions and ends with optimal solution. The following steps of GA are common and it is being followed in almost all GAs.

Procedure GSSGA

2. [Fitness] Evaluate the fitness function $f(i)$ of each chromosome $i$.
3. [New Population] Create a new population by repeating following steps until the new population is complete.
   (i) [Selection] Select the chromosomes from the population to perform GA operations.
   (ii) [Crossover] Crossover the parents to form new offspring.
   (iii) [Mutation] Mutate new offspring at each position in chromosome, if needed.
   (iv) [Validity] The new offspring has to be checked for its validity.
4. [Accepting] The valid chromosomes are used in the process of producing new generation.
5. [Test] If the end condition is satisfied, stop and return the best solution from the current generation
6. [Loop] Otherwise repeat the same process from step 2.

3.1. Chromosomal Representation of GSSGA

The first phase of the Genetic Algorithm is the proper representation of chromosome. A chromosome is a set of parameters which define a proposed solution of a problem. The key issue of genetic algorithm is to encode the solutions of the problem as chromosomes. Various encoding methods have been created for particular problems to provide effective implementation of genetic algorithms. Binary encoding (i.e., the bit strings) is the most common encoding method because of existing GA’s theories is based on the assumption of using binary numbers. But this encoding is not suitable for scheduling to represent the job/task, resource pair. Many encoding methods have been proposed for scheduling problem [6], in which the value encoding method is suitable for scheduling problem. The scheduling solution consists of task and resource in a sequence. The task and resource are denoted as integer number. For example the set (4, 3) is the 4th task mapped with 3rd resource. This paper considers chromosomes as strings. The examples of encoded chromosomes relating to our problem are:

- Chromosome 1 = (1, 4), (3, 2), (3, 1), … (n, m)
- Chromosome 2 = (1, 2), (4, 1), (5, 3), … (n, m)

where $n$ - number of tasks and $m$ - number of resources. The valid scheduled solutions alone are considered for the chromosome. These chromosomes are used for the initial population.

3.2. Initial Population of GSSGA

The size of the population of chromosomes is assumed to be 20. In which the four chromosomes are generated through Min-Min, HEFT, LCTSA and EDOS algorithms and the rest are generated randomly with task number
and resources id. The validity of the initial population is checked using the fitness function.

3.3. The proposed Fitness function of GSSGA

A fitness function is used to measure the quality of the individuals in the population. The fitness function should encourage the formation of the solution to achieve the objective function. It quantifies the optimality of a solution so that a particular solution may be ranked against all the other solutions. It depicts the closeness of a given solution to the desired result.

All minimization problems should be converted into the maximization problem, at the same time the optimum point should remain unchanged. The proposed fitness function has three parameters such as the makespan, resource idle time and number of resources. The newly developed fitness function $F(i)$:

$$F(i) = \frac{1}{MS} + \frac{NOR}{ARI}$$

where $i$ is the chromosome, $MS$ is makespan, $NOR$ is Number of Resources and $ARI$ is Average Resource Idle time. Using this function the initial population and other scheduling list will be selected.

The fitness function of each string is to be used for selecting the best strings using the Roulette wheel method.

3.4. The Selection operation of GSSGA

The selection operator makes more copies of better strings in a new population which has been extracted from an existing population. The new populations with good strings form a mating pool. To sustain the generation of a new population, the reproduction of the individuals in the current population is necessary. The Roulette wheel selection method is used in GSSGA. In roulette wheel, individuals are selected with a probability that is directly proportional to their fitness values i.e. an individual’s selection corresponds to a portion of a roulette wheel.

The probabilities of selecting a parent can be seen as spinning a roulette wheel with the size of the segment for each parent being proportional to its fitness. Obviously, those with the largest fitness (i.e. largest segment sizes) have more probability of being chosen. The fittest individual occupies the largest segment, whereas the least fit have correspondingly smaller segment within the roulette wheel. Selecting N individual from the population is equivalent to playing N games on the roulette wheel, as each candidate is drawn independently.

This selection operation is adopting the methodology which was proposed by Kalyanmoy Deb [7]. These strings will be passed to undergo the crossover operation.

3.5. The Crossover operation of GSSGA

It is a genetic operator that combines (mates) two chromosomes to produce a new chromosome. It selects genes from parent chromosomes and creates a new offspring. This operator is to produce a new string from the existing one. In this paper single point crossover is applied to produce a new string. From this the best individual string will be selected for the number of population using the fitness function.

The probability of crossover rate is 0.65.

After performing the crossover operation, all new strings will undergo the validity process. The first step of this validation is checking the existence of total number of tasks. The second step is verifying the precedence constraints. If both conditions are satisfied, that strings are valid strings. The valid strings are selected as the new population for the next generation of the GA. Remaining strings i.e. invalid strings will go to the mutation phase.

3.6. The Mutation operation of GSSGA

The invalid strings undergo mutation operation to make as valid strings with one change. The Mutation takes place in any one of the task number or the resource number of a string. The non-existence of a task in a string will be located in the appropriate place. Similarly if the precedence constraints are not satisfied, interchange any one task to fulfill the precedence constraints. Single point mutation is carried out in GSSGA. The probability rate of mutation is 0.01. After this process, the strings are once again checked for validity with fitness function and the valid strings are accepted as the population for next generation.

The termination condition of this algorithm is considered as the hundred percentage possibility of same selected string as the best solution. If the termination condition is
satisfied, the best solution is taken as the solution otherwise it will repeat the process from the step2.

4. Results and Discussion

The proposed Grid Scheduling Strategy using Genetic Algorithm (GSSGA) has been implemented and conducted exhaustive experiments to analyze its performance. For testing the proposed GSSGA, irregular graphs are considered. The screen shots of the results are shown in Figure 1.

The result details are compared with the schedules produced by well known Min-Min [8], Heterogeneous Earliest Finish Time (HEFT) [9] and EDOS [10] algorithms. The performances of these algorithms are compared based on the makespan, resource utilization, communication time and speedup ratio. The Table 1 gives the value of parameters used in GSSGA. These values are followed throughout the experiments of genetic algorithm.

**Table 1 Value of Parameters**

<table>
<thead>
<tr>
<th>The population Size</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover Rate</td>
<td>0.65</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The number of dependent tasks has been taken for experiment is 20, 30, 50, 75, 100 and 200 with different Communication-Computation Ratio (CCR) 0.2, 0.6 and 1.0. The number of resources considered for testing is 4 and 8.

4.1. Makespan

The proposed GSSGA algorithm gives the best results of the other algorithms. For instance, in the arbitrary task graph with 20 tasks and CCR = 0.2, the GSSGA completes the schedule with 88 time units but Min-Min takes 103.1429 time units, HEFT takes 99.8095 time units and EDOS takes 93.619 with 4 resources in the grid environment is shown in Figure 2. Similarly, in the random graph of 30 tasks and 200 tasks with CCR=0.2 GSSGA algorithm yields comparatively better results.

![Figure 2 Comparison of GSSGA, Min-Min, HEFT and EDOS with makespan for arbitrary task graphs](image-url)
4.2. Resource Utilization

The usage of the resources is calculated which are reserved in advance. For instance, in the arbitrary graph with 30 tasks and CCR = 0.6 the GSSGA utilizes the resources with 70.041 percentage but Min-Min , HEFT and EDOS utilizes 46.5613, 51.8463 and 56.15124 percentages respectively with 4 resources in the grid environment as shown in Figure 3. Similarly, in the arbitrary graphs with 20 and 200 tasks having CCR=0.6, the GSSGA algorithm yields comparatively better results.

![Graph](image)

Figure 3 Comparison of GSSGA, Min-Min, HEFT and EDOS with Resource Utilization for arbitrary task graphs

4.3. Communication Time

The time taken to transfer data from one resource to another resource is communication time. For example, in the arbitrary graph with 200 tasks with CCR = 1.0 the communication time of GSSGA is 3214 but Min-Min is 3344, HEFT is 3675 and EDOS is 3691 with 4 resources in the grid environment as exposed in Figure 4. Similarly, in the random graph with 20 and 30 tasks and CCR=1.0 the algorithm yields comparatively better results.

![Graph](image)

Figure 4 Comparison of GSSGA, Min-Min, HEFT and EDOS with Communication cost arbitrary task graphs
4.4. Speedup Ratio

The speedup of the proposed GSSGA algorithm has been analyzed with varying number of tasks as 20, 30, 50, 75, 100 and 200 having different CCR values such as 0.2, 0.6 and 1.0 is shown in Figure 5.

Figure 5 shows the performance of the proposed GSSGA algorithm. In all cases it is found that the performance in computation-intensive graphs is better when the CCR value is low.

From the Figure 5, it is observed that the speedup increases with the number of tasks. Hence, it is concluded that an application with large number of tasks yield good results.

5. Conclusion

The proposed Grid Scheduling Strategy using GA (GSSGA) for dependent tasks in grid environment is used to generate optimal solution from the available schedule. It is an evolutionary approach to derive the best solution from the existing schedule. The initial population is generated using different list scheduling algorithms.

The newly developed fitness function is working well to check the survival of the fittest. The results are encouraging when compared with other algorithms.

References:


