International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 23 (2018) pp. 16424-16429

Tasks Assignment for Unbalanced Assignment Problems in Distributed Systems

Sarvesh Kumar Dubey

Department of Mathematics,
UIET, C S J M University, Kanpur- 208024, U.P., India.

Abstract

Task assignment is one of the most fundamental problems for balanced as well as for unbalanced assignment problems in distributed systems. The efficient method of assignment of tasks in a distributed system provides the optimal solution to the given assignment problem. It raises the performance of the distributed systems. In this research paper, we have presented a mathematical model of task assignment problems which consists of **m** tasks to be assigned to **n** processors (where **m** > **n**). The heuristic model deals with the optimal tasks assignment so that the total throughput of the system can be maximized. The proposed algorithm is capable enough to assign any number of task to the available processors while the Hungarian algorithm is capable only to assign the tasks equal to the number of available processors (i.e. for **m** = **n**) in the system. Several results like processing execution cost, mean service rate, mean service cost and the total throughput have been computed by applying the algorithm.

Keywords: Unbalanced assignment problem, Hungarian method, Distributed systems, Processors, Throughput.

INTRODUCTION

Since the past decade, Distributed Computing Systems (DCSs) are gaining more importance due to the recent advances and developments in the technology used. DCS is a collection of autonomous processors interconnected by a high-speed communication network which share their activities to complete a goal. A user-oriented definition of distributed computing is "Multiple Computers, utilized cooperatively to solve problems"[1, 2]. DCS applies its different components simultaneously to achieve a goal. In this form of computing, the work is accomplished by breaking the large problems into the smaller ones, which are than solved concurrently. Parallel computing has become the dominant paradigm in computer architecture in the form of multi-core processors[3]. The main aim of DCSs are higher throughput, improved availability and better access to a widely communicated web of information[4]. The performance of the DCS depends over the distribution of the resources. To make the best use of available processors in a distributed system, it is essential to assign the tasks to processors in such a manner that the overall throughput is maximized[5].

Time-to-time advancements in software and hardware technologies have made the distributed systems containing multiprocessors, more reliable, safe and with maximum throughputs. The most critical problems arising in the field of communication systems is the assignment of tasks, where, we need to assign a number of modules to different processors for execution to achieve various objectives, such as throughput maximization, reliability maximization, and cost/time minimization. The task assignment problems have been approached from various perspectives. Efficient coordination (scheduling of tasks) is a requirement for success in many applications of task assignment problems. A best allocation of tasks leads to a best balancing of the system[6]. The allocation tasks to processors in a distributed systems is presented by Baca[7] while Bokhari[8] demonstrated dual processor scheduling with dynamic re-assignment. Analysis of load distribution in DCS through systematic Allocation Task and an exhaustive approach of performance analysis to the distributed systems based on cost assignments have been reported by Kumar et. al.[9, 10]. J. B. Sinclair[11] asserted for finding an optimal assignment of the modules of a program to processors in a distributed system by multiprocessor scheduling with the aid of the network flow algorithms.

There are several methods reported in the literature, such as, Integer Programming[12, 13], Branch and Bound techniques[14], Matrix reduction technique[15, 16], Load balancing[17, 18], Task scheduling[19, 20], System cost analysis[21], Reliability optimization[22, 23], Modeling[24, 25] to deal with various design[26, 27] and allocation issues. In this paper we introduce a method of task assignment for the unbalanced assignment problems by formulating an algorithm. The proposed algorithm performs efficient assignment of tasks to processors so that the overall system cost can be minimized and the total throughput can be maximized. We have considered several sets of input to test the efficiency and the complexity of the algorithm and found that our assignment method is more suitable than the Hungarian method of assignment[28] for arbitrary number of processors with the random number of tasks.

NOTATIONS

T : the set of tasks to be executed

P : the set of processors in DCS

m : the number of tasks forming a program

n : the number of processors

t_i : i^{th} task of the given set of tasks

P_j : j^{th} processor in set of processor P

$x_{ij} : x_{ij} = \begin{cases} 1, & \text{if task } t_i \text{ is assigned to processor } p_j \\ 0, & \text{otherwise} \end{cases}$

T_{ass} { } : a linear array to hold assigned tasks

T_{non_ass} { } : a linear array to hold non assigned tasks.

P_{EC} : processor execution cost

N_{EC} : new execution cost

F_{EC} : final execution cost

TASK ASSIGNMENT PROBLEM

Consider an assignment problem which consist of a set P = {p_1, p_2, ..., p_n} of **n** processors. A set T = {t_1, t_2, ..., t_m} of **m** tasks is considered which are to be assigned for execution on **n** available processors. The execution cost ec_{ij} of each task on all the processors is given in the form of Execution Cost Matrix [ECM (,)] of order **m x n**. In order to make the best use of the available resources in systems, it becomes essential to maximize the overall throughput of the processors by allocating the tasks in such a way that the allocated load on all the processors should be evenly balanced. An allocation of tasks to processors from the set T to the set P is defined by A_{alloc} : T → P, where $A_{alloc}(i)$ = j if task t_i is assigned to processor P_j, $1 \le i \le m$, $1 \le j \le n$.

PROPOSED ASSIGNMENT METHOD

To determine the initial assignment apply Hungarian assignment algorithm. During the initial assignment process some of the tasks get allocated while others remain not assigned treated as unassigned. Let T_{ass} denote a set of tasks assigned to processors and T_{non_ass} denote a set of tasks, which are not assigned during the initial assignment process.

Obviously,

$$T = T_{ass} \cup T_{non\text{-}ass}$$

The processors positions of tasks belong to T_{ass} stored in another linear array A_{alloc}(j).

After getting the initial allocation store the execution cost ec_{ij} of each processor in terms of T_{ass} is also stored in a linear array P_{EC}(j), where, j = 1, 2, ……, n.

A task $t_a \in T_{non\_ass}$ is reassigned with one of the task $t_i \in T_{ass}$ executing on processor p_j.

Select a task $t_a \in T_{non\_ass}$ for reassignment with one of the tasks $t_i \in T_{ass}$. Put ec_{ij} of task $t_a \in T_{non\_ass}$ in order to A_{alloc}(j) and store in array N_{EC}(j).

Sum up the execution cost ec_{ij} of task t_a with all the tasks $t_i \in T_{ass}$ using the equation

$$[F_{EC}(j)] = [P_{EC}(j)] + [N_{EC}(j)] \quad \ldots (1)$$

and store the result in a linear array F_{EC}(j).

Find the minimum for F_{EC}(j) say min[ec_{ij}] related to t_i then reassign the task t_a with t_i and delete t_a from T_{non_ass} and store the assignment in T_{ass}, also store the processors positions in A_{alloc}(j). This process is continued until all the $t_a \in T_{non\_ass}$ are reassigned with any of the $t_i \in T_{ass}$.

The processing execution cost [PEC(j)] is obtained as:

$$[PEC(j)] = \sum ec_{ij}, \text{ (where, } 1 \le i \le m, 1 \le j \le n) \quad \ldots (2)$$

where, j = 1, 2, 3, ……., n.

The Total Processing Execution cost (TPEC) is expressed by equation:

$$TPEC = \sum_{i=1}^{m} \sum_{j=1}^{n} ec_{ij} x_{ij}, \quad \ldots (3)$$

where, i = 1, 2, 3, ……., m,

j = 1, 2, 3, ……., n,

and $x_{ij} = \begin{cases} 1, & \text{if task } t_i \text{ is assigned to processor } p_j \\ 0, & \text{otherwise} \end{cases}$

After getting the all assignments the Mean Service Rate [MSR(j)] of the processors is calculated by using the equation:

$$[MSR(j)] = \frac{1}{[PEC(j)]}, \quad \ldots (4)$$

where, j = 1, 2, 3, ……., n,

and store the results in a linear array [MSR(j)].

The Mean Service Cost [MSC(j)] is computed by applying equation as:

$$[MSC(j)] = \frac{1}{[MSR(j)]}, \quad \ldots (5)$$

where, j = 1, 2, 3, ……., n.

The total throughputs [T_{RP}(j)] of the processors is calculated by using the equation:

$$[T_{RP}(j)] = \frac{T_{TASK}(j)}{[PEC(j)]}, \quad \ldots (6)$$

where, T_{TASK}(j) = total number of tasks allocated to processor P_j, j = 1, 2, 3, ……., n,

and store the result values in [TRP(j)].

COMPUTATIONAL METHODOLOGY

The computational method discussed in the present model to determine the tasks allocation in distributed computing system is based on the following components.

- Determine the initial allocation
- Determine the final allocation
- Calculate the total execution cost
- Compute the mean service rate
- Calculate the throughput of the processors.

ALGORITHM

An algorithm for the efficient tasks allocation is given here in the following simple steps:

Step-I: Input [ECM (,)]

Step-II: Apply Hungarian Assignment method to obtain initial allocation as shown below:

T_{ass} = [t_{i1}, t_{i2}, ……, t_{in}]

T_{non_ass} = [t_{j1}, t_{j2}, ……., t_{jn}]

where $1 \le i_a \le m$ and $1 \le t_a \le m$.

Step-III: Store the processors positions where the tasks are initially allocated in a linear array

$A_{alloc}(j)$, where, j = 1, 2, ……, n.

Step-IV: Compute ec_{ij} for all $t_i \in T_{ass}$ on the P_j's and store the result in a linear array $P_{EC}(j)$ in order to $A_{alloc}(j)$.

Step-V: Select task $t_a \in T_{non\_ass}$

Step-V(a): Check ec_{ij} of t_a of all the P_j's and store the value in a linear array $N_{EC}(j)$ in order to $A_{alloc}(j)$.

Step-V(b): Compute the final ec_{ij} of t_a with all $t_i \in T_{ass}$ using equation (1) and store the result in a linear array $F_{EC}(j)$.

Step-V(c): Select the minimum for $F_{EC}(j)$, say min[ec_{ij}]; If min[ec_{ij}] $\in$ more than one t_i then select any of one randomly and reassign t_a with t_i.

Step-V(d): Modify the T_{non_ass} by deleting t_a, also modify the T_{ass} by storing the assignment in T_{ass}.

Step-V(e): Store the current processors positions in $A_{alloc}(j)$.

Step-V(f): Whether there is any $t_a \in T_{non\_ass}$.

If yes, then

Go to Step-V

Else

Go to Step-VI

End if.

Step-V(g): Store the allocation results in the assignment table.

Step-VI: Evaluate processing execution cost applying equation (2) and store the values in the array [PEC(j)].

Step-VII: Calculate the total processing execution cost using equation (3) and store the result obtained in [TPEC(j)].

Step-VIII: Calculate the mean service rate of the processors using equation (4). Store the values obtained in the linear array [MSR(j)].

Step-IX: Compute the mean service cost applying equation (5) and store the values in [MSC(j)].

Step-X: Compute the total throughputs of the processors applying equation (6) and store the results in a linear array over [$T_{RP}(j)$].

Step-XI: Stop.

IMPLEMENTATION OF ALLOCATION MODEL, RESULTS AND DISCUSSION

Algorithm has been formulated to model the execution process of tasks in the distributed computing environment. The algorithm can be used for efficient tasks allocation in distributed computing system. To justify the application and usefulness of the present algorithm, there is considered an example with the following Inputs:

EXAMPLE

Number of tasks to be executed (m) = 6

Number of processors of the system (n) = 3

ECM (,).

$$ECM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \begin{bmatrix} 5 & 3 & 2 \\ 4 & 5 & 9 \\ 4 & 1 & 3 \\ 2 & 6 & 5 \\ 6 & \infty & 3 \\ \infty & 2 & 5 \end{bmatrix} \end{array}$$

Apply the Hungarian assignment method to obtain the initial assignment as:

T_{ass} = [t_1, t_3, t_4]

T_{non_ass} = [t_2, t_5, t_6]

$A_{alloc}(j)$ = [P_3, P_2, P_1]

[TPEC(j)] = 5

Tasks	Processors	EC	$T_{RP}(j)$
t_1	P_3	2	0.500
t_3	P_2	1	1.000
t_4	P_1	2	0.500

Store the processors positions where the tasks are initially allocated in a linear array

$A_{alloc}(j)$ = [P_3, P_2, P_1]

where, j = 1, 2, 3,....., n.

Compute ec_{ij} for all $t_i \in T_{ass}$ on the P_j's in order to $A_{alloc}(j)$,

$[P_{EC}(j)] = [2, 1, 2]$

Select task $t_2 \in T_{non\_ass}$

Check ec_{ij} of t_2 of all the P_j's in order to $A_{alloc}(j)$,

$[N_{EC}(j)] = [9, 5, 4]$

Compute the final ec_{ij} of t_2 with all $t_i \in T_{ass}$ using equation (1) and store the result in a linear array $F_{EC}(j)$ as:

$[F_{EC}(j)] = [11, 6, 6]$

$Min[ec_{ij}] = Min[F_{EC}(j)] = 6$

$Min[ec_{ij}] \in t_4, t_5$

Select t_4 and reassign t_2 with t_4 on P_1.

Modified $T_{ass} = [t_1, t_3, t_4, t_2]$

Modified $T_{non\_ass} = [t_5, t_6]$

Store the current processors positions in $A_{alloc}(j)$ as:

$A_{alloc}(j) = [P_3, P_2, P_1, P_1]$

Repeat Step-V to Step-V(e) till the tasks $[t_5, t_6] \in T_{non\_ass}$ get reassigned.

The results in the assignment table are given as:

Tasks	Processors	Execution Cost
t_1	P_3	2
t_3	P_2	1
t_4	P_1	2
t_2	P_1	4
t_5	P_3	3
t_6	P_2	2

$T_{ass} = [t_1, t_3, t_4, t_2, t_5, t_6]$

$A_{alloc}(j) = [P_3, P_2, P_1, P_1, P_3, P_2]$

$TPEC(j) = 14$

Compute [PEC(j)] using equation (2), [MSR(j)] using equation (4), [MSC(j)] using equation (5), $[T_{RP}(j)]$ using equation (6). The final assignment table is given below:

Tasks	Processors	PEC(j)	MSR(j)	MSC(j)	$T_{RP}(j)$
t_4*t_2	P_1	6	0.167	5.988	0.333
t_3*t_6	P_2	3	0.333	3.003	0.667
t_1*t_5	P_3	5	0.200	5.000	0.400

The graphical representation of the mean service rate and the throughputs of the processors is demonstrated in the following figure:

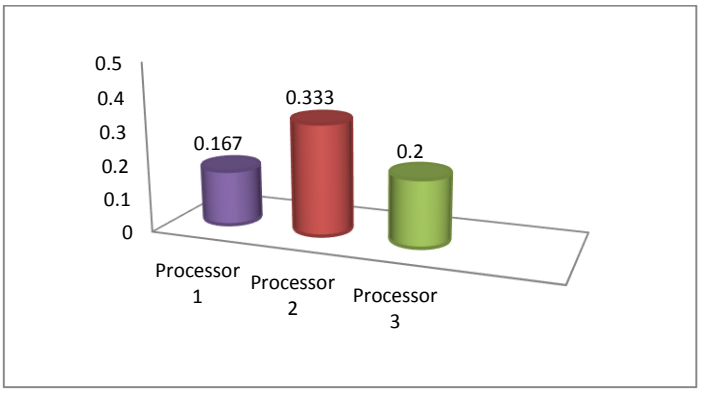

Figure-1: Mean Service Rate of the Processors

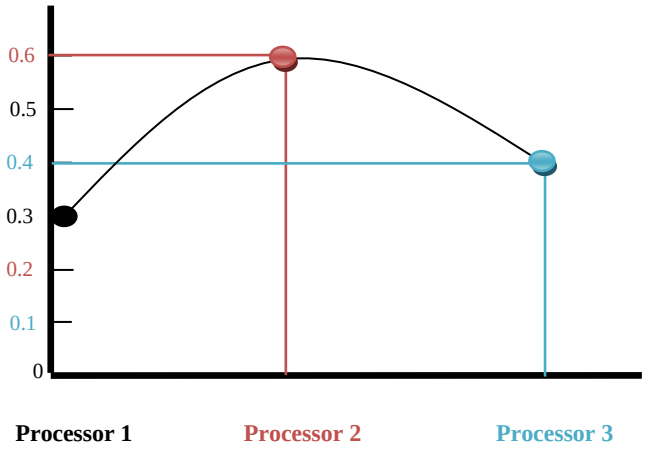

Figure-2: Throughputs of the processors

Here the Hungarian assignment approach is applied to solve the assignment problem and it is seen that this approach is applicable only when the number of tasks is equal to the number of processors that is for balanced assignment problem. For the unbalanced assignment problem, where **m** ≠ **n**, the approach suggest to add the dummy processors to make the effectiveness matrix square and the tasks are then assigned to dummy processors, which may not get executed in reality. It means that these tasks become pending and latter will be ignored. Keeping in view this reason we suggested an algorithm for the problems where, **m** > **n** that is the number of tasks is more than the number of processors (for unbalanced assignment problems) so that none of the tasks get remains unexecuted and also the proposed assignment approach does not require adding dummy processors.

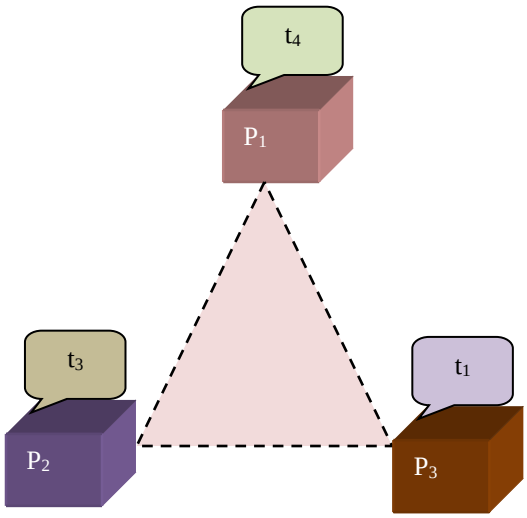

Figure-3

Hungarian's Assignment Method

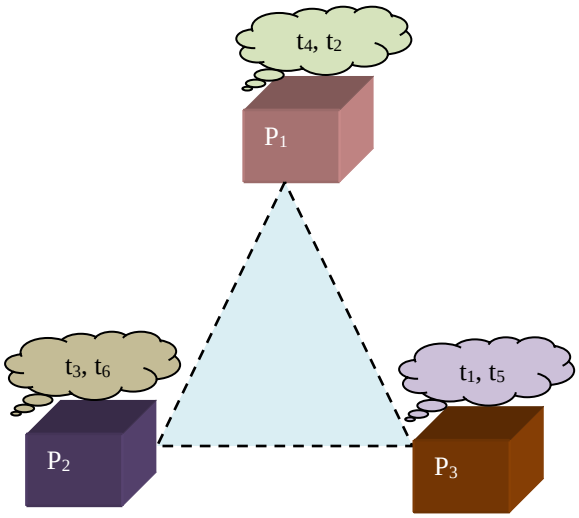

Figure-4

Proposed Assignment Method

CONCLUSION

In Hungarian assignment approach the tasks t_1, t_3 and t_4 are assigned to the processors P_3, P_2 and P_1 as shown in figure-3 while in our proposed assignment approach all the tasks are assigned to the processors as depicted in figure-4. The execution cost in Hungarian approach is 5 while the execution cost recorded in our approach is 14. The increase in execution cost is obvious as in Hungarian approach only three tasks get executed and rest of the tasks goes to the dummy processors but in the proposed approach all the six tasks get executed on the three processors. The mean service rate and the throughputs of processors have also been obtained and it is shown in figure-1 and figure-2 respectively. Several sets of input data are considered to test the effectiveness and efficiency of the algorithm and it is found that the algorithm is suitable for arbitrary number of processors with the random number of tasks for assignment.

REFERENCES

[1] Bhutani, K. K., 1994, "Distributed Computing", The Indian Journal of Telecommunication, pp. 41-44.

[2] Sitaram, B. R., 1965, "Distributed Computing – A User's View Point", CSI Communications, 18(10), pp. 26-28.

[3] Dubey, S. K. and Upadhyay, V., 2017, "Cluster, Grid, Cloud and Parallel Distributed Computing: An Overview with Comparison", International Journal of Applied Research on Information Technology and Computing, 8(2), pp. 152-167.

[4] Kumar, A., Sharma, A. and Dhagat, V. B., 2010, "Maximal Link Module Algorithm For Task Allocation In Distributed Computing System", Proceedings of the 4th National Conference; INDIACom- 2010.

[5] Dubey, S. K., Upadhyay, V. and Kumar, A., 2017, "Optimal Tasks Allocation Process Based on Fusion of the Unallocated Tasks for Distributed Systems", International Journal on Computer Science and Engi., 9(4), pp. 114-121.

[6] Zbakh, M. and Kettani, M. D. El., 2011, "A Task Allocation Algorithm For Distributed Systems", Journal of Theoretical and Applied Information Technology, 33(1), pp. 15-21.

[7] Baca, F. D., 1989, "Allocation Tasks to Processor in a Distributed System", IEEE Transactions on Software Engineering, SE-**15**, pp. 1427-1436.

[8] Bokhari, S. H., 1979, "Dual Processor Scheduling with Dynamic Re-Assignment", IEEE Transactions on Software Engineering, SE-5, pp. 341-349.

[9] Kumar, A., Yadav, P. K. and Sharma, A., 2010, "Analysis of Load Distribution in Distributed Computing systems Through Systematic Allocation Task", International Journal of Mathematics, Computer Science and Information Technology, 3(1), pp.101-114.

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

[11] Sinclair, J. B., 1987, "Efficient Computation of Optimal Assignments for Distributed Tasks", J. Parallel Dist. Comput., 4, pp. 342-362.

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

[13] Misra, K. B. and Sharma, U., 1991, "An Efficient Algorithm to solve Integer Programming Problem

arising in System Reliability Design", IEEE Transactions on Reliability, R-40, pp. 81-91.

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

[15] Sagar, G. and Sarje, A. K., 1991, "Task Allocation Model for Distributed System", Int. J. System Science, 22(9), pp. 1671-1678.

[16] Kumar,V., Yadav, P. K. and Bhatia, K., 1998, "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, pp. 369-378.

[17] Kumar, V., Singh, M. P. and Yadav, P. K., 1996, "An Efficient Algorithm for Multi-Processor Scheduling with Dynamic Re-Assignment", Proceedings of the Sixth National Seminar on Theoretical Computer Science, Banasthali Vidya Pith, pp. 105-118.

[18] Iqbal, S. and Carey, G. F., 2005, "Performance analysis of dynamic load balancing algorithms with variable number of processors", Jour. of Parall. and Distributed Comput., Elsevier Inc., 65(8), pp. 934–948.

[19] Casavent, T. L. and Kuhl, J. G., 1988, "A Taxonomy of Scheduling in General Purpose Distributed Computing System", IEEE Trans. On Software Engineering, 14, pp.141-154.

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

[21] Yadav, P. K., Singh, M. P. and Sharma, K., 2011, "An Optimal Task Allocation Model for System Cost analysis in Heterogeneous Distributed Computing Systems: A Heuristic Approach", International Journal of Computer Applications, 28(4), pp. 30-37.

[22] Tillman, F. A., Hwang, C. L. and Kuo, W., 1977, "Determining Component Reliability and Redundancy for Optimum System Reliability", IEEE Transactions on Reliability, R-26, pp. 162-165.

[23] Coit, D. W. and Smith, A. E., 1996, "Reliability Optimization of Series Parallel Systems using a Genetic Algorithm", IEEE Transactions on Reliability, R-45, pp. 254-260.

[24] Fitzserald, K., Latifi, S. and Srimani, P. K., 2002, "Reliability Modeling and Assessment of the Star Graph Network", IEEE Transactions of Reliability, 51(1), pp. 49-57.

[25] Kennington, J. L., Olinick, E. V. and Spiride, G., 2007, "Basic mathematical programming models for capacity allocation in mesh-based survivable networks", Omega, 35(6), pp. 629-644.

[26] Forter, P. J., 1985, "Design and Analysis of Distributed Real-time System", Inter Text Publication Inc., and McGraw Hill, Inc., New York.

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

[28] Kuhn, H. W., 1955, "The Hungarian Method for Assignment Problem", Naval Research Logistic Quarterly, 2, pp. 83-97.