Int. j. inf. tecnol. (March 2019) 11(1):13–20
https://doi.org/10.1007/s41870-018-0172-6

ORIGINAL RESEARCH

Fuzzy based task allocation technique in distributed computing system

Seema Yadav[1] · Rakesh Mohan[1] · P. K. Yadav[2]

Received: 3 May 2017 / Accepted: 10 April 2018 / Published online: 9 May 2018

Abstract With the rapid growth of Distributed System (DS) technology, the task scheduling has become an important issue. Task scheduling in distributed system is required to improve efficiency in applications such as project management, communications etc. The most important issue while designing any task scheduling algorithm is how to reduce make span time and waiting time. This paper proposes a novel fuzzy based task allocation algorithm. This algorithm can allocate task efficiently over different processors by balancing the load among processors with the objective of reducing execution and response time.

Keywords Distributed system · Task scheduling · Fuzzy execution time · Response time

1 Introduction

For last several years, distributed computing have become user friendly and a very popular choice for effective and efficient use of resources and for information processing. The benefits of distributed computing are: better throughput, effective use of available resources and access to wide web of information. Distributed computing has many challenges. A fair and balanced load distribution is one such challenge. Improperly distributed load results in reduced performance of the system. Therefore task scheduling is a vital step for better performance of the system, which can be done in the following ways:

Static Allocation In static allocation, the information of the current stage of nodes is not used for assigning the tasks. Thus an assignment pattern is needed to be found that holds for a life time of a program and results in optimum throughput [1].

Dynamic Allocation In dynamic allocation, the information of current state of the system is used. To update the information of the system, exchange of information is necessary [2].

Static allocation is simple in implementation but is not adaptable to the changes in system i.e. it does not change the task distribution as the system state changes. Different static allocation algorithms are discussed in [3–7]. Better performance is given by dynamic allocation methods over static methods as in dynamic methods, the distribution configuration changes with change in system, but it results in complicated algorithms [8–11].

A various number of methods are available for task distribution in distributed environment. One of the techniques is branch and bound technique as stated in [12–14]. Another method is integer programming which is a mathematical optimization technique where some variables are restricted to be integers. For task allocation problems, integer programming is simple in application. Another method for task distribution, which gives a near optimal solution is Genetic Algorithm (GA) which fosters a population of strings (chromosomes) using predefined genetic operators [15, 16]. The process of selection, crossover and

✉ Seema Yadav
seemayadav.research@gmail.com

Rakesh Mohan
mohanrakesh.dit@gmail.com

P. K. Yadav
pkyadav@cbri.res.in

[1] Department of Mathematics, DIT University, Dehradun, India

[2] CBRI, Roorkee, India

mutation is repeated until the condition for termination is satisfied.

1.1 Preliminaries

- *Fuzzy Execution Time (FET)*: The execution time, $\tilde{e}_{ij}$, is the amount of time taken by task t_i, which is to be executed on the processor p_j, where $1 \le i \le m, 1 \le j \le n$. If a task t_i is assigned to a processor p_j but is not executed due to absence of some resources, then $\tilde{e}_{ij}$ of the task on the processor is taken to be zero [17].
- *Fuzzy Inter Task Communication Time (FITCT)*: The Fuzzy Inter Task Communication Time, $\tilde{c}_{ik}$, is the amount of time incurred due to the data units exchanged between the tasks t_i and t_k if they are executed on different processors. When some tasks are assigned to same processor, then $\tilde{c}_{ik} = 0$. Fuzzy Inter-Task Communication Times for processor P_j is calculated by using Eq. (1) given as follows [17]:

$$FITCT_j = \sum_{i \neq k} [\tilde{c}_{ik}], (k = 1, 2, 3, \ldots, m), \quad 1 \le j \le n. \tag{1}$$

- *Triangular Fuzzy Number*: A triangular fuzzy number A(x) can be represented by A(a,b,c;1) shown in Fig. 1, with membership function $\mu(x)$ [18, 19].

$$\mu(x) = \begin{cases} \dfrac{x-a}{b-a}, & a \le x \le b \\ 1 & x = b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & Otherwise \end{cases}.$$

- *Trapezoidal Fuzzy Number*: A trapezoidal fuzzy number A(x) represented by A(a,b,c,d;1) as shown in Fig. 2, with membership value $\mu(x)$ [18, 19].

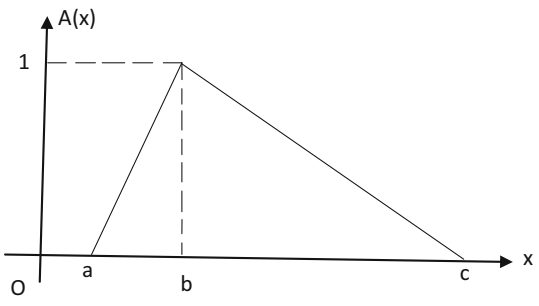

Fig. 1 Triangular fuzzy number

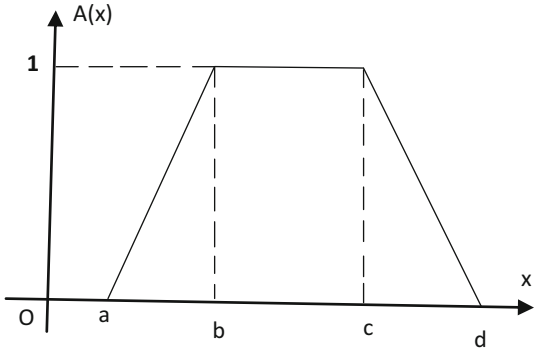

Fig. 2 Trapezoidal fuzzy number

$$\mu(x) = \begin{cases} \dfrac{x-a}{b-a}, & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & Otherwise \end{cases}.$$

- *Defuzzification*: The method of converting the fuzzy number into crisp value is defuzzification. Here the Fuzzy numbers (triangular/trapezoidal) are converted into crisp values by using Robust Ranking Method (RRM), which is represented by Eq. 2 [20].

$$a_{ij} = R(\tilde{a}_{ij}) = \frac{1}{2}\int_0^1 (a_\alpha^L + a_\alpha^U) d\alpha. \tag{2}$$

2 Proposed algorithm

2.1 Fetch the data set

Fetch the data set in the form of triangular/trapezoidal fuzzy numbers. Inputs are:

1. A program of m tasks i.e. $\{t_1, t_2, t_3, \ldots, t_m\}$.
2. A set of n processors i.e. $\{P_1, P_2, P_3, \ldots, P_n\}$.
3. FET $(\tilde{e}_{ij})$ and FITCT $(\tilde{c}_{ik})$ are in the form of triangular/trapezoidal fuzzy numbers. FET and FITCT are taken in the form of matrices as Fuzzy Execution Time Matrix (FETM) and Fuzzy Inter Task Communication Time Matrix (FITCTM).

2.2 Determination of minimum link (ML)

Find those "n" tasks, which have minimum link with other tasks using Eq. (1). This minimum link is stored in a two dimensional array, Minimum Link (ML), the first column of which represents the task number and second column represents the average minimum link between the tasks. The ML in fuzzy form is defuzzified into crisp values using

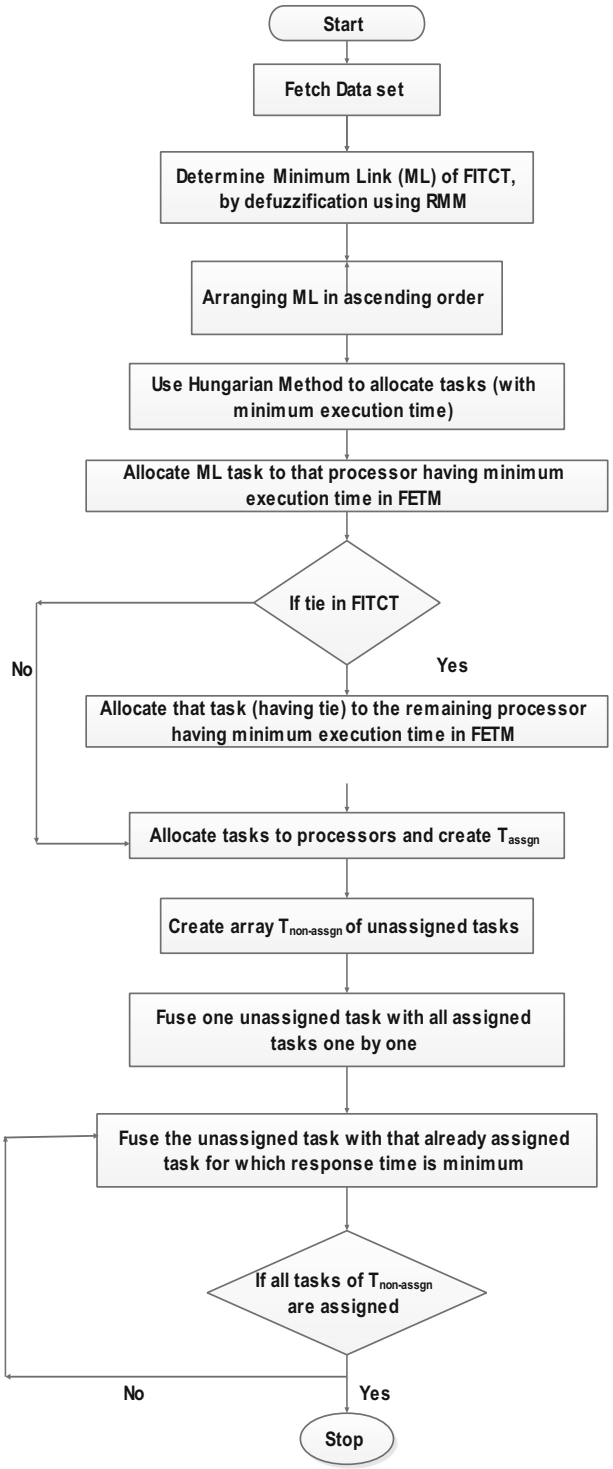

Fig. 3 Flowchart showing the sequence of proposed algorithm

Robust Ranking Method (RRM). This array is sorted in ascending order by assuming second column as sorting key, to find which tasks are to be allocated first.

Table 1 Fuzzy execution time matrix

	P1	P2	P3
t1	(5,10,20)	(5,10,15)	(10,15,20)
t2	(10,15,20)	(10,20,30)	(10,15,25)
t3	(10,20,30)	(10,15,25)	(10,15,20)
t4	(5,10,20)	(10,15,20)	(5,10,15)
t5	(5,10,15)	(5,10,20)	(5,15,20)

Table 2 Fuzzy inter task communication time matrix

	t1	t2	t3	t4	t5
t1	(0,0,0)	(20,30,40)	(10,20,30)	(40,45,50)	(5,10,20)
t2	(20,30,40)	(0,0,0)	(40,50,60)	(10,20,30)	(30,40,50)
t3	(10,20,30)	(40,50,60)	(0,0,0)	(10,15,25)	(10,20,30)
t4	(40,45,50)	(10,20,30)	(10,15,25)	(0,0,0)	(15,25,30)
t5	(5,10,20)	(30,40,50)	(10,20,30)	(15,25,30)	(0,0,0)

Table 3 List of minimum linked task

t1	(75,105,140)
t2	(100,140,180)
t3	(70,105,145)
t4	(75,105,135)
t5	(60,95,130)

Table 4 Listed minimum linked tasks in ascending order

Tasks	FITCT	Crisp value (FITCT)
t5	(60,95,130)	95
t4	(75,105,135)	105
t3	(70,105,145)	106.25
t1	(75,105,140)	106.25
t2	(100,140,180)	140

Table 5 Initial assigned tasks on processors (Using Hungarian Method) are as follows:

	P1/crisp values	P2/crisp values	P3/crisp values
t1	(5,10,20)/11.25	**(5,10,15)/10**	(10,15,20)/15
t2	(10,15,20)/15	(10,20,30)/20	(10,15,25)/16.25
t3	(10,20,30)/20	(10,15,25)/16.25	(10,15,20)/15
t4	(5,10,20)/11.25	(10,15,20)/15	**(5,10,15)/10**
t5	**(5,10,15)/10**	(5,10,20)/11.25	(5,15,20)/13.5

Bold indicates the tasks which are assigned initially to different processors using Hungarian method

Table 6 List of fusion of unassigned task t2 with already assigned tasks

Processor	Tasks fused with assigned tasks	FET (1)	FITCT (excluding time of fused tasks) (2)	PRT = FET + FITCT (1) + (2)	Crisp value of PRT
P1	t2 + t5	(15,25,35)	(100,155,210)	(115,180,245)	180.00
P2	t2 + t1	(15,30,45)	(135,185,240)	(150,215,285)	216.25
P3	t2 + t4	(15,25,40)	(155,205,255)	(170,230,295)	231.25

Table 7 List of fusion of unassigned task t3 with already assigned tasks

Processor	Tasks fused with assigned tasks	FET (1)	FITCT (excluding time of fused tasks) (2)	PRT = FET + FITCT (1) + (2)	Crisp value of PRT
P1	t3 + t5	(15,30,45)	(110,160,215)	(125,190,260)	191.25
P2	t3 + t1	(15,25,40)	(125,170,225)	(140,195,265)	198.75
P3	t3 + t4	(15,25,35)	(125,180,230)	(140,205,265)	203.75

Table 8 Final list of assigned tasks

Processor	Tasks
P1	t2,t3,t5
P2	t1
P3	t4

2.3 Determination of initial assignment for tasks with minimum execution time using Hungarian method

Select first "n" tasks from minimum linked array and apply Hungarian method to these "n" tasks in FETM, to allocate the tasks to the processors, having minimum execution time. Even if there is tie between two or more tasks in defuzzified ML array, the above mentioned method is used for the assignment of tasks. But now the task is assigned to the processor for which execution time is minimum in FETM. Let T_{assgn} denotes the set of tasks assigned to processors $P'_j s, j = 1, 2, 3, \ldots, n$ and $T_{non-assgn}$ denotes a set of $(m - n)$ task, $m \gg n$, which are not assigned to any of the processors. Then all tasks are given by union of these two as Eq. (3).

$$T = T_{assgn} \mathrm{U}\ T_{non-assgn}. \tag{3}$$

2.4 Fusion of remaining unassigned tasks

Remaining $(m - n)$ unassigned tasks are stored in an array $T_{non-assgn}$. Pick one non assigned task and fuse it with all assigned tasks one by one to calculate Process Response

Table 9 Final allocation task list with calculated OPRT value

Processor	Assigned Tasks	FFET (1)	FFITCT (2)	OPRT = FFET + FFITCT (1) + (2)	Crisp value of OPRT
P1	t2 + t3 + t5	(25,45,65)	(70,120,175)	(95,165,240)	166.25
P2	t1	(5,10,15)	(75,105,140)	(80,115,155)	116.25
P3	t4	(5,10,15)	(75,105,135)	(80,115,150)	115.00

Table 10 Fuzzy execution time matrix

	P1	P2	P3	P4
t1	(2,4,6,10)	(8,10,12,14)	(5,8,10,12)	(10,15,17,20)
t2	(6,9,11,14)	(2,4,6,8)	(8,10,12,15)	(6,8,10,12)
t3	(15,20,23,25)	(8,11,14,16)	(4,7,9,13)	(15,17,19,21)
t4	(2,3,5,9)	(4,6,9,12)	(3,4,6,9)	(8,10,12,16)
t5	(7,10,13,15)	(6,10,12,16)	(5,7,10,12)	(1,3,5,8)
t6	(8,10,12,16)	(10,12,13,15)	(6,9,11,15)	(6,8,11,13)

Table 11 Fuzzy inter task communication time matrix

	t1	t2	t3	t4	t5	t6
t1	(0,0,0,0)	(4,6,10,12)	(0,2,4,7)	(10,12,15,17)	(12,14,15,17)	(0,0,0,0)
t2	(4,6,10,12)	(0,0,0,0)	(2,4,7,10)	(5,7,12,15)	(4,6,9,11)	(4,5,7,9)
t3	(0,2,4,7)	(2,4,7,10)	(0,0,0,0)	(10,12,14,16)	(16,17,18,20)	(2,5,9,11)
t4	(10,12,15,17)	(5,7,12,15)	(10,12,14,16)	(0,0,0,0)	(2,4,6,10)	(0,0,0,0)
t5	(12,14,15,17)	(4,6,9,11)	(16,17,18,20)	(2,4,6,10)	(0,0,0,0)	(10,12,15,17)
t6	(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 12 List of minimum linked tasks

t1	(26,34,44,53)	39.25
t2	(19,28,45,57)	37.25
t3	(30,40,52,64)	46.5
t4	(27,35,47,58)	41.75
t5	(44,53,63,75)	58.75
t6	(16,22,31,37)	26.5

Table 13 List of minimum linked tasks in ascending order

Tasks	FITCT	Crisp value (FITCT)
t6	(16,22,31,37)	26.5
t2	(19,28,45,57)	37.25
t1	(26,34,44,53)	39.25
t4	(27,35,47,58)	41.75
t3	(30,40,52,64)	46.5
t5	(44,53,63,75)	58.75

Time (PRT). The Fused Fuzzy Execution Time (FFET) of a task $t_a \in T_{non-assgn}$ with some other task $t_i \in T_{assgn}$ on processor P_j is obtained using Eq. (4).

$$FFET_{ai} = \left[\tilde{e}_{aj} + \tilde{e}_{ij}\right], 1 \le i \le m, 1 \le j \le n, (m-n) \le a \le m, i \ne a. \tag{4}$$

Let $\tilde{c}_{ai}$ be the Fused Fuzzy Inter Task Communication Time (FFITCT) between $t_a \in T_{non-assgn}$ and $t_i \in T_{assgn}$. The FFITCT for t_a *with* t_i is obtained using Eq. (5).

$$FFITCT_{ai} = \sum_{\substack{t_i \in T_{assgn} \\ a \ne i}} [\tilde{c}_{ai}]. \tag{5}$$

Here $\tilde{c}_{ai} = 0$ *if* $a = i$ (i.e. if t_a is fused with t_i) and the remaining values of $\tilde{c}_{ai}$ are added.

2.5 Fused process response time (FPRT)

The Fused Process Response Time (FPRT) is calculated using Eq. (6) as follows:

$$FPRT_{ai} = \min\{(FFET_{a1} + FFITCT_{a1}), ((FFET_{a2} + FFITCT_{a2}), \ldots, (FFET_{am} + FFITCT_{am})\}. \tag{6}$$

FPRT is in the form of triangular/trapezoidal fuzzy numbers, which is then converted into crisp values using RRM given by Eq. (2). Task $t_a \in T_{non-assgn}$ is assigned to that processor for which FPRT, i.e. $(FFET_{ai} + FFITCT_{ai})$, is minimum. This process is continued until all the tasks, $t_a \in T_{non-assgn} \forall (m-n) \le a \le m$, are fused with the already assigned tasks, $t_i \in T_{assgn} \forall 1 \le i \le m$.

2.6 Overall process response time (OPRT)

When the procedure of assigning the tasks to different processors gets over, the OPRT for the distribution is calculated using Eqs. (4) and (5). These values are then converted into crisp values using Eq. (2). The OPRT, after assigning all the tasks, is calculated using Eq. (7) as follows:

$$OPRT = \max\{FFET + FFITCT). \tag{7}$$

Table 14 List of initial assigned tasks on processors using Hungarian method

	P1/crisp values	P2/crisp values	P3/crisp values	P4/crisp values
t1	**(2,4,6,10)/5.5**	(8,10,12,14)/11	(5,8,10,12)/8.75	(10,15,17,20)/15.5
t2	(6,9,11,14)/10	**(2,4,6,8)/5**	(8,10,12,15)/11.25	(6,8,10,12)/9
t3	(15,20,23,25)/20.75	(8,11,14,16)/12.25	(4,7,9,13)/8.25	(15,17,19,21)/18
t4	(2,3,5,9)/4.75	(4,6,9,12)/7.75	**(3,4,6,9)/5.5**	(8,10,12,16)/11.5
t5	(7,10,13,15)/11.25	(6,10,12,16)/11	(5,7,10,12)/8.5	(1,3,5,8)/4.25
t6	(8,10,12,16)/11.5	(10,12,13,15)/12.5	(6,9,11,15)/10.25	**(6,8,11,13)/9.5**

Bold indicates the tasks which are assigned initially to different processors using Hungarian method

Table 15 Fusion of task t3 with already assigned tasks

Processor	Tasks fused with assigned tasks	FET (1)	FITCT (excluding time of fused tasks) (2)	PRT = FET + FITCT (1) + (2)	Crisp value of PRT
P1	t3 + t1	(17,24,29,35)	(56,70,88,103)	(73,94,117,138)	105.5
P2	t3 + t2	(10,15,20,24)	(48,60,83,101)	(58,75,103,125)	90.25
P3	t3 + t4	(7,11,15,22)	(37,51,71,90)	(44,62,86,112)	76
P4	t3 + t6	(21,25,30,34)	(42,52,65,79)	(63,77,95,113)	87

Table 16 Fusion of task t5 with already assigned tasks

Processor	Tasks fused with assigned tasks	FET (1)	FITCT (excluding time of fused tasks) (2)	PRT = FET + FITCT (1) + (2)	Crisp value of PRT
P1	t5 + t1	(9,14,19,25)	(46,59,77,94)	(55,73,96,119)	85.75
P2	t5 + t2	(8,14,18,24)	(55,69,90,110)	(63,83,108,134)	97
P3	t5 + t4	(8,11,16,21)	(67,80,98,113)	(75,91,114,134)	103.5
P4	t5 + t6	(7,11,16,21)	(40,51,64,78)	(47,62,80,99)	72

Table 17 Final list of assigned task

Processor	Tasks
P1	t1
P2	t2
P3	t4,t3
P4	t6,t5

Flow Chart of the algorithm is shown in Fig. 3.

2.7 Illustrated examples

This section will illustrate the proposed method by using two scenarios:

2.8 Scenario I

In this example triangular fuzzy numbers are taken to test the proposed algorithm:

Consider a fuzzy DCS consists of set $T = \{t1, t2, t3, t4, t5\}$ *of tasks* $m = 5$ and a set $P = \{P_1, P_2, P_3\}$ of processors $n = 3$.The execution time of each task on processors and inter task communication time between communicating tasks has been taken in the form of matrix FET $[\tilde{e}_{ij}]$ and FITCT $[\tilde{c}_{ik}]$ of order $m \times n$ and $m \times m$ respectively, whose elements are triangular fuzzy numbers as given in Tables 1 and 2 [17].

From Table 2, minimum linked tasks are calculated shown in Table 3.These tasks are then converted into crisp values using RRM (Eq. 2) and then arranged in ascending order as shown in Table 4.

To assign ML tasks to processors, Hungarian method is used and the tasks are allocated as shown in Table 5.

Tasks are assigned by Hungarian Method by considering t1, t4 and t5 vs P2, P3, P1.

$$T_{assgn} = \{t1, t4, t5\} \rightarrow (P2, P3, P1) \text{ and } T_{non-assgn} = \{t2, t3\}.$$

To allocate task t2, it is fused with the already allocated tasks one by one and finally fused with the task having minimum PRT as shown in Table 6:

Minimum cost is (115,180,245). So task t2 is fused with t5 on processor P1.

$$T_{assgn} = \{t1, t2, t4, t5\} \rightarrow (P2, P1, P3, P1)$$
$$T_{non-assgn} = \{t3\}.$$

Table 18 Final allocation task list with calculated OPRT value

Processor	Assigned Tasks	FFET (1)	FFITCT (2)	OPRT = FFET + FFITCT (1) + (2)	Crisp value of PRT
P1	t1	(2,4,6,10)	(26,34,44,53)	(28,38,50,63)	44.75
P2	t2	(2,4,6,8)	(19,28,45,57)	(21,32,51,65)	42.25
P3	t4 + t3	(7,11,15,22)	(37,51,71,90)	(44,62,86,112)	**76**
P4	t6 + t5	(7,11,16,21)	(40,51,64,78)	(47,62,80,99)	72

Bold indicates the overall (maximum) time taken by the system to execute all tasks i.e. overall process response time

The above step is repeated for task t3 as shown in Table 7. Now fusing the remaining task t3 with initially allocated tasks:

Minimum cost is (125,190,260). So task t3 is fused with t5 on processor P1.

Final list of assignment of all tasks on each processor is shown in Table 8.

For Overall Process Response Time (OPRT), the total cost for all distributed tasks is calculated as shown in Table 9:

The maximum of crisp values is OPRT (Overall Process Response Time), which is **166.25** for this scenario-I.

2.9 Scenario II

In this example trapezoidal fuzzy numbers are taken to test the proposed algorithm:

Consider a fuzzy DCS consists of set $T = \{t1, t2, t3, t4, t5\}$ of tasks $m = 6$ and a set $P = \{P_1, P_2, P_3, P_4\}$ of processors $n = 4$.The execution time of each task on processors and inter task communication time between communicating tasks has been taken in the form of matrix FET $\left[\tilde{e}_{ij}\right]$ and FITCT $\left[\tilde{c}_{ik}\right]$ of order $m \times n$ and $m \times m$ respectively, whose elements are trapezoidal fuzzy numbers as given in the Tables 10 and 11 [17].

From Table 11, minimum linked tasks are calculated and converted into crisp values using RRM (Eq. 2) as listed in Table 12 and then arranged in ascending order as shown in Table 13.

To assign minimum linked tasks to processors, Hungarian method is used and the tasks are allocated as shown in Table 14.

Tasks are assigned by Hungarian Method by considering t1, t2, t4, t6 vs P1, P2, P3, P4.

$$T_{assgn} = \{t1, t2, t4, t6\} \rightarrow (P1, P2, P3, P4) \text{ and } T_{non-assgn} = \{t3, t5\}.$$

To allocate task t3, it is fused with the already allocated tasks one by one and finally fused with the task having minimum PRT as shown in Table 15.

Minimum cost is (44,62,86,112). So task t3 is fused with t4 on processor P3.

$$T_{assgn} = \{t1, t2, t3, t4, t6\} \rightarrow (P1, P2, P3, P3, P4)$$
$$T_{non-assgn} = \{t5\}.$$

The above step is repeated for task t5 as shown in Table 16.

Minimum cost is (47,62,80,99). So task t5 is fused with t6 on processor P4. The final list of all assigned tasks is shown in Table 17.

For overall PRT, the total cost for all distributed tasks is calculated as shown in Table 18.

The maximum of crisp values is OPRT (Overall Process Response Time), which is 76 for this problem.

The crisp value of OPRT (Overall Process Response Time) for both the above mentioned problems (**166.25 and 76**) are less than the OPRT of the paper compared (**250 and 119**).

"A Task Allocation with Fuzzy Execution and Fuzzy Inter Task Communication Times in a Distributed Computing system".

"International Journal of Computer Application (0977-8887) Volume 72–No.12, June 2013".

3 Discussions

The task scheduling in distributed environment is much difficult from the traditional methods. In traditional methods there is only one processor and there is a need for allocation of multiple tasks on it, which is much easier and also various predefined algorithms are present to do this. In distributed environment it becomes difficult because there are more than one processor and large number of tasks are there for allocation. Several studies have been conducted for task scheduling in distributed environment so that total throughput can be reduced. In a study El-Abd [18] reported a fuzzy model for load balancing in distributed system. The author simulated the model and tried to solve the problem of uncertainty in task selection for dynamic load balancing. In a study Sriramdas et al. [19] proposed a model for reliability allocation technique using fuzzy model and an approximation method based on linear programming approach. The model is based on centralized distributed system (DS). In a study Barazandeh et al. [21] proposed an algorithm based on fuzzy logic which works for centralized distributed system. They have considered load, last completed task waiting time as input for fuzzy model and infer specific weights as output variable. They have used MATLAB software for simulation of the model. In a study Park and Kuhl [22] proposed a fuzzy based load balancing consistency model for uncertainty in decision making in a large DS. Also they have simulated the model. In a study Kang et al. [23] proposed an iterative greedy algorithm to maximize the system reliability by considering the wide range of parameters. The model has been simulated using MATLAB. In a study Bey et al. [24] proposed a model which is the combination of Adaptive Network based Fuzzy Inference System (ANFIS) and clustering scheme to estimate the value of CPU load. The proposed study introduces novel algorithm based on fuzzy logic. *The proposed algorithm improves an overall process response time by allocating the task on processors. For this purpose*

Execution Time and Inter Task Communication Time have been taken into consideration. The algorithm uses fuzzy environment and therefore for defuzzification Robust Ranking Method is used, whenever there is a need of crisp values. The proposed algorithm is unique in a way that it uses Hungarian Method for initial allocation of tasks, tasks which are minimally linked, to different processors. From the data sets given in illustrated examples it can be seen that this algorithm improves the total response time in comparison to other methods.

4 Conclusion

In this paper a fuzzy task allocation problem has been formulated and shown in the form of mathematical model. Paper proposes a novel algorithm for allocating the tasks on different processors with the objective of minimum response time by taking Fuzzy Execution Time and Fuzzy Inter Task Communication Time into consideration. *The algorithm uses RRM (for defuzzification) and Hungarian method (for initial allocation of tasks).* Paper illustrated two scenarios for testing the proposed algorithm which gives PRT values 166.25 for scenario-1 and 76 for scenario-2. The model has potential to minimize the Overall Process Response Time by assigning an approximate balanced load to the processors as per literature studied. The limitation of paper is that it is restricted and focused on static load balancing policy. Although the model presented is efficient enough but leaves a number of situations where further work can be done. In future it can be explored for dynamic load balancing on processors.

References

1. Bhatia K (2001) Thesis on "design and analysis of some performance enhancement algorithms for distributed systems".
2. Eager DL, Lazowska ED, Zahorjan J (1985) A comparison of receiver-initiated and sender-initiated adaptive load sharing. ACM Sigmetrics Perform Eval Rev 13(2):1–3
3. Foster I, Kesselman C, Nick JM, Tuecke S (2002) The physiology of the grid: an open grid services architecture for distributed systems integration. Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum
4. Elsadek A, Wells BE (1999) A heuristic model for task allocation in heterogeneous distributed computing. Int Comput Appl 6(1):0–35
5. Yadav PK, Nadeem Ahmad (2011) Performance analysis of heterogeneous distributed processing system through systematic allocation of task. Int J Intell Inf Process 5(1):19–24
6. Yin PY, Yu SS, Wang PP, Wang YT (2006) Multi objective task allocation in distributed computing system by hybrid particle swarm optimization. Appl Math Comput 184:407–420
7. Ahmed AY (2012) Task allocation for minimizing cost of distributed computing systems using genetic algorithm. Int J Adv Res Comput Sci Softw Eng 2(9):202–209
8. Kowk YK, Ahmad I (2005) Multiprocessor task scheduling using efficient state space search approaches. J Parallel Distrib Comput 65:1515–1532
9. Kumar V, Singh MP, Yadav PK (1996) An Efficient algorithm for multi-processor scheduling with dynamic reassignment. In: Proc. of 6th National Seminar on Theoretical Computer Science, Banasthali Vidyapeeth, India, pp 105-18
10. Kumar A (1999) Optimization for dynamic task allocation. In: Proc. of 3rd Conference of the International Academy of Physical Sciences, Allahabad, pp 281–291
11. Yadav PK, Singh MP, Kumar Harendra (2008) Scheduling algorithm: task scheduling algorithm for multiple processors with dynamic reassignment. Int J Comput Syst, Netw Commun 2008:1–9. https://doi.org/10.1155/2008/578180
12. Attiya G, Hamam Y (2004) Two phase algorithm for load balancing in heterogeneous distributed systems. In: Proceeding of 12th Euromicro Conference on Parallel, Distributed and Network Based Processing, pp 434–439
13. Attiya G, Hamam Y (2004) Reliability Oriented task allocation in heterogeneous distributed computing systems. In: IEEE Conference, pp 68–73
14. Attiya G, Hamam Y (2006) Task allocation for maximization reliability of distributed systems: a simulated annealing approach. J Parallel Distrib Comput 66:1259–1266
15. Tripathi AK, Sarkar BK, Kumar N (2000) A GA based multiple task allocation considering load. Int J High Speed Comput 11(4):203–214
16. Yeh YS, Chui CC, Chen RS (2001) Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm. J Syst Architect 47:549–554
17. Kumar H, Singh MP, Yadav PK (2013) A tasks allocation model with fuzzy execution and fuzzy inter-tasks communication times in a distributed computing system. Int J Comput Appl 72(12):24–31
18. El-Abd AE (2002) Load balancing in distributed computing systems using fuzzy expert systems. In: Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2002. Proceedings of the International Conference, pp 141–144
19. Sriramdas V, Chaturvedi SK, Gargama H (2014) Fuzzy arithmetic based reliability allocation approach during early design and development. Expert Syst Appl 41(7):3444–3449
20. Srinivasan A, Geetharamani G (2013) Method for solving fuzzy assignment problem. Appl Math Sci 7(113):5607–5619
21. Barazandeh I, Mortazavi SS, Rahmani AM (2009) Intelligent fuzzy based biasing load balancing algorithm in distributed systems. In: Communications (MICC), 2009 IEEE 9th Malaysia International Conference on, pp 713–718
22. Park C, Kuhl JG (1995) A fuzzy-based distributed load balancing algorithm for large distributed systems. In: Autonomous Decentralized Systems, 1995. Proceedings ISADS 95, Second International Symposium on, pp 266–273
23. Qinma Kang, Hong He, Jun Wei (2013) An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. J Parallel Distrib Comput 73(8):1106–1115
24. Bey KB, Benhammadi F, Mokhtari A, Guessoum Z (2009) CPU load prediction model for distributed computing. In: Parallel and Distributed Computing, 2009. ISPDC'09. Eighth International Symposium on, pp 39–45. IEEE