International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2016 IJSRCSEIT | Volume 1 | Issue 1 | ISSN : 2456-3307
doi : https://doi.org/10.32628/CSEIT2063106

Optimal Task Assignment in Distributed Systems through Greedy algorithm

Kapil Govil*, Arun Kumar Yadav, Harihar Nath Verma
Department of Computer Science & Applications, ITM University, Gwalior, Madhya Pradesh, India

ABSTRACT

A distributed processing environment consists of one or more applications spread over several computers. These computers may be geographically separated from one another. The applications may be executed on different platforms using different operating systems and telecommunication protocols. In short, a distributed operating environment offers a variety of information system solutions regardless of the location of the user, user's operating system or the equipment using by user. Performance enhancement of the distributed networks is a major and challenging problem for the researchers. In this paper, algorithm designed, have to allocate m tasks to n processors (m>n) in the environment of distributed processing. These m tasks are to be assigned on the n processors of system through Greedy algorithm of task scheduling. The starting time and finishing time of the processing of a task is considered and denoted by Si and Fi where i= 1, 2, 3, 4, 5…… m. In order to evaluate the optimal time, present algorithm obtains the set of assigned and on assigned tasks. Based on these sets allocation has been made. The method is presented into algorithmic form and several sets of input data have been implemented to test the effectiveness of the algorithm.

Keywords: Distributed Processing Environment, Performance Enhancement, Greedy algorithm, Allocation

I. INTRODUCTION

A distributed system [6 – 8, 11, 13, 17, 29 – 34] is a collection of loosely coupled processors interconnected by a communication network [5, 15, 19, 22 – 24]. The distributed processing environment [8, 10, 14, 18, 20, 25, 28] is the environment, in which services provided by the network reside at multiple sites. Instead of single large machine being responsible for all aspects of process, each separate processor handles subset. In the distributed environment the programs or tasks are also often developed with the subsets of independent units under various environments. It has drawn tremendous attention in developing cost-effective and reliable applications to meet the desired requirement. Profit density based Greedy Knapsack algorithm [3] is one simple approach that can ensure near-optimal profit. However, profit gain is sometimes not the only factor concerned in making important management decisions. Kovalev et al [1] proposed a research using the isotone property with respect to the canonical order, they described a class of objective functions and a class of polyhedral feasible sets which provide the optimal Greedy [3] solution for the problem. The main research problem for such networks is the allocation problem, in which all the tasks or modules are to be assigned optimally and performance measures are to be optimized.

II. OBJECTIVE

The objective of the present research problem is to enhance the performance of the distributed systems by using the proper utilization of its processors. The present problem minimizes the overall processing time of a distributed system through optimally assigning the tasks of various processors of the system. The distributed system consist of n processors that are denoted through a set P = {p_1, p_2, p_3, ……..p_n}. The processors are interconnected by communication links. A set T = {t_1, t_2, t_3, ……t_m} has also been considered. These m tasks are to be assigned on the n processors of system through greedy algorithm. The number of tasks is more than the number of processors of the system. The starting time and finish time of the processing of a task is considered and denoted by S_i and F_i, where i = 1, 2, 3,……m.

III. TECHNIQUE

It is considered that a distributed system [6 – 8, 11, 13, 17, 29 – 34] having n processors interconnected by communication link. It is denoted by a set P = {p_1, p_2, p_3, ……p_n} of n processors. A set T = {t_1, t_2, t_3,……t_m} of m tasks is to be considered. These m tasks are to be assigned on the n processors, of the distributed systems while m>>n. The set S_i and F_i where i = 1, 2, 3 ….m, consist of starting and finishing time for the processing of the tasks to the system. The difference between the finishing time and starting time shows the duration of processing of a task on a processor of the distributed system.

In order to evaluate the optimal time, present algorithm takes input the number of processors i.e., n and number of tasks i.e., m. Again taking input the starting time and finish time of each task through an array namely, $TTMP_j$(,) of order m x 2. Algorithm reads the task communication matrix CTM(,) of order. Then on storing the difference of starting time and finishing time from $TTMP_j$(,) to DTM graphical representation of $TTMP_j$(,) using Greedy Activity Scheduling Algorithm (GASA) [3] has been created to obtain the pair of all of the non - interfering tasks in T_{ass}(,). Now, algorithm replaces the processing time with ∞ in DTM(,) where tasks are not assigned and store the results in UDTM(,). Algorithm calculates the sum of each row in UDTM(,) and stores into avg_row( ); On storing the sorted avg_row( ) in avg_row_asc( ) and storing the corresponding tasks into $Task_{seq}$, algorithm selects set of n tasks which have at least one processing time in each row and each column, store the results in SM_i(,). On applying algorithm of Kumar et. al [2] on SM_i(,) to make allocations and marking the assignments [9, 12, 16, 17, 21, 26, 27] along with their values overall optimal time by adding optimal time with communication time can be obtained.

IV. ALGORITHM

Start algo
    Read the number of processors in n
    Read the number of tasks in m
    Read the starting and finish time of each task in $TTMP_j$(,)
    Read the task communication matrix CTM(,) of order m x m.
    Calculate the difference of starting and finishing time from $TTMP_j$(,)and store it into DTM(,) for all processors.
    Make the graphical representation of $TTMP_j$(,)
    While (all tasks != SELECTED)
    {
        Select the tasks which are not interfering with other tasks using GASA.
        Make the pair of assigned tasks in T_{ass}(,) for each processor.
    }
    Replace the processing time with ∞ in DTM(,) where tasks are not assigned and store the results in UDTM(,)
    Calculate the sum of each row in UDTM(,) and store into avg_row( )
    Sort then in ascending order and store into avg_row_asc( )
    While (all tasks != SELECTED)
    {
        Select n tasks from $Task_{seq}$() those have at least one processing time in each row and each column and store the results in SM_i(,)
        Apply strategy of Kumar et. al. [2] on SM_i(,) to make allocations
        Mark the assignments along with their values
    }
    Mark the overall allocation along with their value
    Overall Optimal Time = Processing Time + Communication Time
End algo

V. IMPLEMENTATION

In this problem, the distributed system consist a set P of 3 processors $\{p_1, p_2, p_3\}$ and a set T of 10 tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$. The starting and finishing time for all 3 processors are as given in the matrices of the order 10 x 2, namely, Task Time Matrix for Processors p_1, p_2 and p_3 namely [$TTMP_1(,)$], [$TTMP_2(,)$] and [$TTMP_3(,)$] respectively.

$$TTMP_1(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{matrix} S_i & F_i \end{matrix} \\ \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 1 & 4 \\ 3 & 6 \\ 4 & 7 \\ 5 & 8 \\ 6 & 11 \\ 7 & 9 \\ 8 & 11 \\ 9 & 10 \end{bmatrix} \end{array} \qquad TTMP_2(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{matrix} S_i & F_i \end{matrix} \\ \begin{bmatrix} 1 & 4 \\ 2 & 3 \\ 3 & 5 \\ 4 & 7 \\ 6 & 11 \\ 7 & 9 \\ 8 & 11 \\ 10 & 12 \\ 11 & 13 \\ 12 & 14 \end{bmatrix} \end{array}$$

$$TTMP_3(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{matrix} S_i & F_i \end{matrix} \\ \begin{bmatrix} 1 & 3 \\ 2 & 6 \\ 3 & 4 \\ 5 & 7 \\ 7 & 8 \\ 8 & 11 \\ 9 & 11 \\ 10 & 12 \\ 12 & 14 \\ 13 & 14 \end{bmatrix} \end{array}$$

The communication amongst the tasks has also taken into consideration and it is represented by the square symmetric communication matrix namely CTM(,) of order n x n:

$$CTM(\,,\,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \end{matrix} \\ \begin{bmatrix} 0 & 1 & 2 & 5 & 6 & 3 & 2 & 8 & 9 & 4 \\ & 0 & 4 & 7 & 6 & 8 & 2 & 3 & 6 & 2 \\ & & 0 & 1 & 7 & 4 & 3 & 5 & 5 & 6 \\ & & & 0 & 1 & 2 & 9 & 3 & 4 & 7 \\ & & & & 0 & 5 & 8 & 2 & 6 & 3 \\ & & & & & 0 & 1 & 4 & 5 & 2 \\ & & & & & & 0 & 2 & 3 & 1 \\ & & & & & & & 0 & 3 & 6 \\ & & & & & & & & 0 & 7 \\ & & & & & & & & & 0 \end{bmatrix} \end{array}$$

Obtaining the matrix DTM(,) of order 10 x 3, which shows the difference between starting time and finishing time of each task at each processor is as:

$$DTM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{matrix} p_1 & p_2 & p_3 \end{matrix} \\ \begin{bmatrix} 2 & 3 & 2 \\ 3 & 1 & 4 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \\ 3 & 5 & 1 \\ 3 & 2 & 3 \\ 5 & 3 & 2 \\ 2 & 2 & 2 \\ 3 & 2 & 2 \\ 1 & 2 & 1 \end{bmatrix} \end{array}$$

The graphical representation of the matrices $TTMP_1(,)$, $TTMP_2(,)$ and $TTMP_3(,)$ using Greedy Activity Scheduling Algorithm (GASA) [3] are shown in Figures, 1, 2 and 3 respectively.

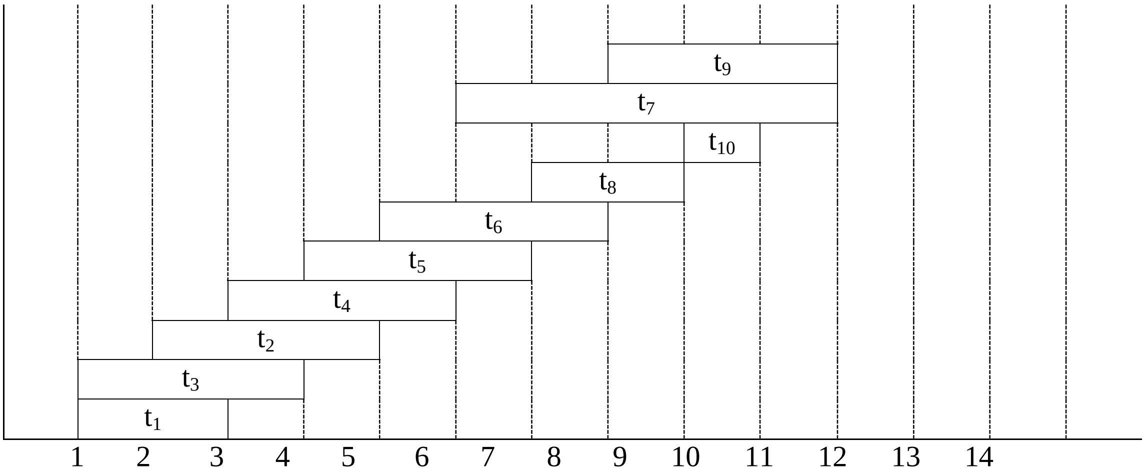

Figure 1. Task assignment graph for p_1

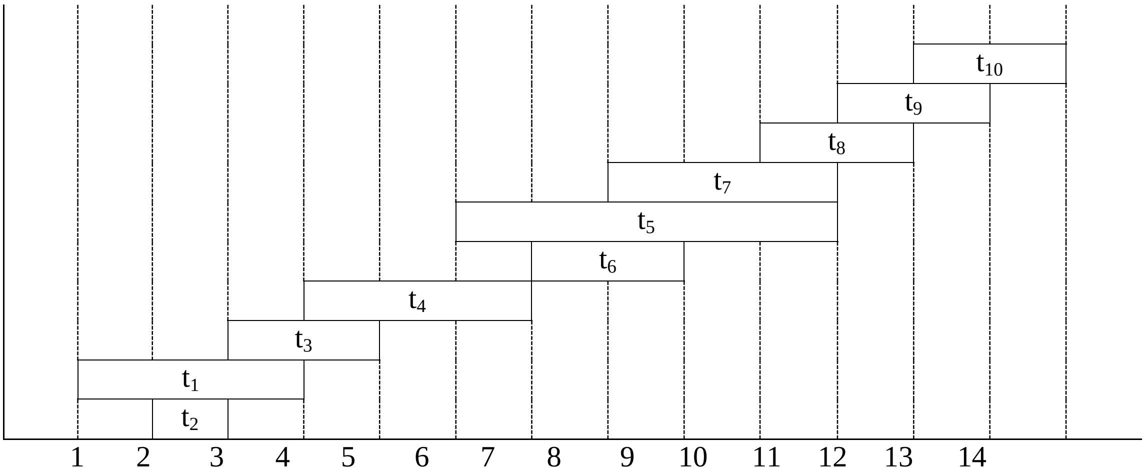

Figure 2. Task assignment graph for p_2

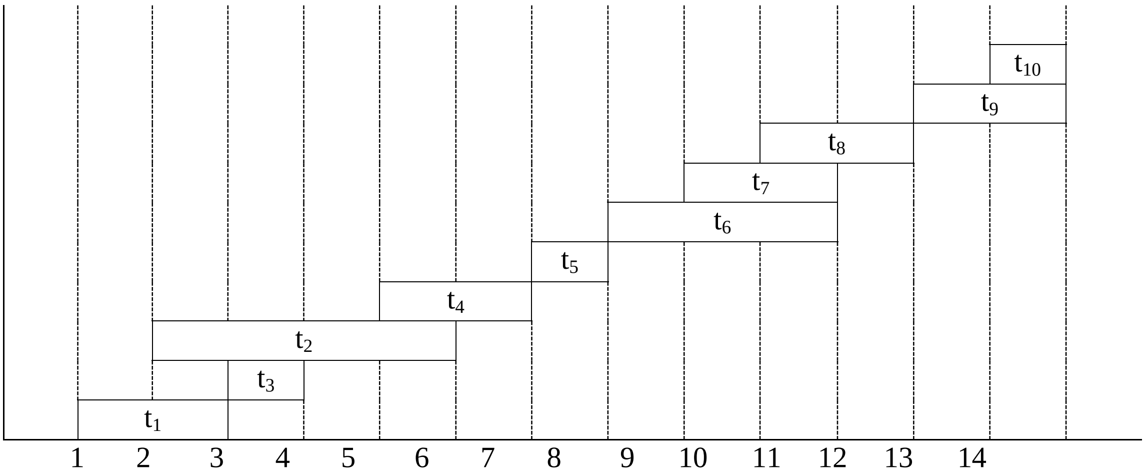

Figure 3. Task assignment graph for p_3

Now, on selecting those tasks, which are not interfering with another task from the Figure1, 2 and 3, selected (shaded) task are obtained as shown in Figure 4, Figure 5 and Figure 6 respectively.

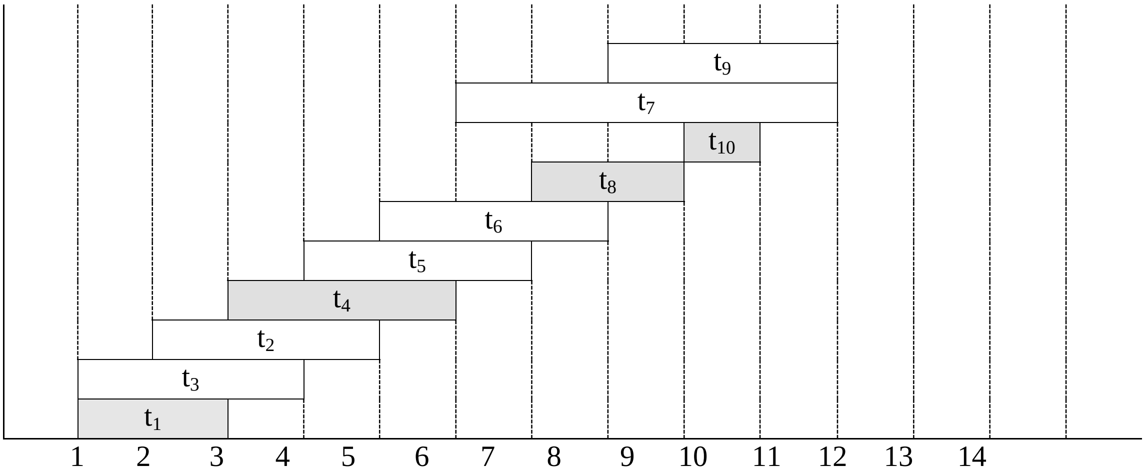

Figure 4. Task assignment graph for p_1

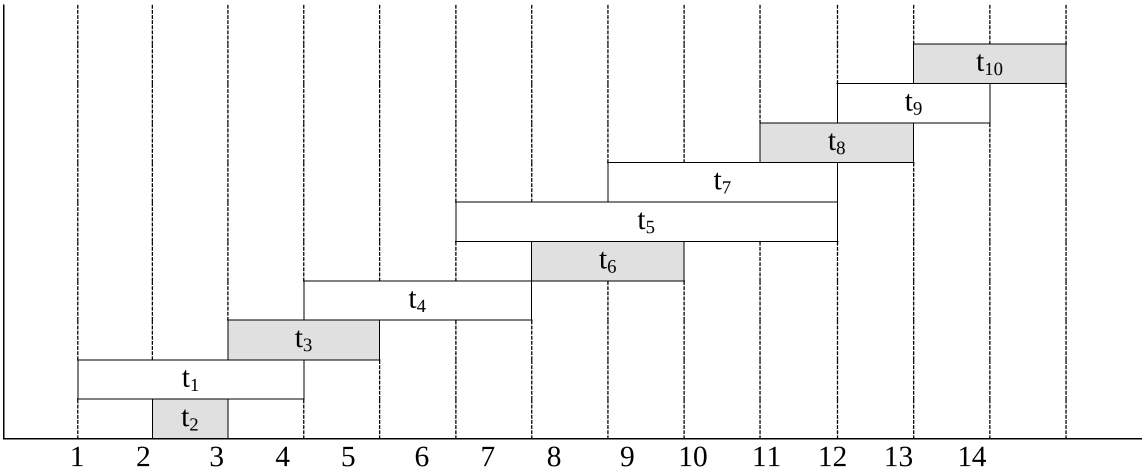

Figure 5. Task assignment graph for p_2

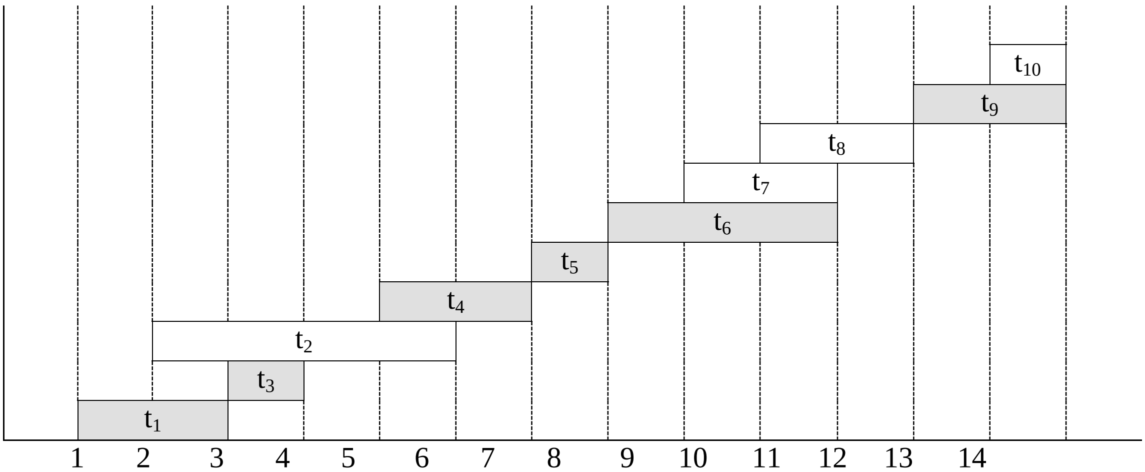

Figure 6. Task assignment graph for p_3

A set of assigned and non-assigned tasks for each processor is created as –

For Processor p_1:

Assigned tasks $T_{ass}(1,) = \{t_1, t_4, t_8, t_{10}\}$

Non-Assigned tasks $T_{non_ass}(1,) = \{t_3, t_2, t_5, t_6, t_7, t_9\}$For Processor p_2:

Assigned tasks $T_{ass}(1,) = \{t_2, t_3, t_6, t_8, t_{10}\}$

Non-Assigned tasks $T_{non\_ass} = \{t_1, t_4, t_5, t_7, t_9\}$

For Processor p_3:

Assigned tasks $T_{ass}(1,)= \{t_1, t_3, t_4, t_5, t_6, t_9\}$
Non-Assigned tasks $T_{non\_ass} = \{t_2, t_7, t_8\}$

Now, on finding out which one task is still in non - assigned set, here, task t_7 is not assigned to any processor, so the process for task t_7 has been repeated. So, the graphical representation of the task t_7 from matrices $TTMP_1(,)$, $TTMP_2(,)$ and $TTMP_3(,)$ are shown in the Figures namely Figure 7, 8 and 9 respectively.

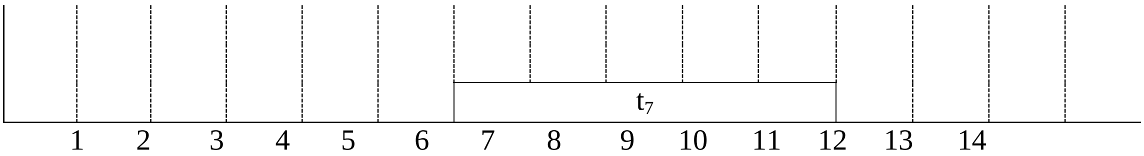

Figure 7. Task assignment graph for p_1

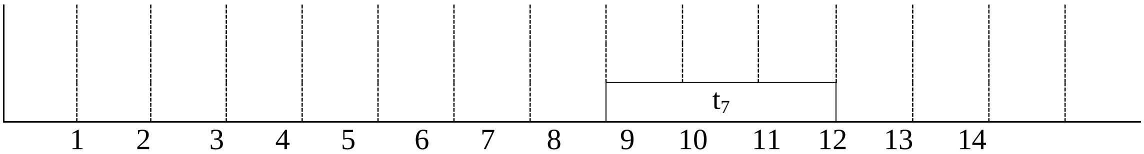

Figure 8. Task assignment graph for p_2

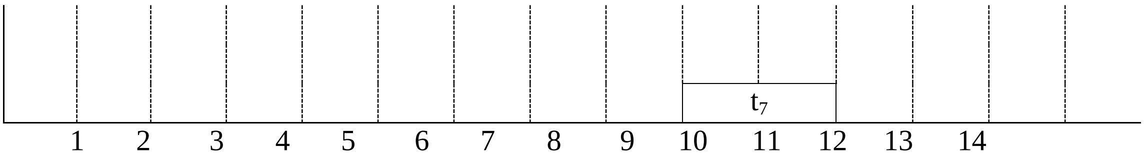

Figure 9. Task assignment graph for p_3

As there is only one task in Figure 7, 8 and 9, so no other task can interfere task t_7; So the task t_7 will be selected with all 3 processors namely p_1, p_2, and p_3. The graphical representations are shown in Figure 10, 11 and 12.

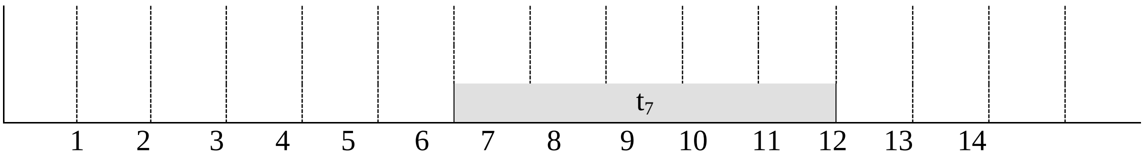

Figure 10. Task assignment graph for p_1

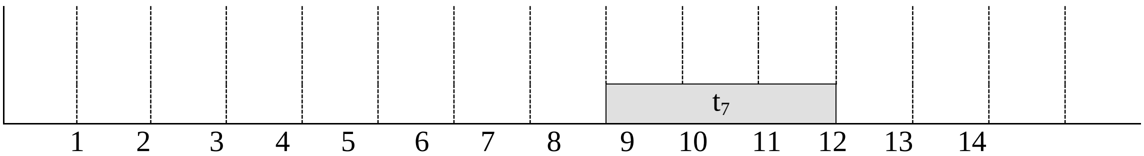

Figure 11. Task assignment graph for p_2

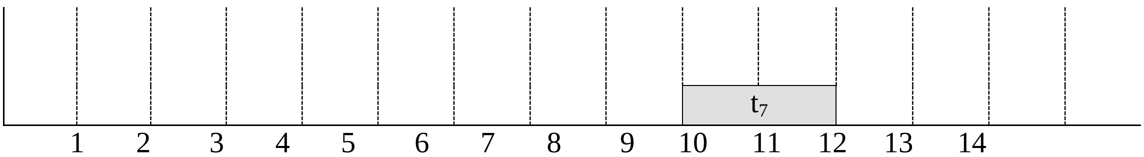

Figure 12. Task assignment graph for p_3

Now, on modifying the set of assigned and non - assigned tasks for each processors as -

For Processor p_1:

Assigned tasks $T_{ass}(1,) = \{t_1, t_4, t_8, t_{10}, t_7\}$
Non - Assigned tasks $T_{non\_ass}(1,) = \{t_3, t_2, t_5, t_6, t_9\}$

For Processor p_2:

Assigned tasks $T_{ass}(1,) = \{t_2, t_3, t_6, t_8, t_{10}, t_7,\}$
Non - Assigned tasks $T_{non\_ass}(1,) = \{t_1, t_4, t_5, t_9\}$

For Processor p_3:

Assigned tasks $T_{ass}(1,) = \{t_1, t_3, t_4, t_5, t_6, t_9, t_7,\}$
Non - Assigned tasks $T_{non\_ass}(1,) = \{t_2, t_8\}$

Now, replacing the processing time with ∞ (infinity) in DTM(,), where the tasks are not assigned to the processor, on storing the updated task matrix DTM(,) and store as UDTM(,), i.e.,

$$UDTM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 2 & \infty & 2 \\ \infty & 1 & \infty \\ \infty & 2 & 1 \\ 3 & \infty & 2 \\ \infty & \infty & 1 \\ \infty & 2 & 3 \\ 5 & 3 & 2 \\ 2 & 2 & \infty \\ \infty & \infty & 2 \\ 1 & 2 & \infty \end{bmatrix} \end{array}$$

Obtaining the sum of each row (by keeping ∞ aside) and storing it in a linear array sum_row( ) along with their corresponding tasks i.e.,

$$avg\_row() = \begin{cases} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \\ 4+\infty & 1+\infty & 3+\infty & 5+\infty & 1+\infty & 5+\infty & 10 & 4+\infty & 2+\infty & 3+\infty \end{cases}$$

On sorting the avg_row( ) in ascending order can keeping ∞ aside and store the results in linear array avg_row_asc( ) along their corresponding tasks. These are given below;

$$avg\_row\_asc() = \begin{cases} t_2 & t_5 & t_9 & t_3 & t_{10} & t_1 \\ 1+\infty & 1+\infty & 2+\infty & 3+\infty & 3+\infty & 4+\infty \end{cases}$$

On storing the corresponding tasks in $Task_{seq}()$ as,

$Task_{seq}() = \{t_2, t_5, t_9, t_3, t_{10}, t_1, t_8, t_4, t_6, t_7\}$

On selecting first 3 tasks from $Task_{seq}$, which have at least one processing time in each row and each column Selection Matrix namely $SM_1(,)$ can be obtained as -

$$SM_1(,) = \begin{array}{c} \\ t_2 \\ t_5 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{bmatrix} \infty & 1 & \infty \\ \infty & \infty & 1 \\ 1 & 2 & \infty \end{bmatrix} \end{array}$$

On applying Kumar et. al. [2] strategy on $SM_1(,)$ the task assignment can be find as -

Processor	Task
p_1	t_{10}
p_2	t_2
p_3	t_5

On again selecting next three tasks, $SM_2(,)$ can be obtained as –

$$SM_2(,) = \begin{array}{c} \\ t_9 \\ t_3 \\ t_1 \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{bmatrix} \infty & \infty & 2 \\ \infty & 2 & 1 \\ 2 & \infty & 2 \end{bmatrix} \end{array}$$

On applying Kumar et. al. [2] strategy on $SM_2(,)$ the task assignment can be obtained as -

Processor	Task
p_1	t_1
p_2	t_3
p_3	t_9

On again selecting next three tasks, $SM_3(,)$ can be obtained as –

$$SM_3(,) = \begin{array}{c} \\ t_8 \\ t_4 \\ t_6 \end{array} \begin{array}{c} \begin{array}{ccc} p_1 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 2 & 2 & \infty \\ 3 & \infty & 2 \\ \infty & 2 & 3 \end{bmatrix} \end{array}$$

On applying Kumar et. al. [2] strategy on $SM_3(,)$ the task assignment can be obtained as -

Processor	Task
p_1	t_8
p_2	t_6
p_3	t_4

Now, there are no next three tasks to allocate; but there is only one task as -

$$\begin{array}{cccc} & p_1 & p_2 & p_3 \\ SM_4(,) = t_7 & [5 & 3 & 2] \end{array}$$

On applying Kumar et. al. [2] strategy on processors the task assignment can be obtained as

Processor	Task
p_3	t_7

The final results are given in the Table I -

TABLE I

OPTIMAL RESULTS

Processor	Task	Processing Time	Communication Time	Overall Optimal Time
p_1	$t_{10}*t_1*t_8$	5	128	145
p_2	$t_2*t_3*t_6$	5		
p_3	$t_5*t_9*t_4*t_7$	7		

VI. CONCLUSION

The algorithm mentioned in this paper is based on the consideration of processing time of the tasks to various processors. It is found that method is useful for network designer working in the area of distributed systems. The graphical representation of the optimal assignment is shown by the Figure 13.

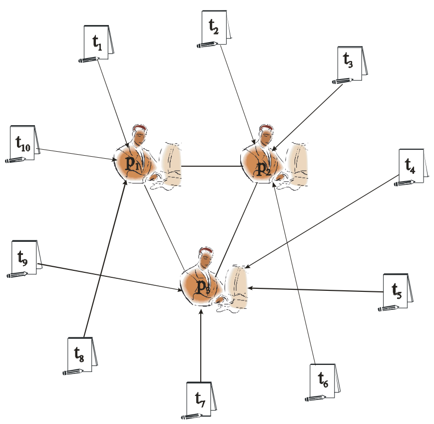

Figure 13. Optimal Assignment Graph

It may be noticed that task t_1, t_8 & t_{10} are being executed by processor p_1, task t_2, t_3 & t_6 are getting executed by processor p_2 and tasks t_4, t_5, t_7 & t_9 are getting executed by processor p_3. The processorwise processing time graph is shown in the Figure 14.

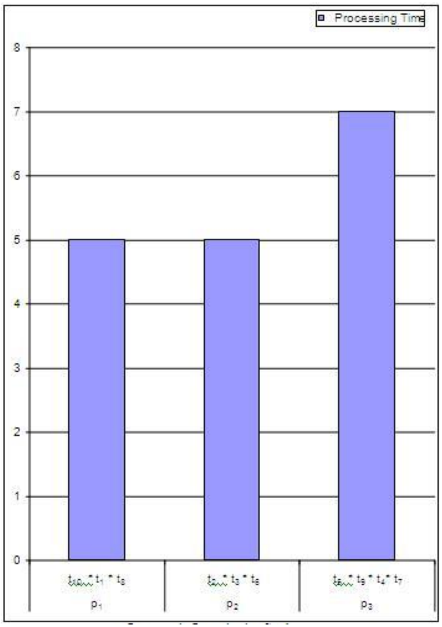

Figure 14. Processorwise Processing time Graph

The optimal result of the example that is considered to test the algorithm and it is mentioned in the implementation section of the problem are as given in Table II.

TABLE II

OPTIMAL RESULTS

Processors	**Tasks**	**Processing Time**	**Communication Time**	**Overall Optimal Time**
p_1	$t_{10}*t_1*t_8$	5	128	145
p_2	$t_2*t_3*t_6$	5		
p_3	$t_5*t_9*t_4*t_7$	7		

As the analysis of an algorithm is mainly focuses on its complexity. The complexity is a function of input size 'n'. It is referred to as the amount of time required by an algorithm to run to completion. The complexity of the above mentioned algorithm is $O(m^2n^2)$. The performance of the algorithm is compared with the algorithm suggested by Richard et al [4]. Table III shows the complexity comparison between algorithm [5] and present algorithm.

TABLE III

TIME COMPLEXITY

Processors n	Tasks m	Time Complexity	
		Algorithm [4] $O(n^m)$	Present algorithm $O(m^2n^2)$
3	4	81	144
3	5	243	225
3	6	729	324
3	7	2187	441
3	8	6561	576
4	5	1024	400
4	6	4096	576
4	7	16384	784
4	8	65536	1024
4	9	262144	1296
5	6	15625	900
5	7	78125	1225
5	8	390625	1600
5	9	1953125	2025
5	10	9765625	2500

From the Table III, it is clear that present algorithm is much better for optimal allocation of tasks that upgrade the performance of distributed system. Graphs 15, 16 and 17 also shows the comparison between algorithm [4] and present algorithm for n=3, 4 and 5 respectively.

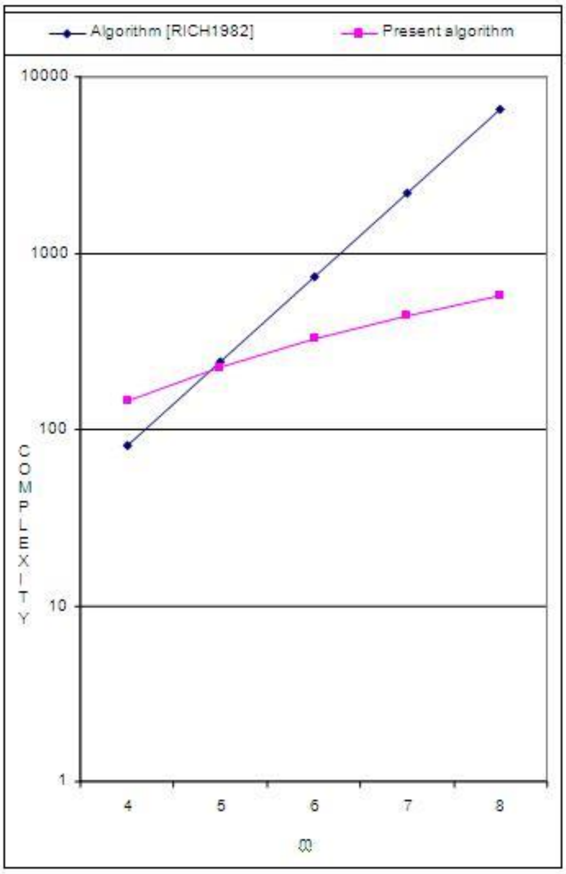

Figure 15. Comparison Graph for n=3

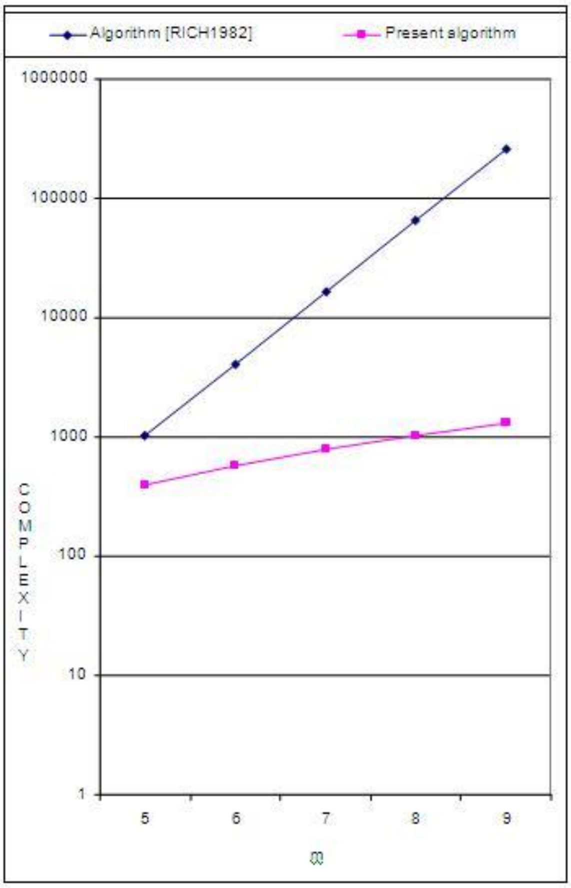

Figure 16. Comparison Graph for n=4

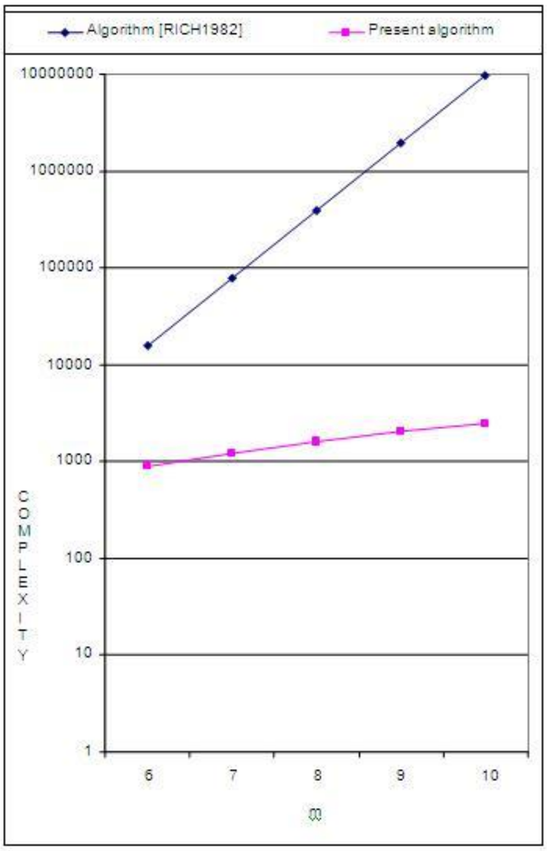

Figure 17. Comparison Graph for n=5

VII. REFERENCES

[1] M. M. Kovalev and D. M.Vasilkov. 1995. The canonical order and greedy algorithms. European Journal of Operational Research. (1995), Vol. 80. No. 2. pp. 446 – 450.

[2] A. Kumar, M. P. Singh and P. K. Yadav. 1995. A Fast Algorithm for Allocating Tasks in Distributed Processing System. In proceedings of the 30th Annual Convention of CSI, Hyderabad. (1995). pp. 347 – 358.

[3] Nitin Upadhyay, "The Design & Analysis of Algorithms", Katson Books, India, 2004.

[4] R. Y. Richard, E. Y. S. Lee and M. Tsuchiya. 1982. A Task Allocation Model for Distributed Computer System. IEEE Transactions on Computer. (1982), Vol. 31. pp. 41 – 47.

[5] T. F. Abelzaher and K. G. Shin. 1995. Optimal combined task and message scheduling in distributed real – time systems. IEEE Real – Time Systems Symposium. (1995), Vol. 16. pp. 162.

[6] Andrew S. Tanenbaum, "Computer Networks", Prentice Hall of India, India, 2001.

[7] D. F. Baca. 1989. Allocation modules to Processor in a distributed system. IEEE Transactions on Software Engineering. (1989), Vol. 15. pp. 1427 – 1436.

[8] T. L. Casavent and J. G. Kuhl. 1988. A. Taxonomy of Scheduling in General Purpose Distributed Computing System. IEEE Transactions on Software Engineering. (1988), Vol. 14. pp. 141 – 154.

[9] A. Farinelli, L. Iocchi, D. Nardi and V. A. Ziparo. 2005. Task Assignment with dynamic perception and constrained tasks in a Multi-Robot System. Proc. of International Conference on Robotics and Automation (ICRA'05). (2005).

[10] A. Y. Hamed. 2012. Task Allocation for Maximizing Reliability of Distributed Computing Systems Using Genetic Algorithms. International Journal of Computer Networks and Wireless Communications. (2012), Vol. 2. No. 5. pp. 560 – 569.

[11] Bassel Arafeh, Khalid Day and Abderezak Touzene. 2008. A multilevel partitioning approach for efficient tasks allocation in heterogeneous distributed systems. Journal of Systems Architecture: The EUROMICRO Journal. (2008), Vol. 54. No. 5. pp. 530 – 548.

[12] Faizul Navi Khan and Kapil Govil. 2014. A tricky task scheduling technique to optimize time cost and reliability in mobile computing environment. International Journal of Research in Engineering and Technology. (2014), Vol. 3. No. 5. pp. 823 – 829.

[13] Faizul Navi Khan and Kapil Govil. 2014. A Static approach to optimize time cost and reliability in Distributed Processing Environment. International Journal of Scientific & Engineering Research. (2014), Vol. 5. No. 5. pp. 1016 – 1021.

[14] Faizul Navi Khan and Kapil Govil. 2013. Cost Optimization Technique of Task Allocation in Heterogeneous Distributed Computing System. International Journal Advanced Networking and Applications. (2013), Vol. 5, No.3. pp. 1913 – 1916.

[15] Faizul Navi Khan, Kapil Govil. 2013. Static Approach for Efficient Task Allocation in Distributed Environment. International Journal of Computer Applications. (2013), Vol. 81. No. 15. pp. 19 – 22.

[16] Geir Horn, B. John Oommen. 2010. Solving multi constraint assignment problems using learning automata. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics. (2010), Vol. 40. No.1. pp. 6 – 18.

[17] Harendra Kumar, M. P. Singh, P. K. Yadav. 2013. Optimal Tasks Assignment for Multiple Heterogeneous Processors with Dynamic Re – assignment. International Journal of Computers & Technology. (2013), Vol. 4. No. 2. pp. 528 – 535.

[18] Kapil Govil. 2011. A Smart Algorithm for Dynamic Task Allocation for Distributed Processing Environment. International Journal of Computer Applications. (2011), Vol. 28. No. 2. pp. 13 – 19.

[19] M. P. Singh, P. K. Yadav, H. Kumar and B. Agarwal. 2012. Dynamic Tasks Scheduling Model for Performance Evaluation of a Distributed Computing System through Artificial Neural Network. Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) (Advances in Intelligent and Soft Computing: Published by Springer). (2012), Vol. 130. pp. 321 – 331.

[20] Manisha Sharma, Harendra Kumar and Deepak Garg. 2012. An Optimal Task Allocation Model through Clustering with Inter –Processor Distances in Heterogeneous Distributed Computing Systems. International Journal of Soft Computing and Engineering. (2012), Vol. 2. No. 1. pp. 50 – 55.

[21] Monika Choudhary and Sateesh Kumar Peddoju. 2012. A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment. International Journal of Engineering Research and Applications (IJERA). (2012), Vol. 2. No. 3. pp. 2564 – 2568.

[22] N. Beaumont. 2009. Using dynamic programming to determine an optimal strategy in a contract bridge tournament. Journal of the Operational Research Society. (2009), Vol. 61. No. 5. pp. 732 – 739.

[23] Palmer, J. and Mitrani I. 2005. Optimal and heuristic policies for dynamic server allocation. Journal of Parallel and Distributed Computing. (2005), Vol. 65. No. 10. pp. 1204 – 1211.

[24] Pradeep Kumar Yadav, M. P. Singh and Harendra Kumar. 2008. Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Reassignment. Journal of Computer Systems, Networks, and Communications. (2008), pp. 1 – 9.

[25] Sagar Dhakal, Majeed M. Hayat, Jorge E. Pezoa, Cundong Yang and David A. Bader. 2007. Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach. IEEE Transactions on Parallel and Distributed Systems. (2007), Vol. 18. No. 4. pp. 485 – 497.

[26] Shen Chenglin and Zhang Xinxin. 2009. Dynamic Mechanisms of Task – assignment for Virtual Enterprises Based on Multi-agent Theory. Proceedings of the 2009 International Symposium on Web Information Systems and Applications (WISA'09). (2009), pp. 525 – 528.

[27] Sunita Bansal, Bhavik Kothari and Chittaranjan Hota. 2011. Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm. IJCSI International Journal of Computer Science Issues. (2011), Vol. 8. No. 2. pp. 472 – 477.

[28] V. Pilloni, P. Navaratnam, S. Vural, L. Atzori and R. Tafazolli. 2014. TAN: A Distributed Algorithm for Dynamic Task Assignment in WSNs. Sensors Journal, IEEE. (2014), Vol. 14. No. 4. pp. 1266 – 1279.

[29] Xiangzhen Konga, Chuang Lina, Yixin Jianga, Wei Yana and Xiaowen Chub. 2011. Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. Journal of Network and Computer Applications. (2011), Vol. 34. No. 4. pp. 1068 – 1077.

[30] Yishuang Hu, Yi Ding, Fan Wen and Lei Liu. 2019. Reliability Assessment in Distributed Multi-State Series-Parallel Systems. Energy Procedia. (Feb 2019), Vol. 159. pp. 104 – 110.

[31] Amit Kumar Srivastava and Shishir Kumar. 2018. Dynamic Reconfiguration of robot software component in real time distributed system using clustering techniques. Procedia Computer Science. (2018), Vol. 125. pp. 754 – 761.

[32] Mohammed I. Alghamdi, Xunfei Jiang, Ji Zhang, Jifu Zhang and Xiao Qin. 2017. Towards two – phase scheduling of real-time applications in distributed systems, Journal of Network and Computer Applications. (Apr 2017), Vol. 84. pp. 109 – 117.

[33] Vikash Mishra and Vikram Singh. 2015. Generating Optimal Query Plans for Distributed Query Processing using Teacher – Learner Based Optimization. Procedia Computer Science. (2015), Vol. 54. pp. 281 – 290.

[34] Qingchao Jiang and Biao Huang. 2016. Distributed monitoring for large – scale processes based on multivariate statistical analysis and Bayesian method. Journal of Process Control. (Oct 2016), Vol. 46. pp. 75 – 83.

Cite This Article :

Kapil Govil, Arun Kumar Yadav, Harihar Nath Verma, "Optimal Task Assignment in Distributed Systems through Greedy algorithm", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6, Issue 3, pp.416-427, May-June-2020. Available at doi : https://doi.org/10.32628/CSEIT2063106
Journal URL : http://ijsrcseit.com/CSEIT2063106