ISSN (e): 2250 – 3005 || Volume, 07 || Issue, 08|| August – 2017 ||
International Journal of Computational Engineering Research (IJCER)

Task assignment by Robust's ranking method with fuzzy execution and communication time for optimizing the distributed computing system

*Dr. Abhilasha Sharma
DIT,University Dehradun
Corresponding Author:Dr. Abhilasha Sharma

ABSTRACT
A Distributed Computing System (DCS) has the potential to enhance the system programs which is connected by a communication network. The optimality of the system can be achieved by performing the execution of the tasks in such a way that the response time of the system be optimal in DCS. In this paper we are assigning the tasks on that processor by the method of minimal linked tasks which has the strong characteristic with appropriate execution time by fusion of the tasks. We are taking fuzzy execution time and fuzzy inter task communication time in this paper. The defuzzification is used in fuzzy modeling and in fuzzy logic control to convert the fuzzy output from the system to crisp values. We are using defuzzification in this problem by Robust's ranking method. A mathematical model has been developed to determine the optimal response time and maximize throughput of the system with fuzzy execution time and fuzzy inter task communication time.

Keywords:*Fuzzy execution times, Fuzzy inter tasks communication times, minimal linked tasks, Robust's ranking method, defuzzification, crisp values, Fusion of tasks.*

Date of Submission: 24-07-2017 Date of acceptance: 10-08-2017

I. INTRODUCTION

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 [9]. 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 [7].Many approaches [1-5] have been given the various techniques for solving the task allocation problem in DCS. The static tasks assignment have been identified in the past with the main concern on the performance measures such as minimizing the total sum of execution and communication time, response time of the system, maximization of the system reliability. Sarje et al. [6] presented a method for static allocation of modules to processors, with the constraints of minimizing inter-processor communication cost and load balancing. Tripathi et al. [8, 10] developed a genetic approach in 2001 for allocating the tasks to the processors. Yadav et al.[11] in 2008 developed a mathematical model for multiple processors with dynamic re-assignment. Nagarajan et al. [12] in 2010, has developed a model for solving a fuzzy assignment problem using Robust's ranking method. The cost has been considered trapezoidal and triangular numbers. The paper [13] in 2012 deals a heuristic task allocation model which performs the proper allocation of task to most suitable processor to get an optimal solution.The paper [15] also proposed ranking and defuzzification methods. A tasks allocation model is developed by [14] for the optimization of reliability and cost in DCS. Kumar et al. [16] proposed a fuzzy approach to minimize the total cost by defuzzificationand converted into crisp indices.

II. PROBLEM STATEMENT AND DEFINITIONS

An allocation of tasks to processors is defined by $f: N \rightarrow M$, where M represents set of numbers of tasks and N represents set of numbers of processors. $N(i) = j$ the task t_i is assigned to processor p_j, $1 \leq i \leq n$, $1 \leq j \leq m$. Each processor catch the data from the local memory only not global memory. The fuzzy execution times (FUET), $et_{i,j}$ of the tasks on the processors is taken in the form of matrix named as fuzzy execution time matrix (FUETM), $FUETM = et_{i,j}$ of order m x n. The fuzzy inter-tasks communication times (FUITCT), $ct_{i,j}$ is taken in the form of a symmetric matrix named as fuzzy inter task communication time matrix (FUITCM), $FUITCTM = ct_{i,j}$ of order m. In this paper times $et_{i,j}$ and $ct_{i,j}$ have been considered to be triangular numbers. The objective of the problem is to find an assignment for a set N = {t_1, t_2….t_n} of m tasks to a set M = {p_1, p_2….p_m} of n processors in such a way that the total response time of the system is minimum.

2.1 Fuzzy Execution Time

The fuzzy execution time $et_{i,j}$ is the time to represent the performanceof the execution of the task t_i on the processor p_j from the set of tasks allocation N, the overall fuzzy execution time and fuzzy execution time for each processor are calculated by using equations (1) and (2) respectively as:

$$FUET(f) = \sum_{1 \leq i \leq n} et_{i,f(i)} \qquad (1)$$

$$PFUET(f) = \sum_{\substack{1 \leq i \leq n \\ i \in TAS_j}} et_{i,f(i)} \qquad (2)$$

Where $TAS_j = \{i: f(i) = j, \; j = 1, 2 \dots n\}$.

2.3 Fuzzy Inter Task Communication Time

The fuzzy inter task communication time $ct_{i,j}$ shows that the data communication between the tasks t_i and t_j if they are on different processors during the process of execution.for an allocation f, the overall fuzzy inter-task communication times and fuzzy inter-task communication time for each processor are calculated by using equations (3) and (4) respectively as:

$$FUITCT(f) = \sum_{\substack{1 \leq i \leq n \\ i+1 \leq j \leq n \\ f(i) \neq f(j)}} et_{f(i),f(j)} \qquad (3)$$

$$PFUITCT(f)_j = \sum_{\substack{1 \leq i \leq n \\ i+1 \leq j \leq n \\ f(i) \neq f(j)}} et_{f(i),f(j)} \qquad (4)$$

2.4 Response Time of the System

The response time is a function of the computation to be performed by each processor and the communication time. This function is defined by considering the processor with the heaviest aggregate computation and communication loads of the processors. The fuzzy response time of the system for the allocation is defined as:

$$FURT(f) = \underbrace{max}_{1 \leq j \leq n} \{PFUET(f)_j + PFUITCT(f)_j\} \qquad (5)$$

2.5Average 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 *FET*. 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 below.

$$L_{avg}(p_j) = \frac{W_j}{n}, j = 1,2, \dots\dots n \; where \; W_j = \sum_{1 \leq i \leq m} et_{i,j} \qquad (6)$$

$$Total\ Load[TL] = \sum_{1}^{m} L_{avg}(p_j) \qquad (7)$$

The average load on a processor $L_{avg}(p_j)$ 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.

III. ASSUMPTIONS

Several following assumptions have been made to keep the algorithm reasonable in size while designing the algorithm:

(a) The program is assumed to be the collection of "m" tasks, which are to be executed on a set of "n" processors that have different processing capabilities. A task may be a portion of an executable code or a data file.

(b) The number of tasks to be allocated is more than the number of processors, as normally is the case in the real life distributed computing environment. It is assumed that the fuzzy execution time of each task on each processor is known.

(c) 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 which being computed on the same processor, it reads the data from the local memory.

(d) The communication system of the processors is collision free, thus no messages are lost and all messages are sent in a finite amount of time. We assume a contention free communication for the processors.

(e) A processor can simultaneously execute a task and communicate with another processor. The overhead incurred by this is negligible, so for all practical purposes overhead will be considered as zero. Using this fact, the algorithm tries to allocate the heavily communicating tasks to the same processor. Whenever a group of tasks is assigned to the same processor, the FITCT between them is zero.

IV. TASK ALLOCATION MODEL

The tasks allocation procedure is completed by three major steps

- Inputs of the number of tasks.
- Defuzzification of the inputs times into crisps values.
- Tasks fusion by Minimal Linked tasks procedure and allocation of the fused tasks to the processors.

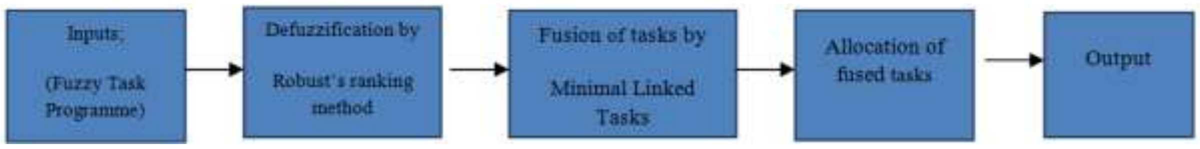

Figure 1: Structure of the Tasks Allocation Model

***Step 1:* Inputs**

(a): A set N ={$t_1.,t_2.....\,..t_n$} of m tasks.

(b): A set M = {$p_1, p_2 ......p_m$} of n processors.

(c): Fuzzy execution times $et_{i,j}$ and fuzzy inter tasks communication times $ct_{i,j}$ which is in triangular form. Write Fuzzy execution times $et_{i,j}$and fuzzy inter tasks communication times $ct_{i,j}$these times are given in the form of matrices[$et_{i,j}$]and [$ct_{i,j}$]respectively.

Step 2: Defuzzification

The input times $et_{i,j}$and $ct_{i,j}$are converted into crisp ones. This step is called defuzzification or defuzzified. In the present paper we are using Robust's ranking method for the defuzzification of the fuzzy costs. If (g_α^L, g_α^U) is a α- cut for a triangular fuzzy times (either $et_{i,j}$or$ct_{i,j}$) then its corresponding defuzzified Crisps value is calculated by the following equation as:

$$\mathrm{et}_{i,j} = \mathrm{RR}(\mathrm{et}_{i,j}) = \frac{1}{2}\int_0^1 (g_\alpha^L, g_\alpha^U)\,\mathrm{d}\alpha \qquad (8)$$

$$\mathrm{ct}_{i,j} = \mathrm{RR}(\mathrm{ct}_{i,j}) = \frac{1}{2}\int_0^1 (g_\alpha^L, g_\alpha^U)\,\mathrm{d}\alpha(9)$$

Defuzzified fuzzy execution times $\mathrm{et}_{i,j}$ and fuzzy inter tasks communication times $\mathrm{ct}_{i,j}$ are stored in the form of matrices [$\mathrm{et}_{i,j}$] and [$\mathrm{ct}_{i,j}$] respectively.

The Robust's ranking method is transform the $FUETM$ into a crisp value problem. The membership function of the triangular fuzzy number $et_{i,j,}$ =(a, b, c) is:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a} & , a \le x \le b \\ 1 & , x = b \\ \frac{c-x}{c-b} & , c \le x \le d \end{cases}$$

The α-cut of the fuzzy triangular number (a, b, c) isobtained by the following expression

$$\left(g_\alpha^L, g_\alpha^U\right) = [Lowervalue + (b-a)\alpha, Uppervalue - (c-b)\alpha].$$

Then we get Crisps value $= \frac{1}{2}\int_0^1 [Lowervalue + (b-a)\alpha + Uppervalue - (c-b)\alpha]\, d\alpha.$

Step 3: Fusion of Tasks and allocations to the processors

A distributed execution is the execution of processes across the distributed system to collaboratively achieve a common goal. The excessive inter-processor communication is always most costly and least reliable factor in the loosely coupled distributing computing system. Therefore, an efficient task allocation strategy is required for minimization of inter-processor communication that arises when the interacting tasks reside on different processors. The basic idea of forming the tasks fusion is to assign the minimum inter communication tasks to the processor to reduce the processing time of the system. In the present model we are using the concept of Minimally Linked Task (MLT) to allocate the tasks on the processors. Proper allocations of the tasks are very useful in complex situations. The values between each pair of communicating tasks are calculated by using the defuzzified crisp indices (step2).To determine the "m" Minimally Linked Task (MLT) we use the following expression in this model and the sum of FUITCT of task t_i with all task t_{l-i}arrange the MLT (,) in ascending order.

$$MLT(i,l) = \sum_{i,l=1}^{n} ct_{il} \qquad (10)$$

To make the one to one correspondence between fused tasks and processors the number of fused tasks should be equal to the number of processors. These fused tasks will be fixed throughout their execution. The overall efficiency of a computation of the DCS can be improved to an acceptable level by simply balancing the load among the processors. Increasing the overall efficiency will typically reduce the response time of the computation. The main objective of the fused tasks allocation is to minimize the response time of the distributed program by properly mapping the tasks to the processors. Here we are using Hungarian method [19] for mapping of the fused tasks to the processors which is a well-known method for assignment. The mapping of the fused tasks to the processors takes place according to the following algorithm:

Algorithm:

1. *Calculate the average load on the processors p_j and the total load by using the equation (6) and (7) with tolerance factor.*
2. *Calculate the Minimally Linked Task (MLT) from equation (9) and arrange the MLT (,) in ascending order with crisp indices.*
3. *Determine the defuzzified crisps value from equation (8).*
4. *Determine the assignment of the tasks on the basis of minimum defuzzified crisps value of MLT ( ) to the processors using Hungarian Method.*
5. *Store the assigned tasks in a linear array $T_{as}(i)$, non-assigned tasks in a linear array $T_{nas}(\,)$ and the processor position are also store in an another linear array $N_{alloc}(j)$.*
6. *All theremaining k =(n-m) tasks $T_{nas}(k)$ fused with those assigned tasks stored in $T_{as}(i)$ on the bases of minimum average of FUET and FUITCT.*
7. $N = T_{as}(i) \cup T_{nas}(k)$.
8. *The fused fuzzy execution time (FFUET) of a task $t_a \in T_{nas}(k)$ with some other task $t_i \in T_{as}(i)$ on processor p_j is obtained by*
$$FFUET(j)_{ai} = \left[et_{ai} + ct_{ij}\right], (1 \le a \le n, 1 \le i \le n, 1 \le j \le m, i \ne a).$$
9. *Fused fuzzy inter task communication time (FFUITCT) between t_a with t_i is obtained by the following expression Where ct_{ai} is the FFUITCT between $t_a \in T_{nas}(k)$ and $t_i \in T_{as}(i)$.*
$$FFUITCT(j)_{ai} = \sum_{i/t_i \in Tas(\,)} [ct_{ai}]\ , ct_{ai} = 0\ if\ i = a$$
10. *Calculate the Minimum Average Fused Fuzzy Time (MAFFUT) by*

$MAFFUT(j)_{ai} = \min\{(FFUET_{a1} + FFUITCT_{a1}), (FFUET_{a2} + FFUITCT_{a2}), \ldots\ldots\ldots\ldots (FFUET_{am} + FFUITCT_{am})\}$.

11. *This process will be continued until all the tasks storedin* $T_{nas}(\ )$ *are fused.*
12. *Make the same assignment of the fused tasks in to the original of matrix* $[\mathrm{et}_{i,j}]$.
13. *Calculate FUET (f), PFUET (f)*$_j$*, FUITCT (f), PFUITCT (f)*$_j$*, and FURT (f).*
14. *End.*

V. ILLUSTRATED EXAMPLE

The given example has been illustrated below using the above method. Example:
Let us consider a fuzzy DCS consisting a set N= {t_1, t_2, t_3, t_4, t_5} of "n=5" tasks and a set M= {p_1, p_2, p_3} of "m=3" processors. The execution time of each task on processors has been taken in the form of matrix $FUETM$ = [$et_{i,j}$] of order m x n whose elements are fuzzy triangular numbers as given in Table 1. Inter tasks communication time between the tasks has been taken in the form of matrix $FUITCTM$ = [$ct_{i,j}$] of order m whose elements are also fuzzy triangular numbers as given in Table 2.

Table 1: Fuzzy Execution Time Matrix

	p_1	p_2	p_3
t_1	(5,10,20)	(5,10,15)	(10,15,20)
t_2	(10,15,20)	(10,20,30)	(10,15,25)
t_3	(10,20,30)	(10,15,25)	(10,15,20)
t_4	(5,10,20)	(10,15,20)	(5,10,15)
t_5	(5,10,15)	(5,10,20)	(5,15,20)

Table 2: Fuzzy InterTask Execution Time Matrix

	t_1	t_2	t_3	t_4	t_5
t_1	(0,0,0)	(20,30,40)	(10,20,30)	(40,45.50)	(5,10,20)
t_2	(20,30,40)	(0,0,0)	(40,50,60)	(10,20,30)	(30,40,50)
t_3	(10,20,30)	(40,50,60)	(0,0,0)	(10,15,25)	(10,20,30)
t_4	(40,45,50)	(10,20,30)	(10,15,25)	(0,0,0)	(15,25,30)
t_5	(5,10,20)	(30,40,50)	(10,20,30)	(15,25,30)	(0,0, 0)

Applying the Robust's ranking method we get the crisp indices for the fuzzy execution times $et_{i,j,}$are calculated and shown in the table 3.

Table3: Crisp indices $et_{i,j}$ for the fuzzy execution times $et_{i,j}$

	p_1	p_2	p_3
t_1	11.25	10	15
t_2	15	20	16.25
t_3	20	16.25	15
t_4	11.25	15	10
t_5	10	11.25	13.75

Crisp indices $ct_{i,j}$for the fuzzy inter tasks communication times $ct_{i,j}$are calculated using the Robust's ranking method and are shown in the table 4.

Table 4: Crisp indices $ct_{i,j,}$for the fuzzy inter tasks communication times $ct_{i,j}$

	t_1	t_2	t_3	t_4	t_5
t_1	0	30	20	45	11.25
t_2	30	0	50	20	40
t_3	20	50	0	16.25	20
t_4	45	20	16.25	0	23.75
t_5	11.25	40	20	23.75	0

Actual average load assigned to the processor after introducing the 10 % Tolerance is.

$$L_{avg}\left(p_j\right) = \begin{Bmatrix} p_1 & 15 \\ p_2 & 16 \\ p_3 & 15 \end{Bmatrix}$$

Determine the Minimally Link Task (MLT) in ascending order with crisp indices shown in the Table 5.

Table 5: Minimally Link Task (MLT)

Tasks	FUITCT	Crisp values
t_5	(60,95,130)	95
t_3	(70,105,145)	106
t_4	(75,105,135)	105
t_1	(75,105,140)	106
t_2	(100,140,180)	140

Determine the assignment of the tasks by Hungarian Method.

$T_{as}(i) = \{t_5, t_1, t_4\}$

$T_{nas}(k) = \{t_2, t_3\}$, $N_{alloc}(j) = \{p_1, p_2, p_3\}$

Table6: Processors position after allocating the tasks

	p_1	p_2	p_3
t_1	(5,10,20)	**(5,10,15)**	(10,15,20)
t_2	(10,15,20)	(10,20,30)	(10,15,25)
t_3	(10,20,30)	(10,15,25)	(10,15,20)
t_4	(5,10,20)	(10,15,20)	**(5,10,15)**
t_5	**(5,10,15)**	(5,10,20)	(5,15,20)

Table 7: Fusion of remaining task $t_2$with assigned tasks $T_{as}(i)$

Processor	**Fusion of Task with the assigned tasks**	**FFUET (1)**	**FFUITCT (2)**	**FFUET+FFUITCT (1)+(2)**
p_1	t_2+t_5	(15,25,35)	(100,155,210)	**(115,180,245)**
p_2	t_2+t_1	(15,25,40)	(135,185,240)	(140,210,280)
p_3	t_2+t_4	(15,25,40)	(155,205,255)	(170,230,295)

Minimum optimal time is **(115,180,245)**so that fusion of task $t_2$will be assigned with t_5 on processor p_1.

$T_{as}(i) = \{t_5, t_2, t_4, t_1\}$.

$T_{nas}(k) = \{t_3\}$,$N_{alloc}(j) = \{p_1, p_1, p_3, p_2\}$.

Table 8: Fusion of remaining task $t_3$with assigned tasks $T_{as}(i)$

Processor	**Fusion of Task with the assigned tasks**	**FFUET (1)**	**FFUITCT (2)**	**FFUET+FFUITCT (1)+(2)**
p_1	t_3+t_5	(15,30,35)	(110,150,215)	**(115,180,250)**
p_2	t_3+t_1	(15,30,50)	(125,170,225)	(140,200,275)
p_3	t_3+t_4	(15,25,35)	(125,180,230)	(140,205,265)

Minimum optimal timeis **(115,180,250)** so that fusion of task $t_3$will be assigned with t_5 on processor p_1.

$T_{as}(i) = \{t_5, t_2, t_4, t_1, t_3\}$. $N_{alloc}(j) = \{p_1, p_1, p_3, p_2, p_1\}$.

Table 9 : Optimal assignment of the tasks to the processors

Processor	Tasks
p_1	t_2 , t_5,t_3
p_2	t_1
p_3	t_4

Table 10: Optimal Result of the given example

Processor	assigned tasks	PFUET (1)	PFUITCT (2)	FURT(f) =(1)+(2)	FURT(f)	Crisp Values ofFURT(f)
p_1	t_2 , t_5,t_3	(25,45,65)	(70,120,175)	(95,165,240)	**(95,165,240)**	166
p_2	t_1	(5,10,15)	(75,105,140)	(80,115,155)		
p_3	t_4	(5,10,15)	(75,105,135)	(80,115,150)		

Table 9 and figure 2 areshowing the optimal assignment of tasks to the processors. Tasks t_2, $t_5$and$t_3$executes on processor p_1, taskt_1 executes on processor p_2 and tasks t_4 executes on processor p_3. The fuzzy response time of the system is **(95,165,240)**units with crisp value **166.**

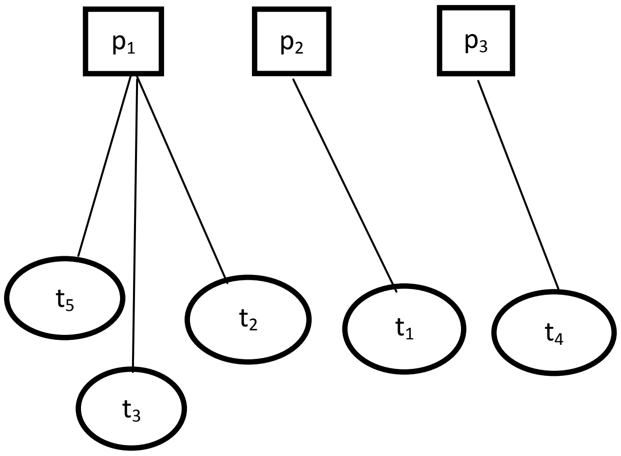

Figure 2: Optimal assignment of tasks to the processors

VI. RESULT COMPARISON

The maximum of FURT(f)is **166** i.e. the total busy time of the system is 166 which is corresponds to processor p_1.The model deals with the problem of optimal task allocation 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 FUET and FUITCT. It can also be perceived from the example presented here that wherever the algorithm of better result is encountered in [16]. Present technique gets an upper hand by producing better optimal results with slight enhancement in the response time of the system due to the optimal assignment of the tasks. The result comparison is shown by the Figure 3.

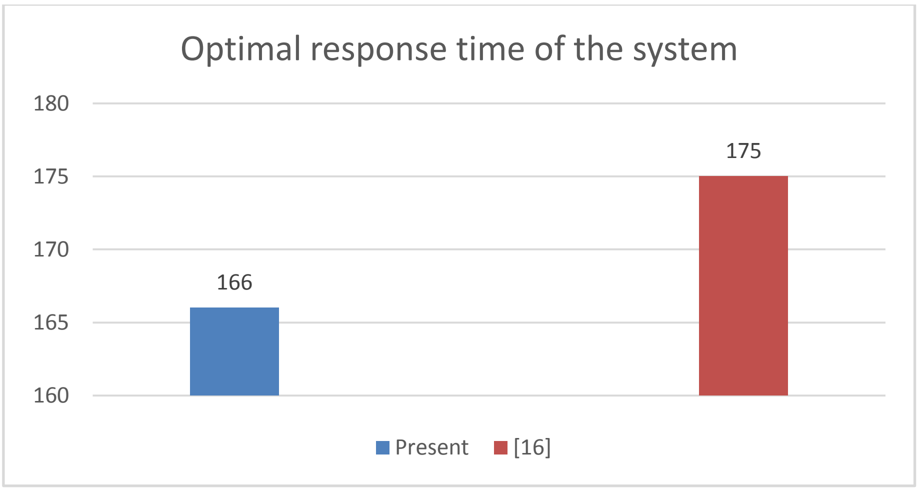

Figure 3: Response time of the system

VII. CONCLUSION

In this paper we have introduced a new fuzzy tasks allocation problem and its solution procedure. The problem has been formulated and depicted by a mathematical model. Fuzzy execution times$et_{i,j}$ and fuzzy inter tasks communication times $ct_{i,j}$ has been used while developing the model which are more realistic and general in nature. Many examples have been tested and it is found that the model is simple in use and has the potential to minimize the response time of the system by properly balancing the load on each processor. The present model can also be used in solving tasks allocation problems with fuzzy times. The present model is very useful in telephone networks, cellular network, computer games, image processing, cryptography, industrial process monitoring, simulation of VLSI circuits, sonar and radar surveillance, signal processing, simulation of nuclear reactor, power plants, airplanes, banking system etc. Although the model presented in this paper is efficient but still do not cover the full range of situations which would exist. A number of opportunities exist for future work. In future we can improve the model by using fuzzy logic approach for allocating the tasks clusters to the processors. The next important key issue in DCS is dynamic scheduling of the tasks program. In the present we have focused only on static load balancing policies, dynamic load balancing policies are yet to be considered.

REFERENCES

[1] H. Bokhari,, Dual Processor Scheduling with Dynamic Re Assignment, IEEE Trans. On Software Engineering, Vol.SE-5 ,1979, pp. 341-349.

[2] R.Y Richard, E.Y.S Lee, M. Tsuchiya, A Task Allocationmodel for Distributed ComputingSystem, IEEE Trans.OnComputer,Vol.C-31, 1982, pp.41-47.

[3] J.B Sinclayer, Optimal Assignment in Broadcast Network,IEEE Trans. On Computer,Vol.37(5), 1988, pp.521-351.

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

[5] D.F. Baca, "Allocation Tasks to Processor in a Distributed System, "IEEE TransactionsonsSoftware Engineering, Vol.15 pp.1427-1436, 1989.

[6] G. Sagar, A.K. Sarje, Task Allocation Model for Distributed System, Int.J. System Science, Vol. 22, 1991, pp. 1671-1678.

[7] H.G. Rotithor, Taxonomy of Dynamic Task Scheduling in Distributed Computing Systems, IEEE Proc. Computer Digit Tech., Vol. 14, 1994, pp. 1-10.

[8] A.K.Tripathi, D.P. Vidyarthi,.,. A.N. Mantri, A GeneticTask Allocation Algorithm for Distributed Computing System Incorporating Problem Specific Knowledge, International J. of High Speed Computing, Vol.8,No.4, 1996, pp. 363-370.

[9] A.A. Elsade, B.E. Wells, A Heuristic Model for Task Allocation in Heterogeneous Distributed Computing System, International Journal of Computers and Their Applications, Vol.6 (1), March 1999.

[10] D.P. Vidyarthi, A.K. Tripathi, Maximizing Reliability of Distributed Computing System with Task Allocation using Simple Genetic Algorithm, J. of Systems Architecture,Vol. 47,2001, pp. 549-554.

[11] M.P. Singh, H. Kumar, P.K. Yadav, Scheduling of Communicating modules of Periodic Tasks in Distributed Real-Time Environment, International Journal of Applied Mathematics &Engineering Sciences,Vol.2,No.2, 2008, pp.193-200.

[12] R Nagarajan, A. Solairaju, A.,Computing Improved Fuzzy Optimal Hungarian Assignment Problems with Fuzzy Costs under Robust Ranking Techniques, International Journal of Computer Applications , Vol. 6, No.4, 2010, pp.6-13.

[13] P.K..Yadav, P. Pradhan, P.P. Singh, A Fuzzy Clustering Method to Minimize the Inter Task Communication Effect for Optimal Utilization of Processor's Capacity in Distributed Real Time Systems, Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) (Advances in Intelligent and Soft Computing Published by Springer) Vol.130, 2012, pp 159-168.

[14] P.K. Yadav, M.P. Singh, K. Sharma., Tasks Allocation Model for Reliability and Cost Optimization in Distributed Computing System, International Journal Of Modeling, Simulation, and Scientific Computing, Vol.2, No.2,2011,pp.131-149.

[15] P.Fortemps and M.Roubens "Ranking and defuzzification methods based area compensation" Fuzzy sets and systems Vol.82.PP 319-330,1996 4Scientific Computing, Vol.2, No.2,2011,pp.131-149.

[16] Harendrakumar, M.P. Singh,P.K.Yadav Tasks Allocation Model with fuzzy execution and fuzzy Inter task communication times in a Distributed Computing System,International Journal of Computers and Their Applications,Vol 72, No.12, June 2013,pp.24-31.

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

[18] Abhilasha Sharma and Surbhi Gupta, "An Optimal Apporach for the Tasks Allocation Based on the Fusion of EC and ITCC in the Distributed Computing Systems"American International Journal of Research in Science, Technology, Engineering & Mathematics, 5(2), December 2013-February 2014, pp. 184-189.

[19] Gillett, Introduction to Operations Research: A computer Oriented Algorithmic Approach, McGraw-Hill, New York, 1984.

*Dr. Abhilasha Sharma " Task assignment by Robust's ranking method with fuzzy execution and communication time for optimizing the distributed computing system" International Journal of Computational Engineering Research (IJCER) 7.8 (2017): 12-19.