"A Noble Approach for Scheduling of Task Graph Using DAG to Reduce the Overall Makespan"

Prasant Singh Yadav, P.K Yadav, Karunesh Paratp Yadav

Abstract: ***Arrangement of the tasks is the most essential factor to attain equilibrate load, at the distributed computing environment. The arrangement of the task networks is very perilous to the functioning of the multiprocessor computation scheme. It pacts toward the distribution of errands to most appropriate node with maintaining the order of consignment with regard to the sequence of task execution on each node. Here the primary goal is to decrease the overall achievement time or makespan. It's well known that the direct Acyclic Graph (DAG) arrangement problem is NP-complete for this numerous heuristics approaches have been offered. Through this research paper we suggest a noble approach of task Scheduling algorithm for dissimilar processors having different competencies. As we discovered in printed approaches used to allocate a weight or task into the processors ominously alter the presentation of arrangement procedures. We proposed a brand new tactic Performance Effective Task Organization Algorithm (PETOA) for locale the tasks from the order with respective precedence based on Highest Directed Edge Path (HDEP) and following it in accordance with the tasks priorities the selected task will be allocated on these suitably capable processors on the task takes the Minimal Completion Time MCT). The projected method was confirmed on a number of arbitrarily created problems of various dimensions plus types, resulting from better results than earlier methods.***

KEYWORDS: ***TASK SCHEDULING, DAG, MCT, DCS, MCT, PETOA, TASK GRAPH, HDEP.***

I. INTRODUCTION

The linking of computers via communications channels is quite common in today's world, and it has generated many new application areas by facilitating the sharing of software and hardware tools and enabling the immediate transfer of information across distances which range from less than a meter in a single room to millions of kilometers. This type of arrangement allows tasks to be distributed to separate computer where they can be implemented in parallel leading to an escalating number of tasks performed per unit of time. Issues which pertains both the software and hardware design of like the interconnection of processing components can usually be classified as multi-processing systems. It pertain the efficiency to scattered computing where various processors are interrelated in the exact same style to ensure the transferability of control to a process, running on one machine to another machine.

Revised Manuscript Received on June 12, 2019

Prasant Singh Yadav, Ph.D CS Scholar E, Dr APJ AKTU Lucknow India

Dr. Pradeep .Kumar Yadav, principal Scientist CSIR-CBRI Roorkee India

Dr.K.P Yadav, Vice Chancellor, Sangam University, Bhilwara Rajsthan, India

A process defines a collection of modules, which live on distinct processors. The connections between the modules represent the fact that the transferring of control among the modules at more than one point throughout the life time of a program. The processors can also be interconnected by inter processor communication connections. The processor has native memory only and they don't bother about global memory. In such kinds of systems, our purpose is to assign optimally the modules of a program to specific processing unit. Distributed Computing Systems (DCS) consist of processors that communicate with each other only by passing messages and which don't have a common memory. Allocation of task in distributed computing systems is a crucial issue for enhancing the general performance. Task Allocation pertains to the process of assigning tasks to available processors in DCS. The tasks are assigned to the processors in such a manner that a certain measure of efficacy is optimized. This measure of efficacy might be the total execution time of the program, the costs incurred in running the program, or some other quantity with respect to the use of resources. Assigning m tasks to n processors requires m^n exhaustive listings [17]. It must keep in mind to avoid excessive Inter-Processor Communication [IPC] in an efficient task allocation policy in order to exploit the particular efficacies of the processors whereas system having a similar processor, with level best transparency the evenly distribution of task or module is kept. The IPC is the bottleneck in providing linear speed- up with the increase in the number of processors [22].

Optimal tasks allocation to processors is a vital step in harnessing the capabilities of an analogous or distributed computing system, and may be performed in in following style: **Static Allocation** When the assignment of tasks to the processors is done so that after the module is assigned to the processor, it remains there, while the features of the computation are constant. When the attributes of the computation vary, a brand new assignment has to be computed. The characteristic expressions of the enumeration mean the ratio of the times that a program spends in different parts of the program.

Therefore, in static allocation, one is intrigued in finding the assignment replica that holds for the lifetime of a program and results at the best value of the measure of efficacy. **Dynamic Allocation**-to make the best resources use within a distributed system, it's vital to reassign modules during program execution, in order to take benefit of changes in the reference routines of the application [23].

In spite of having impending recital benefits in dynamic distribution the static

distribution is calmer for implementation having comparatively lesser complexity and nearly lesser versatile nature.

II. RELATED WORK

In multiprocessor computing system the arrangement of task graph is really a critical measure of performance. It schemes distribution of particular tasks to appropriate processors to accomplish the common goal to minimalize the total accomplishment time or makespan [7], [14], [27], [29]. It is well known that the DAG task arrangement problem is NP-complete; many heuristics approaches are introduced for this.In general List scheduling heuristics approaches usually create trustworthy schedule with respect to sensible charge. In [30], [32], [33] authors projected different methods to select the most appropriate processor for the nodes with respect to their priorities. For mostly homogeneous systems where speed, capability and network bandwidth between any pair of processors are the same, many List scheduling heuristics are initially considered. This scheduling algorithm has been classified in two ways: firstly numerous dynamic list scheduling algorithms are projected in [29], [30], [3].

In such algorithms the priorities and scheduling of each node is updated on each step and like customary list scheduling algorithms highest priority node is designated for scheduling. In spite of creating better task arrangement Dynamic list approaches can considerably upsurge the time complexity of the algorithms. Secondly In [1], [2], [5] authors projected numerous list Scheduling algorithms with regards to heterogeneous environment. Assessment of such algorithms says that during the appropriate processor selection for the priority task : (1) insertion-based strategy, is far healthier than non-insertion based complements. Where it is allowed to potential inclusion of a task in a first indolent time slit amid two earlier-arranged tasks with respect to a resource, (2)Earliest start time factor was not considered during the Processor selection criteria where different processor speed (e.g., Earliest Completion Time) considered giving outperform. Even though the DAG scheduling, in general, is a planned problem, many authors taken suppositions as processors having similar capabilities, i.e. all the tasks may be executed with different speeds. On the other hand, some authors did not take such suppositions and also not bothered about the impending consequence of diverse competencies [5], [16], [27].

So without any appropriate amendment above algorithms undergo in performance while scheduling at such situations and becomes inappropriate. For example, the Critical-Path-on-a-Processor (CPOP) algorithm projected in [9] all critical tasks are assigned to a single processor in an attempt in order to limits overall execution time for perilous tasks. This algorithm fails no processors can execute all the critical tasks. The dynamic level scheduling DLS algorithm [33] fixes the priority of tasks with respect to f dynamic level attribute. The bottleneck of DLS algorithm not maintaining a scheduling list on scheduling process Another class of algorithms unsuitable are the clustering algorithms [13], [17], [19], [21] in such algorithm cluster are made of different task and assigned them to alike processing unit. Under the status of processors with different capacities, it is unlikely that none of the processors can fulfill all the potentially enormous numeral of tasks in the identical cluster. As a result, the clustering algorithm is bound not to be used directly under these conditions except it is efficiently revised.

Through this paper we proposed a new list scheduling algorithm for heterogeneous processors, having different capabilities. As projected in [34], adopted to allocate weights to the nodes considerably mark the performance of scheduling algorithms. Here we projected a new method Performance **Effective Task Organization Algorithm (PETOA)** for setting the tasks in the sequence with their priories based on highest directed edge weight and after it according to the tasks priorities the selected task will be assigned on those suitably capable processors on which the task takes the minimum completion time (MCT). The algorithm has been experienced on a hefty numeral of arbitrarily created problems of diverse dimensions and sorts.

We equate PETOA with former list arrangement algorithms; the Dynamic level scheduling algorithm (DLS), Levelized Min Time (LMT) which are projected in the literature in respect to heterogeneous distributed computing systems. The intricacy of DLS, LMT, algorithms is O (m^3n^2), O (m^2n^2) respectively. Here we selected some recently projected algorithms for upgrading even numerous performance measures such as the tardiness or the total run time are recommended in the literature [16]. Here our projected aim in this research is to minimize the overall scheduling length L or Makespan.

III. PROBLEM

The DAG is a common model of a task stream application containing of a set of tasks (nodes) amongst which superiority constraints exist. It is represented by G = (V, E), where V is the set of m tasks T= {t_1, t_2, t_3… t_m} that can be executed on a set of the available processors P= {p_1, p_2,……,p_n}. E is the set of directed arcs connecting the tasks that preserve a partial arrangement among them. The partial order introduces precedence constraints, i.e. if edge e_{ik}, then task t_k cannot start its execution before t_i completes. Matrix C of size m × m denotes the communication cost, where c_{ik} is the amount of communication cost to be transferred from t_i to t_k. A task graph is a weighted graph. The weight w_i of a node t_i usually represents its computation cost (execution cost) The weight of an edge stands for the communication required between the connected tasks (the amount of data that must be communicated between them). We introduce a new approach to assign node execution cost in Section 5. In a given task graph, a root node is called an entry task and a leaf node is called an exit task. Here we take up that the task graph is a single-entry and single-exit one. If this is not followed, we can always link them to a zero-cost quasi exit or entry task with zero-cost edges, resulting in will not affect the schedule.

A resource graph is an undirected weighted graph having weighted nodes and edges. Processor of a resource graph is represented by a node and the link between a pair of connected processors is denoted by an edge. The resource graph is a complete

graph containing n completely node tasks. Weight or execution cost for a node signifies the processor computation capability (per unit time computation performed by the processor). Likewise, the weight of an edge stands for its communication cost (per unit time data transmission through the link). Here we take suppositions that all inter-processor communications are executed with no disputation.

IV ASSUMPTION AND DEFINITION

A few suppositions are considered here to preserve the Algorithm sensible in size when structuring it.

a) "m" tasks, are to be executed on "n" processors which have diverse preparing capacities.

b) A task might be a segment of an executable code or an information document.

c) The quantity of tasks to be dispensed is more than the quantity of processors (m>>n), is the general situation in the DCS figuring condition.

d) Each task's execution time for each processor is already acknowledged.

e) On the off chance that an task isn't executable on any of the processor because of nonappearance of certain assets, execution time of same assignment on particular processor is assumed infinite (∞) interminable.

f) We expect that after finishing task execution, the processor stores the obtained information for executed task local memory, if the information is required by another task residing on a similar processor; it peruses that information from that local memory.

g) The communiqué structure of the processors is without crash, in this way all messages are sent in a determinate degree time unit without any loss.

h) We accept correspondence without any conflict for the processors. A processor can communicate with another processor simultaneously while executing a task.

i) The overhead brought about by this is unimportant, so for every single useful reason we will think about it as zero. Utilizing this reality, the algorithm attempts to allot the intensely imparting task to a similar processor.

j) Cluster of task residing on identical processor have inter task communication time (ITCT) is zero.

Execution Cost: The execution cost e_{ij} of the assignment (node) ti relies upon the measure of computation cost designate accomplished by each task t_i on processors p_j, j=1, 2… n. Since the framework is heterogeneous in this way execution cost of every task t_i on every one of the processors are different.

Inter-task Communication Cost: The between task correspondence cost c_{ik} of the cooperating ancestor task t_i to successor t_k is acquired because of the information units traded between them amid the procedure of execution on a DAG.

IV. PROPOSED METHOD AND ALGORITHM

The present model has two main parts; first a task ordering phase, second is processor determination phase. The first stage calculates urgencies of all the tasks other phase select the tasks in mandate with accordance to urgencies and allots every nominated task to optima suitable processor in order to minimization of the task consummation period and finally the makespan is the scheduling span L of the authentic completed time of the exodus task t_{exit}. On the DAG L = FT (m_{exit}).

First we structure the precedence of task for their execution which depends on the ordering of the task in descending order of their Highest Directed Edge Path (HDEP) is determined by HDEP (ti) = {Highest total of correspondence cost of edges barring the execution cost on DAG from node t_i to the exit node}; i =1,2,3,… … m.

Stock the task order in a line array $\{TS\}_{non\text{-}ass}(\,)$ which is defined the priorities of task. Select first tasks with highest priority i.e. entry node form $\{TS\}_{non\text{-}ass}(\,)$ say t_i and for the entry node the earliest started time (EST) is defined as:

$$EST(t_i) = 0, pred(i) = \Phi \qquad (1)$$

And, the MCT of task t_i on the processors state Pj are given by

$$MCT(t_i,p_j) = EST(t_i,p_j) + w_{ij} \qquad (2)$$

The undertaking ti is doled out on that processor Pj on which finishing time is least and store the errand ti in {TS}ass( ) and erase it from {TS}non-ass( ) likewise store the processor position in . Select the new assignment with most noteworthy need from {TS} non-ass ( ) state t_k .The EST of errand t_k on processors ps are determined as

$$EST(t_k, p_s) = \max[avail(t_k, p_s), \max_{t_i \,\varepsilon\, pred(t_k)}\{EST(t_i, p_j) + c(t_i, t_k)\}] \qquad (3)$$

The predecessor (t_k) is the arrangement of quick forerunner assignments of task t_i, and avail (t_i, P_s} is most punctual period for which processor p_k is prepared for task accomplishment. The MCT of errand tk on processors Ps are determined as

$$MCT(t_k, p_s) = EST(t_k, p_s) + w_{k,s} \qquad (4)$$

And assigned it on that processor on which it's culmination time is least and store the assignment tk in {TS}ass( ) and erased from {TS}non-ass( ), store the processor position in Aalloc( ). This procedure proceeds until $\{TS\}_{non\text{-}ass}(\,) = \{\Phi\}$. In this way all assignments in task graph are premeditated, the schedule span (e.g. the overall accomplishment time) is the genuine completion time of the departure task; at long last the schedule length is characterized as Schedule Length = Max {MCT (t_k); k = 1, 2 . . . m}; The algorithm is depicted beneath:

Algorithm:

Step1: Input: m task and n processor (m>>n)

Step2: HDEP ( ) ← 0 // Linear array to store the highest sum of Communication cost of directed edges excluding the execution cost on DAG from node i to the exit node //

$\{TS\}_{non\text{-}ass}(\,)$ ← // Lined array for storing the non-assigned m tasks with their decreasing priorities //.

$\{TS\}_{ass}$ ( ) ← 0 // Lined for storing the assigned tasks //

A_{alloc} ( ) ← 0 // Linear array to store the set of all allocated tasks in the form $\{(t_i, P_j)$ /A (i) = j, if task ti is allotted to processor P_j, $1 \leq i \leq m$, $1 \leq j \leq n\}$ //

Step3: Compute HDEP {i}: {i =1, 2, .3……...m}

HDEP (i) = {largest sum of communication cost directed edges excluding the execution cost on DAG as of node i to the exit node corresponding with task (t_1, t_2, t_3 ……t_m)

Step4: for i← 1to m

Arrange the tasks in descendant direction of their HDEP (i): after it, assign the priority to the task in decreasing order then store the m tasks in $\{TS\}_{non\text{-}ass}$

Step5: Assigning tasks to processors are given in the following steps

Begin,

$P_j \leftarrow 0$, j = 1, 2……n;

EST (t_i) ← 0, i = 1, 2 …… m;

MCT (t_i) ← 0, i = 1, 2 …… m;

{TS} non-ass← {store the non-assigned m tasks with their decreasing priorities}

$\{TS\}_{ass}$←0, i = 1, 2. . . ……m;

STEP5.1: Select the highest priority node i.e. entry node ti from the $\{TS\}_{non\text{-}ass}$ ( ).For entry node the earliest started time on each processor is defined as:

EST (t_i, p_j) = 0, pred (i) = Φ and the MCT of task t_i on any processor p_j $1 \leq j \leq n$ are given by

MCT (t_i, p_j) = EST (t_i, p_j) + w_{ij}

The task t_i is allocated on that processor Pj on which the earliest starting time EST is minimum. If Minimum completion time MCT is same for two or more processors then assign it on any processor randomly, After assigning store it in $\{TS\}_{ass}$ and deleted from {TS}non-ass.

STEP 5.2 Select the new task with highest priority from $\{TS\}_{non\text{-}ass}$ ( ) say it tk and calculate the EST on the processors p_s as

EST (t_k, p_s) = max [avail (t_k, p_s), max{EST (t_i, p_j) + $t_i \varepsilon$ pred(t_k)
c (t_i, t_k)}]

And MCT of task t_k on processors p_s are calculated as MCT (t_k, p_s) = EST (t_k, p_s) + w_{ks}

The task t_k is assigned on the processor p_s, MCT is minimum.

$\{TS\}_{ass} \leftarrow \{TS\}$ass ∪ $\{t_k\}$ And deleted it from $\{TS\}_{non\text{-}ass}$( ).

Step6: Repeat the step5 until $\{TS\}_{non\text{-}ass}$( ) = {Φ}

Step7: output: schedule $\{TS\}_{ass}$

Step8: makespan ← max {MCT (t_k); k = 1, 2 . . . m};

Step9: End.

V. RESULT AND DISCUSSION

Example: For justification of the application and usefulness of projected algorithm we apply it to the example taken from the literature. Here we taken a heterogeneous distributed computing systems of having 10{m=10} executable tasks represented by set of T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}} which can be slices of a program or a data wallet and consisting of group of "n = 3" processors P = {p_1, p_2, p_3} connected by an arbitrary network.Figure1 and table 1 depicted intertask communiqué cost amid the predecessor task t_i and its successor task t_k and execution cost respectively.

Firstly we want to schedule m=10 tasks and assign the priority by using the HDEP for the simple DAG as shown in Fig 1. Now for organizing the tasks in decreasing direction of the Highest sum of communication cost of edges excluding the execution cost on DAG from node i to the exit node, firstly we tried to find foremost schedule t_1 which is the just qualified task for this place because it has the highest directed edge path, we assign the 1st priority to it and store in the $\{TS\}_{non\text{-}ass}$. Next, from the remaining unscheduled tasks { t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}}, now task t_2 is most eligible node for scheduling because it has the highest HDEP in remaining unscheduled task so assign the 2nd priority to it , put in the task sequence {TS}non-ass.

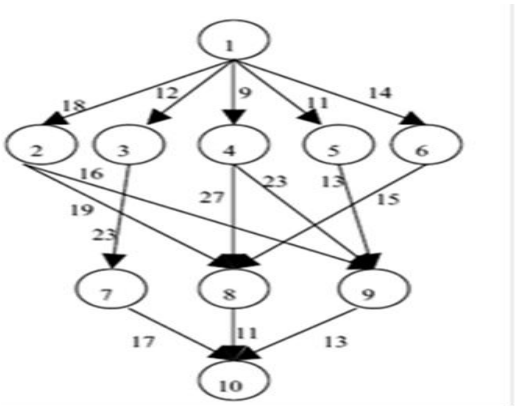


Figure 1: DAG (Direct Acyclic Graph)

Computation Cost Matrix =

	p_1	p_2	p_3
t_1	14	16	9
t_2	13	19	18
t_3	11	13	19
t_4	13	9	17
t_5	11	13	12
t_6	13	16	9
t_7	8	15	12
t_8	5	11	14
t_9	18	12	20

Table 1: Computation Cost Matrix

In similar way we takes steps 3 and 4 until all the remaining unscheduled tasks are

completely scheduled to {TS}non-ass and finally we get the task sequence find in the form of {TS}non-ass={{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}} }with their deceasing priorities this is presented in following table 2.

Entry node	HDEP of node	Max of HDEP of node	Priority of node	$\{TS\}_{non-ass}$
$\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$	$\{t_1=52,t_2=30,t_3=40,t_4=38,t_5=26,t_6=26, t_7=17,t_8=11,t_9=13,t_{10}=0\}$	$\{t_1=52\}$	1	$\{t_1\}$
$\{t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10}\}$	$\{t_2=30,t_3=40,t_4=38,t_5=26,t_6=26,t_7=17, t_8=11,t_9=13,t_{10}=0\}$	$\{t_3=40\}$	2	$\{t_1,t_3\}$
$\{t_2,t_4,t_5,t_6,t_7,t_8,t_9,t_{10}\}$	$\{t_2=30,t_4=38,t_5=26,t_6=26,t_7=17,t_8=11, t_9=13,t_{10}=0\}$	$\{t_4=38\}$	3	$\{t_1,t_3,t_4\}$
$\{t_2,t_5,t_6,t_7,t_8,t_9,t_{10}\}$	$\{t_2=30,t_5=26,t_6=26,t_7=17,t_8=11,t_9=13, t_{10}=0\}$	$\{t_2=30\}$	4	$\{t_1,t_3,t_4,t_2\}$
$\{t_5,t_6,t_7,t_8,t_9,t_{10}\}$	$\{t_5=26,t_6=26,t_7=17,t_8=11,t_9=13,t_{10}=0\}$	$\{t_5=26, t_6=26\}$	5	$\{t_1,t_3,t_4,t_2,t_5\}$
$\{t_6,t_7,t_8,t_9,t_{10}\}$	$\{t_6=26,t_7=17,t_8=11,t_9=13,t_{10}=0\}$	$\{t_6=26\}$	6	$\{t_1, t_3, t_4, t_2, t_5, t_6\}$
$\{t_7,t_8,t_9,t_{10}\}$	$\{t6=_{26},t_7=17,t_8=11,t_9=13,t_{10}=0\}$	$\{t_7=17\}$	7	$\{t_1,t_3,t_4,t_2,t_5, t_6,t_7\}$
$\{t_8,t_9,t_{10}\}$	$\{t_8=11,t_9=13,t_{10}=0\}$	$\{t_9=13\}$	8	$\{t_1,t_3,t_4,t_2,t_5,t_6,t_7,t_9\}$
$\{t_8,t_{10}\}$	$\{t_8=11,t_{10}=0\}$	$\{t_8=11\}$	9	$\{t_1,t_3,t_4,t_2,t_5,t_6,t_7,t_9,t_8\}$
$\{t_{10}\}$	$\{t_{10}=0\}$	$\{t_{10}=0\}$	10	$\{t_1,t_3,t_4,t_2,t_5, t_6,t_7,t_9,t_8,t_{10}\}$

Table 2: A task sequence with ordered and their decreasing priorities

According to further step, we allocate the tasks to the most suitable processors from the above maintained task order $\{TS\}_{non-ass}$ according their priority. In the example, just three processors are used, Now select the first task t_1 i.e. entry node from {TS} non-ass and calculate the EST of the task t_1 is given by

EST (t_1, p_j) = 0, pred (t_1) = Φ and j =1, 2, 3

I.e. EST (t_1, p_1) = 0 and EST (t_1, p_2) = 0,

Next we calculate the MCT of task t_1 on processors pj,

Where j=1, 2, 3….

MCT (t_i, p_j); MCT (t_1, p_1) = EST (t_1, p_1) + $w_{1,1}$ = 0 + 14 = 14.

MCT (t_1, p_2) = EST (t_1, p_2) + $w_{1,2}$ = 0 + 16 = 16

MCT (t_1, p_3) = EST (t_1, p_3) + $w_{1,3}$ = 0 + 9 = 9

Now the task t_1 is assigned on the processor p_3 because the task t_1 has minimum MCT on p_3.after assigning the task t_1 on processor p_3 we store $\{TS\}_{ass}$ ={t_1}and delete it from the task sequence $\{TS\}_{non-ass}$.

Next we select the task with new highest priority i.e. with 2nd priority, from {TS} non-ass which. t_3 and calculate the EST of task t_5 on the processors p_s , s =1,2,3 as defined in step 5.2

$$EST(t_k, p_s) = \max [avail(t_k, p_s), \max\{EST(t_i, p_j) + c(t_i, t_k)\}] \quad t_i \varepsilon pred(t_k)$$

i.e. EST(t_3, p_1) = max [avail{ t_3, p_1}, max{MCT(t_1, p_3) + c(t_1, t_5)}] =max [0, max {9 + 12}] = 21

And MCT (t_k, p_s) = EST (t_k, p_s) + $w_{k,s}$

MCT (t_3, p_1) = EST (t_3, p_1) + $w_{3,1}$ = 21 + 11 =32

EST (t_3, p_2) = max [avail {t_3, p_2}, max {MCT (t_1, p_3) + c (t_1, t_5)}] =max [0, max {9 + 12}] = 21

And MCT (t_3, p_2) = EST (t_3, p_1) + $w_{3,2}$ = 21 + 13 =34

EST (t_3, P_3) = max [avail {t_3, p_3}, max {MCT (t_1, p_3) + c (t_1, t_5)}] =max [9, max {9 + 0}] = 9

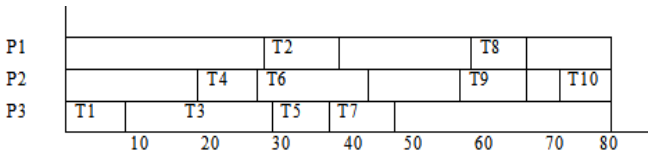
And MCT (t_3, p_3) = EST (t_3, p_3) + $w_{3,3}$ = 9 + 19 = 28

Now task t_3 is assigned on that processor p_s, s=1, 2, 3 on which it has the minimum MCT.

Thus task t_5 is assigned on p_1. {TS} ass ← {TS} ass ∪ {t_3} and deleted it from {TS} non-ass ( ).

In a similar way, we repeat the step 5 calculate the EST and MCT for the unscheduled remaining task {TS} non-ass on all the processors p_j, j=1, 2, 3 i.e. {TS} non-ass. = {Φ}

and the assigned tasks on the processors are stored in the task sequence {TS}ass={t_1,t_3,t_4,t_2,t_5,t_6,t_7,t_9,t_8,t_{10}}.Which is represented by beneath fig 2 Gantt chart.



Gantt chart 1: Obtained Task scheduling Length

The accompanying table speaks the allocation of all the tasks on the processors-Table 4 demonstrates that, makespan ← max {MCT (t_{10})} =80; for example the makespan of the task assignments schedule is characterized as the time it takings from the moment the principal task starts execution to the moment where the last task finishes execution is 80 units of time.

Priority Task	Entry Task	p_1		p_2		p_3		Predecessors	Processor
		EST	MCT	EST	MCT	EST	MCT		
1	t_1	0	14	0	16	0	9	Null	p_3
2	t_2	21	32	21	34	9	28	t_1	p_3
3	t_3	18	31	18	26	28	45	t_1	p_2
4	t_4	27	40	27	46	28	46	t_1	p_1
5	t_5	40	52	26	39	28	38	t_1	p_3
6	t_6	40	53	26	42	38	47	t_1	p_2
7	t_7	51	58	51	66	38	49	t_3	p_3
8	t_8	51	69	56	68	56	76	t_2, t_4, t_5	p_2
9	t_9	57	62	68	79	59	73	t_2, t_4, t_6	p_1
10	t_{10}	81	102	73	80	81	97	t_7, t_8, t_9	p_2

Table 3: The Showing EST, MCT

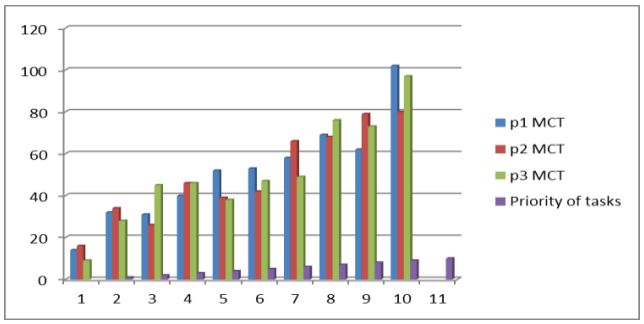


Fig 2: (a) Comparison of MCT values on Processors p_1, p_2, and p_3 for the tasks Priorities

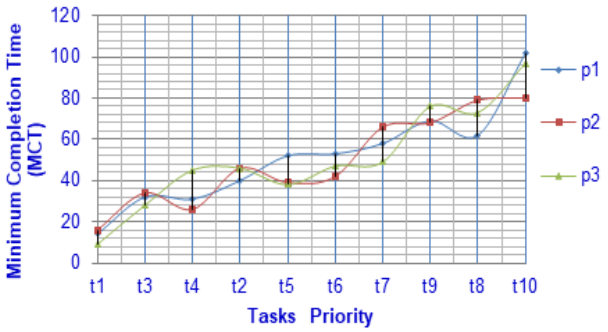


Fig: 2 (b) line chart Comparison of MCT values

VI CONCLUSION AND FUTURE WORK

Comparison table present the comparative result between the present algorithm and earlier published task scheduling algorithm on DAG. The time Complexity of PETOA algorithm is represented by O(m+e), where m is the numeral of task and e is the numeral of edges in the task graph. Configure PETOA algorithm sequence takes $O(m^2)$ time choose the processor takes O(n). Hence this algorithm has time complexity $O(m^2n)$ and by comparison we found that the complexity of PETOA is lesser than the earlier published scheduling algorithms. We projected here new list scheduling algorithm based on HDEP and offered simulation results to show higher efficiency or less minimal completion time in compare to earliest algorithms. Hence our proposed algorithm has less complexity than other projected algorithms. The comparison of our projected algorithm with earlier published algorithm is shown by table 4 and fig 3(a), fig 3(b) with respect to complexity and run time.

Algorithm	Complexity	Processor	Run time
LMT	$O(m^2n^2)$	3	95
DLS	$O(m^3n)$	3	91
PETOA (Present Algorithm)	$O(m^2n)$	3	80

Table 4: Comparison of Complexity Algorithms

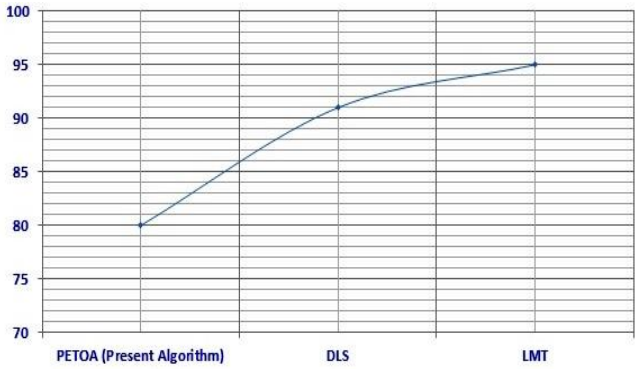


Fig 3: (a) Comparison of Algorithms with respect to Run time

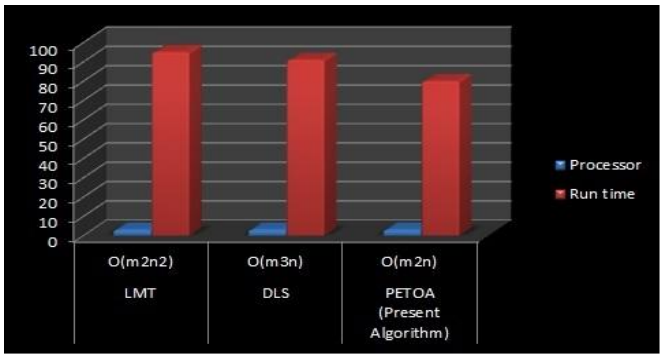


Fig 3: (b) Comparison of algorithm with respect to Complexity and Runtime

Further we can consider task breeding on the processor in an iterative way. The primary thought is to copy task assignments on various machines with the goal that the consequences of the copied undertakings are accessible on different machines to calculate the computation time for communication time. By doing this the inter-task communiqué cost of processor can be minimized. In the future, we will utilize dynamic strength dealing with the technique for task scheduling. A task at that point must be executed on the doled out processors regardless of whether a positive rescheduling is conceivable on another processor. To stay away from horrible pre-duty, planning choices ought to be submitted as late as could be expected under the circumstances, i.e., undertakings ought to be progressively allotted to sit processors that require task assignments as indicated by the genuine execution advance. From scheduling list if processor has no task to running such processor is known as inactive processor.

REFERENCES

1. Singh M.P, P.K.Yadav and Singh Jugmendra,"Optimal utilization of processors in computer

communication environment" National Conference on Modern Development in Engineering & Applied Seiences (NCMDES-09) at College of Engineering &Applied Research, Feb 27-28, 2009.
2. Ahmed Younes. Hamed ,"Task Allocation for Minimizing Cost of Distributed Computing Systems Using Genetic Algorithms" International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 2, Issue 9, September 2012 ISSN: 2277 128X
3. PankajSaxena , Dr. Kapil Govil ,"Cost Optimization for Allocation of Tasks to Processors in a Distributed Processing System", International Journal of Electronics and Computer Science Engineering ,ISSN-2277-1956 ISSN 2277-1956/V2N2-752-762
4. P.K. Yadav, Preet Pal Singh, P. Pradhan,,"A Tasks Allocation Algorithm for Optimum Utilization of Processor's in Heterogeneous Distributing Computing Systems ",IJRREST: International Journal of Research Review in Engineering Science and Technology (ISSN 2278-6643) Volume-2 Issue-1, March 2013.
5. Abhilasha Sharma, Surbhi Gupta , "An Optimal Approach For The Tasks Allocation Based On The Fusion Of Ec And Itcc In The Distributed Computing Systems:, American International Journal Of Research In Science, Technology, Engineering & Mathematics, ISSN (Print): 2328-3491, ISSN (Online): 2328-3580, ISSN (CD-ROM): 2328-3629, 2014.
6. Ruchi Gupta, P.K Yadav, "Mathematical modeling of load Distribution problem in distributed Computing Enviornment-A State of Art." International Journal of Advance Research in Computer Science Software Engineering,ISSN 2277-128X, Volume $ Issue 7 page 1106-1119, July 2014.
7. Sagar Gulati, K. Bhatia, P. K. Yadav "Reliability Optimization for Distributed Systems Through Task Clustering", 2015 Fifth International Conference on Advanced Computing & Communication Technologies ,IEEE 2015.
8. Sagar Gulati, K. Bhatia and P. K. Yadav "Execution Time and Failure Rate based model for Reliability Optimization in Distributed Systems" IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501 Vol. 4, No. 5, October 2014.
9. Harendra Kumar "A Heuristic Model for Tasks Scheduling in Heterogeneous Distributed Real Time System under Fuzzy Environment" International Journal of Computer Applications (0975 – 8887) Volume 111 – No 2, February 2015.
10. Aida A. NASR, Nirmeen A. EL-BAHNASAWY, Ayman EL-SAYED "Performance Enhancement of Scheduling Algorithm in Heterogeneous Distributed Computing Systems" (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 6, No. 5, 2015.
11. Azita Jooyayeshendi, Abbas Akkasi "Genetic Algorithm for Task Scheduling in Heterogeneous Distributed Computing System" International Journal of Scientific & Engineering Research, Volume 6, Issue 7, July-2015 1338 ISSN 2229-5518.
12. Suman Joshi, Gagangeet Singh "Overloading in Distributed Computing System- A Review" International Journal of Computer Science and Mobile Computing, ISSN 2320–088X ,IJCSMC, Vol. 4, Issue. 6, June 2015, pg.601 – 605.
13. Aida A. Nasr, Nirmeen A. El-Bahnasawy, 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.
14. Harendra Kumar, "A heuristic Model for Task Scheduling in Heterogeneous Distributed Real Time System under Fuzzy Logic", International Journal of Computer Application (0975-8887) Volume 111 No 2 February, 2015, pp. 35-43.
15. Suejb memeti, Sabri Pllana, "Combinatorial Optimization of Work Distribution on Heterogeneous System", In IEEE International Conference ICPPW June 2016, pp 39-50.
16. Chetna R, Neelakantappa BB, Dr. Ramesh B, "Survey on Adaptive task Assignment in Heterogeneous Hadoop Cluster", IEAE International Journal of Engineering, Volume 1 Issue 1, July 2016, pp 24-28.
17. P. Neelakantan* and S. Sreekanth, "Task allocation in distributed systems", Indian Journal of Science and Technology, Vol 9(31), DOI: 10.17485/ijst/2016/v9i31/89615, August 2016.
18. Guozeng Zhao1, Xiang Gao1, Weidong Zheng1,2 and Zhiguo , "A Novel Optimization Strategy for Job Scheduling based on Double Hierarchy" Journal of Engineering Science and Technology Review, Volume 10 issue 1, February 2017 pp 61-67.
19. Seyedeh Leili Mirtaheri, Seyed Arman Fatemi, Lucio Grandinetti, "Adaptive Load Balancing Dashboard in Dynamic Distributed Systems", This paper is published with open access at SuperFri.org in Dec 2017, Volume 4 issue 4, pp 34-49.
20. Promila Ahuja, Preeti Sethi, Dimple Juneja, Saurabh Mukherjee, "Task Allocation Strategy for Multi Agent: A Review", Journal of Network Communications and Emerging Technologies (JNCET), Volume 7, Issue 1, January 2017, pp 18-21.
21. Kun He, Member, IEEE, Xiaozhu Meng, "A Novel Task-Duplication based DAG Scheduling Algorithm for
22. Heterogeneous Environments", IEEE Transactions on Parallel and Distributed System, VOL. 14, NO. 8, AUGUST 2018, pp 01-13.
23. Menglan Hu, Jun Luo, Member, IEEE, Yang Wang, and Bharadwaj Veeravalli "Adaptive Scheduling of Task Graphs with Dynamic Resilience" IEEE Transactions on on Computer, VOL. 10, NO. 6, July 2018, pp 224-230.
24. Corbalan J, Martorell X, Labarta J. Performance-driven processor allocation. IEEE Transactions on Parallel and Distributed Systems 2005; 16(7):599–611.
25. Hwok Y, Ahmad I (1999) Static scheduling algorithms for allocating directed task graphs to multiprocessors.ACM Comput Surv 31(4):407–471.
26. Braun T, Siegel H, Beck N et al (1999) a comparison studies of static mapping heuristics for a class of meta-tasks on heterogeneous computing system. In: Eighth heterogeneous computing workshop (HWC), pp 15–23.
27. Davidovic T, Crainic T (2006) Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems. Compute Open Res 33:2155–2177.
28. Ibarra O, Kim C (1977) Heuristic algorithms for scheduling independent tasks on non-identical processors. J Assoc Comput Mach 24(2):280–289
29. Djordjevic G, Tosic M (1996) A heuristic for scheduling task graphs with communication delays onto multiprocessors. Parallel Comput 22(9):1197–1214.
30. Borriello G, Miles D (1994) Task scheduling for real-time multi-processor simulations real-time operating systems and software. In: Proceedings of RTOSS '94, 11th IEEE workshop, pp 70–73.
31. Onbasioglu E, Ozdamar L (2003) Optimization of data distribution and processor allocation problem using simulated annealing. J Supercomput 25(3):237–253
32. Porto S, Ribeiro C (1995) "A tabu search approach to task scheduling on heterogeneous processors under precedence constraints". Int J High-Speed Computation 7(1):45–71.
33. G. C. Sih and E. A. Lee, "Compile time scheduling heuristic for interconnection–constrained heterogeneous processor architecture," IEEE Transaction on Parallel and Distributed System, vol. 4, no. 2, pp. 75-87, Feb. 1993.

AUTHORS PROFILE

Prasant Singh Yadav `completed his bachelor of Engineering in Computer Science & Engineering from Dr. B.R Ambedkar University Agra in 2005, Master of Engineering from NITTTR (an Institute of national importance) Chandigarh Punjab University Chandigarh (U.T) India in the year 2011, and currently Ph.D. Scholar in Computer in Computer science & Engineering at Dr. A P J AKTU Lucknow, UP, India Having more than 12 year academic experience. He is the author of more than 65 research papers published in Different national and international journals, conferences and proceedings. He has been member of more than 12 international / national Educational Society.

Dr. Pradeep kumar Yadav, Currently working as Principal Technical Officer CSIR-CBRI, Roorkee & Associate Professor, Mathematical and Information Sciences at (AcSIR), completed his M.Sc mathematics in 1983 from Grukul kangri university hardwar U.P India, received his Ph.D. in 1996 in Distributed Computing System Grukul kangri university hardwar U.K India Uttar Pradesh, India; He has more than 20+ years of work experience in teaching 10+ experience in R&D. He has been published more than 69 technical research papers in numerous international journal and conferences; He also more than 42 research papers published in various

National Journals, conferences proceedings. He has been Life Member of International Academy of Physical Sciences; Indian Society for Wind Engineering, Computer Society of India (CSI) and has is awarded many research awards and working with many research project grants in India.

Dr. K. P Yadav, Currently working as Vice chancellor SANGAM University Bhilwara, Rajasthan, India., completed his B.E CSE in 1996 from KNIT Sultanpur and M.Tech BITs Mesra, Ranchi, India , obtained his Ph.D. in 2009 from Corllins University, CA (USA), and Ph.D from BIET Jhansi/Bhagwant University Ajmer in 2012 India; He obtained Doctor of Science (D. Sc.) from MNIT Jaipur/ OPJS university Churu Rajsthan India. He is having vast experience of 24+ years. He has been published more than 85 technical research papers in various international journal and conferences; He also more than 33 exploration papers published in various National Journals. He has written more than 12 books in engineering in reputed publication of India.