設 計 自 動 化　51－6
（1990．2．20）

論理式を分離加法形式で表現する一手法

松田秀雄　　宮腰　隆

富山大学　工学部

多値入力二値出力関数を分離加法形で表す方法を提案し、二つの木形アルゴリズムと比較検討している。本方法は精度が良い（加法形式の項数が少ないこと）が 一般には計算時間がかかる。しかし、項の数の小さい関数では、 分離加法形が早く求まり、変数の数の影響をあまり受けない。この性質により、本方法を木形アルゴリズムと結びつけて、木形アルゴリズムの計算時間をそれ程落とさずに、精度を上げれる、特長ある使い方ができることも示す。

A Method to Derive the Disjoint Disjunctive Form of Logical Expressions

Hideo Matsuda　　Takashi Miyagoshi

Faculity of Enginerring,Toyama University

Gofuku,Toyama-shi,930 Japan

A method to obtain disjoint disjunctive form for a binary function with p-valued input is proposed. It is compared with two tree type algorithms. Our method has a high accuracy that produces as few product terms as posible, whereas requires an excessive computing time in general. However ,if it is applied to the functions which expressed in a few hundred product terms, the result is derived very quickly, and is not so dependent on the number of inut variables as the tree type algorithms. Therefore, it is showed that our method being incorporated into the tree type algorithm becomes very efficient technique in which the computing time is approached it of the tree type algorithm and the accuracy is held in the same degree as that of original our method.

1．はじめに

論理式を分離加法形式で表現すると、一般の加法形式より項の数が増加する傾向にあるが、次のような利点があって、重要な表現形式である[1]。(1) 分離加法形では項が互いに共通部分を持たないので、それ自体、与えられた論理式の一種の簡単化になっており、より厳密な簡単化を行うさいの初期解にできる。(2) 論理式に含まれる最小項の数が算出できるので、トートロジィの判定や，論理式同志の包含関係の判定，あるいは、等しいかどうかの検証に使え、(3) 更に、任意に発生させた関数の真理値表濃度が計算できるので、統計的性質をみるのに好都合である。

この場合も、論理式の簡単化同様、多変数の関数まで扱える、高速で精度の良い（加法形式で項の数が小さいこと）手法の開発が望まれる。

分離加法形式はシャープ演算やディスジョイント・シャープ演算（以下、㊋ 演算と略記）[1] を用いても導出できるが、木形アルゴリズムの方が効率的である。このさい、分岐の仕方により二つに分類できる。一つはＣＨＡＮの方法（決定木 decision tree による方法なのでＤＴ法と略）[2] で、分岐点で、ある入力変数 X_iが選ばれ，X_iがp_i値をとり得るならば、高々p_i 個の部分木が生ずる（高橋らや ＥＳＰＲＥ-ＳＳＯで使われている二分法[1] は $p_i=2$ の特別な場合で、この方法に含まれる）。

他の一つは、分岐点で、一つの項が選ばれ、その項の否定を ㊋ 演算で求めた結果生ずる高々 ｎ－１個（ｎは変数の数）の項により、それぞれ、部分木を生成していく方法で、笹尾の否定を求める手法[3] がこれにあたる（以下、ＳＡＳ法と呼ぶ）。

ここに提案する手法は、部分マップ（ＳＵＢＭＡＰ）法の考え[4] に基づいているので、略してＳＵＢ法と呼ぼう。すなわち、今回、部分マップ法を関数や部分マップ、それに許容キューブといったこの方法の主要な項目をキューブで表現することにより、多値入力、多変数の関数まで扱えるよう計っている。

ＳＵＢ法は基本的には分岐しないで、直接、関数の分離加法形を求めるので、木形アルゴリズムに比べ、非常に精度が良いが、その反面、計算時間がかかり過ぎるきらいがある。

しかし、これは、与えられた関数の項の数が数千個と多いからであって、数百個以下に限定するなら、計算時間は変数の数にそれ程影響されず、多変数で、むしろＤＴ法より早く分離加法形が求まる。

この特徴に注目すると、ＤＴ法、ＳＡＳ法と本方法とを組合せて、これら双方の計算速度をそれ程落さずに、精度を高める手法が考えられ、それぞれ、ＤＴ－ＳＵＢ法、ＳＡＳ－ＳＵＢ法と呼ぼう。

２章では基礎的事項を、３章ではＤＴ法、ＳＡＳ法、ＳＵＢ法について記述する。４章では、二値入力関数についての実験結果から、これら三つの手法の特質を明かにする。続いて，ＤＴ－ＳＵＢ法，ＳＡＳ－ＳＵＢ法について触れ、四値入力多変数の関数のいくつかの実験例で、ＤＴ－ＳＵＢ法が優れていることを示す。ここでは、分離加法形から計算した、関数の真理値表濃度も付記してある。

2．基礎的事項

ｎ変数多値入力二値関数ｆは$P_1\times P_2\times\cdots\times P_n$からＢ＝{0,1} への写像で、但し、$P_i=\{0,1,\cdots,p_i-1\}$、次のように加法形式で表現できる。

$$f=t_1\vee t_2\vee\cdots\vee t_m \qquad (1)$$

ここで、項t_j (j=1,2,･･･,m) は入力変数X_iの論理積で、ｆにより定まるS_i ($\subseteq P_i$) により、

$$t_j=X_1^{s1}X_2^{s2}\cdots X_n^{sn}$$

と表わされ、リテラルと呼ばれるX_i^{si}は、$X_i\notin S_i$ のとき、$X_i^{si}=0$、$X_i\in S_i$のとき、$X_i^{si}=1$である。

二つの項ｔ，ｔ'の間にｔ≦ｔ'が成り立てば、ｔはｔ' に含まれるという。任意の項ｔがｔ≦ｆのとき、ｆの内項、ｔを含む内項が他になければｔを主項という。二つの項 t_j, t_k が $t_j t_k=0$ のとき、分離しているといい、式(1)のすべての項が互いに分離している場合を分離加法形式という。特に項数の少ない分離加法形式の関数を求めることが分離加法形式の関数の簡単化といわれ、本稿の主題である。

項 t_iはキューブともいわれ、次のようにビット表現もできる。

$$t_i=b_1^0b_1^1\cdots b_1^{p1-1}-b_2^0b_2^1\cdots b_2^{p2-1}-\cdots\cdots-b_n^0b_n^1\cdots b_n^{pn-1}$$

ここで、－で区切られた各部分は左よりX_1^{s1}、X_2^{s2}、･･･、X_n^{sn}の各リテラルに対応し、$k\in P_i$ なるｋが、$k\in S_i$なら、$b_i^k=1$、$k\notin S_i$なら、$b_i^k=0$ とする。

例えば、２変数四値入力二値出力関数、ｎ＝2、$P_i=\{0,1,2,3,\}$,

$$f_1=X_1^{(0,1,3,)}X_2^{(0,1)}\vee X_1^{(1,2)}X_2^{(1,2)}\vee X_1^{(0,3)}X_2^{(2,3)}$$

の各項は、

t_1=1101-1100, t_2=0110-0110, t_3=1001-0011

と、それぞれ表される。

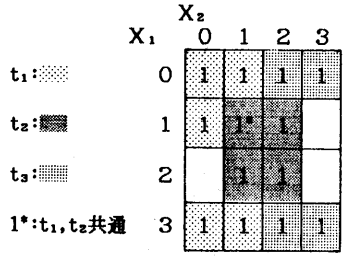

図1　関数f_1

関数は変数の小さい場合はマップで表した方が分かりやすく、本例では図1となる。但し、図1ではf_1が1及び0の値をとる入力の組合せのセル（四角います目）を、それぞれ、1及び無印で表している。

(1) **基本キューブ**　あるキューブが、もし、すべての部分が一つだけ1、他は全部0の桁よりなれば、マップ上の一つのセルを表わし、通常、最小項と呼ばれるが、ここでは基本キューブともいう（例、1000-0100）。各基本キューブ（あるいは、セル、又は最小項）はそれら表わすビット表現による2進数の大きさで自然に順序づけられる。特に一番小さなセルは、図1のf_1の例では　1000-1000、一番大きなセルは　0001-0001　である（但し、左側より右側に桁が上がるとして）。一般に考えて、一番大きなセル 00･･･01-00･･･01-････-00･･･01　を　C_M　で示そう。

(2) **キューブの最小（最大）基本キューブ**　キューブC_iにおいて、各部分の最左端（最右端）の1のみ残し、他の桁のビットをすべた0と置いてできる基本キューブ$C_{i,m1}$　(C_{i,m_0}) はC_iに含まれる基本キューブのうち、自然の順序で比較して、最小（最大）のものである。例えば、上の　t_1=1101-1100　をC_iとして、$C_{i,m1}$=1000-1000、C_{i,m_0}=0001-0100　である。

(3) **共通キューブ**　二つの基本キューブ C_iとC_jの各ビットごとに論理和をとってできるキューブをC_iとC_jの共通キューブといって、CO（C_i、C_j）で表そう。

(4) **共通キューブの拡大**　共通キューブにおいて、ある部分に1が二つあって、かつ、その間にいくつかの0があれば、これらの0を順次1変えていくと、キューブは、順次前のものを含むより大きなキューブとなる。この操作を共通キューブの拡大と名づける。

例えば、基本キューブ C_1=1000-0100、C_j=0001-0001　の共通キューブは　$1\dot{0}\dot{0}1$-$0\dot{1}0\dot{1}$　である。・印した0を順次左より1に変えていくと、1101-0101、1111-0101、1111-0111　と拡大される。

(5) **部分マップ**　基本キューブ C_iの各部分において1である桁の左に0である桁があれば、これらを全部1と置き換えてできるキューブは、C_iと等しいかC_iより小さい2進数のセルからなるマップ上の特定の部分を形成するので、部分マップC_iと呼びS（C_i）で示そう。例えば、基本キューブ C_i=0010-0010 の部分マップはS（C_i）=1110-1110　である。

(6) **許容キューブ**　部分マップC_iの範囲で考える。基本キューブ C_iとC_jの共通キューブを最大に拡大したキューブ　をその部分マップの許容キューブといって、P（C_i、C_j）で表わす。二値の場合、CO（C_i、C_j）＝P（C_i、C_j）である。

(7) **基本キューブの間の距離**　二つの基本キューブC_i、C_jの各部分を考えたとき、互いに1が一つしかないので、どちらかを桁移動して、1の位置を合わせることができる。この桁移動の全部分の総和をC_i、C_jの距離と定義しよう。

以上は、本稿独自の用語であるが、以下に記述するものは論理設計の文献ではよく使われている。

(8) **真理値表濃度**　マップの全セル数 $\prod_{i=1}^{n}|P_i|$ と関数値1をとるセルの総数$|f^{-1}(1)|$　の比率を真理値表濃度dという。

(9) **ディスジョイント・シャープ演算（ ⊕ 演算）**[5]　二つのキューブをC_i、C_jとしたとき、C_i⊕C_jとは、次の操作により、C_1'、C_2'、･･･、C_n'のキューブを生成することである（但し、すべて、0のみからなる部分をもつキューブは捨てる。すなわち、キューブC_k' (k=1,2,･･･,n) のm番目の部分は、

m＜kなら、C_iとC_jのm番目の部分の2進数をビットごとに論理積をとったものとし、

m＝kなら、C_iのm番目の部分と、C_jのm番目の部分の2進数の（桁ごとに）否定をとったものとをビットごとに論理積したものとし、

m＞kなら、C_iのm番目の部分のビットをそのまま、C_k'の m番目の部分とする。

又、二つの関数

$f=C_1 \vee C_2 \vee \cdots \vee C_m$

$g=G_1 \vee G_2 \vee \cdots \vee G_p$

としたとき、

$f \oplus G_j=(C_1 \oplus G_j) \vee (C_2 \oplus G_j) \vee \cdots \vee (C_m \oplus G_j)$

$f \oplus g=((\cdots((f \oplus G_1) \oplus G_2)\cdots) \oplus G_{p-1}) \oplus G_p$

である。

(10) **併合**　二つのキューブ C_iとC_jが、一つの部分kでのみ異なり、他のn－1の部分が等しければ、C_iとC_jのビットごとの論理和をとってできる、一つのキューブC_kに併合できる。

$f=C_1 \vee C_2 \vee \cdots \vee C_m$ が与えられたとき、任意の二つのキューブC_iとC_jを順次選んで、併合できれば、併合して、段々、キューブの個数を減らしていくことを f の併合という。本稿4章では、分離加法形を求めるに先立ち、関数に併合を行う場合があるが、そのときにはＭＩＮＩ[5]の最小化プロセスで使われた併合の方法によるものとする。

3. 方　法

初めに、述べる二つの手法は決定木の構造を有するアルゴリズムで、関数 f （又はその否定 $\bar{f}$ ）が分離加法形式で求めることができる。

3.1　変数により分岐する手法(ＤＴ法)

Ａ．Ｈ．ＣＨＡＮによる方法 [2] で、関数 f の決定木は次のように再帰的に定義される。木は根付きで、

1) $f=1$ (あるいは0)なら、値１(0)の唯一つの葉からなる。
2) f が１変数Ｘのみの関数なら、f を表すリテラル X_i をその値とする唯一つの葉からなる木で、
3) f が２変数以上の関数なら、ヒューリスティックな方法によって、変数の一つ X_k を選んで根に割り当て、この根より、$X_k^{(j)}$ $(j=0,1,\cdots,p_k-1)$ ごとに枝を出し、その先に $f(X_k=j)$ の部分木の根をつなぐ。但し、

$$f(X_k=j)=f(X_1,\cdots,X_{k-1},j,X_{k+1},\cdots,X_n)$$

なお、このとき使われるヒューリスティックは、単項 $X_k^{S_k}$ $(S_k \subset P_k)$ があれば、X_k が優先的に採用され又、このような単項がなければ、リテラルの数の少ない項に含まれる変数 X_k が先に選ばれるなど、木の枝出しが最小になるよう工夫がなされている。

［例題3.1］　３値入力２値関数

$$f=X_1^0X_3^1 \vee X_2^{(1,2)}X_3^1 \vee X_1^1X_3^2 \vee X_1^{(1,2)}X_2^{(0,1)}X_3^2 \vee X_1^1X_2^0X_3^{(0,2)} \vee X_1^0X_2^{(1,2)}X_3^{(0,1)}$$

の場合、最も少ないリテラルの項に表われ、かつ、一番多くの項に表われる変数 X_3 がまず選ばれる。この根の $X_3=0$ の枝には、$f(X_3=0)=X_1^1X_2^0 \vee X_1^0X_2^{(1,2)}$ の木がつながる。$f(X_3=0)$ の木の根には X_1 が選ばれ $X_1=0,1,2$ の各枝には $f(X_3=0,X_1=0)=X_2^{(1,2)}$, $f(X_3=0,X_1=1)=X_2^0$、$f(X_3=0,X_1=2)=0$ の木、つまり、各葉がつながって、枝出しは終わる。同様に、X_3 の根から、$X_3=1,2$ についての枝出しが行われて図２のように決定木ができる。

(0でない)各葉から、根までの道の枝のリテラルをかけ合わせてできる項の和がfの分離加法形を形成する。

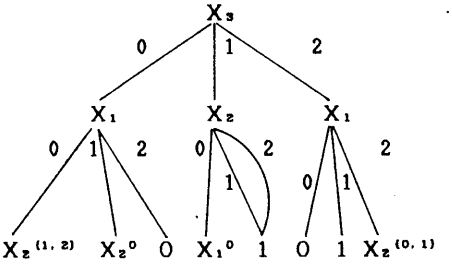

図2　決定木

$$f=X_2^{(1,2)}X_1^0X_3^0 \vee X_2^0X_1^1X_3^0 \vee X_1^0X_2^0X_3^1 \vee X_2^{(1,2)}X_3^1 \vee X_1^1X_3^2 \vee X_2^{(0,1)}X_1^2X_3^2 \quad (2)$$

ここで、もし、葉の値を否定(1→0,0→1, $X^{R1} \to X^{P1-R1}$)にすると、f の決定木は $\bar{f}$ の決定木に変わり、上述ように根まで逆のぼって、各項を作れば $\bar{f}$ が分離加法形で得られる。

なお、一つの根から出ている枝が同じ値の葉をもてば、一つの葉にまとめるので、図2のように多重枝も生ずる。我々は、ここのところを、同じ部分木をもてば、一つの部分木にまとめるようプログラム化した。これは、葉から根へと向う積項の合成過程で、併合操作を行っていることに相当する。

3.2　項により分岐する手法（ＳＡＳ法）

笹尾は関数 f の否定を高速に導出する手法を文献[3]で提案した。それを f の分離加法形を求める手法に応用するとしたら木の構成（概要）は次の通りとなろう。

1)　上記と同じ。
2)　f が唯一つの項 t_i よりなれば、t_i を値とする葉をつくる。
3)　f が二つ以上の項よりなれば、ヒューリスティックな手法により一つの項 t を選んで根に配し、これより、t_1' , t_2' ,$\cdots$, t_n' を付した枝を出して、その先に、それぞれ $f(|t_j')$ $(j=1,2,\cdots,n)$ の木の根をつなぐ。但し、t_1' , t_2' ,$\cdots$, t_n' は C_u ⓐ t の結果得られる各項（C_u はuniversal cube)で、
$f(|t_j')=t_j' \cdot f$　は t_j' と f の各項とのビットごとの論理積をとってできる関数（これを f の項 t_j' への拘束といおう）とする。

木が構成されると、各葉から木の元の根までさかのぼる道の、葉の値および枝に付した項の積が分離加法形の各一項をなし、これらに、部分木の根に配した項の和を更に加えると、f の分離加法形が形成される。

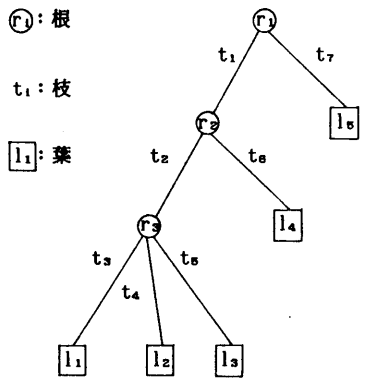

図3　SAS法の例

ここで、3)のヒューリスティックとしては、リテラル数最小の項を選ぶ。又、C_n⊛tを求めるときも、ディスジョイント・シャープ演算をとる変数の順序を変えるなど、決定木が大きくならないよう種々の工夫がなされている。

［例題3.2］［例題3.1］の決定木を本方法でつくると、図3のようになる。但し、r_iは木の根（節）で、

r_1=111-011-010, r_2=010-111-001, r_3=001-110-001

各枝t_iは（i=1,2,･･･は枝出しを行った順）

t_1=111-111-101, t_2=101-111-111, t_3=110-111-111

t_4=001-001-111, t_5=001-110-110, t_6=010-111-110

t_7=111-100-010

各葉l_iは

l_1=100-011-100, l_2=0, l_3=0, l_4=010-100-100

l_5=100-100-010

この例の分離加法形は0でない葉から根r_1までの道をたどることにより得られる。

$l_1 \cdot t_3 \cdot t_2 \cdot t_1$=100-011-100

$l_4 \cdot t_6 \cdot t_1$ =010-100-100

$l_5 \cdot t_7$ =100-100-010

の三つの項と、根r_1, r_2, r_3 の三つの項の和からなる。

なお、本方法では上記の手法のように、同じ値の葉あるいは、同じ部分木を一つにまとめるような併合操作は行わないものとする。このことが後述するように二つの方法の結果に著しい差異を生じる。

3.3 我々の手法（SUB法）

ここに提案する手法は分岐しないで、直接与えられた関数の分離加法形を求めるので、前二つのように否定は求まらない。

方法　所望の結果はFに求まるものとし、はじめに

F=φ（空集合）にしておく。

1) 関数fをキューブの集合$\{C_1, C_2, \cdots, C_r\}$で与える。fを併合する。簡単のため、結果のキューブも$\{C_1, C_2, \cdots, C_r\}$としておこう。

2) 各キューブC_i(i=1,2,･･･,r)の最小基本キューブC_{i,m_i}、及び、最大基本キューブC_{i,m_a}を求める。

3) 最大基本キューブの集合$\{C_{i,m_a}\}$のうちC_Mとの距離が最大のキューブをC_{max}とする。$S(C_{max})$を生成。

4) 最小基本キューブの集合$\{C_{i,m_i}\}$のうち、$S(C_{max})$に含まれる基本キューブ（以下の5,6のステップではこのような基本キューブについてのみ考える）の中から、C_{max}と最も距離の大きなものを一つ選ぶ（同一距離のものが複数個あれば自然の順序で）。これを、C_{min}とする。

5) C_{min}とC_{max}の共通キューブ$C_0 = CO(C_{min}, C_{max})$を生成する。

6) C_0がfの内項かどうか調べる。内項でなければ、C_{max}との距離が次に大きな基本キューブをC_{min}として5)へ。内項なら、

7) C_0をできるだけ大きな内項になるよう、拡大する。拡大したキューブをC_0として、

8) C_0をFに加える。f⊛C_0を新たにfとする。$f \neq \phi$なら 1)へ。$f = \phi$なら、STOP。

［例題3.3］［例題3.1］の関数fの分離加法形を本方法で求める。式(2)の第一項、第二項、･･･、第五項をそれぞれキューブC_1、C_2、･･･、C_5とすると、

$$\{C_i\} = \begin{Bmatrix} 100\text{-}111\text{-}010 \\ 111\text{-}011\text{-}010 \\ 010\text{-}111\text{-}001 \\ 011\text{-}110\text{-}001 \\ 010\text{-}100\text{-}101 \\ 100\text{-}011\text{-}110 \end{Bmatrix} \quad \text{（但し、上より } i=1,2,\cdots,5)$$

マップで表わすと、図4(a)のようになる。本例では併合しても変わらない。但し、C_1に含まれるセルを1で，C_2に含まれるセルを2で･･･というように表現している。又、i*でC_iの最小基本キューブを、i*で最大基本キューブをそれぞれ示す。したがって、3)のC_{max}は 3* のセルで、$C_{max}=C_{3,m_a}$=010-001-001、$S(C_{max})$=110-111-111で、図4(b)の網掛けした部分となる。

今の場合、最小基本キューブの集合$\{C_{i,m_i}\}$が全部$S(C_{max})$に含まれるので、4)のC_{min}として、5*のセル がきまる。C_{min}=010-100-100。

5)の共通キューブC_0はC_0=010-101-101、これは図

4(c)の網掛けのセルからなり、fに含まれないセルを含むので、内項ではない。続いて、C_{max}との距離の大きさの順でC_{min}として、基本キューブを 6*、1*、2*と選んで、5)、6) を繰り返すも、これらとC_{max}との共通キューブC_0は内項ではない。そして、C_{min}として 3* (= 4*)=010-100-001を選ぶと、共通キューブC_0=010-101-001 は内項となり、7) で C_0はC_3まで拡大できる。

8)で、C_0=C_3=010-111-001 をF加える。f ⑧ C_0の結果、C_1,C_2,C_6は前のまま変わらず、C_3は取り除かれ、C_4とC_5はそれぞれ C_4' =001-110-001 と C_5'=010-100-100とに変わる。これらのキューブの集合を新たにfとして、再び、1)に返り同様の手順を繰り返していくと、あと五つのキューブ 001-110-001, 111-011-010, 100-011-100, 100-100-010, 010-100-100が順次Fに求まってきて、本例題の関数の分離加法形を形成する。

なお、ステップ1)の併合は、他の方法の前処理的に行う併合に合わせて入れたもので、省略してもよい。

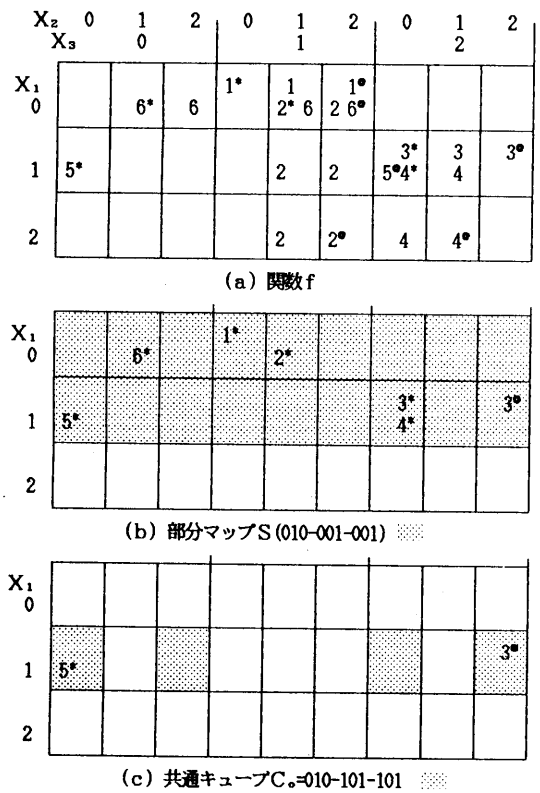

図4 SUB法の説明図

4. 結 果

我々はCHANの決定木による方法（DT法）、笹尾の方法（SAS法），本方法（SUB法）を、FORTRAN言語でプログラム化し，IBM 3081-KX4（富山大学情報処理センター）で計算を行なった。

表1は二値入力二値出力関数の計算例である。変数の数nが小さい場合，すなわち、nが9から12までは、各nごと、真理値表濃度が、0.2,0.5,0.8 になるよう乱数を使って最小項を発生させた、各々につき10個ずつ計30個の関数の平均値につき示したものである。

SAS法では否定を求めるさい、最小項で与えたキューブをまず併合してから、否定を求めている。それにならって、ここでも、最小項をまず併合してから、それぞれの手法を適用している。結果として得られる分離加法形の項数は、併合後の項数に対する減少の割合（百分率 、負の場合は増加）で示している。

SAS法では併合後の項数よりかえって増加し，DT法ではほとんど改善されず、SUB法のみが項数の減少が見られる。最小項の形で与えた関数は併合しても分離加法性を保っている。それ故、いまのような場合、SAS法は適用しない方がよい。併合してせっかく大きなキューブにしても、DT法もSAS法も分岐するたびに、マップの領域を細分し、細分した領域に限定したキューブのみを扱うので、葉に到達したときには、細かなキューブとなる。しかし、DT法では合成過程で併合作用があるので、ほぼ元のキューブの大きさまで回復するのに対し、SAS法では併合作用がないので、著しい項数の増加につながると考えられる。

表1の下半分は12変数で、d=0.5のときの最小項の項数2048をそのままに固定して、変数の数nを24, 48,96と増加させて、計算時間を調べてみたものである。併合操作のないSAS法が 最も早く、SUB法はDT法と比べても相当遅いことがわかる。更に、SAS法の計算時間はほとんど初めの併合のためのもので、この併合を取り去ると、96変数の計算時間が6.2秒になってしまう。しかし、項数は増加し、nが9から12で、減少率は－70%にも達する。

DT法では、初めの併合を取り去ると、表1の計算時間より1／2ないし1／3減少するが、項数は変わらない。SUB法では、変数の数の小さい場合（n＝9から12）、計算時間はかえって3倍ぐらいに増加するが、項数の減少率は11%と改善される。

さて、DT法とSUB法を比較すると精度（分離加法形の項数）ではSUB法がよく、計算時間ではDT法がよい。しかし、計算時間については、表1よりも

表１　二値入力二値出力関数について

変数の数n	併合後の項数	項数の平均減少率%			平均計算時間(CPU-TIME:S)		
		ＳＡＳ法	ＤＴ法	ＳＵＢ法	ＳＡＳ法	ＤＴ法	ＳＵＢ法
9	102.87	-10.82	0.29	5.57	0.8	0.8	4.7
10	201.87	-13.42	0.07	7.10	3.1	3.3	23.1
11	392.17	-16.79	0.25	6.88	12.8	13.1	123.6
12	768.57	-19.01	0.24	7.22	53.4	54.5	734.1
12	769	-15.47	0.39	4.68	45.6	46.3	567.3
24	2046	0.0	0.0	0.0	373.0	450.7	10125.3
48	2048	0.0	0.0	0.0	789.9	1196.0	13865.3
96	2048	0.0	0.0	0.0	1706.6	3628.6	20287.8

n=9～12については真理値表濃度 0.2, 0,5, 0.8 各10個ずつ、計30個の関数の平均値
n=12～96については各１個だけの関数の計算値

っと項数の少ない50ないし100個程度の関数でみると、１２変数までは数倍程しか差がなく、２４変数を越えると逆にＳＵＢ法が早くなり、例えば、９６変数の関数ではＳＵＢ法がＤＴ法より４倍早くなる。

ＤＴ法とＳＵＢ法のこれらの特長を生かし、もし、ＤＴ法で部分木に分解し、項の数がしきい値 JMP (20ないし100程度) 以下になると、ＳＵＢ法を適用して、分離加法形を求め、元の関数に復帰するようにすれば、計算時間はＤＴ法に近く、精度はＳＵＢ法に近い方法が得られると考えられる。すなわち、ＤＴ－ＳＵＢ法とはＤＴ法の決定木を作るステップ１),２)を関数ｆの項の数がJMP以下なら次の　1')　に改めたものである。

1') ＳＵＢ法を適用して、分離加法形を求め、それを値とする葉を作る。

ついでに、ＳＡＳ法でも上記の条件のときスッテプ１),２)の代わりにこの　1')　を、実行するようにすると、ＳＡＳ－ＳＵＢ法が、できる。

表２に、表１と同じ関数について行なったこれらの方法の計算結果を示す。但し、初めに併合がある場合で、ＤＴ－ＳＡＳ法では項数の減少率はＳＵＢ法の値に近く、計算時間はＤＴ法のそれに近くなっている。併合を取り除くと、二つの方法とも、項数は多少増加するが、計算時間は1/4ないし1/10に減少する。JMPの値を小さくすれば、ＤＴ法あるいはＳＡＳ法に近い特性となり、大きくすれば，ＳＵＢ法の特性に近づく。

次に、四値入力二値出力関数についての実験結果を述べる。変数の数ｎが 4,5,6と小さい場合は、二値入力のとき同様、最小項を乱数で生成した関数の平均的特性でみると、ＳＡＳ法，ＤＴ法，ＳＵＢ法の相互間の、項の数の減少率、平均計算時間についての関係は表１(ｎ＝９から１２)の場合と、ほぼ同じ傾向がある

表２　決定木の方法とＳＵＢ法と組み合わせた方法

変数の数n	項数の平均減少率%		平均計算時間 [S]	
	ＤＴ－ＳＵＢ法	ＳＡＳ－ＳＵＢ法	ＤＴ－ＳＵＢ法	ＳＡＳ－ＳＵＢ法
9	4.80	- 2.11	6.4	5.7
10	5.75	- 3.40	15.9	12.3
11	3.99	- 7.45	41.0	33.0
12	4.06	- 8.82	114.7	97.2
12	3.77	- 6.50	90.6	111.9
24	0.0	0.0	614.0	650.5
48	0.0	0.0	1425.7	1370.6
96	0.0	0.0	3522.9	2933.0

DT-SUB法 JMP=70 , SAS-SUB法 JMP=100

といえる。

変数の数が大きく、かつ、関数も最小項でなく、より大きなキューブで与えると、ＳＡＳ法では項数が増大し、ＤＴほうでは35変数以上で急速に計算時間がかかるようになり、ＤＴ-ＳＵＢ法が有効な手法となる。表３はこれらのことを示すいくつかの計算例である。

但し、関数は我々独自のプログラムで発生させている。すなわち、このプログラムでは、変数の数ｎと、キューブの個数(表中、与えられた項数)ＣＯと、部分濃度d_1, d_2, d_3, d_4を指定すると、各部分に１が１個表れる、２個表れる、３個表れる、４個表れるという確率が各々d_1, d_2, d_3, d_4であるｎ変数のキューブがＣＯ個発生するようになっている。例えば　d_1, d_2, d_3, d_4を1,0,0,0とおけば最小項を発生し、0,0,0,1とすれば　UNIVERSAL CUBEが生成され、部分濃度を変えて種々のキューブが生成できる。

なお、表中の真理値表濃度は分離加法形から求めた

表3　四値入力二値出力関数の計算例

変数の数n	部分濃度 d_1、d_2、d_3、d_4	与えた項数	分離加法形の項数 SAS法	DT法	SAS-SUB法	計算時間(CPU-TIME) SAS法	DT法	SAS-SUB法	真理値表濃度	SAS-SUB法のJMP値
20	0.5、0.2、0.2、0.1	500	17058	1099	887	1分10秒	19分43秒	2分28秒	0.155×10^{-4}	50
20	0.3,0.2,0.2,0.2	100	6833	3092	2187	20秒	8分 8秒	4分26秒	0.162×10^{-2}	10
40	0.3,0.2,0.2,0.1	100	4375	—	100	1分 0秒	—	25秒	0.626×10^{-8}	50
48	1.0,0.0,0.0,0.0	2048	2048	2048	2048	26秒	5分21秒	5分 7秒	0.258×10^{-25}	50

— 数時間経過するも求まらず。

ものである。

5．まとめ

論理式の分離加法形式を得るための新しい手法、ＳＵＢ法を提案し、すでに発表されている（決定木による）ＤＴ法、(笹尾の)ＳＡＳ法と比較を行なった。

本方法では項の数の少ない加法形式が得られるという特長があるが、はじめに与えた論理式が多くの項を含む場合には、長い計算時間がかかるという欠点がある。ＳＡＳ法では短い計算時間で分離加法形が求まるが、加法形式の項の数が多くなりすぎ、メモリ容量からの制約が出てくる。一方，ＤＴ法ではＳＡＳ法より項数の少ない加法形が求まるが、（特に四値入力の場合）多変数の論理式で急速に計算時間がかかるようになり、実用性を失ってくる。

本方法でも，与えられた関数の項数が 50ないし100程度と小さい場合は、比較的早く分離加法形が求まり変数の数の大きいところでは、ＤＴ法を上まわる。したがって、ＤＴ法とＳＵＢ法を組合せたＤＴ－ＳＵＢ法は、これらの長所：短所を相補う特性となり、変数の数の大きい場合に非常に有効となる。又，ＳＡＳ法とＳＵＢ法とを組合せたＳＡＳ－ＳＵＢ法も考えられＳＡＳ法に比べて、さ程計算時間を落とさずに、ある程度項数を減らすことができる。

いったん、分離加法形式に直してしまえば、相当多変数の論理式でも、その真理値表濃度が算出できることをいくつかの例で示しておいた。

ＳＵＢ法は原理的には分岐しないで、直接関数の分離加法形式を求める方法であるが、内項かどうかの判定には木形アルゴリズムを使っている。このように、多変数の論理式を扱う場合は、一つの手法にこだわらず、いくつかの特色ある手法を組み合わせて使うことが必要かと思われる。

参考文献

[1] 高橋，向殿：”論理関数の分離加法形式による表現とその応用”，電子通信学会論文誌 vol. J69-D,NO.8,PP.1145-1152,1986。

[2] A.H.Chan:"Using decision tree to derive the compliment of a binary function with multiple-valued inputs",IEEE Trans.on Comput.,Vol.C-36,No.2,pp212-214,Feb.1987.

[3] T.Sasao:"An algorithm to derive the complement of a binary function with multiple-valued inputs",IEEE.Trans.on Comput. C-34,2,pp131-140,Feb.1985.

[4] 宮腰，松田：”２値論理関数簡単化の一手法－部分マップ法について－”，電子情報通信学会論文誌,Vol.J71-D,No.11,pp.2259-2265, 1988.

[5] S.J.Hong,R.G.Cain,and D.L.Ostapko; "MINI:A heuristic approach for logic minimization",IBM J.Res.& Dev.,18,5, p.443-458 ,Sept.1974.

[6] 松田，宮腰：”論理式を分離加法形式で表現する－手法”，平成元年度電気関係学会北陸支部連合大会.