INTERNATIONAL JOURNAL OF ENHANCED RESEARCH IN SCIENCE TECHNOLOGY & ENGINEERING
VOL. 1, ISSUE 2, NOV. 2012 ISSN NO: 2319-7463

Review on Genetic Algorithm and Its Application

Hemant kumar bansal[1], Devesh Mahor[2]
[1]M.Tech. Scholar (VLSI), [2]Assisstant Professor (ECE), MMU, Mullana, India.
kumar.hemant4289@gmail.com, mahor.devesh@gmail.com

Abstract: Genetic algorithm is very powerful technique to find approximate solution to search problems or patterns through application based on biological terms. Genetic algorithms use biologically inspired techniques such as genetic inheritance, natural selection, mutation, and sexual reproduction (recombination, or crossover. The general strengths of genetic algorithms lie in their ability to explore the search space efficiently through parallel evaluation of fitness and mixing of partial solutions through crossover, maintain a search frontier to seek global optima and solve multi-criterion optimization problems. In this paper, Application based on scheduling static tasks in homogeneous parallel system is discussed. Especially uni-processor system would not be sufficient enough to execute all the tasks parallel so it requires an efficient algorithm to determine when and on which processor a given task should execute A static task can be partitioned into a group of subtasks and represented as a DAG (Directed Acyclic Graph), that problem can be stated as finding a schedule for a DAG to be executed in a parallel multiprocessor system**.**

Keywords: Genetic Algorithm (GA), evolutionary computation, application.

I. INTRODUCTION

The field of genetic and evolutionary computation (GEC) was first described by Turing, who suggested a base model for the genetic algorithm in 1945.Firstly genetic algorithm [8, 9] is used in 1950s to study about actual biological evolution. Today genetic algorithm is used in many scientific research fields including computer science and electronics and communication like microchip design, embedded system, task scheduling, game theory etc. Also genetic algorithm is very popular searching procedure for pattern matching on the law of selection, genetic operation and mutation. Because genetic algorithms are based on a biological model, much of the terminology is taken from the biological analogy.
GAs can be used to perform meta-learning, or higher-order learning, by extracting features (Raymer *et al.*, 2000), or selecting training instances (Cano *et al.*, 2003), selecting features (Hsu, 2003). They have also been applied to combine or fuse classification functions

II. GENETIC ALGORITHM CYCLE

Firstly we have to generate the initial population which consists of a group of chromosomes. Initial population is generated randomly and each chromosome is bit strings which carries the information about the problem. Then we have to check the fitness for each chromosomes present in the initial population. Fitness means the maximum possibilities of that chromosome to be the part of final search or the chromosomes gives the best possible solution of the given problem. This can be evaluated by calculating the main objective function in decoded form. According to the fitness level of chromosomes we select the chromosomes to create the new chromosomes from current population using genetic operation which includes crossover and mutation.
Crossover is a process of recombination of one highest fitness level chromosomes to the other highest fitness level of chromosomes present in current population to produce offspring that contain the property of its parent chromosomes. Parent chromosomes are selected on the basis of survival of the fitness mechanism in nature and thus offspring has higher chance of surviving in the next succeeding generation. There are many recombination processes but simplest one is randomly select one chromosomes bit data and swaps the leading or trailing bit with the other chromosomes respective leading of trailing bit data. Many GA practitioners consider the crossover operator to be the determining factor that distinguishes the GA from all other optimization algorithms.
Some of the chromosomes are chosen for the mutation in which random bits are changed. In the mutation process, an arbitrary number of bits from a chromosome are changed, dependent on a random variable associated with each bit. There are two reasons for mutation. First one is mutation reduces the chance that the population converges upon a solution that is not optimum (a local

minimum) and second is the diversity of the population is increased and the introduced mutation may start the population heading towards the solution.
Once crossover and mutation phase is completed then again fitness level of new chromosomes are checked same as their parent chromosomes. Then replacement strategies are used to remove existing chromosomes from current population to make space for new chromosome. Few of the best chromosomes generated after mutation are copied into the succeeding generation. The best method for replacement is elitist strategy which may increase the speed of domination of new old population by new chromosomes and enhance the performance.
After replacement, algorithm checks whether the process should continue or not. And it is based on either the number of iteration is complete or predefined fitness level is reached by the top most chromosomes. If a termination criterion is not met then again we have to go to above procedure that is crossover and mutation.

III. HEURISTICS GENETIC ALGORITHM

In general genetic algorithm, all the stages cycles are very simple that is creation of population generation, selection of parent chromosomes, recombination, mutation and replacement. Number of bit strings depends upon the given problem which varies with different specification of problem. But in heuristics genetic algorithm [2, 3, 4, 5, 6, 7, 8] numbers of bit strings are equal to the number of nodes (tasks) present in direct acyclic graph. In basic genetic algorithm, initial population is created randomly which create more irrelevant bit strings or difficulties. Hence to reduce this problem we used heuristics genetic algorithms to create initial population.

IV. FRAME OF SOLUTION

a) Scheduling the task according to its priorities and that condition should be satisfied.
b) Each task should be done once in the schedule [2].

Each schedule contain list of few tasks. And each list the order of all tasks in which tasks should be run.

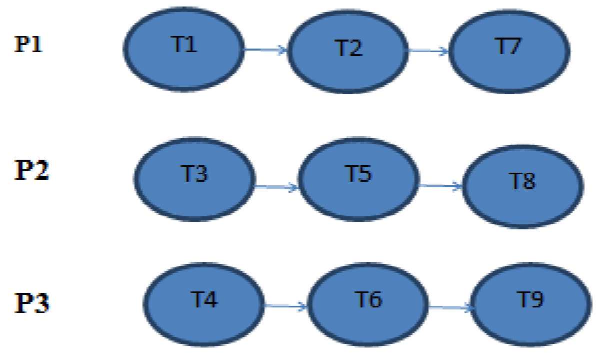

FIGURE 1: List Representation of a schedule

V. INITIAL POPULATION

To generate initial population, we require number of task, number of processor and population size. To avoid problem in assigning task, we set the priority of each task. Priority is decided on the basis of their completion time. It means the task which takes lesser time to complete will get the highest priority. This strategy is called min-min heuristic [1] or minimum time execution.
Once precedence is done, the problem of same execution time is removed which generally comes in homogeneous parallel system as all the processor take same execution time to complete one task.

Rules for scheduling task for each iteration:

a. Arrange the order of task according to their execution time in ascending order.
b. Calculate top-level and bottom-level [1] for each task.

c. Arrange the task having same execution time according to top-level or bottom-level in ascending or descending order respectively.
d. Assign the task to processor in order of top-level or bottom-level.

Numbers of bit strings are equal to the number of nodes (tasks) present in direct acyclic graph (DAG).

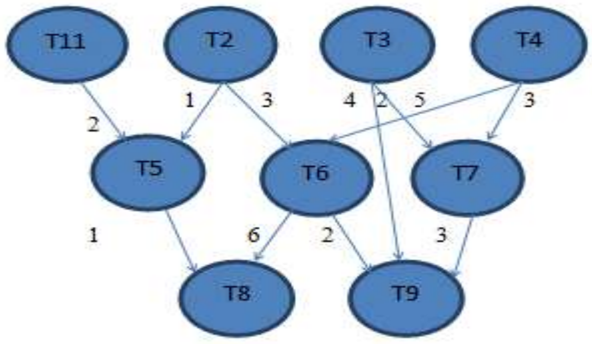

FIGURE 2: An example of a DAG

Bottom-level is defined as the length of longest path from current task to exit task. Top-level is defined as the length of longest path from entry task to current task.

TABLE 1: PRIORITY OF EXECUTION OF TASKS BASED ON THEIR EXECUTION TIME, TOP-LEVEL

Task no.	Execution time	Top level	Order of execution according to execution time	Order of execution according to Top level
1	4	0	7	1
2	3	0	3	2
3	5	0	9	3
4	2	0	1	4
5	3	6	4	5
6	3	7	5	6
7	4	8	8	7
8	2	16	2	9
9	3	15	6	8

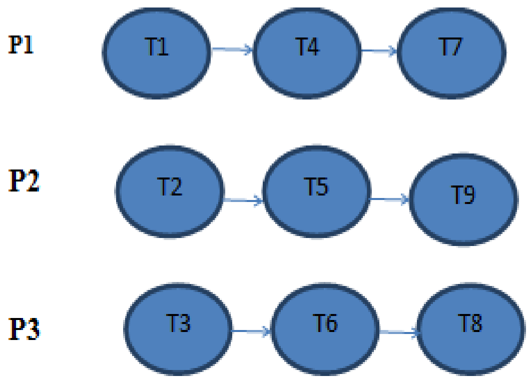

FIGURE 3: Initial Population using top-level resolution

TABLE 2: PRIORITY OF EXECUTION OF TASKS BASED ON THEIR EXECUTION TIME, BOTTOM-LEVEL

Task no.	Execution time	bottom level	Order of execution according to execution time	Order of execution according to bottom level
1	4	9	7	6
2	3	17	3	2
3	5	17	9	3
4	2	18	1	1
5	3	6	4	7
6	3	11	5	4
7	4	10	8	5
8	2	2	2	9
9	3	3	6	8

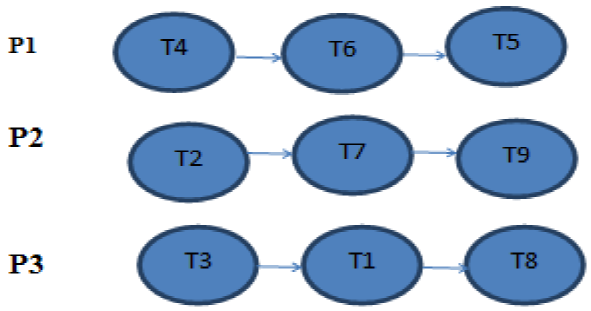

FIGURE 4: Initial Population using bottom-level resolution

VI. FITNESS VALUE [10, 11]

Some optimization criteria are also considered under which makespan and flowtime is minimized. *Makespan* is the time required by the job which is finished in last. *Flowtime* is the sum of finalization times of all the jobs.

Makespan: min {max Fj}
Si€Schj€jobs
Flowtime: min {∑ max Fj}
Si€schj€jobs

Fj is the time required by the job which is finished in last. Sch is the set of all possible schedules

VII. SELECTION OPERATOR

Selection operation is done with the help of above fitness value. Genetic algorithm uses selection operator to select the good chromosomes from the parent chromosomes to generate next generation according to their fitness value.

VIII. CROSSOVER OPERATOR

Crossover operator picks up two parent chromosomes from initial population having higher fitness values and randomly chooses their crossover points, and mates them to generated two child called offspring. Offspring represents the property of their parent chromosomes. Crossover points can be one and two both.

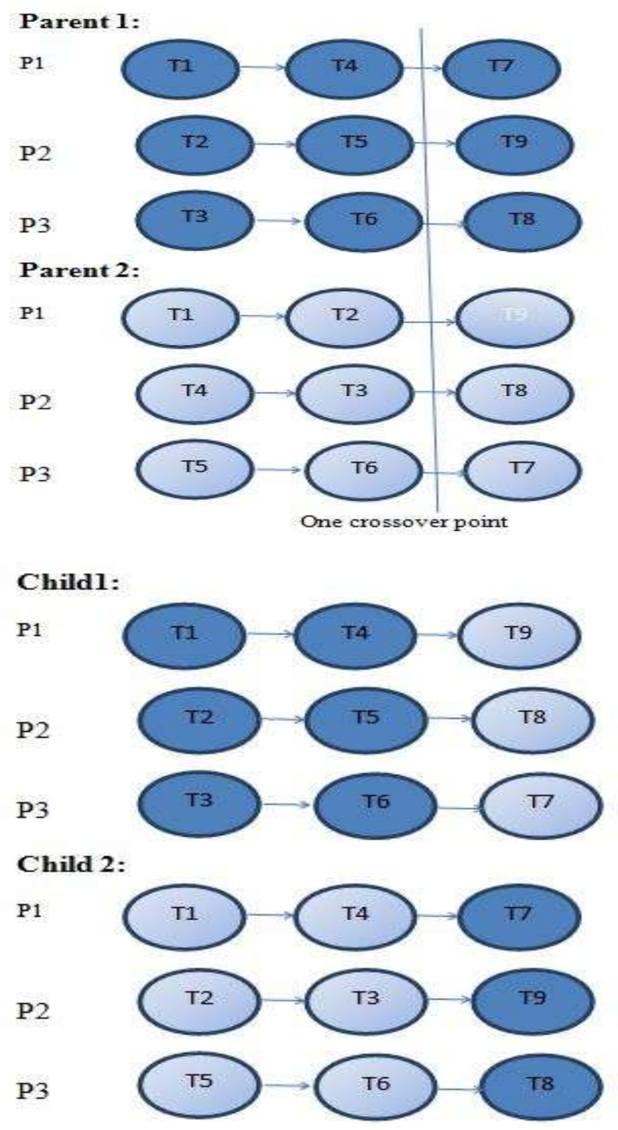

FIGURE 5: One Point Crossover

IX. MUTATION [2]

Genetic algorithm uses mutation operator which performs two major operations. One operation is swapping of allocation of two tasks which is selected randomly and other operation is picked up one task and changes its allocation. By doing so, makespan and flowtime is reduced.

CONCLUSION

By using this heuristic genetic algorithm, we reduce the execution time and hence performance of the system is enhanced. It gives the best possible solution for assigning tasks to parallel multiprocessor system without any conflicts.

REFERENCES

[1]. Kamaljit Kaur, Amit Chhabra, Gurvinder Singh, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security (IJCSS), Volume (4), Issue (2).

[2]. Tracy D. Braunt, Howard Jay Siegel, Noah Beck, Bin Yao, Richard F. Freund, "A Comparison Study of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, Volume 61, Issue.

[3]. Martin Grajcar, "Genetic List Scheduling Algorithm for Scheduling and Allocating on a Loosely Coupled Heterogeneous Multiprocessor System", Proceedings of the 36th annual ACM/IEEE Design Automation Conference, New Orleans, Louisiana, United States, Pages:280 – 285, 1999, ISBN: 1-58133-109-7, ACM New York, NY, USA.

[4]. Martin Grajcar, "Strengths and Weaknesses of Genetic List Scheduling for Heterogeneous Systems", IEEE Computer Society, Proceedings of the Second International Conference on Application of Concurrency to System Design, Page: 123, ISBN: 0-7695-1071-X, IEEE Computer Society Washington, DC, USA, 2001.

[5]. HesamIzakian, Ajith Abraham, Vaclav Snasel, "Comparison of Heuristics for scheduling Independent Tasks on Heterogeneous Distributed Environments", Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, Volume 01,Pages: 8-12, 2009, ISBN:978-0-7695-3605-7, IEEE Computer Society Washington, DC,USA.

[6]. Prof. Sanjay R Sutar, Jyoti P. Sawant, Jyoti R. Jadhav, "Task Scheduling For Multiprocessor Systems Using Memetic Algorithms", http://www.comp.brad.ac.uk/het-net/tutorials/P27.pdf.

[7]. Clayton S. Ferner and Robert G. Babb, "Automatic Choice of Scheduling Heuristics for Parallel/Distributed Computing", IOS Press Amsterdam, The Netherlands, Volume 7, Issue 1,Pages: 47 – 65, January 1999, ISSN:1058-9244.

[8]. D. E. Goldberg, "Genetic algorithms in search, optimization & machine learning", Addison Wesley, 1990.

[9]. Melanie Mitchell, "An Introduction to Genetic algorithms", The MIT Press, February 1998.

[10]. Edwin S. H. Hou, Nirwan Ansari, Hong Ren, "A Genetic Algorithm for Multiprocessor Scheduling", IEEE Transactions on Parallel and Distributed Systems, vol. 5, Issue. 2,February2003, Pages: 113-120, ISSN: 1045-9219, IEEE Press Piscataway, NJ, USA.

[11]. Amir Masoud Rahmani, Mohammad Ali Vahedi, "A novel Task Scheduling in Multiprocessor Systems with Genetic Algorithm by using Elitism stepping method", Science and Research branch, Tehran, Iran, May 26, 2008.