Finding Fuzzy Reasoning Path on Fuzzy Deduction Graph using Parallel CYK Algorithm on a PRAM model

Pragya Shukla
Institute of Engineering and Technology, Devi Ahilya Vishwavidyalaya, Indore (M.P.), INDIA

Sanjiv Tokekar
Institute of Engineering and Technology, Devi Ahilya Vishwavidyalaya, Indore (M.P.), INDIA

Suresh Jain
KCB Technical Academy, Indore (M.P.), INDIA

ABSTRACT
In this paper a new parallel version of finding the Fuzzy Reasoning Path (FRP) using the knowledge representation model, introduced by Chandwani & Chaudhari [1] known as Fuzzy Deduction Graph (FDG) is presented. In an FDG, a systematic method of finding the Fuzzy Reasoning Path (FRP) already exists, which is based on Dijkstra's shortest path framework [2]. Our FRP algorithm is conglomeration of CYK algorithm of parsing and FRP algorithm for fuzzy reasoning which generates the path with the greatest fuzzy value. In FDG the weights of edges are real numbers in the fuzzy interval [0-1]. The maximum of multiplication is obtained on weights instead of minimum of summation of weights [1].

CYK algorithm employs a bottom up approach with the principle of Dynamic Programming (DP) to determine the FRP from source node to the destination node. The concurrency and synchronization in finding FRP process are inherently maintained through parallel PRAM model of construct. We present a complete formulation along with analysis of parallel algorithm for finding FRP.

General Terms
Algorithms, Fuzzy Logic and Parallel Processing.

Keywords
Deduction Graph, Fuzzy Deduction Graphs, Rule-based systems, Horn clauses, Fuzzy Reasoning Path, CYK-Algorithm, Dynamic Programming, PRAM Model, WRAM Model, Knowledge-base system.

1. INTRODUCTION
A parallel version of CYK- parsing algorithm for context – free grammar is presented by Chandwani, Puranik and Chaudhari[3, 4]. We present an algorithm which can store FRP in indexed array structure requiring simple parallel control constructs. We provide a detailed formulation and in-depth analysis of finding FRP using CYK-algorithm on parallel PRAM model of computation. In a rule-based expert system in which the knowledge is composed of a rule-base [5, 6, 18] an FDG can be used for graphical representation. An FDG is a graphically structured subset of a rule base R with rules of fuzzy propositions [7]. It can be used to perform automated fuzzy reasoning. A fuzzy reasoning path (FRP) can be established to define the antecedent-consequent relationship between source and goal propositions. One way to establish FRP is through fuzzy logic. In fuzzy logic, FRP defines an antecedent- consequent relationship of two propositions that leads to the greatest fuzzy value of the consequent proposition. The output of CYK algorithm is a pyramid that can be used to find the FRP. CYK algorithm's framework presented in this paper establishes the fuzzy reasoning path in FDG such that starting with source node; the goal node is reached with the greatest fuzzy value. The weights of edges in an FDG are real numbers in the fuzzy interval [0-1].The maximum of multiplication is obtained on weights instead of minimum of summations of weights. In the proposed PRAM formulation, the multiple entries are written exclusively in a location by means of effective ordering in different iterations. The analysis of algorithm illustrates that our algorithm requires 2n-2 time steps using quadratic number of processors. In particular, we show the number of the processors required in each iteration of the algorithm.

The rest of paper is organized as follows: Section 2 and Section 3 describe the fuzzy knowledge and reasoning and gives definition of FDG. Section 4 gives concepts of parallel CYK algorithm. Section 5 describes the operations on the cells of pyramid for finding FRP and algorithm of sequential CYK algorithm. Section 6 presents the proposed algorithm for finding FRP using parallel CYK algorithm on a PRAM model with example and its complexity issues for time and processor requirements. The paper ends with discussion and conclusion in Section 7.

2. FUZZY KNOWLEDGE AND REASONING
Lofti Zadeh proposed a mathematical way of looking at the vagueness of the natural human language; he called his approach fuzzy logic [8, 12]. The objective of fuzzy logic has been to make computer think like human. One way to represent fuzzy knowledge is through fuzzy propositions and fuzzy production rules. A fuzzy production rule defines the fuzzy relationship between antecedent and consequent propositions. If the fuzzy value of antecedent proposition and rule are known then fuzziness of consequent can be determined in fuzzy logic. For example, let p and q be fuzzy proposition in rule r: "IF p Then q" and let the fuzzy value of p be denoted by f(p) and the fuzzy value associated with rule r be denoted by C(r), then fuzziness of q is computed by (1). Normally, the fuzzy value of a rule is specified by certainty factor; i.e., how much certain the rule is applicable [9]

$$f(q) = C(r)*f(p). \quad (1)$$

3. FUZZY DEDUCTION GRAPH
An FDG is an extension of a Deduction Graph (DG) [10, 11, 16, and 17] by applying fuzzy concepts. It is a graphically structured subset of a rule base R with fuzzy rules of fuzzy propositions. A proposition in rule of R is defined to be simple node in an FDG. A rule is represented by a full edge. This full edge is labeled with weight equal to certainty factor of the rule.

A generalized FDG of Chandwani [4] is a 4-tuple FDG=<V, E, f, C> where

$V = \{v_1, v_2, \ldots, v_m\}$ is a set of nodes;

$E = \{e_1, e_2, \ldots, e_n\}$ is a set of directed edges;

$f : V \rightarrow [0\text{-}1]$ is a function from nodes to the real values in the fuzzy interval [0-1];

$C: E \rightarrow [0\text{-}1]$ is a function from edges to the real values in the fuzzy interval [0-1] such that

$$f(v') = f(v) * C(e) \qquad (2)$$

where e = <v, v'> is a directed edge in E from node v to node v'. Equation (2) indicates that the fuzzy value of a node, on which an edge is incident, is multiplication of certainty factor labeling the edge with the fuzzy value of node from which the edge is emerging.

3.1 Example

We take an example of a production system describing the relationships amongst fuzzy propositions and rules. Let the FDG of the production system contain, V = {A, B, C, D, E, F} and E = {r1, r2, r3, r4, r5, r6, r7, r8, r9} with the following relationships between propositions:

r1: A→B (C=0.9); r2: B→C (C=0.95); r3: A→D(C=0.75); r4: B→D (C=0.85); r5: B→E(C=0.7); r6: D→E(C=0.9); r7: C→Z (C=0.75); r8: C→E(C=0.85); r9: E→Z(C=0.8);

An FDG for above rules is depicted in Fig 1.

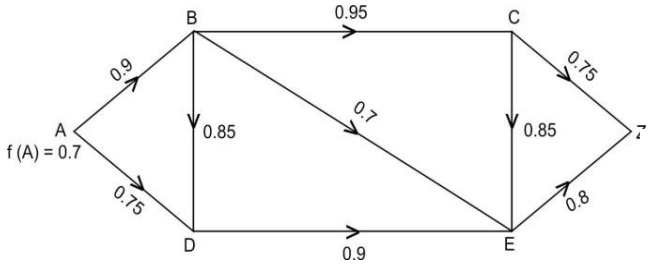

Fig 1: An OR – Type FDG

4. CYK – ALGORITHM AND PARALLELISM

The Cocke-Younger-Kasami (CYK) algorithm is generally used for recognition and parsing of context-free language. This algorithm is simple and uses bottom-up approach that is based on dynamic programming. Its simplicity is achieved from the input grammar being in CNF. The algorithm creates a recognition table that can be used to find the parse structure of the input string. The recognition table is usually represented in the form of an (n+1) × (n+1) upper triangular matrix. The table can also be built up in the form of pyramid of cells. In this paper we are using CYK algorithm for finding fuzzy reasoning path from source node to the destination node. A fuzzy reasoning path of length n can be finding out by building a pyramid of size n with base consisting of n+1 cell and the top row containing the fuzzy reasoning path with maximum fuzzy value. The elements of the pyramid are denoted by Cell[i, j]. Cell[i, j] contains the partial path of length i. The elements of the base are denoted by Cell[i,j] for i=0 and $1 \leq j \leq n$ and the element of the top row is denoted by Cell[n, 1]. Fig. 2 shows a pyramid structure of size n=4 and indexing of elements. Cells contain partial paths and multiplication is the basic operation performed on cells.

PRAM and WRAM models: A PRAM model of parallel computation is a shared memory model which allows any number of processors to read simultaneously from the same memory location. It does not allow two processors to write simultaneously into the same location. A model in which many processors are allowed to write into the same location simultaneously is known as WRAM model of parallel computation.

In practice, a WRAM model is not used as there will be conflicts due to concurrent writes in the same location. WRAM model based algorithms are more of theoretical interest [14]. A PRAM model is most often used in the design of parallel algorithms and their practical implementations. The parallelism in our algorithm is described in the following statement [14]:

For all x in X **in parallel** do

instruction(x)

where x is an element of the set X and the statement instruction(x) includes the assignment of processor to the element x and other operations specified in instruction(x). The time required by above statement in any algorithm is independent of cardinality of X and taken as constant.

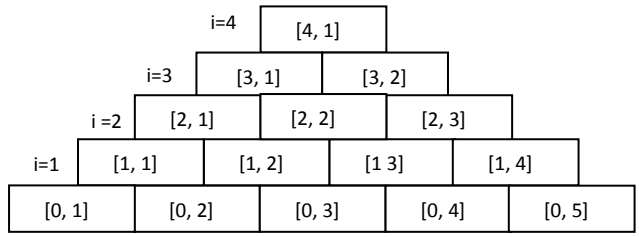

Fig 2: CYK – Pyramid of size n = 4.

5. FUZZY REASONING PATH WITH CYK-ALGORITHM

On applying CYK algorithm on FDG we can obtained fuzzy reasoning path with greatest fuzzy value. The algorithm starts working as soon as the first node A of the graph is entered, where A is a node representing the source proposition and Z is a node representing the goal proposition. In other words, the algorithm is required to prove the formula (Z←A) with the greatest degree of truth value of goal node Z from FDG. It is assumed here that there exists a path from the node A to the node Z. Let G=<V, E, f, C> be a FDG and we want to find the FRP between A and Z i.e., FRP(A, Z). We shall use pyramid representation for our algorithm. Path consist of n nodes can be searched by building a pyramid of size n. A pyramid of size n has n rows indexed by i such that 1<i<n and (n-i+1) column indexed by j such 1<j< (n-i+1) in each row i. The base row of the pyramid consists of n columns and the top row contains the paths from source to destination with greatest FRP value. Let the pyramid be named as cell and let its elements be denoted by Cell[i, j]. Content of each Cell[i, j] is a set of partial paths to the goal node with its FRP value. Initially we have source node with its fuzzy value f(A).

5.1 Operations on the Cells of Pyramid

Two operations are performed on the content of pyramid's cell[i,j]. They are as follows:

5.1.1 Operation for finding FRP

Suppose the content of Cell[i,j] is filled by the content of Cell[X,Y] and Cell[M,N]. If Cell[X,Y] contain partial path p_k = ABC and Cell[M,N] contain partial path p_{k+1} = CD. Binary operation Θ over Cell[X,Y] and Cell[M,N] is as follows:

Cell[i,j] = Cell[X,Y] Θ Cell[M,N] = $p_K = p_k$ Θ p_{k+1} = ABCD

5.1.2 Operation for finding fuzzy reasoning value for FRP

Suppose the content of Cell[i,j] is filled by the content of Cell[X,Y] and Cell[M,N]. Cell[X,Y] contain partial path p_k = ABC with fuzzy reasoning value $f(p_k)$ and Cell[M,N] contain partial path p_{k+1} = CD with fuzzy reasoning value $f(p_{k+1})$, then the fuzzy reasoning value of partial path ABCD is $f_K = f(p_k)* f(p_{k+1})$, where * is multiplication operator is stored in Cell[i,j]

The description of sequential algorithm is presented as follows:

5.2 Algorithm 1

Sequential CKY algorithm for fuzzy reasoning path is as follows:

Let G = <V, E, f, C> be a FDG, finding FRP can be done by constructing a pyramid and filling up the contents in Cell[i,j] through Θ and * operation:

Step 1. // Construct the bottom row from the FDG G //

If i=0 and j=1

Cell[i, j]:= insert the source node with its fuzzy value;

j = j+1;

if j>1 do

Until all the nodes and its adjacent nodes of the FDG are stored in the base cell of the pyramid do

Cell[0,j]:=insert the adjacent nodes of the node stored in Cell[0,j-1] with its certainty value;

j:=j+1

Step 2 // Construct the upper rows using rule P→Q for length i>1 //

For j:= 1 to n-i+1 do

Cell[1,j]:= Insert the partial path and calculate its fuzzy value from Cell[0,j] and Cell[0,j+1], if there is an edge from nodes stored in Cell[0,j] to cell[0,j+1] in FDG.

For i:= 2 to n do

For j:=1 to n-i+1 do

For m:= 1 to i-1 do

Cell[i,j]:=Cell[m,j] Θ Cell[i-m,j+m]

// Insert the partial path by applying operation 'Θ' //

$f(p_k)*f(p_{k+1})$

// Calculate the fuzzy reasoning value of the partial path (path) by applying operation * on path obtained by Cell[m,j] Θ Cell[i-m,j+m] //

Remove the duplicate paths from the Cell[i,j] if any;

If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the greatest fuzzy value;

Step 3. // Examine the partial path whose last node is destination node //

For i:= 1 to n do

Paths stored in cell[i,1] contain partial path with its fuzzy reasoning value;

End of Algorithm 1.

Algorithm construct the pyramid and Cell[n,1] contains the destination node with maximum fuzzy value. The FRP can be obtained in linear time if each Cell[i, j] is associated with the pointers of variables stored in the matching pair Cell[m, j] and Cell[i-m, j+m] during the construction of the pyramid. The time-complexity for finding fuzzy reasoning path is $O(n^3)$.

The space complexity of algorithm is $O(n^2)$, where n is the number of level in the FDG

6. PROPOSED ALGORITHM

6.1 Framework

To establish the parallel framework to be adopted in a PRAM model based algorithm, let us take possible entries of Cell[4,1] of the pyramid of size n=4 from three different ways, as shown in Fig. 3. The entries in the root element Cell[4,1] are filled from the following different pairings:

Cell[3,1] and Cell[1,4];

Cell[2,1] and Cell[2,3];

Cell[1,1] and Cell[3,2];

Entries to Cell[4,1] can be filled only after the cells involved in pairing have completely been constructed. By complete construction we mean that no more partial paths will be entered in it. For instance, Cell[3,1] must be completely constructed from pairing of Cell[1,1] and Cell[2,2] as well as from pairing of Cell[2,1] and Cell[1,3]. The entries in Cell[4,1] shall be filled sequentially in three different iterations. In general, Cell[i,j] is constructed by operations from i-1 pairing in i-1 different iterations. Thus in the PRAM model based algorithm, the merge operation will be done sequentially in different iterations in the design of the proposed algorithm, we shall consider the following points:

(i) No concurrent writes in a cell are allowed. The Operations in any cell is done sequentially in different iterations

(ii) Read operation can be done simultaneously from the same location by a number of processors.

(iii) In any iteration, the entries can be filled up in more than one row.

(iv) A cell can be used for pairing in any iteration if it has been constructed completely in any previous iteration. Thus, the ordering of pairings will be decided by the complete construction of cells.

The entries in the different cells of rows and columns of the pyramid can be filled up by employing the required number of processors.

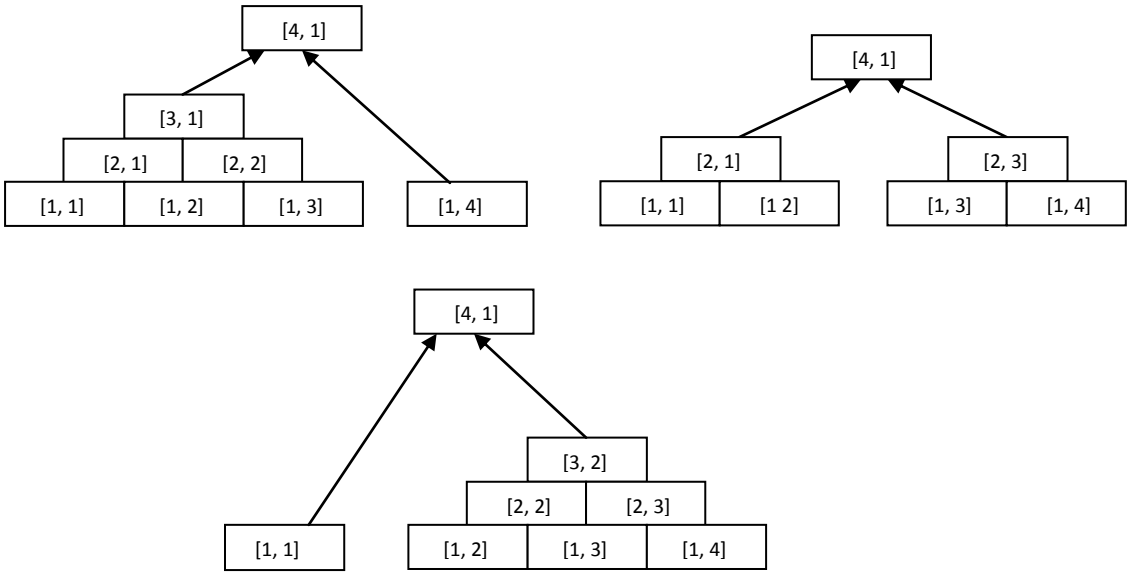

Fig 3: Three possible ways of constructing Cell[4,1] of pyramid of size n=4.

6.2 INFORMAL DESCRIPTION

Before we present the formal description of the algorithm, we shall discuss it informally. Let us take a pyramid of size i=4. First we take inputs in the base cell i=0 from FDG. In Cell[0,1] source node and its fuzzy value is entered from FDG, then the adjacent nodes of source node in entered in Cell[0,2]. Similarly adjacent of Cell[0,i-1] is entered in Cell[0,1]. All the nodes of FDG is entered in this way. After the entry of all the adjacent nodes of FDG, we can start the construction of upper row of the pyramid. We illustrate the construction of upper row (i>1) of the pyramid, iteration by iteration, as follows:

Iteration 1: In this iteration, the row i=1 can be filled up in parallel by the entries in base cell of the pyramid. All cells in this row will be filled up in time O(c) where c is constant using n (=4) processors.

Iteration 2: The cells in row of length i=2 can be filled up in parallel from pairing of the cells of the row i=1. For instance, Cell[2,1] can be constructed by pairing of Cell[1,1] and Cell[1,2] and so on. Construction of this row will require n-1, i.e. 3 processors in constant time, O(c).

Iteration 3: In the previous iterations, the constructions of the rows of length i=1 and i=2 were complete and these rows can be used for construction of the cells in upper rows. In the third iteration, Cell[3,1] in the row of length 3 can be filled up from one of two different pairings, i.e. Cell[1,1] and Cell[2,2] or Cell[2,1] and Cell[1,3]. The output obtained by operations on both pairings cannot be written simultaneously to avoid concurrent writes. Therefore, the output of first pairing is filled in this iteration and the output from other pairing will be filled in the next iteration. Cell[3,2] can also be filled up in a similar way. The number of processor required in this iteration will be 2.

Iteration 4: In this iteration, the second entry in the cells of the third row can be computed. Cell[3, 1] will be constructed by pairing of Cell[2,1] and Cell[1,3]. This entry was not filled in the third iteration to avoid concurrent writes. First entry in Cell[4,1] is made by pairing of Cell[2,1] and Cell[2,3]. The number of processors required 2.

Iteration 5: In this iteration, the second entry in the cell of row i= 4 can be filled by pairing of cells in the rows i=3 and i=1. Cell[4,1] can be constructed from the from the pairing of Cell[1,1] and Cell[3,2] using one processor.

Iteration 6: In this iteration, the cell in the row of length i=4 can be filled up. The construction of Cell[4,1] is completed in this from pairing of Cell[3,1] and Cell[1,4].

Thus all cells of the pyramid of size n=4 can be filled up in 6 iterations.

Figure.4. shows the iterations required for constructing the pyramid for size n=4 and n=5. The iterations are shown at different rows.

Now we can generalize the relationship between iteration k and row i. Table 1 shows different iteration required for construction of cell of row i. Table 2 shows different rows constructed in each iteration k.

6.3 FORMAL DESCRIPTION OF THE ALGORITHM 2

From the observation of the above algorithm reveals that loops for index variables i and j in the algorithm can be executed in parallel by employing more number of processors. After inserting all the nodes in the base cell (i=0) of pyramid in Step 1, the entries for the cell of row i=1 of the pyramid can be filled in parallel by employing n processors. In step 2, the cells in rows i>1 can be computed in parallel. For even i(i > 2) the computation of entries in Cell[i,j] starts at iteration number i-1 and finishes at 2i-2. No entry is computed during iteration i. For odd i(i>1), the computation of entries in Cell[i,j] starts at iteration i and finishes at 2i-2. For i ($i \geq 2$), the first entry starting at iteration i-1 can be delayed by one step and made to start at iteration i. Thus, the computation of entries in cells of even numbered rows can start at iteration i and finish at iteration 2i-2

Table 1. Iteration required for construction of row in Algorithm 2

Row (i)	**Iteration (k) required**
1	1
2	2
3	3,4
4	4, 5, 6
5	5, 6, 7, 8
6	6, 7, 8, 9, 10
7	7, 8, 9, 10, 11
8	8, 9, 10, 11, 12, 13, 14
9	9, 10, 11, 12, 13, 14, 15, 16
10	10, 11, 12, 13, 14, 15, 16, 17, 18
. . .	. . .
odd i (i>1)	i, i+1,..., 2i-2
even i (i>2)	i, i+1,..., 2i-2

Table 2. Rows constructed in each iteration of Algorithm2.

Iteration No. (k)	Rows (i) constructed
1	1
2	2
3	3
4	3, 4
5	4, 5
6	4, 5, 6
7	5, 6, 7
8	5, 6, 7, 8
9	6, 7, 8, 9
10	6, 7, 8, 9, 10
. . .	. . .
odd k (k>1)	(k+3)/2 to k
even k (k≥2)	(k+2)/2 to k

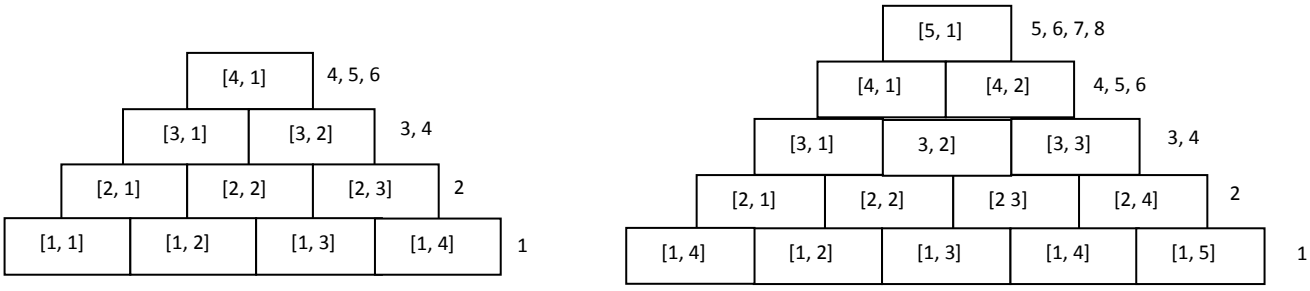

Fig 4 : Numbers beside the rows indicate the iterations required at each row of pyramids of size n=4 and n=5.

6.4 Algorithm 2

On-line Parallel algorithm for finding maximum FRP in FDG using CKY-algorithm:
Let G=<V, E, f, C> be a FDG and let l be the maximum level of the tree in G. Construct a pyramid of size l from parallel construct of i and j as follows, where each Cell[i,j] contain partial path and its fuzzy reasoning value.

Step 1. // Construct the bottom row (i=0) from the FDG G using the rule P→Q //

If i=0 and j=1 do

Cell[0,1]:= insert the source node with its fuzzy value;

If j ≥ 2 and all the nodes and its adjacent nodes of the FDG are stored in the base cells of the pyramid do

Cell[0,j]:=insert the adjacent nodes of the node stored in Cell[0,j-1];

j:=j+1;

EndIf;

Step 2. // Construct the upper rows using rule P→Q for length i>0 //

n:= number of cells in the base row -1;

for k:= 1 to 2n – 2 do

begin

if (k is odd) then

p_1:= (k+3) / 2

else

p_2:= (k+2) / 2;

if k > n then

p_2 := n

else

p_2:= k;

// p_1 and p_2 are lower and upper limits of row numbers in a particular iteration //

For i:= p_1 to p_2 in parallel do // parallel construction of rows //

For j:= 1 to n-i+1 in parallel do // parallel construction of cells in a row //

If k is even then

Cell[i, j] := Cell[k/2, j] Θ Cell[(i-k/2), (j + k/2)]

// Insert the partial path by applying operation Θ //

$f(p_k)*f(p_{k+1})$

// Calculate the fuzzy reasoning value of the partial path (path) by applying operation * on path obtained by Cell[k/2, j] Θ Cell[(i-k/2), (j + k/2)] //

Else

Cell[i, j] := Cell[i-(k+1)/2, j] Θ Cell[(k+1)/2, j+i-(k+1)/2]

// Insert the partial path by applying operation Θ //

$f(p_k)*f(p_{k+1})$

// Calculate the fuzzy reasoning value of the partial path (path) by applying operation * on path obtained by Cell[i-(k+1)/2, j] Θ Cell[(k+1)/2, j+i-(k+1)/2] //

Remove the duplicate paths from the Cell[i,j] if any;

If two or more paths reach a common node, delete all those paths except the one that reaches the common node with the greatest fuzzy value;

Step 3.

For i:= 1 to n do

Paths stored in cell[i,1] contain partial path with its fuzzy reasoning value;

End of Algorithm 2.

6.5 TIME COMPLEXITY OF ALGORITHM

Step 1 of the algorithm takes n+1 time to enter the values from FDG. In step 2, the loop of variable k is sequential and it executes 2n-2 times. Other loops (for variables i and j) execute in parallel and take c_1 time, where c_1 is a constant. Thus, step 2 takes 2n-2 steps.

Therefore, the time complexity of Algorithm2 is given as follows:

$$(n+1)+c_1 \sum_{k=2}^{2n-2} 1 = n+1+c1\ (2n-3) \qquad (3)$$

The constant c1 depends upon the FDG. Thus the time complexity of Algorithm is O(n), where n is the length of the longest path in FDG G.

Processor complexity of Algorithm We now show that Algorithm 2 takes $O(n^2)$ processors to find the fuzzy reasoning path of having n edges (length of path n). Step 1 of the algorithm takes input from FDG by finding adjacent nodes of the current node. It will take n iterations to enter all the nodes and adjacent nodes in FDG using one processor. After the entry of base row of the pyramid Step 2 of the algorithm computes the entries in the cells of row i=1 using n processors. In iteration k=2, n – 1 cells of row i=2 are constructed. There will be n – 1 processor required to construct the row i=2. Now in other iterations k>2, the number of processors will depend upon the number of rows being constructed. For instance, in the third iteration k = 3, row i=3 is constructed, using n – 3 + 1 processors. The number of processors required for constructing row i in any iteration is n – i + 1. Thus in any iteration k≥3, the numbers of processors required are given as follows

For odd $k \leq n$, $N = \sum_{i=(k+3)/2}^{k}(n - i + 1)$ (4)

For odd k>n, $N = \sum_{i=(k+3)/2}^{n}(n - i + 1)$. (5)

For even $k \leq n$, $N = \sum_{i=(k+2)/2}^{k}(n - i + 1)$. (6)

For even k>n, $N = \sum_{i=(k+2)/2}^{n}(n - i + 1)$. (7)

Table 3 shows different rows i constructed and the number of processors required in each iteration k for input size $n \geq 3$. The number of processors given by (4) and (6) is higher than given by (5) and (7) as the upper limit of i is restricted to n in (4) and (6). We can generalize the result of the maximum number in terms of n from the relationship between n and odd k as given in (3). After applying algebraic manipulation, the summation of the series in (3) can be obtained as follows:

$N = 1/2(n+1)(k) - 1/8(k)(3k+5)$ (8)

Since the value of k is constrained by 2n-2, the expression of N in (7) is $O(n^2)$. Thus the processor complexity of Algorithm 2 is $O(n^2)$.

Table 3. No. of processors required in each iteration of Algorithm 2

		N							
k	i	3	4	5	6	7	8	9	10
1	1	3	4	5	6	7	8	9	10
2	2	2	3	4	5	6	7	8	9
3	3	2	2	3	4	5	6	7	8
4	3,4	1	3	5	7	9	11	13	15
5	4,5		1	3	5	7	9	11	13
6	4,5,6		1	3	6	9	12	15	18
7	5,6,7			1	3	6	9	12	15
8	5,6,7,8			1	3	6	10	14	18
9	6, 7, 8, 9				1	3	6	10	14
10	6,7,8,9,10				1	3	6	10	15

6.6 EXAMPLE

Now we can observe the applicability of finding FRP using CYK algorithm with an example. We take an example of a FDG shown in Figure 1.

Let A be the source node and Z be the goal node. We want to establish the FRP between node A and node Z with the greatest fuzzy value. The application of algorithm results in the pyramid of Fig 5. We get the fuzzy reasoning path with maximum fuzzy value as follows:

First it entered the source node A and its fuzzy value (0.9) in the Cell[0,1] of the pyramid. Then it will check the adjacent nodes of the source node and store it in Cell[0,2] . The

adjacent node of the source node A is B and D, it will store the node B and D in Cell[0, 2] of the pyramid. Similarly node C, E and D are stored in Cell[0,3], node E, Z are stored in Cell[0,4] and node Z is stored in Cell[0, 5]. In this way all the nodes and its adjacent nodes are stored in the base row of the pyramid. After the entry of FDG in the pyramid it will start building of upper row of the pyramid. Following are the iteration for building upper row and finding the paths and partial path with its fuzzy reasoning value.

First Iteration: Algorithm executes two operations Ɵ and * on the content of cells. Using one processor it stores partial fuzzy reasoning paths in Cell [1,1] by applying operation Ɵ on Cell [0,1] and Cell [0,2] and calculates the fuzzy value of all the partial paths by applying operation *. Partial paths are AB, AD and its fuzzy values are stored in Cell[1,1]. Its fuzzy value is

f(AB) = f(A)*C(AB) = 0.9*0.9 = 0.81

f(AD) = f(A)*C(AD) 0.9*0.75 = 0.675

Using second processor, entries in Cell [1,2] is made from Cell [0,2] and cell [0,3]. Entries are edges BC, BD, BE and DE with its fuzzy value

C(BC) = 0.95, C(BD) = 0.85, C(BE) = 0.7, C(DE) = 0.9

Using third processor, entries in Cell [1,3] is made from Cell [0,3] and Cell [0,4]. Entries are edges CE, DE, CZ, EZ with its fuzzy value

C(CE)=0.85, C(DE)=0.9, C(CZ)=0.75, C(EZ)=0.8

Using Forth processor, Entries in Cell [1,4] is made with Cell [0,4] and Cell [0,5]. Entry in Cell [1,4] is edge EZ with its fuzzy value C(EZ)=0.8.

After this all entries in the row i=1 is completed using 4 processors.

Second Iteration: In this iteration construction of row i=2 starts and completed using three processor. Using First processor, entries in Cell [2,1] is filled from entries in Cell [1,1] and Cell [1,2]. Partial paths ABC, ABD, ABE, AND ADE and fuzzy partial path with the following values

f(ABC)=f(AB)*C(BC)=0.81*0.95=0.7695

f(ABD)=f(AB)*C(BD)=0.81*0.85=0.6885

f(ABE)=f(AB)*C(BE)=0.81*0.7 =0.567

f(ADE)=f(AD)*C(DE)=0.675*0.9=0.6075

are stored in Cell[2,1] of the pyramid. Partial path ABE and ADE reaches at the same node with different fuzzy value. Partial path with greatest fuzzy value (ADE) is selected and other paths are discarded according to the dynamic programming principle. Partial path with greatest fuzzy value is ABD; its label is set to one.

Using second processor entries in Cell [2,2] are made with Cell [1,2] and Cell [1,3]. Partial paths BCE, BDE, BCZ, BEZ, DEZ are entered in Cell[2,2]

C(BCE)=C(BC)*C(CE)=0.95*0.85=0.8075

C(BDE)=C(BD)*C(DE)=0.85*0.9=0.765

C(BCZ)=C(BC)*(CZ)=0.95*0.75=0.7125

C(BEZ)=C(BE)*C(EZ)=0.7*0.8=0.56

C(DEZ) =C(DE)*C(EZ)=0.9*0.8=0.72

Using third processor, entries in Cell [2,3] are filled with Cell [1,3] and Cell [1,4]. Partial path CEZ and DEZ are stored with its following fuzzy value:

C(CEZ)=C(CE)*C(EZ)=0.85*0.8=0.68

C(DEZ)=C(DE)*C(EZ)=0.9*0.8=0.72

Third Iteration: Construction of row i=3 is made in this iteration using two processor. First entries in Cell [3,1] is made with Cell [1,1] and Cell [2,2]. Following are the entries:

f(ABCE)=f(AB)*C(BCE)=0.81*0.8075=0.6541

f(ABCZ)=f(AB)*C(BCZ)= 0.81*0.7125=0.577125

f(ABDE)=f(AB)*C(BDE)= 0.81*0.765=0.6197

f(ABEZ)=f(AB)*C(BEZ)= 0.81*0.56=0.4536

f(ADEZ)=f(AD)*C(DEZ)= 0.675*0.72=0.486

f(ABCE)=f(ABC)*C(CE)=0.7695*0.85=0.6541

f(ABCZ)=f(ABC)*C(CZ)= 0.7695*0.75=0.5771

f(ABDE)=f(ABD)*C(DE)= 0.6885*0.9=0.6197

f(ADEZ)=f(ADE)*C(EZ)= 0.6075*0.8=0.486

First entry in cell[4,1] can also be made in this iteration from Cell [2,1] and Cell [2,3]. Partial path ABCEZ and ABDEZ are entered with the following entries:

f(ABCEZ) = f(ABC)*C(CEZ)=0.7695*0.68=0.52326

f(ABDEZ) = f(ABD)*C(DEZ)=0.6885*0.72=0.49572

No of processors required for this iteration is 3.

Fifth Iteration: This iteration fills second entry in Cell[4,1] from Cell[1,1] and Cell[3,2]. Entries are partial paths ABCEZ and ABDEZ with their following fuzzy values:

f(ABCEZ)= f(AB)*C(BCEZ)=0.81*0.646=0.52326

f(ABDEZ)=f(AB)*C(BDEZ)=0.81*0.612=0.49572

Sixth Iteration: This iteration fill third entry in Cell [4,1] from Cell [3,1] and Cell [1,4]. Construction of row i= 4 is completed in this iteration.

f(ABCEZ) = f(ABCE)*C(EZ)=0.6541*0.8=0.52326

Cell[i,1] contain the partial paths with their fuzzy values. Cell[i,1] has the paths of length i. In the above example, Cell[4,1] has two paths with path length four. The path ABCEZ with maximum fuzzy value (0.52326) is selected.

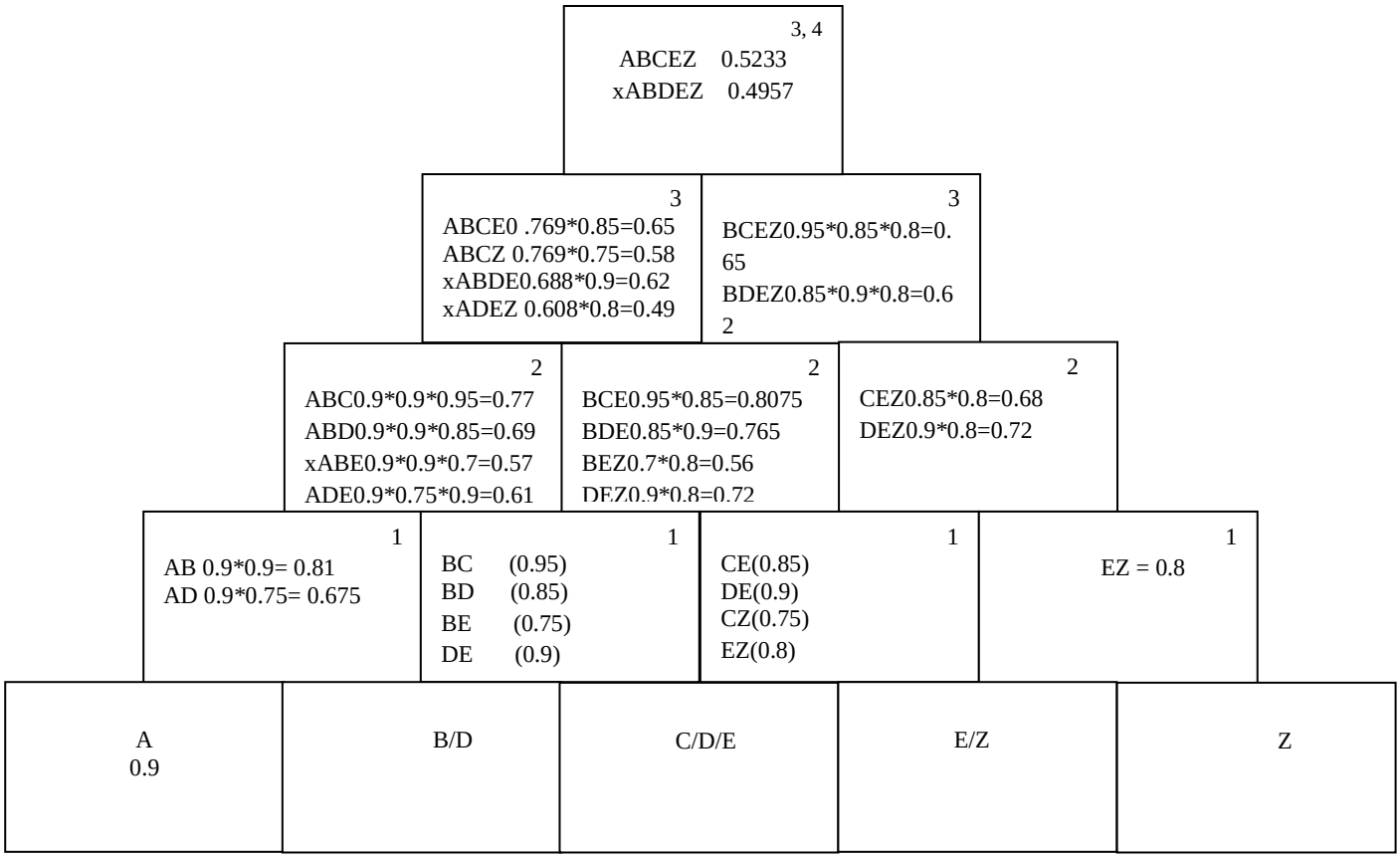

Fig. 5 : Parallel construction of CYK – pyramid for finding FRP.

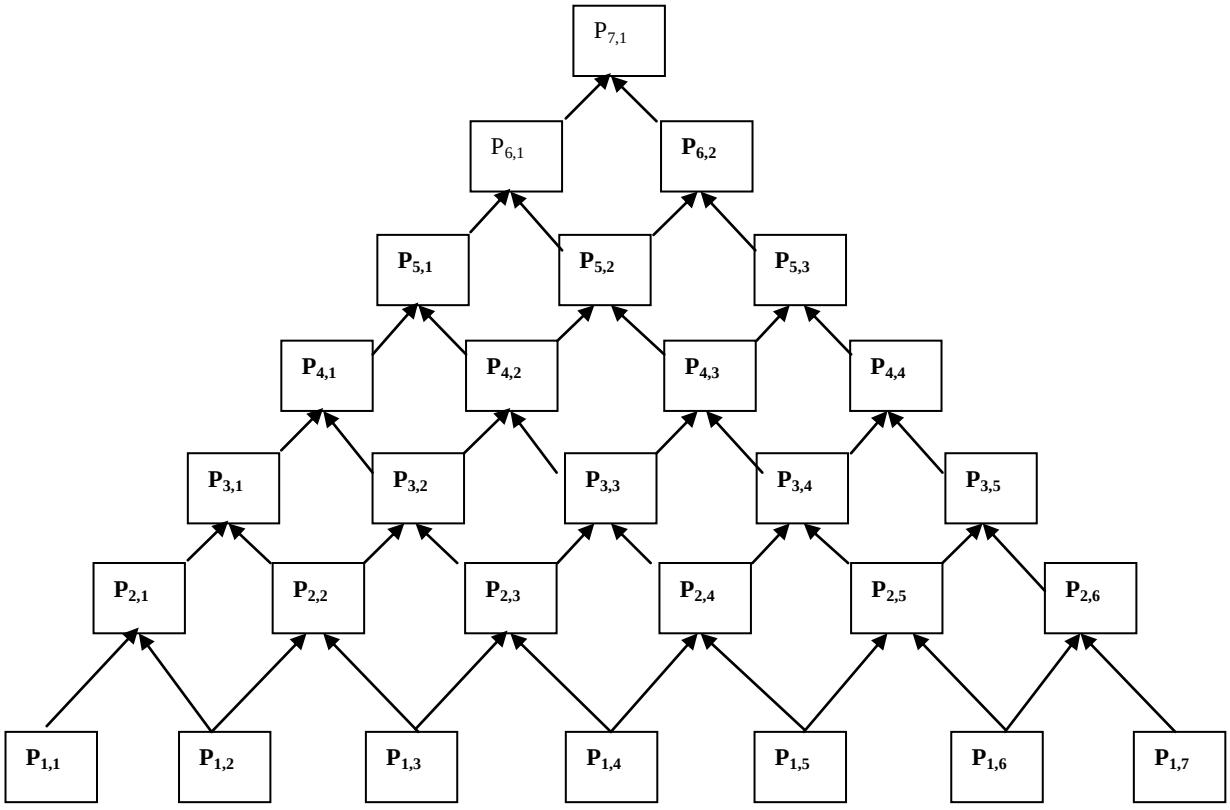

Fig 5: Processor architecture for pyramid (n=7).

7. ALGORITHM WORKING ON A MESH-CONNECTED ARCHITECTURE

The structure of pyramid constructed through our algorithm is suitable for two dimensional mesh connected architecture. Corresponding to each Cell[i,j], there is a processor $P_{i, j}$, which computes the entries for Cell[i, j]. Fig. 6 shows the processor interconnection structure for n=7.

For each entry cell[i,j] for i≥2 of the pyramid, the processor $P_{i, j}$ receives entries from processors $P_{i-1, j}$ and $P_{i-1, j+1}$. Processor $P_{i, j}$ transmits the entries received from $P_{i-1, j}$ to $P_{i+1, j}$ and entries received from $P_{i-1, j+1}$ to $P_{i+1, j-1}$. Each processor has got two input ports and two output ports, except the processors at boundaries of the pyramid. The order in which the entries are received by the processor $P_{i, j}$, $i \geq 2$ is given by the algorithm. The number of entries received by the processor $P_{i, j}$ is i-1. Each processor is provided with a local memory in which the row i and column j are stored. Since the number of entries required by the processor $P_{i, j}$ is i-1, the number of locations required by the processor $P_{i, j}$ is also i-1 and the total memory required will be $O(n^3)$. For instance, the processor $P_{5, 2}$ will have four locations to store the entries received in four iterations 5, 6, 7, 8 from two lower processors i.e., $P_{4, 2}$ and $P_{6,}$

8. DISCUSSION AND CONCLUSION

We have presented a new parallel algorithm for finding fuzzy reasoning path (FRP) in an OR-type FDG. The algorithm has been formulated on an 'indexed' PRAM model of computation. An OR-type FDG is represented by nodes and edges such that each node u is representative of a proposition p and each directed edge <u, v> represents a rule with antecedent proposition p corresponding to node u and consequent proposition q corresponding to node v. The FRP is a path that leads to the greatest fuzzy value to the goal node. For finding maximum FRP multiplication is the basic operation. The algorithm is based on dynamic programming search technique, which reduces the size of search tree by eliminating the redundant entries from the cell. Time complexity of this algorithm is O(n) [1, 13 and 15], where n is the length of longest path in the FDG. The number of processors required in the algorithm is $O(n^2)$. The algorithm is also suitable for mesh-connected architecture of processors.

9. REFERENCES

[1] M. Chandwani and N.S. Chaudhari, "Knowledge representation using fuzzy deduction graphs," IEEE Trans. Syst., Man, Cybern., vol. 26, no. 6 pp. 848-854, Dec. 1996

[2] C.L. Liu, "Elements of Discrete Mathematics", Third Edition, McGraw Hill Internationa Editions, 1985.

[3] M. Chandwani, M. Puranik and N.S. Chaudhari, "On CKY-Parsing of Context-Free Grammar in Parallel," IEEE region 10 Conference, Tencon, Australia, pp. 141-145, November 1992.

[4] M. Chandwani, N.S. Chaudhari, "Formulation and analysis of parallel context-free recognition and parsing on PARAM model," Elsevier Science, parallel Computing 22, 1996, 845-868.

[5] F. Hayes-Roth, "Rule-based Systems", Commun. ACM. vol. 28, no. 9, pp. 921-932, Sept. 1985.

[6] Hisao Ishibuchi,Takashi Yamamoto, "Rule Weight Specification in Fuzzy Rule-Based Classification Systems," IEEE Trans. Fuzzy Syst.,vol. 13,No. 4 pp.428-443,Aug. 2005.

[7] Elaine Rich,Kevin Knight, "Artificial Intelligence", Second Edition, Tata McGraw-Hill Editions, 1991.

[8] Ahmad M. Ibrahim, Introduction to Applied Fuzzy Electronics, Prentice-Hall of India, 1999.

[9] L.Zadeh, "Fuzzy logic," IEEE Comp. Mag., vol. 21, no. 4, pp. 83-93, Apr. 1988.

[10] C.C. Yang, "Deduction graph: An algorithm and Application," IEEE Trans. Software Eng., vol. 15, no. 1, pp. 60-67, Jan. 1989

[11] C.C. Yang, J.J. Chen, and H. L. Chau, "Algorithms for constructing minimal deduction graphs," IEEE Trans. Software Eng., vol. 15, no. 6, pp. 760-771, June 1989.

[12] A. Kandel, Fuzzy Mathematical Techniques with Applications. Reading, MA:Addison-Wesley,1986.

[13] Corman, Leiserson, Rivest, Stein, "Introduction to Algorithms", Second Edition, PHI Pub., 2006.

[14] A. Gibbons and W. Rytter, Efficient Parallel Algorithms,Cambridge University Press, Cambridge, UK, 1988.

[15] Thomas A. Sdkamp, "An Introduction to the Theory of Computer Science, Languages and Machines," Third Edition,Pearson Education, 2006.

[16] P. Shukla, M. Chandwani, "Taxonomy of Fuzzy Deduction Graph," International Journal of Computer and Electronics Engineering (IJCEE), Vol. 3, no.1, Page 99-115, 2010.

[17] S. M. Chen,"Representing knowledge using fuzzy deduction graphs based on fuzzy numbers, " Proceedings of the 2003 Joint Conference on AI, Fuzzy System, and Grey System, Taipai, Taiwan, Republic of China, Dec. 2003.

[18] Z. H. Tan, " Fuzzy Metagraph and Its Combination with the indexing Approach in Rule-based Systems," IEEE Transaction knowledge and Data Engineering, vol. 18,no. 6, pp. 829-841, June, 2006.