International Journal of Mathematical Archive-8(2), 2017, 30-38
Available online through www.ijma.info ISSN 2229 – 5046

AN EFFECTIVE JOBS PROGRAMMING EXAMPLE IN DISTRIBUTED TREATING ORGANIZATIONS EMPLOYING ANN

AKHIL KUMAR*[1], Prof. D. P SINGH[2]

[1]Research Scholar Pacific University, Udaipur – (R.J.), India.

[2]Prof in RBCET Bareilly, India.

(Received On: 23-01-17; Revised & Accepted On: 17-02-17)

ABSTRACT

The individual channel is shared by all the processors for Inter-Processor Communication (IPC) in an administered organization. A program whose executing is broadcast amongst several C.P.U.s in a broadcast organization has the total actioning cost equal to the sum of Executing Costs and Inter-Processor Communicating, which are occasion of the number of information broadcast. An optimum appointment is a dispersion of jobs that has lowest total executing cost. The neural specification is used to form jobs clusters between maximally linked tasks and to restrict the clusters size, the average load to be assigned to each processor. The Artificial Neural Network (ANN) based model has been discussed in this paper. The model provides near to optimal solution for assigning a set of "m" tasks of a program to set of "n" processors where "m >> n" in such a way that allocated load on all processors is balanced according to the average load. For testing the efficiency of the algorithm, several sets of input data have been considered for all program categories, and it is found that our algorithm resulted optimal allocation in most of the cases.

Keywords: *Distributed system, inter-processor communication, execution cost, artificial neural network, and optimization.*

1. INTRODUCTION

Distributed arrangement is a computing system in which multiple CPUs associated together via the high band-width communicating associates. These connections furnish a medium for each central processor to accession information and plans on outside processors. Dispersion of resourcefulness in broadcast organization is seen as a way to amend organization outturn and availability. A significant imagination for administered organization is user's program that comprises of a set of jobs.

Response time is the significant argument for evaluating functioning of broadcast organization. A distributed system is designed for figuring out some particular real time coatings. The organization is required to efficient scheduling of the task in computer communication network and finish a certain task within a specific time limit [15]. The excessive Inter-Processor Communication is always most costly and least reliable factor in the loosely coupled distributed system. Therefore, an efficient strategy is required for optimal utilization of processor's capacity in distributed system. For optimal utilization of processor's capacity it is essential to denigrate IPC that develops when the acting tasks reside on different processors. Only denigrating the IPC alone may not furnish a good result. A processor's load designates the sum of carrying out times of the jobs residing on that CPU. Equilibrating the load on CPUs and denigrating the IPC can denigrate the reaction time of the establishment.

Several approach paths have been described for figuring out the job allotment problem in broadcast system. Program breakdown into the small jobs and their allotments on the broadcast computing system for marching play the significant role in employing the broadcast system power. The task breakdown, allocation and coarseness [1] actions determine the distributed software ownerships such as Inter Processors Communication (IPC) price and potential for correspondence. The mode in which this division is done decides the efficiency of a given application program when it is performed on circulated computing systems.

Corresponding Author: Akhil Kumar*[1]
[1]Research Scholar Pacific University, Udaipur – (R.J.), India.

So an effective job allotment scheme is commanded for the thoroughly usage of computational imaginations and minimization of IPC. Various advances to solve the trouble of job appointment in allotted organization have been arose but most of these method acting administers with homogeneous systems and the complication of these algorithmic program gain quickly as the problem size heightened. Some of the heuristic program example for task allotment in administered computer systems has been talked about by [2-4]. The jobs allotment example based on graph theoretical access has been acquired by Hesham H. Ali and Hasham EI-Rewini [5] for NP-hard trouble. An example was talked about by Chu, T., and Abraham J [6] conceiving the load equilibrating through taxonomical programming of tasks in distributed computing surroundings. Chu, W., Holloway, L., Lan, M., and Efe, K., [7] talked about a task allotment example for distributed data processing. A jobs allotment model arose by Ma, P., Lee, E. and Tsuchiya, M [8] employing branch & bound method acting. An Algorithmic program for figuring out the Unbalanced Appointment Jobs has been arose by [9]. Yadav, P.K., Kumar, Avanish and Gupta, A. R [10] has furnished the answer for task allotment in distributed system. Cost Based Static Approach path for Operation Enhancement of Broadcast Networks has been talked about by [11]. Improved Dependability of Broadcast Computing Systems using Mathematical Scheduling has been talked about by [12]. Tasks programming algorithmic program for Multiple CPUs system with Active Re-appointment has been lately acquired by [13].

By equilibrating load on the C.P.U.s, it is potential to get effective usage of available computational ability. In this research paper, authors demonstrate a task allotment algorithmic program with the help of ANN working in union to figure out particular troubles.

Neural networks litigate data in kind the human brain does. The electronic network is wrote of a large number of extremely interconnected marching components (neurons) working in analogue to figure out a particular trouble.

An artificial neuron is a device with various inputs and one output. The neuron acts in two manners of functioning; the aiming mode and the using manner. In the aiming mode, the neuron can be aimed to fire (or not), for especial input forms. In the using mode, when a instructed input pattern is discovered at the input, its linked output becomes the current yield. If the input pattern does not belong in the instructed list of input conventions, the firing rule is used to determine whether to fire or not.

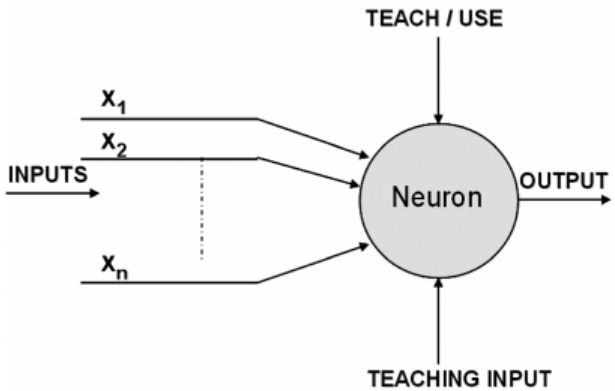

Figure 1: Elementary Nerve cell

Feed-forward ANNs appropriate indicates to go one way only; from input to output. There is no resubmit (loops) i.e. the output of any level does not impress that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. In this paper we use feed-forward ANN to assign tasks to the processor and find the near optimal solution. Neural networks do not perform miracles. But if used reasonably they can bring about some awesome answers.

The allotment is done before the existent run of the application program. It is assumed that the execution cost of the subtasks and the IPC cost, which arise due to the interacting subtasks residing on different processors are known. It apportions the modules or subtasks of an application problem as evenly as possible among the C.P.U.s and tries to denigrate the IPC cost.

2. JOB ALLOTMENT TROUBLE

We ideate a distributed arrangement as accumulated as a set of "*n*" actioning nodes $P = \{p_1, p_2, \ldots p_n\}$ and an interconnection structure furnishing full connectivity amongst the nodes and an application comprises a set of "*m*" tasks $T = \{t_1, t_2, \ldots t_m\}$ to be accomplished on these marching nodes. An allotment of tasks to C.P.U.s is defined by a function Aalloc (), from the set of jobs to the set of C.P.U.s.

Aalloc: T P, where A*alloc* (*i*) = *j* If the task *ti* is assigned to processor *Pj*, 1 £ *i* £ *m*, 1 £ *j* £ *n*.

3. DEFINITIONS

3.1. Executing Cost

The execution cost *eij*, is the amount of the work to be performed by the executing task *ti* on the processor *pj*. The overall execution cost (EC) of a given allocation (*Aalloc*) and execution time for each processor (PEC) are calculated by using equation (1) and (2) respectively.

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

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

where $TS_j=\left\{i: Aalloc(i)=j, \quad j=1,2...n\right\}$

3.2. Inter Processor Communicating Cost

The inter processor communicating cost *cik* is obtained due the data bounds replaced between the accomplishing task *ti* and occupying task *tk* if they are on unlike processors. The overall inter-C.P.U. communicating (IPC) cost of a given allotment Aalloc is then calculated by equation (3) and for each Processor Inter-Processor Communicating (PIPC) cost is then given by equation (4).

$$IPC\left(Aalloc\right)=\sum_{\substack{1\le i\le m\\ i+1\le j\le m\\ Aalloc(i)\ne Aalloc(j)}} c_{Aalloc(i),Aalloc(j)} \tag{3}$$

$$PIPC\left(Aalloc\right)_j=\sum_{\substack{1\le i\le m\\ i+1\le j\le m\\ Aalloc(i)-j\ne Aalloc(k)}} c_{Aalloc(i),Aalloc(k)} \tag{4}$$

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

3.3. Response Time (RT) of the Arrangement

Response time of the arrangement is a function of quantity computing to be executed by each processor and the computation time. This function is defined by conceiving the processor with the heaviest aggregate calculation and communication load. Response time of the system for a given assigning is defined by

$$RT(Aalloc)=\max_{1\le j\le n}\left\{EC(Aalloc)_j+PIPC(Aalloc)_j\right\} \tag{5}$$

3.4. Maximally Associated Job

A job *tk* in a program graph is called a maximally associated task if the sum of IPC costs of *tk* with other jobs is larger than the sum of IPC prices of any neighboring job i.e.

$$\sum_i c_{ki} > \sum_j c_{kj} \text{ for all } h\ne k \text{ such that } c_{kh}\ne 0$$

Where c_{ij} is the IPC cost among the tasks *ti* and *tj* , when *ti* and *tj* are attributed to different processors.

3.5. Average Adulterate

Average adulterate on a C.P.U. is the sum of EC of the tasks assigned to it than the average load that must be assigned to each processor *pj* is computed as

$$Lavg(P_j)=\frac{W_j}{m},\ j=1,2,...n.$$

Where $W_j=\sum_{1\le i\le m} e_{i.j},\ j=1,2...n.$

$$Tlod=\sum_1^n Lavg(P_j)$$

4. ASSUMPTIONS

To keep the algorithmic program sensible in size, several presumptions have been made while planning the algorithmic program. A program is accepted to be collection of "*m*" tasks to be executed on a set of "*n*" processors, which have different processing potentialities. A task may be portion of an practicable code or a data file and size of all the tasks are equal. 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 time of a task on each processor is known, if a task is not executable on any of the C.P.U. due to absence of some resourcefulness's. The execution time of that task on that processor is taken to be infinite. It is also assumed that once a task has accomplished its execution on a processor, the processor stores the output data of the task in its local memory. If the data is demanded by some another task being computed on the same processor, it reads the data from the local computer memory. Using this fact, the algorithmic program tries to assign maximally linked tasks to the same processor. Whenever groups of tasks or cluster are assigned to the same C.P.U., the data transfer between them is zero. Completion of a program from computational point of view means that all associated tasks have got executed.

5. JOB ALLOTMENT EXAMPLE

In many heuristic program accesses, a search is made for a pair of module with maximal communicating cost between them. Such a module pair is assigned to a C.P.U. to minimize the IPC cost. But, it increases when a module is acting with many other modules. In this research paper, authors do not explore for such pairs of modules, but search maximally linked modules. The flowchart of whole program is shown in Figure 2.

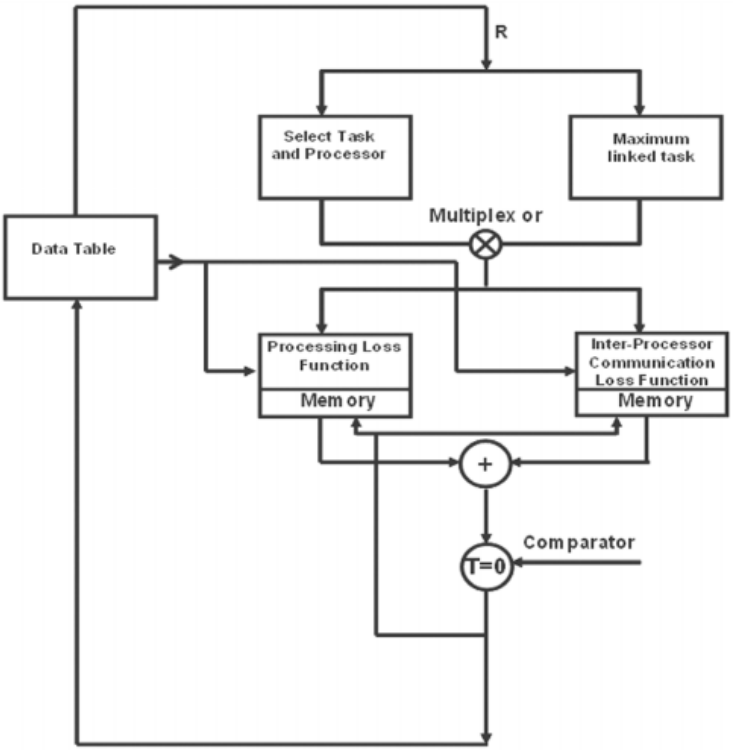

Figure 2: Flow Chart of the Program.

5.1. MAXIMUM ASSOCIATED JOB

It is three levels feed forward network using nerve cell having the unipolar activating function given in figure 3. Each of the electronic network levels is depicted by the formula $O = G(WX)$ At first layer, the input vector constitutes the elements of the upper triangular matrix of the IPC matrix *T* and output contains the *m* components. The value of each weight for the first layer is 1. At the second layer the output of the first layer work as an input vector and the output vector has $m(m-1)$ components. In this layer, each element is compared with others $(m-1)$ components to search the maximal linked module. When *i*th component is compared with *j*th component then the weight $Wij = Sgn\ (tij) - 0.5)$ and for others it is equal to one. At third layer each weight has value one. The output will be one for maximal linked module and zero for others. If maximum linked module is not equal to the number of processors then we select remaining modules according to maximal inter processing communication cost. To obtain them, the max comparator has been used. Thus the module clusters are formed around the maximally linked module and these *n* faculties are assigned on each processor.

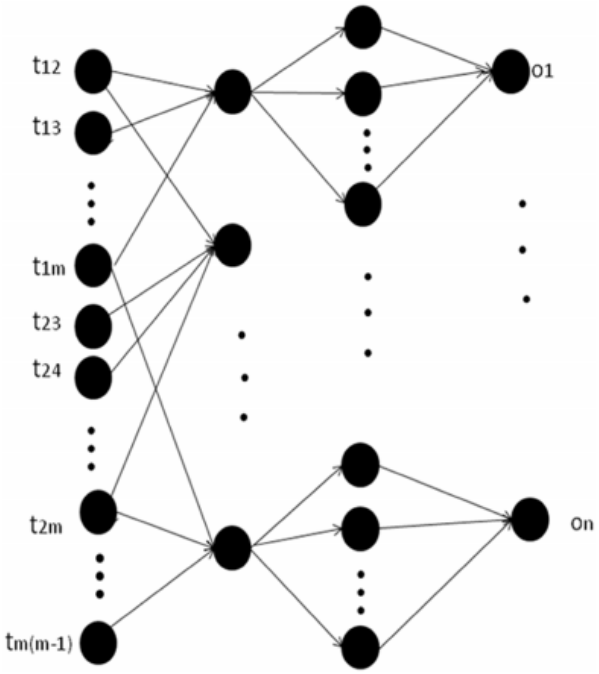

Figure 3: Network for Maximally Associated Job.

5.1.1 ACCOMPLISHING DEPRIVATION FUNCTION

This function is attempting to allot *i*th task *ti* to *j*th processor *pj* in memory the average load of each C.P.U. is stored. Now marching loss function find the conflict of the intermediate load and total executing cost of tasks which are attributed on processor *pj*. This difference is called marching loss.

5.1.2 IPC DEPRIVATION FUNCTION

When an *i*th job is allotted at *j*th processor, then IPC cost of the *i*th task with the task which is so soon assigned at *j*th processor will turns zero and it will be added with some other task which are not assigned at *j*th C.P.U. Memory comprises the previous IPC cost of assigned tasks at each processor. The difference of opinion of these IPC cost is called IPC loss. Neural network of IPC loss and processing loss of both is shown in figure 4. If the sum of the processing loss and IPC loss is positive then the *ith* task can be assigned to *j*th processor. This process is continued until the processing loss approaches zero/less than zero.

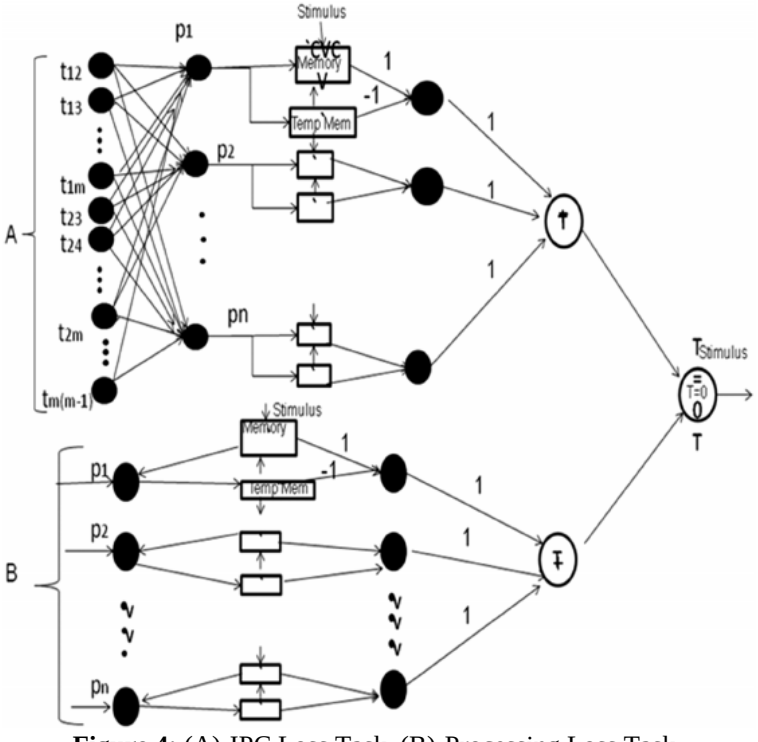

Figure 4: (A)-IPC Loss Task, (B)-Processing Loss Task.

6. EXAMPLE

Conceive a administered program consisting of a set $T = \{t_1.t_2, t_3, \ldots t_9\}$ of 9 jobs to be allocated to a set $P = \{p_1, p_2, p_3\}$ of 3 central processing unit. The objective of task appointment is to minimize the completion cost of a distributed program by properly mapping the tasks to the C.P.U.s. Authors of [14] have devised a trouble and developed a solution for it. We have tested our result on that problem. To illustrate the algorithm, a program graph has been constructed and is shown in figure 5 with the execution time matrix *E* being shown in Table 1 for three processors. Average load on each processor is shown in Table 2.

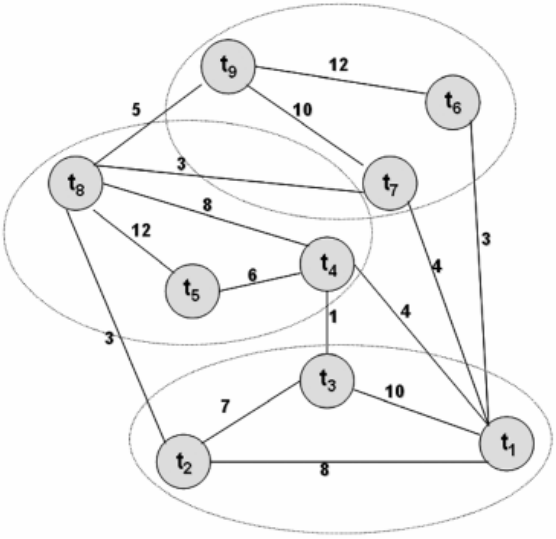

Figure 5: Example Program Graph.

Table 1
Task Execution Time Matrix E.

Task number	p_1	p_2	p_3
t_1	174	156	110
t_2	95	15	134
t_3	196	79	156
t_4	148	215	143
t_5	44	234	122
t_6	241	225	27
t_7	12	28	192
t_8	215	13	122
t_9	211	11	208

Table 2
Average Load on Each Processor.

Average Load	p_1	p_2	p_3	*Total Load*
	445.333	325.333	404.667	1175.333

In beginning first three maximally linked tasks are chose. Which are t_1, t_8, t_9 assigned to their best C.P.U. according to minimal execution cost on processor? In this case task t_1 is assigned to p_3, task t_8 to p_2 and task $t9$ to p_1 which is the left apportion processor. After this the algorithm proceeds as described in section 4.2 and 4.3. It should be noted that after each assignment of a task to the processor the total IPC time associated with that processor decreases. Results are shown in section 6.

7. RESULTS & CONCLUSION

The static job allotment is that when a job is assigned to CPU, it remains there while the feature of the calculation change and a new assignment is to be calculated. The hinted piece of research is in finding the appointment pattern that holds for the life time of a task, and consequence in the optimum value of the measure of effectuality. An artificial neural network based result of static allocation of tasks to processors is used. In distributed calculating surround the problem of distributing tasks to process is discussed and the results obtain are given in Table 3 shows the result obtained by the model.

Table 3
Result

Task/ Order	*Cumulative Execution Cost*	*Aggregate IPC*	*Processing Loss*
	Processor-P_1		
$t_{9/3}$	211	27	234.3333
$t_{6/6}$	452	18	–6.667
$t_{7/9}$	464	15	–18.667
	Response time of Processor p_1 is = 479		
	Processor-P_2		
$t_{8/1}$	13	31	312.333
$t_{5/4}$	247	25	78.333
$t_{4/7}$	462	16	–136.667

Contd. Table 3

Contd. Table 3

	Response time of Processor p_2 is = 478		
	Processor-P_3		
$t_{1/2}$	110	29	294.6667
$t_{3/5}$	266	27	138.6667
$t_{2/8}$	400	15	4.667
	Response time of Processor p_3 is = 415		

As it is not potential to achieve, a distributed system is conceived to be balanced if the load on each CPU is equal to the average load, within a reasonable tolerance that is of 20-30% of average load is generally chosen. Figure 6 show the maximum busy time of the system is 479 which relates to processor *p*2 after getting the assignment from the algorithm is concluded that the total assigned load on the system is 1372 with the tolerance of 20%. This is very near to the calculated load given in Table 2. The result obtained by our algorithmic program is similar to [14] but in present algorithmic program the solution aimed is highly parallel and can be carried out on chip level.

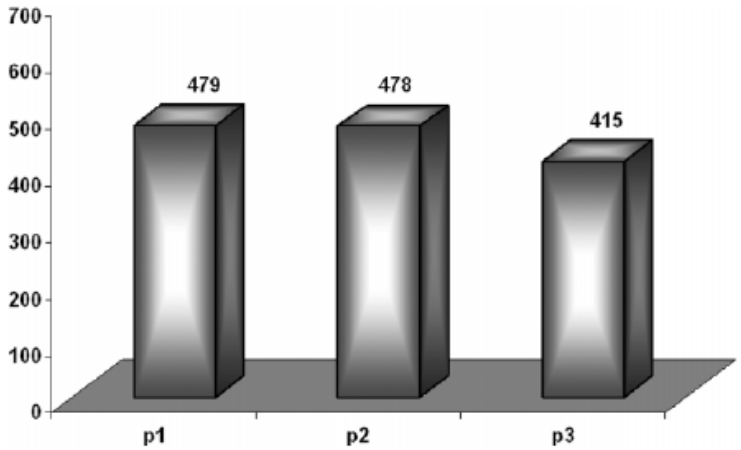

Figure 6: Maximum Busy Time of the System.

As we have assumed the size of all the tasks are equal and from the result it is concluded that all the processors are executing three tasks form the figure7, the through put of the *p*3 processor is maximum because in the execution cost of processor *p*3 is minimum.

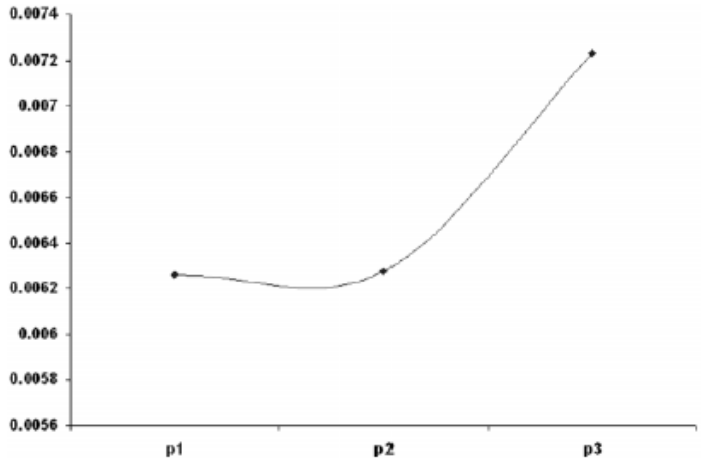

Figure 6: Throughput of the Processors.

The run time complexity of the algorithm suggested by Elsade [14] is $O(n_2 + m_2 + m_2n)$ and the run complexity of the algorithm presented in this paper is $O(m_2n + mn + n_2)$. Figure 7 shows the comparison between both the method and it concluded that present method is much better than Elsade [14].

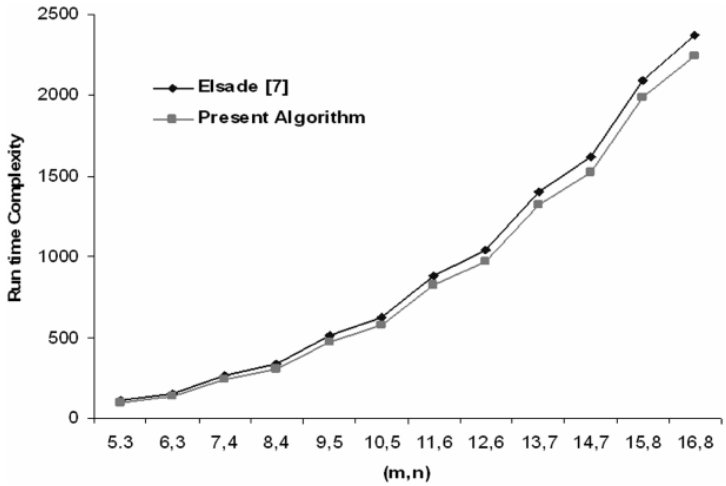

Figure 7: Complexity Comparison.

REFERENCES

1. Vetanovic, Z, "The Effects of Problem Partitioning, Allocation and Granularity on the Performance of Multiple Processor System", *IEEE Trans*., 1987, C-36, pp. 421-431.
2. A.K. Sarje, and G. Sagar, "Heuristic Model for Task Allocation in Distributed Computer Systems", *IEE Proceedings*, E., 138, No. 5, September 1991.
3. Efe. K., "Heuristic Models of Task Assignment Scheduling in Distributed Systems", *IEEE Transaction on Computers*, June 1982, pp. 50-56.
4. Virginia Mary Lo, "Heuristic Algorithm for Task Assignment in Distributed Systems", *IEEE Transaction on Computers*, 37, No. 11, November 1988, pp. 1384-1397.
5. Hesham H. Ali and Hasham EI-Rewini, "A Graph Theoretic Approach for Task Allocation", *IEEE Transaction on Computers*, 1992.
6. Chu, T., and Abraham J., "Load Balancing in Distributed Systems", *IEEE Transaction on Software Engineering*, SE-8, No. 4, July 1981.
7. Chu, W., Holloway, L., Lan, M., and Efe, K., "Task Allocation in Distributed Data Processing", *IEEE Computer*, November 1980, pp. 57-69.
8. Ma, P., Lee, E. and Tsuchiya, M., "A Task Allocation Model for Distributed Computing Systems", *IEEE Trans. Computer*., Jan. 1982, pp. 41-47.
9. Yadav, P.K., Kumar, Avanish and Singh, M.P., "An Algorithm for Solving the Unbalanced Assignment Problems", *International Journal of Mathematical Sciences*, 12(2), pp. 447-461 2004.
10. Yadav, P.K., Kumar, Avanish and Gupta, A.R., "An Exhaustive Approach of Performance Analysis to the Distributed Systems Based on Cost Assignments", *Published in South East Asian Journal of Mathematics and Mathematical Sciences*, 5, No.1, pp. 29-44, 2006.

11. Yadav, P.K, Govil, Kapil, and Kumar, Avanish, "Cost Based Static Approach for Performance Enhancement of Distributed Networks", *Published to the Journal Reflections des ERA Journal of Mathematical Sciences*, 2, issue 1, 2007, pp. 23-42.
12. Kumar, Avanish, Yadav, P.K., and Mudit Bansal, "Evaluation of Improved Reliability of Distributed Computing Systems using Mathematical Programming", *Published to the Journal of Mathematical Biosciences*, Gwalior Academy of Mathematical Sciences, 3, No. 1, 2007, pp. 88-96.
13. Yadav, P.K., Singh, M.P. and Kumar, Harendra, "Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Re-assignment", *Journal of Computer Systems, Networks, and Communications*, 2008, pp. 1-9, 2008.
14. Elsade A.A., Wells B.E., "A Heuristic Model for Task Allocation in Heterogeneous Distributed Computing System", *The International Journal of Computers and Their Applications*, 6, no.1, March 1999.
15. Yadav, P.K., Singh Jumindera and Singh, M.P (2009)., "An Efficient Method for Task Scheduling in Computer Communication Network", *International Journal of Intelligen*

Source of support: Nil, Conflict of interest: None Declared.