



AN OPTIMAL APPROACH FOR THE TASKS ALLOCATION BASED ON THE FUSION OF EC AND ITCC IN THE DISTRIBUTED COMPUTING SYSTEMS

Abhilasha Sharma

DIT University, Dehradun, Uttarakhand, India

Surbhi Gupta

AMITY University, Noida, Uttar Pradesh, India

Abstract: Distributed computing systems (DCS) are of current interest due to the advancement of microprocessors technology and computers networks. It consists of multiple computing nodes that communicate with each other by message passing mechanism. The advancement of the new technologies in communication and information lead to the development of the Distributed System. The task allocation is an essential phase in the Distributed computing systems. We consider the problem of m tasks and n processors (where $m \gg n$). In this paper a heuristic model is presented, which performs static allocation of a set of " m " tasks of a program to a set of " n " processors (where, $m \gg n$) with the constraints of minimizing Inter Task communication Cost (ITCC) and maximize the overall throughput of the system in such a way that allocated load on all the processors should be balanced. While designing the algorithm the Execution Cost (EC) and ITCC have been taken into consideration.

Keywords: Distributed Computing Systems (DCS), Task Allocation, Execution Cost, Inter Task Communication Cost.

I. Introduction

Distributed Computing systems have become popular due to the advances in computer hardware and interconnection network technology. Distributed processing 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. The performance of any Distributed Processing System may suffer if the distribution of resource is not carefully implemented. In order to make the best use of the resources in a DCS, it becomes essential to maximize the overall throughput by allocating the tasks to processors in such a way that the allocated load on all the processors should be balanced. Partitioning of the application software into a number of small groups of modules among dissimilar processors is important parameter to determining the efficient utilization of available resources. It also enhances the computation speed. The task partitioning and task allocation activities influence the distributed software properties such as IPC [1, 2]. Many approaches have been reported for solving the task assignment problem in DCS. The problem of finding an optimal dynamic assignment of a modular program for a two-processor system is analyzed by [3]. One measure of usefulness of a general-purpose distributed computing system is the system's ability to provide a level of performance commensurate to the degree of multiplicity of resources present in the system. Taxonomy of approaches to the resource management problem is reported [4]. The taxonomy, while presented and discussed in terms of *distributed scheduling*, is also applicable to most types of resource management. A model for allocating information files have been reported by [5] the model considers storage cost, transmission cost, file lengths, and request rates, as well as updating rates of files, the maximum allowable expected access times to files at each computer, and the storage capacity of each computer. The model is formulated into a nonlinear integer zero-one programming problem, which may be reduced to a linear zero-one programming problem. Known solutions of a large number of important (and difficult) computational problems called NP-complete problems depend on enumeration techniques which examine all feasible alternatives. The design of enumeration schemes in a distributed environment have been reported by [6]. A task allocation model that allocates application tasks among processors in distributed computing systems satisfying: 1) minimum inter-processor communication cost, 2) balanced utilization of each processor, and 3) all engineering application requirements has been reported by Perng-Yi Richard Ma et.al [7]. This problem of task allocation in heterogeneous distributed systems with the goal of maximizing the system reliability has been addresses [8]. The model is based on the well-known simulated annealing (SA) technique. Yadav et al have reported an algorithm for Reliability Evaluation of Distributed System Based on Failure Data Analysis [9]. An Efficient Algorithm for Optimal Tasks Allocation through Optimizing Reliability Index in Heterogeneous Distributed Processing

System has been discussed by [10]. J. B. Sinclair[11] considers the problem of finding an optimal assignment of the modules of a program to processors in a distributed system. A module incurs an execution cost that may be different for each processor assignment, and modules that are not assigned to the same processor but that communicate with one another incur a communication cost. An optimal solution to the problem of allocating communicating periodic tasks to heterogeneous processing nodes (PNs) in a distributed real-time system have been reported [12]. Matrix reduction technique has been used by Sagar et al according to the criteria given therein a task is selected randomly to start, with and then assigned to a processor [13]. A fast algorithm for allocation task in distributed processing system has been reported by Kumar et al [14, 16] in this method the author try to clustered heavily communicated tasks and allocate them to same processor. An efficient Algorithm for allocating tasks to processors in a distributed system has also been reported by Kumar et al. [15]. Distributed computing systems offer the potential for improved performance and resource sharing. To make the best use of the computational power available in multi processing system, it is essential to assign the tasks dynamically to that processor whose characteristics are most appropriate for the execution of the tasks in distributed processing system [17, 18]. The new methodology augments the Maximally Linked Module concept by using stochastic techniques and by adding constructs which take into account the limited and uneven distribution of hardware resources often associated with heterogeneous systems has been discussed by Elsadde et al. Authors used pure simulated annealing and the randomized algorithm to randomly-generated systems and synthetic structures which were derived from real-world problems [19]. Communication technology has also opened many avenues for processing, sharing and transferring of data Yadav, et al [20] reported task scheduling in computer communication network. An Artificial Neural Network (ANN) base task scheduling model has been discussed by Yadav et al [21] while developing the model authors use feedback neural network architecture. Sing et al reported an ANN based model for Load Distribution in fully connected client server Network [22]. Analysis of Load Distribution in Distributed Computing systems 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 [23, 24].

The main objective of this paper is to minimize the total program execution cost. A heuristic model is presented, which performs static allocation. While designing the algorithm it is assumed that a program is a collection “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 distributed processing environment. It is assumed that the EC 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 EC of that task on that processor is taken to be (∞) infinite. We assume that once a task has completed its 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 a group of tasks is assigned to the same processor, the ITC cost between them is zero. The proposed method is based upon the matrix reduction techniques. The performance of the algorithm have been tested by generating the random data and found that the algorithm is suitable for arbitrary number of processor with the random program structure and workable in all the cases.

II. Task Assignment Problems

The specific problem being addressed is as follows: Given application software that consists of “m” communicating tasks, $T = \{t_1, t_2, \dots, t_m\}$, and a heterogeneous distributed computing system with “n” processors, $P = \{p_1, p_2, \dots, p_n\}$, where it is assumed that $m \gg n$, assign (allocate) each of the “m” tasks to one of the “n” processors in such a manner that the ITCC time is minimized and the processing load is balanced. The processing cost of these tasks on all the processors and ITCC between the tasks are taken in the form of Execution Cost Matrix [ECM (,)] of order $m \times n$, and Inter Task Communication Cost Matrix [ITCC(,)] of order m respectively and will be used throughout this text.

Definitions:

Execution Cost: The execution cost ec_{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 . The processing Execution cost EC of the tasks on all the processors is given in the form of Execution Cost Matrix (ECM) of order $m \times n$. The Execution Cost of a given assignment on each processors are calculated by Equation (1)

$$PEC(j) = \sum_{i=1}^n ec_{ij} x_{ij}, i = 1, 2, \dots, m \quad (1)$$

Where x_{ij} is the

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

Inter Task Communication Cost: The Inter Task Communication cost ec_{ik} of the interacting tasks t_i and t_k is the minimum cost required for the exchange of data units between the processors during the process of execution.

$$ITCC(j) = \sum_{i < k} c_{ik} x_{ij}, j = 1, 2, \dots, n \quad \text{where } x_{ij} \text{ is the } \begin{cases} 1, & \text{if task } t_i \text{ is assign to processor } p_j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Response Time (RT) of the System: Response time of the system is a function of amount computation to be performed by each processor and the computation time. This function is defined by considering the processor with the heaviest aggregate computation and communication load. Response time of the system for a given assignment is defined by $RT(Aalloc) = \max_{1 \leq j \leq n} \{PEC(Aalloc)_j + IPC(Aalloc)_j\}$ (3)

Average load: For each node there are N values representing the execution cost required for the corresponding module to be processed on each of the N processors. These values are in the matrix ECM . Each edge is labeled with a value that represents the communication cost needed to exchange the data when the modules reside on different processors. The total workload W is the summation of the maximum module execution cost on the different processors as shown

$$Lavg(P_j) = \frac{W_j}{m}, j=1, 2, \dots, n \quad \text{Where} \quad (4)$$

$$W_j = \sum_{1 \leq i \leq m} et_{i,j} \quad j=1, 2, \dots, n$$

$$Tlod = \sum_{j=1}^n Lavg(P_j) \quad (5)$$

The average load on a processor $Lavg(P_j)$ depends upon the different tasks on each processor. The system is considered to be *balanced* if the load on each processor is equal to the processor average load within a given (small percentage) tolerance.

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 ($m \gg n$), as normally is the case in the real life. It is assumed that the execution time 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 time of that task on that processor is taken to be (∞) infinite. We assume that once a task has completed its 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. 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 data transfer between them is zero. Completion of a program from computational point of view means that all related tasks have got executed.

III. The Proposed Method

Initially the average load on the processors P_j is obtained by using equation (4) and the total load may be calculated by using the equation (5). The load on each processor is equal to the average load within a reasonable tolerance. In the present study a tolerance factor of 20% of average load has been considered. Determine the “n” Minimally Linked Task (MLT) by using the equation (6) and store the result in two-dimensional array $MLT(,)$ the first column represents the task number and second column represents the sum of ITCC of task t_i with all task t_{k-i} . Rearrange the $MLT(,)$ in ascending order assuming the second column as sorted key.

$$MLT(i, k) = \sum_{i,k=1}^m cc_{ik} \quad (6)$$

Augmented $ECM(,)$ by introducing the $MLM(,)$ and sort $ECM(,)$ in increasing order considering MLT as sorting key. To determine initial allocation apply the Yadav et al algorithm [24] and store in an array $T_{ass}(j)$ (where $j = 1, 2, \dots, n$). The processor position are also store in a another linear array $A_{alloc}(j)$. The value of $T_{TASK}(j)$ is also computed by adding the values of $A_{alloc}(j)$ if a task t_i is assigned to processor p_j otherwise continue. The remaining ($m-n$) task are then store in a liner array $T_{non_ass}(,)$. Tasks assigned to processors p_j and stored in $T_{non_ass}(,)$ which are obviously:

$$T = T_{ass} \cup T_{non_ass}$$

All the tasks stored in $T_{non_ass}()$ fused with those assigned tasks stored in $T_{ass}()$ on the bases of minimum average of EC and ITCC.

The Fused Execution Cost (FEC) of a task $t_a \in T_{non_ass}()$ with some other task $t_i \in T_{ass}()$ on processor p_j is obtained as:

$$FEC(j)_{ai} = [ec_{aj} + ec_{ij}] \quad (1 \leq a \leq m, 1 \leq i \leq m, 1 \leq j \leq n, i \neq a)$$

Let cc_{ai} be the ITCC between $t_a \in T_{non_ass}()$ and $t_i \in T_{ass}()$. Fused Inter Task Communication Cost (FITCC) for t_a with t_i is computed as:

$$FITCC(j)_{ai} = \sum_{i | t_i \in T_{ass}()} [cc_{ai}]$$

Here, $c_{ai} = 0$ if fused with t_i or $a = i$ and remaining c_{ai} value are added and Minimum Average Fused Cost (MAFC) is calculated as follows:

$$MAFC(j)_{ai} = \min \{ (FEC_{a1} + FITCC_{a1}), (FEC_{a2} + FITCC_{a2}), \dots, (FEC_{an} + FITCC_{an}) \}$$

This process will be continued until all the tasks stored in $T_{non_ass}()$ are fused. After complete allocation is achieved $PEC(j)$ and $ITCC(j)$ by using equation (1) and (2) respectively. Finally, Calculate the Total Optimal Cost (TOC) by summing up the values of $PEC(j)$ and $ITCC(j)$, and store the result in a linear array Over TOC (j) where $j = 1, 2, \dots, n$. The maximum value of TOC (j) will be the optimal cost of the system:

$$TOC(j) = \max \{ PEC(j) + ITCC(j) \}$$

The Mean Service Rate [MSR] of the processors in terms of $T_{ass}(j)$ to be computed as and store the results in $MSR(j)$ (where $j = 1, 2, \dots, n$).

$$MSR(j) = \frac{1}{PEC(j)} \quad (\text{where } j = 1, 2, \dots, n)$$

The overall throughput of the processors are calculated as and store the results of throughput in the linear arrays $T_{RP}(j)$, where $j = 1, 2, \dots, n$

$$T_{RP}(j) = \frac{T_{TASK}(j)}{PEC(j)} \quad (\text{where } j = 1, 2, \dots, n)$$

IV. Results and Discussions

Input: Data required by the Algorithm is given below:

Number of processors available in the system (n) = 3
Number of tasks to be executed (m) = 4

$$ECM(.) = \begin{matrix} & p_1 & p_2 & p_3 \\ \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix} & \begin{bmatrix} 8 & 4 & 6 \\ 6 & 5 & 4 \\ 8 & 6 & 5 \\ 4 & 7 & 3 \end{bmatrix} \end{matrix} \quad ITCCM(.) = \begin{matrix} & t_1 & t_2 & t_3 & t_4 \\ \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix} & \begin{bmatrix} 0 & 100 & 3 & 5 \\ 100 & 0 & 6 & 4 \\ 3 & 6 & 0 & 5 \\ 5 & 4 & 5 & 0 \end{bmatrix} \end{matrix}$$

Actual average load to be assign to the processor after introducing the 20 % Tolerance Factor (TF) the average load may be assign to the processors is calculated and same may be store in linear array $L_{AVG}()$

$$L_{AVG}() = \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \left| \begin{matrix} 10 \\ 09 \\ 07 \end{matrix} \right|$$

Determine the Minimally Link Task (MLT) and store the result in MLT (.) as follows:

$$MLM(.) = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix} \left| \begin{matrix} 108 \\ 110 \\ 14 \\ 14 \end{matrix} \right|$$

Augmented ECM (.) by introducing the MLM () and sort ECM (.) in increasing order considering MLT as sorting key.

$$ECM(.) = \begin{matrix} & p_1 & p_2 & p_3 & MLT \\ t_1 & \left(\begin{matrix} 8 & 4 & 6 \end{matrix} \right) & 108 \\ t_2 & \left(\begin{matrix} 6 & 5 & 4 \end{matrix} \right) & 110 \\ t_3 & \left(\begin{matrix} 8 & 6 & 5 \end{matrix} \right) & 14 \\ t_4 & \left(\begin{matrix} 4 & 7 & 3 \end{matrix} \right) & 14 \end{matrix} \quad ECM(.) = \begin{matrix} & p_1 & p_2 & p_3 & MLT \\ t_3 & \left(\begin{matrix} 8 & 6 & 5 \end{matrix} \right) & 14 \\ t_4 & \left(\begin{matrix} 4 & 7 & 3 \end{matrix} \right) & 14 \\ t_1 & \left(\begin{matrix} 8 & 4 & 6 \end{matrix} \right) & 108 \\ t_2 & \left(\begin{matrix} 6 & 5 & 4 \end{matrix} \right) & 110 \end{matrix}$$

Apply Yadav et al [Yada04] algorithm to determine the initial allocation The initial allocation than store in an linear array $T_{ass}()$ and the processor position are store in $A_{alloc}()$. The remaining m-n tasks are stored in $T_{non_ass}()$. The results are asfollows:

$$\begin{aligned} T_{ass}(j) &= \{t_1, t_3, t_4\} \\ A_{alloc}(j) &= \{p_2, p_3, p_1\} \\ T_{non_ass}(k) &= \{t_2\} \end{aligned}$$

Tasks	Processors	$FEC(j)$
t_2+t_1	p_2	9
t_2+t_3	p_3	9
t_2+t_4	p_1	10

After getting the initial allocation the a task stored in $T_{non_ass}()$ has been selected for assignment i.e. $t_2FEC(j)$, for t_2 with all the stored in $T_{ass}()$ are
 $FITCC(j)$ for t_2 with the other assigned tasks stored in $T_{ass}()$ are:

Tasks	Processors	$FITCC(j)$
t_2+t_1	p_2	18
t_2+t_3	p_3	112
t_2+t_4	p_1	116

$MAFC(j)$ by summing up the value of $FEC(j)$ and $FITCC(j)$ are:

Tasks	Processors	$FEC(j)$	$FITCC(j)$	$MAFC(j)$
t_2+t_1	p_2	9	18	27
t_2+t_3	p_3	9	112	121
t_2+t_4	p_1	10	116	126

Table 1: Final Results

Processors	PEC(j)	ITCC(j)	TOC(j)	MSR(j)	$T_{RP}(j)$
p_1	4	14	18	0.056	0.065
p_2	9	18	27	0.074	0.037
p_3	5	14	19	0.053	0.053

The $MAFC(2)$ is minimum i.e. 27. Therefore, task t_2 is fused with Task t_1 executing on processor p_2 . Compute PEC (j), ITCC (j), TOC (j), MSR (j) and $T_{RP}(j)$ given in table 1.

The maximum of $TOC(j)$ is 27 i.e. the total busy cost of the system is 27 which is corresponds to processor p_2 . The model deals with the problem of optimal task allocation and load balancing in DCS. The load balancing mechanism is introduced in the algorithm by fusing the unallocated tasks of the basis of minimum of the average impact of EC and ITCC. It can also be perceived from the example presented here that wherever the algorithm of better complexity is encountered [19] 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 [19] is $O(n^2 + m^2 + m^2n + m + n)$ and the run complexity of the algorithm presented in this paper is $O(m^2n + 2mn)$. Figure 1 shows the run time complexity of present methods and [19] considering different cases of tasks and processors. The utility of our algorithm will ultimately depend on how it can be incorporated into the designing and managing activities of distributed computing system.

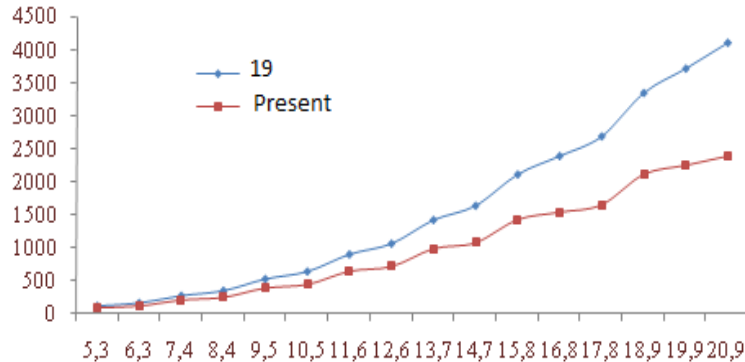


Figure 1: Run time complexity compression

REFERENCES

- [1]. SHATZ, S.M., and WANG, J.P., "Introduction to distributed software engineering", Computer, 1987,20 (10), pp. 23-31.
- [2]. D.F. Baca, "Allocation Tasks to Processor in a Distributed System," IEEE Trans. On Software Engineering, Vol.15 pp.1427-1436, 1989.
- [3]. S.H. Bokhari, "Dual Processor Scheduling with Dynamic Re-Assignment", IEEE Trans. On Software Engineering, Vol.SE-5 pp. 341-349, 1979.
- [4]. T.L. Casavant, and J.G. Kuhl, "A Taxonomy of Scheduling in General Purpose Distributed Computing System", IEEE Trans. On Software Engineering, Vol.14 pp.141-154, 1988.
- [5]. W.W. Chu, "Optimal File Allocation in a Multiple Computing System", IEEE Trans. On Computer, Vol.C-18 pp.885-889, 1969.
- [6]. O.I. Dessouki-EI and W.H. Huna, " Distributed Enumeration on Network Computers," IEEE Trans. On Computer, Vol.C-29 pp.818-825, 1980.
- [7]. Ma, P.-Y. R., Lee, E. Y. S., & Tsuchiya, M.. A Task Allocation Model for Distributed Computing Systems. Computers IEEE Transactions on, C-31(1), 41-47, 1982.
- [8]. Gamal Attiya and Yskandar Hamam, "Task allocation for maximizing reliability of distributed systems: A simulated annealing approach", Journal of Parallel and Distributed Computing, Volume 66, Issue 10, Pages 1259-1266, 2006
- [9]. Yadav P. K. , Bhatia K. and Gulati Sagar ,Reliability Evaluation of Distributed System Based on Failure Data Analysis, International Journal of Computer Engineering, Vol 2(1) pp.113-118, 2010.
- [10]. Singh M.P.,Kumar Harendra ,Yadav P.K." An Efficient Algorithm for Optimal Tasks Allocation through Optimizing Reliability Index in Heterogeneous Distributed Processing System" in "3rd International Conference on Quality, Reliability and Infocom Technology" held at Indian National Science Academy, New Delhi –India, December 2-5,2006.
- [11]. J.B. Sinclair, "Efficient computation of optimal assignments for distributed tasks," J. parallel Dist. Comput., vol.4, pp.342-362, 1987.
- [12]. Peng, Dar-Tezen; K.G. Shin, Abdel, T.F. Zoher, "Assignment Scheduling Communicating Periodic Tasks in Distributed Real-Time System," IEEE Trans. On software Engg., vol.13, no.12, pp.745-757, 1997.
- [13]. G. Sagar, and A.K. Sarje, "Task allocation model for distributed system," Int. J. system science, vol. & no. 22, 9, pp. 1671-1678, 1991.
- [14]. Vinod Kumar, M.P. Singh, P.K. Yadav, "A fast algorithm for allocation task in distributed processing system," proceedings of the 30th Annual convention of computer society of India, Hyderabad, pp.347-358, 1995.
- [15]. Vinod Kumar, M.P. Singh, P.K. Yadav, "An efficient Algorithm for allocating tasks to processors in a distributed system," Proceedings of the Nineteenth National system conference (NSC-95), PSC College of technology, Coimbatore, organized by system society of India, New Delhi, pp. 82-87, 1995.
- [16]. Vinod Kumar, P.K. Yadav, and K. Bhatia, "Optimal Task Allocation in Distributed Computing Systems Owing to Inter Task Communication Effects." Proc. Of CSI-98 : IT for the New Generation, pp.369-378, 1998.
- [17]. Vinod Kumar, M.P. Singh, P.K. Yadav, "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, 1996.
- [18]. Yadav, P. K., Kumar, Singh, M. P, and Harendra Kumar "Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with dynamic Reassignment" International Journal of Computer System, Network and Communication, Vol. (2008), pp. 1-9 2008.
- [19]. Elsadde A.A.,Wells B.E. "A Heuristic Model for Task Allocation in Heterogeneous Distributed Computing System" The International Journal of Computers and Their Applications. Vol.6, no.1, March 1999.
- [20]. Yadav, P. K., Singh Jumindera and Singh, M. P "An efficient method for task scheduling in computer communication network" , International Journal of Intelligent Information Processing Vol. 3(1) pp 81-89, 2009
- [21]. Yadav P.K. , Singh M.P. , Kumar Avanish and Agarwal Babita, "An Efficient Tasks Scheduling Model in Distributed Computing systems Using ANN", International Journal of Computer Engineering, Vol 1(2) pp.57-62, 2009.
- [22]. Singh.M.P, Kumar Avanish, Yadav, P.K. and Krishna Garima, " Study of Load Distribution in fully connected client server Network using feedback neural network architecture", Journal of Engineering and Technology Research, Vol 2(4) pp. 58-72, 2010
- [23]. Kumar Avanish, Yadav P.K., and Sharma Abhilasha "Analysis of Load Distribution in Distributed Computing systems Through Systematic Allocation Task" IJMCSIT "International, Journal of Mathematics, Computer Science and Information Technology", Vol 3 (1), pp.101-114, 2010

- [24]. *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.*