Genetic Algorithm with Descendants Idea for Scheduling Tasks Graph in the Multi-Processor Architecture

Mostafa Mahi and Halife Kodaz

***Abstract*—Nowadays, multi-processor systems are being used extensively in parallel computing. The effective scheduling system for implementing parallel programs to achieve high performance is crucial. The timing should be done in such a way that the total execution time of the program with according to time and tasks to be minimize communication between processors. This is a NP-Hard problem which tasks graph scheduling approaches based on deterministic methods are not effective in this context while the use of evolutionary computing and genetic algorithms to solve this problem effectively are mainly. In this paper, a new genetic algorithm is proposed for the problem of scheduling tasks graph which is able to spend less time to get. This algorithm is based on a new approach to minimize the length of the critical path and the cost of communication between processors. In this paper, we calculate the number of descendants for each node in which tries to minimize the total execution time of the program.**

***Index Terms*—Scheduling multi-processor, tasks graph, algorithm genetic.**

I. Introduction

Nowadays, multi-processor systems are used extensively in parallel computing. The effective scheduling system for implementing parallel programs to achieve high performance is crucial. The timing should be done in such a way that the total execution time of a program with regarding to minimize time of tasks and communication between processors. The task graph scheduling problem is a NP Hard problem [1], [2]. Therefore approaches based on deterministic methods will not work in this field. Previous approaches to solve this problem in scheduling tasks graph are based on a multi-processor architecture is presented by some non-genetic algorithms. The six non-genetic algorithm in the context are as follows: ETF [3]; DLS [4] LAST [5]; HLFET [6]; DSH [7]; and MCP [8]. Among these algorithms, MCP is the best non-genetic algorithm to solve this problem and according to that, it is used for evaluation and comparison with other algorithms [9]. Old algorithms try simplifying their assumptions about tasks graph. For example, some algorithms assume the duties or tasks are the same computational costs and certain other costs are considered discretionary. A number of algorithms, weighted edges, cost of communication between processors is assumed to be zero [9]. Number of processors is limited and in the come cases, it is considered unlimited. Also, new algorithms are designed to accept any arbitrary graph [9]. It is also applicable as parallel [10].

However, the numbers of presented genetic algorithms to solve the issue are very little and each of them has advantages and disadvantages. The main difference between the solutions is how chromosomes and genetic operators are displayed. Constraints on the displaying structure of chromosome have a major effect on the genetic operations complexity. Compare and evaluate the performance of these algorithms is difficult for two reasons [9]. First, most algorithms are based on assumptions of proposers, so it is difficult to compare. Secondly, it is the lack of a standardized instrument to test these algorithms. Also, many algorithms have been evaluated on problems with small size and performance for various sizes of problems is unclear [9]. Most heuristic methods for scheduling methods are based on the use of lists and queues with main idea to prioritize for graph nodes and put them in a sorted list by the priorities as down [9].

In this paper in the Section II, are presented scheduling tasks graph, view of the chromosomes and how produce initial population and selection operators, and crossover and mutation which are based on the number of mutations that have been expressed. In the Section III, the proposed algorithms are presented and finally in the Section IV and Section V, the proposed algorithm comparison and the results by means of tables and figure are indicated.

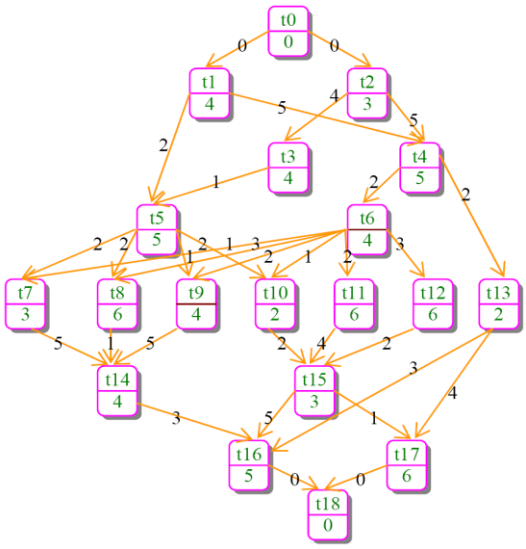

Fig. 1. Tasks Graph [1].

II. Scheduling Tasks Graph

Before scheduling algorithm should be expressed, in terms of the input algorithm as graph is described in this section. The input to the directional weighted graph called tasks graph

Manuscript received June 14, 2013; revised October 15, 2013.

Mostafa Mahi is with the Department of Computer Engineering and Information Technology, Payame Noor University, Tehran, Iran (e-mail: mahi@pnu.ac.ir).

Halife Kodaz is with the Department of Computer Engineering, Selcuk University, Konya, Turkey (e-mail: hkodaz@selcuk.edu.tr).

DOI: 10.7763/JACN.2014.V2.72

is shown as G = (V, E) in [11]. Each node is a member of the V set and represents a work unit of program so that the weight of the nodes determines the execution time of work unit. This graph contain set of edges shown with E, which implies that the prerequisite relationships between work units are to be having an edge (t_i, t_j), until the execution time of t_i not finished, t_j cannot begin to implement. These edges are weighted and the weight of each edge denotes the communication and message transforming costs between the two work units. The relevant work units run on different processors and if the processors are same, therefore the communication costs to be zero. Fig. 1 shows an example of a tasks graph [1].

The target is finding an optimal schedule for tasks graph execution on a multi-processor architecture where the total time of execution as the time of the last work unit to be minimized.

A. Showing Chromosomes

In order to displaying chromosomes, we use the new string representation for chromosomes. If we consider a separate string for each processor, only we will have order of running work units on the processors and the order of overall work units are unknown in which case should create other vector namely, location order vector. Therefore, we select different representation and use a one-dimensional array in which the order of overall running of work units to be specified. Each member of the structure has two fields: work unit number and processor number. Thus, the length of all chromosomes is equal and the overall structure of a chromosome will be as follow Table I [1].

TABLE I: HOW CODING CHROMOSOMES

.....	t_i	.....	t_j	.....	t_k	.....
.....	p_l	.....	p_m	.....	p_n	.....

B. How to Generate Initial Population

To generate the initial population, we create a arranged queue firstly that contains a number of work units. This queue is arranged based on the earliest possible start time as non-down for arranged work units. The algorithm calculates the earliest possible start time for each node (n_i) is as follows in which the critical path and the length of them is easily to be calculated [1]. Table II shows a sample chromosome structure.

TABLE II: A SAMPLE CHROMOSOME STRUCTURE.

T5	T4	T3	T2	T1
P1	P2	P0	P1	P0

The algorithm for calculating earliest possible start time:

- Tt_level (n_i) = 0 (Running time for root node)
- For each node, insert number of references as equal with the number of parent nodes and give it to the available list with all the nodes which the number of references is initialized as zero.
- Until available list is not empty, remove the node n_k from list and perform the follows works for each child node of it with the name n_j:
- Reduce the number of references only one unit and if the number of references to be zero, then add n_j to the available list.

We will assume a reference number for each work unit, originally is equal to the number of its immediate parent for it work unit. Then, we start from the beginning of queue and from among of available work units which all of their parents were running, we select one of them randomly and specify it to the processor. Then, we decrease the reference number of all children's work units to one unit less. So, the available work units for running will have the reference number, 0. This is done until all work units to be scheduled. It is mention that the selection of processor will not be randomly in order to assign work units. In this algorithm, if the current work units are members of critical paths, so it is allocated to the processor so that the previous work unit is running on that processor in the critical path. But, if the current work unit is not member of critical path, it is assigned to the processor which the work unit of them has the maximum communication cost with current work unit which has been run on it. If there were multiple communication costs, one of them will be chosen randomly. As this regard, algorithm tries to minimize amount of communication and will produce better solutions. If all members of the initial population will be generating with this algorithm, then it is will be faced to the risk of premature convergence and replicating solutions. Therefore, it is better to adding some number of members randomly to the population in order to maintain the diversity of population [1].

C. Operator Selection

The selection of the parent population is placing them in a middle population. Middle populations combine with together and produce children. Selection of proper couples from parent is performed by selection operator and a tournament. Thus, new population which generated is equal to the population. In this section, the purpose of the tournament is to select a random element chosen at random among the elements of the selected element is selection of some elements randomly which between the random selected elements of smallest fitness is chosen as a selected element. In this regard, selection chance of all elements is equal, but the selection chance of better elements is more than other elements.

D. Crossover Operator

TABLE III: A SAMPLE CROSSOVER OPERATION.

Parent 1	T0	T1	T2	T3	T4	T5
	P1	P0	P2	P1	P0	P2
Parent 2	T0	T1	T2	T3	T4	T5
	P0	P1	P0	P2	P1	P2
Child1	T0	T1	T2	T3	T4	T5
	P1	P0	P2	P2	P1	P2
Child2	T0	T1	T2	T3	T4	T5
	P0	P1	P0	P1	P0	P2

This operation was performed according to itself selected parameters and it uses a point method to combine the two chromosomes. After choosing a crossover-point randomly, we copy the first child from the first parent and also we copy the second child from second parent, but we copy from crossover-point to the end for first child of second parent and for second child of the first parent is copying. During the crossover operation, we only replace the parts of relative processors, and keep as constant the work units order. The distance of the crossover-point of chromosomes should be

equal, so the mutation operator can be able to produce valid children. Table III shows a sample crossover operation.

E. Mutation Operator

This operator is based on a parameterized probability and caused to diversity and escape of convergence. To accomplish this task, two work units were randomly selected and their corresponding processors are replaced. Another type of it, a work unit is randomly selected and randomly will be changed the related processor of it. On the other type of mutation, which used in this article, it is selected a work unit randomly and we will change the related processor of it randomly [1]. Table IV shows a sample mutation operation.

TABLE IV: A SAMPLE MUTATION OPERATION

Before of Mutation	**T 0**	**T 1**	**T 2**	**T 3**	**T 4**	**T 5**
	P1	**P2**	**P2**	**P0**	**P1**	**P2**
After of Swap Mutation	**T 0**	**T 1**	**T 2**	**T 3**	**T 4**	**T 5**
	P1	**P0**	**P2**	**P2**	**P1**	**P2**
Before of Mutation	**T 0**	**T 1**	**T 2**	**T 3**	**T 4**	**T 5**
	P1	**P0**	**P2**	**P2**	**P1**	**P2**
After of Change Mutation	**T 0**	**T 1**	**T 2**	**T 3**	**T 4**	**T 5**
	P1	**P0**	**P2**	**P0**	**P1**	**P2**

1) Mutation based on the number of descendants

In this type, the mutation firstly navigates the desired tasks graph and calculates the number of children and descendents of each node and then we counted them. Then, based on the number of descendants of each node, if the node has a number of descendants more than other tasks graph nodes, since it has been assigned higher priority to implementing and running on that node in the queue. Thus, applying a mutation operation among the nodes with high priority will be minimal execution time in the graph. Table V shows number of descendant tasks and processors.

TABLE V: NUMBER OF DESCENDANT TASKS AND PROCESSORS.

Tasks	T0	T1	T2	T3	T4	T5
Number of Descendants	12	10	9	5	3	0
Processors	P0	P1	P0	P2	P1	P0

III. PROPOSED ALGORITHM

In the follow, overall structure of this algorithm is presented as pseudo-code which the main idea of it is increase rapidly to reach a solution based on reducing communication costs and length of the critical path.

Step1:
-Create initial population as described in Section 2.2.
-Produce a quarter of the population randomly.
-produce the rest of the population as flows.
-Find the earliest start time(EST) for each task.
-Identify all the tasks according to EST in a linear list.
-Repeat
-Select randomly a task amongst the tasks ready to
-Schedule at the beginning of the linear list.
-if the select
Else Assign the task to the processor including a task
with highest interconnections with the task.
-Remove the selected task from the linear list.
-Until the linear list of tasks is empty.
Step2:
-While termination criteria not satisfied **Do**
-For each Chromosome in current population Do
- Calculate its fitness value as described in Section 2.3.
Create intermediate generation as follows:
-Add the fittest chromosome to the intermediate population.
-Repeat
-apply tournament selection to select two chromosomes.
-apply the crossover operator, described in section 2.4.
-apply the mutation operator, described in selection 2.5.
-Until the intermediate operation size is completed.
-Copy the intermediate population over current population.

IV. PROPOSED ALGORITHM COMPARISON WITH OTHER ALGORITHMS

According to the parameters in Table VI, the program runs for 20 times on the tasks graph and the results of them are presented in the Table VII and Fig. 2, show the results of this proposed algorithm are two unit less than previous algorithms.

TABLE VI: PARAMETER VALUES.

Parameters	Values
Number of Processors	3
Population Size	30
Number of Produced Generation	200
Probability of Crossover Operation	80%
Probability of Mutation Operation	5%
Probability of Mutation Operation based on Number of Descendants	50%
Probability of Initial population generation Randomly	20%

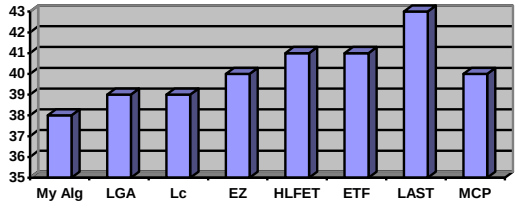

Fig. 2. Results of presented algorithms

TABLE VII: PROPOSED ALGORITHM COMPARISON WITH PREVIOUS ALGORITHMS

My Alg	LGA	LC	EZ	HLFET	ETF	LAST	MCP
(Best Run:37) (Average Run:39.21)	39	39	40	41	41	43	40

V. CONCLUSION

Due to the scheduling of tasks graph is NP-Hard task, approaches based on uncertain techniques in this regard such as evolutionary computing mainly genetic algorithms will be effective. Therefore, in this paper a specific algorithm (attention to the DNA), a genetic algorithm is proposed for scheduling the tasks graph that can be obtained an appropriate scheduling with spending less time. A new approach in this algorithm is reducing the descendants of the parent node and based on the decreasing the length of critical path and reducing of communication costs between the processors. Finally, the experimental results show that the proposed

method can be implemented timely in comparison with other similar algorithms. The practical results indicated that this algorithm dealing with graphs without communication costs is similar to other algorithms such as MCP algorithm which acts quickly. As the future work, it is better to using the proposed idea in this algorithm into the other genetic algorithm to find a fast and efficient algorithm for this problem.

REFERENCES

[1] S. Parsa, S. Lotfi, and N. Lotfi, "An evolutionary approach to task graph scheduling," Springer-Verlag Berlin Heidelberg, ICANNGA 2007, Part I, LNCS 4431, 2007, pp. 110–119.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[3] J. J. Hwang, Y. C. Chow, F. D. Anger and C. Y. Lee, "Scheduling precedence graphs in systems with inter-processor communication times," *SIAM Journal on Computer*, vol. 18, no. 2, pp. 244-257, April 1989.

[4] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures," *IEEE Transaction on Parallel and Distributed Systems*, vol. 4, no. 2, pp. 75-87, February 1993.

[5] J. Baxter and J. H. Patel, "The LAST algorithm: a heuristic-based static task allocation algorithm," in *Proc. International Conference on Parallel Processing*, vol. 2, August 1989, pp. 217-222.

[6] Y. Kwok and I. Ahmed, "Benchmarking the task graph scheduling algorithms," in *Proc. IPPS/SPDP*, 1998, pp. 531-537.

[7] B. Kruatrachue and T. G. Lewis, "Duplication scheduling heuristics (DSH): A new precedence task scheduler for parallel processor systems," Technical Report, Oregon State University, Corvallis, OR 97331, 1987.

[8] M. Y. Wu and D. D. Gajski, Hypertool: A programming aid for message-passing systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 1, no. 3, pp. 330-343, July 1990.

[9] Y. K. Kwok and I. Ahmad, "Benchmarking and comparison of the task graph scheduling algorithms," *Journal of Parallel and Distributed Computing*, vol. 59, pp. 381-422, 1999.

[10] S. Hunold and J. Lepping, "Evolutionary scheduling of parallel tasks graphs onto homogeneous clusters," in *Proc. 2011 IEEE international conference on cluster computy*, Austin, Texas USA, 2011.

[11] Kwok and I. Ahmad, "Benchmarking and comparison of the task graph scheduling algorithms," Department of Electrical and Electronic Engineering, University of Hong Kong, Pokfulam Road, Hong Kong, March 17, 1999.

Mostafa Mahi is from the Department of Computer Engineering and Information Technology, payame Noor University, Tehran, Iran. Now, he is pursuing his Ph.D. degree at the Computer Engineering Department of Selcuk University. His research interests are Factoid Question Answering System and Machine Learning Algorithms.

Halife Kodaz was born in Sivas on 15th April 1977. He graduated from Computer Engineering Department of Sel ɕık University with B.Sc. degree in 1999, from Computer Engineering Department of Sel ɕık University with M.Sc. degree in 2002 and from Electrical-Electronics Engineering Department of Sel ɕık University with Ph.D. degree in 2008. He works as an assistant professor at Sel ɕık University. His research interests are Artificial Immune Systems and Machine Learning Algorithms.