A Tasks Allocation Model with Fuzzy Execution and Fuzzy Inter-Tasks Communication Times in a Distributed Computing System

Harendra Kumar
Department of Mathematics and Statistics
Gurkula Kangari University,
Hardwar-249404, Uttarakhand (INDIA)
m

M. P. Singh
Department of Mathematics and Statistics
Gurkula Kangari University,
Hardwar-249404, Uttarakhand (INDIA)

Pradeep Kumar Yadav
Central Building Research Institute
Roorkee-247667, Uttarakhand (INDIA)

ABSTRACT
Distributed computing system [DCS] offer the potential for improved performance and resource sharing. To make the best use of the computational power available it is essential to assign the tasks to that processor whose characteristics are most appropriate for the execution. In this paper we have investigated a tasks allocation problem with fuzzy execution times $\tilde{e}_{i,j}$ and fuzzy inter tasks communication times $\tilde{c}_{i,j}$ which is more realistic and general in nature. Times $\tilde{e}_{i,j}$ and $\tilde{c}_{i,j}$ have been considered to be triangular and trapezoidal numbers. The fuzzy tasks allocation problem is defuzzified and converted into crisp ones using fuzzy number ranking method. A mathematical model has been developed to determine the optimal allocation of the tasks for the crisp problem that minimizes the total cost of the program. The allocation plan that minimizes the total cost for the new crisp problem also minimizes the total time for the original fuzzy tasks allocation. Numerical examples show that the model presented in this paper offers an effective tool for handling the fuzzy tasks allocation problem

Keywords: Distributed computing system; Fuzzy execution times; Fuzzy inter tasks communication times; Triangular and trapezoidal numbers; Crisp value.

* Corresponding Author

1. INTRODUCTION
Although computer speed has been increased by several orders of magnitude in recent decades, the demand for computing capacity increase at an even faster pace. The required processing power of many applications that are lengthy and repetive in nature cannot be achieved with a single processor. One approach to this problem is to use DCS. DCS provide faster computation by facilitating parallel execution of tasks of a program. Program partitioning into the tasks and their allocations are two major steps in designing of distributing processing systems. If these steps are not done properly, an increase in the number of processor in the system may actually result in the decrease of the total throughput due to the saturation effect that arises from excessive inter-processor communication. The excessive inter-processor communication is always most costly and least reliable factor in the loosely coupled DCS. Therefore an efficient task allocation strategy is required for the proper utilization of computational resources and minimization of inter-processor communication that arises when the interacting tasks reside on different processor.

It is often the case that a certain processor has very few tasks to handle at a given time, while another processor has many. It is desirable to spread the total workload of the DCS over all of its nodes. Load balancing policies may be either static or dynamic. Static load balancing policies use only the statistical information of the system in making the load balancing decisions. On the other hand, dynamic load balancing policies attempt to dynamically balance the workload reflecting the current system state and are therefore thought to be able to further improve the system performance.

An allocation policy can be static or dynamic, depending upon the time at which the allocation decisions are made. Many approaches have been reported for solving the ***Static*** tasks assignment problem in a DCS. The major thrust of research for evaluating the optimal tasks allocation is centered on providing solution that are scalable to large scale DCS.

Several approaches [1-6] to 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. [7] 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. [8-9] developed a genetic approach for allocating the tasks to the processors. Bokhari [10] analyzed the problem of dynamic assignment in a two-processor system which permits relocation of tasks from one processor to other at certain points during the execution of the program. Such relocation incurs a predefined relocation cost that contributes to the total cost of running the program and code-conversion overheads. Also it is shown that an optimal dynamic assignment corresponds to a "dynamic" partition imposed by a minimum weight cut in an extended graph. 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] deals a heuristic task allocation model which performs the proper

allocation of task to most suitable processor to get an optimal solution. A fuzzy membership functions is developed for making the clusters of tasks with the constraints to maximize the throughput and minimize the parallel execution time of the system. A tasks allocation model is developed by [14] for the optimization of reliability and cost in DCS. Sabeghi et al. [15] proposed a fuzzy approach to multiprocessor real-time scheduling in which the scheduling parameters are treated as fuzzy variables.

In this paper we are proposing a new mathematical approach for assigning a set of "**m**'' tasks of a program to a set of "**n**" processors using fuzzy execution times and fuzzy inter tasks communication times with the objective of minimizing the total processing time of the system. The rest of the paper is organized as follows: Task allocation problem and definitions used in the paper are defined in section 2. The assumptions used while preparing the model are discussed in section 3. Tasks allocation model is addressed in section 4. The implementation of the model is presented in section 5 and finally section 6 gives the conclusions of this model.

2. PROBLEM STATENENT AND DEFINITIONS

An allocation of tasks to processors is defined by a function, A from the set T of tasks to the set P of processors such that:

A: T→P, where A (i) =j if task t_i is assigned to processor p_j , 1≤i≤m,1≤j≤n.

Each processor has local memory only and do not share any global memory. The fuzzy execution times (FET), $\tilde{e}_{i,j}$ of the tasks on the processors is taken in the form of matrix named as fuzzy execution time matrix (FETM), $FETM = [\tilde{e}_{i,j}]$ of order m x n. The fuzzy inter-tasks communication times (FITCT), $\tilde{c}_{i,j}$ is taken in the form of a symmetric matrix named as fuzzy inter task communication time matrix (FITCM), $FITCTM = [\tilde{c}_{i,j}]$ of order m. In this paper times $\tilde{e}_{i,j}$ and $\tilde{c}_{i,j}$ have been considered to be triangular and trapezoidal numbers. The objective of the problem is to find an assignment for a set T = {t_1, t_2….t_m} of m tasks to a set P = {p_1,p_2,….p_n} of n processors in such a way that the response time of the system is minimum.

2.1 Fuzzy Execution Time

The fuzzy execution time $\tilde{e}_{i,j}$ is the amount of the work to be performed by the executing task t_i on the processor p_j. For an allocation A, the overall fuzzy execution time and fuzzy execution time for each processor are calculated by using equations (1) and (2) respectively as:

$$FET(A) = \sum_{1\le i\le m} \tilde{e}_{i,A(i)} \quad (1)$$

$$PFET(A)_j = \sum_{\substack{1\le i\le m \\ i\in TS_j}} \tilde{e}_{i,A(i)} \quad (2)$$

where TS_j= {i: A(i) =j, j=1, 2…n}

2.2 Fuzzy Inter Task Communication Time

The fuzzy inter task communication time $\tilde{c}_{i,j}$ is incurred due to the data units exchanged between the tasks t_i and t_j if they are on different processors during the process of execution. For an allocation A, 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:

$$FITCT(A) = \sum_{\substack{1\le i\le m \\ i+1\le j\le m \\ A(i)\neq A(j)}} \tilde{c}_{A(i),A(j)} \quad (3)$$

$$PFITCT(A)_j = \sum_{\substack{1\le i\le m \\ i+1\le j\le m \\ A(i)=j\neq A(k)}} \tilde{c}_{A(i),A(k)} \quad (4)$$

2.3 Response Time of the System

The response time [RT] is a function of the amount of 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:

$$FRT(A) = \max_{1\le j\le n} \{ PFET(A)_j + PFITCT(A)_j \} \quad (5)$$

3. ASSUMPTIONS

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

A(1): 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.

A(2): 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.

A(3): 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.

A(4): 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.

A(5): 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.

4. TASKS ALLOCATION MODEL

The tasks allocation model that we are presenting in this paper completed in four major steps as illustrated in Figure 1, namely Inputs of the tasks programme, defuzzification of the inputs times into crisps values, tasks clustering and allocation of the clusters to the processors. During the tasks clustering

process, the numeric crisps $c_{i,j}$ values are transformed into their corresponding linguistics variables.

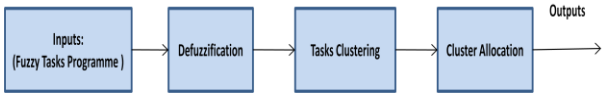

Figure 1: Structure of the Tasks Allocation Model

Step 1: Inputs

Inputs are as:

(a): A program of m tasks$\{t_1, t_2 \ldots . t_m\}$.
(b): A set P = $\{p_1, p_2, \ldots . p_n\}$ of n processors.
(c): Fuzzy execution times $\tilde{e}_{i,j}$ and fuzzy inter tasks communication times $\tilde{c}_{i,j}$ which are either in triangular or trapezoidal form. Write Fuzzy execution times $\tilde{e}_{i,j}$ and fuzzy inter tasks communication times $\tilde{c}_{i,j}$ these times are given in the form of matrices $[\tilde{e}_{i,j}]$ and $[\tilde{c}_{i,j}]$ respectively.

Step 2: Defuzzification

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

$$e_{i,j} = R(\tilde{e}_{i,j}) = \frac{1}{2}\int_0^1 \left(a_\alpha^L, a_\alpha^U\right) d\alpha \qquad (6)$$

$$c_{i,j} = R(\tilde{c}_{i,j}) = \frac{1}{2}\int_0^1 \left(a_\alpha^L, a_\alpha^U\right) d\alpha \qquad (7)$$

Defuzzified fuzzy execution times $\tilde{e}_{i,j}$ and fuzzy inter tasks communication times $\tilde{c}_{i,j}$ are stored in the form of matrices $\left[e_{i,j}\right]$ and $\left[c_{i,j}\right]$ respectively.

Step 3: Tasks Clustering

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 cluster is to assign the heavily communicated tasks on the same processor to reduce the communication times. In the present model we are using the concept of linguistic variables for making clusters of the tasks which are very useful in dealing with a complex situation. The membership values between each pair of communicating tasks are calculated using the defuzzified crisp indices $c_{i,j}$ by the following membership function as:

$$\mu_T\left(c_{i,j}\right) = \frac{c_{i,j}}{C} \qquad (8)$$

where

$C = max\{c_{i,j} : i = 1,2, \ldots . . m \ , j = 1,2, \ldots \ldots m\}$

For the sake of convenience and considering the human way of perceiving differences we are selecting five types of linguistic variables {Very Low Communicating Task [VLCT], Low Communicating Task [LCT], Average Communicating Task [ACT], Highly Communicating Task [HCT], Highly Communicating Task [HCT], Very Highly Communicating Task [VHCT]} to grade the high and low communicating pairs of tasks.. Tasks pairs are graded into five categories on the basis of their membership values as shown in **Table 1.**

Table 1: Grading of tasks pair

Membership Values	Linguistic Variables
[0, 0.2)	Very Low Communicating Tasks [VLCT]
[0.2, 0.4)	Low Communicating Tasks [LCT]
[0.4, .6)	Average Communicating Tasks [ACT]
[0.6, 0.8)	Highly Communicating Tasks [HCT]
[0.8, 1]	Very Highly Communicating Tasks[VHCT]

To make the one to one correspondence between tasks clusters and processors the number of tasks clusters should be equal to the number of processors. These clusters 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. If the new cluster, resulting from combining two clusters becomes too large, it would be impossible to obtain a load-balancing scheduling. We are making restriction on the maximum number of tasks in a cluster by $t_C = \left\lceil \frac{m}{n} \right\rceil$ for defining the size of the resulting clusters. To make the clusters of the tasks we will use the following algorithm:

Algorithm (A):

1. *Compute the membership values between each pair of communicating tasks using equation **8**.*
2. *Grade the tasks pair into five categories in terms of linguistic variables according to Table **1**.*
3. *Calculate* t_c.
4. *Define tasks pair priorities order for making the cluster as:*
   $VHCT \rightarrow HCT \rightarrow ACT \rightarrow LCT \rightarrow VLCT$
5. *Consider each task* t_i *as a distinct cluster as:*
   $\text{Cluster\_C(i)} \leftarrow \{t_i\} \ , i = 1,2, \ldots \ldots m$
6. *While (number of tasks clusters are not equal to number of processors) do*

   begin
   Select the pair of tasks clusters (say Cluster_C(p) *and* Cluster_C(q) *) with higher priorities // (Starting from VHCT)*

   Calculate

   $NT\{p.q\} = no.of\ tasks\ in\ \text{Cluster\_C(p)} + no.of\ tasks\ in\ \text{Cluster\_C(q)}$

   If

   $NT\{p,q\} \leq t_C$

   then

   (a) *fuse tasks clusters* Cluster_C(p) *with* Cluster_C(q) *i.e.*
   $\text{Cluster\_C(p)} \leftarrow \text{Cluster\_C(p)} \cup \text{Cluster\_C(q)}$

   (b) *delete the tasks cluster* Cluster_C(q)

else
go to step ***6(c).***

(c) *Select the next tasks cluster pair with next higher priorities for their fusion*
end while

7. *End*

Step 4: Cluster Allocation

The main objective of the clusters 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 [17] for mapping of the tasks clusters to the processors clusters which is a well known method for assignment. The mapping of the tasks clusters to the processors clusters takes place according to the following algorithm:

Algorithm (B):

1. *Modify the crisp indices* $e_{i,j}$ *in the matrix* $[e_{i,j}]$ *by fusing the tasks according to the algorithm* **(A)**
2. *Find the assignment of the tasks cluster to the processors using Hungarian Method.*
3. *Make the same assignment of the tasks clusters in to the original of matrix* $[\tilde{e}_{i,j}]$.
4. *Calculate FET(A), PFET(A)$_j$,FITCT(A), PFITCT(A)$_j$, and FRT(A).*
5. *End.*

5. ILLUSTRATED EXAMPLES

Two examples have been illustrated below using the above method.

Example1: Let us consider a fuzzy DCS consisting a set T= {t_1, t_2, t_3, t_4, t_5} of "m=5" executable tasks and a set P= {P_1, P_2, P_3} of "n=3" processors. The execution time of each task on processors has been taken in the form of matrix $FETM = [\tilde{e}_{i,j}]$ of order m x n whose elements are fuzzy triangular numbers as given in Table 2 and also depicted as tasks execution graph in Figure 2. Inter tasks communication time between the tasks has been taken in the form of matrix $FITCTM = [\tilde{c}_{i,j}]$ of order m whose elements are also fuzzy triangular numbers as given in Table 3 and also shown in Figure 3 as inter tasks communication graph.

Table 2: 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)

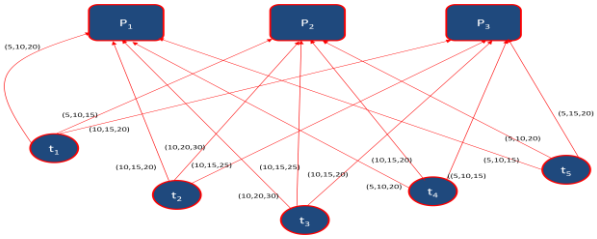

Figure 2: Task Execution Graph

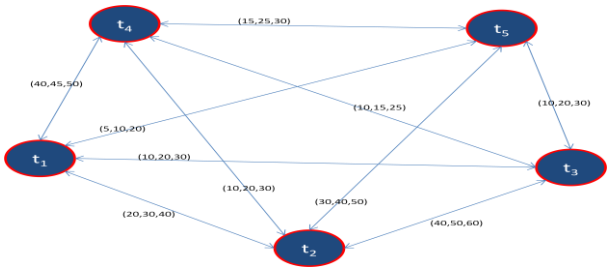

Figure 3: Inter Tasks Communication Graph

Using Robust's ranking indices, fuzzy DRTS is transformed into a crisp DRTS problem. The membership function of the triangular fuzzy number $\tilde{e}_{1,1} = (5,10,20)$ is:

$$\mu(x) = \begin{cases} \dfrac{x-5}{5}, & 5 \le x \le 10 \\ 1, & x = 10 \\ \dfrac{20-x}{10}, & 10 \le x \le 20 \\ 0, & \text{otherwise} \end{cases}$$

The α-cut of the fuzzy triangular number (5, 10, 20) is:

$$(a_\alpha^L, a_\alpha^U) = (5\alpha + 5, 20 - 10\alpha)$$

Now applying the Robust's ranking method we get:

$$R(\tilde{e}_{1,1}) = e_{1,1} = \frac{1}{2}\int_0^1 (25 - 5\,\alpha)d\alpha = 11.25$$

Applying the same procedure, crisp indices $e_{i,j}$ for the fuzzy execution times $\tilde{e}_{i,j}$ are calculated and shown in the table 4.

Table 4: Crisp indices $e_{i,j}$ for the fuzzy execution times $\tilde{e}_{i,j}$

	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 $c_{i,j}$ for the fuzzy inter tasks communication times $\tilde{c}_{i,j}$ are calculated using the Robust's ranking method and are shown in the table 5.

Table 3: Fuzzy Inter Task Communication 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)

Table 5: Crisp indices $c_{i,j}$ for the fuzzy inter tasks communication times $\tilde{c}_{i,j}$

	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

Table 6: Corresponding membership values

	t_1	t_2	t_3	t_4	t_5
t_1	0.00	0.60	0.40	0.90	0.23
t_2	0.60	0.00	1.00	0.40	0.80
t_3	0.40	1.00	0.00	0.33	0.40
t_4	0.90	0.40	0.33	0.00	0.48
t_5	0.23	0.80	0.40	0.48	0.00

Table 7: Grading of Communicating Tasks Pairs

	t_1	t_2	t_3	t_4	t_5
t_1	-	HCT	ACT	VHCT	LCT
t_2	HCT	-	VHCT	ACT	VHCT
t_3	ACT	VHCT	-	LCT	ACT
t_4	VHCT	ACT	LCT	-	ACT
t_5	LCT	VHCT	ACT	ACT	-

Table 8: Optimal Result of the Example1

Processor	**Tasks Assigned**	**PFET(A)$_j$ (1)**	**PFITCT(A)$_j$ (2)**	**(1)+(2)**	**FRT(A)**
P_1	t_1, t_4	(10,20,40)	(70,120,175)	(80,140,215)	**(110,175,250)**
P_2	t_5	(5,10,20)	(60,95,130)	(65,105,150)	
P_3	t_2, t_3	(20,30,45)	(90,145,205)	(110,175,250)	

Table 9: Fuzzy Execution Time Matrix

	P_1	P_2	P_3	P_4
t_1	(2,4,6,10)	(8,10,12,14)	(5,8,10,12)	(10,15,17,20)
t_2	(6,9,11,14)	(2,4,6,8)	(8,10,12,15)	(6,8,10,12)
t_3	(15,20,23,25)	(8,11,14,16)	(4,7,9,13)	(15,17,19,21)
t_4	(2,3,5,9)	(4,6,9,12)	(3,4,6,9)	(8,10,12,16)
t_5	(7,10,13,15)	(6,10,12,16)	(5,7,10,12)	(1,3,5,8)
t_6	(8,10,12,16)	(10,12,13,15)	(6,9,11,15)	(6,8,11,13)

Table 10: Fuzzy Inter Task Communication Time Matrix

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	(0,0,0,0)	(4,6,10,12)	(0,2,4,7)	(10,12,15,17)	(12,14,15,17)	(0,0,0,0)
t_2	(4,6,10,12)	(0,0,0,0)	(2,4,7,10)	(5,7,12,15)	(4,6,9,11)	(4,5,7,9)
t_3	(0,2,4,7)	(2,4,7,10)	(0,0,0,0)	(10,12,14,16)	(16,17,18,20)	(2,5,9,11)
t_4	(10,12,15,17)	(5,7,12,15)	(10,12,14,16)	(0,0,0,0)	(2,4,6,10)	(0,0,0,0)
t_5	(12,14,15,17)	(4,6,9,11)	(16,17,18,20)	(2,4,6,10)	(0,0,0,0)	(10,12,15,17)
t_6	(0,0,0,0)	(4,5,7,9)	(2,5,9,11)	(0,0,0,0)	(10,12,15,17)	(0,0,0,0)

Table 11: Crisp indices $e_{i,j}$ for the fuzzy execution times $\tilde{e}_{i,j}$

	P_1	P_2	P_3	P_4
t_1	5.5	11	8.75	15.5
t_2	10	5	11.25	9
t_3	20.75	12.25	8.25	18
t_4	4.75	7.75	5.5	11.5
t_5	11.25	9	8.5	4.25
t_6	11.5	12.5	10.25	9.5

Table 12: Crisp indices $c_{i,j}$ for the fuzzy inter tasks communication times $\tilde{c}_{i,j}$

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	0	8	2.75	13.5	14.5	0
t_2	8	0	5.75	9.75	7.5	6.25
t_3	2.75	5.75	0	13	17.75	6.75
t_4	13.5	9.75	13	0	5.5	0
t_5	14.5	7.5	17.75	5.5	0	13.5
t_6	0	6.25	6.75	0	13.5	0

Table 13: Corresponding membership values

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	0.00	0.45	0.15	0.76	0.82	0.00
t_2	0.45	0.00	0.32	0.55	0.42	0.35
t_3	0.15	0.32	0.00	0.73	1.00	0.38
t_4	0.76	0.55	0.73	0.00	0.31	0.00
t_5	0.82	0.42	1.00	0.31	0.00	0.76
t_6	0.00	0.35	0.38	0.00	0.76	0.00

Table 14: Grading of Communicating Tasks Pairs

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	-	ACT	VLCT	HCT	VHCT	VLCT
t_2	ACT	-	LCT	ACT	ACT	LCT
t_3	VLCT	LCT	-	HCT	VHCT	LCT
t_4	HCT	ACT	HCT	-	LCT	VLCT
t_5	VHCT	ACT	VHCT	LCT	-	HCT
t_6	VLCT	LCT	LCT	VLCT	HCT	-

Table 15: Optimal Result of the Example2

Processor	**Tasks Assigned**	**PFET(A)$_j$ (1)**	**PFITCT(A)$_j$ (2)**	**(1)+(2)**	**FRT(A)**
P_1	t_1, t_5	(9,14,19,25)	(46,59,77,94)	(55,73,96,119)	**(55,73,96,119)**
P_2	t_2	(2,4,6,8)	(19,28,45,57)	(21,32,51,65)	
P_3	t_4, t_3	(7,11,15,22)	(37,51,71,90)	(44,62,86,112)	
P_4	t_6	(6,8,11,13)	(16,22,31,37)	(22,30,42,50)	

The **Step3** of the method is applied for these tasks. Table 6 is showing membership values between each pair of communicating tasks and Table 7 is showing grading of communicating tasks pairs.
According to the **algorithm (A),** the following tasks clusters have been formed:

Cluster_C(1): {t_1, t_4}
Cluster_C(2): {t_2, t_3}
Cluster_C(3): {t_5}

On applying the **Step 4** of the present model we get the following optimal assignment as:

Cluster_C(1)→P_1
Cluster_C(2)→P_3
Cluster_C(3)→P_2

Table 8 and Figure 4 are showing the optimal assignment of tasks to the processors. Tasks t_1, t_4 executes on processor P_1, task t_5 executes on processor P_2 and tasks t_2, t_3 executes on processor P_3. The fuzzy response time of the tasks programme is (110,175,250) units. The FET (A) and FITCT (A) for the tasks program are (35, 60,105) and (110,145,205) units respectively.

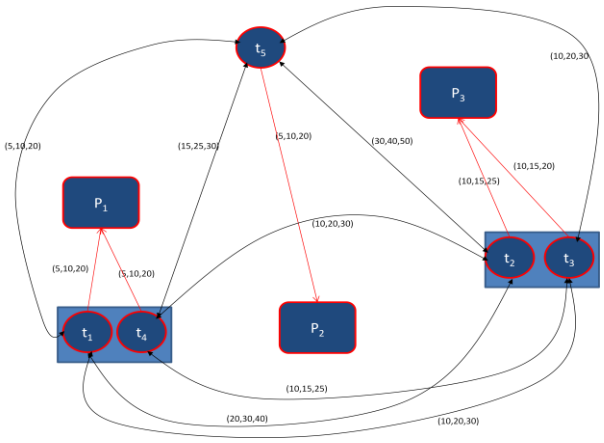

Figure 4: Optimal Assignment Graph

Example 2: Let us consider a fuzzy DRTS consisting a set T= {t_1, t_2, t_3, t_4, t_5, t_6} of "m=6" executable tasks and a set P= {P_1, P_2, P_3, P_4} of "n=4" processors. The execution time of each task on processors has been taken in the form of matrix $FETM = [\tilde{e}_{i,j}]$ of order m x n whose elements are fuzzy trapezoidal numbers as in Table 9 also depicted in Figure 5 as tasks execution graph. Inter tasks communication time between the tasks has been taken in the form of matrix $FITCTM = [\tilde{c}_{i,j}]$ of order m whose elements are also fuzzy trapezoidal numbers as given in Table 10 and also shown by Figure 6 as inter tasks communication graph.

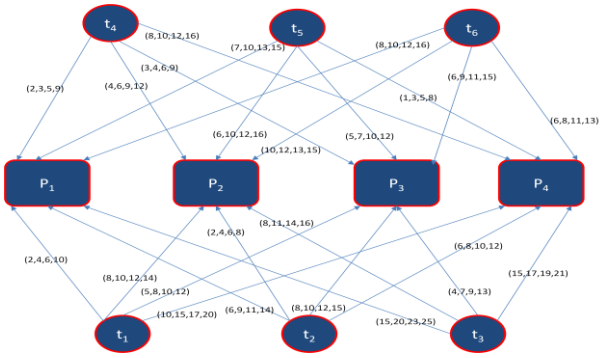

Figure 5: Task Execution Graph

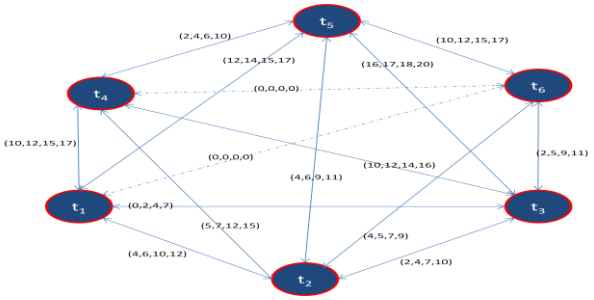

Figure 6: Inter Tasks Communication Graph

The crisp indices $e_{i,j}$ for the fuzzy execution times $\tilde{e}_{i,j}$ are calculated and shown in the Table 11. Similarly, crisp indices $c_{i,j}$ for the fuzzy inter tasks communication times $\tilde{c}_{i,j}$ are and are shown in the Table 12.

Applying the **Step3** of the method, the membership values between each pair of communicating tasks and their corresponding grading are calculated which are given in the Tables 13 & 14 respectively.

According to the **algorithm (A),** the following tasks clusters have been formed:

Cluster_C(1): {t_1, t_5}
Cluster_C(2): {t_4, t_3}
Cluster_C(3): {t_2}
Cluster_C(4): {t_6}

On applying the **Step 4** of the present model we get the following optimal assignment as:

Cluster_C(1)→P_1
Cluster_C(2)→P_3
Cluster_C(3)→P_2
Cluster_C(4)→P_4

Table 15 and Figure 7 are showing the optimal assignment of tasks to the processors. Tasks t_1, t_5 executes on processor P_1, task t_2 executes on processor P_2, tasks t_4, t_3 executes on processor P_3 and task t_6 executes on processor P_4. The fuzzy response time of the tasks program is (55,73,96,119) units. The FET (A) and FITCT (A) for the tasks program are (24, 37, 51, 68) and (50, 72,106,135) units respectively

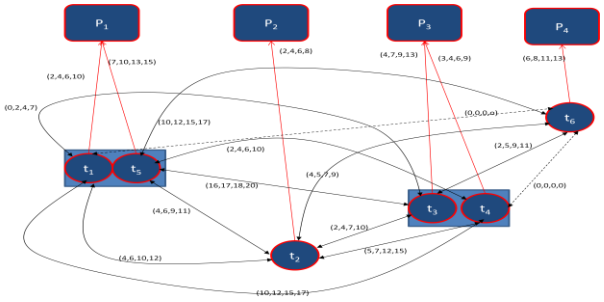

Figure 7: Optimal Assignment Graph

6. CONCLUSION

In this paper we have introduce a new fuzzy tasks allocation problem and its solution procedure. The problem has been formulated and depicted by a mathematical model. Fuzzy execution times $\tilde{e}_{i,j}$ and fuzzy inter tasks communication times $\tilde{c}_{i,j}$ has been used while developing the model which are more realistic and general in nature. The numeric crisps $c_{i,j}$ values are transformed into linguistics variables by generating a new membership function $\mu_T = (c_{i,j})$ for making the cluster of the tasks. 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 having mix type (triangular or trapezoidal), Bell shaped or Gaussian types of membership function. 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] R.Y Richard, E.Y.S Lee, M. Tsuchiya, A Task Allocation Model for Distributed Computer System , IEEE Trans. on Computer, Vol.C-31, 1982, pp.41-47.

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

[3] 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.

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

[5] 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.

[6] 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.

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

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

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

[11] P.K. Yadav, M.P. Singh, H. Kumar, Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Reassignment, Journal of Computer Systems, Networks and Communications, Article ID-578180, 2008, pp.1-9.

[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] M. Sabeghi, H. Deldari, V. Salmani, M. Bahekmat, A Fuzzy Algorithm for Real-Time Scheduling of Soft Periodic Tasks on Multiprocessor System, Procedding of IADIS International Conference Applied Computing, 2006, pp.467-471.

[16] L.A. Zadeh, Fuzzy Sets versus Probability, Proc. IEEE, Vol.68, March1980, pp.421-421.

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