Tasks Management Algorithm for Distributed System

A. Kumar[1] and P.K. Yadav[2]

[1] *Department of Mathematical Sciences and Computer Applications*
Bundelkhand University, Jhansi-284128, Uttar Pradesh (India)
dravanishkumar@yahoo.com
[2] *Central Building Research Institute, Roorkee-247667, Uttarakhand (India)*
prd_yadav@rediffmail.com

Abstract
The term "Distributed Real-Time Computing System" is described whenever number of computers interconnected in some fashion such that a program or procedure utilizes this distributed but combined power and gets executed in Real Time. The term has different meanings with regard to different systems, because processors can be interconnected in many ways for various reasons. In its most general form, the word distribution implies that the processors are fixed in geographically separated locations. Occasionally, the term is also applied to an operating environment using multiple mini-computers not connected with each other with the help of physical communication lines but are connected through satellite. In a Distributed Real Time System (DRTS), the single communication channel share by all the processors for Inter-Processor Communication (IPC). A program whose execution is distributed among several processors in a distributed system has the total processing cost equal to the sum of processors costs and Inter Tasks Communication Cost (ITCC), which are function of the amount of data transmitted. An optimal assignment is a distribution of modules that has lowest total Execution Cost (EC). The model discussed in this paper provides an optimal solution for assigning a set of "m" tasks to a set of "n" processors where m >> n, in such a way that allocated load on all processors is balanced according to the relative speed. Distributed Real-Time System finds extensive applications in the faculties, where large amount of data is to be processed in relatively short period of time, or where Real-Time computations are required. Meteorology, Cryptography, Image Analysis, Signal Processing, Solar and Radar Surveillance, Simulation of VLSI circuits and Industrial process monitoring are areas of such applications. These applications require not only very fast computation speeds but also different strategies involving distributed task allocation systems. In such applications the quality of the output is proportional to the amount of Real-Time computations. In a Distributed Real-Time System the execution of a program may be distributed among several processing elements to reduce the overall system cost by taking the advantage of heterogeneous computational capabilities and other resources within the system. For the optimal utilization of processors systematic allocation of task is necessary.

Keyword: Distributed real time system; Task assignment; Inter-task communication cost; Execution cost

1. Introduction

The term "Distributed Real-Time Processing System" (DRTS) is described whenever several computers interconnected in some fashion such that a program or procedure utilizes this distributed but combined power and gets executed in Real Time. The term has different meanings with regard to different systems, because processors can be interconnected in many ways for various reasons. In its most general form, the word distribution implies that the processors are fixed in geographically separated locations. Occasionally, the term is also applied to an operating environment using multiple mini-computers not connected with each other with the help of physical communication lines but are connected through satellite. A user-oriented definition have been discussed of distributed computing is "Multiple Computers, utilized cooperatively to solve problems" [1, 2]. Distributed Real-Time System finds extensive applications in the faculties, where large amount of data is to be processed in relatively short period of time, or where Real-Time computations are required. Meteorology, Cryptography, Image Analysis, Signal Processing, Solar and Radar Surveillance, Simulation of VLSI circuits and Industrial process monitoring are areas of such applications. These applications require not only very fast computation speeds but also different strategies involving distributed task allocation systems. In such applications the quality of the output is proportional to the amount of Real-Time computations. In a DRTS, the execution of a program may be distributed among several processing elements to reduce the overall system time by taking the advantage of heterogeneous computational capabilities and other resources within the system. For the optimal utilization of processors Systematic allocation of task is necessary. Allocation of tasks in a DRTS may be done by following two ways:

Static 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. The phrase "characteristics of the computation" means the ratios of the times that a program spends in different parts of the program. Thus

in a static allocation, one is interested in finding the assignment pattern that holds for the life time of a program, and result in the optimum value of the measure of effectiveness. These problems may be categorized in static [3 -13].

Dynamic Allocation: In order to make the best use of resources in a DRTS, it is essential to reassign modules or tasks dynamically during program execution, so as to the advantage of changes in the local reference patterns of the program [14-18].

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, 11, 13], reliability evaluation to deal with various design and allocation issues in a DPS by [24-34]. The main objective of this paper is to minimize the total program execution time by optimal utilization of processors in such a way the allocated load on all the processors should be balance. The developed model is programmed in Visual C++ and implemented the several sets of input data are used to test the effectiveness and efficiency of the algorithm. It is found that the algorithm is suitable for arbitrary number of processors with the random program structure.

2. Task Assignment Problem

The allocation of tasks in a DRTS is the fundamental requirement for optimal utilization of processors. The specific task allocation problem being addressed as follows: Considered an application program consists a set of "m" tasks $T = \{t_1, t_2,\ldots t_m\}$ and a DRTS consisting a set of "n" processors $P = \{p_1, p_2,\ldots p_n\}$, interconnected by communication links. The processor graph is a convenient abstraction of the processors together with interconnection network. 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 edge between processors p_i and p_j represent the delay involved in sending or receiving the message of unit length from one processor to another. In order to have an approximate estimate of this delay, 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 delay. 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. The ITCC is taken in the form of a symmetric matrix named as Inter Task Communication Cost Matrix [ITCCM (,)], which is of order m. The process of allocation problem can be formulated by a function Aalloc, from the set of tasks to the set of processors. Aalloc: $T \rightarrow P$, where Aalloc (i) = j, if the task t_i is assigned to processor p_j, $1 \leq I \leq m$, $1 \leq j \leq n$. The load balancing, which involves sending load from overloaded processors to under loaded processors, should be carried out with due regard for communication overhead so that it is accomplished as quickly 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 shall have to be balanced. Therefore, the systematic allocation of tasks in a DPS is the fundamental requirement for efficient execution of tasks. In distributed processing environments the services provided for 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 environments the program or tasks are also often developed with the subsets of independent units under various environments. While there are several components related to the tasks allocation problem, we are primarily concentrated to efficient utilization of processors and minimize the total program execution cost. The proposed methodology includes:

i. Identification of Initial Assignments
ii. Identification of Final Assignment
iii. Calculate the Execution Cost of each Processor
iv. Calculate the Inter Tasks Communication Cost of each Processor
v. Evaluation of Optimal busy time of the Distributed Real Time System

3. Definitions:

3.1 Execution Cost:

The execution cost e_{ij} Where $1 \leq i \leq m$, $1 \leq j \leq n$ of each task t_i depends on the processor p_j to which it is assigned and the work to be performed by each of tasks of that processor p_j.

3.2 Inter Tasks Communication Cost:

The Inter Task Communication Time C_{ik} of the interacting tasks t_i and t_k is incurred due to the data units exchanged between them during the process of execution

3.3 Response Time Cost:

The Response Time Cost (RTC) is a function of the amount of computation to be performed by each processor. This function is defined by considering the processor with the heaviest aggregate computation and communication loads. The degree to which communication latency will be hidden by overlapping computation with communication depends on such factors as the hardware attributes of the DRTS and the module scheduling methodology that is employed

4. Assumptions:

To keep the algorithm reasonable in size several assumptions have been made while designing the algorithm. A program is assumed to be collection of "m" tasks to be executed on a set of "n" processors, which have different processing capabilities. A task may be portion of an executable code or a data file. The number of tasks to be allocated is more than the number of processors (m >> n), as normally is the case in the real life. It is assumed that the execution cost of a task on each processor is known, if a task is not executable on any of the processor due to absence of some resources. The execution cost of that task on that processor is taken to be (∞) infinite. We assume that once a task has completed is execution on a processor, the processor stores the output data of the task in its local memory, if the data is needed by some another task being computed on the same processor, it reads the data from the local memory. The overhead incurred by this is negligible, so for all practical purposes we will consider it as zero. Using this fact, the algorithm tries to assign heavily communicating tasks to the same processor. Whenever groups of tasks or cluster are assigned to the same processor, the Inter tasks communication cost between them is zero. Completion of a program from computational point of view means that all related tasks have got executed.

5. Proposed Method

To determine the allocation, initially select those tasks "n" which has minimum ITCC and store the tasks in $T_{tasks}(j)$ (where j = 1,2,…,n) and also store the remaining m-n tasks in another linear array $TN_{tasks}(k)$ (where k = 1,2,…,m-n). Reduce the ECM(i,j) (where i = 1,2,…n and j = 1,2,…m) in n x n by deleting tasks stored in $TN_{tasks}(k)$ which is the intersection of $T_{tasks}(i)$. Determine the initial assignment apply the algorithm developed by Kumar et al [35]. The initial allocation is stored in an array $T_{ass}(j)$ (where j = 1,2,…,n) and also the processor position are stored in a another linear array ALLOC(j). Get the value of TTASK (j) by adding the value of ALLOC (j) if a task is assigned to a processor otherwise continues. The overall Execution time of a given allocation (Alloc) is then obtained by equation (1) and for each Processor Execution Time (PET) is defined by equation (2)

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

$$PEC(Aalloc)_j = \sum_{\substack{1 \le i \le m \\ i \in TS_J}} e_{i,Aalloc(i)} \quad (2)$$

Where TS_j= {i: Aalloc (i) = j, j=1, 2…n}

The Tasks stored in $TN_{tasks}(k)$ are restored in $T_{non\text{-}ass}(k)$ (where k = 1,2,…,m-n). The Mean Service Rate [MSR] of the processors in terms of $T_{ass}(j)$ is then calculated by using the equation (3) and store the results in MSR(j) (where j = 1,2,…,n).

$$MSR(j) = \frac{1}{EC(Aalloc)_j} \quad j=1, 2\ldots\ldots\ldots n \quad (3)$$

Once the MSR (j) is calculated, select the processor, which has maximum value of MSR say p_j i.e. fastest processor. If the value of MSR is equal or more than one processor then select a processor randomly for next assignment. Find out the tasks which is assigned to processor p_j (say t_i), and than select a task from $T_{non\text{-}ass}$ ( ) for assignment which has maximum communication with ti (say t_k). If the task tk has the value of EC on processor p_j is ≠ ∞ as, then assign the task t_k to the processor p_j. If the value of EC for tasks t_k on processor p_j = ∞, than select next heavely communicated tasks for assignment. After assigning the task modify the PEC and MSR of processor p_j by using the equation (1 & 2) respectively. Store the assignment and their position in T_{ass}() and ALLOC(j)) respectively and get the value of TTASK (j) by adding the value of ALLOC (j).Modify the $T_{non\text{-}ass}$() by deleting the tasks t_k. This process of assignment is continuing till the remaining "m-n" tasks are get allocated. The overall mean service time and throughout of the processors are calculated by using the equation (4) and (5) respectively. Store the results of mean service time and throughout in the linear arrays MST (j) and TRP (j), where j=1, 2…….,n respectively.

$$MST(j) = \frac{1}{MSR(j)} \quad (\text{where } j = 12,...n) \quad (4)$$

$$TRP(j) = \frac{TTASK(j)}{EC(Aalloc)_j} \quad j=1, 2\ldots\ldots n \quad (5)$$

where TTASK (j) is number of tasks assigned to processor p_j.

The overall inter-task communication Cost (ITCC) of a given allocation Aalloc is then calculated by equation (6) and for each Processor Inter-Task Communication Cost (PITCC) is then given by equation (7)

$$ITCC(Aalloc) = \sum_{\substack{1\le i\le m \\ i+1\le j\le m \\ Aalloc(i)\neq Aalloc(j)}} c_{Aalloc(i),Aalloc(j)} \quad (6)$$

$$PITCC(Aalloc)_j = \sum_{\substack{1\le i\le m \\ i+1\le j\le m \\ Aalloc(i)=j\neq Aalloc(k)}} c_{Aalloc(i),Aalloc(k)} \quad (7)$$

$$RTC(Aalloc) = \{EC(Aalloc)\ j + PITCC(Aalloc)\ j\} \quad (8)$$

4. Implementation of the model:

To justify the application and usefulness of the present method an example of a distributed real time system is considered consisting of a set of "n = 3" processors P = {p_1, p_2, p_3} connected by an arbitrary network. The processors only have local memory and do not share any global memory. A set of "m = 11" executable tasks T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}} which may be portion of an executable code or a data file. The execution time of each task on each processors has been taken in the form of ETM (,) of order m x n. The Inter tasks communication time between the tasks has been taken in the form of ITCCM (,) of order m.

Execution Cost Matrix = ECM(,) =

	p_1	p_2	p_3
t_1	6	3	5
t_2	4	2	3
t_3	3	1	2
t_4	5	2	∞
t_5	3	4	2
t_6	6	∞	6
t_7	5	6	7
t_8	∞	2	5

Inter Tasks Communication Cost Matrix =

ITCCM(,) =

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	3	4	2	6	8	1	0
t_2	3	0	0	0	0	0	0	5
t_3	4	0	0	4	3	2	0	0
t_4	2	0	4	0	5	3	2	5
t_5	6	0	3	5	0	0	0	0
t_6	8	0	2	3	0	0	6	8
t_7	1	0	0	2	0	6	0	5
t_8	0	5	0	5	0	8	5	0

After Implementation of the model following optimal assignment are obtained and shown in Table 1

Table 1:

Tasks	Processor
t_1	p_1
t_2	p_2
t_3	p_3
t_4	p_2
t_5	p_3
t_6	p_3
T_7	p_1
T_8	p_2

The optimization results from the algorithm ensure overall system cost as well as load on the processors are optimally balanced. Table 1 and Fig. 1 are shows the optimal assignment of tasks to the processors. Table 2 shows the final Results of the algorithm.

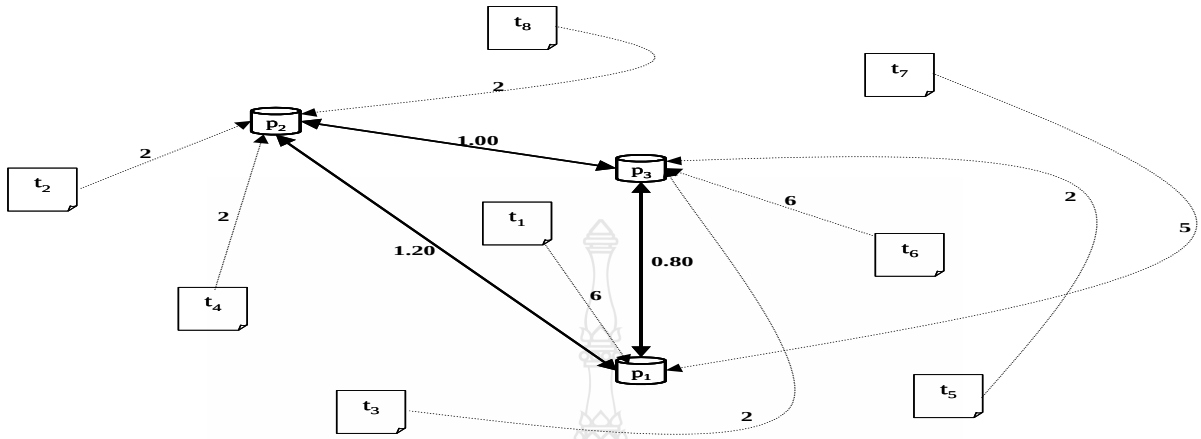

Fig. 1 Optimal assignment graph

Table 2: Processors wise EC and ITCC and total of EC and ITCC

Processors	EC	ITCC	Mean service rate	Throughput of the processors	Mean service time	TOTAL
1	2	3	4	5	6	(3+6)
p_1	11	36	0.091	0.182	10.989	37.989
p_2	6	30	0.167	0.501	5.988	35.988
p_3	10	44	0.100	0.300	10.000	54.000

The mean service rate and throughput of the processors are given in form of graph in Fig. 2. The Maximum busy time of the system is 54, which is related to processor p_3 depicted in Fig. 3.

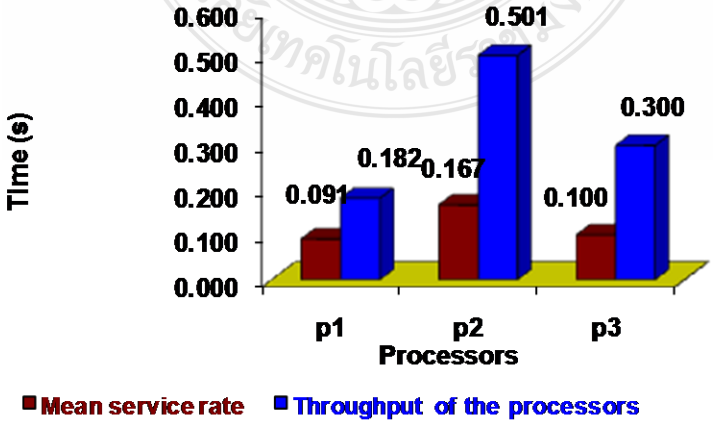

Fig. 2 Mean service rate and throughput of the processors

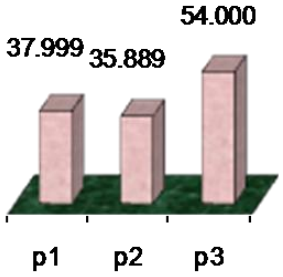

Fig. 3 Maximum busy time of the system

5. Conclusions

The present paper deals with a simple yet efficient mathematical and computational algorithm to identify the Optimal Allocation of tasks for evaluation of performance of the Distributed Real-Time Systems. A simple procedure has been developed to determine the following:

1. Systematic Allocation of tasks in DPS
2. Mean service rate,
3. Mean service time
4. Throughput of the processors

Table 1 shows that 2 tasks are executing on processor p_1 3 tasks are executing on p_3 and 2 tasks are executing on p_3. Table 2 shows that results of the algorithm form the table it is concluded that maximum busy time of the systems as 54 which is related to processor p_3. Therefore, the Optimal RTC of the DTRS is 54. Throughput of the processors is 0.182, 0.501 and 0.300. The average throughput of the system is 0.328.

The Performance of the algorithm is compared with [13]. The algorithm suggested in [13] is not considered the criteria of load balancing and proper utilization of each processor whereas our model considered both the issues. The run time complexity of the algorithm suggested by R.Y. Richard et al [22] is o (n^m) which to high and the show the problem is NP-Hard. The algorithm suggested by G. Sagar et al. [13] runs o (m^2n). The run complexity of the algorithm presented in this paper is o [$1/2(5m^2+2mn)$], which is much less then that of [13]. Table 3 and figure 4 represents the complexity comparisons of the algorithms.

Table 3: Results of run time complexity of the algorithms

Number of Tasks (m)	Number of processors (n)	Run time complexity of the algorithms	
		G. Sagar. et al. [13]	Present Model
5	3	75.0	78.0
6	3	108.0	108.0
7	4	196.0	151.0
8	4	256.0	192.0
9	5	405.0	248.0
10	5	500.0	300.0
11	6	726.0	396.0
12	6	864.0	432.0
13	7	1183.0	514.0
14	7	1372.0	588.0
15	8	1800.0	683.0
16	8	2048.0	768.0

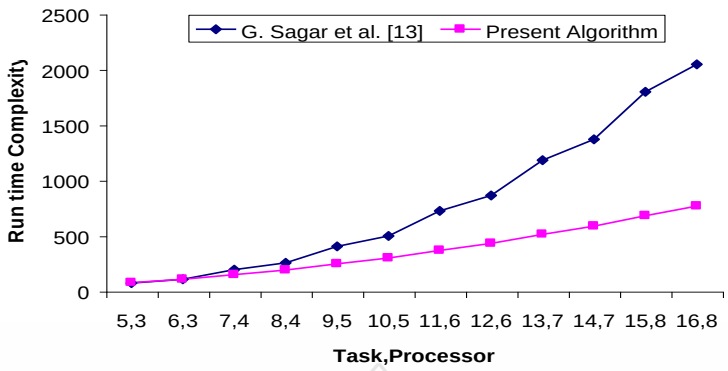

Fig. 4 Comparisons of the complexity of the algorithms

References

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

[2] B.R. Sitaram, "Distributed Computing – A User's View Point", CSI Communications, Vol.-18 No. 10, pp.26,28, 1965.

[3] Baca, D.F. (1989), Allocation Modules to Processor in a Distributed System, IEEE Transactions on Software Engineering, vol. SE-15, 1427-1436.

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

[5] Kumar, V. Singh, M.P. and Yadav, P.K. (1995), A Fast Algorithm for Allocating Tasks in Distributed Processing System, Proc. of the 30th Annual Convention of CSI, held at Hyderabad, India 347-358.

[6] Kumar, V. Singh, M. P. and Yadav, P.K. (1995), An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System, Proc. of the 19th National system conference, SSI, held at Coimbatore, India 82-87.

[7] 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, held at New Delhi, India 369-378.

[8] Singh, M.P., Kumar, V. and Kumar, A. (1999), An Efficient Algorithm for Optimizing Reliability Index in Tasks-Allocation, Acta Ciencia Indica, vol. xxv m, 437-444.

[9] Srinivasan, Santhanam and Jha. K. Niraj (1999), Safety and Reliability Driven Task Allocation in Distributed System, IEEE Transactions on Parallel and Distributed Systems, vol. 10, 238-250.

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

[11] Yadav, P. K. and Kumar, Avanish (2002), An Efficient Static Approach for Allocation through Reliability Optimization in Distributed Systems", presented at the International conference on Operations Research for Development (ICORD2002) held at Chennai.

[12] Zahedi, E., and Ashrafi, N. (1991), Software Reliability Allocation based on Structure, Utility, Price and Cost, IEEE Transactions on Software Engineering, vol. -17, 345-356.

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

[14] Bokhari, S.H. (1979), Dual Processor Scheduling with Dynamic Re-Assignment, IEEE Transactions on Software Engineering, vol. SE-5, 341-349.

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

[16] Kumar, Avanish (1999), Optimizing for the Dynamic Task Allocation", published to the proceedings of the III Conference of the International Academy of Physical Sciences held at Allahabad, 281-294.

[17] Kumar, V. Singh, M.P. and Yadav, P.K. (1996), An Efficient Algorithm for Multi-processor Scheduling with Dynamic Reassignment, Proc. of the 6th National seminar on theoretical Computer Science, held at Banasthally Vidyapeeth, India 105-118.

[18] Rotithor, H.G. (1994), Taxonomy of Dynamic Task Scheduling in Distributed Computing Systems, IEEE Proc. Computer Digit Tech., vol. 14, 1-10.

[19] Misra, K. B. and Sharma, U., (1991) An Efficient Algorithm to solve Integer Programming Problem arising in System Reliability Design, IEEE Transactions on Reliability, vol. R-40, 81-91.

[20] Bulfin, R. L. and Liu, C. Y. (1985), Optimal Allocation of Redundant Components for large Systems, IEEE Transactions on Reliability, vol. R-34, 241-247.
[21] Chu, W.W. (1969), Optimal File Allocation in a Multiple Computing System, IEEE Transactions on Computer, vol. C-18, 885-889.
[22] Richard R.Y., Lee, E.Y.S. and Tsuchiya, M. (1982), A Task Allocation Model for Distributed Computer System, IEEE Transactions on Computer, vol. C-31, 41-47.
[23] Dessoukiu-EI, O.I. and Huna, W. H., (1980), Distributed Enumeration on Network Computers, IEEE Transactions on Computer, vol. C-29. 818-825.
[24] Fitzgerald, Kent, Latifi, Shahram and Srimani, Pradip K. (2002), Reliability Modeling and Assessment of the Star-Graph Networks, IEEE Transactions on Reliability, vol. R-51, 49-57.
[25] Fyffe, D.E., Hines, W. W. and Lee, N. K. (1968), System Reliability Allocation and Computational Algorithm, IEEE Transactions on Reliability, vol. R-17, 64-69.
[26] Ghare, P. M. and Taylor, R. E. (1969), Optimal Redundancy for Reliability in Series Systems, Operational Res., vol. 17, 838-847.
[27] Kumar, Avanish (2001), An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost", published to the proceedings of International Conference on Mathematical Modeling held at Roorkee, 150-155.
[28] Lin, Min-Sheng (2002), A Linear-time Algorithm for Computing K-terminal Reliability on Proper Interval Graphs, IEEE Transactions Reliability, vol. R-51, 58-62.
[29] Lyu, Michael R., Rangarajan, Sampath and Moorsel, Aad P. A. Van (2002), Optimal Allocation of test Resources for Software Reliability growth modeling in Software Development, IEEE Transactions on Reliability, vol. R-51, 183-192.
[30] Nakagawa, Y. and Miyazaki, S. (1981), Surrogate Constraints Algorithm for Reliability Optimization Problems with two Constraints, IEEE Transactions on Reliability, vol. R-30, 175-181.
[31] Ormon, Stephen W., Cassady, C. Richard and Greenwood, Allen G. (2002), Reliability Prediction model to Support Conceptual Design, IEEE Transactions on Reliability, vol. R-51, 151-157.
[32] Painton, L. and Campbell, J. (1992), Genetic Algorithm in Optimization of System Reliability, IEEE Transactions on Reliability, vol. R-44, 172-178.
[33] Peng, Dar-Tezen, Shin, K. G. and Abdel, Zoher T. F. (1997), Assignment Scheduling Communication Periodic Tasks in Distributed real time System, IEEE Transactions on software Engg. vol. SE-13, 745-757.
[34] Ramesh, Anapathur V., Twigg, David W., Sandadi, Upender R. and Sharma, Tilak C. (2002), Reliability Analysis of System with Operation Time Management, IEEE Transactions on Reliability, vol. R-51, 39-48.
[35] Yadav, P. K., Kumar, Avanish and Singh, M. P., "An Algorithm for Solving the Unbalanced Assignment Problems", International Journal of Mathematical Sciences, Vol. 12(2), pp. 447-461 2004.