IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. II (Sep – Oct. 2014), PP 10-18
www.iosrjournals.org

Task Allocation in heterogeneous Distributed Real Time System for Optimal Utilization of Processor's Capacity

Shipra Singh* and M. L. Garg
Department of Computer Science, DIT University, Dehradun, India

Abstract: *In Distributed Real Time System (DRTS), systematic allocation of the tasks among processors is one of the important parameter, which determine the optimal utilization of available resources. If this step is not performed properly, an increase in the number of processing nodes results in decreasing the total system throughput. The Inter-Task Communication (ITC) is always the most costly and the least reliable parameter in the loosely coupled DRTS. In this paper an efficient task allocation algorithm is discussed, which performs a static allocation of a set of "m" tasks $T = \{t_1, t_2, ...t_m\}$ of a program to a set of "n" processors $P = \{p_1, p_2, ...p_n\}$, (where, m >> n) to minimize the application program's Parallel Processing Cost(PPC) with the goal to maximize the overall throughput of the system through and allocated load on all the processors should be approximately balanced. While designing the algorithm the Execution Cost (EC) and Inter Task Communication Cost (ITCC) have been taken into consideration.*

Keywords: *Distributed Real time System, Execution Cost, Inter Task Communication Cost, Task Allocation, Load Balancing*

I. Introduction

The increasing complexity of various real life problems results in greater demand for faster computer components. One of the approaches to meet this growing demand is the use of parallel processing. An alternative and closely related to parallel computers is the concept of DRTS. Distributed real time system is a computer system in which multiple processors connected together through a high-bandwidth communication link. These links provides a medium for each processor to access data and programs on remote processors.

A user-oriented definition of distributed computing is reported by [1,2] that " The Multiple Computers utilized cooperatively to solve problems i.e. to process and maintained the large scale database of the programs which are to be executed on these type of computing environment". The assignment of task to processors is an essential step in exploiting the capabilities of a DRTS and may be done in a variety of ways (i) Static Allocation and (ii) Dynamic Allocation. In static allocation, when a task is assigned to processor, it remains there while the characteristic of the computation change and a new assignment must be computed. These problems may be categorized in static [3 -10]. In order to make the best use of resources in a distributed real time computing environment, it is essential to reassign the tasks dynamically during program execution, so as to the benefit of changes in the local reference patterns of the program [11-18]. Although the dynamic allocation has potential performance advantages, Static allocation is easier to realize and less complex to operate.

Several other methods have been reported in the literature, such as, Integer programming [19, 21], Branch and bound technique [22-23], Matrix reduction technique [7], and reliability evaluation to deal with various design and allocation issues in a DRTS by [24-30]. In this paper we introduce an algorithm which performs static allocation of such program tasks in a heterogeneous DRTS to minimize the application program's Parallel Processing Cost with the goal to maximize the overall system throughput and allocated load on all the processors should be approximately balanced. Because strictly balanced load distribution may not be possible to achieve, a system is considered to be balanced if the load on each processor is equal to the average load, within a reasonable tolerance. A tolerance of 10-15% of average load is generally chosen. We assume that the number of program modules is much larger than the number of processors, so that no processor remains idle. Several sets of input data are considered to test the efficiency and complexity of the algorithm. It is found that algorithm is suitable for arbitrary number of processors with the random program structure and is workable in all the cases.

II. Definitions

Execution Cost: The eij, is the amount of the work to be performed by the executing task ti on the processor pj where $1 \leq i \leq m$, $1 \leq j \leq n$. If a task is not executable on any of the processor due to absence of some resources, execution cost of same task on that processor is taken to be (∞) infinite. The process of allocation of the problem can be formulated by a function Aalloc, for the task assign to processors j. Aalloc: $T \rightarrow P$, where Aalloc (i) = j, if the task ti is assigned to processor pj, the overall EC of a given assignment Aalloc is then computed by equation (1).

$$\mathrm{EC}(\mathrm{Aalloc}) = \sum_{1 \le i \le m} e_{i,\mathrm{Aalloc}\ (i)} \tag{1}$$

and the per processor EC for processor p_j is defined to be

$$\mathrm{EC}(\mathrm{Aalloc})_j = \sum_{\substack{1 \le i \le m \\ i \in TS_j}} e_{i,\mathrm{Aalloc}\ (i)} \tag{2}$$

where TS_j is the set of tasks allocated to processor p_j

TS_j= {i: Aalloc (i) =j, j=1, 2…n}

Inter Task Communication Cost: The ITCC c_{ij} is incurred between tasks, where $c_{ij} = g > 0$ if tasks t_i communicates with task t_j for some cost g when Aalloc (i) ≠ Aalloc (j). Whenever a group of tasks is assigned to the same processor, the $c_{ij} = 0$. The overall ITCC of a given assignment Aalloc can be expressed by equation (3).

$$\mathrm{ITCC}(\mathrm{Aalloc}) = \sum_{\substack{1 \le i \le m \\ i+1 \le k \le m \\ \mathrm{Aalloc}\ (i) = j \ne \mathrm{Aalloc}\ (k)}} C_{\mathrm{Aalloc}\ (i),\mathrm{Aalloc}\ (k)} \tag{3}$$

and the per processor ITCC is given by equation (4)

$$\mathrm{ITCC}(\mathrm{Aalloc})_j = \sum_{\substack{1 \le i \le m \\ i+1 \le k \le m \\ \mathrm{Aalloc}\ (i) = j \ne \mathrm{Aalloc}\ (k)}} C_{\mathrm{Aalloc}\ (i),\mathrm{Aalloc}\ (k)} \tag{4}$$

III. Mathematical Model

Considered a application program that consists of "m" communicating tasks, t_1, t_2….t_m, and a heterogeneous distributed real time system with "n" processors, p_1, p_2,….p_n, unified by communication relations. It has processors as nodes and there is a weighted edge between two nodes if the corresponding processors can communicate with each other. The weight w_{ij} on the adjoin between processors p_i and p_j represent the time lag involved in sending or receiving the message of unit length from one processor to another. In order to have an approximate estimate of this time lag, irrespective of the two processors, we use the average of the weights on all the edges in the processor graph. This is called the average unit time lag. The load balancing, which involves sending load from over utilized processors to underutilized processors, should be carried out with due regard for communication overhead so that it is completed as speedily as possible. It becomes essential to optimize the overall throughput of the processors by allocating the tasks in such a way that the allocated load on all the processors should be approximately balanced. Therefore, the systematic allocation of tasks in a DRTS is the fundamental requirement for optimal utilization of processor's capacity. While developing the algorithm, it is assumed that the processing cost of these tasks on all the processors is given in the form of Execution Cost Matrix [ECM] of order m x n and the ITCC incurring between two communicating when they are assigned to two distinct processors is taken in the form of a symmetric matrix named as Inter Task Communication Cost Matrix [ITCCM], which is of order m.
The proposed methodology will work as follow:

- **Computation of Average Load must be assigned to each Processor**

Select ECM (,) and compute the average load must be assigned to each processor p_j by using the equation 5 and total load to be assigned on the system by equation 6.

$$\mathrm{Lavg}(p)_j = \frac{W_j}{n}, j = 1,2, \dots \dots n \tag{5}$$

where $W_j = \sum_{1 \le i \le m} ec_{ij}$

$$T_{lod} = \sum_{1}^{n} \mathrm{Lavg}(p_j) \qquad (6)$$

- **Determination of Average Minimum Link**

First concentrate on those "n" tasks which have the average minimum link between the tasks using equation (7). The average minimum link is stored in a two dimensional array AML (,) the first column of the array represents the task number and second column represents the average minimum link between the tasks. The array is sorted in ascending order by assuming second column as sorting key.

$$\mathrm{ITCC}_{avg}(t_i) = \frac{CC_i}{m}, \text{ where } CC_i = \sum_{1 \le i \le m} c_i \qquad (7)$$

- **Determination of Initial Assignment**

Select first "n" task from AML (,) and apply the Yadav et al Algorithm [28] on these "n" tasks in ECM (,). Store these assignment in $T_{ass}\{\ \}$and also store the processors position in Aalloc{ }. The total number of task allocated to the processor is than stored in TTASK (j) which can be computed by adding the values of Aalloc (j) if a task t_i is assigned to processor p_j otherwise continue. Remaining unassigned (m-n) task are then store in $T_{non\text{-}ass}\{\ \}$.

- **Clustering of remaining unassigned task**

Remaining (m-n) tasks store in $T_{non\text{-}ass}\{\ \}$ are clustered based on their communication requirement. Highly communicating tasks are clustered together to reduce communication delays. Usually number of tasks clusters should be equal to the number of processor so that one to one mapping may result.

These clusters will be fixed throughout their execution. Since we have 'n' number of processors in DRTS, therefore we will make 'n' number of tasks clusters. A cluster may contain up to maximum number $C = \left\lceil \frac{(m-n)}{n} \right\rceil$ of tasks. Store the ITCCM (,) in NITCCM (,) and reduce NITCCM (,) by removing the Tasks Store in $T_{ass}\{\ \}$ and the upper diagonal k=[{(m-n)*((m-n)-1)}/2]/-1 values of NITCCM (,) are stored in a array CCMAX (,) of order k x 3 the first column represents first task (say r^{th} task), second column represent the second task (say s^{th} task) and third column represent the ITCC (crs) between these r^{th} and s^{th} tasks. The CCMAX (,) is sorted in descending order by assuming third column as sorting key. Initially each task is treated like a cluster $C_i=\{t_i\}$for i=1 to m-n. Store these clusters in a linear array CLS= {Ci, $1 \le i \le$ m-n }. Select the first tasks pair say (t_r , t_s) (say t_r ε C_r and t_s ε Cs) from CCMAX (,). If the sum of number of tasks for clusters C_r and C_s is $\le$ C, than fuse the clusters C_r with C_s otherwise select the next tasks pair from CCMAX (,). Modify CLS= {} by replacing

the cluster C_r as $C_r \leftarrow Cr \cup Cs = \{t_r, t_s\}$ and deleting the cluster C_s. Modify the CCMAX (,) by deleting this tasks pair (t_r , t_s) and replace the value between t_r and t_s to zero in NITCCM (,) also reduce the matrix by add the r^{th} row with s^{th} and r^{th} column with s^{th}. Some of the tasks may not involve in any cluster may be treated as independent task. The above procedure is repeated until and unless we do not get number of tasks clusters equal to number of processors

- **Identification of Final Assignment**

The ECM (,) is also radiuses by summing the corresponding row and apply the Yadav et al Algorithm [28] on these "n" cluster for their allocation.

- **Computation of overall EC, ITCC and per processor EC, ITCC**

The overall EC, ITCC and per processor EC, ITCC for processor p_j of a given assignment Aalloc is then computed by equation (1), equation (2) equation (3) and equation (4) respectively.

- **Identification the Service Rate and Throughput of each processor**

The Service Rate (SR) and Throughout (TRP) of the processors are calculated by using the equation (8) and (9) respectively.

$$SR_j = \frac{1}{EC(\mathrm{Aalloc})_j} \qquad (8)$$

$$TRP_j = \frac{\mathrm{TTASK}(j)}{EC(\mathrm{Aalloc})_j} \qquad (9)$$

- **Identification of Parallel Processing Cost (PPC)**

The PPC is a function of the amount of computation to be performed by each processor and the communication load. This function is defined by considering the processor with the heaviest aggregate computation and communication loads. PPC for a given assignment Aalloc is defined guardedly by assuming that computation cannot be overlapped with communication) calculated by using the equation (10)

$$\mathrm{PPC(Aalloc)} = \max_{1 \le j \le n}\left\{\mathrm{EC(Aalloc)}_j + \mathrm{ITCC(Aalloc)}_j\right\} \qquad (10)$$

- **Computation of Overall System Cost (OSC)**

Overall OSC of the DRTS for the given assignment Aalloc is calculated by using the equation (11)

$$\mathrm{OSC(Aalloc)} = \mathrm{EC(Aalloc)} + \mathrm{ITCC(Aalloc)} \qquad (11)$$

Procedure

Step-1 Input m, n, ECM (,) and ITCCM (,)

Step-2 AVERAGE_LOAD(:)// Select ECM (,)and Compute the average load must be assigned to each processor p_j also determine the total load can be allocated to the system

Step-3 AVERAGE_ MINIMALLY_ LINKED(:) Select ITCCM (,) and Determine the average minimally linked between the tasks and store these link in a two dimensional array AML (,) the first column of the array represents the task number and second column represents the average minimum link between the tasks. The array is sorted in ascending order by assuming second column as sorting key.

Step-4 TASK_MAPPING ():// Select ECM (,) and apply Yadav et al Algorithm [28] in respect of first "n" tasks of ALM (,)
Store these assignment in T_{ass}\{ \}and also store the processors position in Aalloc\{ \}
Stored in TTASK (j) which can be computed by adding the values of Aalloc (j)

Step-4.1 Remaining unassigned (m-n) task are then store in $T_{non\text{-}ass}$\{ \}.

Step-5 TASK_CLUSTER ():// Select ITCCM (,)and store NITCCM (,)←ITCCM(,)

Step-5.1 Reduce NITCCM (,) by removing the Tasks Store in Tass\{ \}
Prepared "n" Cluster of the remaining (m-n) tasks
Compute maximum number of tasks in cluster $\mathrm{C} = \left\lceil \frac{(m-n)}{n} \right\rceil$

Step-5.2 k= [\{(m-n)*((m-n)-1)\}/2]/-1 Upper diagonal values of NITCCM (,) are stored in a array CCMAX (,) of order k x 3 the first column represents first task (say r–th task), second column represent the second task (say s-th task) and third column represent the ITCC (crs)

Step-5.3 The CCMAX (,) is sorted in descending order by assuming third column as sorting key

Step-5.4 Initially each task is treated like a cluster Ci = \{ti \} for i=1 to m-n.
Store these clusters in a linear array CLS= \{Ci, 1 ≤ i ≤ m-n)
Select the first tasks pair say (tr , ts) (say tr ε Cr and ts ε Cs) from CCMAX (,)

Step-5.5 If number of tasks for clusters Cr and Cs is ≤ C, than fuse the clusters Cr with Cs otherwise select the next tasks pair from CCMAX (,)

Step-5.6 Modify CLS= \{\} by replacing the cluster Cr as Cr ← Cr ∪ Cs=\{tr , ts\} and deleting the cluster Cs. Modify the CCMAX (,) by deleting this tasks pair (tr , ts) and replace the value between t_r and t_s to zero in NITCCM (,) also reduce the matrix by add the r^{th} row with s^{th} and r^{th} column with s^{th}

Step-6 Modify the ECM (,) by summing the r^{th} row with s^{th}

Step-7 If Ci≠nThen Go to step 5.4 Otherwise

Step-8 FINAL_TASK_MAPPING ():// Yadav et al Algorithm [28]
Store these assignment in T_{ass}\{ \}and also store the processors position in Aalloc\{ \}
Stored in TTASK (j) which can be computed by adding the values of Aalloc (j)

Step-9 COST_COMPUTATION ()://Compute overall EC , per processor EC for processor pj of a given assignment Aalloc
Compute overall ITCC, per processor ITCC for processor pj of a given assignment Aalloc
Compute overall OSC of the DRTS for the given assignment Aalloc
Compute PPC of a given assignment Aalloc
Compute Service Rate (SR) and Throughout TRP of the processors

Step-10 Stop

IV. Result & Discussions

To justify the application and usefulness of the present algorithm, a DTRS have been taken which consist of a set of "n = 3" processors P = $\{p_1, p_2, p_3\}$ connected by an arbitrary network. The execution cost matrix, ECM (,) of order m x n is considered. A typical program graph of a set of "m = 9" tasks T = $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ has been taken from the literature as considered by Yadav et al [31], Younes [32] and Elsadek [3].

Example: Input m = 9, n = 3 ECM (,) and ITCCM (,)

$$ECM(,)= \begin{array}{c|ccc|} & p_1 & p_2 & p_3 \\ t_1 & 174 & 156 & 110 \\ t_2 & 95 & 15 & 134 \\ t_3 & 196 & 79 & 156 \\ t_4 & 148 & 215 & 143 \\ t_5 & 44 & 234 & 122 \\ t_6 & 241 & 225 & 27 \\ t_7 & 12 & 28 & 192 \\ t_8 & 215 & 13 & 122 \\ t_9 & 211 & 11 & 208 \end{array}$$

$$ITCCM(,)= \begin{array}{c|ccccccccc|} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \\ t_1 & 0 & 8 & 10 & 4 & 0 & 3 & 4 & 0 & 0 \\ t_2 & 8 & 0 & 7 & 0 & 0 & 0 & 0 & 3 & 0 \\ t_3 & 10 & 7 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ t_4 & 4 & 0 & 1 & 0 & 6 & 0 & 0 & 8 & 0 \\ t_5 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\ t_6 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ t_7 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ t_8 & 0 & 3 & 0 & 8 & 0 & 0 & 0 & 0 & 5 \\ t_9 & 0 & 0 & 0 & 0 & 0 & 12 & 10 & 5 & 0 \end{array}$$

Average and total load to be assigned on each processor is shown in table 1 after calculating by using the equation 5 and 6.

Table-1: Average and total load to be assigned on each processor

Load	p_1	p_2	p_3	Total Load
Actual Load	445	325	405	1175
Tolerance of 10	45	33	40	118
Total	490	358	445	118

Compute the Average Minimally Linked between the Tasks using equation (7)

Original ALM(,) Sorted ALM(,)

$$ALM(,)= \begin{array}{|cc|} t_1 & 3.22 \\ t_2 & 2.00 \\ t_3 & 2.00 \\ t_4 & 2.11 \\ t_5 & 0.67 \\ t_6 & 1.67 \\ t_7 & 1.56 \\ t_8 & 1.78 \\ t_9 & 3.00 \end{array} \qquad ALM(,)= \begin{array}{|cc|} t_5 & 0.67 \\ t_7 & 1.56 \\ t_6 & 1.67 \\ t_8 & 1.78 \\ t_2 & 2.00 \\ t_3 & 2.00 \\ t_4 & 2.11 \\ t_9 & 3.00 \\ t_1 & 3.22 \end{array}$$

Initial Allocation obtained by applying Yadav et el [28] Algorithm are given in Table-2

Table-2

Tasks	Processor	Initial allocated Load
t_5	p_1	44
t_6	p_2	27
t_7	p_3	28

T_{ass} ={ t_5, t_7, t_6}, Aalloc(j) = (p_1, p_3, p_2) and TTASK (j) = (1,1,1)

$T_{non\text{-}ass}$= { t_8, t_2, t_3, t_4, t_9, t_1}

Clustering of remaining unassigned task

Maximum number of tasks in cluster c = 2

Store the ITCCM (,) in NITCCM (,) and reduce NITCCM (,) by removing the Tasks Store in T_{ass} ={ t_5, t_7, t_6}

$$\text{NITCCM}(,)= \begin{array}{c|cccccc|} & t_1 & t_2 & t_3 & t_4 & t_8 & t_9 \\ t_1 & 0 & 8 & 10 & 4 & 0 & 0 \\ t_2 & 8 & 0 & 7 & 0 & 3 & 0 \\ t_3 & 10 & 7 & 0 & 1 & 0 & 0 \\ t_4 & 4 & 0 & 1 & 0 & 8 & 0 \\ t_8 & 0 & 3 & 0 & 8 & 0 & 5 \\ t_9 & 0 & 0 & 0 & 0 & 5 & 0 \end{array}$$

Upper diagonal k=[{(m-n)*((m-n)-1)}/2]/-1= 14 values of NITCCM (,) are stored in a array CCMAX (,) of order k x 3

Original CCMAX (,) $\qquad\qquad$ Sorted CCMAX (,)

$$\text{CCMAX}(,)= \left|\begin{array}{ccc} t_1 & t_2 & 8 \\ t_1 & t_3 & 10 \\ t_1 & t_4 & 4 \\ t_1 & t_8 & 0 \\ t_1 & t_9 & 0 \\ t_2 & t_3 & 7 \\ t_2 & t_4 & 0 \\ t_2 & t_8 & 3 \\ t_2 & t_9 & 0 \\ t_3 & t_4 & 1 \\ t_3 & t_8 & 0 \\ t_3 & t_9 & 0 \\ t_4 & t_8 & 8 \\ t_4 & t_9 & 5 \end{array}\right| \qquad \text{CCMAX}(,)= \left|\begin{array}{ccc} t_1 & t_3 & 10 \\ t_1 & t_2 & 8 \\ t_4 & t_8 & 8 \\ t_2 & t_3 & 7 \\ t_4 & t_9 & 5 \\ t_1 & t_4 & 4 \\ t_2 & t_8 & 3 \\ t_3 & t_4 & 1 \\ t_1 & t_8 & 0 \\ t_1 & t_9 & 0 \\ t_2 & t_4 & 0 \\ t_2 & t_9 & 0 \\ t_3 & t_8 & 0 \\ t_3 & t_9 & 0 \end{array}\right|$$

Flowing three clusters are form and shown in Table-3

Table-3 Final Clusters

C_1	t_1+t_3
C_2	t_4+t_8
C_3	t_2+t_9

After implementation of full procedure the final assignments and EC, ITCC, PPC, Service Rate and Throughput of different processors achieved by the model are shown in Table 4.

Table -4 Final Results Obtained by the Algorithm

Task	Processor	EC	ITCC	PPC	Service Rate	Throughput
(1)	(2)	(3)	(4)	(5)	(6)	(7)
t_5+ t_2+t_9	p_1	350	51	**401**	0.00249	0.0075
t_6+ t_1+t_3	p_2	263	33	296	0.00340	0.0101
t_7+ t_4+t_8	p_3	292	34	326	0.00307	0.0092
Total allocated Lode		905	118	1023		

Table 4 and Figure 1 shows the results of the proposed model from the table and figure it is concluded that the Parallel Processing Cost of the system is 401 which is related to processor p_1. Figure 2 shows the through put and services rate of the processor. From the figure it is concluded that both are the ideally linked. The Figure 3 depicted the comparisons between calculated load and allocated load form the figure concluded that the allocated load assigned to the processors is much less than the calculated load.

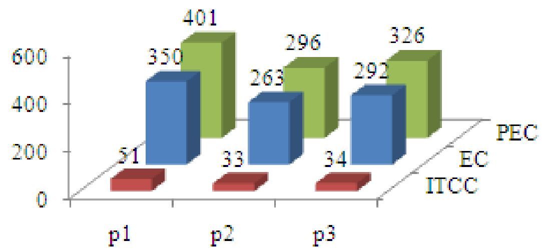

Fig. 1: Results of the proposed model

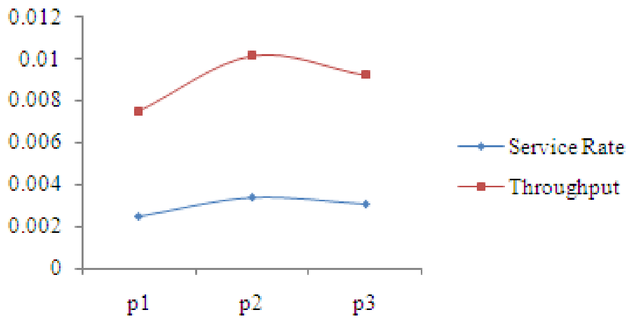

Fig. 2: Processor's Services Rate and Throughput

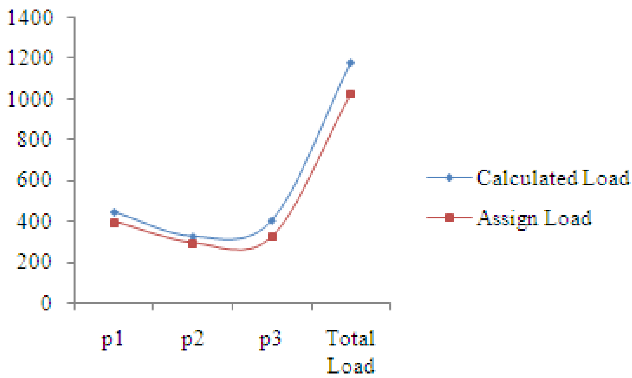

Fig. 3: Comparisons between Allocated Load and Calculated Load

The present paper deals with a simple, yet efficient mathematical and computational algorithm to identify the optimal busy cost of the system. The performance of the algorithm is compared with the algorithm reported by Yadav et al [31], Elsadek et al [3] and Younes [32]. The figure 4 and table 5 shows the comparisons of optimal cost of the system reported by the [31, 32] and the present method. From the figure it is concluded that the PPC of the system is much less obtained by the present method.

Table -5: Comparisons of PPC of the system

Yadav et al [31]	528
Elsadek [3]	479
Younes [32].	459
Present Model	401

It can also be perceived from the example presented here that wherever the algorithm of better complexity is encountered Kumar et al [6] present technique gets an upper hand by producing better optimal results with slight enhancement in the cost due to minor in complexity factor. The worst case run time complexity of the algorithm suggested by Kumar et al [6] is O (m^2n+n^2 + 2mn) , Elsade et a [3] is O (n^2+ m^2+ m^2n+2mn) and the run complexity of the algorithm presented in this paper is O (m^2n+mn+ n^2). Table 6 and Figure 4 shows the run time complexity of Kumar et al [6], Elsade el al [3] and Present Method considering different cases of tasks and processors.

Table- 6: Run time complexity compression

m.n	Elsade el al [3]					Kumar el al [6]				Present Algorithm			
	$O(n^2 + m^2 + m^2n+m+n)$					$O(m^2n+2mn+n^2)$				$O(m^2n+mn+m(m+n))$			
	n^2	m^2	m^2n	m+n	Total	m^2n	2mn	n^2	Total	m(m+n)	m*n^2	m*n	
	1	2	3	4	(1+2+3+4)	5	6	7	(5+6+7)	8	9	10	(8+9+10)
5,3	9	25	75	30	139	75	30	9	114	40	45	15	100
6,3	9	36	108	36	189	108	36	9	153	54	54	18	126
7,4	16	49	196	56	317	196	56	16	268	77	112	28	217
8,4	16	64	256	64	400	256	64	16	336	96	128	32	256
9,5	25	81	405	90	601	405	90	25	520	126	225	45	396
10,5	25	100	500	100	725	500	100	25	625	150	250	50	450
11,6	36	121	726	132	1015	726	132	36	894	187	396	66	649
12,6	36	144	864	144	1188	864	144	36	1044	216	432	72	720
13,7	49	169	1183	182	1583	1183	182	49	1414	260	637	91	988
13,7	49	196	1372	196	1813	1372	196	49	1617	294	686	98	1078
15,8	64	225	1800	240	2329	1800	240	64	2104	345	960	120	1425
15,8	64	256	2048	256	2624	2048	256	64	2368	384	1024	128	1536

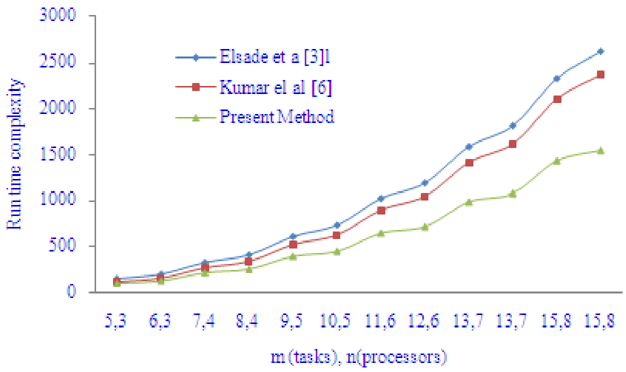

Fig. 4: Run time complexity compression

References

[1] K.K. Bhutani, Distributed Computing, The Indian Journal of Telecommunication, 1994, pp. 41-44.
[2] B.R. Sitaram, Distributed Computing – A User's View Point, CSI Communication, vol.-18 No. 10, 1965, pp. 26-28.
[3] A. A. Elsadek and B. E. Wells, A Heuristic model for task allocation in heterogeneous distributed computing systems, The International Journal of Computers and Their Applications, Vol. 6, No. 1, 1999, pp 0-35
[4] D.W. Coit, and A.E. Smith, Reliability Optimization of Series Parallel Systems using a Genetic Algorithm. IEEE Transactions on Reliabilit,. vol. R-45, 1996, pp. 254-260.
[5] P. K. Yadav and Nadeem Ahmad, Performance Analysis of Heterogeneous Distributed Processing System through Systematic Allocation of Task, International Journal of Intelligent Information Processing, Vol. 5(1), 2011. ,pp. 19– 24.
[6] V. Kumar, M. P. Singh, and P.K. Yadav, An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System. Proc. of the 19th National system conference, SSI, Coimbatore, India, 1995, pp. 82-87.
[7] V. Kumar, P.K. Yadav and K. Bhatia, Optimal Task Allocation in Distributed Systems owing to Inter Tasks Communication Effects. Proc. of the 33rd Annual convention of system society of India, New Delhi, India, 1998, pp. 369-378.
[8] M.P Singh,., V. Kumar and A. Kumar, An Efficient Algorithm for Optimizing Reliability Index in Tasks-Allocation. Acta Ciencia Indica. 1999, pp. 437-444.
[9] Srinivasan, Santhanam and K. Niraj Jha, Safety and Reliability Driven Task Allocation in Distributed System, IEEE Transactions on Parallel and Distributed Systems. vol. 10, 1999, pp. 238-250.
[10] P. K.Yadav, M. P Singh and Harendra Kumar, Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with dynamic Reassignment. International Journal of Computer System, Network and Communication,2008, pp. 1-9
[11] S.H. Bokhari, Dual Processor Scheduling with Dynamic Re-Assignment, IEEE Transactions on Software Engineering. vol. SE-5, 1979, pp. 341-349.
[12] T.L. Casavent and J. G Kuhl, A Taxonomy of Scheduling in General Purpose Distributed Computing System, IEEE Transactions on Software Engineering, vol. SE-14, 1988, pp. 141-154.
[13] Avanish Kumar, Optimizing for the Dynamic Task Allocation, Proceedings of the III Conference of the International Academy of Physical Sciences, Allahabad, 1999, pp. 281-294.
[14] V. Kumar, M. P. Singh, and P.K. Yadav, An Efficient Algorithm for Multi-processor Scheduling with Dynamic Reassignment, Proc. of the 6th National seminar on theoretical Computer Science, Banasthally Vidyapeeth, India, 1996, pp. 105-118.
[15] H.G. Rotithor, Taxonomy of Dynamic Task Scheduling in Distributed Computing System., IEEE Proc. Computer Digit Tech., vol. 14, , 1994, pp. 1-10.
[16] K. B. Misra and U. Sharma, An Efficient Algorithm to solve Integer Programming Problem arising in System Reliability Design, IEEE Transactions on Reliability, vol. R-40, 1991, pp. 81-91.
[17] R. L. Bulfin, and C. Y. Liu, Optimal Allocation of Redundant Components for large Systems, IEEE Transactions on Reliability, vol. R-34, 1985, pp. 241-247.
[18] W.W. Chu, Optimal File Allocation in a Multiple Computing System. IEEE Transactions on Computer, vol. C-18,1969, pp. 885-889.

[19] R.Y. Richard, E.Y.S Lee, and M. Tsuchiya, A Task Allocation Model for Distributed Computer System, IEEE Transactions on Computer, vol. C-31, 1982, pp. 41-47.

[20] O.I. Dessoukiu-EI, and W. H. Huna, Distributed Enumeration on Network Computers. IEEE Transactions on Computer, Vol. C-29, 1980, pp. 818-825.

[21] Kent Fitzgerald, Shahram Latifi and Pradip K. Srimani, Reliability Modeling and Assessment of the Star-Graph Networks. IEEE Transactions on Reliability, vol. R-51, 2002, pp. 49-57.

[22] P. K. Yadav, M. P Singh and Harendra Kumar, Scheduling of Communicating Modules of Periodic Tasks in Distributed Real Time Environment, International Journal of Applied Mathematics & Engineering Sciences, vol. 2(2), 2008, pp. 193- 200.

[23] M.P. Singh, Monika and P.K. Yadav, Queuing discipline to reduce the connection tracking flaws in protocols that uses quantum periodicity for scalable network services. International Transactions in mathematical Sciences & Computer, vol. I, 2008, pp. 102-122.

[24] Avanish Kumar, An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost, Proceedings of International Conference on Mathematical Modeling, Roorkee, 2001, pp. 150-155.

[25] Min-Sheng Lin, A Linear-time Algorithm for Computing K-terminal Reliability on Proper Interval Graphs, IEEE Transactions Reliability, vol. R-51, 2002, pp. 58-62.

[26] Michael R.Lyu, Sampath Rangarajan, and Aad P. A. Van Moorsel, Optimal Allocation of test Resources for Software Reliability growth modeling in Software Development, IEEE Transactions on Reliability, vol. R-51, 2002, pp. 183-192.

[27] P. K.Yadav, Jumindera Singh and M. P Singh, An efficient method for task scheduling in computer communication network, International Journal of Intelligent Information Processing, Vol. 3(1),2009, pp. 81-89

[28] P. K. Yadav, Kumar, Avanish Kumar and M. P. Singh, An Algorithm for Solving the Unbalanced Assignment Problems. International Journal of Mathematical Sciences, Vol. 12(2), 2004, pp. 447-461.

[29] K. Bhatia, P. K. Yadav, and Sagar Gulati, Design and Simulation of a Reliable Distributed System Based on Fault Tree Analysis, CiiT International Journal of Networking and Communication Engineering, Vol 4 (11), 2012, pp. 684-688.

[30] M.P. Singh, P.K. Yadav and A. Aggarwal, Tasks Scheduling in a Distributed Processing Environment: A Genetic Approach, International Journal of Information & Computation Technology, Vol. 3(2), 2013, pp. 93-97.

[31] P.K. Yadav, M. P. Singh and Kuldeep Shama, An Optimal Task Allocation Model for System Cost analysis in Hetrogeneous Distributed Computing Systems: A Heuristic Approach, International Journal of Computer Applications, Vol. 28, No. 4, 2011, pp. 30-37.

[32] Younes Hamed Ahmed, Task Allocation for Minimizing Cost of Distributed Computing Systems Using Genetic Algorithms, International Journal of Advanced Research in Computer Science and Software Engineering, Vol 2(9), 2012, pp.202-209.