LGR: A Genetic Based Approach for Grid Computing Systems

PRAVAT KUMAR RAUTRAY
Assistant Professor, Dept. of Computer Science & Engineering, Aryan Institute of Engineering & Technology, Bhubaneswar
DEEPAK PANDEY
Department of Computer science and Engineering, Raajdhani Engineering College, Bhubaneswar, Odisha

Dr.PRAKASH KUMAR SARANGI
Department of Computer science and Engineering, NM Institute of Engineering and Technology,Bhubaneswar , Odisha

***Abstract*—The computational grid provides a promising platform for the deployment of various high-performance computing applications. In computational grid, an efficient scheduling of task onto the processors that minimizes the entire execution time is vital for achieving a high performance.**

Solving this problem is very hard and many attempts have been made to solve the problem. Using classical algorithms, With regard to the complexity of this problem, is not the good way; so the indefinite method acts better than classical method. Evolutionary algorithms are the best choice for solving this hard problem. In this paper, contrary to prior ways, the new string representation has been used, communication costs has not been ignored and presents as a major factor for reaching to optimum solution

***Index Terms*—Grid Computing Systems, LGR, Genetic Algorithm, Scheduler**

I. INTRODUCTION

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today [1]. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce [1].Recent research on these topics has led to the emergence of a new paradigm known as grid computing [2].These powerful paradigm has been used in various sciences such as spaceship process imaging and medical science [3], [4]. To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important [1].

Simply, grid scheduling is allocating a set of tasks to the processor such that the scheduling time minimized. It is known to be NP-complete for the general case and even for many restricted cases [5].Because of, the classical algorithms are not dynamic, they can not achieve the optimal scheduling for all situations, and therefore these algorithms cannot adapt themselves with all situations. Genetic algorithm has been widely used to solve this problem. In most cases the methods are quite effective but not efficient enough, and some important aspects such as the time of transferring data between processes are ignored [5].

We assume that the multiprocessor system didn't have much different in their powerfully. They are preemptive. They have not priority among task. The communication costexists. We use the Genetic Algorithm (GA) to solving this complete problem that is a randomly method for solving the problem, GA inspired from natural evolutionary process. However, this way has not guarantee the optimum response but always find solution close to optimum or optimum.

Simply, Scheduler assigns task to resources to achieve specified time requirements. The goal of grid scheduling is to find an optimization algorithm to minimize the overall execution time for a collection of tasks. One grid computing system consists of m processor with different communication cost and during the processing, each processor could be in contact with several linking line [6], [7], [8], [9] .simply, we want to decide what processor, at when time can schedule thesupposed task.

In grid scheduling problem, representation of priority of tasks is very important so we want to take precedence relations among the tasks in addition; their communication cost, number of task, execution time of each task and numberof processor. The relationship between tasks can be represented via a DAG, G = (V, E) where G is a graph with Vas nodes representing tasks and E as edges representing prerequisite constraints and communication links. Figure 1 illustrates one sample for DAG.

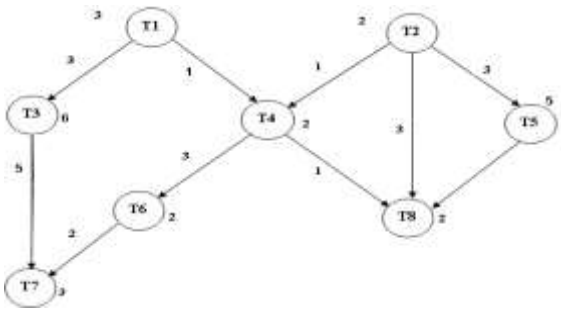

Figure 1. Example of one DAG

Figure 1 show that the T3 can't execute until the executing of T1 finished, the communication cost between T1 and T3 is 3 time unit and T3 need 2 time unit for executing.

This paper aims to present the new approach and it is organized as follows: The next subsection presents related work. The following one recalls the genetic algorithms. The fourth section presents Linear Genetic Representation (LGR) method. Finally, experimental results and conclusion to this work are proposed.

II. Related Work

The previous solutions [10], [11], [12], [9], [13], [14], [15], [16], [17], [5], [4] have some lack, that we listed them:

1. The chromosomes representation isn't linear, that means if we have m processor the representation will have m rows. This representation has difficulties such as: The crossover and mutation operators will be difficult to work with relations among tasks [18].Figure 2 illustrates one chromosome for DAG of Figure 1 with this method.

P1	3	4	6	7	8
P2	2	1	5		

Figure 2. Traditional string representation

2. The communication cost almost is ignored, so on real machine the solution doesn't work.
3. The probability of producing the better child with crossover and mutation operators almost is very low and so destroys the best solutions.

III. Genetic Algorithm

GA is an efficient searching tool that was invented by John Holland [19].The genetic algorithm has great application for optimization of complicated problems particularly in where isn't adequate information about search space. Although, considering that, genetic algorithm isn't guarantee best possible solution, but normally, would provide optimum or partly optimum solution by suitable approximate at short time. For solving any problem by genetic algorithm, eight components must be defined [20]:

- **Representation (definition of individual):** represents each chromosome in the real world. A chromosome is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve.
- **Fitness function:** These function shows the fitness of each chromosome. It is used to evaluate the chromosome and also controls the genetic operators.
- **Population:** The role of the population is to hold possible solution.
- **Parent selection mechanism:** The role of parent selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation.
- **Reproduction:** The reproduction operator is based on the Darwinian notion of "survival of the fittest". Individuals taking part in successive generations are obtained through a reproduction process or evolution operation. Individual strings are copied into a mating pool according to their respective fitness values. The higher the fitness values of the strings, the higher the probability of contributing one or more offspring in the next generation.
- **Crossover operators:** Recombination operator selects two or more chromosomes and then produces two new children from them. It aims at mixing up genetic information coming from different chromosomes to make a new individual.
- **Mutation operators:** Mutation operator selects one chromosome and then produces one new child from it by a slight change over the parent.
- **Survivor selection mechanism:** The role of survivor selection is to distinguish among individuals based on their quality. This mechanism survives the individual among the passing from one generation to the next generation.
- **Termination Condition:** The condition to ending the running of genetic algorithm.

IV. The Linear Genetic Representation (LGR) Method

As mentioned before, the main drawback of previous genetic algorithms is their representation and crossover operations. Hence, we present the LGR method, which have different representation and consequently different crossover operations, at the below we discuss the LGR components:

A. Representation

For representation of individual we use modular arithmetic. As we know, Modular arithmetic can be handled mathematically by introducing a congruence relation on the integers that is compatible with the operations of the ring of integers: addition, subtraction, and multiplication. For a fixed modulus n, it is defined as follows: Two integers are said to be congruent modulo n, if their difference a − b is an integer multiple of n. If this is the case, it is expressed as: $a \equiv b \pmod n$ [21].

In LGR we use this arithmetic principle for defining individuals, for example Figure 3 is the one random string that produced for DAG of Figure 1, and two processors system.

0	2	4	1	0	6	3	0	0	7	5	0	0	0	8	0
P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2

Figure 3. One random LGR representation with two processors

Then in our method zero value location can be ignored. For example in Figure 3, P1 execute following tasks: T4, T3, T5 and T8 and P2 execute: T2, T1, T6 and T7. It can be more than one representation for only one task order. We discard the representation that did not observe the task order rule that yielded by DAG. Figure 4 shows another example of Figure 1's DAG with three processors.

0	1	0	6	4	0	3	0	2	0	0	7	5	8	0	0
P1	P2	P3	P1	P2	P3	P1	P2	P3	P1	P2	P3	P1	P2	P3	P1

Figure 4. One random LGR representation with three processors

As we see the processor P1 execute T6, T3 and T5, the processor P2 execute T1, T4 and T8 and the processor P3 execute T2 and T7.In this method the deadlock does not exist, all the tasks are present and the precedence relations among the tasks regards. If we want to know the special place like S with P processor in one chromosome, we only find this relation: $S \equiv b \pmod P$ that b is the digit of processor in those places. For example, if we want to know that, which processor executes the 8th location with three processors system, we use this relation: $7 \equiv b \pmod 3$ and b=1 so the task would be executes in P1 ($7 \equiv b \pmod 3$).

So our LGR solves the traditional representation's problem, and does not have their crossover difficulty.

B. Fitness Function

For evaluating fitness of each chromosome, we first evaluate the finishing time of each chromosome. Figure 5 shows the Gantt chart of the Figure 3's chromosome.

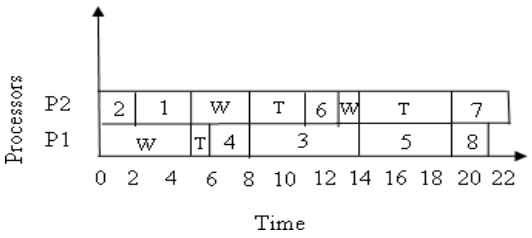

Figure 5. The Gantt chart of Figure 3's chromosome

At Figure 5, W shows waiting time and T shows transfer time. For example for starting the execution of the P1, this processor must be waiting 5 time unit (ET (2) + ET (1)) because it need for the T1 and T2 output data, 2 time unit for transferring data from T1 to T4 and considering that transferring data from T1 to T4 has been done during the executing of the T1.As you see the finishing time of executing is 22. If we use the finishing time for presenting the fitness of the chromosome, we most minimized the fitness of each chromosome. But basically the genetic algorithms have maximized fitness, so we reverse the finishing time for achieving fitness of them, the fitness of this chromosome be 1/22.

C. Population

In our method we consider the population size about 50 chromosomes. For initial the population, we produce the random number that is between the 1 and 16, and then if this location be empty, filled it by the loop number and allele (is the value of each gene) of remaining gene be zero. After that each producing checks the accuracy of the chromosome and if they are not accurate chromosome, discard it and produce new chromosome.

D. Parent selection mechanism

The LGR method for selecting the parent, evaluates the fitness of each chromosome. Then selects 10 chromosomes randomly, next for mutation operating this method selects the 5 best chromosomes, then selects one chromosome randomly. For crossover operating, this method select one chromosome from 5 best chromosomes randomly and another chromosome selecting from 9 remaining chromosome randomly.

E. Recombination operators

The LGR method uses the order crossover for recombination operator. After crossover the simulator check the priority relation among the tasks, if didn't observe the produced chromosome must be discarded and do the crossover again. LGR uses one point crossover and as we know, permutation-based representations present particular difficulties for the design of recombination operators [20].In LGR method we use order crossover that work in this way: [20]

1. Choose two crossover points at random, and copy the segment between them from the first parent (P1) into the first offspring.

2. Starting from the second crossover point in the second parent, copy the remaining unused numbers into the first child in the order that they appear in the second parent, wrapping around at the end of the list and if the gene is zero, this method copy the gene directly. At the end if the chromosome fuelled (there are no empty genes) and the entire task doesn't appear in the child for copying the gene, our method copy remaining method at the first gene that their allele is zero.

3. Create the second offspring in an analogous manner, with the parent roles reserved.

Suppose that the chromosome of the Figure 3 and Figure 6 are selected for recombination.

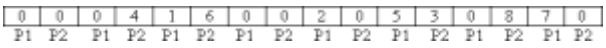

Figure 6. One random chromosome that selected for crossover

And suppose that the crossover point is after the ninth gene so after the crossover one child will be the chromosome of Figure 7.

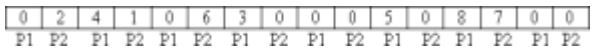

Figure 7. The produced chromosome by crossover operation

After each crossover operation, the LGR checks the accuracy of the offspring, and then it discards the illegal child and do the crossover again, supposes, the recombination probability is 75%.

F. Mutation operators

In the LGR method we use the swap mutation that is the permutated based mutation operator. This operator works by randomly picking two positions (genes) in the string and swapping their allele values [20]. Obviously after each mutation, the LGR checks the accuracy of the child and didn't pick the gene that their allele is zero. For example after the mutation on the Figure 3's chromosome, with the random 6 and 3 genes, we have the Figure 8's chromosome.

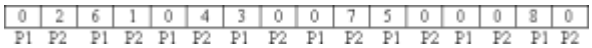

Figure 8. The produced chromosome by mutation operation

The mutation probability is 1/n, that n is the length of chromosome.

G. Survivor selection mechanism

After that the middle population size (mating pool size) is the 75% of the population size, the 25% of the best chromosome of the old population directly copped to the new population. The remaining chromosome, 75% of population size, replaced with the new chromosome from the middle population.

H. Termination condition

In this method, our algorithm running until no improvement in the fitness of the best member of the population has been observed for 20 generation.

I. Summary of LGR

In the previous sections, we described the LGR method's

summarized in Table 1.

Table1. Summary of the LGR

Representation	Permutations (The numbers are between 1 and t, that, t is number of task.)
Recombination	"Order one" crossover
Recombination probability	75%
Mutation	Swap mutation
Mutation probability	1/n (n=m*t)
Parent selection	First, selects 10 chromosomes randomly, and then selects 2 from 5 best chromosomes randomly.
Survival selection	Replace worse
Population size	50
Initialization	Random
Termination condition	No improvement in Last 20 generations.

In this table the 'm' stands for number of machine (processor),'t' stands for number of tasks and n is length of chromosome.

V. EXPERIMENTAL RESULT

After simulating Figure 1's DAG and considering the preference between tasks with two processors, the LGR method gives the Figure 9 chromosome for response.

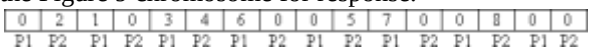

Figure 9. The produced chromosome for response

Then the P1 executes task in order to: T1, T3, T6 and T7 and P2 executes: T2, T4, T5, and T8. The fitness of this chromosome is 0.0769 so the finishing time is 1/0.0769 = 13time unit. This chromosome's fitness is almost good. Figure 10 is the Gantt chart of this chromosome, as we see the finishing time is 13 then the fitness is about 0.0769.

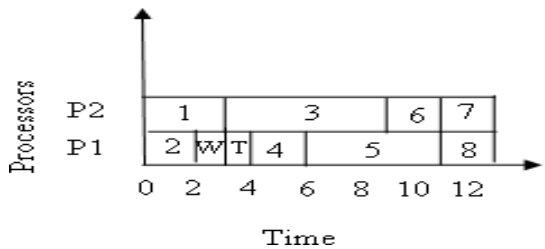

Figure 10. The Gantt chart of result chromosome

As we see in the figure 10, the time that each processor didnot execute the task, minimized and the finishing time become better.

VI. CONCLUSION

In grid computing properly assigning the tasks among processors are important factors. In this paper, we described the computational grid scheduling problem, genetic algorithm and LGR method for solving this hard problem and our method contrary to previous method considers the communication costs. Our method causes that the crossover be very simple.

REFERENCES

[1] Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing": State of the Art and Open Problems, School of Computing, Queen's University Kingston, Ontario January 2006.
[2] R. Buyya and D. Abramson and J. Giddy and H. Stockinger, "Economic Models for Resource Management and Scheduling in GridComputing", in J. of Concurrency and Computation: Practice andExperience, Volume 14, Issue.13-15, pp. 1507-1542,Wiley Press, December 2002.
[3] F. Berman, High-Performance Schedulers, chapter in The Grid: Blueprint for a Future Computing Infrastructure, edited by I. Foster and C. Kesselman, Morgan Kaufmann Publishers, 1998.
[4] Imtiaz Ahmad , Muhammad K. Dhodhi, "Short Communication Multiprocessor Scheduling in a Genetic Paradigm", Elsevier , pp 395-706, 1996.
[5] Mohammad Hassan Shenassa, Mahdi Mahmoodi , "A novel intelligentmethod for task scheduling in multiprocessor systems using genetic algorithm", Science Direct , Journal of the Franklin Institute 343 (2006)361–371 , 2006.
[6] Yi-Wen Zhongiz, Jian-Gang Yang', "A Genetic Algorithm For Tasks Scheduling In Parallel Multiprocessor Systems", Proceedings Of The Second International Conference On Machine Learning And Cybernetics, Xi'an, 2-5 November 2003.
[7] Pai-Chou Wang, Willard Korfhage, "Process Scheduling Using Genetic Algorithms", IEEE, 1995.
[8] E. S. H. Hou, R. Hong, And N. Ansari, "Efficient Multiprocessor Scheduling Based On Genetic Algorithms", IEEE, 1990.
[9] Ricardo C. CorreA, Afonso Ferreira and Pascal Rebreyend, "Scheduling Multiprocessor Tasks with Genetic Algorithms", IEEE Transactions on Parallel and Distributed Systems, Vol. 10, No.8, August 1999.
[10] Adam TL, Chandy KM, Dicksoni JR. "A comparison of list schedules for parallel processing systems". Communications of the ACM 1974;17(12):685–90.
[11] Wu MY, Gajski DD. Hypertool:"a programming aid for message-passing systems". IEEE Transactions on Parallel and Distributed Systems 1990;1(3):330–43.
[12] Yang T, Gerasoulis A. DSC: "scheduling parallel tasks on an unbounded number of processors". IEEE Transactions on Parallel and Distributed Systems 1994;5(9).
[13] Thanalapati T, Dandamudi S. "An efficient adaptive scheduling scheme for distributed memory multicomputers". IEEE Transactions on Parallel and Distributed Systems 2001;12(7):758–68,2001.
[14] Nissanke N, Leulseged A, Chillara S. "Probabilistic performance analysis in multiprocessor scheduling". Journal of Computing and Control Engineering 2002;13(4):171–9.
[15] Corbalan J, Martorell X, Labarta J. "Performance-driven processor allocation". IEEE Transactions on Parallel and Distributed Systems 2005;16(7):599–611.
[16] Marin Golub , Suad Kasapovic , "Scheduling Multiprocessor Tasks With Genetic Algorithms", Zagreb, Croatia , 2001.
[17] Hou ESH, Ansari N, Hong R. "A genetic algorithm for multiprocessor scheduling". IEEE Transactions on Parallel and Distributed Systems 1994;5(2):113–20.
[18] Reakook Hwang, Mitsuo Gen, Hiroshi Katayama," A comparison of multiprocessor task scheduling algorithms with communication costs", ScineceDirect , Computers & Operations Research 35 (2008) 976 – 993,2008.
[19] Holland, J. H. (1975) "Adaption in Natural and Artificial Systems". The University of Michigan Press, Ann Arbor, MI, 1975.
[20] A.E.Eiben, J.E.Smith, "Introduction to Evolutionary Computing", Springer, 2004.
[21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein." Introduction to Algorithms", Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 31.3: Modular arithmetic, pp.862–868, 2001.