ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TASK SCHEDULING ALGORITHM FOR HETEROGENEOUS MULTI PROCESSING COMPUTING SYSTEMS

[1]A. YOUNES, [2]A. BEN SALAH, [3]T. FARAG, [4]F. A. ALGHAMDI, [5]U. A. BADAWI

[12345]Dept. of Management Information Systems, College of applied studies and community Service, Imam Abdulrahman Bin Faisal University, Dammam 34212 – Saudi Arabia

E-mail: [1]ayhamed@iau.edu.sa, [2]asalah@ iau.edu.sa, [3]thhanafy@iau.edu.sa, [4]faghamdi@ iau.edu.sa, [5]ubadawi@iau.edu.sa

ABSTRACT

The main purpose of task scheduling is to assign tasks onto available processors with the aim of producing minimum schedule length and without violating the precedence constraints. In heterogeneous multi-processing systems, task assignments and scheduling have a great impact on the system operation. In a heuristic based task scheduling algorithm, different process will result different task execution time (makespan) on a heterogeneous computing system. Thus, a good scheduling algorithm should be capable to efficiently assign precedence to each subtask depending on the resources required to reduce makespan. In this report, we propose a genetic algorithm (PGA) to resolve a task assignment and scheduling for homogeneous and heterogeneous multi-processing problem. The basic idea of this process is to exploit the advantages of heuristic-based algorithms to decrease space search and the time needed to get the best solution. The achieved results show that the suggested approach significantly outperforms the other approaches in terms of task execution time.

Keywords: *Heterogeneous Processors, Genetic algorithm, Heuristi algorithms, Task scheduling, Multiprocessing*

1. INTRODUCTION

One of the key preperformance measurements of any computing system is the execution time, and to reduce the execution time faster processors have been developed but it has a physical limitation, accordingly multi-processing system had been utilized. In multiprocessing system, a program is to be divided into tasks, such that each task executed on one of the processors. That task assignment and processors association is called task scheduling in multi-processing system.

To reach an optimal task scheduling and processor utilization in heterogeneous multi-processing system is computationally difficult goal. The term optimal may refer to many objectives combined. Usually the main objective is minimizing the makespan. Finding an optimal task scheduling is NP hard problem [1]. According to that, the heuristic algorithms are a good candidate to tackle that problem. Genetic algorithm has been implemented to solve task scheduling in many articles.

In this paper, a multi-processing system has been studied. Homogeneous and heterogeneous multi-processing have different capabilities of processing. Task processing time can be only determined when the task is assigned to a certain processor, i.e. task processing time is processor dependent. A genetic algorithm has been proposed (PGA) to find an optimal task scheduling assigned to heterogeneous multi-processing system.

The rest of this paper is presented as follows. Section 2 represents some related work for task scheduling problem for different structures of multiprocessing systems. The problem description is presented in section 3. In Section 4, the operations of the proposed algorithm (PGA) are illustrated. Our PGA approach to find the optimal task scheduling for homogeneous and heterogeneous multiprocessor system is described in section 5. The results obtained by applying the PGA and compared with other results of SGA, [13] present in section 6. Section 7 concludes the paper and future work.

2. RELATED WORK

The objective of task scheduling in multiprocessor system is to assign a dependent task on the processors and the processing time will be reduced. To minimize the processing time, the genetic algorithm has applied to the processors to obtain various solutions faster processing time. Task scheduling considers two aspects namely earliest start time (EST) and some task dependencies (NTD). This comparison made by using Java simulation and the result obtained that proposed algorithm solve minimum EST attains fasters processing time than maximum EST [1].

The study addressed the multiprocessor scheduling problems indicated directed acyclic task graph (DAG) with the communication costs. The authors proposed Priority-based Genetic Algorithm/Shortest Processor First schedule provide better solutions than the existing algorithms with respect to the completion time of resulting schedules and reduce the task execution time effectively [2].

The task scheduling algorithms using Efficient State Space Search Genetic Algorithm (ESSSGA) which use the benefits of heuristic-based algorithms to minimize space search and time to obtain effective solutions [3]. The task to processor mapping has made by using a heuristic-based earliest finish time approach which reduces the time regarding task execution time.

A genetic algorithm (GA) for the task scheduling in the multiprocessor systems has indicated that task execution priority depends upon the height of task graphs to perform scheduling. This method is simulated and used to compare with the basic genetic algorithm [4]. GA efficiency could be attained by the optimization of different parameters like mutation, crossover, and selection function and crossover probability. These GA parameters on the reduction of bi-criteria fitness functions and parameter setting will be accomplished by central composite design approach with design experiments. The experiments use these parameters and analysis of variance which reduce the total completion time and makespan [5].

MCP algorithms used for solving the problems in scheduling task graph. The algorithm is entirely dependent on the new approach to reduce the communication cost of processors and length of critical time. In order to solve the scheduling of task graph, effective GA has applied. GA proposed for this scheduling the task graph which can be acquired an effective scheduling with low time. The results obtained from the study stated that algorithm related to graph without communication cost can act quickly when compared to other [6].

The GA chromosomes like task list (TL), processor list (PL) and integration of both (TLPLC). The experiments on real-world application graph like Gaussian elimination, Gauss Jordan and Laplace equation and LU decompositions. TLPLCGA is related to GA and heuristic algorithms regarding time and efficiency of the processor have conducted. The result experienced hybrid approach performs better than the other algorithms [7].

NP-complete problem is solved by using the integration of heuristics and search techniques. GAs offer a robust and stochastic solution for many optimization problems. The design and implementation of the schedule length of task graph performed on the processors. To solve the scheduling problem, the GA has been applied [8]. The fitness evaluation approach is the time-consuming operation of a genetic algorithm that influences the GA performance.

A synchronous master-slave algorithm performs better than the sequential algorithm concerning a difficult and large number of generation problems. This research uses GA in SCHEDULE program to resolve the problem of multiprocessor scheduling. SCHEDULE is the simple tool for scheduling task and modeling on the multiprocessor system. It can easily modify the complex task graph into specified multiprocessor architectures. The result obtained from the study indicated that user change number of processors on the system and small changes to program would manage the inter processor communication delays and overhead costs [9].

The effectiveness of Node Duplication Genetic Algorithm (NGA) based approach against the existing deterministic scheduling techniques for reducing the inter processor traffic communication. The results get from the simulations indicates that GA can use schedule task to meet deadlines and acquire high processor utilization. Performance analysis of NGA is compared with GA, FCFS and List Scheduler [10].

The effective method on the GA is created to solve the problem of multiprocessor scheduling. This paper used GA for scheduling precedence task graph with inters task communication onto multiprocessors without considering the communication channel. Experimental results show that hard problems have taken from internet illustrates that GA with optimization of parameters

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[11]. To produce effective results, the large probability of global optimum should be considered.

Table 1: Notations

Notations	
G	A task graph
DAG	A Directed Acyclic Graph
t_i	Task i
p_i	Processor i
M	Number of tasks
N	Number of processors
n_i	Node i
$c(n_i)$	Cost of node i
$ST(n_i,p)$	Start time of node i on a processor p
$FT(n_i,p)$	Finish time of node i on a processor p
$RT(p_i)$	Ready time of the processor i
LT	A list of tasks according to the topological order of DAG.
$DAT(t_i,p_j)$	The Data Arrival Time of task i at processor j
CP	A critical Path of G
Pc	Crossover ratio
Pm	Mutation ratio
Pop_size	Population size
GN	Number of Generations
Maxgn	Maximum generation

3. PROBLEM DESCRIPTION

Figures The model of task scheduling in this work, can be described as distributed M tasks to be executed on N processors, the processors can be with different computing capabilities in general. A task graph can be mapped to descript the problem structure.

A task graph G is a Directed Acyclic Graph (DAG) composed of M nodes n1, n2, n3, …, n M. Each node of the graph is termed a task. A task is assumed to be a set of instructions that must be executed sequentially in an assigned processor. A task (node) may have pre-demanded data (inputs) before being executed. When all inputs are received, the node can be triggered to execute. These inputs expected to be delivered after some other task end their execution, as these tasks evaluate them. We named such relying as task dependency. If a task (t) is depended on other tasks data then we consider that tasks as the parents of the task (t), and task (t) is their child. A node with no parent is called an entry node and a node with no child is called an exit node, [13]. As shown in Fig.1.

The execution time of a task we call it the computation cost. Whenever, the computation cost of a node ni is denoted by (ni) weight. The graph also has E directed edges representing a partial order among the tasks. The partial order introduces a precedence-constrained DAG and implies that if (ni -> nj), then nj is a child, which cannot start until its parent ni finishes.

The weight on an edge represents the communication cost between the tasks and is denoted by c(ni; nj) , the communication costs matrix of Fig. 1 graph is shown in Fig. 2. The communication cost is considered only if ni and nj are assigned into different processors, otherwise it is calculated to be zero, in that case ni and nj are assigned to the same processor.

If a node ni is assigned into processor p, the start time and finish time of the node are denoted by ST(ni; p) and FT(ni; p) respectively. After a scheduling is assumed, the schedule length is defined as max {FT(ni; p)} across all processors.

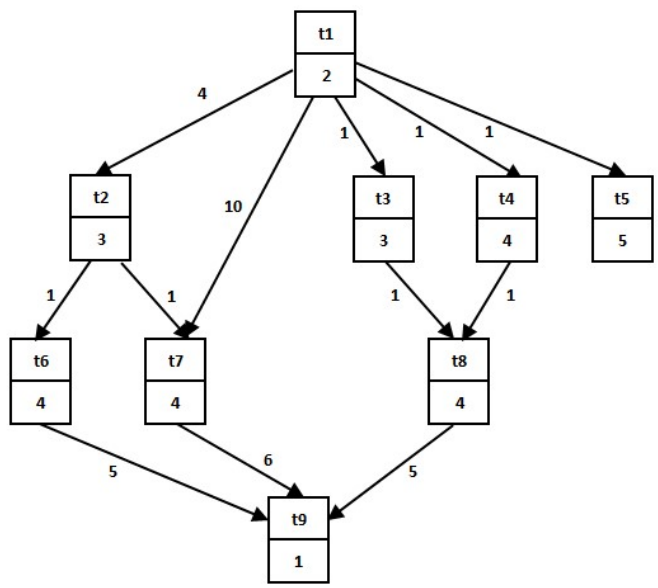


Figure 1. DAC (Homogeneous processors)

Journal of Theoretical and Applied Information Technology
30th June 2019. Vol.97. No 12

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	$t1$	$t2$	$t3$	$t4$	$t5$	$t6$	$t7$	$t8$
$t1$	0	0	0	0	0	0	0	0
$t2$	4	0	0	0	0	0	0	0
$t3$	1	0	0	0	0	0	0	0
$t4$	1	0	0	0	0	0	0	0
$t5$	1	0	0	0	0	0	0	0
$t6$	0	1	0	0	0	0	0	0
$t7$	10	0	0	0	0	0	0	0
$t8$	0	0	1	1	0	0	0	0
$t9$	0	0	0	0	0	5	6	5

Figure 2. Communications Matrix

The task scheduling problem is to find a schedule of the tasks into the processors such that the schedule length is minimized over possible schedules, where the tasks dependency constrains are preserved. Task dependency constrains is stated that, any task cannot start until all parents have finished.

Let pj be the processor on which the kth parent task tk of task ti is scheduled. The Data Arrival Time (DAT) of ti at processor pj is time in which the per-demanded data for the task execution become available, at is defined as in [13] by the following :

$$DAT=\max_{k=1.2.......N\_Parent}\{FT(t\_k.\, p\_j\,) + c(t\_i.\, t\_k\,)\}; \quad (1)$$

where, N_Parent is the number of ti's parents.

$$c(t_i.\, t_k) = 0 \;\; ; \qquad (2)$$

if task i and k are scheduled on the same processor

4. THE PROPOSED GENETIC ALGORITHM (PGA)

The proposed genetic algorithm (PGA) starts with the first population of feasible solutions. Then, by applying some operators, the best solution can be found after some generations. The selection of the best solution is determined according to the value of the objective function. In the suggested algorithm (PGA), we notice that the four components are: (1) an encoding method that is a generic representation (genotype) of solutions to the software. (2) A way to make a primary population of chromosomes, (3) the objective function (4) the genetic operators (crossover and mutation) that affect the genetic makeup of offspring during reproduction.

4.1 Encoding Method

In PGA, the chromosome is divided into two sections; distributing and scheduling sections. The distributing section contains the processors indices where tasks are to be run on it. The schedule part determines the sequence for the processing of tasks. Fig. 3, show an example of a representation of chromosome. Where, tasks t1, t3, t5 will be scheduled on processor p1, tasks t2, t6 will be scheduled on processor p2 , and tasks t4, tm will be scheduled on processor pn. The chromosome length is linearly proportional to the quantity of tasks.

t_1	t_2	t_3	t_4	t_5	t_6	…	t_m							
p_1	p_2	p_1	p_n	p_1	p_2	…	p_n	t_1	t_4	t_3	t_5	t_2	….	t_m

Figure 3. Representation of chromosome

4.2 Initial Population

Subs The initial population is randomly generated according to the following steps:

1. A chromosome X in the initial population is possible to generated as shown in Fig. 3.
2. The first part of X (i.e. distributing) is selected randomly from 1 to N in the system.
3. The second part (i.e. the scheduling) is randomly generated so that the topological order of the graph is conserved.
4. Repeat steps 1 to 3 to generate the number of pop_size of the chromosomes.

4.3 The Objective Function(OF|)

The main objective of scheduling problem is to reduce schedule length. That is:

$$OF=a/(S\_Length) \quad (3)$$

where a is a constant and S_Length is the schedule length which is determined by the following equation:

$$S\_Length=\max_{i=1.2.\cdots\cdots.M}(FT[t\_i]) \quad (4)$$

The pseudo code for the task schedule using SGA,[13] is as follows:

```
For all processor pj RT[pj]=0; j=1,………N.
   For i = 1 to M
   {
     Remove the first task ti form list LT.
     For j = 1 to N
```

```
        {
            If ti is scheduled to processor Pj
                ST[ti]=max{RT[Pj]; DAT(ti; Pj)}
                FT[ti]= ST[ti]+ weight[ti]
                RT[pj] = FT[ti]
            End If
        }
    }
  S_Length=max{FT}
```

4.4 The Genetic Operations

4.4.1 The crossover operation

The crossover process is used to breed the child from two parents via one cut point. The crossover process will perform if the crossover ratio (Pc<=0.95) is verified. The cut point is randomly chosen. The crossover process is performed as follows:

- Randomly select two chromosomes from current population.
- Select randomly the cut point.
- Fill the chromosome components:
  a. Via taking the components from first chromosome (from first gene up to cut point) and filling the child.
  b. Too, taking the components from second chromosome (from cut point+1 up to last gene) and filling the child.

The offspring generated the crossover operation as shown in Fig. 4.

Cut point

Parent	p_1	p_2	p_1	p_n	p_1	p_2	...	p_n	t_1	t_4	t_3	t_5	t_2	....	t_m
child	p_1	p_2	p_1	p_n	p_1	p_2	...	p_n	t_1	t_2	t_4	t_6	t_8	....	t_m
Parent	p_3	p_1	p_2	p_n	p_3	p_1	...	p_n	t_1	t_2	t_4	t_6	t_8	....	t_m

Figure 4. The crossover operation

4.4.2 The mutation operation

The mutation process is conducted on a per-bit basis. The proposed approach in this paper, the mutation process will implement if the mutation ratio (Pm<=0.02) is verified and is randomly estimated. A point is determined to be randomly mutated. Fig. 5, shows the offspring resulting from the operation of the mutation.

child	p_1	p_2	p_1	p_n	p_1	p_2	..	p_n	t_1	t_2	t_4	t_6	t_8	...	t_m
child	p_1	p_2	p_1	p_n	p_3	p_2	..	p_n	t_1	t_2	t_4	t_6	t_8	...	t_m

Figure 5. The mutation operation

5. THE PGA ALGORITHM

The following algorithm clarifies how we can solve the task scheduling problem. That is to find an assignment and the start times of the tasks to processors such that the schedule length is minimized, and in the same time, the precedence constrains are preserved.

Algorithm 1: Genetic algorithm to solve scheduling problem

1. Input: : Set the parameters: Pop_size, Maxgn, Pm, Pc , GN, N, P.
2. Steps:
3. Initial population generation as in section 4.2.
4. gen←1.
5. While (gen <= maxgen) Do
6. P ← 1
7. While (p <= pop_size) Do
8. Genetic processes:
   - Select the chromosome of two maternal populations at random
   - Apply the crossover according to P_c.
   - Mutate of new child according to P_m.
9. For each task i in the schedule section
   - Compute the start time ST[t_i]
   - Compute the Final time FT[t_i]
   - Compute the ready time of the processor p RT[p] which task i is processed on it.
10. Save this child as nominee solution.
11. P ← p+1.
12. End Do
13. Set gen =gen + 1
14. End Do
15. Compare all solutions to get the best solution

6. THE PROPOSED GENETIC ALGORITHM (PGA)

In this section, we show effectiveness of the PGA by applying it on three examples. First example of 9 tasks and three homogeneous processors. Second example, also of 9 tasks and three heterogeneous processors. The third one of 10

Journal of Theoretical and Applied Information Technology
30th June 2019. Vol.97. No 12

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

tasks and three heterogeneous processors. The parameters setting in this algorithm are: pop_size = 20, Pm<= 0.02, Pc<=0.95.

Example 1: We consider an example of 9 tasks {t1, t2, t3, t4, t5, t6, t7, t8, t9} to be executed on three homogeneous processors {p1, p2, p3}, the cost of execution each task on different processors is shown in Fig. 1, and inter task communication cost between the tasks in the form of matrices as shown in Fig. 2, [13].

The best solution which obtained by the PGA as shown in Fig. 6. The schedule length after applying the PGA with Maxgn =100 is 16 as shown in Fig. 7.

p3	p3	p1	p2	p1	p2	p3	p2	p2	t1	t3	t2	t7	t5	t4	t8	t6	t9

Figure 6. The best solution

The schedule length after applying the PGA is 16 as shown in Fig. 7, but the schedule length obtained by SAG, [13] is 17 as Fig. 8.b.

	P1		P2		P3	
	ST	FT	ST	FT	ST	FT
t1	-	-	-	-	0	2
t2	-	-	-	-	2	5
t3	3	6	-	-	-	-
t4	-	-	3	7	-	-
t5	6	11	-	-	-	-
t6	-	-	11	15	-	-
t7	-	-	-	-	5	9
t8	-	-	7	11	-	-
t9	-	-	15	16	-	-

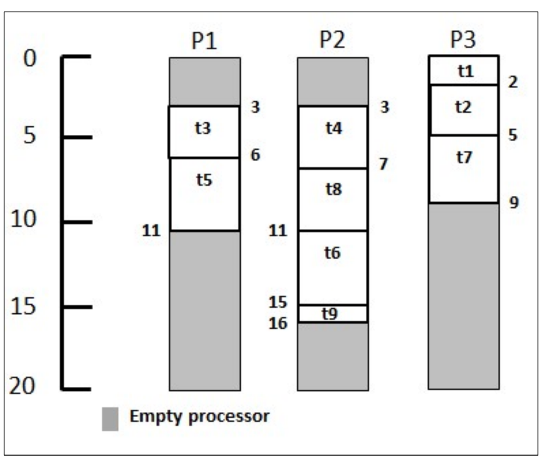


Figure 7. The Scheduling Length of The PGA (HOMOGENEOUS PROCESSORS)

The comparison between the results obtained by PGA and SAG is shown in Fig. 8.a. and Fig. 8.b. It is clearly from the figures; we see the results by PGA are better than SAG.

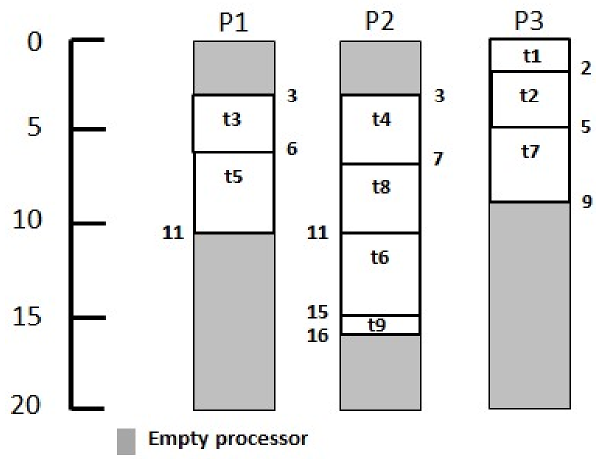


Figure 8.A

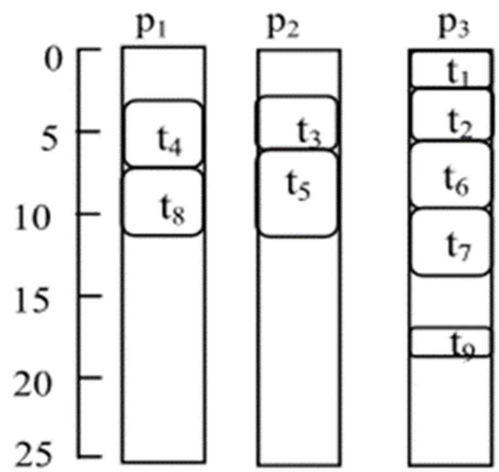


Figure 8.B

Example 2:We consider an example of 9 tasks {t1, t2, t3, t4, t5, t6, t7, t8, t9} to be executed on three heterogeneous processors {p1, p2, p3}, the execution cost of each task on different processors is generated randomly from 1 to 5 and is shown in Fig. 9. And inter task communication cost between the tasks in the form of matrices as shown in Fig. 3.

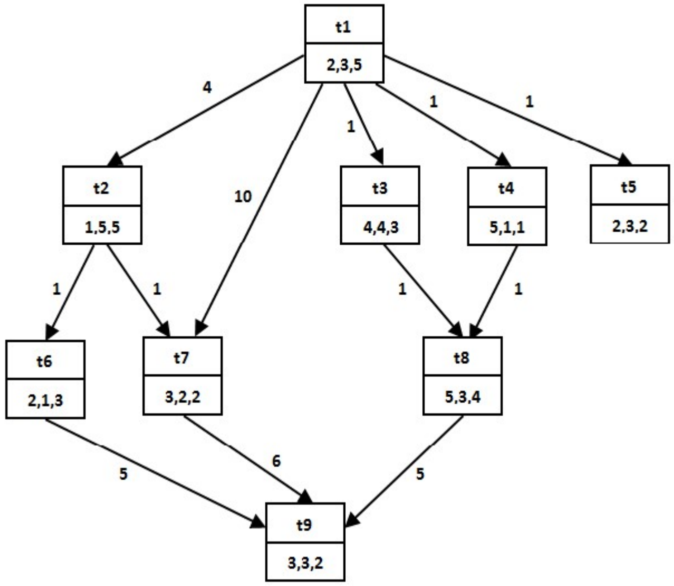


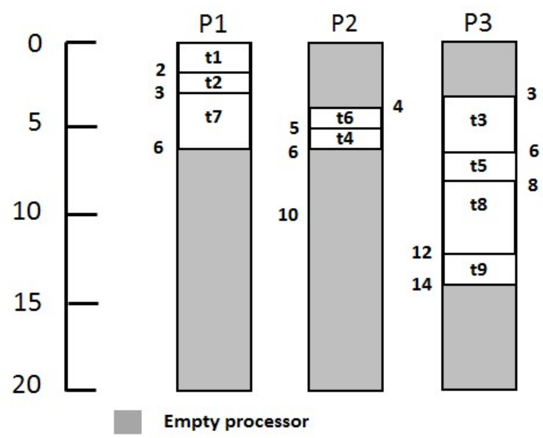
Figure 9. DAC (HETEROGENEOUS PROCESSORS)

The best solution obtained by the PGA with maxgen =100 as shown in Fig.10.

p1	p1	p3	p2	p3	p2	p1	p3	p3	t1	t2	t6	t3	t4	t5	t7	t8	t9

Figure 10. The Best Solution

The schedule length after applying the PGA is 14 as shown Fig. 11.



	P1		P2		P3	
	ST	FT	ST	FT	ST	FT
t1	0	2	-	-	-	-
t2	2	3	-	-	-	-
t3	-	-	-	-	3	6
t4	-	-	5	6	-	-
t5	-	-	-	-	6	8
t6	-	-	4	5	-	-
t7	3	6	-	-	-	-
t8	-	-	-	-	8	12
t9	-	-	-	-	12	14

Figure 11. THE SCHEDULING LENGTH OF THE PGA HETEROGENEOUS PROCESSORS)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The Comparison between the results obtained by PGA (Homogenous and Heterogonous processors) is shown below in Fig. 12.a, and Fig. 12.b. respectively.

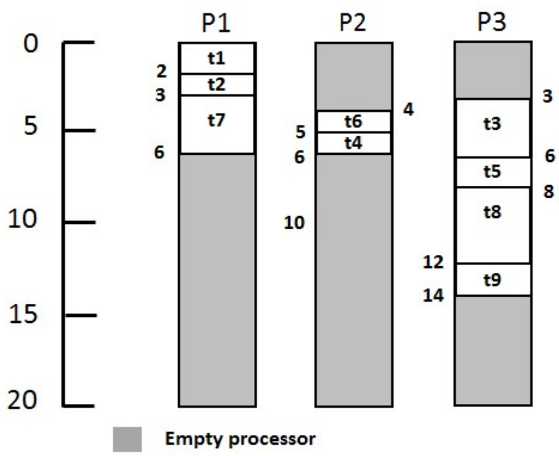


Figure 12. A

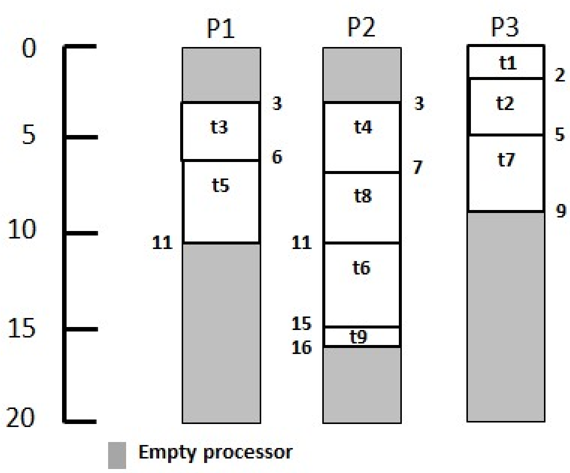


Figure 12. B

Example 3: In this example, the number of tasks {t1, t2, t3, t4, t5, t6, t7, t8, t9, t10} to be executed on three heterogeneous processors {p1, p2, p3}, the cost of execution each task on different processors and inter task communication cost between the tasks in the form of matrices as shown in Fig. 13, [14].

t_i	P1	P2	P3
t_1	22	21	36
t_2	22	18	18
t_3	32	27	43
t_4	7	10	4
t_5	29	27	35
t_6	26	17	24
t_7	14	25	30
t_8	29	23	36
t_9	15	21	8
t_{10}	13	16	33

Figure 13. SAMPLE DAG AND COMPUTATION COST MATRIX, [14]

The best solution obtained by the PGA as shown in the following figure:

p1	p1	p2	p3	p2	p1	p2	p3	p2	t1	t2	t4	t5	t9	t3	t7	t6	t8	t10

Figure 14. THE BEST SOLUTION

The schedule length after applying the PGA with generation =97 is 117 as shown in Fig. 15, but the schedule length obtained by [14] is 125.

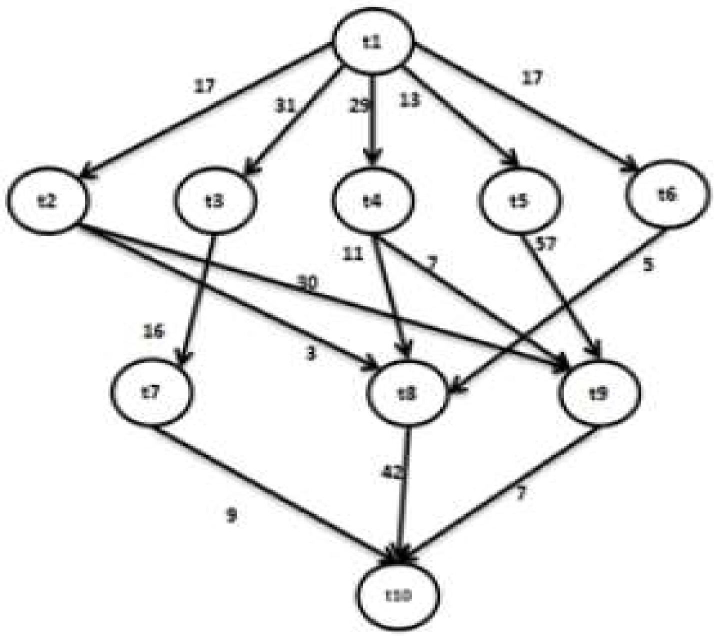


Figure 15. THE SCHEDULING LENGTH OF THE PGA

The Comparison between the results obtained by PGA and [14] is shown below in Fig. 16.a and Fig. 16.b, respectively.

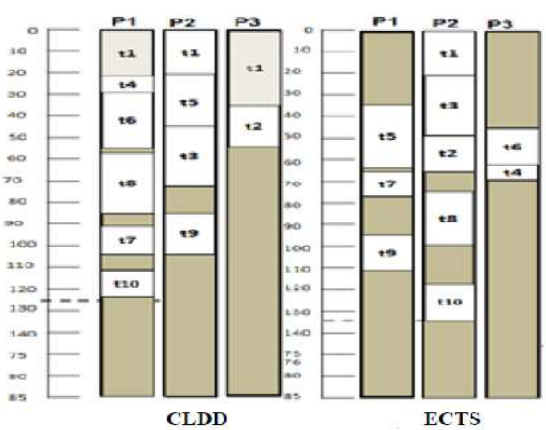


Figure 16. A

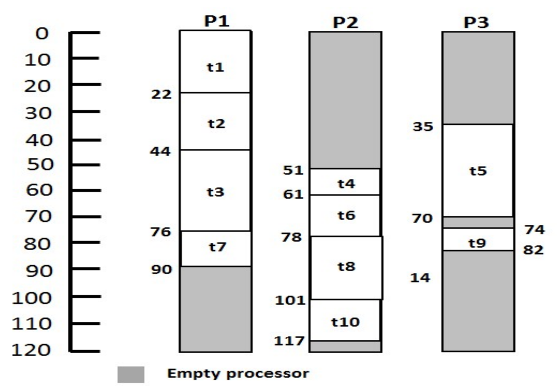


Figure 16. B

7. COMPARISON OF PROPOSED WORK WITH SELECTED ALGORITHMS

In this paper, we tackle the problem of scheduling problem in a general case, where we have multiple processors and heterogenous processors. Now a days, many computations are done in multi-processing systems like multi-core system of personal computers. This paper proposed a fast effect algorithm to solve the secluding problem in multi-processing system. In addition, we considered the multi-processing system in a general form, as in the cloud computing or cluster computing the use multi-nodes with different kind of processors, which can be described as heterogeneous multi processors system. Task scheduling for multi heterogenous processors is a hard problem, all researches are proposed a heuristic algorithm to solve it. In this paper under certain constrains we aim to get a fast-genetic algorithm to solve that problem. The conclusion section will summarize the propose algorithm performance against the other selected algorithm with the same constrains.

Here we will sum up the comparison results, in example 1, we can see that the with the same parameters as in [13] the proposed structure of the genetic algorithm gives more better results. In example 2, we illustrate the need no of iterations for the certain situation. Which shows that the no. of iteration is very efficient, and that is reflect the speed of the proposed Algorithm. In example 3, we consider a system with same parameters as in example shown in [14], and we can see the efficiency of the proposed algorithm.

8. CONCLUSION

In this paper, we presented a genetic algorithm PGA to solve the task scheduling problem in distributed systems. The system was as the two processes. The first process consists of a limited number of fully connected homogeneous processors and the second process consists of a limited number of fully connected heterogeneous processors. We compared the results of PGA and SAG, [13], we founded that the schedule length of PGA was less than the SAG's schedule length. We compared between the results of PGA in the case of homogenous and heterogeneous processors, we founded that the schedule length of PGA heterogeneous processors was less than the PGA homogeneous processors. Also, we compared between the results of PGA and [14] in the case of heterogeneous processors, we founded that the schedule length of PGA heterogeneous processors was less than the PGA heterogeneous processors.

9. FUTURE WORK

The scheduling problem have many versions for different structures and systems. Especially, with the current importance of computations capabilities. In the meantime, the time efficiency of any system is highly demanded, and that is achieved through new algorithms rather than faster hardware. Even for new 5G communication is depend on the software computation efficiency.

In [15], they introduced an new heuristic algorithm based on ant-colony method for important problem structure. In [16] also, the proposed a new algorithm based on OpenCL framework and compare it with the conventional scheduling scheme. Even in [17] they consider the same system under the study but with energy- awareness. Our future work will consider these systems using the proposed genetic algorithm (PGA).

REFRENCES:

[1] Kaiser, T., Jegede, O., Ferens, K. and Buchanan, D. (2015). A Genetic Algorithm for Multiprocessor Task Scheduling. PhD. Dept. of Electrical and Computer Engineering, University of Manitoba,

[2] Hwang, R., Gen, M. and Katayamaa, H. (2006). A Performance Evaluation of Multiprocessor Scheduling with Genetic Algorithm. Asia Pacific Management Review, 11(2), pp.67-72.

[3] Akbari, M. and Rashidi, H. (2015). an efficient algorithm for compile-time task scheduling problem on heterogeneous computing systems. international journal of academic research, 7(1), pp.192-200.

[4] Sharma, A. and Kaur, M. (2015). An Efficient Task Scheduling of Multiprocessor using Genetic Algorithm based on Task Height. International Journal of Hybrid Information Technology, 8(8), pp.83-90.

[5] Dhingra, S., Gupta, S. and Biswas, R. (2014). Genetic Algorithm Parameters Optimization for Bi-Criteria Multiprocessor Task Scheduling Using Design of Experiments. International Journal of Computer, Electrical, Automation, Control and Information Engineering, 8(1), pp.661-665.

[6] Mahi, M. and Kodaz, H. (2014). Genetic Algorithm with Descendants Idea for Scheduling Tasks Graph in the Multi-Processor Architecture. Journal of Advances in Computer Networks, 2(1), pp.10-13.

[7] Mohamed Rashid,, M. And AWADALLA, M. (2011). Hybrid Algorithm for Multiprocessor Task Scheduling. International Journal of Computer Science Issues, 8(3(2), pp.79-88.

[8] Al-Angari, N. and ALAbdullatif, A.. Multiprocessor Scheduling Using Parallel Genetic Algorithm. College of Computer & Information Sciences, King Saud University.

[9] Bohler, M., Moore, F. and Pan, Y. (1999). Improved Multiprocessor Task Scheduling Using Genetic Algorithms. In: American Association for Artificial Intelligence. Proceedings of the Twelfth International FLAIRS Conference.

[10] Heidari, H. and Chalechale, A. (2012). Scheduling in Multiprocessor system using a genetic algorithm. International Journal of Advanced science and technology, 43, pp.81-92
[11] Golub, M. And Kasapovic, S. (n.d.). scheduling multiprocessor tasks with genetic algorithms. University of Zagreb.
[12] E. S. Hou, N. Ansari and H. Ren, "A Genetic Algorithm for Multiprocessor Scheduling," in IEEE Transactions on Parallel and Distributed Systems, 1994
[13] Fatma A. Omaraa, and Mona M. Arafa, "Genetic algorithms for task scheduling problem", J. Parallel Distrib. Comput. 70 (2010) 13_22.
[14] Aida A. Nasr,Nirmeen A. El-Bahnasawy,and Ayman El-Sayed, "Task Scheduling Algorithm for High Performance Heterogeneous Distributed Computing Systems", International Journal of Computer Applications (0975 – 8887) Volume 110 – No. 16, January 2015.
[15] Edward, N., & Elcock, J. (2018). Task Scheduling in Heterogeneous Multiprocessor Environments–An Efficient ACO-Based Approach. Indonesian Journal of Electrical Engineering and Computer Science, 10(1), 320-329.
[16] Fang, J., Yu, T., & Wei, Z. (2019, April). Heterogeneous Multiprocessor Matching Degree Scheduling Algorithm Based on OpenCL Framework. In IOP Conference Series: Materials Science and Engineering (Vol. 490, No. 4, p. 042045). IOP Publishing.
[17] Qin, Y., Zeng, G., Kurachi, R., Matsubara, Y., & Takada, H. (2019). Energy-Aware Task Allocation for Heterogeneous Multiprocessor Systems by Using Integer Linear Programming. Journal of Information Processing, 27, 136-148.