最小被覆問題の素数表現による解法

松田　秀雄・本庄　邦幸*・宮腰　隆

緒　　言

論理設計を行なう場合，まず与えられた関数のプライムインプリカントを導出し，次いでこれらのプライムインプリカントの中から，関数を実現する最小被覆を求める，といった2つのステップが必要である。プライムインプリカントの計算機による導出法として，筆者らはすでに関数変換法，及び縮小カルノー図法の2つの方法を考案している[(1)]。

最小被覆問題の解法としては，0－1計画法[(2)]や擬似ブール計画法[(3)]が一般に知られているが，ここでは，プライムインプリカントを計算機上素数で表現することにより，最小被覆を求める方法について述べる。この方法は着想が簡単で，数式処理で行なえるのでプログラムが容易になり，5～6変数までの論理関数なら，最小被覆が非常に早く求まるという特長がある。

1．最小被覆問題の素数表現による解法

1．1　原　　理

最小被覆の問題はカバー表（プライムインプリカントチャート）により表わされるが，表1に簡単な例を示す。表の上欄1，2，3は最小項，左の欄のA，B，Cはプライムインプリカント（以下PIと略）である。Aの行には1と2の列に×印があるが，これはAが最小項1と2をカバーしていることを示す。又B，Cはそれぞれの行の×印のある列の最小項2，3及び1，2をカバーしている。問題は1，2，3のすべての最小項をカバーするために必要な最も少ない個数のPIをいかに選ぶかということである。表1の最小被覆を手計算で求める通常の方法はA，B，Cを論理変数のごとくみなして

表1　カバー表の例

	1	2	3
A	X	X	
B		X	X
C	X	X	

$$(A+C)(A+B+C)(B)$$
$$=(A\cdot A+A\cdot B+A\cdot C+C\cdot A+C\cdot B+C\cdot C)(B)$$
$$=(A+C)(B)=A\cdot B+C\cdot B \qquad (1)$$

と論理展開して解を求める。但し，かけ算は論理積，＋記号は論理和の演算を表わす。ここで用いた簡単化のための公式は

（べき等律）　$A\cdot A=A$　(2)

（吸 収 律）　$A+A\cdot B=A$　(3)

の2つである。

計算機上で上記の手法で最小カバーを求めるとすればPIの内部表現が計算機上次の性質をもつことが要求される。

(i)　公式(2)，(3)の処理が容易にできること。

(ii)　式(1)の最終結果から，それぞれの積項を構成している変数成分（$A\cdot B$ならAとB）が識別できること。

* インテック

そこでA，B，C……などの各変数に素数p_1，p_2，p_3……などをわりあてるとこの要求をみたし，最小被覆の選出ができることを以下に示そう。

表１の例ではまず，A，B，Cに素数２，３，５をわりあて式(１)に対応して

（２＋５）（２＋３＋５）（３）

＝（２・２＋２・３＋２・５＋５・２＋５・３＋５・５）（３）　　（４・ａ）

と展開する。かけ算は通常のかけ算だが，＋記号は和の意味はなくここでは区切り記号の役目をはたす。従って，計算機上では各積項は別々の変数に記憶される。

ここで式（4･a）の（　）の中第一項2･2（実際４と記憶されている）は２でわり切れるので，２そのものとする。一般に素数p_iのｎ乗の項があればp_iでわり切れるのでp_iそのものとおく。この操作によって，式(２)が処理される。

又，２＋２・３は第２項が第１項でわり切れるので，第２項を省略し，２＋２・３＝２とおく。一般に素数の積項$p_1 \cdot p_2 \cdot \cdots \cdot p_n$に対し，他のより小さい積項$p_{1'} \cdot p_{2'} \cdot \cdots \cdot p_{k'}$（$\{1', 2', \cdots\cdots, k'\} \subseteq \{1, 2, \cdots\cdots, n\}$）があって，わり切れると，はじめの大きな積項を省略する。この操作によって，公式(３)の処理を行なう。これらの公式により，上式の計算を続けると，

＝（２＋５）・３

＝２・３＋５・３　　（４・ｂ）

となる。最後の結果は計算機上では６（＝２・３）と15（＝５・３）の２つの数として記憶されていることになる。これらの数を，いまPIにわりあてた素数２，３，５でそれぞれ順次わって，わり切れればその積項を構成しているPIであると判定する。６は２と３でわり切れ，15は３と５でわり切れるので最小被覆はA，BとB，Cであることがわかる。一般に素数の積項$p_1^{n_1} \cdot p_2^{n_2} \cdot \cdots\cdots \cdot p_m^{n_m}$は$p_1$，$p_2$，……，$p_m$でわり切れ，他の素数$p_j$ではわり切れない。このことから性質(ii)の論理積を構成している変数を識別することができる。

さて，以上，原理の大筋を述べたが，カバー表が大きくなるとそのために生ずる若干の問題点を解決するための工夫が必要となる。以下，流れ図を示しながら，この点を明らかにしたい。

１．２　流　れ　図

図１は素数表現によって最小被覆を求める方法の流れ図である。又図２は，図１のⓐ－ⓑ間を詳しく書いた流れ図である。

まず，実現したい論理関数をデータとして与え，これよりPIを求め，カバー表を作成する。次にカバー表にエッセンシャルプライムインプリカント（EPI）があれば抽出して行及列の縮約をはかる。更に行支配，列支配の技法を使ってカバー表の縮約を行なう[(4)]。縮約した表から新たにEPIが見つかれば，それを抽出して，縮約の技法を続ける（図１ループ１）。

カバー表の縮約が終わり，行の数が制限以内であれば，各PIに素数をわりあて，すべての最小項がカバーされるよう式(４)のように論理展開式の計算処理を行なう。

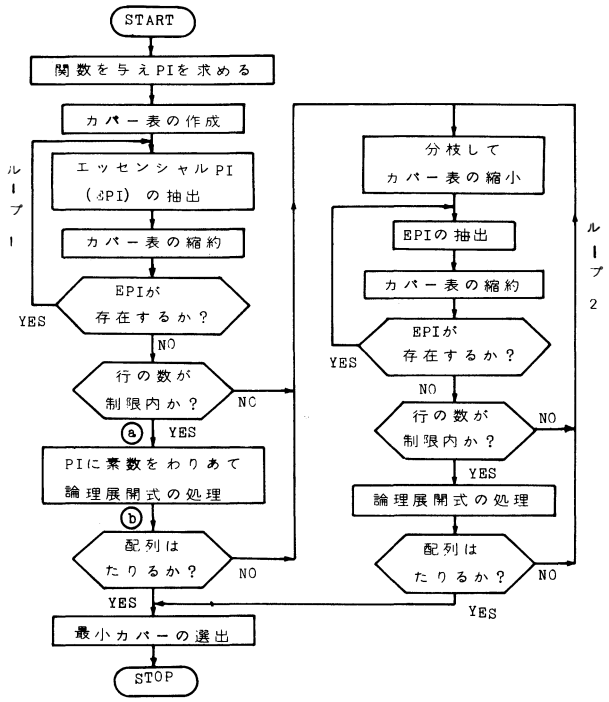
しかし，カバー表が大きくなるとPIの数がふえ，それにわりあてられる素数も大きくなり，積項の数値が著しく増大して，計算機（FACOM230－45S　本学計算センター）の整数型変数の最大値$2^{31}-1$（約21億）をこえover flowを生じ計算不可能となる。この over flow を生じないで計算し得るカバー表の大きさは本学の計算機でおよそ15行×15列位までである。

しかし，実際問題としては，更に大きなカバー表を処理しなければならない。そのための１つの技法として，表２のようにPIをいくつかのグループに分け，それぞれのグループで一番小さい素数から順次わりあてていくようにする。この場合，同じ素数がいくつものPIにわりあてられることになるが，グループごとにベクトルの異なった成分になるよう表現すれば混同する恐れはない。表２の例では，グループⅠのPIは（p_1，１），（p_2，１），（p_3，１），（p_4，１）と表わし，グループⅡのPIは（１，p_1），

（1，p_2），（1，p_3）と表わす。もし、論理展開式でPIの積項AB，EF，CFなどが表われるとするなら，これらは素数表現として，それぞれ，（$p_1 p_2$，1），（1，$p_1 p_2$），（p_3，p_1）と表わされる。このように素数の積項を各グループに対応した部分積項にわけて多次元的に表現すれば，上記のover flowの問題も解決できる。

原理的にはこれでいくらでも大きなカバー表を一度に処理できることになるが，いま1つの問題が生じてくる。それは表が大きくなると，論理展開式の項が急速に増加し，そのために用意した配列の大きさに不足をきたすということである。本学の計算機では積項の数の最大は2000を限度とし，カバー表の大きさでいえば，20行×20列位のものまでしか扱えない。

図1　素数表現によって最小被覆を求める方法の流れ図



しかし，これらの積項で最小被覆の解となるものはごく一部であり，他の大部分は解より多くの個数のPI（あるいは素数）の積からなる項である。そこで，この不用な項を見出して捨て去れば，配列が節約でき，論理展開式の処理時間も短縮できるものと考えられる。このための一手法として，あらかじめ近似被覆解を求め，積項の中でPIの数がこれを上まわるものがあらわれると，それを取り除くよう操作する。この方法を取り入れることによって，今まで一度に処理不可能であった5変数で最大のカバー表（26行×26列）となる図3の関数も計算が可能となり，5変数以下の関数なら，一応すべて解けるようになった。

表2　グループ化の例

素数	PI	
p_1	A	
p_2	B	グループ I
p_3	C	
p_4	D	
p_1	E	
p_2	F	グループ II
p_3	G	

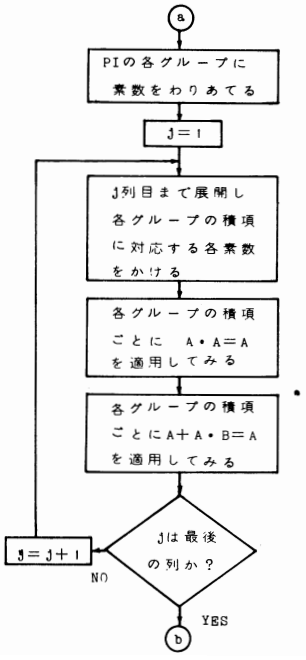
実際の計算では，この近似被覆解が本当の最小被覆解の1つとなっていなければ，この方法を取り入れた利点がほとんど得られない。例えば最小解より1個だけ多くのPIを含む近似解を条件として，それより多いPIをもつ項を取り除いていっても，上の26行×26列のカバー表では積項の数が2000を越えて配列に不足が生じて計算が不可能となった。

以上，グループ化の方法と近似被覆解の2つの技法を用いると，26行×26列のカバー表まで一度に処理できる。

6変数以上の関数になると，カバー表は更に大きくなることがあるが，この場合は行の数（PIの数）に制限をもうけ，これを越える大きさのカバー表はこれより小さないくつかのカバー表に分けて処理

図2

図1の流れ図のⓐ—ⓑ間の詳細図



する方法をとる。この操作を分枝するという。

図3　5変数で最大のカバー表となる関数

X_4	X_5	X_1=0 0 0 0 1 1 1 1	X_2=0 0 1 1 1 1 0 0	X_3=0 1 1 0 0 1 1 0
0	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1	1	0
1	1	0	1	1	1	1	0	1	1
1	0	1	1	1	0	1	1	0	1

この分枝の手法は次の通りである。すなわち、ある最小項をカバーするPIの中から，任意に2つq_1，q_2を選ぶ。このとき，最小被覆解に

(1)　q_1だけが入る

(2)　q_2だけが入る

(3)　q_1とq_2の双方が入る

(4)　q_1，q_2ともに入らない

の4つの場合が考えられる。それぞれの場合について条件を仮定すると，はじめのカバー表より，小さなカバー表を得る。これらの縮小されたカバー表がそれぞれ，指定された数より小さな行数をもてば，計算を行ない，制限以上であれば，さらに分枝の段数をふやして計算を行なう。図1の流れ図のループ2の部分がこの手順に対応し，行の数が制限内でなければ，こちらへ移る。又，論理展開式の項数がそのために用意した配列の大きさを上まわる場合も分枝のループへとび，カバー表をより小さくして処理を行なうようになっている。

なお，ある最小項をカバーするPIがq_1とq_2の2つだけなら，(4)の場合を考えなくてよく，分枝の枝の数が少なくなって好都合である。一回の分枝では，大体2ないし4行，4ないし6列カバー表が小さくなり，もしこれにEPIの抽出などの縮約技法がつかわれれば，一挙に10行，10列位小さくできることもある。

2．実際の計算例

表3，表4は実際にプログラムを組み計算してみた結果である。

表3は5変数関数の2つの例である。カバー表を分枝せず，一度に処理した場合が最も計算時間が長い。これは論理展開式の積項の数が非常に多くなることと，PIのグループ化のための手数の増加によるものである。特に論理展開式の処理時間が計算時間の大部分を占める。論理展開式の積項について調べてみると，PIの個数（カバー表の行の数）の制限を20とすると，

26行×26列の表で　　最大約 140

23行×25列の表で　　最大約 130

であったものが，制限なしでは

26行×26列の表で　　最大約1500

23行×25列の表で　　最大約 500

表３　５変数関数の例

PIの数	最小項の数	PIの制限	分枝数	計算回数	解の数	計算時間 M　S
23	25	なし	なし	1	25	1 － 33
		22	1	3	25	1 － 15
		20	2	5	21	19
		18	2	7	21	13
		16	3	9	12	10
		14	3	11	12	9
		12	3	15	7	9
		10	3	15	7	9
26	26	なし	なし	1	6	1 － 34
		24	1	3	4	44
		22	1	3	4	44
		20	2	7	4	19
		18	2	9	4	17
		16	2	9	4	17
		14	3	15	3	15
		12	2	23	2	15
		10	2	41	2	16

表４　６変数関数の例

PIの数	最小項の数	PIの制限	分枝数	計算回数	解の数	計算時間 M　S
48	50	24	3	23	53	8 － 54
		22	4	33	46	5 － 33
		20	5	49	46	4 － 7
		18	5	69	37	2 － 37
		16	6	113	37	2 － 28
		14	7	187	37	2 － 3
		12	7	217	37	2 － 3
		10	7	255	5	2 － 1

と著しく増加する。

分枝して計算を行なうと，計算回数はふえるが，各計算ごとの論理展開式の積項の数が小さいので，全体として計算時間が短縮されることになる。計算時間の短縮は分枝数の大きいもの程大きい。表で分枝数と計算回数が必ずしも一致しないのは，分枝のときの状態により，分枝数が５のものでも４番目の分枝で，あるいは３番目の分枝で計算が終了しているためであり，表の分枝数は最も分枝の段数の大きな数である。解の数は分枝なしで一番多く，分枝数の増加と共に次第に少なくなる。

表４の６変数関数の例でも同じことがいえる。但し，この場合カバー表が大きくなるので，一度に

処理することはできない。

PIの制限数を小さくすると，それだけ分枝の段数もふえ計算時間が短かくなることがわかったが，必要以上小さく（例えば18）としても，今度は分枝の数がふえてきて，計算回数の増加が論理展開式の処理数の減少の効果を打消すようになり，計算時間の改善が見られなくなる。

カバー表を一度に処理する方法は計算時間で不利であるが，最小被覆解がすべて求まるという利点がある。一方，分枝して処理する方法は計算時間が早いという特長をもつが，分枝ごとに一部の解が失われて，代表的な解しか求まらない。しかし，実際上，代表的な解が得られればそれで十分であるという場合が多い。

次に，表5は以上述べた素数表現法と0－1計画法（バラスの加法アルゴリズム）で解いた5変数の最小カバー問題の5つの例の計算時間を示したものである。特に例5は図3の関数である。カバー表は26行×26列になるが，本方法で15秒で解けたのに対し，0－1計画法では1時間18分も要した。

このことから本方法の特長は計算時間が短かくてすむこと，及び，分枝しないで計算した場合には最小被覆解がすべて求まることがあげられよう。

表5　計算時間の比較

	PIの数	最小項の数	計算時間	
			素数法	0－1計画法
例1	10	22	50 MS	781 MS
例2	11	22	43 MS	1478 MS
例3	18	22	354 MS	19 S
例4	23	25	9 S	14 M 28 S
例5	26	26	15 S	1 H 18 M

3．結　　言

素数表現法では一度に処理できるカバー表の大きさは本学の計算機で26行×26列であるが，分枝の手法をとり入れると，更に大きなものが取扱える。しかし，分枝段数が大きくなり過ぎると，計算回数が分枝数のべき乗でふえてくるので，計算時間が大きくなり，結局やはり制限がある。およそのところ，PIの制限数を18としたときで，分枝段数が7ないし8といったところが実用限度であろうと思われる。これはカバー表の大きさでいって，50行×50列位，関数なら6変数の論理関数まで設計できる。

参考文献

1）宮腰　隆，松田　秀雄；電気四学会北陸支部連合大会（1978．10）
2）黒田，豊田他訳；整数計画法入門，培風館（1976）
3）稲垣康善，福村晃夫；電子通信学会誌，**50**，6，P．1005（1967）
4）足立暁生；論理設計の基礎，東海大学出版会（1973）

昭和52，53年　　電気四学会北陸支部連合大会で一部発表

Computer Algorithm for Minimal Cover Problems by Representation of Prime Implicant with Prime Number

Hideo MATSUDA, Kuniyuki HONJO, and Takashi MIYAGOSHI

In this paper it is shown that, if any prime implicant on cover tables is represented by its "specified prime number" on the digital computer, a solution to a minimal cover problem can be very simply determined in a numerical procedure.

The algorithm is suitable for the computer program, and it is efficient for the computing time in comparison with the other known procedures.

In our computer, this technique is applied to logical functions of less than six variables.

(1978年10月24日受理)