論理式を分離加法形式で表す手法の時間計算量について

松田　秀雄，宮腰　　隆，畠山　豊正

1．緒　　言

論理式を分離加法形式で表現すると，項が互いに共通部分を持たないので，論理式に含まれる最小項の数が算出でき，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るなど，一般の加法形式にみられない，いくつかの特徴があって重要な表現形式である。論理式を分離加法形式で表現する手法としては，我々は，すでにＳＵＢ法を提案し[(1)]，更にその後，これより精度のよい（求まった分離加法形式の項数がより少ないこと）ＭＡ法を発表した[(2)]。これらの方法では，関数が与えられると，直接この関数について分離加法形式を求めていくものである。一方で，関数をより小さな部分関数に分割して分離加法形式を求めていく，いわゆる，木形アルゴリズムがある。本論文ではＳＡＳ法，ＤＴ法と称する方法がそれで，多変数の関数まで扱えるが，精度が悪い。これに対して，ＭＡ法では精度の点では木形アルゴリズムよりよいが一般に計算時間がかかるということを実験的に示した[(1),(2)]。

ここでは，ＭＡ法，ＳＡＳ法，ＤＴ法の手数，つまり，計算時間量を理論的に求めている。結果は実験（手法をプログラム化し計算機で実行した）値と比較し，その妥当性が示されている。このような手数を求めておくことは，各アルゴリズムが何変数までの関数が実用時間の範囲内で扱えるかを見当つけるために重要である。また，ＭＡ法とＤＴ法とを結びつけたＭＡ—ＤＴ法は双方の欠点を相補って効率のよい方法となる。このとき，ＤＴ法を使って，何個のキューブをもつ部分関数まで分岐させてから，ＭＡ法を適用すると最適かを，理論的に決定するさいにも手数を求めておく問題は必要となってくる。

2．用語の説明

n変数多値入力2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，$P_i=\{0,\ 1,\ \cdots,\ p_i-1\}$，次の例のようにキューブ C_j（j＝1，2，3）の集合で表される。

$$F_1=\begin{bmatrix}C_1=1101-1100\\C_2=0110-0110\\C_3=1001-0001\end{bmatrix}$$

ここに，$n=2$，$p_i=4$，つまり2変数4値入力関数の場合である。キューブ C_j は左より4ビット（一般にpビット）ずつ一で区切られ，それぞれ変数 X_1，X_2（一般に，X_1，X_2，…，X_n）に対応させているので，これらの区分を左より変数 X_1，X_2（一般に，X_1，X_2，…，X_n）の成分と呼ぼう。各成分の4桁のビットは左の桁より，その成分に割り当てられた変数 X_j がとる真理値 0，1，2，3に

対応し，これらの桁で，ビットが1か0によって，X_jがその桁の値を取ったとき，キューブの第j成分はそれぞれ1または0となる。C_1=1101−1100 はX_1の真理値が0または1または3で，かつX_2の真理値の0または1のとき，このキューブは1となる。この様な関係はマップで表すと便利で，図

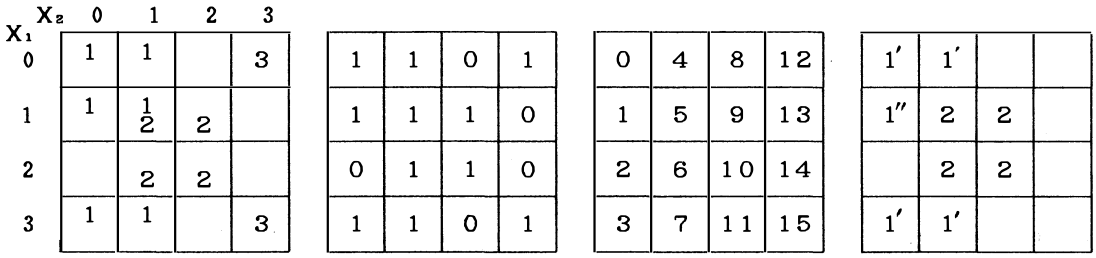

図1　4値2変数のマップを使ってのキューブ，関数の真理値，セルの番号，㊋演算の説明図

1(a)で示される。四角います目はセルと呼ばれ，数字1，2および3と示したセルで各キューブC_1，C_2，C_3はそれぞれ関数値1を取る。従って関数F_1の真理値は図1(b)となる。セルは入力変数X_1, X_2の取る真理値の組合せによって識別できる。この真理値の組合せはp進数とみなされる(但し，左の桁より右の桁に上がるとして)，その大きさで番号付けできる。図1(c)はこれを示す。セル1，2，…，15はキューブで表現して，1000−1000，0100−1000，…0001−0001 と各成分に1のビットが1つだけある。セルのことを基本キューブという。また，最小項ともいわれる。一般にn変数のマップで考えて，一番大きな番号のセル 00…01−00…01−…−00…01 をC_Mで示そう。

(基本キューブとC_Mとの間の距離) 基本キューブC_iの各成分には1のビットが1つだけあり，これを右に桁移動して，C_Mの1の位置に合わせることができる。この桁移動の全成分の総和をC_iとC_mとの距離と定義しよう。

(キューブの最大番号の基本キューブ) キューブC_iにおいて，各成分の最右端の1のみ残し，他の桁のビットをすべて0と置いてできる基本キューブ$C_{i,ma}$はC_iに含まれる基本キューブのうち，番号最大のものでキューブC_iの最大番号の基本キューブという。例えば，上のC_2=0110−0110をC_iとし，$C_{i,ma}$=0010−0010（セル10のこと）である。

(キューブの体積) キューブにおいて，各成分の1の数U_iの積$\prod_{i=1}^{n} U_i$をこのキューブの体積という。

(キューブの拡大) キューブC_iは初め関数Fの1 (true) のセルのみを含んでいるものとしよう(このようなキューブはFの内項といわれる)。各成分の1の数を増やしていくと，キューブC_iの体積は大きくなるが，しまいには関数Fの0 (false) のセルをも含むようになる。この0のセルを含むようになる寸前のキューブをFの主項という。また，このような操作をキューブの拡大という。

この他に，真理値表濃度，併合という言葉も出くる[(1)]が，ここでは，説明を省略する。また，キューブC_iからキューブC_jに含まれるセルを取り除く操作にディスジョイント・シャープ演算 (㊋演算) が使われる。これも文献(1)にあるので，ここでは1例C_1 ㊋ C_2の結果を示すにとどめる。図1(d)のように，C_1のセルのうちC_2にも含まれるセル5が除かれ，C_1=1101−1100はC_1'=1001−1100(図

中，数字1′と印したセル）と C_1''＝0100－1000（図中，数字1″と印したセル）とに別れる。キューブの集合 C_1'，C_1''，C_2，C_3 は互いに共通部分がなく，分離しているという。関数 F が与えられたとき，でるだけ個数の少ない分離したキューブの集合（これは代数的には加法形式とみられる）で表すことが本稿の主題である。

3．方　法

初めに，二つの木形アルゴリズムについて述べる。いずれの方法も次のキューブの集合で表される関数 f で説明する。

$$f = \{C_i\} = \begin{bmatrix} 100-111-010 \\ 111-011-010 \\ 010-111-001 \\ 011-110-001 \\ 010-100-101 \\ 100-011-110 \end{bmatrix}$$

（但し，上より $i = 1, 2, \cdots, 6$）

3.1　変数により分岐する手法（DT法）[1),(3)]

Chan の決定木（decision tree）による方法[3]なので，DT法と略記する。キューブ表現において，各成分は変数に対応するのでここでは変数と呼ぼう。キューブで全部1のみからなる変数を **1** の変数といい **1** でない変数を $\bar{\mathbf{1}}$ の変数といおう。まず，

（1）　根となるべく，変数の一つを次のようにして選ぶ。$\bar{\mathbf{1}}$ の変数の数が最も少ない項に注目し，その項の $\bar{\mathbf{1}}$ の変数の中，他の最も多くのキューブおいても $\bar{\mathbf{1}}$ の変数となっている変数を選ぶ。本例では X_3 がまず選ばれる。この根の $X_3 = 0$ の枝には，キューブの内で X_3 の成分の（左より）1ビット目が1のキューブを集める。但し，X_3 の成分を **1** に変える。

$$\begin{vmatrix} 010-100-111 \\ 100-011-111 \end{vmatrix}$$

これをfと考え再び（1）で変数を選ぶ。X_1 が選ばれこの根の $X_1 = 0, 1, 2$ の各枝にはfのキューブの内で X_1 の成分の（左より）1ビット目，2ビット目，3ビット目がそれぞれ1のキューブを集める。すると，この順で {111－011－111}，{111－100－111}，{ϕ} となる。但し，X_1 の成分を **1** に変える。これらのキューブの集合はいずれも **1** の変数が一個か空であるので，それぞれを葉とし，これ以上の枝出しはしない。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。

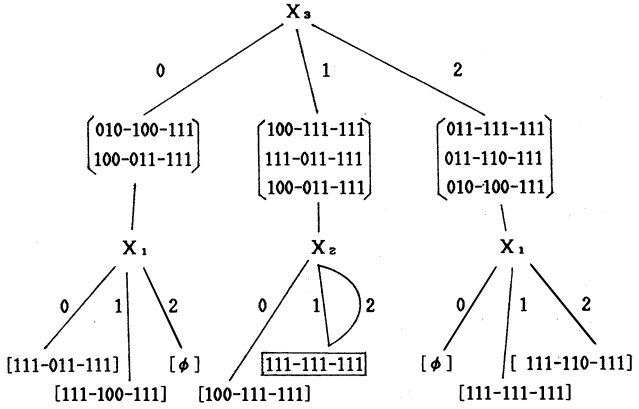

図2　DT法の法定木

（2）　（ϕ でない）各葉から，根までの道の枝のリテラル（但し，ビット表現したもの）をかけ合わせてできるキューブの和（集合）がfの分離加法形式を形

成する。

例えば，図2の一番左の葉 111－011－111 は X_1 が0の枝のリテラル 100－111－111 X_3 が0の枝のリテラル 111－111－100 をかけて 100－011－100 となり，他の葉も根までかけ合わせて，010－100－100，100－100－010，111－011－010，010－111－001，001－110－001 の計6個のキューブを得る。

なお，ここでは，図2の四角のます目で囲んだように，一つの根から出ている枝が同じ値の葉をもてば，一つの葉にまとめて，多重枝にしている。すなわち，X_2 が1と2のリテラルは一つにまとめて 111－011－111 とする。我々は，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ている。これは，葉から根へと向うキューブの合成過程で，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）[(1),(4)]

笹尾の否定を求めるアルゴリズム[(4)]から分離加法形式を求めるので，ＳＡＳ法と略する。

(1)　根となるべく，キューブ C を $\bar{1}$ の変数の数が最小のものから選ぶ。

(2)　この根からディスジョイント・シャープ演算 C_u ㊋ C で求まっただけのキューブの枝出しが行われる。但し，C_u は universal cube（全成分が1のキューブ）であり，ディスジョイント・シャープ演算を取る変数の順序も木が大きくならないよう工夫がされている。

関数 f の例の場合(図3参照)，(1)で，まず根 r_1 として $t_1=C_2=$111－011－010 が選ばれ変数 X_1，X_3，X_2 の順で㊋演算が行われる。その結果，図3のように二つのキューブ t_1, t_7 が得られ，それぞれの枝で f との論理積をとる。$f\cdot t_7$（右側の枝）の方は一つのキューブ l_5 しか残らないからこれを葉とする。$f\cdot t_1$（左側の枝）の方は［ ］中で示した複数個のキューブがあるので，(1)へもどって，再び根となるキューブ r_2 を選ぶ。C_u ㊋ r_2 で更に枝出しを行い同様につづけると図3の木が完成する。

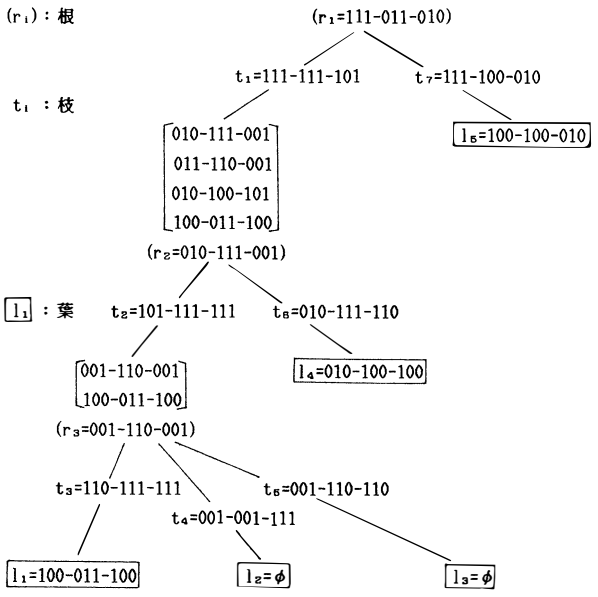

図3　ＳＡＳ法の例

この例の分離加法形は ϕ でない葉から根 r_1 までの道を形成する各枝のキューブをかけ合わせることにより得られる次の三つのキューブ

$l_1\cdot t_3\cdot t_2\cdot t_1=$100－011－10,

$l_4\cdot t_6\cdot t_1=$010－100－100,

$l_5\cdot t_7=$100－100－010

と，根 r_1, r_2, r_3 の三つのキューブの和からなる。

なお，本方法では上記の手法のように，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。

3.3　我々の手法（ＭＡ法）[(2)]

ここに提案の手法は木形アルゴリズムではないので，マップで直接説明できる。本章の例で使っている f のマップは図4(a)のようになる。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。まず，$F=\phi$（空集合）としておく。

そして，

(1) 各キューブ C_i(i=1, 2, …, r)の最大番号の基本キューブ $C_{i,\ ma}$ を求める（但し，本例では r=6)。f の例では図4(a)の i^a で示したセルが各 C_i の最大番号の基本キューブである。

(2) 最大番号の基本キューブの集合 $\{C_{i,\ ma}\}$ のうち C_M との距離が最も小さいものを選び出して $\{C_{max}\}$ とする。$\{C_{max}\}$ の基本キューブを含む元のキューブの集合の中から体積最大のものを取り出して（複数個あれば，任意に一つ選んで）C_0 とする。

いまの例では(2)の $\{C_{max}\}$ は $\{4^a, 3^a, 2^a\}$，これらの基本キャーブを含む元のキューブは C_4, C_3, C_2 である。この中で体積最大のものは C_2=111-011-010，$C_0=C_2$ ときまる。

(3) C_0 を関数 f の主項に拡大する。拡大したキューブを C_0 として，

(4) 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。

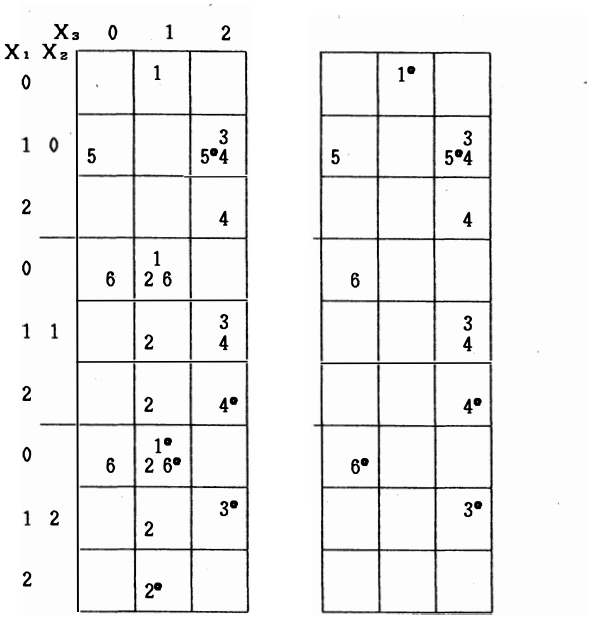

図4　我々の手法（MA法）の説明図

本例では $C_0=C_2$=111-011-010 を(3)で拡大するが，これ以上大きくならない。(4)で C_0=111-011-010 を F に加える。f ㊋ C_0 の結果，C_0 と交わらない C_3, C_4, C_5 は前のまま変わらず，C_0 と交わる C_1 と C_6 はそれぞれ，C'_1=100-100-010，C'_6=100-011-100 とに変わる。これらのキューブ集合を新たな f のして（図4(b)，但し，C'_1，C'_6 のセルを1，および6で表している），再び，(1)に返り，同様の手順を繰り返すとステップ(4)の C_0 として，C_4=011-110-001 が求まり，これを F に加える。以下同様にして，あとの4つのキューブ 010-001-001，100-011-100，100-100-010，010-100-100 が順次 F に求まってきて，本例題の分離加法形式を形成する。

なお，上記3つのいずれの方法も関数を与えたとき，一旦，併合を前処理的に行うことがある。

4．時間計算量

4.1 ＭＡ法の手数について

p 値入力，変数の数を n，初めに与えた（あるいは，併合を行うときには，併合後の）関数のキューブの項数を r，得られた分離加法形の項数を q，k_1，k_2，k_3，・・・ら，それと k を定数とする。

まず，併合がある場合，これは初め一回だけで，ＭＡ法では他の処理時間と比べて小さいので無視してよい。そこで，ＭＡ法のステップ(1)から(4)までの処理に対しては次のような手数が考えられる。

a）各キューブ C_i(i=1, 2, …, r)の最大番号の基本キューブ，$C_{i,\ ma}$ を求める手数：k_1npr（このことは，一つのキューブの最大番号の基本キューブを求めるには n 個ある成分のそれぞれにおいて p ビットを調べて1であるビットの一番右端の1だけを残す操作が必要であることから出てくる。）

b）各 $C_{i,\ ma}$ と C_M との距離の計算の手数：k_2npr

c）最大番号の基本キューブの集合$\{C_{i,\ ma}\}$のうちC_Mとの距離が最も小さいものを選び出して$\{C_{max}\}$とするための手数（ヒープ法を使う）：$k_3 r \log_2 r$

d）$\{C_{max}\}$の基本キューブを含む体積最大キューブC_0を主項に拡大するための手数：$k_4 npr$

e）関数fからC_0をデイスジョイント・シャープ演算で取り除くための手数：$k_5 npr$

結局，分離加法形の項が一つ求まるごとにa）～e）を1回繰返すので，全体の手数は

$$\text{全体の手数} = knprq \qquad (2)$$

になる。但し，c）の$k_3 r \log_2 r$はnが大きいときは無視してもよいし，例えば，2値入力関数の場合には，nが9から12で初めの項の数rが100～700のような値になる[2]ので，$\log_2 r$を近似的にnと等しいとおいてもよいとした。また，厳密には，rの値は分離加法形の項が一つ求まるごとに減っていく（ふえる場合もある）と思われるが，簡単のため同じとしている。

4.1.1　2値入力の場合について　関数は乱数を使って最小項で発生させていると考える。まず，2値入力の場合，分離加法形として，最も項の多い関数は図5（a）で示した例のように（カルノー図でかくと）交互のセルで1(true)となる関数で，項の数は2^{n-1}ある。この関数は初めに与えた関数のキューブの項数rも，併合後の項数も，最後に求める分離加法形としての項数qも変らず

$r=q=2^{n-1}$,従って,MA法の手数は式(2)より，$kpn(2^{n-1})(2^{n-1})=kpn(4^{n-1})$でおさえられ，$O(n4^n)$で増加するとみられる。実際我々が発表した2値入力2値出力関数の実験結果は変数の数nが1つふえるごとに4倍ずつ増加している[2]。

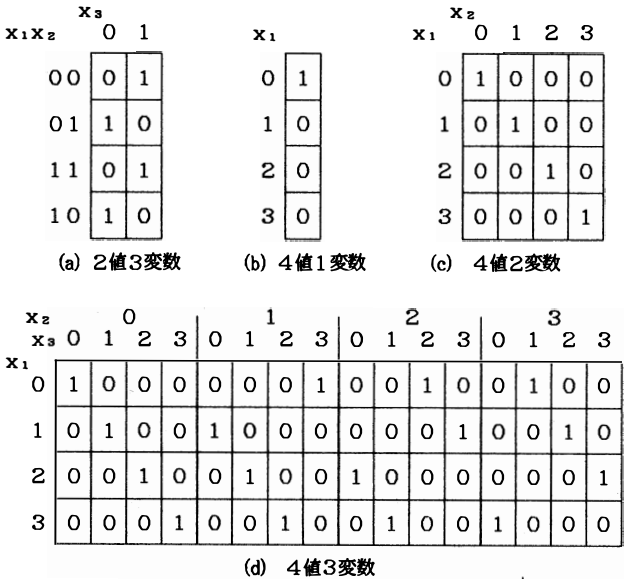

図5　項数が最も多い関数の例

4.1.2　p値入力の場合について　例えば，4値入力の場合，分離加法形として最も項数の多い関数の項数は，一変数で1，2変数で4，3変数で16（図5(b)，(c)，(d)は，それぞれ，1変数，2変数，3変数のそのような関数の例で，これらの関数のどの0のセルを1に変えても分離加法形として項数は減りこそすれ，ふえはしない）で，すなわち，n変数でp^{n-1}となる。従って，MA法の手数は式(2)の　rとqにp^{n-1}を代入して，$knp(p^{n-1})(p^{n-1})=kpn(p^{2n-2})$でおさえられ，$n$が大きいとき，$O(np^{2n})$で増加するとみられる。

表1は4値入力2値出力関数についての計算例である。関数は乱数を使って，trueとなる最小項の数が真理値表濃度dを，0.2，0.5，0.8になるように発生させた。各々につき10個ずつ計30個の関数の平均値について示したものである。MA法では変数の数nが4，5，6のとき，分離加法形の平均項数がそれぞれ37.20，143.87，552.47，平均計算時間(ms)が196，2988，50623になっている。すなわち，項数は約4倍ずつ，計算時間は約16倍ずつふえており，項数が$O(p^n)$，手数が$O(np^{2n})$で増加するであろうという上の推測とよい一致を示している。

4.2　木形アルゴリズムの手数について

木形アルゴリズムの手数については，各部分木を作る手数，すなわち，分岐点での関数の項数の総

表1　四値入力二値関数についての計算例

変数の数n	併合後の項数	分離加法形の項数			平均計算時間(CPU-TIME:ms)		
		SAS法	DT法	MA法	SAS法	DT法	MA法
併合あり 4	38.47	42.13	40.40	37.20	67	77	196
併合あり 5	152.87	174.93	157.87	143.87	881	949	2988
併合あり 6	590.87	705.20	613.87	552.47	13725	14264	50623
併合なし 4	131.80	56.60	40.47	37.47	14	78	289
併合なし 5	512.53	228.73	161.20	145.60	72	515	5011
併合なし 6	2055.53	906.47	624.80	557.93	333	3424	90468

n=4～6については真理値表濃度 0.2，0.5，0.8 各5個ずつ，計15個の関数の平均値，
併合なしの場合は併合後の項数

和を算出すれば求まると思われる。図5の関数についてこれを算出してみる。

4.2.1　DT法の木の項の総数　　DT法では例えば4値3変数の関数（図5(d)）の各分岐点での項数は図6のようになる。すなわち，初めに16個あった項はある一つの変数の値0，1，2，3の各分岐でその1/4に減り，更に他の変数の値0，1，2，3での各分岐で1個になってしまう。項の総数は $16+4\cdot4+1\cdot16=3\cdot4^2$ となる。一般的にいって，p 値 n 変数関数では初めの p^{n-1} 個の項は一つの変数の分岐ごとに $1/p$ になり，$n-1$ 回の分岐で1個の項になりそれ以上分岐しない。また，そういう分岐点の数は初めは1個で1回の分岐ごとに p 倍づつふえる。したがって，初めの分岐点，つまり，根から同じ深さの分岐点の数と分岐点での項の数の積は常に p^{n-1} となり，それが $n-1$ 個あることになるから，木全体の分岐点での項の総数は np^{n-1} となる。

4.2.2　SAS法の木の項の総数　　図7，図8はSAS法の4値3変数の関数（図5(d)）の木である。図8一番上のマップが初めの関数で16個の項数がある。それがマップ一番左上隅のキューブの㊋演算の結果A1，A2，A3の各枝のマップの網掛けの部分に分岐する。A1，A2，A3の各枝の項数はそれぞれ12，3，0で同図に示してある。A1の12個の項はマップ上から2行目左から2列目の1のキューブの㊋演算の結果B1，B2，B3の各枝に分かれる。枝のそれぞれの項数は8，3，0でB1，B2についてはマップの網掛けで示す。A3やB3のように項数が0個の枝は点線で示す。B1の8個の項はマップ上3行3列目の1のキューブの㊋演算の結果C1，C2，C3の各枝に分岐し，各項数は4，3，0でC1，C2についてはマップの網掛けで示す。C1の4個の項はマップ上4行4列目の1のキューブの㊋演算の結果D1，D2，D3の各枝に分岐し，各項数は0，3，0で，

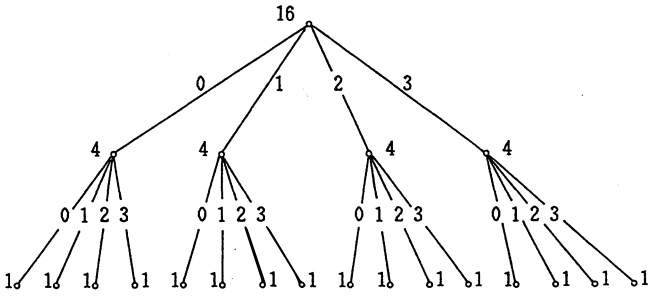

図6　DT法の本の各分岐点での項数

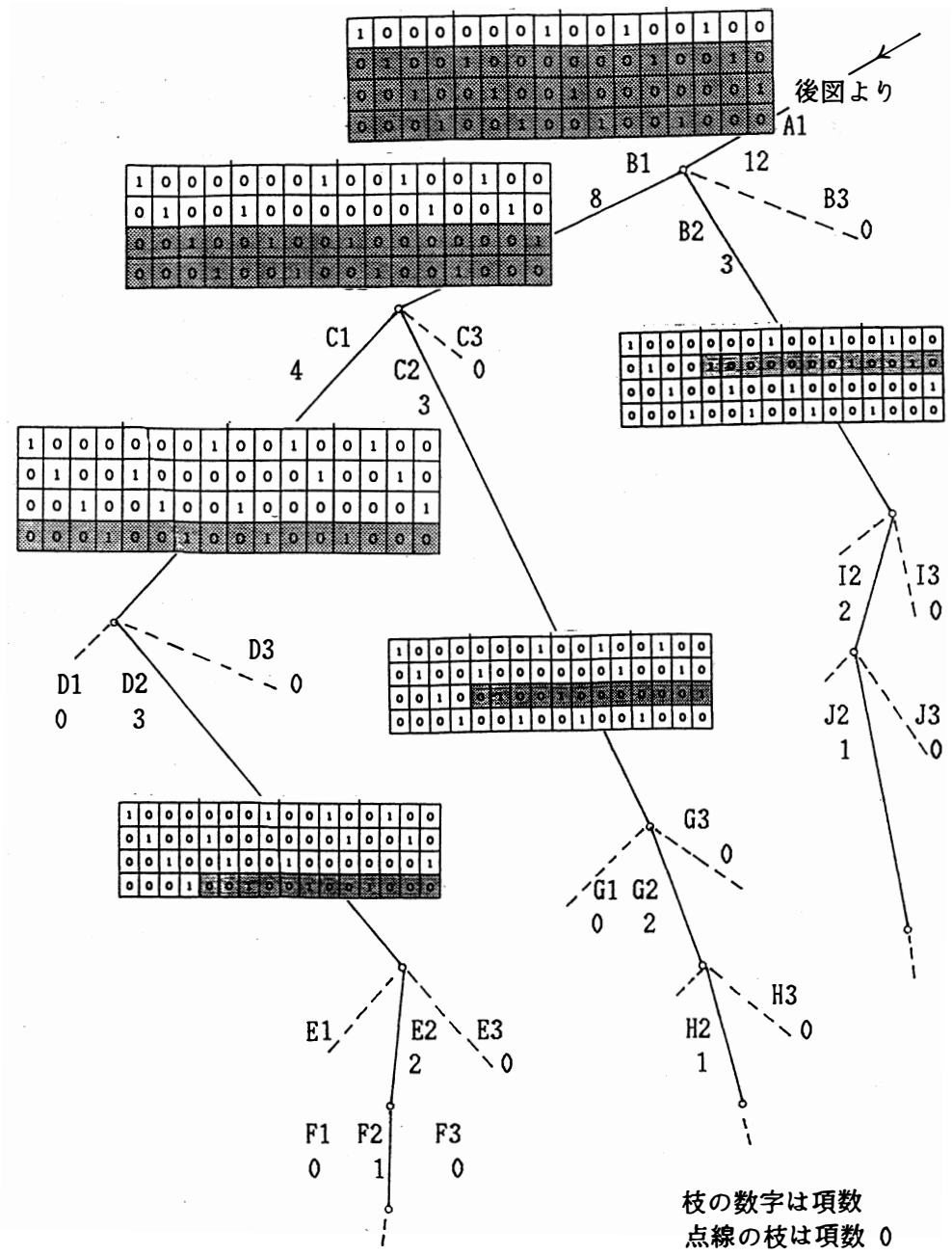

図7　ＳＡＳ法の各枝での項数の例（その１）

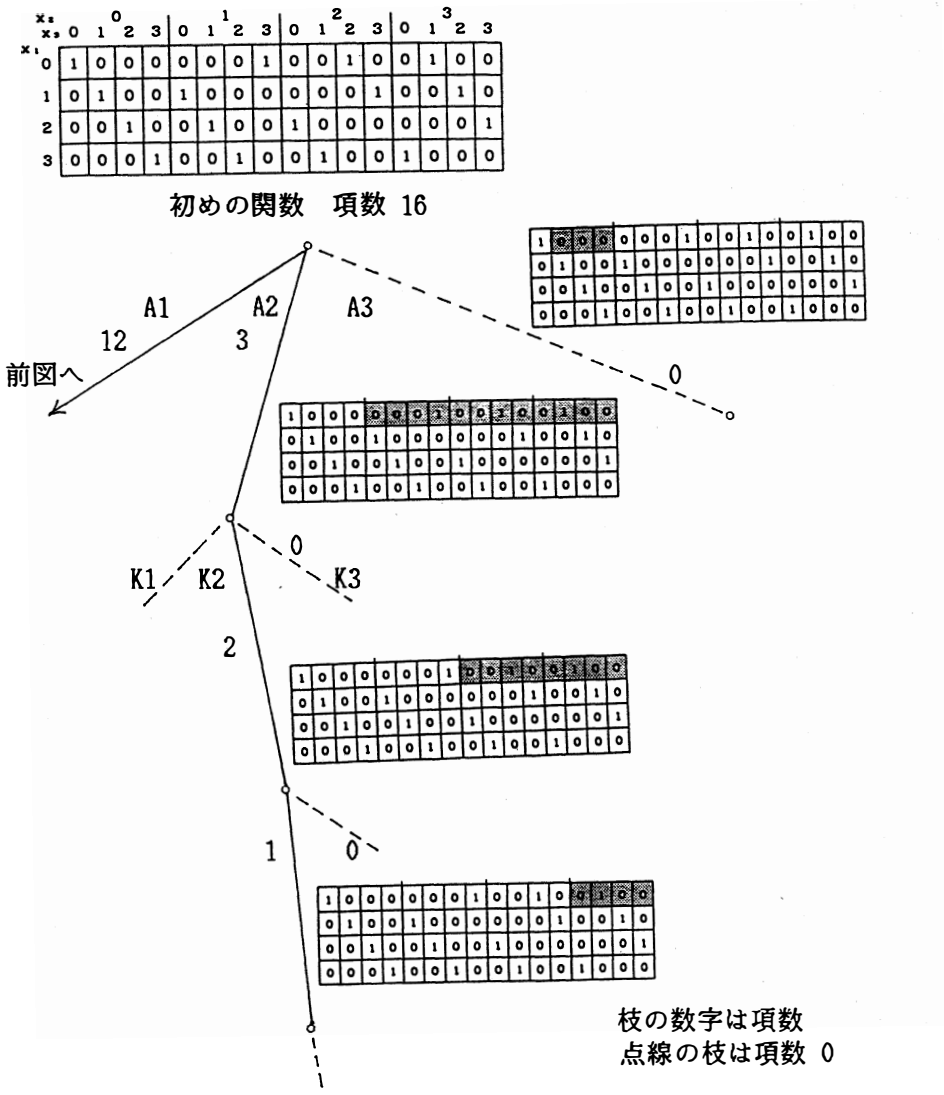

図8　ＳＡＳ法の各枝での項数の例（その２）

D２の３個の項は更にＥ１，Ｅ２，Ｅ３の分岐を生じ，Ｅ２はＦ１，Ｆ２，Ｆ３に分岐した項数が１または０で，ようやく図７の最左端方面の部分木の枝出しが終る。途中のＡ２，Ｂ２，Ｃ２の各枝にも３個の項があるので，Ｄ２以下の部分木と同じ木がこれらにもできて，木が完成する。

項の総数は $16+(4+8+12)+4\cdot(1+2+3)=4^{3-1}+(3-1)\cdot 4^{3-2}\cdot(1+2+3)=p^{n-1}+(n-1)\cdot p^{n-2}\cdot(1+2+3+(p-1))$，但し，$n=3$，$p=4$ である。

また，図９は４値４変数の場合の項数の最も多い関数の木の例である。初め，$4^{4-1}=64$ 個あった項は一連の㋫演算の結果図中Ａ，Ｂ，Ｃ，Ｄで示した分岐を生じ一番左端の枝をたどると，48，32，16の項ができる。更に，Ｄの16個の項から T_D と記した鎖線で囲んだ部分木ができる。この中をみると，分岐点Ｅ，Ｆ，Ｇでの項数がそれぞれ12，８，４，それと分岐点Ｈ，Ｉ，Ｊでの項数３，２，１があり，このＨ，Ｉ，Ｊと同じ部分木が T_D の中の他の３個所の黒丸をつけた分岐点以下にできるので，T_D の項の総数は $12+8+4+4\cdot(3+2+1)$ である。ところが，T_D と同じ部分木が分岐点Ａ，Ｂ，Ｃのそれぞれの左端の枝を除いた三つの枝からできる。結局図９の木全体の項数は $4^{4-1}+48+32+16+4\cdot(12+8+4+4\cdot(3+2+1))=4^{4-1}+(4-1)\cdot 4^{4-2}\cdot(1+2+3)=p^{n-1}+(n-1)\cdot p^{n-2}\cdot(1+2+(p-1))$，但し，$n=4$，$p=4$，となる。以上，$p$ 値入力 n 変数関数の木の項の総数が最大 $p^{n-1}+(n-1)\cdot p^{n-2}\cdot(1+2+\cdots+(p-1))$ になることを $p=4$ で，$n=3,4$ の関数の場合で示したが，一般にも成り立つ。まず，与えられた項数が p^{n-1}，これが連続した $p-1$ 個のキューブの㋫演算により $(((p-1)+(p-2)+\cdots+1)/p)\cdot p^{n-1}$ の項を生じ，$(1/p)\cdot p^{n-1}$ の項からまた連続した $p-1$ 個のキューブの㋫演算により $(((p-1)+(p-2)+\cdots+1)/p)\cdot(1/p)\cdot p^{n-1}$ の項を生じ（但し，このような分岐点は木の中に p 箇所できる），更に，$(1/p)^2\cdot p^{n-1}$ の項から引き続き $p-1$ 個のキューブの㋫演算により $(((p-1)+(p-2)+\cdots+1)/p)\cdot(1/p)^2\cdot p^{n-1}$ の項を生じる（但し，このような分岐点は木の中に p^2 箇所できる）。結局このようなことが p^{n-1} を p で割って１になるまで，つまり $n-1$ 回行われ，項数の総計は上記で示した値となる。

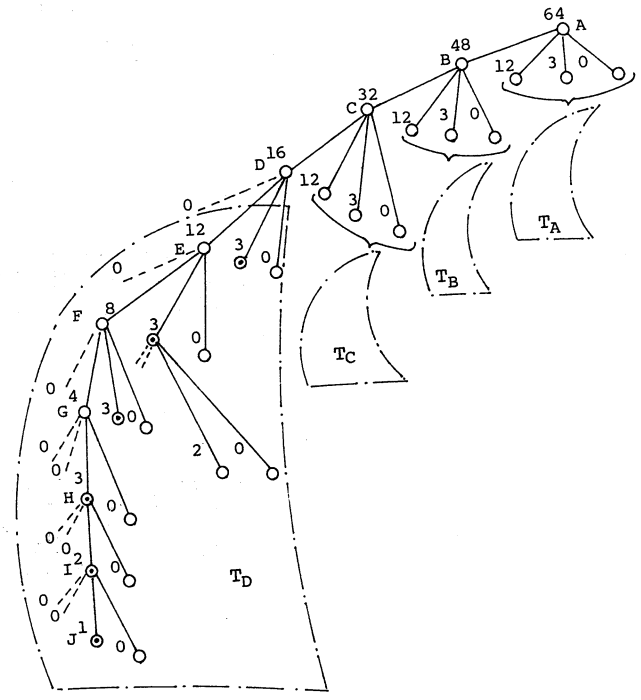

図９　４値４変数の場合で項数の最も多い関数のＳＡＳ法によってできる木

4.3　木形アルゴリズムの手数について

一つの項についての手数（処理時間）は変数の数 n と，成分のビット数，つまり，p の積に比例するので，ＤＴ法の手数は $np\cdot np^{n-1}$ で押さえられ，ＳＡＳ法の手数は $np\cdot(n-1)p^{n-2}(1+2+\cdots+(p-1))+np\cdot p^{n-1}$ で押さえられる。手数のオーダーは n が大きいとして，ＤＴ法，ＳＡＳ法ともに $O(n^2p^n)$ と考えられる。しかし，ＤＴ法では合成過程で併合操作があるので，ＳＡＳ法よりいくらか手

数が多くなる。そうはいっても，初めに関数を与えたときに行う併合の手数が $np^{n-1}\cdot p^{n-1}$ に比例するのに対し，合成過程での併合操作の手数は大略 $(p^{n-1}\cdot p^{n-1})/p$ に比例するので十分小さい手数だと思われる。

4値入力の場合手数は $O(n^2 4^n)$ で，変数の数 n が4，5，6と変化したとき，おおよそ，4倍ずつ計算時間がふえることになるが，表1の併合ありの場合のＤＴ法，ＳＡＳ法の計算時間はともに約15倍ずつふえている。これは初めの併合のための計算時間に対し，ＤＴ法，ＳＡＳ法自体の計算時間が非常に小さくこれにうもれてしまうからである。併合のための計算時間は項数の自乗に比例する。実際，表1では n が4から6と変化したとき，初めに与えた関数の平均項数（あるいは表1の併合後の項数でみても）が4倍ずつふえており，結局16倍ずつ計算時間がふえるものと思われ，計算結果は妥当なものと考えられる。

表1で併合なしの場合の計算時間は変数の数 n が4，5，6と変わったとき，ＤＴ法では約6.6倍ずつふえ，ＳＡＳ法で約5倍ずつふえている。n の値が4から5に変わったとき，および，5から6に変わったときの手数 n^2p^n の比は6，および，5.8である。少しおおまかではあるが，手数についての上の推測値 $O(n^2p^n)$ が，妥当なものといえよう。

4．結　　言

n 変数 p 値入力2値関数を分離加法形式で表現する3つの手法，(我々の)ＭＡ法，(決定木による)ＤＴ法，(笹尾の)ＳＡＳ法の手数，つまり，時間計算量を理論的に求めた。いずれの手法も手数は関数のもつ項の数に関係するので，手数の上限を与える項数の最も多い関数について検討した。n が大きいとき，ＭＡ法の手数は $O(np^{2n})$ とみられ，木形アルゴリズムであるＤＴ法とＳＡＳ法の手数はともに $O(n^2p^2)$ となる。これらの推測値が妥当なことを実験値と照らし合わせて示しておいた。実際にも，ＭＡ法は，一般の関数で変数が増加すると(2値入力の場合なら，n が13以上で)，非常に時間がかかるようになる。しかし，項数が少い関数では逆に多変数で，ＤＴ法よりはるかに早く分離加法形式が求まる。従って，多変数の関数を扱う場合には，ＤＴ法で項数の小さい関数に分割していき，項の数が m 個以下になったら，ＭＡ法で分離加法形式を求めるＭＡ—ＤＴ法は精度，計算時間の両面からみて，優れた方法である。このとき，n に対して m の値をどのように決めれば最適かといった問題が生じてくるが，それを考える上からも，ここで行った時間計算量の変数の数 n への依存性のみならず，関数のもつ項数 r への依存性の検討も必要かと思われる。

参 考 文 献

1) 松田，宮腰：論理式を分離加法形式で表現する一手法，情報処理学会研究会報告，**90-DA-51-6** (1990).

2) 松田，宮腰：論理式を分離加法形式で表現する一手法(2)，平成2年度電気関係学会北陸支部連合大会講演論文集，p.230 (1990).

3) A.H. Chan: Using decision trees to derive the complement of a binary function with multiple-valued inputs, *IEEE Trans. on Comput.*, **Vol. C-36**, No. 2, pp. 212-214 (1987).

4) T. Sasao: An algorithm to derive the complement of a binary function with multiplevalued inputs, *IEEE. Trans. on Comput.*, **Vol. C-34**, No. 2, pp. 131-140 (1985).

On the time Complexity of Methods to Derive the Disjoint Disjunctive Form of Logical Expressions.

Hideo MATSUDA, Takashi MIYAGOSHI and Toyomasa HATAKEYAMA

In this paper, we deal with the time complexty of (our) MA method, (SASAO's) SAS method and (CHAN's) DT method to derive the disjoint disjunctive form of a binary function with p-valued input. The times complexity of MA method is derived directly by estimating the number of trouble expended on each step of the algorithm for one of the function which has the maximum number of terms in the case of n-variables. That of SAS method and DT method which belong to tree type algorithms is derived by enumerating the summation of the number of terms at each node of those trees for the same function. The following time complexities are shown: for MA methode, $O(np^{2n})$: both of SAS method and DT method, $O(n^2p^n)$.

〔英文和訳〕

論理式を分離加法形式で表す手法の時間計算量について

松田　秀雄，宮腰　　隆，畠山　豊正

p値入力2値関数を分離加法形式で表す3つの方法，(我々の)MA法，(笹尾の)ＳＡＳ法，(CHANの) ＤＴ法の時間計算量を求めている。MA法の時間計算量は，n変数の関数で一番項数の多い関数の一つについて，アルゴリズムの各ステップで必要な手数を直接求めて算出している。ＤＴ法，ＳＡＳ法の時間計算量は，これらが木形アルゴリズムなので，同じ関数について各分岐点での項の総数を算出することにより導いている。結果は，MA法では$O(np^{2n})$，ＳＡＳ法，ＤＴ法ではともに$O(n^2p^n)$となる。