多次元ファジー推論の並列アルゴリズム

正 員 林 陽 一 (茨城大)
非会員 中 井 正 人 (茨城大)

1. はじめに

近年、ファジー推論⁽³⁾⁽⁴⁾は制御⁽⁹⁾、プロセスオートメーション⁽²⁾、医療診断⁽⁶⁾など各種のエキスパート・システムに適用され、その有効性が確かめられている。

しかし、多次元の場合のファジー推論についてはほとんど研究されておらず、SugenoとTakagi⁽⁵⁾によるものが知られているだけである。多次元ファジー推論はエキスパート・システムや複雑なシステムのファジー・モデリングであいまい性を扱う場合、特に重要である。多次元ファジー推論は“if (x_1 is P_1 , x_2 is P_2 , ..., x_n is P_n) then z is C ”といった前件が複数個の命題からなるインプリケーションを用いた推論である。ここで重要なことは多次元インプリケーションは単に普通の一次元インプリケーションの集まりではないということである。すなわち、前件の命題の間には“and”だけでなく任意の関係(例えば、 $z=x^2y$)が考えられる。

従来、多次元インプリケーションを扱う場合、“if (x_1 is P_1 , x_2 is P_2) then z is C ”というインプリケーションを“if x_1 is P_1 and x_2 is P_2 then z is C ”あるいは“if x_1 is P_1 then x_2 is P_2 then z is C ”などに変換して推論が行われているが、これらは前件の命題の間に“and”などの特定の論理関係を仮定していることになり適当でない。一般にファジー推論には二つの方法がある。一つはZadehの推論合成則を利用するもの、もう一つはLukasiewiczの無限値論理などの論理を拡張原理により拡張したファジー論理を用いるものである。後者の例として、Tsukamotoの提案したfuzzy modus ponensとfuzzy modus tollens⁽⁷⁾があ

る。SugenoとTakagiはこのTsukamotoの方法を利用した多次元ファジー推論の方法を提案している。SugenoとTakagiの方法では前件の命題の間に特定の関係を仮定せず、これを線形補間に似た方法で処理しており、有効な方法であると言える。しかし、この方法は結論を得るまでに一次元の推論を何回も繰返すため、計算量が非常に多く実用的とは言えない。

本論文では、SugenoとTakagiの多次元ファジー推論に並列処理を導入して高速化を図った方法を提案する。本論文の方法では、 n 次元ファジー推論の場合、2の n 乗個のプロセッサを用いて n に比例する計算量で推論を行うことができる。

2. 多次元ファジー推論

本章では、SugenoとTakagiによって提案された多次元ファジー推論⁽⁵⁾を n 次元に一般化して説明する。

一般に“if x is P then z is Q ”というインプリケーションと“ x is P ”という前提が与えられたとき、結論“ z is Q ”を推論するfuzzy modus ponensは、“ P ”と“ P ”が十分近くなければ有効な結論を得ることができない。有効な結論を得たい場合には、二つのインプリケーション“if x is P_1 then z is Q_1 ”, “if x is P_2 then z is Q_2 ”を用意し、前提が“ P_1 ”と“ P_2 ”の間に与えられたときだけ推論可能とすべきである。

n 次元の場合も同様で、前件を構成する n 個の命題をそれぞれを2種類ずつ用意する必要がある。すなわち以下の2の n 乗個のインプリケーションが必要となる。

$$I_{0..00}: (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1,0}, x_n \text{ is } P_{n0}) \\ \rightarrow z \text{ is } Q_{0..00}$$

$$I_{0..01}: (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1,0}, x_n \text{ is } P_{n1}) \\ \rightarrow z \text{ is } Q_{0..01}$$

$$\vdots \quad \quad \quad \vdots \quad \quad \quad \vdots$$

$$I_{1..11}: (x_1 \text{ is } P_{11}, \dots, x_{n-1} \text{ is } P_{n-1,1}, x_n \text{ is } P_{n1})$$

A Parallel Algorithm for Multi-Dimensional Fuzzy Reasoning.
By Yoichi Hayashi, Member & Masato Nakai, Non-member
(Faculty of Engineering, Ibaraki University)

林 陽一: 正員, 茨城大学工学部情報工学科

中井正人: 非会員, 茨城大学工学部情報工学科

$$\rightarrow z \text{ is } Q_{1..11}$$

多次元ファジー推論とは、上記のような2の n 乗個のインプリケーションおよび前提($x_1 \text{ is } P_1, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n$)が与えられたとき、“ z ”を推論する問題である。これらのインプリケーションが与えられているとして、前提($x_1 \text{ is } P_1, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n$)に対する結論“ $z \text{ is } Q$ ”を求めてみる。まず $I_{0..0}$ について、前提の中の“ x_1 ”から“ x_{n-1} ”までを $I_{0..0}$ の前件の“ $x_1 \text{ is } P_{10}$ ”から“ $x_{n-1} \text{ is } P_{n-1.0}$ ”に固定して次の fuzzy modus ponens を行う。

$$\begin{aligned} & (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n), (x_1 \text{ is } P_{10}, \\ & \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n) \rightarrow z \text{ is } Q_{0..0}/z \\ & \text{is } Q'_{0..0} \dots \dots \dots (1) \end{aligned}$$

ここで問題となるのは、次のような n 次元の逆真理値限定 (Converse of Truth Qualification) をどのように行うかということである。

$$\begin{aligned} & (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n) \\ & \simeq (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n) \\ & \text{is } \tau_{0..0} \dots \dots \dots (2) \end{aligned}$$

このとき前件と前提の“ $x_1 \text{ is } P_{10}$ ”から“ $x_{n-1} \text{ is } P_{n-1.0}$ ”までが共通であることを考えると、(2)式の $\tau_{0..0}$ は“($x_n \text{ is } P_n$) $\simeq$ ($x_n \text{ is } P_n$) is $\tau_{0..0}$ ”の $\tau_{0..0}$ と等価であると言ってよい。従って、(2)式は一次元の逆真理値限定に帰着され、簡単に解くことができる。

よって(1)式の結論は一次元の fuzzy modus ponens と全く同様にして求めることができる。その他のインプリケーション $I_{0..01}$ から $I_{1..11}$ についても同様に前提の“ x_1 ”から“ x_{n-1} ”を前件のそれぞれの命題に固定することによって一次元の fuzzy modus ponens に帰着し、それぞれの結論を得る。

次に得られた結論から新しいインプリケーションを合成する。合成は線形補間と似た方法で行われる。まず前件の“ x_1 ”から“ x_{n-1} ”までが共通であった $I_{0..0}$ と $I_{0..01}$ の結論“ $z \text{ is } Q'_{0..0}$ ”, “ $z \text{ is } Q'_{0..01}$ ”を合成し、以下の新しいインプリケーションを作る。

$$\begin{aligned} & (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n) \\ & \rightarrow z \text{ is } Q_{0..0} \end{aligned}$$

ここで、 $Q_{0..0} = (Q'_{0..0} + Q'_{0..01})/2$ である。

他の結論についても同様に合成を行い、次の新しいインプリケーション $I_{0..0}$ から $I_{1..11}$ を得る。

$$\begin{aligned} & I_{0..0}: (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n) \\ & \rightarrow z \text{ is } Q_{0..0} \\ & I_{0..1}: (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1.1}, x_n \text{ is } P_n) \\ & \rightarrow z \text{ is } Q_{0..1} \end{aligned}$$

$$\begin{aligned} & \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ & I_{1..1}: (x_1 \text{ is } P_{11}, \dots, x_{n-1} \text{ is } P_{n-1.1}, x_n \text{ is } P_n) \\ & \rightarrow z \text{ is } Q_{1..1} \end{aligned}$$

これらの新しいインプリケーションについて推論を繰返す。まず $I_{0..0}$ について前提の“ x_1 ”から“ x_{n-2} ”までを前件のそれぞれの命題に固定して次の fuzzy modus ponens を行う。

$$\begin{aligned} & (x_1 \text{ is } P_{10}, \dots, x_{n-1} \text{ is } P_{n-1}, x_n \text{ is } P_n), (x_1 \text{ is } P_{10}, \\ & \dots, x_{n-1} \text{ is } P_{n-1.0}, x_n \text{ is } P_n) \rightarrow z \text{ is } Q_{0..0}/z \text{ is } Q'_{0..0} \\ & \dots \dots \dots (3) \end{aligned}$$

前提と前件の“ $x_1 \text{ is } P_{10}$ ”から“ $x_{n-2} \text{ is } P_{n-2.0}$ ”および“ $x_n \text{ is } P_n$ ”は共通であるから、この fuzzy modus ponens は一次元の fuzzy modus ponens に帰着でき、簡単に結論を得られる。同様にインプリケーション $I_{0..1}$ から $I_{1..1}$ についても“ x_1 ”から“ x_{n-2} ”までを固定して fuzzy modus ponens を行う。

これらの結論をまた合成して新しいインプリケーションを得る。順番に固定する命題を少なくしながら、以上の手続きを繰返し、最終的結論を得る。最終的結論は次の二つの fuzzy modus ponens の結論を合成したものである。

$$\begin{aligned} & (x_1 \text{ is } P_1, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n), (x_1 \text{ is } P_{10}, x_2 \text{ is } \\ & P_2, \dots, x_n \text{ is } P_n) \rightarrow z \text{ is } Q_0/z \text{ is } Q'_0 \\ & (x_1 \text{ is } P_1, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n), (x_1 \text{ is } P_{11}, x_2 \text{ is } \\ & P_2, \dots, x_n \text{ is } P_n) \rightarrow z \text{ is } Q_1/z \text{ is } Q'_1 \end{aligned}$$

すなわち、 $Q = (Q'_0 + Q'_1)/2$ とすると、最終的結論は“ $z \text{ is } Q$ ”である。

多次元ファジー推論の逆問題は、“ $x_2 \text{ is } P_2$ ”から“ $x_n \text{ is } P_n$ ”と“ $z \text{ is } Q$ ”が与えられて、“ x_1 ”を推論する問題である。これについては、最後の二つの fuzzy modus ponens が次の fuzzy modus tollens に代わり、それらの結論を合成して最終的結論を得る。

$$\frac{z \text{ is } Q, (x_1 \text{ is } P_{10}, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n) \rightarrow z \text{ is } Q_0}{(x_1 \text{ is } P'_{10}, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n)}$$

$$\frac{z \text{ is } Q, (x_1 \text{ is } P_{11}, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n) \rightarrow z \text{ is } Q_1}{(x_1 \text{ is } P'_{11}, x_2 \text{ is } P_2, \dots, x_n \text{ is } P_n)}$$

すなわち、 $P = (P'_0 + P'_1)/2$ とすると、最終的結論は“ $x_1 \text{ is } P$ ”である。

3. 並列多次元ファジー推論

前章では Sugeno と Takagi の多次元ファジー推論について n 次元の場合に一般化して述べたが、本章では多次元ファジー推論に並列処理を導入した並列多次元ファジー推論を提案し、Ada⁽⁸⁾インプリメンテー

ションの例を示す。

前章で述べたように Sugeno と Takagi の方法では、まず前件を構成する n 個の命題の中から一つに着目し、他の $n-1$ 個の命題をそれぞれ前提の対応する命題に固定することにより、一次元のファジー推論に帰着し、推論を行う。インプリケーションは 2 の n 乗個あり、それぞれについて推論が必要であるから、fuzzy modus ponens が 2 の n 乗回適用されることになる。次に新しいインプリケーションが合成される。二つの結論から一つの新しいインプリケーションが合成されるから、新しいインプリケーションは 2 の $n-1$ 乗個できる。この新しいインプリケーションについて同様の推論、合成を繰返すことによって最終的結論を得る。この過程を図 1 にまとめた。

図 1 の $R11 \sim R1.2^n, R21 \sim R2.2^{n-1}, \dots, Rn1 \sim Rn2$ は fuzzy modus ponens に対応する。まず第 1 段階で 2^n 個の fuzzy modus ponens $R11$ から $R1.2^n$ が順番に行われる。このとき $R11$ から $R1.2^n$ のそれぞれは与えられた 2 の n 乗個のインプリケーションの中の一つについてのみ推論を行うだけで、他の fuzzy modus ponens に影響を与えないし、影響を受けない。同様に第 i 段階についてもそれぞれの fuzzy modus ponens は全く独立に推論を行う。

ここで 2 の n 乗個のプロセッサをもつ MIMD 方式の並列計算機⁽¹⁾の使用を仮定する。それぞれのプロセッサの一つの fuzzy modus ponens を行うタスクを割当てる。すなわちプロセッサ 1 に $R11$ 、プロセッサ 2 に $R12$ 、 $\dots$ 、プロセッサ 2^n に $R1.2^n$ を割当てる。各プロセッサは他のプロセッサに独立に計算を進めることができるので、第 1 段階の 2 の n 乗個の fuzzy modus ponens を同時に処理することができる。同様

図 1 多次元ファジー推論の流れ

Fig.1. Process of multi-dimensional fuzzy reasoning.

に第 i 段階についても一つのプロセッサに一つの fuzzy modus ponens を割当てることによって、それぞれの段階の fuzzy modus ponens を同時に処理できる。インプリケーションの合成についても同様であり、図 1 に示してあるように合成操作は前の段階の結論のみを参照するだけで、それぞれの合成操作は互いに独立に計算を進められる。従って、合成操作も一つ一つをプロセッサに割当てることで同時に処理することができる (表 1)。

この考えに基づいた並列処理言語 Ada によるインプリメンテーションを図 2 (a)~(c) に示す。これは N 次元ファジー推論を並列に実行するものである。

まず、使われているデータ形の説明をする。MF 形はメンバシップ関数 (Membership Function) を表す形である。LTV 形は言語的真理値 (Linguistic Truth Value) を表す形である。PREMISES 形は前提を表す MF 形の配列である。すなわち、PREMISES 形の変数 PREMISE は次のような前提を表す。

$(x_1 \text{ is PREMISE}(1), \dots, x_N \text{ is PREMISE}(N))$

RULES 形はインプリケーションを表現する形であり、ANT, CONC, LTV-RULE からなるレコード形である。ここで ANT はインプリケーションの前件を表す MF 形の配列であり、CONC は結論、LTV-RULE はインプリケーション自身の言語的真理値を表す。すなわち、RULES 形は次のようなインプリケーションを表す形である。

$(x_1 \text{ is ANT}(1), x_2 \text{ is ANT}(2), \dots,$

$x_N \text{ is ANT}(N)) \rightarrow z \text{ is CONC}$

変数 RULE は RULES 形の配列であり、初期状態では N 次元推論に必要な 2 の N 乗個のインプリケーション ($RULE(1) \sim RULE(2^n)$) からなっているとす。タスク FMP はインプリケーション (RULE)、前提 (PREMISE)、前件の何番目に着目するのかの情報 (I) を受取り、PREMISE と RULE の前件の I

表 1 各プロセッサとタスクの割当て
Table 1. Task assignment to each processor.

時間	プロセッサ 1	プロセッサ 2	プロセッサ 2^{n-1}	プロセッサ 2^n	プロセッサ 2^n
1	$R 11$	$R 12$	$R 1.2^{n-2}$	$R 1.2^{n-1}$	$R 1.2^n$
2	合成 11	合成 12	合成 1.2^{n-3}	合成 1.2^{n-1}	—
3	$R 21$	$R 22$	$R 2.2^{n-1}$	$R 2.2^{n-1}$	—
4	合成 21	合成 22	合成 2.2^{n-2}	—	—
5	$R 31$	$R 32$	$R 3.2^{n-2}$	—	—
2^{n-1}	$R n 1$	$R n 2$	—	—	—
2^n	結論	—	—	—	—

```

procedure PARALLEL_MULTI_DIMENSIONAL_FUZZY_REASONING is
  type PREMISES is array(1..N) of MF;
  type RULES is
    record
      ANT :array(1..N) of MF;           -- collection of antecedents
      CONC :MF;                         -- consequence
      LTV_RULE :LTV;                   -- truthfulness(LTV) of inference rule
    end record;
  task type FMP is                     -- one dimensional fuzzy modus ponens
    entry INITIALIZE(RULE : in RULES; PREMISE :in PREMISES; I :in integer);
    entry COLLECT(CONSEQUENCE :out MF);
  end FMP;
  task body FMP is
    begin
      accept INITIALIZE(RULE :in RULES; PREMISE :in PREMISES; I :in integer) do
        MY_RULE := RULE; MY_PREMISE := PREMISE; STEP := I;
      end INITIALIZE;
      LTV_ANT := CTQ(MY_PREMISE(STEP),MY_RULE.ANT(STEP));
      LTV_CONC := REASONING(LTV_ANT,MY_RULE.LTV_RULE);
      CONC := TQ(LTV_CONC,MY_RULE.CONC); -- Tsukamoto's one dimensional fuzzy reasoning
      -- truth qualification
      accept COLLECT(CONSEQUENCE :out MF) do
        CONSEQUENCE := CONC;
      end COLLECT;
    end FMP;
end PARALLEL_MULTI_DIMENSIONAL_FUZZY_REASONING;

```

(a)

```

procedure PARALLEL_FMP(I :in integer) is
  FMP_TASK :array(integer range 1..2*I) of FMP;
  -- create FMP type tasks FMP_TASK(1),...,FMP_TASK(2*I)
  begin
    for J in 1..2*I loop
      FMP_TASK(J).INITIALIZE(RULE(J),PREMISE,I); -- entry call,
      -- transmit an inference rule, a premise and step No. to each task
    end loop;
    for J in 1..2*I loop
      FMP_TASK(J).COLLECT(RULE(J).CONC);
      -- entry call, receive a consequence of reasoning from each task
    end loop;
  end PARALLEL_FMP;

  task type COMBINE is -- two rules, RULE1 and RULE2 are combined to make a NEW_RULE
    entry INITIALIZE(RULE1, RULE2 :in RULES);
    entry COLLECT(NEW_RULE :out RULES);
  end COMBINE;
  task body COMBINE is
    begin
      accept INITIALIZE(RULE1, RULE2 :in RULES) do
        MY_RULE1 := RULE1; MY_RULE2 := RULE2;
      end INITIALIZE;
      MY_NEW_RULE.CONC := INTERPOLATION(MY_RULE1.CONC,MY_RULE2.CONC);
      -- MY_NEW_RULE.CONC := 1/2(MY_RULE1.CONC + MY_RULE2.CONC)
      accept COLLECT(NEW_RULE :out RULES) do
        NEW_RULE := MY_NEW_RULE;
      end COLLECT;
    end COMBINE;
end COMBINE;

```

(b)

```

procedure COMBINE_RULE(I :in integer) is
  COMBINE_TASK :array(integer range 1..2*(I-1)) of COMBINE;
  -- create COMBINE type tasks COMBINE_TASK(1),...,COMBINE_TASK(2*(I-1))
  begin
    for J in 1..2*(I-1) loop
      COMBINE_TASK(J).INITIALIZE(RULE(2*J-1),RULE(2*J));
      -- entry call, transmit two inference rules to be combined to each task
    end loop;
    for J in 1..2*(I-1) loop
      COMBINE_TASK(J).COLLECT(RULE(J));
      -- entry call, receive a new inference rule from each task
    end loop;
  end COMBINE_RULE;

  begin
    -- main routine
    for I in reverse 1..N loop -- N is the number of propositions in an antecedent
      PARALLEL_FMP(I);
      COMBINE_RULE(I);
    end loop;
  end PARALLEL_MULTI_DIMENSIONAL_FUZZY_REASONING;
  -- RULE(1).CONC is a final consequence of reasoning
  -- MF is membership function type
  -- LTV is linguistic truth value type
  -- RULE is an array of rules type

```

(c)

図 2 Ada による多次元ファジー推論の並列アルゴリズム

Fig. 2. Ada implementation of a parallel algorithm for multi-dimensional fuzzy reasoning.

番目の命題について以下のような fuzzy modus ponens を行い、その結論 (CONSEQUENCE) を返す。

PREMISE (I), RULE. ANT (I)

→ RULE. CONC/CONSEQUENCE

タスク COMBINE は二つのインプリケーション (RULE1, RULE2) を受取り、それらを合成して新しいインプリケーション (NEW-RULE) を返す。

手続き PARALLEL-FMP(1) はタスク FMP を行うための 2 の I 乗個のプロセッサ (FMP-TASK(1)~FMP-TASK(2')) を用意し、各プロセッサ (FMP-TASK(J), $J=1, \dots, 2'$) に RULE(I), PREMISE, I を引き渡し、推論を行わせる。各プロセッサは引き渡されたデータをもとに独立に推論を進める。各プロセッサが推論を終了したなら、PARALLEL-FMP は各プロセッサ (FMP-TASK(J)) から結論 (CONSEQUENCE) を RULE(J). CONC に回収する。

手続き COMBINE - RULE(I) はタスク COMBINE を行うための 2 の $I-1$ 乗個のプロセッサ (COMBINE - TASK(1)~COMBINE - TASK(2'⁻¹)) を用意し、各プロセッサ (COMBINE - TASK(J), $J=1, \dots, 2'^{-1}$) に合成される二つのインプリケーション RULE(2* $J-1$), RULE(2* J) を引き渡す。各プロセッサは引き渡されたデータをもとに独立に合成を行う。各プロセッサが新しいインプリケーションの合成を終了したなら、COMBINE-RULE は各プロセッサ (COMBINE-TASK(J)) から新しいインプリケーション (NEW-RULE) を RULE(J) に回収する。

メインルーチンは PARALLEL-FMP(I), COMBINE-RULE(I) を、前件の次元数 N から 1 まで I を 1 ずつ減らしながら繰返す。最終的結論は RULE(1)-CONC に得られる。

4. 理論的計算量の比較

本章では、第 2 章で述べた多次元ファジー推論と第 3 章で提案した並列多次元ファジー推論の理論的計算量の比較を行う。

Sugeno と Takagi の n 次元ファジー推論について次の二つの定理を示す。

〔定理 1〕 Sugeno と Takagi の方法では、 n 次元ファジー推論のために fuzzy modus ponens を $2(2^n - 1)$ 回適用する。

〔証明〕 一次元ファジー推論のために fuzzy modus ponens が 2 回適用されることは明らか。

($n-1$) 次元ファジー推論のために fuzzy modus ponens が $2(2^{n-1} - 1)$ 回適用されると仮定する。

n 次元ファジー推論の場合、まず " x_1 ", " $\dots$ ", " x_{n-1} " がインプリケーションの命題に固定され、" x_n " について fuzzy modus ponens が適用される。次にインプリケーションの合成により、2 の $n-1$ 乗個のインプリケーションができる。合成された新しいインプリケーションは " x_n " が前提の命題に固定されているので、 $n-1$ 次元ファジー推論の場合に帰着できる。すなわち

IF (x_1 is $P_{1*}, \dots, x_{n-1}$ is $P_{(n-1)*}$, x_n is P_n)
THEN z is Q'

は

IF (x_1 is $P_{1*}, \dots, x_{n-1}$ is $P_{(n-1)*}$)
THEN z is Q'

ここで、" $*$ " : " 0 " か " 1 "

に帰着できる。

従って、数学的帰納法により定理 1 は証明される。

〔証明終〕

〔定理 2〕 Sugeno と Takagi の方法では n 次元ファジー推論の逆問題を解くために fuzzy modus ponens を、 $2(2^{n-1} - 1)$ 回適用し、fuzzy modus tollens を 2 回適用する。

〔証明〕 前件 " x_1 ", " $\dots$ ", " x_n " の ($n-1$) 次元について、〔定理 1〕より、 $2(2^{n-1} - 1)$ 回の fuzzy modus ponens が適用される。これにより 2 個のインプリケーションができる。これらのインプリケーションにそれぞれ fuzzy modus tollens を適用する。ここで得られた結論を合成することより最終的結論を得る。よって fuzzy modus tollens は 2 回適用される。

従って〔定理 2〕は証明される。〔証明終〕

本論文の方法による n 次元ファジー推論についてその計算効率を考える。本論文の方法では、まず図 1 の第 1 段階の fuzzy modus ponens が行われる。このとき 2 の n 乗個のプロセッサが用意され、一つのプロセッサに一つの fuzzy modus ponens が割当てられ計算が行われる。各プロセッサは与えられたインプリケーションと前提から結論を求めるが、計算にあたって他のプロセッサの計算した結論や途中経過を参照することはない。すなわち、各プロセッサは他のプロセッサと独立に計算を行う。2 の n 乗個の fuzzy modus ponens を同時に処理することにより、計算時間は fuzzy modus ponens 1 回分で計算を行えることになる。

次にインプリケーションの合成が行われる。第 1 段

階のインプリケーションの合成は2の $n-1$ 乗個のプロセッサが用意され、一つのプロセッサが一つの新しいインプリケーションを合成する。このとき各プロセッサは他のプロセッサを参照することはない。すなわち、インプリケーションの合成についても合成1回分程度の計算時間で2の $n-1$ 乗個のインプリケーションを合成することができる。

同様に第 i 段階においても、もし第 $i-1$ 段階でインプリケーションの合成が行われ、新しいインプリケーションが与えられるなら、2の $n-i+1$ 乗個のプロセッサを用意することにより、fuzzy modus ponens 1回分程度の計算時間で2の $n-i+1$ 乗個のfuzzy modus ponensを行うことができる。

以上の議論から図1の各段階のfuzzy modus ponensはfuzzy modus ponens 1回分程度の計算時間で一度に行えるので全体の推論は段階の数、すなわち次元数 n に比例した計算時間で行える。逆問題についても第 n 段階でfuzzy modus tollensが行われるだけで全く同様である。これらのことから次の定理が得られる。

〔定理3〕 本論文の方法では、 n 次元ファジー推論のために2の n 乗個のプロセッサを用い、その計算時間は n に比例した時間となる。

〔定理4〕 本論文の方法では、 n 次元ファジー推論の逆問題を解くために、2の n 乗個のプロセッサを用い、その計算時間は n に比例した時間となる。

5. 例題

本章では、多次元ファジー推論について例題を通して詳しく解説する。例題として三次元ファジー推論を考える。インプリケーションは次の八つが与えられている。このとき前提 $(x_1 \text{ is } P_1, x_2 \text{ is } P_2, x_3 \text{ is } P_3)$ について結論を求める。

$$\begin{aligned} I_{000} &: (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_{30}) \rightarrow z \text{ is } Q_{000} \\ I_{001} &: (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_{31}) \rightarrow z \text{ is } Q_{001} \\ I_{010} &: (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{21}, x_3 \text{ is } P_{30}) \rightarrow z \text{ is } Q_{010} \\ I_{011} &: (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{21}, x_3 \text{ is } P_{31}) \rightarrow z \text{ is } Q_{011} \\ I_{100} &: (x_1 \text{ is } P_{11}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_{30}) \rightarrow z \text{ is } Q_{100} \\ I_{101} &: (x_1 \text{ is } P_{11}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_{31}) \rightarrow z \text{ is } Q_{101} \\ I_{110} &: (x_1 \text{ is } P_{11}, x_2 \text{ is } P_{21}, x_3 \text{ is } P_{30}) \rightarrow z \text{ is } Q_{110} \\ I_{111} &: (x_1 \text{ is } P_{11}, x_2 \text{ is } P_{21}, x_3 \text{ is } P_{31}) \rightarrow z \text{ is } Q_{111} \end{aligned}$$

まず、インプリケーション I_{000} について次のfuzzy modus ponensを行う。

$$\begin{aligned} (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_3), (x_1 \text{ is } P_{10}, x_2 \text{ is } \\ P_{20}, x_3 \text{ is } P_{30}) \rightarrow z \text{ is } Q_{000}/z \text{ is } Q'_{000} \dots\dots (R11) \end{aligned}$$

このとき次の三次元の逆真理値限定を行う必要がある。

$$(x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_3)$$

$$\simeq (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_{30}) \text{ is } \tau_{30}$$

この τ_{30} は $(x_3 \text{ is } P_3) \simeq (x_3 \text{ is } P_{30}) \text{ is } \tau_{30}$ の τ_{30} と等価であると考えてよいから、三次元の逆真理値限定は一次元の場合に帰着でき推論 R_{11} の結論 $z \text{ is } Q'_{000}$ が求められる。同様に他のインプリケーション $I_{001} \sim I_{111}$ についても前提の“ x_1 ”, “ x_2 ”に関する命題を前件の命題に固定し, “ x_3 ”に関する一次元のfuzzy modus ponens $R_{12} \sim R_{18}$ に帰着し結論を求める。それぞれの結論を $Q'_{001}, Q'_{010}, Q'_{011}, Q'_{100}, Q'_{101}, Q'_{110}, Q'_{111}$ とする。

次に新しいインプリケーションの合成を行う。 R_{11} と R_{12} の結論からインプリケーション I_{00} を合成する。

$$I_{00} : (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_{00}$$

ここで、 $Q_{00} = (Q'_{000} + Q'_{001})/2$ として得られるファジー集合である。同様に $I_{01} \sim I_{11}$ を以下のように合成する。

$$I_{01} : (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{21}, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_{01}$$

$$I_{10} : (x_1 \text{ is } P_{11}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_{10}$$

$$I_{11} : (x_1 \text{ is } P_{11}, x_2 \text{ is } P_{21}, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_{11}$$

ここで $Q_{01} = (Q'_{010} + Q'_{011})/2$, $Q_{10} = (Q'_{100} + Q'_{101})/2$, $Q_{11} = (Q'_{110} + Q'_{111})/2$ である。

更にこの新しいインプリケーション I_{00} を用いてfuzzy modus ponens R_{21} を行う。

$$\begin{aligned} (x_1 \text{ is } P_{10}, x_2 \text{ is } P_2, x_3 \text{ is } P_3), (x_1 \text{ is } P_{10}, x_2 \text{ is } \\ P_{20}, x_3 \text{ is } P_{30}) \rightarrow z \text{ is } Q_{00}/z \text{ is } Q'_{00} \dots\dots (R21) \end{aligned}$$

同様にインプリケーション $I_{01} \sim I_{11}$ についてもfuzzy modus ponens $R_{22} \sim R_{24}$ を行い、それぞれの結論 $Q'_{01}, Q'_{10}, Q'_{11}$ を求める。推論の方法は R_{11} で述べた方法と同様である。例えば R_{21} について $(x_1 \text{ is } P_{10}, x_2 \text{ is } P_2, x_3 \text{ is } P_3) \simeq (x_1 \text{ is } P_{10}, x_2 \text{ is } P_{20}, x_3 \text{ is } P_{30}) \text{ is } \tau_{20}$ と $(x_2 \text{ is } P_2) \simeq (x_2 \text{ is } P_{20}) \text{ is } \tau_{20}$ の τ_{20} が等価であると考え、一次元のfuzzy modus ponensに帰着し推論を行う。 $R_{21} \sim R_{24}$ の結論から I_0 と I_1 を合成する。

$$I_0 : (x_1 \text{ is } P_{10}, x_2 \text{ is } P_2, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_0$$

$$I_1 : (x_1 \text{ is } P_{11}, x_2 \text{ is } P_2, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_1$$

ここで

$$Q_0 = (Q'_{00} + Q'_{01})/2$$

$$Q_1 = (Q'_{10} + Q'_{11})/2$$

である。

I_0, I_1 を用いてfuzzy modus ponens R_{31}, R_{32} を行う。

$$(x_1 \text{ is } P_1, x_2 \text{ is } P_2, x_3 \text{ is } P_3), (x_1 \text{ is } P_{10}, x_2 \text{ is } P_2,$$

$$x_3 \text{ is } P_3 \rightarrow z \text{ is } Q_0/z \text{ is } Q'_0 \dots\dots\dots (R31)$$

$$(x_1 \text{ is } P_1, x_2 \text{ is } P_2, x_3 \text{ is } P_3), (x_1 \text{ is } P_{11}, x_2 \text{ is } P_2, x_3 \text{ is } P_3) \rightarrow z \text{ is } Q_1/z \text{ is } Q'_1 \dots\dots\dots (R32)$$

最後に R31, R32 の結論 Q'_0, Q'_1 から Q を得る。

$$Q = (Q'_0 + Q'_1)/2$$

この Q を最終的結論とする。すなわち最終的結論は “ $z \text{ is } Q$ ” となる。以上の例題で示したように従来の三次元ファジー推論では $2(2^3-1)=14$ 回の fuzzy modus ponens が逐次行われる。

これに対して本論文の方法では、まず R11~R18 の fuzzy modus ponens が同時に処理される。次に R21~R24 の fuzzy modus ponens が同時に処理される。最後に R31, R32 の fuzzy modus ponens が同時に処理される。そしてこの R31, R32 の結論を合成して最終的結論を得る (図 3 参照)。このように本論文の方法では、3 段階の fuzzy modus ponens で最終的結論を得ることができる。

6. 実験結果

本章では、本論文で提案した並列多次元ファジー推論の有効性を VAX 11/785 上で VAX-Ada によって行った実験の結果によって示す。

実験には三次元および五次元ファジー推論を用いた。三次元ファジー推論の場合、第 5 章の例題において $P_{10}=P_{20}=P_{30}=\text{TRUE}$, $P_{11}=P_{21}=P_{31}=\text{FALSE}$, $Q_{000}=\text{V. T.}$, $Q_{001}=Q_{010}=Q_{101}=\text{TRUE}$, $Q_{100}=Q_{101}=Q_{110}=\text{FALSE}$, $Q_{111}=\text{V. F.}$ とした 8 個のインプリケーションを用いて推論を行った。

ここで TRUE, FALSE, V. T. (Very True), V. F. (Very False) は言語的真理値でありファジー集合である。更に、インプリケーション自身の言語的真理値

はすべて TRUE である。与えられた前提は “($x_1 \text{ is V. T.}$, $x_2 \text{ is V. T.}$, $x_3 \text{ is V. T.}$)” である。五次元ファジー推論の場合も同様に、前件および結論の各命題に

図 3 並列三次元ファジー推論の流れ

Fig. 3. Process of parallel 3-dimensional fuzzy reasoning.

表 2 各プロセッサの計算時間
(三次元ファジー推論)

Table 2. CPU time of each processor in parallel 3-dimensional fuzzy reasoning

		(単位 1/100 秒)							
段階	プロセッサ	1	2	3	4	5	6	7	8
fuzzy modus ponens 1		21	21	17	21	21	21	21	21
合成 1		6	5	7	8	—	—	—	—
fuzzy modus ponens 2		18	18	20	18	—	—	—	—
合成 2		8	7	—	—	—	—	—	—
fuzzy modus ponens 3		19	20	—	—	—	—	—	—
最終的結論		7	—	—	—	—	—	—	—
合計時間		79	71	44	47	21	21	21	21

表 3 各プロセッサの計算時間 (五次元ファジー推論)

Table 3. CPU time of each processor in parallel 5-dimensional fuzzy reasoning

		(単位 1/100 秒)																															
段階	プロセッサ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
fuzzy modus ponens 1		20	19	18	19	21	19	20	20	22	19	20	19	21	19	19	20	20	21	20	20	18	21	21	21	20	21	23	20	20	21	21	19
合成 1		7	6	8	6	6	7	7	7	7	9	9	7	8	9	7	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
fuzzy modus ponens 2		20	19	21	20	19	21	19	20	20	21	20	22	19	18	18	20	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
合成 2		7	7	7	8	7	8	7	6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
fuzzy modus ponens 3		16	20	18	22	17	19	21	16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
合成 3		8	7	6	7	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
fuzzy modus ponens 4		21	18	20	18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
合成 4		6	8	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
fuzzy modus ponens 5		19	19	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
最終的結論		6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
合計時間		130	123	98	100	70	74	74	69	49	47	49	50	47	45	46	47	20	21	20	20	18	21	21	21	20	21	23	20	20	21	21	19

表4 Sugeno と Takagi の多次元ファジー推論の計算時間

Table 4. CPU time of 3-dimensional fuzzy reasoning and 5-dimensional fuzzy reasoning by using Sugeno and Takagi's method.

段 階	三 次 元	五 次 元
fuzzy modus ponens 1	158	602
合成 1	25	103
fuzzy modus ponens 2	74	294
合成 2	13	51
fuzzy modus ponens 3	40	148
合成 3	—	27
fuzzy modus ponens 4	—	76
合成 4	—	13
fuzzy modus ponens 5	—	39
最終的結論	6	6
合 計 時 間	316	1,359

ファジー集合 (TRUE, FALSE など) を用いたインプリケーションを与え、推論を行った。

三次元および五次元ファジー推論の結果をそれぞれ表2、表3に示す。これらの表は、各プロセッサが1回の fuzzy modus ponens および一つのインプリケーションの合成に費やした時間、すなわち各プロセッサが必要なデータを受取ってから結論を返すまでのCPU時間を示している。

比較のため、Sugeno と Takagi の多次元ファジー推論を実行し、最終的結論を得るまでにかかった時間を表4に示す。VAX 11/785 は単一プロセッサの逐次計算機であるため厳密な議論は困難であるが、これらの結果から本論文で提案した方法の有効性が確かめられた。

7. 終 り に

本論文では、多次元ファジー推論の高速化の一手法として並列多次元ファジー推論の方法を提案した。多次元のインプリケーションはエキスパート・システム

を効率良く動作させるのに欠かせない知識である。しかし、多次元のインプリケーションをうまく扱う方法は、今までにほとんど発表されていなかった。本論文では、fuzzy modus ponens およびそれらの合成を並列に処理することにより一般に計算量の多い多次元ファジー推論が高速に (前件の命題の組数に比例して) 行えることを示した。

一般に多次元のインプリケーションは五次元以下のものがほとんどであり、実際に非常に高次元なインプリケーションを知識として得ることが困難であることを考え合えると、プロセッサは32個程度あれば十分であるから、提案した方法は商用の並列計算機上に容易にインプリメントでき、実用的であると考えられる。本研究の一部は文部省科学研究費海外学術研究62045007、奨励研究(A)63750393および電気通信普及財団の援助により行われた。

(昭和63年6月24日受付)

文 献

- (1) 笠原: 「マルチプロセッサシステムの研究動向」, 電学論C, 108, 96 (昭63-2)
- (2) 三浦, 他: 「自動化推進のためのエキスパートシステム開発例」, 同上C, 108, 238 (昭63-4)
- (3) 水本: 「Fuzzy 推論法」, システムと制御, 28, 436 (昭59-7)
- (4) E. Sanchez & L. A. Zadeh: "Approximate Reasoning in Intelligent Systems, Decision and Control", (1987) Pergamon Press
- (5) M. Sugeno & T. Takagi: "Multi-dimensional Fuzzy Reasoning", *Fuzzy sets and Systems*, 9, 313 (1983)
- (6) E. Tazaki, Y. Hayashi, K. Yoshida & A. Koiwa: "Development of Automated Health Testing and Services System via Fuzzy Reasoning", *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, p. 342 (1986)
- (7) Y. Tsukamoto: "An Approach to Fuzzy Reasoning Method", *Advances in Fuzzy Set Theory and Applications* (ed. M. M. Gupta et al.), North-Holland (1979)
- (8) United States Department of Defense: "Ada Programming Language Reference Manual", United States Department of Defense, Washington, D. C. (1983)
- (9) 安信: 「予見 Fuzzy 制御方式によるコンテナクレーン自動運転」, 計測自動制御学論, 22, 1066 (昭61-10)