報告番号 甲第 1241 号

# 整数計画問題の組み合せ論的解法に関する研究

名古屋大学
今　井　正　治

図・本館

# 整数計画問題の組み合せ論的解法に関する研究

名古屋大学図書
和 767445

名古屋大学
今井正治

# 目　　次

はじめに …… 1

第1章　整数計画法とその問題点 …… 3

1.1　まえがき …… 3

1.2　整数計画法 …… 3

1.3　整数計画問題の解法とその問題点 …… 4

1.4　Branch－and－Bound 法の問題点 …… 5

1.5　本論文の概要 …… 6

第2章　目的関数と制約条件式の特徴を利用した擬似ブール計画法のアルゴリズム …… 9

2.1　まえがき …… 9

2.2　擬似ブール計画問題の定式化 …… 9

2.3　目的関数の下界および上界の評価 …… 11

2.4　問題の分割 …… 15

2.5　部分問題の縮小 …… 18

2.6　擬似ブール計画法のアルゴリズム …… 20

2.7　実験およびその結果 …… 26

2.7.1　アルゴリズムの実現 …… 26

2.7.2　問題の作製方法 …… 26

2.7.3　実験1 …… 27

2.7.4　実験2 …… 30

2.8　むすび …… 32

第3章　ハッシングの技法を用いた数式処理アルゴリズムとその擬似ブール計画法への応用 …… 35

3.1　まえがき …… 35

3.2　ブール関数，擬似ブール関数およびこれらの内部表現 …… 36

3.2.1　ブール関数と擬似ブール関数 …… 36

3.2.2　数式の内部表現 …… 38
3.2.3　数式の処理 …… 39
3.3　数式処理アルゴリズム …… 41
3.3.1　擬似ブール関数の処理アルゴリズム …… 41
3.3.2　ブール関数の処理アルゴリズム …… 43
3.4　アルゴリズムの計算時間の理論的解析 …… 47
3.4.1　アルゴリズム3.1，3.2の計算時間 …… 47
3.4.2　アルゴリズム3.1，3.2の計算時間の上限 …… 48
3.4.3　アルゴリズム3.4，3.5の計算時間 …… 50
3.5　実験およびその結果 …… 51
3.5.1　実験1 …… 51
3.5.2　実験2 …… 53
3.6　むすび …… 55

第4章　擬似ブール計画法のアルゴリズムにおける
評価関数の強力さとアルゴリズムの計算時間の関係 …… 57
4.1　まえがき …… 57
4.2　擬似ブール計画問題の定式化 …… 58
4.3　含意の性質を用いた目的関数の下界の評価 …… 59
4.4　問題の分割方法 …… 62
4.5　擬似ブール計画法のアルゴリズム …… 65
4.6　評価関数の強力さと全体の計算時間に関する考察 …… 67
4.6.1　目的関数の下界の評価方法とアルゴリズムの計算時間 …… 67
4.6.2　分割変数の選択方法とアルゴリズムの計算時間 …… 69
4.7　実験およびその結果 …… 70
4.7.1　アルゴリズムの実現と問題の作製方法 …… 70
4.7.2　実験1 …… 71
4.7.3　実験2 …… 75
4.8　むすび …… 76

第5章　最小被覆問題の擬似ブール計画法による解法 …… 79

5.1 まえがき …… 79
5.2 最小被覆問題の擬似ブール計画法による定式化 …… 79
5.3 目的関数の下界の評価 …… 80
5.4 問題の分割 …… 82
5.5 最小被覆問題のアルゴリズム …… 84
5.5.1 問題の縮小 …… 84
5.5.2 アルゴリズム …… 85
5.5.3 例題とその解法 …… 86
5.6 実験およびその結果 …… 89
5.6.1 アルゴリズムの実現と問題の作製方法 …… 90
5.6.2 実験1 …… 90
5.6.3 実験2 …… 91
5.6.4 実験3 …… 94
5.7 むすび …… 96

第6章 Branch-and-Bound アルゴリズムの並列化とその評価 …… 97
6.1 まえがき …… 97
6.2 マルチプロセッサ・システムのモデル …… 98
6.3 Branch-and-Bound アルゴリズムの並列化 …… 99
6.3.1 諸定義および記法 …… 99
6.3.2 アルゴリズムの並列化の方針 …… 100
6.3.3 並列形 Branch-and-Bound アルゴリズム …… 101
6.3.4 例題とその解法 …… 103
6.4 PDFAの性質 …… 104
6.4.1 PDFAの実現に必要な記憶容量 …… 104
6.4.2 PDFAの計算時間 …… 105
6.5 シミュレーションとその結果 …… 106
6.5.1 実験1 …… 107
6.5.1.1 問題の発生方法 …… 107
6.5.1.2 実験方法および結果 …… 108
6.5.2 実験2 …… 111

6.5.2.1 問題の発生方法 …… 111
6.5.2.2 実験方法および結果 …… 112
6.6 むすび …… 113

第7章 必要な記憶容量を自由に制御できる
Branch-and-Bound 法のサーチ・ストラテジ …… 115
7.1 まえがき …… 115
7.2 諸定義および記法 …… 116
7.3 Branch-and-Bound アルゴリズム …… 118
7.4 従来のサーチ関数の性質とその問題点 …… 121
7.4.1 アルゴリズムの計算時間 …… 121
7.4.2 必要な記憶容量 …… 121
7.5 必要な記憶容量を自由に制御できるサーチ関数 …… 123
7.5.1 サーチ関数 pdfs …… 123
7.5.2 サーチ関数 slim …… 124
7.6 サーチ関数 pdfs および slim の性質 …… 125
7.6.1 pdfs の性質 …… 125
7.6.2 slim の性質 …… 126
7.7 実験および結果 …… 127
7.7.1 問題の作製方法 …… 127
7.7.2 実験方法および結果 …… 128
7.7.3 考　察 …… 131
7.8 むすび …… 132

第8章 あとがき …… 133

謝　辞 …… 137

参考文献 …… 139
第1章の参考文献 …… 139
第2章の参考文献 …… 139

第 3 章の参考文献 …… 140
第 4 章の参考文献 …… 140
第 5 章の参考文献 …… 141
第 6 章の参考文献 …… 141
第 7 章の参考文献 …… 142
第 8 章の参考文献 …… 143

付　　録 …… 145
付録 1（第 2 章の付録） …… 145
(1) 定理 2. 1 の証明について …… 145
(2) 定理 2. 2 の証明について …… 147
付録 2（第 3 章の付録） …… 150
(1) 式（ 3. 2 0 ）の導出について …… 150
(2) 式（ 3. 2 4 ）の導出について …… 150
(3) 式（ 3. 2 1 ），式（ 3. 2 2 ）の導出について …… 151
(4) 式（ 3. 2 5 ），式（ 3. 2 6 ）の導出について …… 151
付録 3（第 4 章の付録） …… 152
積項間の含意の関係の判定について …… 152
付録 4（第 5 章の付録） …… 153
(1) 定理 5. 1 の証明について …… 153
(2) 定理 5. 2 の証明について …… 154
付録 5（第 6 章の付録） …… 156
(1) 定理 6. 1 の証明について …… 156
(2) $Z_{(p+1)} < Z_p$ であるための必要十分条件について …… 156
付録 6（第 7 章の付録） …… 159
(1) 定理 7. 1 の証明について …… 159
(2) 定理 7. 3 の証明について …… 160
(3) 定理 7. 4 の証明について …… 161

# はじめに

本論文は，筆者が名古屋大学大学院工学研究科博士課程（前期および後期）において行なった，整数計画問題の組み合せ論的解法に関する研究の結果をまとめたものである．

整数計画法は，組み合せ最適化問題に対する一般的定式化の一つであり，その汎用性から考えて，整数計画問題の効率的解法を明らかにすることは，実用的見地からも大きな意義があると考えられる．整数計画問題の解法に関しては，古くからさまざまな角度より研究が行なわれているが，いまだに未解決の問題も多い．

本論文が整数計画問題の解法に関する研究の一助となれば幸いである．

# 第1章　整数計画法とその問題点

## 1.1　まえがき

1940年代の半ばに，最初の電子式計算機ENIACが，Eckert および Mauchlyによって開発されて以来，この30年余りの間に，電子計算機のハードウェア技術およびソフトウェア技術は長足の進歩をとげてきた．そして，それに伴なって，人間が紙と鉛筆だけを用いて行なった場合には物理的に不可能とさえ思われるような，大量の情報の複雑な処理が可能となってきた．

その結果，我々の生活を支える各種の生産活動も，ますます大規模かつ複雑なものとなりつつあり，したがって，これらの生産活動を行なうシステムの最適化の問題の重要さもますます増しつつある．

これらのシステムには，連続系と離散系とが存在する．線形連続系の最適化問題の代表的な解法である線形計画法については，すでに多くの研究がなされ，非常に大規模な問題が妥当な時間の範囲内で解かれている．一方，離散系の最適化問題の一般的定式化である整数計画法では，組み合せ的問題を対象としているために，大規模な問題を解くためには，最新の高速計算機を用いたとしてもなお，かなり多くの計算時間を必要とする．

しかし，上で述べた最適化問題の中で整数計画問題の占める割り合いが決して少なくないこと，および，整数計画法の汎用性を考慮すると，整数計画問題に対する効率的解法を明らかにすることは，理論的興味のみならず，実用的見地からも，非常に大きな意味があると考えられる．

以下，本章では，初めに，1.2で整数計画問題の一般的定式化とその中で0－1計画問題の占める位置について述べる．次に，1.3では，整数計画問題の解法とその問題点，および，Branch-and-Bound法の重要さについて述べる．1.4では，Branch-and-Bound法にもとづく解法の構成上の問題点について述べる．最後に1.5では，本論文の各章で取り扱かうことがらについて概観する．

## 1.2　整数計画法

整数計画問題は，一般に，次のように定式化される．

〔整数計画問題〕

$x_1$，$x_2$，…，$x_n$ を，整数値をとる変数とする．このとき，制約条件 $F(x_1, x_2, \cdots, x_n)$ を真とする $x_1, x_2, \cdots, x_n$ の値の組合せのうち，目的関数 $y(x_1, x_2, \cdots, x_n)$ を最小（または最大）にするものを求めよ．

このような形で定式化される組み合せ最適化問題は非常に多く知られており，これらの全てを示すのは困難であるが，代表的な例としては，たとえば，乗務員のスケジューリングの問題，ジョブ・ショップのスケジューリングの問題，巡回セールスマンの問題，目的関数が2次の割り当て問題，ナップザック問題，固定費つきの輸送問題，非線形関数の近似の問題，データベースの設計の問題などがある.[3, 5-9]

これらの整数計画問題のかなりの部分は，変数が0または1の値しかとらない0－1計画問題である．また，0－1計画問題以外の整数計画問題であっても，適当な変換をほどこすことにより，0－1計画問題に帰着できる問題がかなり多く存在する．以上の理由により，本論文では，主として0－1計画問題を扱うことにする．

### 1．3　整数計画問題の解法とその問題点

上で述べた整数計画問題の一般的解法としては，主として，以下のような方法が用いられている．

(i)　線形計画法の修正による解法（たとえば，文献(4)）

(ii)　列挙法または動的計画法による解法（たとえば，文献(1)）

(iii)　Branch-and-Bound法による解法（たとえば，文献(8)）

これらの解法のうち，(i)は，制約条件が線形不等式または等式として与えられ，かつ，目的関数も線形である場合には，かなり大規模な問題をも扱いうる．しかし，取り扱える問題の形式が限られていること，および，問題を解く過程で制約条件が順次増えてゆくことなどの理由から，この方法が有効とはいえない場合も多い．

次に，(ii)の解法は，取り扱える問題の形式がかなり自由であるという特徴を持つが，いわゆるしらみつぶし的解法であるため，計算時間および記憶容量の点で，大規模な問題の解法としては適していないと考えられる．

さいごに，(iii)の方法は，(ii)の方法と同様に，取り扱える問題の形式がかなり自由である．また，この方法では，目的関数の下界（または上界）を与える評価関数，解空間を分割する手続き，および，次に処理すべき部分問題を選択する手続きを巧みにえらべば，(ii)の方法よりも，かなり強力な方法となる．

さらに，この方法は，アルゴリズムの実行途中で生成された未解決な部分問題を並列に処理できることなどの理由から，最近各所で開発されている，マルチプロセッサ・システムに非常に適した解法であると考えられる．

また，Branch-and-Bound法での，解空間の探索過程が，人工知能の問題での状態空間の

探索過程とかなり類似していることから，(iii)の解法に関する研究は，人工知能の問題に対して寄与するところも大きいと考えられる．

以上の観点から，本論文では，整数計画問題の解法として，Branch-and-Bound法にもとづくアルゴリズムをとりあげて論ずることにする．

## 1．4　Branch－and－Bound法の問題点

Branch-and-Bound法にもとづく整数計画法のアルゴリズムを構成する場合，以下の3つの問題に対する方策がアルゴリズムの効率に大きな影響を及ぼす．

(i)解空間の分割方法：k－値をとる$n$個の変数によって記述される最小化問題を考える．このとき，各変数$x_1$，$x_2$，…，$x_n$に与えうる値の組合せは，$k^n$ 通り存在する．本論文で述べるBranch-and-Boundアルゴリズムでは，与えられた問題に含まれる変数の中から，変数を1つえらび，この変数に値を与えることによって，与えられた問題をk個の部分問題に分割する．この処理は，$k^n$ 個の要素からなる解空間をk個の部分空間に分割することを意味する．ここに，各部分空間は，与えられた問題を分割して得られるk個の部分問題のいずれか1つに1対1に対応する．問題の分割を行なうために値が与えられる変数を，以下では分割変数と呼ぶことにする．

与えられた問題に含まれる$n$個の変数の中から，どの変数を分割変数として選択するかによって，アルゴリズム中で処理される部分問題の個数は大きく変化すると考えられる．

(ii)目的関数の下界の評価方法：Branch-and-Boundアルゴリズムでは，与えられた問題を分割して得られる各部分問題の最適解に対する目的関数の値の下界を評価する．そして，もしその下界が，アルゴリズムですでに得られている，制約条件をみたす可能解による目的関数の値よりも大きいことが知られれば，その部分問題は，与えられた問題の最適解を与えることはないので，その処理を中止することができる．

よって，各部分問題の目的関数に対する，より厳密な下界が得られれば，アルゴリズム中で処理される部分問題の個数が削減されることは明らかである．

(iii)部分問題の選択方法：Branch-and-Boundアルゴリズムの実行途中で得られる，未解決な部分問題の集合の中から，アルゴリズムで次に処理すべき部分問題を選択する手続きは，サーチ関数またはサーチ・ストラテジとよばれる．

Branch-and-Boundアルゴリズムでは，与えられた問題に対する最適解による目的関数の値に，より近い値を与える可能解が，より早くみつけられるほど，(ii)で述べた理由により，アルゴリズム中で処理される部分問題の個数は，より削減される．

よって，未解決な部分問題の集合中から，次に処理すべき部分問題を選択するサーチ関数のえらび方によって，アルゴリズム中で処理される部分問題の個数は，大きく変化しうる．

以上の3つの問題点に対する方策が，アルゴリズム中で処理される部分問題の個数に大きな影響を及ぼすが，アルゴリズムの実際の計算時間に影響を及ぼす他の要因は，もちろん，部分問題1個当りの処理に要する計算時間である．すなわち，計算機内部での，部分問題の表現方法，および，部分問題に対する，各種の処理手続きの効率が，アルゴリズムの実際の計算時間に及ぼす影響は大きい．

組み合せ最適化問題のBranch-and-Bound法による解法では，解かれる問題が大規模となるのにともなって，アルゴリズム中で処理される部分問題の個数が指数関数的に増加することはさけがたいことであるように思われる．よって，アルゴリズムの計算時間をさらに短縮するためには，新たな方法を導入する必要がある．

高い処理能力を持つ計算機システムを実現するための方法の一つとして，マルチプロセッサ・システムが提案されてきた．また，近年におけるＬＳＩ技術の発達にともない，いくつかのシステムがハードウェアとして実現されてきた．このような計算機システムの上で動作する並列形アルゴリズムの構成方法とその性質を明らかにすることは，単に，より大規模な組み合せ最適化問題を解くことを可能にするのみならず，将来の計算機システムの発展に対しても寄与するところは大きいと考えられる．

また，アルゴリズムの実現に必要な記憶容量も，アルゴリズムを計算機の上で実現する際の問題の1つである．一般に，取り扱う問題が大規模になればなるほど，計算機の内部には，より大量の情報が記憶されなければならない．

アルゴリズムの実現に必要な記憶容量は，問題を計算機内部で表現するために用いられる方法，および，アルゴリズムで用いられるサーチ関数の種類によって大きく影響される．

一方，近年におけるＬＳＩ技術の発展にともなって，大容量の記憶装置を用いることが可能になってきているとはいえ，アルゴリズムの実現に用いることができる記憶容量はやはり限られている．

よって，より大規模な問題を，より効率良く解くためには，問題を計算機内部で表現するためのデータ構造とその処理手続き，および，アルゴリズムの実現に必要な記憶容量を自由に制御しうるサーチ関数の開発の2つが必要不可欠な問題となる

## 1．5　本論文の概要

前節で述べた考察にもとづき，本論文の各章では，次のような問題を扱う．

第2章では，まず，0－1計画問題の一般的定式化の1つである，擬似ブール計画法について考え，制約条件式と目的関数の特徴を用いて目的関数の下界および上界を評価する方法を提案する．つぎに，目的関数の下界および上界を用いて分割変数を決定する方法について述べる．さらに，これらの評価関数を用いた擬似ブール計画法のアルゴリズムを与え，このアルゴリズムの効率を実験的に調べた結果について述べる．

第3章では，まず，擬似ブール計画問題の記述に用いられる，擬似ブール関数およびブール関数の計算機内部での効率的な表現方法について述べる．つぎに，これらの関数を，ハッシングの技法を用いて効率良く処理する手続きを提案し，これらの手続きの実行に必要な計算時間を理論的に解析する．さらに，これらの処理手続きを用いて実現される擬似ブール計画法のアルゴリズムと，従来の数式処理方法による擬似ブール計画法のアルゴリズムとの計算時間の比較を行なった実験結果について述べる．

第4章では，まず，含意の関係を用いて目的関数の下界をより厳密に評価する方法，および，第2章で述べた問題の分割方法よりも融通性に富む，分割変数の決定方法を提案する．つぎに，アルゴリズム中で用いられる評価関数の強力さと評価関数の計算時間が，アルゴリズム全体の計算時間に与える影響について考察する．さらに，上で述べた2つの評価関数の有効さを調べるための実験を行ない，上記の考察の結果と比較する．

第5章では，0－1計画問題の中で，効率的な解法が最も得られにくい問題の1つとして知られている，最小被覆問題について考察する．まず，制約条件の性質を用いて目的関数の下界を評価する方法を示し，つぎに，制約条件の特徴を用いて分割変数の選択を行う手続きを示す．さらに，これらの評価関数に関するいくつかの性質を明らかにし，最小被覆問題を解くアルゴリズムを与える．また，このアルゴリズムの計算時間を，従来の解法の1つである阿部のアルゴリズムの計算時間と比較する．

第6章では，マルチプロセッサ・システムの上で動作する並列形Branch-and-Boundアルゴリズム PDFAを提案し，PDFAの実現に必要な記憶容量とその計算時間に関する，いくつかの性質を明らかにする．さらに，PDFAの効率を調べるためのシミュレーションの結果について述べる．

第7章では，アルゴリズムの実現に必要な記憶容量を自由に制御できる，Branch-and-Bound法のサーチ関数を2つ提案する．さらに，これらのサーチ関数を用いるアルゴリズムの計算時間について考察し，実験結果と比較する．

最後に，第8章では，残された問題点と，今後の研究の方向について述べる．

# 第２章　目的関数と制約条件式の特徴を利用した擬似ブール計画法のアルゴリズム

## 2.1　まえがき

第１章でも述べたように，Branch-and-Bound 法にもとづく整数計画法のアルゴリズムを設計する場合，次の３つの点がアルゴリズムの効率を左右する重要な問題である：

(i)　可能解の集合の分割方法

(ii)　可能解の部分集合に対する目的関数の下界の評価方法

(iii)　次に解くべき部分問題の選択方法

本章では，上記の３つの問題点のうち，主として，(i)および(ii)の点について考察し，これらの問題点に対する方策を明らかにする．さらに，これらの方策にもとづいて，擬似ブール計画法のアルゴリズムを構成する．このアルゴリズムは，以下の章での議論の基礎を与える．

以下，本章では，初めに 2.2 で，以下で扱う，擬似ブール計画問題の定式化を行なう．つぎに 2.3 では，目的関数の下界および上界を，目的関数および制約条件式の特徴を用いて評価する方法について述べる．2.4 では与えられた問題をその部分問題に分割する方法について述べる．2.5 では，制約条件式および目的関数の性質を用いて問題に含まれる変数の個数を削減する方法について述べる．さらに，2.6 では，以上の考察にもとづいて，擬似ブール計画法のアルゴリズムを構成する．2.7 では，このアルゴリズムを計算機の上に実現し，実験を通じてその効率を評価した結果について述べる．

## 2.2　擬似ブール計画問題の定式化

この節では，以下で取り扱う擬似ブール計画問題を一般的に定式化する．

まず，変数の定義域を，つぎのように定める．

$$x_j \in \{0,\ 1\} \quad \text{for} \quad 1 \leq j \leq n \tag{2.1}$$

つぎに，$x_1$，$x_2$，…，$x_n$ を要素とする $n$ 次元ベクトル $\mathbf{x}$ をつぎのように表わす．

$$\mathbf{x} \triangleq (x_1,\ x_2,\ \cdots,\ x_n) \tag{2.2}$$

〔定義 2.1〕ブール関数 $F(\mathbf{x})$ を次式で与える．

$$F(\mathbf{x}) \triangleq \bigvee_{i=1}^{s} F_i(\mathbf{x})$$
$$F_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\varepsilon_{ij}},\ (i=1,\ 2,\ \cdots,\ s) \tag{2.3}$$

ここに，$\vee$および$\Pi$はそれぞれ論理和および論理積を表わす．また，$x_j^{\varepsilon_{ij}}$は，$\varepsilon_{ij}\in\{0, 1, u\}$として，次式で与えられる．

$$x_j^{\varepsilon_{ij}} \triangleq \begin{cases} \bar{x}_j & \text{if} \quad \varepsilon_{ij}=0 \\ x_j & \text{if} \quad \varepsilon_{ij}=1 \\ 1 & \text{if} \quad \varepsilon_{ij}=u \end{cases} \tag{2.4}$$

ここに，$\bar{x}_j$ は変数 $x_j$ の否定を表わす．

〔定義 2.2〕擬似ブール関数 $y(\mathbf{x})$ を次式で定義する．

$$\begin{cases} y(\mathbf{x}) \triangleq \sum_{i=1}^{m} a_i Y_i(\mathbf{x}) \\ Y_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\delta_{ij}}, \quad (i=1, 2, \cdots, m) \end{cases} \tag{2.5}$$

ここに，$a_i$ は実数とする．また，$x_j$ は，$\delta_{ij}\in\{0, 1\}$として，次式で与えられる．

$$x_j^{\delta_{ij}} \triangleq \begin{cases} 1 & \text{if} \ \delta_{ij}=0 \\ x_j & \text{if} \ \delta_{ij}=1 \end{cases} \tag{2.6}$$

〔定義 2.3〕擬似ブール計画問題 P（$y$，F）は，式（2.3），式（2.5）の $F(\mathbf{x})$ および $y(\mathbf{x})$ を用いて，次のように与えられる．

制約条件 $F(\mathbf{x})=0$ のもとで，$y(\mathbf{x})$ を最小にする $\mathbf{x}$ を求めよ．

与えられた問題が最大化問題の場合には，この問題を $-y(\mathbf{x})$ の最小化問題として定式化しなおすことができる．また，全てのブール関数は，式（2.3）の積和形のブール関数として表現できる．さらに，$y(\mathbf{x})$ を構成する項 $Y_i(\mathbf{x})$ が式（2.5）の形をしていない場合には，次式に示すようなブール式の代数式への変換公式を用いて式（2.5）の形を得ることができる．

$$x \vee y = x + y - x \cdot y \tag{2.7}$$

$$\bar{x} = 1 - x \tag{2.8}$$

よって，定義 2.3 の定式化は一般性を失なわない．

定義 2.3 で定式化される擬似ブール計画問題では，制約条件 $F(\mathbf{x})=0$ を満足する可能解の集合の要素の個数は $O(2^n)$ となりうる．この種の問題を Branch-and-Bound 法によって解く場合，

(i) 可能解の集合の分割方法

(ii) 可能解の部分集合に対する目的関数の下界（上界）の評価方法

の 2 つは，アルゴリズムの効率に影響を及ぼす重要な要因である．次節以降では，これらの問題

について考察する．

## 2．3　目的関数の下界および上界の評価

この節では，目的関数 $y(\mathbf{x})$ および制約条件式 $F(\mathbf{x})$ の性質を用いて，$y(\mathbf{x})$ の上界および下界の値を評価する方法について述べる．

ブール制約条件 $F(\mathbf{x})=0$ を満たすベクトル $\mathbf{x}$ の集合を $S(F)$ で表わす．すなわち，

$$S(F) \triangleq \{ \mathbf{x} \mid \mathbf{x} \in \{0,1\} \ , \ F(\mathbf{x})=0 \} \tag{2.9}$$

制約条件 $F(\mathbf{x})=0$ のもとでの，$y(\mathbf{x})$ の最小値および最大値を与える関数 $\inf(y, F)$ および $\sup(y, F)$ を次のように定義する．ただし，以下では，$y(\mathbf{x})$，$F(\mathbf{x})$ に含まれる変数 $x_j$ に値 $v \in \{0,1\}$ を与えて得られる式をそれぞれ $y(x_j=v)$，$F(x_j=v)$ で表わす．すなわち，

$$y(x_j=v) \triangleq y(x_1, \cdots, x_{j-1}, v, x_{j+1}, \cdots, x_n) \tag{2.10}$$

$$F(x_j=v) \triangleq F(x_1, \cdots, x_{j-1}, v, x_{j+1}, \cdots, x_n) \tag{2.11}$$

〔定義 2.4〕

$$\inf(y, F) \triangleq \begin{cases} y(\mathbf{x}) \ ; & y(\mathbf{x}) \text{ が定数関数のとき} \\ \min\limits_{1 \le j \le n} \left[ \min\limits_{v \in \{0,1\}} \inf(y(x_j=v), 0) \right]; & F(\mathbf{x}) \equiv 0 \text{ のとき} \\ \max\limits_{1 \le i \le s} \left[ \min\limits_{j \in I_i} \inf(y(x_j=\overline{\varepsilon}_{ij}), F(x_j=\overline{\varepsilon}_{ij})) \right]; & \text{それ以外のとき} \end{cases} \tag{2.12}$$

$$\sup(y, F) \triangleq \begin{cases} y(\mathbf{x}) \ ; & y(\mathbf{x}) \text{ が定数関数のとき} \\ \max\limits_{1 \le j \le n} \left[ \max\limits_{v \in \{0,1\}} \sup(y(x_j=v), 0) \right]; & F(\mathbf{x}) \equiv 0 \text{ のとき} \\ \min\limits_{1 \le i \le s} \left[ \max\limits_{j \in I_i} \sup(y(x_j=\overline{\varepsilon}_{ij}), F(x_j=\overline{\varepsilon}_{ij})) \right]; & \text{それ以外のとき} \end{cases} \tag{2.13}$$

ここに，$I_i$ は次式で定義される集合である．すなわち，$F(\mathbf{x})$ が式（2.3）で与えられるとき，

$$I_i = \{ j \mid 1 \le j \le n, \ \varepsilon_{ij}=0 \text{ or } \varepsilon_{ij}=1 \} \tag{2.14}$$

また，$\overline{\varepsilon}_{ij}$ は次式で与えられる．

$$\overline{\varepsilon}_{ij} \triangleq \begin{cases} 1 & \text{if} \quad \varepsilon_{ij}=0 \\ 0 & \text{if} \quad \varepsilon_{ij}=1 \end{cases} \tag{2.15}$$

関数 $\inf(y, F)$，$\sup(y, F)$ に関して，次の定理が成りたつ．

〔定理 2.1〕

$$\inf(y, F) \leq \min\{y(x) \mid x \in S(F)\} \tag{2.16}$$

$$\sup(y, F) \geq \max\{y(x) \mid x \in S(F)\} \tag{2.17}$$

〔証明〕 定理 2.1 の証明は付録 1 (1)に示す.

関数 inf( $y$, F), sup( $y$, F) はそれぞれ, 制約条件 F( x ) = 0 のもとでの $y$ ( x ) の値の下界と上界とを与えている. ところが, これらの関数の値を計算するためには, 全ての可能解に対する $y$ ( x ) の値を知らなければならないことが式 ( 2.12 ), および式 ( 2.13 ) より容易に知られる. よって, これらの関数を $y$ ( x ) の下界および上界の評価に用いることは実際的とは言えない. そこで, 次のように, 帰納的演算の深さを $k$ 段以下に制限した関数 $\ell b_k(y, F)$ および $ub_k(y, F)$ を与える.

〔定義 2.5〕 $y$ ( x ) が次式の形をしていると仮定する.

$$y(x) \triangleq a_0 + \sum_{i=1}^{m} a_i Y_i(x) \tag{2.18}$$

ここに, $a_0$ は $y$ ( x ) の定数項を表わす. また, F ( x ) は式 ( 2.3 ) で与えられるとする. このとき, $\ell b_0(y, F)$, $ub_0(y, F)$ を, それぞれ次式で定義する.

$$\ell b_0(y, F) \triangleq a_0 + \sum_{i=1}^{m} \min(a_i, 0) \tag{2.19}$$

$$ub_0(y, F) \triangleq a_0 + \sum_{i=1}^{m} \max(a_i, 0) \tag{2.20}$$

$k \geq 1$ に対する $\ell b_k(y, F)$ および $ub_k(y, k)$ はそれぞれ次のように定義される.

$$\ell b_k(y, F) \triangleq \begin{cases} y(x) \quad ; \quad y(x) \text{ が定数関数のとき} \\ \min\limits_{1 \leq j \leq n} \left[ \min\limits_{v \in \{0,1\}} \ell b_{(k-1)}(y(x_j = v), 0) \right]; F(x) \equiv 0 \text{ のとき} \\ \max\limits_{1 \leq i \leq s} \left[ \min\limits_{j \in I_i} \ell b_{(k-1)}(y(x_j = \bar{\varepsilon}_{ij}), F(x_j = \bar{\varepsilon}_{ij})) \right]; \\ \qquad \text{それ以外のとき} \end{cases} \tag{2.21}$$

$$ub_k(y, F) \triangleq \begin{cases} y(x) \quad ; \quad y(x) \text{ が定数関数のとき} \\ \max\limits_{1 \leq j \leq n} \left[ \max\limits_{v \in \{0,1\}} ub_{(k-1)}(y(x_j = v), 0) \right]; F(x) \equiv 0 \text{ のとき} \\ \min\limits_{1 \leq i \leq s} \left[ \max\limits_{j \in I_i} ub_{(k-1)}(y(x_j = \bar{\varepsilon}_{ij}), F(x_j = \bar{\varepsilon}_{ij})) \right]; \\ \qquad \text{それ以外のとき} \end{cases} \tag{2.22}$$

関数 $\ell b_k$, $ub_k$ は，それぞれ制約条件 $F(\mathbf{x})=0$ のもとでの $y(\mathbf{x})$ の下界および上界となっており，しかも $k$ の値を増すことによって，それぞれ $\inf(y, F)$ および $\sup(y, F)$ の値に単調に収束する．すなわち，次の定理が成り立つ．

〔定理 2.2〕

(i) $\forall_k \geq 0,\ \ell b_{k+1}(y, F) \geq \ell b_k(y, F)$ (2.23)

$\forall_k \geq 0,\ ub_{k+1}(y, F) \leq ub_k(y, F)$ (2.24)

(ii) $\forall_k \geq n,\ \ell b_k(y, F) = \inf(y, F)$ (2.25)

$\forall_k \geq n,\ ub_k(y, F) = \sup(y, F)$ (2.26)

〔証明〕 定理 2.2 の証明は付録 1 (2)に示す．

次に，例題を示し，この例題に対して，$\ell b_k(y, F)$ の値を求める．

〔例題 2.1〕

制約条件

$$F(\mathbf{x}) = \bar{x}_1 \bar{x}_2 \vee \bar{x}_2 \bar{x}_3 \vee \bar{x}_3 \bar{x}_1 = 0 \tag{2.27}$$

のもとで，目的関数

$$y(\mathbf{x}) = -5x_1 - 7x_2 - 9x_3 + x_1 x_2 + 6x_2 x_3 + 4x_3 x_1$$

を最小化する $\mathbf{x}$ を求めよ．

例題 2.1 の $F(\mathbf{x})$，$y(\mathbf{x})$ に対する，$\ell b_k(y, F)$，$(k=0, 1, 2, 3)$ の値は，それぞれ次のようにして得られる．

(i) $\ell b_0(y, F)$：式 (2.19) より，

$$\ell b_0(y, F) = -5 - 7 - 9 = -21 \tag{2.29}$$

(ii) $\ell b_1(y, F)$：式 (2.21) より，

$$\begin{aligned}\ell b_1(y, F) &= \max(\min(\ell b_0(y(x_1=1), F(x_1=1)), \ell b_0(y(x_2=1), F(x_2=1))),\\ &\quad \min(\ell b_0(y(x_2=1), F(x_2=1)), \ell b_0(y(x_3=1), F(x_3=1))),\\ &\quad \min(\ell b_0(y(x_3=1), F(x_3=1)), \ell b_0(y(x_1=1), F(x_1=1)))) \end{aligned} \tag{2.30}$$

である．一方，

$$y(x_1=1)=-5-6x_2-5x_3+6x_2x_3 \tag{2.31}$$

$$y(x_2=1)=-7-4x_1-3x_3+4x_3x_1 \tag{2.32}$$

$$y(x_3=1)=-9-x_1-x_2+x_1x_2 \tag{2.33}$$

であるので，

$$\ell b_0(y(x_1=1),\ F(x_1=1))=-5-6-5=-16 \tag{2.34}$$

$$\ell b_0(y(x_2=1),\ F(x_2=1))=-7-4-3=-14 \tag{2.35}$$

$$\ell b_0(y(x_3=1),\ F(x_3=1))=-9-1-1=-11 \tag{2.36}$$

を得る．式（2.34）～式（2.36）を式（2.30）に代入することにより，次式を得る．

$$\begin{aligned}\ell b_1(y,\ F)&=\max(\min(-16,\ -14),\ \min(-14,\ -11),\ \min(-11,\ -16))\\&=\max(-16,\ -14,\ -16)=-14\end{aligned} \tag{2.37}$$

(iii) $\ell b_2(y,\ F)$：$\ell b_1(y,\ F)$ の場合と同様にして，

$$F(x_1=1)=\overline{x}_2\,\overline{x}_3 \tag{3.38}$$

$$F(x_2=1)=\overline{x}_3\,\overline{x}_1 \tag{3.39}$$

$$F(x_3=1)=\overline{x}_1\,\overline{x}_2 \tag{3.40}$$

であることから，次式を得る．

$$\begin{aligned}&\ell b_1(y(x_1=1),\ F(x_1=1))\\&=\min(\ell b_0(y(x_1=1,\ x_2=1),\ F(x_1=1,\ x_2=1))\\&\qquad\ell b_0(y(x_1=1,\ x_3=1),\ F(x_1=1,\ x_3=1)))\\&=\min(-11,\ -10)=-11\end{aligned} \tag{3.41}$$

$$\begin{aligned}&\ell b_1(y(x_2=1),\ F(x_2=1))\\&=\min(-11,\ -10)=-11\end{aligned} \tag{3.42}$$

$$\begin{aligned}&\ell b_1(y(x_3=1),\ F(x_3=1))\\&=\min(-10,\ -10)=-10\end{aligned} \tag{3.43}$$

よって，式（2.41）～式（3.43），式（2.21）より，

$$\begin{aligned}\ell b_2(y,\ F)&=\max(\min(-11,\ -11),\ \min(-11,\ -10),\ \min(-10,\ -11))\\&=\max(-11,\ -11,\ -11)=-11\end{aligned} \tag{3.44}$$

を得る．

(iv) $\ell b_3(y,\ F)$：$\ell b_1(y,\ F)$，$\ell b_2(y,\ F)$ の場合と同様にして，次式を得る．

$$\ell b_2(y(x_1=1),\ F(x_1=1))=-11 \tag{3.45}$$

$$\ell b_2(y(x_2=1),\ F(x_2=1))=-11 \tag{3.46}$$

$$\ell b_2(y(x_3=1),\ F(x_3=1))=-10 \tag{3.47}$$

よって，式（3.45）～式（3.47）および式（2.21）より次式を得る．

$$\ell b_3(y, F)$$
$$= \max(\min(-11, -11), \min(-11, -10), \min(-10, -11))$$
$$= \max(-11, -11, -11) = -11 \qquad (3.48)$$

例題2.1の最適解は，$(x_1, x_2, x_3) = (1, 1, 0)$で，このとき，$y(\mathbf{x})$の値は$-11$である．上記の計算により$\ell b_0(y, F) = -21$, $\ell b_1(y, F) = -14$, $\ell b_2(y, F) = -11$, $\ell b_3(y, F) = -11$であることが知られる．この例題で，$\ell b_k(y, F)$で$k$の値を増すことにより，より$\inf(y, F)$に近い，$y(\mathbf{x})$の下界が得られることが知られる．また，$\ell b_2(y, F) = \ell b_3(y, F) = \inf(y, F)$がなり立つことも知られる．

## 2.4 問題の分割

この節では，まず与えられた問題をその部分問題へ分割する方法が，アルゴリズム中で処理される部分問題の個数に対して及ぼす影響について述べる．次に，次節で述べる擬似ブール計画法のアルゴリズムで用いられる分割方法について述べる．

次節で述べる擬似ブール計画法のアルゴリズムでは，問題に含まれる変数に値を与えることによって，問題をその部分問題に分割する．以下では，分割を行なうために値を与えられる変数を分割変数と呼ぶことにする．問題に含まれる変数の中からどの変数を分割変数にえらぶかによって，アルゴリズム中で処理される部分問題の個数が大きな影響を受けると考えられる．たとえば，次の例題を考える．

〔例題2.2〕

制約条件，

$$F(\mathbf{x}) = \bar{x}_1 \bar{x}_2 \vee \bar{x}_2 \bar{x}_3 \vee \bar{x}_3 \bar{x}_1$$

のもとで，目的関数，

$$y(\mathbf{x}) = x_1 + 3x_2 + 5x_3$$

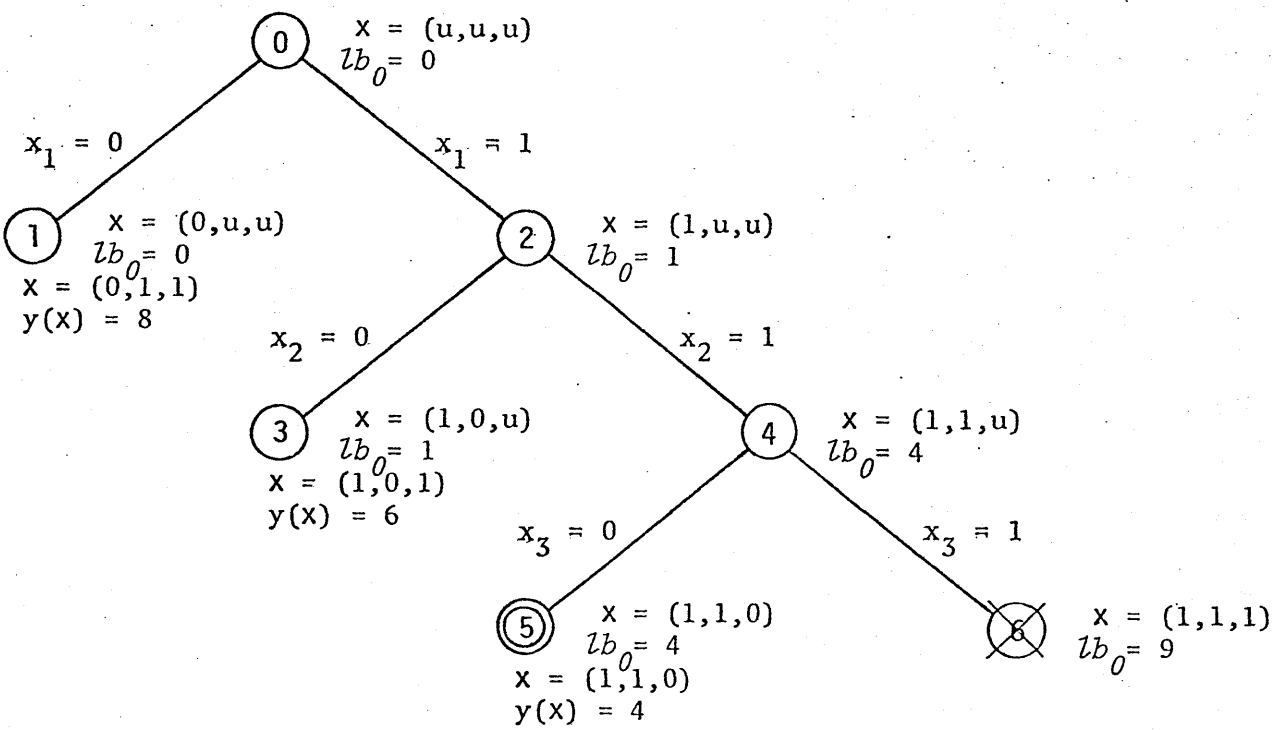
を最小化する$\mathbf{x}$を求めよ．

Branch-and-Bound法で問題が解かれる過程は木によって表わされる．ここに，木の節点には部分問題が1対1に対応し，木の枝は，部分問題の分割を表わす．上で述べた例題を，分割変数のえらび方を変えて，次節に示すアルゴリズムと同様のアルゴリズムで解いたときの木を図2.1に示す．同図で，分割変数を$x_1, x_2, x_3$の順にえらんだ場合，最適解を得るまでに6個

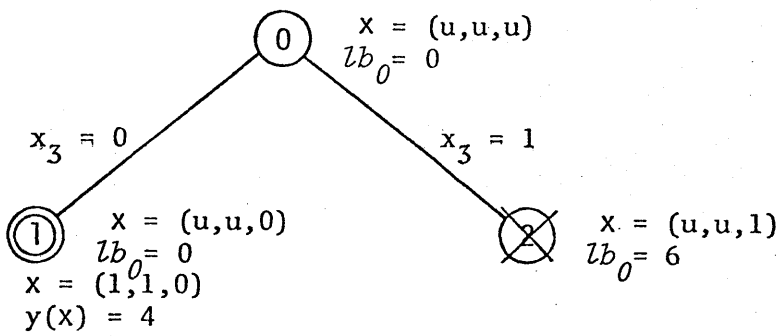
の部分問題が処理されることが知られる．一方，分割変数を，$x_3$，$x_2$，$x_1$ 又は，$x_3$，$x_1$，$x_2$ の順にえらぶことにより，わずか 2 個の部分問題だけを処理すればよいことが知られる．

このように，分割変数のえらび方によって，アルゴリズム中で処理される部分問題の個数は大きく変化する．以下では，いわゆるヒューリスティックを用いた，分割変数の選択方法について述べる．

与えられた問題が $n$ 個の変数を含む場合，分割変数の選び方は，$n$ 通りある．これらの $n$ 通りの選び方のうちで，最も効率的であると思われる方法は，分割変数として，次の条件をみた




(a) 分割変数を，$x_1$，$x_2$，$x_3$の順にえらんだ場合




(b) 分割変数を，$x_3$，$x_2$，$x_1$ 又は $x_3$，$x_1$，$x_2$ の順にえらんだ場合

図2.1　例題2.1に対する〝木〟

╳ は，枝刈りが行なわれたことを表わす

す変数 $x_j$ を選ぶことである。

$$\ell b_k(y(x_{j^*}=v^*),\ F(x_{j^*}=v^*))$$
$$\geq \ell b_k(y(x_j=v),\ F(x_j=v)),$$
$$\text{for } 1 \leq \forall j \leq n,\ \forall_v \in \{0,\ 1\} \qquad (2.51)$$

ここに，$v^* \in \{0, 1\}$，$1 \leq j^* \leq n$ とする。

式（2.51）の条件を満足するように $x_{j^*}$，$v^*$をえらぶとすると，部分問題，

$$P(y(x_{j^*}=v^*),\ F(x_{j^*}=v^*)) \qquad (2.52)$$

は，可能な $n$ 通りの分割の結果得られる，$2n$ 個の部分問題の中で最も早く処理が打ち切られる（枝刈りされる）部分問題であることは明らかである。

以上の考え方にもとづいて，まず，変数 $x_j$ の，分割変数としての適切さ，および不適切さを評価するための関数 $\mathrm{merit}_k$，および $\mathrm{demerit}_k$ を，それぞれ次のように定義する。

$$\mathrm{merit}_k(y, F, j)$$
$$\triangleq \max_{v \in \{0,1\}} (\ell b_k(y(x_j=v),\ F(x_j=v))) \qquad (2.53)$$

$$\mathrm{demerit}_k(y, F, j)$$
$$\triangleq \min_{v \in \{0,1\}} (u b_k(y(x_j=v),\ F(x_j=v))) \qquad (2.54)$$

次に，分割変数を選ぶ手続き $\mathrm{select}_k$ を，次のように与える。

〔手続き 2.1〕($\mathrm{select}_k$)

**入力**：目的関数 $y(\mathbf{x})$，制約条件式 $F(\mathbf{x})$

**出力**：分割変数

**方法**：

(i) $J \leftarrow \{x_i \mid 1 \leq i \leq n,\ \text{for } 1 \leq \forall_j \leq n,$
$\mathrm{merit}_k(y, F, i) \geq \mathrm{merit}_k(y, F, j)\}$ とする；

(ii) $|J| = 1$ ならば，J の要素を $\mathrm{select}_k$ の値として，手続き終了；

(iii) $|J| \geq 2$ ならば，$K \leftarrow \{x_i \mid x_i \in J,\ \text{for } \forall x_i \in J,$
$\mathrm{demerit}_k(y, F, i) \leq \mathrm{demerit}_k(y, F, j)\}$ とする；

K中の任意の要素を $\mathrm{select}_k$ の値とする；

手続き終了。

先に示した例題 2.2 で，各変数の $\mathrm{merit}_0$，$\mathrm{demerit}_0$ の値は次のように計算される。

$merit_0(y, F, 1)=max(0, 1)=1$

$merit_0(y, F, 2)=max(0, 3)=3$

$merit_0(y, F, 3)=max(0, 5)=5$

$demerit_0(y, F, 1)=min(8, 9)=8$

$demerit_0(y, F, 2)=min(6, 9)=6$

$demerit_0(y, F, 3)=min(4, 9)=4$

よって，変数 $x_3$ が $select_0(y, F)$ によって与えられる分割変数となる．

## 2.5 部分問題の縮小

この節では，部分問題をその性質を用いて縮小する方法について述べる．部分問題の縮小は，部分問題に含まれる変数の個数を減少させることによって行なわれる．

〔定義 2.6〕（ブール関数の既約形）

式（2.3）で与えられるブール関数 F(x) を構成する各積項の次数がすべて 2 以上のとき，すなわち次の条件式（2.55）が満たされるとき，F(x) は既約形であるという．

$$|I_i| \geq 2 \text{ for } 1 \leq \forall_i \leq s \qquad (2.55)$$

ここに，$I_i$ は式（2.14）で与えられる集合である．

F(x) が既約形でない場合，F(x) を構成する積項中に，次式のような項 Fr(x)，（$1 \leq r \leq s$）が存在する．

$$Fr(x) = x_p^{\varepsilon rp} \qquad (2.56)$$

F(x) は積和形のブール関数であるので，F(x)=0 であるためには，Fr(x)=0 でなければならない．すなわち，$x_p^{\varepsilon rp}=0$ である．よって，変数 $x_p$ の値は，

$$x_p = \overline{\varepsilon}_{rp} \qquad (2.57)$$

でなければならない．

以上の考察にもとづいて，F(x) を既約化する手続きは，次のように書かれる．

〔手続き 2.2〕（F(x) の既約化）

**入力**: F(x)

**出力**: F(x) を既約化して得られるブール関数

$$F'(x) = \bigvee_{i=1}^{s'} F'_i(x) \qquad (2.58)$$

方法：

Step1：$F'(x) \leftarrow F(x)$とする；

Step2：$F'(x)$中から，$F'_i(x) \equiv 0$である項をすべて取り除く；

(i) $F'(x)$中に，$F'_i(x) \equiv 1$である項が存在すれば，$F'(x) \leftarrow 1$として，手続き終了；

(ii) そうでなくて，かつ，$F'(x)$中に，単一の変数のみからなる積項が存在しなければ，手続き終了；

(iii) それ以外の時，Step 3へ；

Step3：$F'(x)$中から，単一の変数のみからなる積項を1つえらび，$F'_i(x) \equiv x_j^{\varepsilon_{ij}}$とする；

変数$x_j$に，値$\overline{\varepsilon}_{ij}$を代入する；

$F'(x) \leftarrow F'(x_j = \overline{\varepsilon}_{ij})$として，Step 2へ．

$F(x)$を，手続き2.2によって既約化して得られるブール関数を$F'(x)$とする．このとき，擬似ブール計画問題$P(y, F)$の最適解が，擬似ブール計画問題$P(y, F')$の最適解と一致することは，この節のはじめに述べた考察より明らかである．ただし，$F(x)$の既約化を行う時に値が代入された変数の値は，$P(y, F')$においても，そのままとする．

次にアルゴリズムの実行途中で少くとも1つの最適解の候補が得られている時に用いられる，部分問題の縮小の方法について述べる．

〔定理2.3〕

Cを定数とする　制約条件$F(x) = 0$のもとで，

$$y(x) \leq C \tag{2.59}$$

を満足するxの集合を$T$とする．すなわち，

$$T \triangleq \{ x \mid x \in \{0, 1\}^n,\ F(x) = 0,\ y(x) \leq C \} \tag{2.60}$$

また，

$$a_i > C - \ell b_0(y, F) \tag{2.61}$$

$i$の集合をIとする．

このとき，次の命題が成り立つ．

$$Y_i(\xi) = 0,\ \text{for } \forall_i \in I,\ \forall \xi \in T;\ \text{if } T \neq \emptyset \tag{2.62}$$

〔証明〕

$T \neq \emptyset$とすると，

$$\ell b_0(y, F) \leq y(x) \leq C \tag{2.63}$$

であることから，次式が成り立つ．

$$a_i > 0, \ \forall i \in I \tag{2.64}$$

このとき，

$$Y_i(\xi) = 1 \tag{2.65}$$

をみたす$i \in I$および$\xi \in T$が存在すると仮定すると，$a_i > 0$であることから，

$$y(\xi) \geq \ell b_0(y, F) + a_i > C \tag{2.66}$$

となり，$\xi \in T$であることに反する．よって，式(2.65)を満たす$i$および$\xi$は存在しない．　Q. E. D

アルゴリズムのある時点までに得られている，最適解の候補に対する目的関数の値をCoptとする．定理2.3によれば，この時，$a_i > \mathrm{Copt} - \ell b_0(y, F)$を満たす$i$に対して，$Y_i(x) = 0$を制約条件につけ加えることによって，部分問題を縮小することができる．すなわち，新たに得られる制約条件式$F'(x)$は次式のようになる．

$$F'(x) = F(x) \vee (\bigvee_{i \in U} Y_i(x)) \tag{2.67}$$

ここに，Uは次式で与えられる

$$U = \{ i \mid 1 \leq i \leq m, \ a_i > C - \ell b_0(y, F) \} \tag{2.68}$$

式(2.67)で，$\{Y_i(x)\}$の中に，単一の変数のみからなる項が含まれている場合には，$F'(x)$は既約形とはならない．この場合には，この節の前半で述べた$F(x)$の既約化の手続き(手続き2.2)を用いて，部分問題をさらに縮小することができる．

## 2.6　擬似ブール計画法のアルゴリズム

この節では，前節までに行なってきた考察にもとづいて，擬似ブール計画法のアルゴリズムを与える．

〔アルゴリズム2.1〕

**入力**：擬似ブール計画問題$P(y, F)$；

**出力**：$P(y, F)$の全ての最適解の集合Xopt；

Xoptに対する目的関数の値Copt；

ただし，問題P（$y$，F）が，制約条件F（x）＝0を満足する可能解を持たないときには，Xopt＝$\emptyset$，Copt＝$\infty$が出力される．

記法：以下のアルゴリズム中では，未解決な部分問題に対応する変数のベクトルxを要素とする集合を$I$とする．また，変数$x_j$に値が割り当てられていないことを，$x_j$＝■で表わす．

方法：

Step1：〔初期設定〕

$I \leftarrow \{(u, u, \cdots, u)\}$，Xopt←$\emptyset$，Copt←$\infty$とする；Step2へ；

Step2：〔部分問題の選択〕

(i) $I=\emptyset$ならば，Step8へ；

(ii) $I \neq \emptyset$ならば，手続き2.3（後述）を用いてS中から，変数のベクトルを1つ選び，これを$\mathbf{x}_0$とする；$\mathbf{x}_0$をSから取り除いてStep3へ；

Step3：〔制約条件の処理〕

F（$\mathbf{x}_0$）を手続き2.2を用いて既約化して得られる式をF′（x）とする；

(i) F′（x）≡1ならば，Step2へ；

(ii) そうでなければ，Step4へ；

Step4：〔目的関数の処理〕

$y$（$\mathbf{x}_0$）を簡単化して得られる式を$y'$（x）とする；

(i) $\ell b_k(y', F')$ ＞ Coptならば，Step2へ；

(ii) $y'$（x）が定数関数ならば，Step7へ；

(iii) それ以外の時，Step5へ；

Step5：〔部分問題の縮小〕

$y'(\mathbf{x}) = a'_0 + \Sigma\ a'_i\ Y'_i(\mathbf{x})$とする；

(i) $a'_i$ ＞Copt $-\ \ell b_0(y, F)$かつ，$Y'_i$（x）が単一の変数のみからなるような$i$が存在すれば，この条件をみたす全ての$i$について，

$$Y'_i(\mathbf{x}) \equiv x_{j_i} \leftarrow 0$$

として，Step3へ；

(ii) 上記の条件を満足する$i$が存在しなければ，Step6へ；

Step6：〔部分問題の分割〕

$x_j \leftarrow \mathrm{select}_k(y', F')$とする；

$\mathbf{x}_■ = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n)$とするとき，

$\ell b_k(y'(x_j=v), F'(x_j=v)) \leq Copt$ (2.69)

を満足する全ての $v \in \{0, 1\}$ について，

$I \leftarrow I \cup (\tilde{x}_1, \cdots, \tilde{x}_{j-1}, v, \tilde{x}_{j+1}, \cdots, \tilde{x}_n)$

として，Step 2へ；

Step 7：〔最適解の候補の更新〕

$y'(\mathbf{x})$は定数関数である．$y'(\mathbf{x})$の値を$a'_0$とする．

(i) $a'_0 < Copt$ ならば，

$Xopt \leftarrow \{\mathbf{x}_0\}$，$Copt \leftarrow a'_0$

とする．$I$中から$\ell b_k(y(\mathbf{x}), F(\mathbf{x})) > Copt$となる要素を取り除く；

Step 2へ；

(ii) $a_0 = Copt$ ならば，

$Xopt \leftarrow Xopt \cup \{\mathbf{x}_0\}$

としてStep 2へ；

Step 8：〔停止〕

アルゴリズム終了．

〔手続き 2.3〕（部分問題の選択）

**入力**：部分問題の集合$I$；

**出力**：部分問題$\mathbf{x}$；

**方法**：

$I$の要素の中で，$\ell b_k(y(\mathbf{x}_0), F(\mathbf{x}_0))$の値が最小であるような要素の集合を$I'$とする；次に$I'$の中で，含まれるuの個数が最小である要素を1つえらび，$\mathbf{x}$とする；手続き終了．

次に，例題を与え，この例題がアルゴリズム2.1によって解かれる過程を少しくわしく示す．

〔例題 2.3〕

制約条件

$$F(\mathbf{x}) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_4\bar{x}_5\bar{x}_6 \vee x_1x_4 \vee x_2x_5 \vee x_3x_6 = 0 \quad (2.70)$$

のもとで，次式の目的関数を最小にする$\mathbf{x}$を求めよ．

$$y(\mathbf{x}) = 13x_1 + 13x_2 + 26x_3 + 11x_4 + 11x_5 + 22x_6 + 36x_1x_2 + 18x_1x_3 + 6x_1x_5 + x_1x_6$$
$$+ 18x_2x_3 + 6x_2x_4 + x_2x_6 + 5x_3x_4 + 5x_3x_5 \quad (2.71)$$

以下では，$I$ の要素を，次の形で表わす．

$$〔(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n), \ell b_k〕 \qquad (2.72)$$

ここに，$\tilde{x}_j$ は，変数 $x_j$ に割り当てられている値を表わす．また，$\ell b_k$ は，$\ell b_k(y(\mathbf{x}), F(\mathbf{x}))$ の値を表わす．ただし，以下では $k=0$ とした．

〔例題の解法〕

Stage #0

Step 1 : $I \leftarrow \{〔(u, u, u, u, u, u), -\infty〕\}$

$\mathrm{Xopt} \leftarrow \emptyset$, $\mathrm{Copt} \leftarrow \infty$ ;

Step 2 : $I = \{〔(u, u, u, u, u, u), -\infty〕\}$

$\mathbf{x}_0 \leftarrow (u, u, u, u, u, u,)$ ;

Step 3 : $F'(\mathbf{x}) = F(\mathbf{x}_0) = F(\mathbf{x})$ ;

Step 4 : $y'(\mathbf{x}) = y(\mathbf{x}_0) = y(\mathbf{x})$ ;

Step 5 : ——

Step 6 : $\mathrm{select}_0(y', F') = x_3$

$I \leftarrow I \cup \{〔(u, u, 0, u, u, u), 0〕, 〔(u, u, 1, u, u, u), 26〕\}$ ;

Stage #1

Step 2 : $I = \{〔(u, u, 0, u, u, u), 0〕, 〔(u, u, 1, u, u, u), 26〕\}$ ;

$\mathbf{x}_0 \leftarrow (u, u, 0, u, u, u)$ ;

Step 3 : $F'(\mathbf{x}) = F(\mathbf{x}_0)$

$= \bar{x}_1 \bar{x}_2 \vee \bar{x}_4 \bar{x}_5 \bar{x}_6 \vee x_1 x_4 \vee x_2 x_5$ ;

Step 4 : $y'(\mathbf{x}) = y(\mathbf{x}_0)$

$= 13x_1 + 13x_2 + 11x_4 + 11x_5 + 22x_6 + 36x_1x_2 + 6x_1x_5 + x_1x_6 + 6x_2x_4 + x_2x_6$ ;

Step 5 : ——

Step 6 : $\mathrm{select}_0(y', F') = x_6$ .

$I \leftarrow I \cup \{〔(u, u, 0, u, u, 0), 0〕, 〔(u, u, 0, u, u, 1), 22〕\}$ ;

Stage #2

Step 2 : $I = \{〔(u, u, 0, u, u, 0), 0〕, 〔(u, u, 0, u, u, 1), 22〕,$

$〔(u, u, 1, u, u, u), 26〕\}$,

$\mathbf{x}_0 \leftarrow (u, u, 0, u, u, 0)$ ;

Step 3 : $F'(\mathbf{x}) = F(\mathbf{x}_0)$

$= \bar{x}_1 \bar{x}_2 \vee \bar{x}_4 \bar{x}_5 \vee x_1 x_4 \vee x_2 x_5$ ;

Step 4 : $y'(\mathbf{x}) = y(\mathbf{x}_0)$

$= 13x_1 + 13x_2 + 11x_4 + 11x_5 + 36x_1x_2 + 6x_1x_5 + 6x_2x_4$ ;

Step 5 : ——

Step 6 : $\mathrm{select}_0(y', F') = x_1$ ;

$I \leftarrow I \cup \{[(0, u, 0, u, u, 0), 0], [(1, u, 0, u, u, 0), 13]\}$ ;

Stage # 3

Step 2 : $I = \{[(0, u, 0, u, u, 0), 0], [(1, u, 0, u, u, 0), 13],$

$[(u, u, 0, u, u, 1), 22], [(u, u, 1, u, u, u), 26]\}$

$\mathbf{x}_0 \leftarrow (0, u, 0, u, u, 0)$ ;

Step 3 : $F(\mathbf{x}_0) = \bar{x}_2 \vee \bar{x}_4 \bar{x}_5 \vee x_2 x_5$ ;

$x_2 \leftarrow 1, \quad x_5 \leftarrow 0, \quad x_4 \leftarrow 1,$

$\mathbf{x}_0 = (0, 1, 0, 1, 0, 0)$ ;

$F'(\mathbf{x}) \equiv 0$ ;

Step 4 : $y'(\mathbf{x}) = y(\mathbf{x}_0) = 30$ ;

Step 7 : $\mathrm{Xopt} \leftarrow \{(0, 1, 0, 1, 0, 0)\}$,

$\mathrm{Copt} \leftarrow 30$ ;

Stage # 4

Step 2 : $I = \{[(1, u, 0, u, u, 0), 13], [(u, u, 0, u, u, 1), 22],$

$[(u, u, 1, u, u, u), 26]\}$

$\mathbf{x}_0 \leftarrow (1, u, 0, u, u, 0)$ ;

Step 3 : $F(\mathbf{x}_0) = \bar{x}_4 \bar{x}_5 \vee x_4 \vee x_2 x_5$

$x_4 \leftarrow 0, \; x_5 \leftarrow 1, \; x_2 \leftarrow 0$

$\mathbf{x}_0 = (1, 0, 0, 0, 1, 0)$

$F'(\mathbf{x}) \equiv 0$ ;

Step 4 : $y'(\mathbf{x}) = 30$ ;

Step 7 : $\mathrm{Xopt} \leftarrow \mathrm{Xopt} \cup \{(1, 0, 0, 0, 1, 0)\}$

Stage # 5

Step 2 : $I = \{[(u, u, 0, u, u, 1), 22], [(u, u, 1, u, u, u), 26]\}$

$\mathbf{x}_0 \leftarrow (u, u, 0, u, u, 1)$ ;

Step 3 : $F'(\mathbf{x})=F(\mathbf{x}_0)$

$=\bar{x}_1\bar{x}_2 \vee x_1 x_4 \vee x_2 x_5$ ;

Step 4 : $y'(\mathbf{x})=22+14x_1+14x_2+11x_4+11x_5+36x_1x_2+6x_1x_5+6x_2x_4$ ;

Step 5 : Copt $-\ell b_0(y', F')=30-22=8,\quad 14>8,\quad 11>8,$

$\therefore x_1 \leftarrow 0_1\ x_2 \leftarrow 0_1\ x_4 \leftarrow 0_1\ x_5 \leftarrow 0$ ;

$\mathbf{x}_0=(0, 0, 0, 0, 0, 1)$ ;

Step 3 : $F'(\mathbf{x})\equiv 1$ ;

Stage #6

Step 2 : $I=\{[(u, u, 1, u, u, u), 26]\}$ ;

$\mathbf{x} \leftarrow (u, u, 1, u, u, u)$ ;

Step 3 : $F(\mathbf{x}_0)=\bar{x}_4\bar{x}_5 \vee x_6 \vee x_1x_4 \vee x_2x_5 \vee x_6,\ x_6 \leftarrow 0,\ \mathbf{x}_0=(u, u, 1, u, u, 0),$

$F'(\mathbf{x})=\bar{x}_4\bar{x}_5 \vee x_1x_4 \vee x_2x_5$ ;

Stcp 4 : $y'(\mathbf{x})=26+31x_1+31x_2+16x_4+16x_5+36x_1x_2+6x_1x_5+6x_2x_4$ ;

Step 5 : Copt $-\ell b_0(y', F')=30-26=4,\ 31>4,\ 16>4,$

$\therefore x_1 \leftarrow 0_1\ x_2 \leftarrow 0_1\ x_4 \leftarrow 0_1\ x_5 \leftarrow 0$

$\mathbf{x}_0=(0, 0, 1, 0, 0, 0)$;

Step 3 : $F'(\mathbf{x}) \equiv 1$ ;

Stage # 7

Step 2 : $I=\varnothing$ ;

Step 8 : 停止.

例題 2.3 が解かれる様子を図 2.2 に木を用いて示す.

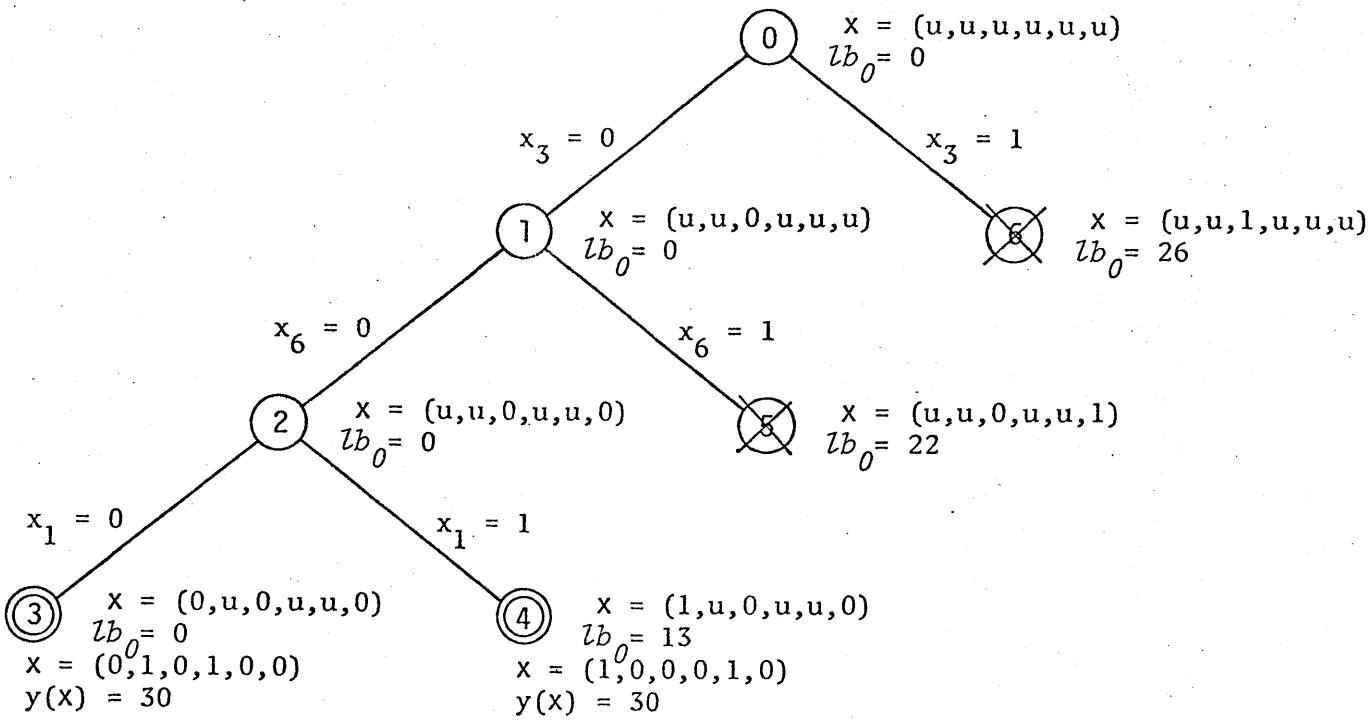



図２．２　例題２．３の解かれる過程

## ２．７　実験およびその結果

この節では，前節で示した擬似ブール計画法のアルゴリズムを実際に計算機のプログラムとして実現し，その効率を調べた結果について述べる．

### ２．７．１　アルゴリズムの実現

前節で示したアルゴリズム２．１を，FORTRAN（JIS7000程度）で実現した．使用した計算機は，名古屋大学大型計算機センターのFACOM230／60，使用したコンパイラはFORTRAN-Dである．また，このプログラムを実行するのには約45KW（1W=36bits）を要した．

### ２．７．２　問題の作制方法

以下で示す実験に用いられた擬似ブール計画問題は，次の３つのパラメータを与えて，計算機の一様乱数を用いて発生された：

(i)　$n$：問題に含まれる変数の個数；

(ii)　$s$：制約条件式F（x）の項数；

(iii)　$m$：目的関数$y$（x）の項数．

さらに，問題の形を指定するために，制約条件式および目的関数の各項の次数を，それぞれ，次のように定めた．

(i) 制約条件式については，各積項$F_i(x)$に含まれる変数の個数が2, 3, 4である確率を，それぞれ，27%，46%，27%とした．

(ii) 目的関数については，各積項$Y_i(x)$に含まれる変数の個数が，1, 2, 3, 4である確率をそれぞれ，36%，36%，21%，21%とした．

### 2.7.3 実験 1

実験1は，アルゴリズム2.1で用いられる，目的関数の下限の評価関数$\ell b_k$の有効さを調べるためのもので，次のようにして実行された．

2.7.2で述べた方法によって，擬似ブール計画問題を発生させ，これらの問題を，$\ell b_0$を用いた場合および$\ell b_1$を用いた場合について，アルゴリズム2.1で解き，それぞれの場合の計算時間およびアルゴリズム中で生成された部分問題の個数を測定した．

表2.1および表2.3に，$n = 10$の場合および$n = 20$の場合のアルゴリズムの計算時間を示

表 2.1　アルゴリズムの計算時間（n = 10）

| s | m | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| 10 | $lb_0$ | 0.310 | 0.801 | 1.40 | 2.28 | 2.82 |
| | $lb_1$ | 0.456 | 1.03 | 1.71 | 2.72 | 3.23 |
| | 比 | 1.47 | 1.29 | 1.22 | 1.19 | 1.14 |
| 20 | $lb_0$ | 0.302 | 0.656 | 1.17 | 1.47 | 2.05 |
| | $lb_1$ | 0.511 | 1.03 | 1.62 | 2.09 | 2.82 |
| | 比 | 1.69 | 1.57 | 1.38 | 1.42 | 1.38 |
| 30 | $lb_0$ | 0.359 | 0.665 | 0.835 | 1.06 | 1.16 |
| | $lb_1$ | 0.585 | 1.02 | 1.34 | 1.73 | 1.86 |
| | 比 | 1.63 | 1.53 | 1.60 | 1.63 | 1.60 |
| 40 | $lb_0$ | 0.313 | 0.535 | 0.732 | 1.06 | 1.05 |
| | $lb_1$ | 0.504 | 0.886 | 1.20 | 1.75 | 1.75 |
| | 比 | 1.61 | 1.66 | 1.64 | 1.65 | 1.67 |
| 50 | $lb_0$ | 0.332 | 0.490 | 0.706 | 0.898 | 0.906 |
| | $lb_1$ | 0.544 | 0.809 | 1.13 | 1.42 | 1.52 |
| | 比 | 1.64 | 1.65 | 1.60 | 1.58 | 1.68 |

（単位：秒）

表2.2　生成された部分問題の個数（ n = 10）

| s | m | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| 10 | $lb_0$ | 13.6 | 21.0 | 25.9 | 36.9 | 41.1 |
| | $lb_1$ | 13.5 | 18.9 | 22.8 | 33.3 | 35.1 |
| | 比 | 0.993 | 0.900 | 0.880 | 0.902 | 0.854 |
| 20 | $lb_0$ | 12.6 | 16.6 | 22.4 | 23.2 | 24.9 |
| | $lb_1$ | 12.3 | 15.0 | 18.5 | 19.7 | 22.4 |
| | 比 | 0.976 | 0.938 | 0.826 | 0.849 | 0.900 |
| 30 | $lb_0$ | 11.8 | 13.1 | 12.8 | 13.9 | 12.6 |
| | $lb_1$ | 11.5 | 11.5 | 12.3 | 12.5 | 11.7 |
| | 比 | 0.975 | 0.878 | 0.961 | 0.899 | 0.929 |
| 40 | $lb_0$ | 8.70 | 8.00 | 8.20 | 10.1 | 9.30 |
| | $lb_1$ | 8.40 | 7.50 | 7.90 | 9.30 | 9.30 |
| | 比 | 0.966 | 0.938 | 0.963 | 0.921 | 1.00 |
| 50 | $lb_0$ | 6.40 | 6.00 | 7.00 | 7.40 | 5.60 |
| | $lb_1$ | 6.30 | 6.00 | 6.80 | 7.30 | 5.20 |
| | 比 | 0.984 | 1.00 | 0.971 | 0.986 | 0.929 |

表2.3　アルゴリズムの計算時間（ n = 20）

| s | m | 20 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|---|
| 20 | $lb_0$ | 5.25 | 18.7 | 39.6 | 101 | 111 |
| | $lb_1$ | 7.75 | 22.6 | 55.5 | 106 | 166 |
| | 比 | 1.48 | 1.21 | 1.40 | 1.04 | 1.50 |
| 40 | $lb_0$ | 4.77 | 11.7 | 27.2 | 37.7 | 48.6 |
| | $lb_1$ | 7.16 | 18.4 | 41.7 | 54.9 | 75.6 |
| | 比 | 1.50 | 1.57 | 1.53 | 1.46 | 1.56 |
| 60 | $lb_0$ | 4.40 | 10.0 | 11.4 | 16.2 | 17.9 |
| | $lb_1$ | 7.46 | 15.5 | 22.1 | 28.2 | 33.7 |
| | 比 | 1.70 | 1.55 | 1.94 | 1.74 | 1.88 |
| 80 | $lb_0$ | 2.43 | 4.84 | 6.90 | 9.11 | 8.09 |
| | $lb_1$ | 4.82 | 8.65 | 13.0 | 16.0 | 15.6 |
| | 比 | 1.98 | 1.79 | 1.88 | 1.76 | 1.93 |

表2.4　生成された部分問題の個数（n＝20）

| s | m | 20 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|---|
| 20 | $lb_0$ | 58.7 | 113 | 208 | 380 | 400 |
| | $lb_1$ | 50.2 | 81.8 | 159 | 288 | 311 |
| | 比 | 0.855 | 0.724 | 0.766 | 0.759 | 0.776 |
| 40 | $lb_0$ | 45.4 | 86.8 | 127 | 163 | 174 |
| | $lb_1$ | 35.2 | 71.7 | 102 | 130 | 138 |
| | 比 | 0.775 | 0.826 | 0.802 | 0.795 | 0.796 |
| 60 | $lb_0$ | 40.1 | 53.4 | 51.4 | 51.2 | 52.7 |
| | $lb_1$ | 34.3 | 47.4 | 46.3 | 46.8 | 50.2 |
| | 比 | 0.855 | 0.888 | 0.901 | 0.914 | 0.953 |
| 80 | $lb_0$ | 17.4 | 18.4 | 20.6 | 19.8 | 16.4 |
| | $lb_1$ | 16.3 | 18.2 | 20.1 | 19.3 | 16.0 |
| | 比 | 0.937 | 0.989 | 0.976 | 0.975 | 0.976 |

す．また，表2.2および表2.4に，$n$＝10の場合および$n$＝20の場合，アルゴリズム中で生成された部分問題の個数を示す．これらの計算時間および部分問題の個数は，いずれも各々の場合についての20問題の平均である．

表2.2および表2.4より，アルゴリズム中で生成される部分問題の個数は，評価関数として$\ell b_1$を用いたときの方が，$\ell b_0$を用いたときよりも減少することが知られる．しかしながら，アルゴリズムの計算時間は，逆に，$\ell b_1$を用いた時の方が多くなっている．これは，次の理由によると考えられる．$\ell b_1$の値を計算するためには，最悪の場合，2×$n$個の$\ell b_0$の値が必要である．* このため，一般に，1つの部分問題を処理するために必要な計算時間は，$\ell b_1$を用いた時の方が，$\ell b_0$を用いた時よりも増大する．$\ell b_1$を用いることによる計算時間の増加の割合が，$\ell b_1$を用いることによる部分問題の減少の割合よりも大きい場合には，アルゴリズム全体の計算時間は，$\ell b_1$を用いた時の方がかえって増大することになる．

*）式（2.21）参照

### 2.7.4 実 験 2

実験2では，アルゴリズム2.1の効率と，従来の擬似ブール計画法のアルゴリズムの効率との比較を行なった．

従来の擬似ブール計画問題の解法としては，P. L. Hammerらのアルゴリズム[3]などが知られている．このアルゴリズムは，制約条件を持たない擬似ブール関数の最大化の問題を扱うものである．

2.7.2で述べた方法によって，制約条件を持たない（ s = 0 ）擬似ブール計画問題を発生させ，これらの問題をアルゴリズム2.1で解いた時の計算時間およびアルゴリズム中で生成された部分問題の個数を測定した．ただし，評価関数は $\ell b_0$を用いた．表2.5に，計算時間および部分問題の個数の，各10問題についての平均値と標準偏差とを示す．この実験で，$n = 30$，$m \geq 80$の問題では，記憶容量の不足のため完全には解かれない問題も存在した．

表2.5　アルゴリズムの計算時間および生成された部分問題の個数（制約条件なし）

| $n$ | $m$ | 計算時間（秒） 平均値 | 計算時間（秒） 標準偏差 | 生成された部分問題の個数 平均値 | 生成された部分問題の個数 標準偏差 |
|---|---|---|---|---|---|
| 10 | 10 | 0.172 | 0.067 | 20.8 | 6.9 |
| 10 | 20 | 0.455 | 0.135 | 36.3 | 13.4 |
| 10 | 30 | 0.834 | 0.270 | 47.6 | 18.2 |
| 10 | 40 | 1.28 | 0.345 | 51.1 | 17.2 |
| 10 | 50 | 1.67 | 0.610 | 54.0 | 23.7 |
| 20 | 10 | 0.890 | 0.830 | 64.4 | 51.3 |
| 20 | 20 | 1.05 | 0.480 | 46.8 | 16.9 |
| 20 | 30 | 2.98 | 1.63 | 88.7 | 47.6 |
| 20 | 40 | 7.18 | 2.47 | 146 | 50.5 |
| 20 | 50 | 27.8 | 21.3 | 395 | 268 |
| 30 | 20 | 4.08 | 4.82 | 105 | 130 |
| 30 | 40 | 30.8 | 33.7 | 269 | 228 |
| 30 | 60 | 91.8 | 43.9 | 558 | 235 |

$n$：変数の個数

$m$：目的関数の項数

表 2.6　P. L. Hammer らのアルゴリズムの計算時間

| 問題番号 | 変数の項数 | 目的関数の項数 | くり返しの回数 | 計算時間（秒）（I/Oを含む） |
|---|---|---|---|---|
| 1 | 10 | 5 | 6 | 0.48 |
| 2 | 10 | 10 | 8 | 2.02 |
| 3 | 10 | 11 | 6 | 0.90 |
| 4 | 10 | 20 | 16 | 2.31 |
| 5 | 10 | 29 | 33 | 7.35 |
| 6 | 10 | 48 | 34 | 11.40 |
| 7 | 20 | 10 | 12 | 2.02 |
| 8 | 20 | 19 | 17 | 4.99 |
| 9 | 20 | 29 | 28 | 9.36 |
| 10 | 20 | 34 | 97 | 37.96 |
| 11 | 20 | 40 | 118 | 65.44 |
| 12 | 20 | 50 | 148 | 110.95 |
| 13 | 30 | 30 | 50 | 24.06 |
| 14 | 30 | 38 | 120 | 106.17 |
| 15 | 30 | 50 | 212 | 239.03 |
| 16* | 30 | 70 | >77 | >116.22 |
| 17* | 40 | 54 | >84 | >117.57 |
| 18* | 40 | 74 | >154 | >238.38 |
| 19* | 40 | 143 | >6 | >72.45 |
| 20* | 50 | 80 | >9 | >57.44 |
| 21* | 50 | 98 | >7 | >51.62 |

*）記憶容量の不足のため，完全には解かれていない

表 2.6 に，P. L. Hammer らの行なった実験[3] の結果を示す．ただし，この実験は次の条件のもとで行なわれた．まず，アルゴリズムは，FORTRAN IVG で記述され，IBM360/50 上で実現された．また，実験に用いられた問題は，変数の個数と目的関数の項数とを与えて，乱数を用いて発生された．この実験で，一部の問題は記憶容量の不足のため完全には解かれていない．

表 2.5，表 2.6 より，アルゴリズム 2.1 は，少くとも P. L. Hammer らのアルゴリズムと同程度の能力を持つと考えられる．

## 2.8 むすび

本章では，Branch-and-Bound法にもとづく擬似ブール計画法のアルゴリズムを構成するために必要な，いくつかの問題点について論じてきた．さらに，擬似ブール計画法のアルゴリズムを提案し，実験をとおしてその効率を調べた結果について述べた．実験の結果によれば，実現されたシステムは少なくとも従来の解法と同程度の能力を持つことが知られた．

さらに大規模な問題を効率良く解くためには，次の問題点を解決しなければならない．

(i) 本章で示した擬似ブール計画法のアルゴリズムでは，ブール関数および擬似ブール関数の数式処理を行なっている．よって，アルゴリズムの計算時間を短縮するためには，これらの関数を効率良く処理するための，データ構造と数式処理アルゴリズムを開発する必要がある．

(ii) 評価関数 $\ell b_1$ は，$\ell b_0$ よりも目的関数の下界に関して，より良い評価を与える．しかしながら，$\ell b_1$の計算に要する時間は，$\ell b_0$よりも多くなる．そのため2.7節で示した実験の範囲内では，アルゴリズムの計算時間は，$\ell b_1$を用いた場合の方が$\ell b_0$を用いた時よりも多くなっている．一般に，アルゴリズム中で用いられる評価関数の強力さと評価関数の計算時間の間には，trade-off があると考えられるから,アルゴリズム全体の計算時間を最小にする，適度に強力な評価関数を導びき得る可能性がある．

(iii) 本章で示した擬似ブール計画法のアルゴリズムでは，次に処理すべき部分問題として，$\ell b_k$ の値を最小にするものを選択している（手続き2.3）．この方策は，Best-Bound Search と呼ばれる方法である．この方策を用いて全ての最適解を求める場合に処理される部分問題の個数は，他の方策を用いた場合よりも少ないかあるいは等しいことが容易に知られる．しかしながら，この方策を用いた場合には，アルゴリズムの実行途中で保持される未解決な部分問題の個数は，変数の個数 $n$ の増加に伴なって指数関数的に増加する．よって，記憶容量の不足のために，大規模な問題を処理するのは困難となる場合が多い．よって，与えられた記憶容量の範囲内で効率良く問題を解く方策を開発する必要がある．

(iv) 本章で示したアルゴリズムは，いわば汎用の擬似ブール計画問題の解法である．これに対して，特定の形の問題の処理に適した，専用のアルゴリズムを構成することは，理論的興味のみならず，実用的見地からも大いに意味があると考えられる．

(v) Branch-and-Bound 法の実行途中の未解決な部分問題は，互いに独立に処理が可能である．よって，Branch-and-Bound アルゴリズムを，各部分問題のレベルで，並列に実行することが可能である．並列形 Branch-and-Bound アルゴリズムの開発およびその特性の解析は，実用的見地からも大きな意味があると考えられる．

以下の章では，これらの問題について論ずる．すなわち，第3章では(i)について，第4章で

は(ii)について，第5章では(iv)について，第6章では(v)について，さらに第7章では，(iii)について，それぞれ述べることにする．

# 第3章　ハッシングの技法を用いた数式処理アルゴリズムとその擬似ブール計画法への応用

## 3.1　まえがき

第2章で述べた擬似ブール計画法のアルゴリズムは，ブール制約条件のもとで擬似ブール関数で与えられる目的関数を最小化する．このアルゴリズムでは，部分問題は原問題に含まれる変数の一部に値を与えることによって得られる．よって，このアルゴリズムを実現するためには，ブール関数および擬似ブール関数の数式処理が必要である．

これらの関数の数式処理を行う際の問題点の一つは，数式の簡単化をどのようにして行うかということである．簡単化の処理の中心となるのは，同類項をみつけ出す操作であり，これは，与えられた集合中に,ある要素が存在するか否かの判定を通して行なわれる．この判定を実現する方法として，主に次の3通りの方法が考えられる．[1]

(i)　線形探索にもとづく方法

(ii)　2分探索にもとづく方法

(iii)　ハッシングによる方法

数式の簡単化に要する計算時間を，項と項との比較の回数によって評価することにする．処理される数式の項数を$m$で表わすと，上記(i)および(ii)の方法を用いた場合の，数式の簡単化に要する計算時間は，それぞれ$O(m^2)$および$O(m\log_2 m)$となる．よって(i)の方法では，項数の多い数式を効率良く処理することは困難である．一方，(iii)の方法は，計算時間が$O(m)$でよいので，大規模な数式の処理に適している．

この章では，ハッシングの技法を用いた，ブール関数および擬似ブール関数の数式処理アルゴリズムを提案する．さらに，これらの数式処理アルゴリズムを，擬似ブール計画法のアルゴリズムに適用することを試みる．

これらの数式処理アルゴリズムは，次の特長を持っている，まず第一に，数式の処理に要する計算時間が$O(m)$である．第二に，ハッシュ関数として特別な形のものを採用することで，積項が単一の変数からなるか否かの判定が高速に行える．これらの点から，この章で提案する数式処理アルゴリズムは，多数の項からなる数式の処理に適するものである．

本章では初めに，3.2で，この章で扱うブール関数および擬似ブール関数を定義する．また，これらの関数の，計算機の記憶装置内部での表現方法（以下では内部表現と呼ぶ）について述べる．次に，3.3では，ハッシングの技法を用いた数式処理アルゴリズムと，線形探索にもとづく数式処理アルゴリズムとについて述べる．3.4では，これらの数式処理アルゴリズムの計

算時間の理論的解析を行なう．

最後に，3.5では，両者のアルゴリズムを用いて実現された，2つの擬似ブール計画法のアルゴリズムの効率の比較を行なった実験の結果について述べる．

## 3.2 ブール関数および擬似ブール関数とそれらの内部表現

この節では，初めに，以下で扱われるブール関数および擬似ブール関数を定義する．さらに，これらの関数が擬似ブール計画法のアルゴリズムの中で果す役割について述べる．次に，これらの数式の効率的な処理を目的とした内部表現を与える．最後に，数式を構成する項のレベルでの基本的な演算がどのようにして実現されるかを示す．

### 3.2.1 ブール関数と擬似ブール関数

$x_j \in \{0, 1\}$，$(j=1, 2, \cdots, n)$，$\mathbf{x} \triangleq (x_1, x_2, \cdots, x_n)$とする．ブール関数$F(\mathbf{x})$および擬似ブール関数$y(\mathbf{x})$を，それぞれ次のように定義する．

〔定義3.1〕

ブール関数$F(\mathbf{x})$は次式で与えられるものとする．

$$\begin{cases} F(\mathbf{x}) \triangleq \bigvee_{i=1} F_i(\mathbf{x}) \\ F_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\varepsilon_{ij}}, (i=1, 2, \cdots, s) \end{cases} \tag{3.1}$$

ここに，$\vee$および$\prod$は，それぞれ論理和および論理積を表わす．また，$x_j^{\varepsilon_{ij}}$は，$\varepsilon_{ij} \in \{0, 1, t, f\}$として，次式で与えられる．

$$x_j^{\varepsilon_{ij}} \triangleq \begin{cases} \bar{x}_j & ; \text{if } \varepsilon_{ij} = 0 \\ x_j & ; \text{if } \varepsilon_{ij} = 1 \\ 1 & ; \text{if } \varepsilon_{ij} = t \\ 0 & ; \text{if } \varepsilon_{ij} = f \end{cases} \tag{3.2}$$

ここに，$\bar{x}_j$は変数$x_j$の否定を表わす．

〔定義3.2〕

擬似ブール関数$y(\mathbf{x})$は次式で与えられるものとする．

$$\begin{cases} y(\mathbf{x}) \triangleq \sum_{i=1}^{m} a_i Y_i(\mathbf{x}) \\ Y_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\delta_{ij}}, (i=1, 2, \cdots, m) \end{cases} \tag{3.3}$$

ここに，$a_i$，（$i=1,2,\cdots,m$）は整数とする．また，$x_j^{\delta_{ij}}$ は $\delta_{ij}\in\{0,1\}$ として，次式で与えられる．

$$x_j^{\delta_{ij}} \triangleq \begin{cases} 1 & ; \quad \text{if} \quad \delta_{ij}=0 \\ x_j & ; \quad \text{if} \quad \delta_{ij}=1 \end{cases} \tag{3.4}$$

第2章で述べた擬似ブール計画法のアルゴリズムは，ブール制約条件F（x）＝0のもとで，擬似ブール関数で与えられる目的関数 $y$（x）を最小化する．その過程で，これらの数式に対して次の処理が必要である．

(1) ブール関数の処理

式（3.1）で与えられるブール関数F（x）に含まれる変数の一部に，0または1を代入して得られる式を，

$$\tilde{F}(x) \triangleq \bigvee_{i=1}^{s} \tilde{F}_i(x) \tag{3.5}$$

で表わす．このとき，ブール式の処理は次の(i)～(iii)の操作からなる．

(i) $\tilde{F}_i(x)$，（$1\leq i\leq s$）を求めること

(ii) $\tilde{F}_i(x)$，（$1\leq i\leq s$）が単一の変数からなるか否かの判定*

(iii) $\tilde{F}(x)$の簡単化，すなわち，

$$\tilde{F}_i(x) \not\equiv \tilde{F}_j(x) \quad \text{if} \quad i\neq j \tag{3.6}$$

となるようにすること**

(2) 擬似ブール関数の処理

式(3)で定義される擬似ブール関数 $y$（x）に含まれる変数の一部に，0または1を代入して得られる式を，

$$\tilde{y}(x) \triangleq \sum_{i=1}^{m} a_i \tilde{Y}_i(x) \tag{3.7}$$

として，次の(i)，(ii)を擬似ブール関数の処理と呼ぶ．

(i) $\tilde{Y}_i(x)$，（$1\leq i\leq m$）を求めること

(ii) $\tilde{y}$（x）の簡単化，すなわち，

$$\tilde{Y}_i(x) \not\equiv \tilde{Y}_j(x) \quad \text{if} \quad i\neq j$$

となるようにすること

---

*）積項が単一の変数になれば，その変数の値が自動的に定まるという理由による．

**）式 $\tilde{F}_i(x)$ と $\tilde{F}_j(x)$ の内部表現が等しいことを，$\tilde{F}_i(x)\equiv\tilde{F}_j(x)$ で表わす．なお，ここでいう式の簡単化には，相補律，吸収律などを適用するものは含まれない．

### 3. 2. 2　数式の内部表現

定義 3. 1，3. 2 によって定義されたブール関数および擬似ブール関数を計算機で処理するためには，これらの式を記憶装置内部で表現する必要がある．以下では，ここで用いた表現方法について述べる．

(1)　ブール関数の内部表現

まず，式（3. 1）の各 $F_i(\mathbf{x})$ に対して，次式で定義される 2 つの整数 $\sigma_i$，$\tau_i$ の順序対（以下では，単に対という）（$\sigma_i$，$\tau_i$）を対応させる．

$$\begin{cases} \sigma_i \triangleq \sum_{j \in I_i} 2^{j-1} \\ \tau_i \triangleq \sum_{j \in J_i} 2^{j-1} \end{cases} \tag{3.9}$$

ここに，$I_i, J_i$（$i = 1, 2, \cdots, m$）は次式で与えられる．

$$\begin{cases} I_i \triangleq \{ j \mid 1 \leq j < n,\ (\varepsilon_{ij} = 1) \vee (\varepsilon_{ij} = 0) \} \\ J_i \triangleq \{ j \mid 1 \leq j < n,\ (\varepsilon_{ij} = 1) \vee (\varepsilon_{ij} = f) \} \end{cases} \tag{3.10}$$

式（3. 1）のブール関数 F（x）に対して，次式で与えられる集合を対応させる．

$$\{ (\sigma_i, \tau_i) \mid 1 \leq i \leq m \} \tag{3.11}$$

式（3. 1 1）で与えられる整数の対の並びは，計算機の記憶装置内部では，整数形 2 次元配列などで表現される．以下では，$\sigma_i$，$\tau_i$ の 2 進数表現をそれぞれ $\boldsymbol{\sigma}_i$，$\boldsymbol{\tau}_i$ で表わすことにする．たとえば，積項 $F_i(\mathbf{x}) = x_1 \cdot \bar{x}_3 \cdot x_5$ は，計算機内部では，$\boldsymbol{\sigma}_i = 10101$，$\boldsymbol{\tau}_i = 10001$ によって表わされる．

(2)　擬似ブール関数の内部表現

式(2)の各 $Y_i(\mathbf{x})$ に対して，次式で定義される整数 $\eta_i$ を対応させる．

$$\eta_i \triangleq \sum_{j \in K_i} 2^{j-1} \tag{3.12}$$

ここに $K_i$ は，次式で与えられる．

$$K_i \triangleq \{ j \mid 1 \leq j \leq n,\ \delta_{ij} = 1 \} \tag{3.13}$$

式（3. 3）の擬似ブール関数 $y$（x）に対して，次式で与えられる，$\eta_i$ と $a_i$ との対の並びを対応させる．

$$\{ (\eta_i, a_i) \mid 1 \leq i \leq m \} \tag{3.14}$$

以下では $\eta_i$ の 2 進数表現を $\boldsymbol{\eta}_i$ で表わす．

(3)　ベクトル x の内部表現

擬似ブール計画法のアルゴリズムの途中に現われる，未定要素を含む $n$ 次元ベクトル x に対

し，次式で与えられる2つの整数$\xi$，$\theta$の対（$\xi$，$\theta$）を対応させる．

$$\begin{cases} \xi \triangleq \sum_{j \in M} 2^{j-1} \\ \theta \triangleq \sum_{j \in N} 2^{j-1} \end{cases} \tag{3.15}$$

ここに，M，Nはそれぞれ次式で定義される．

$$\begin{cases} M \triangleq \{ j \mid 1 \leq j \leq n,\ x_j = u \} \\ N \triangleq \{ j \mid 1 \leq j \leq n,\ x_j = 1 \} \end{cases} \tag{3.16}$$

ここに，$x_j = u$は，変数$x_j$の値が未定であることを表わす．

以下では，$\xi$，$\theta$の2進数表現をそれぞれ$\boldsymbol{\xi}$，$\boldsymbol{\theta}$で表わすことにする．たとえば，$\mathbf{x} = (1, 0, u, 1, 0)$は$x_1 = x_4 = 1$，$x_2 = x_5 = 0$，$x_3 = u$であることを表わす．この$\mathbf{x}$は，計算機内部では，$\boldsymbol{\xi} = 00100$，$\boldsymbol{\theta} = 10010$によって表わされる．

### 3.2.3 数式の処理

$F(\mathbf{x})$，$y(\mathbf{x})$を構成する各項およびベクトル$\mathbf{x}$を上記のように内部表現することによって計算機のビット処理機能を利用して，数式に対する種々の演算が容易にかつ高速で実行される．たとえば，積項どうしの比較，積項に含まれる変数の一部への値の代入，積項が単一の変数からなるか否かの判定などの演算は，次のようにして実行される．

(1) 積項どうしの比較

$F_i(\mathbf{x}) \equiv F_j(\mathbf{x})$であるか否か，および$Y_i(\mathbf{x}) \equiv Y_j(\mathbf{x})$であるか否かは，次の関係によって判定される．

(i) $F_i(\mathbf{x}) \equiv F_j(\mathbf{x}) \Leftrightarrow [\sigma_i = \sigma_j] \wedge [\tau_i = \tau_j]$ (3.17)

(ii) $Y_i(\mathbf{x}) \equiv Y_j(\mathbf{x}) \Leftrightarrow \eta_i = \eta_j$ (3.18)

ここに$\Leftrightarrow$は対等を表わす．

(2) $F_i(\mathbf{x})$に含まれる変数の一部への値の代入

積項$F_i(\mathbf{x})$に含まれる変数の一部に値を代入して得られる積項を，次式で表わす．

$$\tilde{F}_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\tilde{\varepsilon}_{ij}} \tag{3.19}$$

変数$x_j$に与える値を$c_j$とし，$x_j$に値が与えられないことを$c_j = u$で表わすことにする．$\varepsilon_{ij}$と$c_j$との値の組み合せに対する$\tilde{\varepsilon}_{ij}$の値は表3.1に与えられる．

表3.1によれば，任意の項$F_i(\mathbf{x})$の内部表現と$\mathbf{x}$の内部表現とから，$\tilde{F}_i(\mathbf{x})$の内部表現を

導びくことができる．たとえば，積項$F_i(x) = x_1 \cdot \bar{x}_3 \cdot x_5$に対して値$(c_1, \cdots, c_5) = (1, 0, u, 1, 0)$が与えられた時，得られる項は，$F_i(x) = 1 \cdot \bar{x}_3 \cdot 0$となる．よって，$F_i(x)$の内部表現は，$\sigma_i = 00100$, $\tau_i = 00001$となる．

表3.1　$\varepsilon_{ij}$と$c_j$の値に対する$\tilde{\varepsilon}_{ij}$の値

| $\varepsilon_{ij}$ ＼ $c_j$ | 0 | 1 | u |
|---|---|---|---|
| 0 $(\bar{x}_j)$ | t | f | 0 |
| 1 $(x_j)$ | f | t | 1 |
| t (1) | t | t | t |
| f (0) | f | f | f |

一般に，$\tilde{F}_i(x)$の内部表現$\tilde{\sigma}_i$, $\tilde{\tau}_i$は，$F_i(x)$の内部表現$\sigma_i$, $\tau_i$および$x$の内部表現$\xi$, $\theta$から，次式によって導かれる．（付録2(1)参照）

$$\begin{cases} \tilde{\sigma}_i = \xi \wedge \sigma_i \\ \tilde{\tau}_i = (\theta \wedge \sigma_i) \oplus \tau_i \end{cases} \tag{3.20}$$

ここに，$\wedge$および$\oplus$は，それぞれ論理積および排他的論理和を表わす．ただし，各演算は要素（桁）ごとに行なわれるとする．

さらに，$F_i(x) = 0$あるいは$F_i(x) = 1$となるか否かは，次のようにして知られる．（付録2(3)参照）

(i)　$F_i(x) = 0 \Leftrightarrow (\neg \sigma_i) \wedge \tau_i \neq 0$　(3.21)

(ii)　$F_i(x) = 1 \Leftrightarrow \sigma_i = 0$かつ$\tau_i = 0$　(3.22)

ここに，$\neg$は，各要素ごとの否定を表わし，0は，整数0の2進数表現（00…0）を表わす．

(3)　$Y_i(x)$に含まれる変数の一部への値の代入

積項$Y_i(x)$に含まれる変数の一部に値を代入して得られる積項を次式で表わす．

$$\tilde{Y}_i(x) \triangleq \prod_{j=1}^{n} x_j^{\tilde{\delta}_{ij}}$$

$\delta_{ij}$と変数$x_j$に与えられる値$c_j$との組み合わせに対する$\tilde{\delta}_{ij}$の値は，表3.2に与えられる．積項$\tilde{Y}_i(x)$の内部表現$\tilde{\eta}_i$は，$Y_i(x)$の内部表現$\eta_i$および$x$の内部表現$\xi$, $\theta$のうち$\xi$を用いて次式によって導かれる．（付録2(2)参照）

図3.2　$\delta_{ij}$と$c_j$の値に対する$\tilde{\delta}_{ij}$の値

| $\delta_{ij}$ ＼ $c_j$ | 0 | 1 | u |
|---|---|---|---|
| 0 (1) | 0 | 0 | 0 |
| 1 $(x_j)$ | * | 0 | 1 |

$\tilde{\delta}_{ij} = *$は，$\tilde{Y}_i(x) = 0$であることを表わす．

$$\tilde{\eta}_i = \xi \wedge \eta_i \tag{3.24}$$

また，$Y_i(x) = 0$あるいは$Y_i(x) = 1$となるか否かは，それぞれ以下のようにして知られる．（付録2(4)参照）

(i)　$Y_i(x) = 0 \Leftrightarrow \neg(\xi \vee \theta) \wedge \eta_i \neq 0$　(3.25)

(ii)　$Y_i(x) = 1 \Leftrightarrow [\neg(\xi \vee \theta) \wedge \eta_i = 0] \wedge [\tilde{\eta}_i = 0]$　(3.26)

(4)　$F_i(\mathbf{x})$ が単一の変数からなるか否かの判定：

与えられた積項 $F_i(\mathbf{x})$ の内部表現 $(\sigma_i, \tau_i)$ のうち $\sigma_i$ が次の性質をみたせば，$F_i(\mathbf{x})$ は単一の変数からなる．

$$\sigma_i = 2^{j-1} \quad \text{for some } j,\ 1 \leq j \leq n \tag{3.27}$$

この性質は，次節で述べるハッシングの技法を用いた数式処理アルゴリズムで有効に用いられる．

## 3．3　数式処理アルゴリズム

この節では，3.2.2 で述べた形式で内部表現されたブール関数および擬似ブール関数に対して，3.2.1 で述べた処理をほどこすアルゴリズムを示す．このアルゴリズムはハッシングの技法を用いており，効率良く処理を行うことができる．さらに，このアルゴリズムを，線形探策にもとづく従来のアルゴリズムと比較することで，その特徴を明らかにする．

### 3．3．1　擬似ブール関数の処理アルゴリズム

擬似ブール関数 $y(\mathbf{x})$ の内部表現 $\{(\eta_1, a_1), (\eta_2, a_2), \cdots, (\eta_m, a_m)\}$ を，$\eta_i$ をキーとするハッシュ・テーブルとして実現する．ハッシュ関数としては，次式で定義される $h_p$ を用いる．同一のハッシュ・アドレスを持つ異なる $\eta_i$ に対しては，線形リストを構成する．

$$h_p(\eta) \triangleq \mathrm{mod}\ (\eta, p) + 1 \tag{3.28}$$

ここに，p は 2 を原始根とする素数である．

式（3.7）で与えられる $\tilde{y}(\mathbf{x})$ の簡単化は，以下に示されるアルゴリズム 3.1 によって行なわれる．このアルゴリズムで用いられる $y(\mathbf{x})$ のデータ構造を図 3.1 に示す．

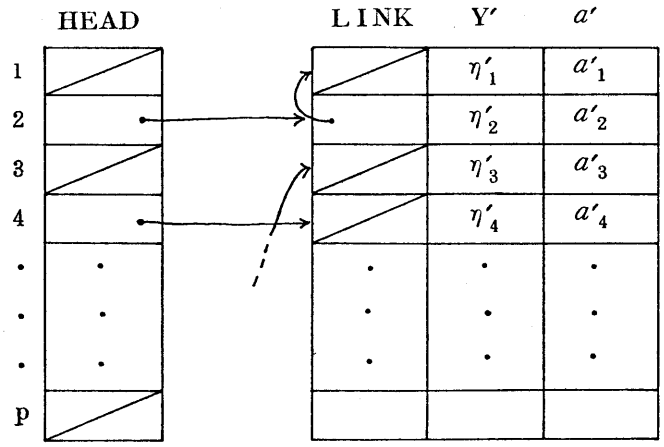



図3．1　ハツシユ・テーブルによる y(x) の実現

〔アルゴリズム 3.1〕(ハッシング)

入力：$\tilde{y}(\mathbf{x}) = \sum_{i=1}^{m} a_i \tilde{Y}_i(\mathbf{x})$ (3.29)

出力：$\tilde{y}(\mathbf{x})$を簡単化して得られる式，

$$y'(\mathbf{x}) = c' + \sum_{i=1}^{\ell} a_i Y_i(\mathbf{x}) \quad (3.30)$$

方法：

Step 1：〔初期設定〕

$i \leftarrow 1$，$\ell \leftarrow 0$，$c' \leftarrow 0$；

HEAD($j$) $\leftarrow 0$ for $1 \leq j \leq n$；Step 2へ；

Step 2：〔$\tilde{Y}_i(\mathbf{x})$が定数か否かの判定〕

(i) $\tilde{Y}_i(\mathbf{x}) = 0$ならば，Step 5へ；

(ii) $\tilde{Y}_i(\mathbf{x}) = 1$ならば，$c' \leftarrow c' + a_i$としてStep 5へ；

Step 3：〔同一項の除去〕

Step 3.1：$j \leftarrow hp(\tilde{\eta}_i)$，$k \leftarrow$ HEAD($j$)とする；

Step 3.2：(i) $k = 0$ならば，Step 4 へ；

(ii) $k \neq 0$ならば，Step 3.3へ；

Step3.3 (i) $Y'_k(\mathbf{x}) \equiv \tilde{Y}_i(\mathbf{x})$ならば，

$a'_k \leftarrow a'_k + a_i$としてStep 5へ；

(ii) $Y'_k(\mathbf{x}) \not\equiv \tilde{Y}_i(\mathbf{x})$ならば，

$k \leftarrow$ LINK($k$)としてStep 3.2へ；

Step 4：〔項の格納〕

$\ell \leftarrow \ell + 1$，$Y'_\ell(\mathbf{x}) \leftarrow Y_i(\mathbf{x})$，$a'_\ell \leftarrow a_i$，

LINK($\ell$) $\leftarrow$ HEAD($j$)，HEAD($j$) $\leftarrow \ell$ として，Step 5へ；

Step 5：〔アルゴリズムの終了の判定〕

$i \leftarrow i + 1$とする；

(i) $i \leq m$ならば，Step 2へ；

(ii) $i > m$ならば，アルゴリズム終了．

ハッシングの技法を用いたこのアルゴリズムに対して，線形探索にもとづく処理アルゴリズムを以下に示す．このアルゴリズムは，アルゴリズム3.1のStep 1，Step 3，およびStep 4を変更することにより得られる．

〔アルゴリズム 3.2 〕( 線形探索 )

入力: $\tilde{y}(x)=\sum_{i=1}^{m} a_i \tilde{Y}_i(x)$ (3.31)

出力: $\tilde{y}(x)$ を簡単化して得られる式,

$$y'(x) = c' + \sum_{i=1}^{\ell} a'_i Y'_i(x) \quad (3.32)$$

方法:

Step 1 :〔初期設定〕

$i \leftarrow 1$, $\ell \leftarrow 0$, $c' \leftarrow 0$ として, Step 2へ;

Step 2 :〔 $\tilde{Y}_i(x)$ が定数か否かの判定〕

(i) $\tilde{Y}_i(x) = 0$ ならば, Step 5へ;

(ii) $\tilde{Y}_i(x) = 1$ ならば, $c' \leftarrow c' + a_i$ としてStep 5へ;

(iii) それ以外のとき, Step 3へ;

Step 3 :〔同一項の除去〕

Step 3.1 : (i) $\ell = 0$ ならば, Step 4へ;

(ii) $\ell \neq 0$ ならば, $k \leftarrow 1$ としてStep 3.2へ;

Step 3.2 : (i) $Y'_k(x) \equiv \tilde{Y}_i(x)$ ならば, $a'_k \leftarrow a'_k + a_i$ としてStep 5へ;

(ii) $Y'_k(x) \not\equiv \tilde{Y}_i(x)$ ならば, Step 3.3へ;

Step 3.3 : $k \leftarrow k + 1$ とする;

(i) $k \leq \ell$ ならば, Step 3.2へ;

(ii) $k > \ell$ ならば, Step 4へ;

Step 4 :〔項の格納〕

$\ell \leftarrow \ell + 1$, $Y'_\ell(x) \leftarrow \tilde{Y}_i(x)$, $a'_\ell \leftarrow a_i$ としてStep 5へ;

Step 5 :〔アルゴリズムの終了の判定〕

$i \leftarrow i + 1$ とする;

(i) $i \leq m$ ならば, Step 2へ;

(ii) $i > m$ ならば, アルゴリズム終了.

### 3.3.2 ブール関数の処理アルゴリズム

ブール関数F(x)の数式処理は, 擬似ブール関数の数式処理と同様に, ハッシングの技法を用いて効率良く行なわれる. ただし, この場合には, $F_i(x)$ の内部表現 $(\sigma_i, \tau_i)$ のうちの $\sigma_i$ をキーとして検索を行なう. また, ブール関数の処理では, 積項が単一の変数からなるか否か

の判定が行なわれる．この判定は，式（3.27）と，整数の剰余類に関する次の性質[5]を用いて容易に実現される．

〔性質3.1〕　pを，2を原始根とする素数とする．このとき，次の式が成り立つ．

$$2^i \not\equiv 2^j \bmod p \quad \text{if} \quad i \neq j,\ 1 \leq i,\ j \leq p. \tag{3.33}$$

性質3.1，式（3.27）およびハッシュ関数$h_p$の定義から，異なる単一の変数からなる積項は，異なるハッシュ・アドレスを持つことは明らかである．

与えられた積項$F_i(\mathbf{x})$が単一の変数からなるか否かを判定するアルゴリズムは，次のように与えられる．

〔アルゴリズム3.3〕

**入力**：積項$F_i(\mathbf{x})$の内部表現$\sigma_i$

**出力**：積項$F_i(\mathbf{x})$が単一の変数からなるとき（$F_i(\mathbf{x}) \equiv x_j$ or $\bar{x}_j$），変数のインデックス$j$を出力する．積項$F_i(\mathbf{x})$が単一の変数からなる項ではないとき，$j=0$を出力する．

**記法**：pを，$n$以上の2を原始根とする素数とする．

2つの配列BASE及びINDEXの各要素は，それぞれ，次の値を保持しているものとする．

$$\text{BASE}(i) \triangleq \begin{cases} 2^{j-1} & ;\ \text{if} \quad i = h_p(2^{j-1}) \\ 0 & ;\ \text{それ以外のとき} \end{cases} \tag{3.34}$$

$$\text{INDEX}(i) \triangleq \begin{cases} j & ;\ \text{if} \quad i = h_p(2^{j-1}) \\ 0 & ;\ \text{それ以外のとき} \end{cases} \tag{3.35}$$

ここに，$1 \leq i \leq p$，$1 \leq j \leq n$である．

**方法**：$k \leftarrow h_p(\sigma_i)$とする．

(i) BASE$(k) = \sigma_i$ならば，$j \leftarrow$ INDEX$(k)$とする．

(ii) BASE$(k) \neq \sigma_i$ならば，$j \leftarrow 0$とする．

アルゴリズム3.3で計算された$F_i(\mathbf{x})$に対するハッシュ・アドレスは，そのまま，以下に示すブール関数の数式処理アルゴリズムで用いることができる．

〔アルゴリズム3.4〕

**入力**：$\tilde{F}(\mathbf{x}) = \bigvee_{i=1}^{s} \tilde{F}_i(\mathbf{x})$　　(3.36)

**出力**：$\widetilde{F}(x)$ を簡単化して得られる式，

$$F'(x) \triangleq \bigvee_{i=1}^{t} F'_i(x) \qquad (3.37)$$

**方法**：

Step 1：〔初期設定〕

$i \leftarrow 1$，$t \leftarrow 0$，flag $\leftarrow 0$；

HEAD$(j) \leftarrow 0$ for $1 \leq j \leq p$ とする；Step 2へ；

Step 2：〔項が定数か否かの判定〕

(i) $\widetilde{F}_i(x) = 0$ ならば，Step 6へ；

(ii) $\widetilde{F}_i(x) = 1$ ならば，Step 7へ；

(iii) それ以外のとき，Step 3へ；

Step 3：〔項が単一の変数からなるか否かの判定〕

(i) $F_i(x)$ が単一の変数 $x_j$ からなる項 $x_j^{\widetilde{\varepsilon}_{ij}}$ ならば，x に $x_j = \neg \widetilde{\varepsilon}_{ij}$ を代入する；

flag $\leftarrow 1$ として，Step 6へ；

(ii) $F_i(x)$ が単一の変数からなる項ではないならば，Step 4へ；

Step 4：〔同一項の除去〕

Step 4.1：$j \leftarrow h_p(\widetilde{\sigma}_i)$；$k \leftarrow$ HEAD$(j)$ とする；

Step 4.2：(i) $k = 0$ ならば，Step 5へ；

(ii) $k \neq 0$ ならば，Step 4.3へ；

Step 4.3：(i) $F'_k(x) \equiv \widetilde{F}_i(x)$ ならば，Step 6へ；

(ii) $F'_k(x) \not\equiv \widetilde{F}_i(x)$ ならば，

$k =$ LINK$(k)$ としてStep 4.2へ；

Step 5：〔項の格納〕

$t \leftarrow t + 1$，$F'_t(x) \leftarrow \widetilde{F}_i(x)$，

LINK$(t) \leftarrow$ HEAD$(j)$，HEAD$(j) \leftarrow t$，Step 6へ；

Step 6：〔アルゴリズムの終了の判定〕

$i \leftarrow i + 1$ とする；

(i) $i \leq m$ ならば，Step 2へ；

(ii) $i > m$ かつ flag $= 1$ ならば，

$\widetilde{F}_i(x) \leftarrow F'_i(x)$ for $1 \leq i \leq t$，$s \leftarrow t$ としてStep 1へ；

(iii) $i > m$ かつ flag=0 ならば，アルゴリズム終了；

Step 7：〔$F'(x) = 1$〕

$F'(x) \leftarrow 1$ とする；アルゴリズム終了．

上記のアルゴリズムで，変数flagは，式の簡単化の途中で単一の変数からなる項を処理する時に，値1が代入される．このとき，擬似ブール計画問題の制約条件が$F(x)=0$であることから，その項を形成する変数の値が一意に定まる．その結果，その時点までに処理された項，$F'_i(x)$，$(1 \leq i \leq t)$もさらに簡単化される可能性がある．よって，単一の変数からなる項が処理された場合には，$F'(x)$全体を再び評価しなおす必要がある．また，Step 7が実行されるのは，Step 2で，$F_i(x)=1$であるような項を処理した場合である．$F'(x)$は積和形のブール関数であるから，この場合には，ただちに$F'(x) \leftarrow 1$としてよい．

上記のアルゴリズム3.4では，ハッシングの技法を用いて，ブール関数の簡単化の処理を行なっている．線形探索にもとづいてブール関数の簡単化を行なうアルゴリズムは，アルゴリズム3.4のStep 1，Step 4，Step 5を変更することによって以下のように与えられる．

〔アルゴリズム3.5〕（線形探索）

**入力**：$\tilde{F}(x) = \bigvee_{i=1}^{s} \tilde{F}_i(x)$ （3.38）

**出力**：$\tilde{F}(x)$を簡単化して得られる式，

$$F'(x) = \bigvee_{i=1}^{t} F'_i(x) \quad (3.39)$$

**方法**：

Step 1：〔初期設定〕

$i \leftarrow 1$，$t \leftarrow 0$，flag $\leftarrow 0$；Step 2へ；

Step 2：〔項が定数か否かの判定〕

(i) $\tilde{F}_i(x)=0$ならば，Step 6へ；

(ii) $\tilde{F}_i(x)=1$ならば，Step 7へ；

(iii) それ以外のとき，Step 3へ；

Step 3：〔項が単一の変数からなるか否かの判定〕

(i) $\tilde{F}_i(x)$が単一の変数$x_j$からなる項$x_j^{\tilde{\varepsilon}_{ij}}$ならば，xに$x_j = \neg\tilde{\varepsilon}_{ij}$を代入する；

flag $\leftarrow 1$として，Step 6へ；

(ii) $\tilde{F}_i(x)$が単一の変数からなる項ではないならば，Step 4へ；

Step 4.1：(i) $t=0$ならば，Step 5へ；

(ii) $t \neq 0$ならば，$k \leftarrow 1$として，Step 4.2へ；

Step 4.2：(i) $F'_k(x) \equiv \tilde{F}_i(x)$ならば，Step 6へ；

(ii) $F'_k(x) \not\equiv \tilde{F}_i(x)$ならば，Step 4.3へ；

Step 4.3：$k \leftarrow k+1$とする；

(i) $k \leq t$ ならばStep 4.2へ；

(ii) $k > t$ ならばStep 5へ；

Step 5：〔項の格納〕

$t \leftarrow t+1$，$F't(x) \leftarrow \widetilde{F}i(x)$ とする；

Step 6へ；

Step 6：〔アルゴリズムの終了の判定〕

$i \leftarrow i+1$ とする；

(i) $i \leq m$ ならば，Step 2へ；

(ii) $i > m$ かつ flag = 1 ならば，$\widetilde{F}i(x) \leftarrow F'i(x)$ for $1 \leq i \leq t$，$s \leftarrow t$ として，Step 1へ；

(iii) $i > m$ かつ flag=0 ならば，アルゴリズム終了；

Step 7：〔$F'(x) = 1$〕

$F'(x) \leftarrow 1$ とする；アルゴリズム終了.

### 3. 4 アルゴリズムの計算時間の理論的解析

この節では，前節で述べた数式処理アルゴリズムの計算時間の理論的評価式を示す．さらに，全ての項が異なる場合について，それぞれのアルゴリズムの計算時間の上限を求める．ただし，数式処理の内容については，擬似ブール関数の処理とブール関数の処理とが本質的には同じであるので，以下の考察では，主として擬似ブール関数の処理アルゴリズムに関して行なう．

#### 3. 4. 1 アルゴリズム3. 1，3. 2の計算時間

式（3.3）で与えられる擬似ブール関数 $y(x)$ に含まれる変数の一部に0または1の値を与え，さらに，アルゴリズム3.1によって簡単化を行なって，次式が得られたとする.*
ただし，以下では，初めに与えられる $Yi(x)$，（$1 \leq i \leq m$）の中には，$\widetilde{Y}i(x) = 0$ 又は $\widetilde{Y}i(x) = 1$ となる項は含まれないとする．

$$y'(x) \triangleq \sum_{i=1}^{\ell} a'i \ Y'i(x) \qquad (3.40)$$

このとき，アルゴリズム3.1，アルゴリズム3.2の計算時間の一般的評価式は，次のように与えられる．

$$T = t_1 + m \cdot t_2 + T_3 + \ell \cdot t_4 + m \cdot t_5 \qquad (3.41)$$

*）同一の入力に対して，アルゴリズム3.1とアルゴリズム3.2の出力は明らかに一致する.

ここに，$t_1$, $t_2$, $t_4$, $t_5$は，それぞれ，アルゴリズムのStep1, Step2, Step4, Step5を1回実行するのに要する時間（$m$，$\ell$に依存しない定数）とする．$T_3$は，アルゴリズムが終了するまでに要するStep 3の計算時間の総和を表わす．$T_3$の値は，処理される擬似ブール関数$y$（x）の性質に強く依存するので，その一般的評価は困難である．

また，$t_1$の値は，アルゴリズム3.2では定数であるが，アルゴリズム3.1では，ハッシュ・テーブルの大きさpの一次式として次のように与えられる．

$$t_1 = a \cdot p + b \qquad (3.42)$$

ここに，$a$，$b$は定数である．

### 3.4.2 アルゴリズム3.1，3.2の計算時間の上限

アルゴリズム3.1およびアルゴリズム3.2の計算時間が最大となるのは，$\tilde{y}$（x）が次の条件を満たす場合である．

〔条件3.1〕

$$\tilde{y}(x) = \sum_{i=1}^{m} a_i \tilde{Y}_i(x) \qquad (3.43)$$

とする．この時，各$\tilde{Y}_i$（x）は以下の式を満足する．

(i) $\tilde{Y}_i \neq 0 \ \wedge \ \tilde{Y}_i \neq 1 \quad \text{for } 1 \leq i \leq m$ (3.44)

(ii) $\tilde{Y}_i(x) \equiv \tilde{Y}_j(x) \quad \text{if} \quad i \neq j,\ 1 \leq i,\ j \leq m$ (3.45)

条件3.1がなり立つ場合の，アルゴリズム3.1およびアルゴリズム3.2の計算時間の上限は，以下のようにして求められる．

(1) アルゴリズム3.1の計算時間の上限

アルゴリズム3.1に関して，次の仮定を置く．

〔仮定3.1〕 ハッシュ・テーブルの要素HEAD（$j$）を始点とする線形リストの長さの期待値は，アルゴリズムの各時点で，$j$の値にかかわらず一様である．

条件3.1および仮定3.1のもとでの，Step3の$i$回目の実行の際の積項間の比較回数の期待値は（$i$−1）/pで与えられる．よって，Step 3の$i$回目の計算時間の期待値は，c，dを定数として，次式で与えられる．

$$c \cdot (i-1)/p + d \qquad (3.46)$$

よって，Step 3の計算時間の総和$T_3$ の期待値$<T_3>$は，次式で与えられる．

$$<T_3> = \sum_{i=1}^{m} [\, c \cdot (i-1)/p + d \,]$$

$$= c \cdot (m-1) \cdot (m-2)/(2p) + d \cdot m \qquad (3.47)$$

式（3.41），（3.42）および（3.47）より，アルゴリズム全体の計算時間の期待値$<T>$は，次式で与えられる．

$$<T> = a \cdot p + b + m \cdot t_2 + c \cdot (m-1) \cdot (m-2)/(2p) + d \cdot m + m \cdot t_4 + m \cdot t_5$$

$$= c \cdot (m-1) \cdot (m-2)/(2p) + (t_2 + t_4 + t_5 + d) \cdot m + a \cdot p + b \qquad (3.48)$$

$<T>$を最小にするpの値$p^*$ は，

$$\partial <T>/\partial p = -c \cdot (m-1) \cdot (m-2)/(2p^2) + a = 0 \qquad (3.49)$$

をpについて解くことにより，次式で与えられる．

$$p^* = \sqrt{c \cdot (m-1) \cdot (m-2)/(2a)} \qquad (3.50)$$

ここで，さらに，$m \gg 1$と仮定すると，$\lambda = \sqrt{c/(2a)}$として，$p^*$ は次式で近似される．

$$p^* \approx \lambda \cdot m$$

この時，$<T>$の最小値$<T>^*$ は，次式で与えられる．

$$<T>^* \approx (\sqrt{2ac} + t_2 + t_4 + t_5 + d) \cdot m + b \qquad (3.52)$$

式（3.52）より，アルゴリズム3.1の計算時間の上限が$O(m)$であることが知られる．

このことにより，与えられた擬似ブール関数$g(x)$の項数$m$に応じて，pを動的に$p^*$に設定するようにアルゴリズム3.1を修正して得られるアルゴリズムを，アルゴリズム3.1*と呼ぶことにする．

(2) アルゴリズム3.2の計算時間の上限

条件3.1のもとでは，アルゴリズム3.2のStep 3の$i$回目の実行の際の積項間の比較回数は$i-1$である．よって，Step 3の$i$回目の実行に要する計算時間はe，$g$を定数として，次式で与えられる．

$$e \cdot i + g \qquad (3.53)$$

よって，Step 3の計算時間の総和は次式で与えられる．

$$T_3 = \sum_{i=1}^{m} (e \cdot i + g)$$

$$= e \cdot m(m-1)/2 + g \cdot m \qquad (3.54)$$

ゆえに，アルゴリズム3.2の計算時間の上限は，次式で与えられる．

$$T = t_1 + m \cdot t_2 + e \cdot m(m-1)/2 + g \cdot m + m \cdot t_4 + m \cdot t_5$$

$$= e \cdot m \cdot (m-1)/2 + (t_2 + t_4 + t_5 + g) \cdot m + t_1 \qquad (3.55)$$

よって，$m \gg 1$ の場合には，アルゴリズム 3.2 の計算時間の上限は $O(m^2)$ であることが知られる．

### 3.4.3　アルゴリズム 3.4，3.5 の計算時間

ここでは，ブール関数の処理と擬似ブール関数の処理との違いについて述べ，ブール関数の処理アルゴリズムの計算時間について簡単な考察を行なう．

3.2.1 および 3.3.2 で述べたように，ブール関数の処理内容は，主に次の 2 点で擬似ブール関数の処理内容と異なっている．すなわち，

(i) 単一の変数からなる項がなくなるまで，すなわちアルゴリズム 3.4，3.5 で flag = 0 となるまで，式全体の処理をくり返さなければならない．

(ii) $F_i(x) \equiv 1$ である項がみつけられた時には，ただちに $F'(x) \leftarrow 1$ として，アルゴリズムを終了してよい．

そのため，ブール関数の処理アルゴリズムの計算時間の解析は，擬似ブール関数の処理アルゴリズムの計算時間の解析よりも一般に困難となる．

たとえば，次の式（3.56）のような $F(x)$ に対して，式全体の処理のくり返し回数は，最悪の場合 $s$ 回必要である．

$$\tilde{F}(x) = \bar{x}_1 x_2 \vee \bar{x}_2 x_3 \vee \cdots \vee \bar{x}_i x_{i+1} \vee \cdots \vee \bar{x}_{s-1} x_s \vee \bar{x}_s \qquad (3.56)$$

この $F(x)$ を処理するためには，アルゴリズム 3.5 で $O(s^3)$ 回の項の比較が必要となる．

しかしながら，式全体の処理の 1 回のくり返しに着目すると，ブール関数の処理内容は，上記の(ii)の点を除いて，本質的には同じである．よって，擬似ブール関数に対して置いた条件 3.1 と同様な以下の条件のもとで，ブール関数の，式全体の処理の 1 回のくり返しに要する計算時間は最大となる．

〔条件 3.2〕

$$\tilde{F}(x) = \bigvee_{i=1}^{s} \tilde{F}_i(x) \qquad (3.57)$$

とする．この時，各 $\tilde{F}_i(x)$ は以下の条件を満足する．

(i) $\tilde{F}_i(x) \neq 0 \wedge \tilde{F}_i(x) \neq 1$ for $1 \leq i \leq s$　　(3.58)

(ii) $\tilde{F}_i(x)$，$1 \leq i \leq s$ は，全て 2 個以上の変数からなる

(iii) $\tilde{F}_i(x) \not\equiv \tilde{F}_j(x)$ if $i \neq j$，$1 \leq i, j \leq s$　　(3.59)

条件 3.2 のもとでは，単一の変数からなる項は存在しないため，式全体の処理は 1 回だけ行

なわれる．また，この条件のもとでは，ブール関数の処理時間に関して，3.4.2で述べた，擬似ブール関数の処理時間に関する結果と同様の結果を得る．

まず，線形探索にもとづくアルゴリズム3.5の計算時間は，$O(s^2)$である．次に，ハッシングの技法を用いたアルゴリズム3.4では，ハッシュ・テーブルの大きさpをアルゴリズム3.1の場合と同様に適当に定めることにより，その計算時間は$O(s)$となることが知られる．

以下では，与えられたブール関数の項数$s$に応じてpの値を動的に最適値p*に設定するようにアルゴリズム3.4を修正して得られるアルゴリズムを，アルゴリズム3.4*と呼ぶことにする．

## 3.5 実験およびその結果

この節では，アルゴリズム3.1およびアルゴリズム3.2の効率を評価するために行なった，2つの実験とその結果について述べる．

実験1では，前節で与えられたアルゴリズム3.1とアルゴリズム3.2の，計算時間の評価が妥当であることを確認する．また，両アルゴリズムの計算時間の比較を行なう．

次に，実験2では，数式処理をハッシングの技法を用いて行なう場合と線形探索にもとづいて行なう場合とについて，2通りの擬似ブール計画法の求解システムを実現し，それらの効率の比較を行なう．

### 3.5.1 実験 1

一様乱数を用いて，条件3.1をみたす30変数の擬似ブール関数$y(\mathbf{x})$を発生させた．この$y(\mathbf{x})$を，アルゴリズム3.1およびアルゴリズム3.2によって処理し，それぞれのアルゴリズムの計算時間を測定した．ただし，$y(\mathbf{x})$の項数$m$は，$m=100, 200, 300, 400, 500$とした．また，アルゴリズム3.1については，$m$の各値に対して，各々20通りの$y(\mathbf{x})$を発生させ，その計算時間を測定した．

図3.2にpの値を変化させた場合の，アルゴリズム3.1の平均計算時間を示す．同図より，アルゴリズム3.1の計算時間の上限の評価式（式（3.48））が妥当であることが知られる．すなわち，式（3.48）で，

$$\begin{cases} k_0 = c\cdot(m-1)\cdot(m-2)/2 \\ k_1 = a \\ k_2 = (t_2+t_4+t_5+d)\cdot m+b \end{cases} \qquad (3.60)$$

---

*）条件3.1が満たされていれば，アルゴリズム3.2の計算時間は，$y(\mathbf{x})$の項数$m$のみに依存し，$y(\mathbf{x})$の形にはよらない．

とおくと，次式の関数形を得る．

$$<T> = k_0 / p + k_1 \cdot p + k_2 \qquad (3.61)$$

この式は，図3.2の各曲線を適切に表現していると考えられる．従って，式（3.48）に基づいて行なったアルゴリズム3.1* の存在の予測は，正しかったことが結論される．

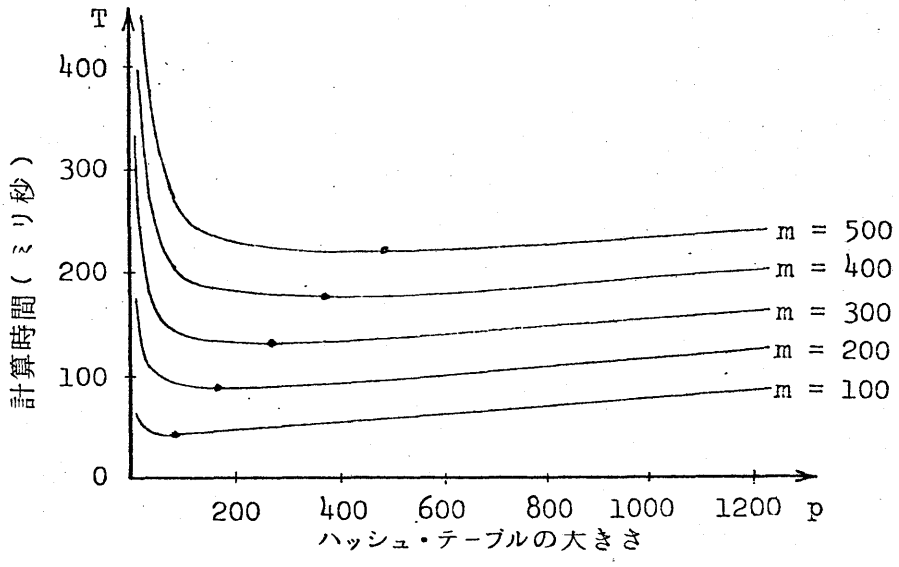



図3.2　アルゴリズム3.1の計算時間
（図中の・は，計算時間の最小値を表わす）

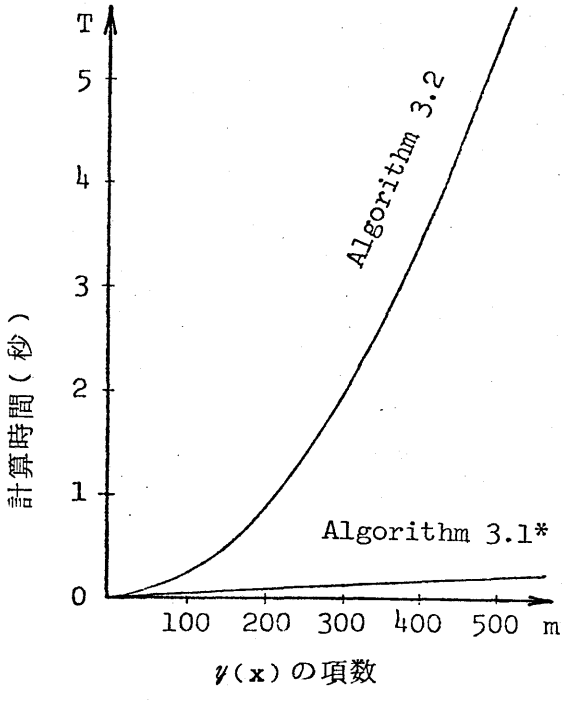



図3.3　アルゴリズム3.1* とアルゴリズム3.2の計算時間の比較

そこで，図 3. 2 において，最小の平均計算時間を与える p の値にもっとも近い，2 を原始根とする素数を，各 $m$ の値について求め，これをアルゴリズム 3. 1 * の p* の値とした．アルゴリズム 3. 1 * の平均計算時間およびアルゴリズム 3. 2 の計算時間を表 3. 3 に示す．またこれをグラフ化したものを図 3. 3 に示す．同図より，アルゴリズム 3. 1* およびアルゴリズム 3. 2 の計算時間がそれぞれ $O(m)$, $O(m^2)$ であることが知られる．このことから，計算時間の上限の評価式が妥当であることが知られる．また，アルゴリズム 3. 1* を用いることによって，特に $m$ の値が大きい場合の計算時間が大幅に改善（ $m = 500$ で，約 4 %に ）されることが知られる．

表3.3　アルゴリズム3.1*とアルゴリズム3.2の計算時間の比較

| m | p* | 計算時間　（ミリ秒） | | 比　(%) |
|---|---|---|---|---|
| | | Alg.　3.1* | Alg.　3.2. | |
| 100 | 83 | 44 | 236 | 18.6 |
| 200 | 163 | 88 | 898 | 9.8 |
| 300 | 269 | 132 | 1,984 | 6.7 |
| 400 | 373 | 175 | 3,501 | 5.0 |
| 500 | 491 | 219 | 5,442 | 4.0 |

m ： $y$ ( x ) の項数
p*: ハッシュ・テーブルの大きさの最適値

### 3.5.2　実　験　2

数式処理アルゴリズムの実現方法が互いに異なる，次の 2 つの擬似ブール計画法の求解ソフトウェア・システムを実現した．システム 3. 1 は，ブール関数および擬似ブール関数の処理を，それぞれアルゴリズム 3. 1 * およびアルゴリズム 3. 4* を用いて行なう．一方，システム 3. 2 は，これらの関数の数式処理を，それぞれアルゴリズム 3. 2 およびアルゴリズム 3. 5 を用いて行なう．一様乱数を用いて発生された 30 変数の擬似ブール計画問題に対する，両システムの計算時間を表 4. 4 に示す．同表で $s$ ，$m$ は，それぞれ制約条件式 F ( x ) の項数および目的関数 $y$(x) の項数を表わす．また，表中の各値は，$s$ ，$m$ の各組み合せについて，それぞれ 20 問題を解いたときの平均値である．

表3.4　システム3.1とシステム3.2との計算時間の比較とシステムの改善度

| s | m / システム | 20 | | | 60 | | | 100 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 3.2 | 3.1 | I | 3.2 | 3.1 | I | 3.2 | 3.1 | I |
| 0 | 全　体 | 3.41 | 3.40 | 0.3 | 95.5 | 88.8 | 7.2 | 414.0 | 360.0 | 12.6 |
| | 数式処理 | 0.386 | 0.379 | 4.2 | 12.7 | 6.05 | 53.2 | 90.9 | 37.1 | 61.1 |
| | 比率+ (%) | 11.2 | 10.9 | | 13.4 | 6.76 | | 20.7 | 9.41 | |
| 60 | 全　体 | 4.42 | 4.09 | 7.7 | 44.2 | 40.2 | 9.0 | 110.0 | 99.0 | 10.3 |
| | 数式処理 | 2.03 | 1.70 | 18.7 | 13.6 | 9.69 | 29.0 | 32.1 | 21.2 | 34.3 |
| | 比率+ (%) | 45.7 | 40.4 | | 31.1 | 24.3 | | 30.1 | 22.1 | |
| 120 | 全　体 | 7.23 | 4.41 | 39.3 | 16.2 | 12.1 | 27.3 | 20.4 | 15.4 | 26.1 |
| | 数式処理 | 5.33 | 2.51 | 53.3 | 9.07 | 4.80 | 48.4 | 9.98 | 5.00 | 53.0 |
| | 比率+ (%) | 73.9 | 56.7 | | 56.4 | 39.9 | | 49.4 | 31.2 | |

+) 比率：数式処理の時間の全体の時間に対する比率；
s：F(x)の項数；$m$：$y(x)$の項数；I：改善度

以下では，システムの改善度として，次式で定義されるIを用いる．

$$I \triangleq (T_2 - T_1)/T_2 \times 100\,(\%) \qquad (3.62)$$

ここに，$T_1$, $T_2$は，それぞれシステム3.1およびシステム3.2の計算時間を表わす．また，数式処理に要した時間の，システム全体の計算時間に対する割合をF／T比と呼ぶことにする．

図3.4にシステムの改善度を，図3.5にF／T比を示す．図3.4より，とくに数式処理に関しては改善度Iが目的関数の項数$m$の値の増大とともに増す傾向がみとめられる．これは，図

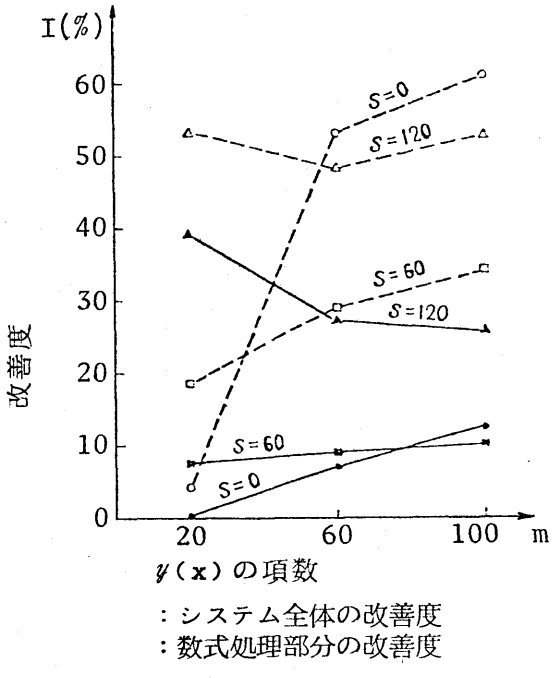



図3.4　システムの改善度

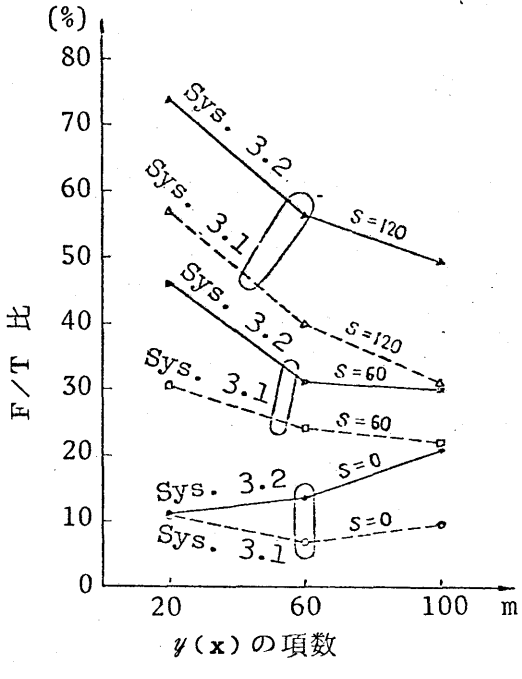



図3.5　F／T　比

3.2にもとづく予測と一致する．

擬似ブール計画法のアルゴリズムはBranch-and-Bound法にもとづいており，アルゴリズムの進行途中においては，原問題に含まれる変数の一部に値を与えることによって，いくつかの部分問題が生成される．あきらかに,これらの部分問題を構成する制約条件の項数および目的関数の項数は，それぞれ原問題の制約条件の項数および目的関数の項数より少ない．したがって，$m$の値の減少によってアルゴリズム3.1* の効果が減少する図3.2の効果を考慮すれば図3.4に示された処理時間の改善度は良好であるといえよう．

システム全体の改善度は，制約条件式の項数$s$が増大するにつれて増す傾向にあることが認められる．これは，数式処理部分の計算時間が全体の計算時間に占める割合，すなわちF/T比が$s$の増加にともなって減ずるためであることが図3.5より知られる．また，$s=120$ の場合に，システム全体の改善度が$m$の増大にともなって減少しているのは，$m$の増大にともなうF/T比の減少がこの場合とくに著しいためであることが同じく図3.5より知られる．

## 3.6 むすび

本章では，ハッシングの技法を用いて数式処理アルゴリズムを構成することにより，多数の項を持つブール関数ならびに擬似ブール関数を効率良く処理できることを示した．また，これらのアルゴリズムを用いて擬似ブール計画法のシステムを実現することにより，システム全体の計算時間もかなり改善されることを示した．

なお，本章での実験に用いた擬似ブール計画法のシステムでは，部分問題を生成するときには，必ず原問題に立ちかえって変数に値を与えるようにしている．この場合，一つ前の部分問題から[illegible]じい部分問題をつくるようにシステムを改良すれば，ハッシングを用いた効果はさらに有効になると考えられる．

また，アルゴリズム3.1* は，変数のべき乗を含む多項式の数式処理の範囲にまで容易に拡張できるので，この種の数式処理に関して，かなりの有効さを持つことが予想される．

本章の実験には，名古屋大学情報工学科のFACOM230/38が用いられた．また，本章に示したアルゴリズムの実現にはFORTRAN Ⅳ－Sが用いられた．

# 第4章　擬似ブール計画法のアルゴリズムにおける評価関数の強力さとアルゴリズムの計算時間の関係

## 4.1 まえがき

0－1計画問題などの整数計画問題をBranch-and-Bound法にもとづくアルゴリズムで解く場合，ほとんどの問題で，アルゴリズム中で処理される部分問題の個数は，問題に含まれる変数の個数に対して，指数関数的に増加する．そのため，これらの問題を解くのに要する計算時間も，含まれる変数の個数に対して指数関数的に増加する．

したがって，これらの問題を効率良く解くためには，アルゴリズム中で処理される部分問題の個数を減少させることが重要な問題となる．アルゴリズム中で処理される部分問題の個数を減少させるためには，効率良く問題を分割すること，部分問題のより厳密な下界（または上界）を用いること，および，未解決な部分問題の集合の中から，次に解くべき部分問題を巧みに選択すること，の3つの点が重要である．これらの目的のために，しばしば，ヒューリスティクスを用いた評価関数が用いられる．

しかしながら，一般に，より強力な評価関数は，より多くの計算時間を必要とする．そのために，必ずしも，強力な評価関数の導入がアルゴリズム全体の計算時間を短縮するとは限らない．それ故，どの程度強力な評価関数を用いるかが，この種のアルゴリズムの効率を決定する要因となる．

本章では，擬似ブール計画法のアルゴリズムで用いられる評価関数の強力さとアルゴリズム全体の計算時間との関係について述べる．すなわち，まず，問題の分割に用いられる変数の選択を行なうのに用いられる評価関数の強力さを変化させ，アルゴリズム中での問題の分割回数と全体の計算時間の変化を調べる．次に，目的関数を構成する積項間の含意の関係を用いてより厳密な目的関数の下界を与える手続きを用いた場合の，アルゴリズム中での問題の分割回数と全体の計算時間の変化を調べる．

以下では，初めに，4.2で擬似ブール計画問題の定式化を行なう．次に，4.3では，含意の関係を用いた，目的関数の下界の評価方法について述べる．4.4では，問題の分割方法について述べる．4.5では，実験に用いられる擬似ブール計画法のアルゴリズムを記述する．さらに，4.6では，4.3および4.4で与えられた評価関数の強力さとアルゴリズム全体の計算時間との関係について考察する．最後に，4.7では，4.5で与えられたアルゴリズムを実現し，評価関数の強力さを変化させたときの，アルゴリズムによる問題分割の回数と，アルゴリズムの計算時間との変化を測定した結果について述べる．

## 4.2 擬似ブール計画問題の定式化*

まず，

$$x_j \in \{0, 1\}, \quad \text{for} \quad 1 \leq j \leq n \tag{4.1}$$

$$\mathbf{x} \triangleq (x_1, x_2, \cdots, x_n) \tag{4.2}$$

とする．

$\mathbf{x}$を変数とするブール関数$F(\mathbf{x})$を次式のように表わす

$$\begin{cases} F(\mathbf{x}) \triangleq \bigvee_{i=1}^{s} F_i(\mathbf{x}) \\ F_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\varepsilon_{ij}}, \quad \text{for } 1 \leq i \leq s \end{cases} \tag{4.3}$$

ここに，$x_j^{\varepsilon_{ij}}$は，$\varepsilon_{ij} \in \{0, 1, u\}$として，次式で与えられる．

$$x_j^{\varepsilon_{ij}} \triangleq \begin{cases} \overline{x_j} \ ; \ \text{if} \quad \varepsilon_{ij} = 0 \\ x_j \ ; \ \text{if} \quad \varepsilon_{ij} = 1 \\ 1 \ ; \ \text{if} \quad \varepsilon_{ij} = u \end{cases} \tag{4.4}$$

次に，$\mathbf{x}$を変数とする擬似ブール関数$y(\mathbf{x})$を次式のように表わす．

$$\begin{cases} y(\mathbf{x}) \triangleq a_0 + \sum_{i=1}^{m} a_i \, Y_i(\mathbf{x}) \\ Y_i(\mathbf{x}) \triangleq \prod_{j=1}^{n} x_j^{\delta_{ij}} \quad \text{for} \quad 1 \leq i \leq m \end{cases} \tag{4.5}$$

ここに，$x_j^{\delta_{ij}}$は，$\delta_{ij} \in \{0, 1\}$として，次式で与えられる．

$$x_j^{\delta_{ij}} \triangleq \begin{cases} 1 \ ; \quad \text{if} \quad \delta_{ij} = 0 \\ x_j \ ; \quad \text{if} \quad \delta_{ij} = 1 \end{cases} \tag{4.6}$$

〔定義 4.1〕

$F(\mathbf{x})$，$y(\mathbf{x})$を，それぞれ式（4.3）および式（4.5）で与えられるブール関数および擬似ブール関数とする．擬似ブール計画問題$P(y, F)$を次のような最小化問題とする．

制約条件，

$$F(\mathbf{x}) = 0 \tag{4.7}$$

のもとで，目的関数$y(\mathbf{x})$を最小にする$\mathbf{x}$を求めよ．

---

*）擬似ブール計画問題の定式化に関しては，すでに，第2章で詳細に述べた．よって，この節では，定式化に関しては簡単に述べる．

第2章で述べた擬似ブール計画法のアルゴリズムはBranch-and-Bound法にもとづいている．第2章でも述べたように，Branch-and-Bound法によって最小化問題を解く場合には，次の2つの評価関数の良し悪しがアルゴリズム全体の効率に大きな影響を及ぼす．

(1) 目的関数の下界を与える評価関数

(2) 問題の分割方法を決定する評価関数

4.3節および4.4節では，4.5節に示される擬似ブール計画法のアルゴリズムで用いられる評価関数について述べる．

## 4.3 含意の性質を用いた目的関数の下界の評価

この節では，含意（logical implication）の性質を用いた，目的関数の下界の評価方法について述べる．

また，この方法の計算時間の複雑さについても述べる．

以下では，式（4.5）で与えられる擬似ブール関数$y(\mathbf{x})$の係数は，次式を満たしていると仮定する．

$$\begin{cases} a_i > 0 & \text{for} \quad 1 \leq i \leq r \\ a_i < 0 & \text{for} \quad r+1 \leq i \leq m \end{cases} \tag{4.8}$$

第2章で示した$y(\mathbf{x})$の下界を与える関数$\ell b_0(y, \mathrm{F})$と等価な関数$\ell b(y)$を，$y(\mathbf{x})$の定数項および負の係数の総和として，次式で定義する．

$$\ell b(y) \triangleq a_0 + \sum_{i=r+1}^{m} a_i \tag{4.9}$$

関数$\ell b(y)$は，計算は容易であるが，$y(\mathbf{x})$のほとんど自明な下界しか与えない．$y(\mathbf{x})$のより良い下界を求める方法として，第2章では，制約条件式の性質を用いる関数$\ell b_k(y, \mathrm{F})$を提案した．第2章の定理2.2に示したように，$k$の値が変数の個数$n$に漸近するのに伴なって，$\ell b_k(y, \mathrm{F})$の値は，制約条件$\mathrm{F}(\mathbf{x}) = 0$のもとでの$y(\mathbf{x})$の最小値に漸近する．しかしながら，$\ell b_k(y, \mathrm{F})$の計算時間は，$k$の値の増加に伴なって指数関数的に増加する．そのため，第2章での実験の結果からは，$\ell b_1(y, \mathrm{F})$を用いた場合ですら，アルゴリズム全体の計算時間は$\ell b_0(y, \mathrm{F})$を用いたときよりも増大してしまった．

そこで，$\ell b_k(y, \mathrm{F})$にかわる方法として，以下ではブール関数の間の含意の関係を用いて$y(\mathbf{x})$の下界を評価する方法を提案する．この方法の基本的な考え方は，次のようである．

$f(\mathbf{x})$, $g(\mathbf{x})$をブール関数とし，$f(\mathbf{x})$, $g(\mathbf{x})$の含意の関係を$f(\mathbf{x}) \rightarrow g(\mathbf{x})$で表わす．この時，次の命題が成り立つことは，含意の定義から明らかである．

$$〔\ g(\mathbf{x}) \rightarrow f(\mathbf{x})\ 〕 \quad \text{iff}$$
$$〔\ f(\mathbf{x}) - g(\mathbf{x}) \geq 0 \quad \text{for} \quad \forall\ \mathbf{x} \in \{0, 1\}^n\ 〕 \tag{4.10}$$

次に，$a$，$b$を任意の正の数とする．このとき，$g(\mathbf{x}) \rightarrow f(\mathbf{x})$であるとすると，次式が成り立つ．

$$a \cdot f(\mathbf{x}) - b \cdot g(\mathbf{x})$$
$$= (a-b) \cdot f(\mathbf{x}) + b \cdot (f(\mathbf{x}) - g(\mathbf{x})) \geq (a-b) \cdot f(\mathbf{x}) \tag{4.11}$$

よって，次式を得る．

$$a \cdot f(\mathbf{x}) - b \cdot g(\mathbf{x}) \geq \min(a-b,\ 0) \tag{4.12}$$

式（4.12）の右辺の値は，式（4.9）で与えられる$\ell b$を用いて得られる$a \cdot f(\mathbf{x}) - b \cdot g(\mathbf{x})$の下界$-b$よりも大きい．すなわち，上記の方法で得られる下界は，$\ell b$を用いて得られる下界よりも良い評価を与える．

上で述べた考え方を一般化すると，次のようになる．

まず，

$$a > 0,\quad b_i > 0,\quad \text{for} \quad 1 \leq i \leq q \tag{4.13}$$

かつ，

$$g_i(\mathbf{x}) \rightarrow f(\mathbf{x}),\quad \text{for} \quad 1 \leq i \leq q \tag{4.14}$$

とする．このとき，次式を得る．

$$a \cdot f(\mathbf{x}) - \sum_{i=1}^{q} b_i \cdot g_i(\mathbf{x})$$
$$= \left(a - \sum_{i=1}^{q} b_i\right) \cdot f(\mathbf{x}) + \sum_{i=1}^{q} b_i \cdot (f(\mathbf{x}) - g_i(\mathbf{x}))$$
$$\geq \left(a - \sum_{i=1}^{q} b_i\right) \cdot f(\mathbf{x})$$
$$\geq \min\left(a - \sum_{i=1}^{q} b_i,\ 0\right) \tag{4.15}$$

また，逆に，

$$a_i > 0,\quad \text{for} \quad 1 \leq i \leq q,\quad b > 0 \tag{4.16}$$

かつ，

$$g(\mathbf{x}) \rightarrow f_i(\mathbf{x}),\quad \text{for} \quad 1 \leq i \leq q \tag{4.17}$$

とすると，次式を得る．

$$\sum_{i=1}^{q} a_i \cdot f_i(\mathbf{x}) - b \cdot g(\mathbf{x})$$
$$= \sum_{i=1}^{q} a_i \cdot (f_i(\mathbf{x}) - g(\mathbf{x})) + \left(\sum_{i=1}^{q} a_i - b\right) \cdot g(\mathbf{x})$$
$$\geq \left(\sum_{i=1}^{q} a_i - b\right) \cdot g(\mathbf{x})$$
$$\geq \min\left(\sum_{i=1}^{q} a_i - b,\ 0\right) \tag{4.18}$$

以上の考察より，含意の関係を用いた$y(\mathbf{x})$の下界の評価手続が以下のように得られる．

〔手続き 4.1〕

**入力**：$y(\mathbf{x})$

**出力**：$y(\mathbf{x})$ の下界 $\ell b$

**記法**：$y(\mathbf{x})$ は，次式の形をしているとする．

$$y(\mathbf{x}) = a_0 + \sum_{i=1}^{r} a_i \cdot f_i(\mathbf{x}) - \sum_{j=1}^{q} b_i \cdot g_j(\mathbf{x}) \tag{4.19}$$

ここに，

$$\begin{cases} a_i > 0 \text{ for } 1 \leq i \leq r \\ b_i > 0 \text{ for } 1 \leq i \leq q \end{cases} \tag{4.20}$$

であるとする．

**方法**：

Step 1：(i) $r = 0$ または $q = 0$ のとき，Step 5へ；

(ii) そうでないとき，Step 2へ；

Step 2：$i = 1, 2, \cdots, r$ について，Step 3，Step 4を行なった後，Step 5へ；

Step 3：$j = 1, 2, \cdots, q$ について，Step 4を行なう；

Step 4：(i) $g_j(\mathbf{x}) \to f_i(\mathbf{x})$ かつ，$a_i \geq b_j$

ならば，

$a_i \leftarrow a_i - b_j$,

$b_j \leftarrow 0$,

とする；

(ii) $g_j(\mathbf{x}) \to f_i(\mathbf{x})$ かつ，$a_i < b_j$

ならば，

$b_j \leftarrow b_j - a_i$,

$a_i \leftarrow 0$,

とする；

Step 5：$\ell b \leftarrow a_0 + \sum_{j=1}^{q} b_j$ とする；

手続き終了．

次に例題を示し，この例題に対して手続き 4.1 を適用して $y(\mathbf{x})$ の下界を求めてみる．

〔例題 4.1〕

$$\begin{aligned} y(\mathbf{x}) = {} & 8x_1x_4 + 7x_2x_4x_6 + 5x_3x_6 + 3x_1x_2x_5 + 2x_5x_6 - 2x_4x_5 - 3x_2x_5x_6 \\ & -4x_1x_5 - 4x_1x_5x_7x_8 - 6x_3x_4x_6 - 6x_4x_7x_8x_9 \end{aligned} \tag{4.21}$$

を最小にする $\mathbf{x}$ を求めよ．

式（4.21）を構成する各項のうち，手続き4.1のStep 4の(i)の条件を満たす項の対は存在しない．しかし，Step 4の(ii)の条件を満たす項の対は，次の2対が存在する．すなわち，

第10項と第3項：$x_3 x_4 x_6 \rightarrow x_3 x_6$ （4.22）

および，

第7項と第5項：$x_2 x_5 x_6 \rightarrow x_5 x_6$ （4.23）

である．これらの2対の項に対して，Step 4の(ii)の処理をほどこし，係数が0となる項を取り除くと，式（4.21）の$y(\mathbf{x})$は，次式のように変換される．

$$y(\mathbf{x}) = 8x_1 x_4 + 7x_2 x_4 x_6 + 3x_1 x_2 x_5 - 2x_4 x_5 - x_2 x_5 x_6 - 4x_1 x_5 - 4x_1 x_5 x_7 x_8 - x_3 x_4 x_6 - 6x_4 x_7 x_8 x_9 \quad (4.24)$$

式（4.24）の$y(\mathbf{x})$の負の係数の総和を求めると，−18となる．一方，式（4.21）の負の係数の総和は−25である．

また，例題4.1の最適解の1つは，$\mathbf{x}=(0, 0, 0, 1, 1, 0, 1, 1, 1)$であり，この$\mathbf{x}$に対して$y(\mathbf{x}) = -8$であることが知られる．

よって，例題4.1に対して，手続き4.1で求めた$y(\mathbf{x})$の下界は，$y(\mathbf{x})$に直接$\ell b_0$を用いて求めた下界よりもかなり改善されることが知られる．

### 4.4 問題の分割方法

この節では，次節で示す擬似ブール計画法のアルゴリズムで用いられる，問題の分割方法について述べる．

ブール関数$\mathrm{F}(\mathbf{x})$および擬似ブール関数$y(\mathbf{x})$に含まれる変数$x_j$に値$v \in \{0, 1\}$を与えて得られる式を，それぞれ，$\mathrm{F}(x_j=v)$および$y(x_j=v)$で表わすことにする．すなわち，

$$\mathrm{F}(x_j=v) \triangleq \mathrm{F}(x_1, \cdots, x_{j-1}, v, x_{j+1}, \cdots, x_n) \quad (4.25)$$

$$y(x_j=v) \triangleq y(x_1, \cdots, x_{j-1}, v, x_{j+1}, \cdots, x_n) \quad (4.26)$$

問題$P(y, \mathrm{F})$に含まれる変数$x_j$に値$v \in \{0, 1\}$を割り当てることによって得られる，$P(y, \mathrm{F})$の2つの部分問題は，

$$P(y(x_j=0), \mathrm{F}(x_j=0)) \quad (4.27)$$

および，

$$P(y(x_j=1), \mathrm{F}(x_j=1)) \quad (4.28)$$

で表わされる．

以下では，問題$P(y, \mathrm{F})$から，式（4.27），式（4.28）の2つの部分問題を得る過程を，問題$P(y, \mathrm{F})$の分割と呼ぶ．また，分割の過程で値が割り当てられた変数$x_j$を，

問題 $P(y, F)$ に含まれる $n$ 個の変数のうち，どの変数を分割変数としてえらぶかによって，$P(y, F)$ の最適解を求めるアルゴリズム中で処理される部分問題の個数が大きな影響を受けると考えられる．よって，分割変数の選択方法は，部分問題の目的関数の下界の評価方法とともに，アルゴリズムの効率を高める上で重要な問題である．

まず，変数 $x_j$ の分割変数としての良さを評価する関数として，2つの関数merit および demerit を，次式で定義する．

$$\mathrm{merit}(y, j) \triangleq \max\{ \ell b(y(x_j = v)) \mid v \in \{0, 1\} \} \tag{4.29}$$

$$\mathrm{demerit}(y, j) \triangleq \min\{ ub(y(x_j = v)) \mid v \in \{0, 1\} \} \tag{4.30}$$

ここに，関数 $\ell b$ は式（4.9）で与えられる．また，関数 $ub$ は，$y(\mathbf{x})$ の定数項および正の係数の総和として，次式で与えられる．

$$ub(y) \triangleq a_0 + \sum_{i=1}^{r} a_i \tag{4.31}$$

次に，関数merit および demerit を用いて，変数の間の順序関係 $\prec$ を，次のように定義する．

〔定義 4.2〕（関係 $\prec$）

変数 $x_i$，$x_j$ に関して，関係

$$x_i \prec x_j \tag{4.32}$$

が成り立つのは，次の(i)，(ii)，(iii)の条件のうちのいずれかが成り立つとき，かつ，そのときに限る．

(i) $\mathrm{merit}(y, i) > \mathrm{merit}(y, j)$

(ii) $\mathrm{merit}(y, i) = \mathrm{merit}(y, j)$，かつ，
$\mathrm{demerit}(y, i) < \mathrm{demerit}(y, j)$

(iii) $\mathrm{merit}(y, i) = \mathrm{merit}(y, j)$，かつ，
$\mathrm{demerit}(y, i) = \mathrm{demerit}(y, j)$，かつ，
$i \leq j$

上記のように定義される関係 $\prec$ は，変数の集合上の全順序関係であることは明らかである．

第2章で述べたアルゴリズムでは，分割変数として，関係 $\prec$ に関する最小元を選択している．しかしながら，問題 $P$ に含まれる変数の個数を $n$ とすると，問題 $P$ に含まれる変数の中から，

関係$\prec$に関する最小元を求めるためには，少くとも，$n$回のmeritの値の計算を必要とする．そのため，第2章に示したアルゴリズムでは，分割変数の選択に多くの計算を必要とした．

以下では，上記の選択方法よりも計算量が少なくてすむ分割変数の選択手続きを与える．

Branch-and-Bound法で問題$P_0$が解かれる過程は，木$B=(V, E)$で表わされる．ここに，$V$は節点の集合，$E$は枝の集合である．$V$中の各節点には，問題$P_0$の部分問題が対応する．また，$B$の根には，問題$P_0$が対応する．$E$中の枝$(P_i, P_j)$は，部分問題$P_j$が，$P_i$を分割して得られた部分問題の一つであることを表わしている．

部分問題$P_i$のレベルを，木$B$の根$P_0$から節点$P_i$に至るパスの長さで定義し，$level(P_i)$で表わす．

すなわち，$(P_j, P_i) \in E$とするとき，

$$level(P_i) \triangleq \begin{cases} 0 \ ; \ \text{if} \ P_i = P_0 \\ level(P_i) + 1 \ ; \ \text{if} \ P_i \neq P_0 \end{cases} \qquad (4.33)$$

以下に示す，分割変数を選択する手続きでは，$d$ $(1 \leq d \leq n)$を与えられたとき，変数の間の順序関係をレベル$d$ごとに計算しなおすことにする．

〔手続き4.2〕(分割変数の選択)

**入力**：部分問題$P(y, \mathrm{F})$

変数の間の順序関係を計算しなおすレベルの間隔$d$；

**出力**：分割変数　$x_j*$；

**記法**：以下では，変数の順序集合を，

$$\Gamma \triangleq (x_{(1)}, x_{(2)}, \cdots, x_{(n)}) \qquad (4.34)$$

で表わす．

**方法**：

Step 1：〔$\Gamma$の再構成〕

(i)　$level(P) = k \cdot d$　$(k = 0, 1, \cdots, \lfloor n/d \rfloor)$

ならば，変数間の順序関係が

$$x_{(i)} \prec x_{(i+1)} \quad \text{for} \quad 1 \leq i \leq n-1, \qquad (4.35)$$

となるように，$\Gamma$を構成する．

(ii)　$level(P) \neq k \cdot d$，for $1 \leq \forall k \leq \lfloor n/d \rfloor$

木$B$の根$P_0$から$P$に至るパス上で構成された$\Gamma$のうち，最も深いレベルに属する部分問題に対応するものを，部分問題$P$に対する$\Gamma$の値とする

Step 2：〔分割変数の選択〕

部分問題$P$に含まれる変数の中で，$\Gamma$の最小の要素を，分割変数$x_j^*$とする．

この手続きで，$d$の値を1とすると，問題の分割を行なうたびに必ず$\Gamma$を構成することになる．また逆に$d$の値が$n$の場合には，原問題$P_0$に対して一回だけ$\Gamma$を構成し，以後は，つねにこの$\Gamma$を用いて分割変数の決定を行なうことになる．

$d$の値を増すと，分割変数の決定に要する平均計算時間は減少するが，$d$の値を増すと，一般には，アルゴリズム中で処理される部分問題の個数は増大すると考えられる．$d$の値と，擬似ブール計画法のアルゴリズムの計算時間との関係については，4.6節で考察する．

## 4.5 アルゴリズム

この節では，前節までで述べた評価関数を用いた，擬似ブール計画法のアルゴリズムを示す．

$I$を，未解決な部分問題に対応する，変数のベクトル$\mathbf{x}$を要素とする集合とする．$I$の中から，次に処理すべき部分問題に対応するベクトル$\mathbf{x}$を選択する手続きを，次のように与える．

〔手続き4.3〕（部分問題の選択）

**入力**：ベクトル$\mathbf{x}$の集合$I$；

**出力**：$I$の要素$\mathbf{x}_0$

**方法**：

$I$の要素中で，最も深いレベルにある部分問題に対応する$\mathbf{x}$を要素とする集合を$I'$とする．すなわち，$I'$は，$I$中で，含まれる$u$の個数が最小であるような$\mathbf{x}$を要素とする集合である．次に，$I'$の中で，$\ell b(y)$を最小にする要素を1つえらび，これを，$\mathbf{x}_0$とする；手続き終了．

擬似ブール計画法のアルゴリズムは，次のように与えられる．

〔アルゴリズム4.1〕

**入力**：擬似ブール計画問題$P(y, \mathrm{F})$；

**出力**：$P(y, \mathrm{F})$の全ての最適解の集合Xopt，

Xoptに対する目的関数の値Copt；

ただし，問題$P(y, \mathrm{F})$が，制約条件$\mathrm{F}(\mathbf{x})=0$を満足する可能解を持たない時には，

Xopt $=\emptyset$，Copt $=\infty$が出力される．

記法：以下では，$I$ を，未解決な部分問題に対応する変数のベクトル $\mathbf{x}$ を要素とする集合とする．また，変数 $x_j$ にまだ値が割り当てられていないことを，$x_j = u$ で表わす．

方法：

Step 1：〔初期設定〕

$I \leftarrow \{ (u, u, \cdots, u) \}$, Xopt $\leftarrow \emptyset$, Copt $\leftarrow \infty$ とする；Step 2へ；

Step 2：〔部分問題の選択〕

(i) $I = \emptyset$ ならば，Step 8へ；

(ii) $I \neq \emptyset$ ならば，手続き4.3を用いて，$I$ 中から，変数のベクトルを1つ選び，これを $\mathbf{x}_0$ とする；$\mathbf{x}_0$ を $I$ から取り除いてStep 3へ；

Step 3：〔制約条件の処理〕

$F(\mathbf{x}_0)$ を既約化して得られる式を $F'(\mathbf{x})$ とする*;

(i) $F'(\mathbf{x}) \equiv 1$ ならば，Step 2へ；

(ii) $F'(\mathbf{x}) \not\equiv 1$ ならば，Step 4へ；

Step 4：〔目的関数の処理〕

$y(\mathbf{x}_0)$ を簡単化して得られる式を $y'(\mathbf{x})$ とする；*

(i) $\ell b(y') >$ Copt ならば，Step 2へ；

(ii) $y'(\mathbf{x})$ が定数関数ならば，Step 7へ；

(iii) それ以外のとき，Step 5へ；

Step 5：〔部分問題の縮小〕

$$y'(\mathbf{x}) = a'_0 + \sum_{i=1}^{m'} a'_i \; Y'_i(\mathbf{x}) \tag{4.36}$$

$$\begin{aligned} a'_i &> 0 \quad \text{for} \quad 1 \leq i \leq r' \\ a'_i &< 0 \quad \text{for} \quad r'+1 \leq i \leq m' \end{aligned} \tag{4.37}$$

とする．

(i) $a'_i > \text{Copt} - \sum_{i=r'+1}^{m'} a'_i$ (4.38)

かつ，$Y'(\mathbf{x})$ が単一の変数のみからなるような $i$ が存在すれば，これらの条件をみたす全ての $i$ について，

$$Y'_i(\mathbf{x}) \equiv x_j \leftarrow 0$$

として，Step 3へ；*

(ii) 上記の条件をみたす $i$ が存在しなければStep 6へ；

---

*) $F(\mathbf{x}_0)$ の既約化および $y(\mathbf{x}_0)$ の簡単化に関しては，第2章2.5節および第3章参照．

*) 目的関数の下界を用いた部分問題の縮小に関しては，第2章2.5節参照．

Step 6：〔部分問題の分割〕

手続き4.2を用いて得られた分割変数を$x_j^*$とする．$\mathbf{x}_0=(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n)$とするとき，

$$\ell b(y(x_j=v)) \leq \mathrm{Copt}$$

を満足する全ての$v \in \{0, 1\}$について，

$$I \leftarrow I \cup (\tilde{x}_1 \cdots, \tilde{x}_{j-1}, v, \tilde{x}_{j+1}, \cdots, \tilde{x}_n)$$

として，Step 2へ；

Step 7：〔最適解の候補の更新〕

$y'(\mathbf{x})$は定数関数である．$y'(\mathbf{x})$の値を$a'_0$とする．

(i) $a'_0 < \mathrm{Copt}$ならば，

$\mathrm{Xopt} \leftarrow \{\mathbf{x}_0\}$，$\mathrm{Copt} \leftarrow a'_0$

とする．$I$中から，$\ell b(y(\mathbf{x}_0)) > \mathrm{Copt}$である要素を全て取り除く；Step 2へ；

(ii) $a'_0 = \mathrm{Copt}$ならば，

$\mathrm{Xopt} \leftarrow \mathrm{Xopt} \cup \{\mathbf{x}_0\}$

としてStep 2へ；

Step 8：〔停　止〕

アルゴリズム終了．

## 4.6　評価関数の強力さと計算時間

この節では，4.3節および4.4節で与えた評価関数の強力さと，それらを用いた擬似ブール計画法のアルゴリズムの計算時間について考察する．

### 4.6.1　目的関数の下界の評価方法とアルゴリズムの計算時間

4.3節では，目的関数の下界を与える関数を2種類示した．一つは，式（4.9）で与えられる評価関数であり，他の一つは，手続き4.1で与えられる，含意の関係を用いた評価関数である．

目的関数$y(\mathbf{x})$の項数を$m$とする．この時，式（4.9）の関数の計算に必要な時間は$O(m)$である．一方，1組の積項$Y_i(\mathbf{x})$と$Y_j(\mathbf{x})$の間の含意の関係を調べるのに要する時間を1単位時間として，手続き4.1の計算に必要な時間は$O(m^2)$となる．* よって，評価関数自体の計

*）手続き4.1の計算時間が最大となるのは，正の係数を持つ項と負の係数を持つ項が同数ずつ存在する場合である．この時，手続き4.1では，含意の関係が$m^2/4$回調べられる．

算時間の複雑さに関しては，式（ 4.9 ）の方が手続き 4.1 よりも小さい．

しかしながら，4.3 節でも述べたように，$y(\mathbf{x})$ を構成する項の中に，

$$a_i > 0 \ , \ a_j < 0 \text{ かつ，} Y_j(\mathbf{x}) \rightarrow Y_i(\mathbf{x}) \qquad (4.40)$$

を満たすような項の組が存在する場合には，手続き 4.1 によって与えられる目的関数の下界の値は，式（ 4.9 ）によって与えられる目的関数の下界の値よりも大きくなる．すなわち，手続き 4.1 の方が，式（ 4.9 ）よりも，目的関数のより良い下界を与える．

Branch-and-Bound 法では，目的関数の下界がより厳密に評価されればされるほど，アルゴリズム中で処理される部分問題の個数は減少すると考えられる．これは，たとえば，アルゴリズム 4.1 の Step 4 での $\ell b(y') > \mathrm{Copt}$ の条件，または，Step 6 での式（ 4.39 ）の条件または，Step 7 (i)での $\ell b(y(\mathbf{x}_0)) > \mathrm{Copt}$ の条件によって処理を中止される部分問題の個数が増加するからである．（以下では，これらの条件によって部分問題の処理を中止することを，枝刈りを行なうと言う．）

よって，評価関数の厳密さと，アルゴリズム全体の計算時間に関して，次の考察が得られる．評価関数 1 と評価関数 2 を考える．$K_1$，$K_2$ を，それぞれ評価関数 1，評価関数 2 を用いた時にアルゴリズム中で処理される部分問題の個数とする．また，$t_0$ を，評価関数の計算時間を除いた，部分問題 1 個当りの計算時間とする．さらに，$t_1$，$t_2$ を，それぞれ評価関数 1，評価関数 2 の 1 回当りの計算時間とする．このとき，$T_1$，$T_2$ を，それぞれ評価関数 1，評価関数 2 を用いた場合のアルゴリズム全体の計算時間とすると，$T_1$，$T_2$ は，それぞれ，$K_1$，$K_2$，$t_0$，$t_1$，$t_2$ を用いて，次式で表わされる．

$$T_1 = (t_0 + t_1) \cdot K_1 \qquad (4.41)$$

$$T_2 = (t_0 + t_2) \cdot K_2 \qquad (4.42)$$

ここで，

$$K_2 = K_1 - \triangle K \qquad (4.43)$$

$$t_2 = t_1 + \triangle t \qquad (4.44)$$

とおくと，次式が得られる．

$$\begin{aligned} & T_2 / T_1 \\ &= \{(t_0 + t_1 + \triangle t) \cdot (K_1 - \triangle K)\} / \{(t_0 + t_1) \cdot K_1\} \\ &= \{(t_0 + t_1 + \triangle t) / (t_0 + t_1)\} / \{(K_1 - \triangle K) / K_1\} \qquad (4.45) \end{aligned}$$

よって，$T_2 / T_1 < 1$ となる条件は，

$$(t_0 + t_1 + \triangle t) / (t_0 + t_1) < (K_1 - \triangle K) / K_1 \qquad (4.46)$$

で与えられる．すなわち，評価関数の厳密さを増すことによって枝刈りされる部分問題の，全体の部分問題の個数に対する割り合いが，評価個数の計算時間の増分の，部分問題1個当りの計算時間に占める割り合いよりも大きければ，アルゴリズム全体の計算時間は減少することになる．

### 4.6.2 分割変数の選択方法と計算時間

4.4節で示した，分割変数を選択する手続きは，変数の間の順序関係を構成しなおす間隔 $d$ をパラメータとして含む．さて，部分問題 $P$ のレベルを $\ell$ とし，木 $B$ の根 $P_0$ から $P$ に至るパス上で，最も深いレベルで $\Gamma$ を再構成した部分問題を $P_i$ とする．また，部分問題 $P_i$ のレベルを $\ell_i$ とする．このとき，一般に $\ell$ と $\ell_i$ の差が大きくなればなるほど，部分問題 $P$ における変数の実際の順序関係と，$\Gamma$ の中でのそれらの変数の順序関係とは一致しなくなる傾向にあると考えられる．よって，一般に，$d$ の値が大きくとられればとられるほど，手続き4.2で選ばれる変数 $x_j$ は分割変数として不適当になりやすい．その結果，$d$ の値を増せば増すほど，アルゴリズム中で処理される部分問題の個数は増すことになる．

一方，手続き4.3の計算時間は，$d$ の値によって次のように変化する．まず，部分問題 $P$ で，手続き4.3によって $\Gamma$ が再構成されたとする．ただし，$P$ のレベルを $\ell$ とするとき，

$$0 \leq \ell \leq n - d \tag{4.47}$$

であるとする．また，$d \geq 2$ と仮定する．

$P$ に，$d-1$ 回以下の分割をほどこして得られる部分問題の集合を $Id$ とする．すなわち，$P \overset{r}{\Rightarrow} P_j$ によって，$P_j$ が $P$ を $r$ 回分割して得られる部分問題の一つであることを表わすと，$Id$ は次式で定義される．

$$Id \triangleq \{ P_j \mid P \overset{r}{\Rightarrow} P_j \ , \ 0 \leq r \leq d-1 \} \tag{4.48}$$

$Id$ 中のレベル（$\ell + r$）にある部分問題の個数は，明らかに，$2^r$ 個である．

レベル $\ell$ にある部分問題が，アルゴリズム4.1中で枝刈りされる確率を $\alpha(\ell)$ とする．また，$Id$ の要素の中で，手続き4.3によって分割変数の決定を行なわれる部分問題の集合を $\mathrm{R}d$ とする．このとき，$\mathrm{R}d$ の要素の個数は，以下のようにして求められる．

$\mathrm{R}d$ 中のレベル（$\ell + r$）にある部分問題の個数を $\mathrm{N}_r$ で表わすと，$\mathrm{N}_r$ は次式で与えられる．

$$N_0 = 1 \tag{4.49}$$

$$\begin{aligned} N_r &= N_{r-1} \cdot (1 - \alpha(\ell + r)) \cdot 2 \\ &= 2^r \cdot \prod_{i=1}^{r} (1 - \alpha(\ell + i)) \ , \quad \text{for} \quad r \geq 1 \end{aligned} \tag{4.50}$$

よって，$\mathrm{R}d$ の要素の個数は，次式で与えられる．

$$|Rd| = \sum_{r=0}^{d-1} N_r$$
$$= \sum_{r=0}^{d-1} 2^r \cdot \prod_{i=1}^{r} (1-\alpha(\ell+i)) \qquad (4.51)$$

部分問題$P$で再構成された$\Gamma$は，$Rd$中の全ての要素で用いられることになる．手続き4.2のStep 2の計算時間は，Step 1の計算時間と比較して無視できるほど小さいと仮定して，部分問題$P$で$\Gamma$を構成するのに要する時間を$t(\ell)$とすると，手続き4.2の平均計算時間$T(\ell)$は，次式で与えられる．

$$T(\ell) = t(\ell) / |Rd|$$
$$= t(\ell) / [\sum_{r=0}^{d-1} 2^r \cdot \prod_{i=1}^{r} (1-\alpha(\ell+i))] \qquad (4.52)$$

式（4.52）で，$\alpha(\ell)$は，$\ell$に依存しない定数であり，かつ$\alpha < 1/2$と仮定すると，式（4.52）より，次式を得る．

$$T(\ell) = t(\ell) / \sum_{r=0}^{d-1} 2^r \cdot (1-\alpha)^r$$
$$= t(\ell)\cdot(2\cdot(1-\alpha)-1)/\{(2\cdot(1-\alpha))^d - 1\} \qquad (4.53)$$

よって，$d$の値が増加するのに伴なって，手続き4.2の平均時間は指数関数的に減少する．

第2章で述べたアルゴリズム2.1の各ステップの計算時間を測定した結果から，分割変数の選択に要する時間がアルゴリズム全体の計算時間に占める割り合いはかなり大きいことが知られている．よって，4.6.1での考察によれば，$d$の値が適切にえらばれれば，アルゴリズム全体の計算時間を最小にすることができると期待される．

## 4 7 実験およびその結果

この節では，4.3節および4.4節で述べた評価関数の有効さを調べるための実験とその結果について述べる．また，これらの実験結果に対して4.6節で述べた，評価関数の強力さとアルゴリズムの計算時間との関係に関する結果と比較考察を行う．

実験は次の2つからなる．まず，実験1では，分割変数の選択を行なう手続4.2で，パラメータ$d$の値を変化させたときの，問題の分割回数およびアルゴリズム全体の計算時間の変化を調べる．次に，実験2では，部分問題の目的関数の各項間の含意関係を用いて目的関数の下界をより厳密に評価する，手続き4.1の有効さを調べる．

### 4.7.1 アルゴリズムの実現と問題の作製方法

4.5節で述べた擬似ブール計画法のアルゴリズムを，FORTRAN Ⅳ で記述し，名古屋大学工学部情報工学科の計算機FACOM 230/38 上で実現した．実現されたシステムでは，第3

章で述べたハッシングの技法を用いて，ブール関数および擬似ブール関数の数式処理を行なっている.*

実験に用いられた問題は，第2章で述べた実験と同様の方法で，計算機の一様乱数を用いて発生された．発生される擬似ブール計画問題の規模を定めるパラメータは，次の3つである．

$n$ ：問題に含まれる変数の個数

$s$ ：制約条件式 $\mathbf{F}(\mathbf{x})$ の項数

$m$ ：目的関数 $y(\mathbf{x})$ の項数

また，以下の実験で，TおよびBはそれぞれ，次の値に関する10問題についての平均値を表わす．

T ：アルゴリズムが終了するまでの計算時間

B ：アルゴリズムが終了するまでに分割された部分問題の個数

### 4.7.2 実 験 1

手続き4.2の有効さを調べるため，手続き4.2を用いて分割変数を選択した場合と，部分問題に含まれる変数の中から，計算機の一様乱数を用いてランダムに分割変数を選択した場合とについて，アルゴリズム4.1の計算時間Tおよび分割の回数Bを測定した．ただし，解かれた問題は，$n=30$；$s=0$；$m=20, 40, 60, 80, 100$ の5種類各10問題である．また，手続き4.2のパラメータ $d$ の値は $30(=n)$ とした．また，実験1では，目的関数の下界は，式（4.9）を用いて評価し，手続4.1は用いない．

図4.1に，分割変数の2通りの選択方法を用いた場合のアルゴリズムの計算時間Tと分割の回数Bを示す．同図より，分割変数をランダムにえらんだ場合に比べ，手続き4.2を用いた場合には，計算時間，分割回数ともに大幅に削減されることが知られる．よって，目的関数の下界および上界を用いて分割変数を選択する手続き4.2は，極めて有効であると結論される．

---

*）第3章で述べた，数式の内部表現の方法を用いることにより，手続き4.1で用いられる，積項間の含意の関係の検査は，計算機のビット処理機能を用いて容易に実現される．具体的な方法に関しては，付録3で述べる．

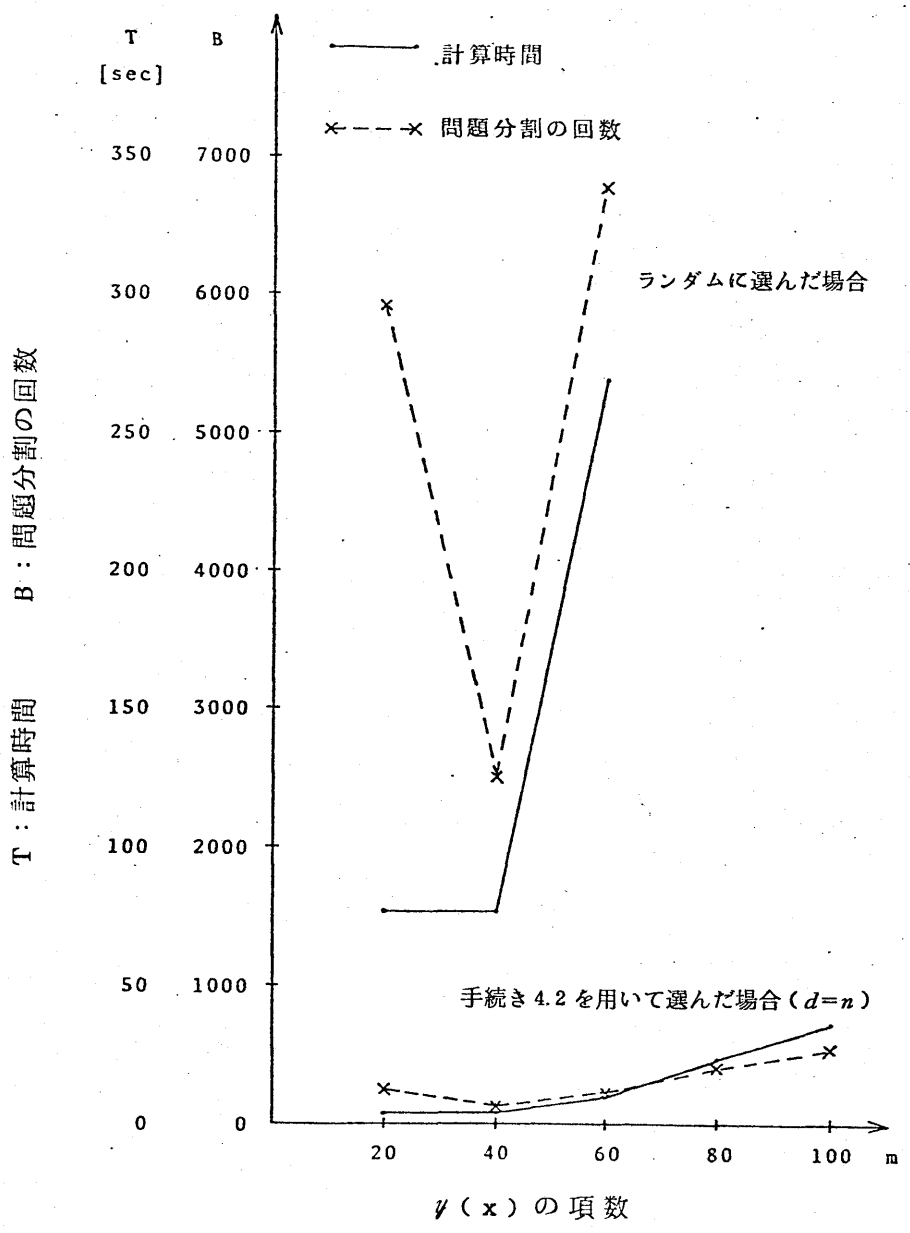



**図 4. 1　分割変数の選択方法とアルゴリズムの計算時間および分割回数の関係**

次に，手続き 4. 2 でパラメータ $d$ の値を変化させて，アルゴリズム 4. 1 の計算時間 T および分割の回数 B を測定した．T／B の値は，ほぼ，部分問題 1 個あたりの平均計算時間を表わしていると考えられる．この値を，以下では T／B 比と呼ぶことにする．

パラメータ $d$ の値を変化させたときの，T，B および T／B の変化を図 4. 2 および図 4. 3 に示す．ただし，図 4. 2 は，$n=30$，$s=0$，$m=20$ の場合を，また図 4. 3 は，$n=30$，$s=0$，$m=100$ の場合をそれぞれ表わしている．

両図より，次のことが知られる．まず，$d$の値が増すと，分割の回数はほぼ単調に増加するが，T／B比は逆にほぼ指数関数的に減少する．その結果，図4.2では，$d=4$および5で計算時間Tが最小値を持つ．また，図4.3では，$d$の値が増すと，Tの値はほぼ単調に減少して，$d=30$で最小となる．これは，$d$の値が10以上では，分割の回数がほとんど一定になるためである．

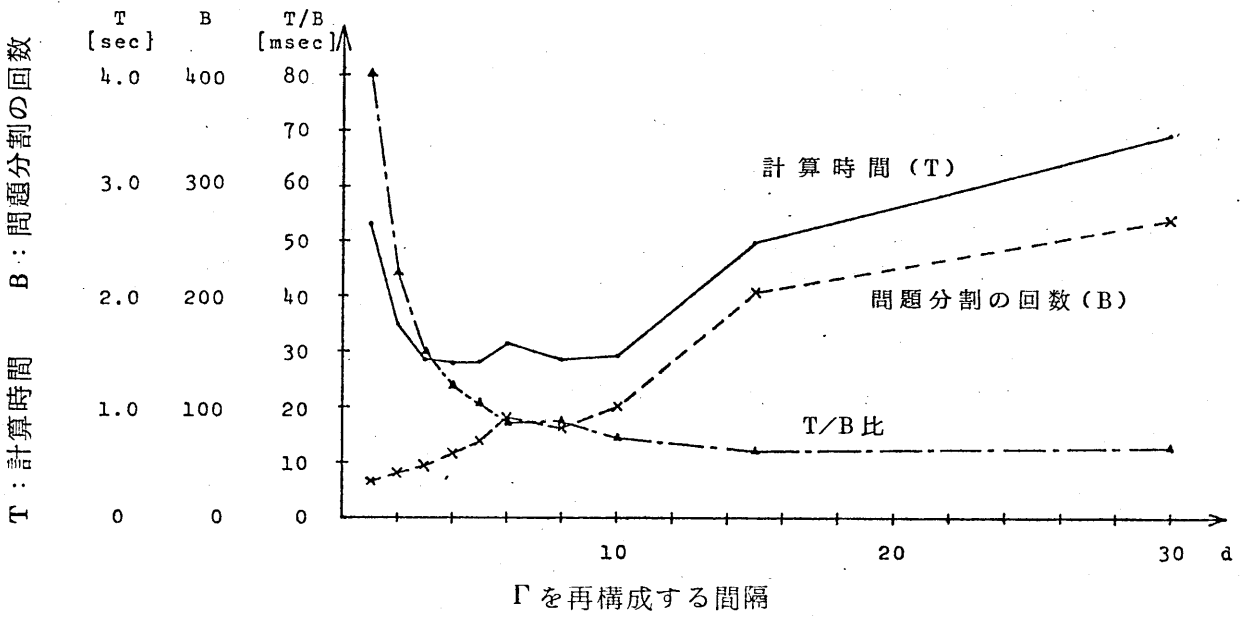



図4.2　アルゴリズムの計算時間（T），分割の回数（B）およびT/B比とパラメータdの関係（n=30, s=0, m=20）

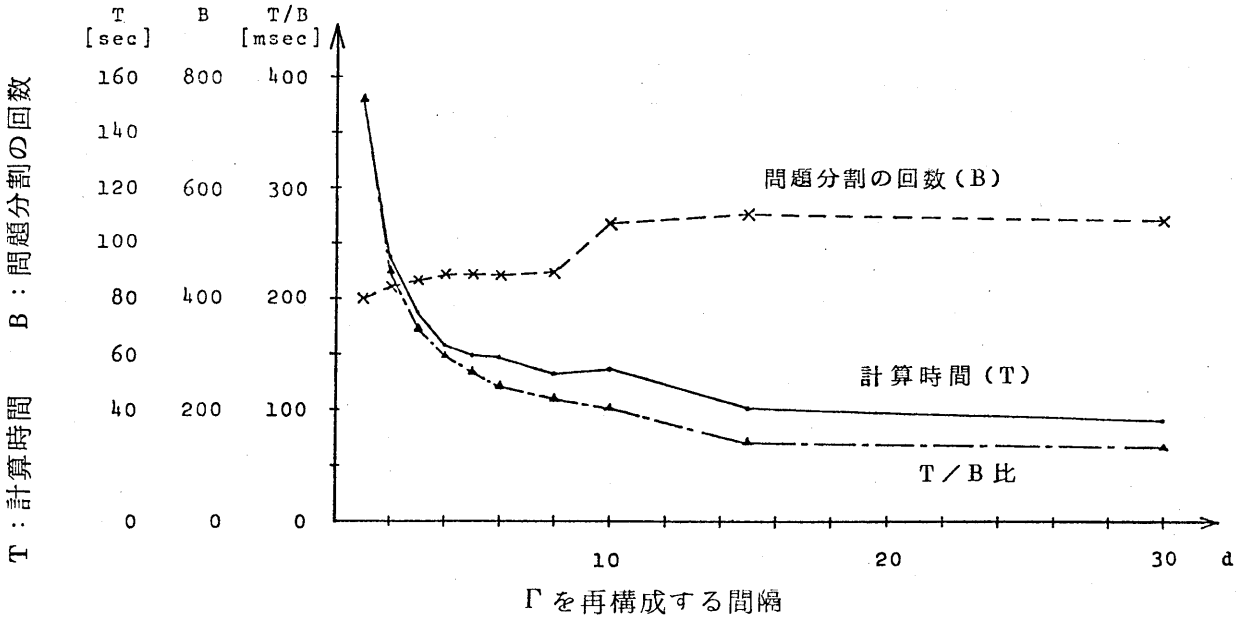



図4.3　アルゴリズムの計算時間（T），分割の回数（B）およびT/B比とパラメータdの関係（n=30, s=0, m=100）

両者の問題での分割回数の増え方の違いは，両者の問題での変数の順序関係の変動によるものと考えられる．すなわち，前者の問題では，原問題 $P$ 中の変数の順序関係 $\prec$ と，各部分問題中の変数の順序関係が大きくくい違うのに対して，後者の問題では，原問題 $P$ 中の変数の順序関係が，各部分問題中の変数の間でも比較的よく保たれていると考えられる．

次に，変数の個数 $n$，制約条件式の項数 $s$ および目的関数の項数 $m$ のいくつかの値についてパラメータ $d$ の値を変化させた時のアルゴリズム 4.1 の計算時間を表 4.1 に示す．同表より，次のことが知られる．

(1) まず，$n=20$, $s=40$, $m=20$；$n=30$, $s=0$, $m=20$；$n=30$, $s=60$, $m=20$ の 3 種類の問題では，それぞれ $d=10$, $d=5$, $d=10$ で計算時間が最小になる．

(2) 次に，$d$ の値を変化させた場合の計算時間の最小値を Tmin で，また，$d=1$ の場合の計

**表 4.1 アルゴリズム 4.1 の計算時間とパラメータ d の関係**

(a) n = 20

| s | m d | 1 | 2 | 3 | 4 | 5 | 10 | 20 | Tmin / T(d=1) |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 20 | 1.22 | 0.80 | 0.64 | 0.55 | 0.49 | 0.42 | 0.41 | 0.366 |
| | 40 | 4.61 | 3.21 | 2.54 | 2.21 | 2.20 | 1.66 | 1.58 | 0.343 |
| | 60 | 16.2 | 10.7 | 8.89 | 8.07 | 7.53 | 5.84 | 5.13 | 0.316 |
| | 80 | 26.4 | 17.2 | 13.6 | 12.2 | 12.5 | 9.01 | 8.08 | 0.306 |
| | 100 | 37.5 | 23.8 | 19.5 | 17.4 | 16.5 | 13.5 | 12.2 | 0.327 |
| 40 | 20 | 1.62 | 1.32 | 1.18 | 1.15 | 1.13 | 1.05 | 1.06 | 0.648 |
| | 40 | 4.67 | 3.65 | 3.20 | 3.08 | 2.76 | 2.55 | 2.38 | 0.510 |
| | 60 | 7.84 | 5.95 | 5.35 | 4.59 | 4.30 | 3.86 | 3.60 | 0.459 |
| | 80 | 10.2 | 7.22 | 6.51 | 5.87 | 5.70 | 3.92 | 3.73 | 0.367 |
| | 100 | 13.8 | 9.77 | 8.33 | 7.51 | 7.07 | 6.12 | 5.47 | 0.395 |

(b) n=30

| s | m d | 1 | 2 | 3 | 5 | 10 | 15 | 30 | Tmin / T(d=1) |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 20 | 2.65 | 1.74 | 1.42 | 1.40 | 1.64 | 2.48 | 3.46 | 0.528 |
| | 40 | 13.5 | 8.59 | 7.06 | 5.34 | 4.53 | 3.86 | 3.71 | 0.276 |
| | 60 | 37.1 | 23.3 | 18.3 | 15.8 | 13.1 | 10.9 | 9.69 | 0.261 |
| | 80 | 94.3 | 58.9 | 45.4 | 35.3 | 33.2 | 24.8 | 23.4 | 0.248 |
| | 100 | 152. | 95.4 | 74.9 | 59.7 | 54.4 | 41.1 | 36.5 | 0.240 |
| 60 | 20 | 2.74 | 2.25 | 2.04 | 1.75 | 1.62 | 1.66 | 1.80 | 0.591 |
| | 40 | 9.26 | 7.23 | 6.36 | 5.17 | 4.32 | 3.86 | 3.77 | 0.407 |
| | 60 | 21.4 | 15.7 | 13.6 | 11.0 | 9.55 | 8.49 | 8.22 | 0.384 |
| | 80 | 39.4 | 27.7 | 25.1 | 21.7 | 18.4 | 16.7 | 15.1 | 0.383 |
| | 100 | 55.7 | 41.4 | 35.7 | 29.9 | 25.8 | 23.1 | 21.3 | 0.382 |

$\mathrm{Tmin} \triangleq \min_{d}(\mathrm{T})$

算時間をT（$d=1$）で，それぞれ表わすことにする．第2章で述べた擬似ブール計画法のアルゴリズム（アルゴリズム2.1）で用いられている，分割変数を選択する手続き（手続き2.3）は，本章で述べた手続き4.2で$d=1$の場合と等価である．よって，Tmin／T($d$=1)は，分割変数の選択方法の改良にともなう，システムの計算時間の短縮度と考えられる．表4.1より，Tmin／T($d$=1)の値は，$n$が大きい場合；$s$が小さい場合；$m$が大きい場合に，ともに減少する傾向が見られる．すなわち，手続き4.2は，第2章で述べた選択方法と比較して，より大規模な問題に対して，効率がよいと考えられる．

### 4.7.3 実 験 2

次に，目的関数を構成する積項の間の含意関係を用いて，目的関数の下界のより厳密な評価を与える，手続き4.1の有効さを調べるため，次の実験を行なった．

まず，アルゴリズム4.1における目的関数の下界を，式（4.9）を直接用いて評価するシステムと，手続き4.1を用いて評価するシステムとを実現した．ただし，これらのシステムで，分割変数の選択を行なう手続き（手続き4.2）のパラメータ$d$の値は，変数の個数$n$に固定した．

次に，$n=30$，$s=0$，$m=20, 40, 60, 80, 100$ の5種類の問題各10問を上記の2つのシステムで解き，システムの計算時間および分割の回数を測定した．図4.4に，両システムの平均計算時間および分割の回数の平均値を示す．

同図より，手続き4.1を用いることにより，分割の回数，計算時間が共にかなり改善されることが知られる．すなわち，計算時間は，$m=20$の場合を除いて，23～33％程度短縮され，$m$の値が増すのに従って，より短縮される傾向がある．また，分割の回数は，$m$の値が増すのに従ってより削減される傾向がある．特に，$m=100$の場合には，約67％削減されている．$m$の値が増すのに従って分割の回数がより削減されるのは，$m$の値が増すのに従って，含意関係が成り立つ積項の対が増すことにより，式（4.9）の方法よりも厳密な下界が得られやすくなるためであろう．また，$m$の値が増すときに，計算時間が短縮される割合ほどには計算時間が短縮されないのは，4.3節でも述べたように，式（4.9）の計算時間が$O(m)$であるのに対し，手続き4.1の計算時間が$O(m^2)$であることによると考えられる．

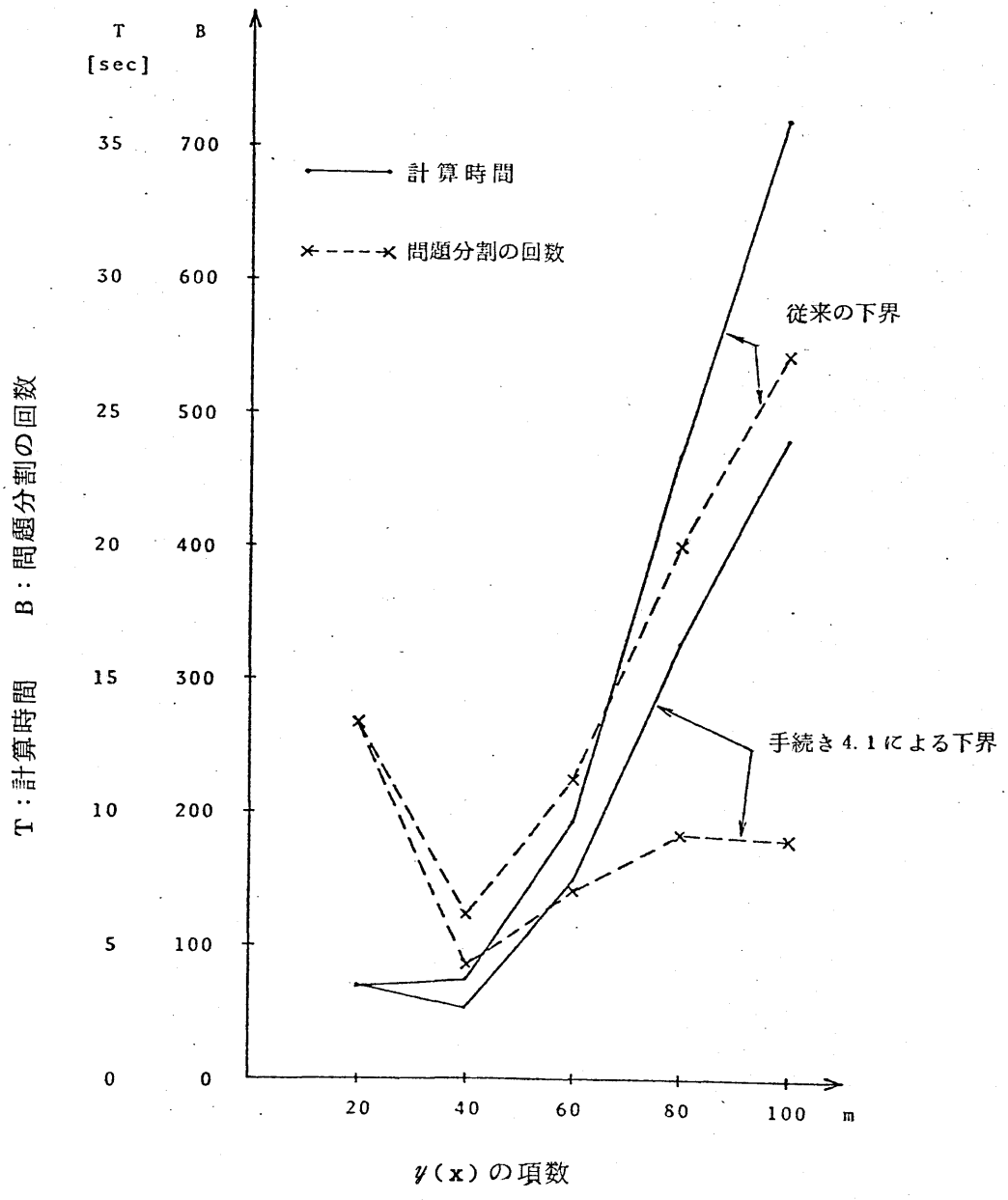



図4.4　含意の関係を用いた目的関数の下界の下界の評価方法の効果

## 4.8 むすび

本章では，擬似ブール計画法のアルゴリズムで用いられる評価関数の強力さとアルゴリズム全体の計算時間との関係について考察を行なった．さらに，4.7の実験を通じて，次の2点を明らかにした．

まず，手続き4.2のパラメータ$d$の値を，解かれる問題の性質に応じて変化させることにより，アルゴリズム全体の計算時間を短縮することができる．つぎに，含意の関係を用いて目的関数の下界を評価することにより，アルゴリズムの計算時間が効果的に短縮される．

手続き4.2を用いて問題を解く場合，計算時間を最小にするパラメータ$d$の値は，解かれる

問題の性質によって異なる．よって，解かれる問題の性質に応じてパラメータ $d$ の値を動的に変更して問題を解くシステムを構成することが可能である．そのためには，擬似ブール計画問題の特徴の抽出と自動分類を行なう必要があるが，これは今後の課題である．

また，4.5で示した擬似ブール計画法のアルゴリズムでは，手続き4.2によって分割変数の決定を行なっており，かつ，次に解くべき部分問題が，depth-first mannerで選ばれる．よって，それ以降で解かれる部分問題の分割変数の決定を，ブール関数および擬似ブール関数の処理とある程度独立に行なうことができる．そこで，アルゴリズム4.1から分割変数の決定を行なう部分を分離し，それぞれの部分を別の処理装置（CPU）上で，同期をとりながら並行（concurrent）に実行することが可能である．これによって，アルゴリズムの計算時間がさらに短縮されると期待される．

この並行処理は，第6章で述べる，Branch-and-Boundアルゴリズムの並列化とは異なっているが，第6章の並列化とも共存しうる方法である．上で述べた並行処理のためのアルゴリズムの開発は今後の課題である．

# 第5章　最小被覆問題の擬似ブール計画法による解法

## 5.1　まえがき

最小被覆問題は，論理回路の簡単化の問題に端緒を発している.[5]　また，この問題以外にも，グラフ理論，オペレーションズ・リサーチなどの分野で，最小被覆問題に帰着されるものがいくつか知られている.[2,3]

最小被覆問題は，典型的な0－1計画問題の1つであるが，問題の解法としては，線形計画法の修正によるもの[2]およびBranch-and-Bound法によるもの[1]などが知られている．整数計画問題としての最小被覆問題は，非常に単純な形をしているが，また，効率的な解法が得られにくい問題として知られている．その理由の一つとしては，線形の目的関数の各変数の重み（係数）が全て1であるために，一般的な整数計画問題の解法で用いられる，目的関数の下界がそのままでは有効に利用できないことがあげられる．

よって，最小被覆問題の効率的解法を明らかにすることは，理論的興味のみならず，実用的見地からも意義があると考えられる．本章では，制約条件の性質を利用して目的関数の良い下界を得る方法を提案する．つぎに，第2章，第4章で述べてきた汎用アルゴリズムを基礎とする，最小被覆問題向きの効率的解法を明らかにする．さらに，このアルゴリズムを計算機プログラムとして実現し，その効率を従来の解法と比較する．

以下では，まず5.2で，最小被覆問題を擬似ブール計画問題として定式化する．次に，5.3では，制約条件式の性質を用いて，目的関数の下界を評価する方法について述べる．5.4では，制約条件における各変数の出現回数を用いて分割変数をえらぶ方法について述べる．5.5では，5.3および5.4での考察にもとづいて最小被覆問題を解くアルゴリズムを提案する．最後に，5.6では，このアルゴリズムの効率を調べる実験の結果について述べる．

## 5.2　最小被覆問題の擬似ブール計画法による定式化

まず，$x_j \in \{0, 1\}$, for　$1 \leq j \leq n$とし，変数$x_j$を要素とするベクトルを，$\mathbf{x}=(x_1, x_2, \cdots, x_n)$で表わす．次に，$\mathbf{x}$を変数とするブール関数$F(\mathbf{x})$を次式で定義する．

$$F(\mathbf{x}) \triangleq F_1(\mathbf{x}) \cdot F_2(\mathbf{x}) \cdot \cdots\cdots \cdot F_m(\mathbf{x}) \qquad (5.1)$$

$$F_i(\mathbf{x}) \triangleq a_{i1} \cdot x_1 \vee a_{i2} \cdot x_2 \vee \cdots\cdots \vee a_{in} \cdot x_n, \quad \text{for} \quad 1 \leq i \leq m \qquad (5.2)$$

ここに，$a_{ij} \in \{0, 1\}$，$1 \leq i \leq m$，$1 \leq j \leq n$とする．

$F(\mathbf{x})$を制約条件式とする最小被覆問題$P(y, F)$を，次のような擬似ブール計画問題とし

て定式化する.*

〔最小被覆問題〕

ブール制約条件：

$$F(x) = 1 \qquad (5.3)$$

のもとで，目的関数：

$$y(x) \triangleq x_1 + x_2 + \cdots + x_n \qquad (5.4)$$

を最小化するxをもとめよ.

上記のように定式化される最小被覆問題は，一般の擬似ブール計画問題と比較して，次の2つの特徴を持つ.

(1) 制約条件式F(x)に含まれる変数は，常に肯定の形で現れる.

(2) 目的関数は線形，かつ，各変数の係数は全て1である.

本章で示す最小被覆問題の解法は，第2章，第4章などで示した一般の擬似ブール計画問題の解法を，これらの特徴を積極的に用いて，問題向きに改良して得られたものである.

## 5.3 目的関数の下界の評価

この節では，前節で述べた，最小被覆問題の特徴を用いて，目的関数の下界に関する定理を導く．さらに，この定理を用いて，目的関数の下界を評価する手続きを与える.

まず，式(5.1)で与えられるブール関数F(x)において，変数$x_j$を含む和項$F_i(x)$の個数を$t_j$で表わす．すなわち，

$$t_j \triangleq |\{ i \mid 1 \leq i \leq m,\ a_{ij} = 1 \}| \qquad (5.5)$$

次に，$t_j$, $1 \leq j \leq n$を大小順にならべかえて得られる$n$項組を$T$で表わす．すなわち，

$$T \triangleq (t_{(1)}, t_{(2)}, \cdots, t_{(n)}) \qquad (5.6)$$

ここに，

$$t_{(j)} \geq t_{(j+1)} \quad , \text{for} \quad 1 \leq \forall j \leq n-1 \qquad (5.7)$$

とする.

---

*) 最小被覆問題の制約条件は，次式のような不等式条件で与えられる場合が多い.

$$a_{i_1} \cdot x_1 + a_{i_2} \cdot x_2 + \cdots + a_{i_n} \cdot x_n \geq 1, \ \text{for} \ 1 \leq i \leq m$$

明らかに，この条件は，式(5.3)のブール条件と等価である．本章では，第2章などでの擬似ブール計画問題の定式化と統一的に扱うために，最小被覆問題を式(5.3)を用いて定式化する.

〔定義 5.1〕

最小被覆問題 $P(y, \mathrm{F})$ に対して，ブール制約条件：$\mathrm{F}(\mathbf{x}) = 1$ を満足する可能解の集合を $S(\mathrm{F})$ で表わす．すなわち，

$$S(\mathrm{F}) \triangleq \{ \mathbf{x} \mid \mathbf{x} \in \{0, 1\}^n, \ \mathrm{F}(\mathbf{x}) = 1 \} \tag{5.8}$$

また，最小被覆問題 $P(y, \mathrm{F})$ の最適解の集合を $S^*(y, \mathrm{F})$ で，また，最適解に対する目的関数の値を $C(y, \mathrm{F})$ で，それぞれ表わす．すなわち，

$$S^*(y, \mathrm{F}) \triangleq \{ \xi \mid \xi \in S(\mathrm{F}), \ [\forall \mathbf{x} \in S(\mathrm{F}), \ y(\xi) \leq y(\mathbf{x})] \}, \tag{5.9}$$

$$C(y, \mathrm{F}) \triangleq \min \{ y(\mathbf{x}) \mid \mathbf{x} \in S(\mathrm{F}) \} \tag{5.10}$$

与えられた $P(y, \mathrm{F})$ に対して，$C(y, \mathrm{F})$ の下界を評価する手続きの基礎を与える定理を以下に示す．

〔定理 5.1〕(Bounding Theorem)

$\mathrm{F}(\mathbf{x})$ を式（5.1）で与えられるブール関数，$T$ を $\mathrm{F}(\mathbf{x})$ に対して，式（5.6）で定義される $n$ 項組とする．この時，次の 2 つの命題が成り立つ．

(1) $C(y, \mathrm{F}) = 1$ iff $t_{(1)} = m$ （5.11）

(2) $C(y, \mathrm{F}) \geq k$ if $\sum_{i=1}^{k-1} t_{(i)} < m$, for $k \geq 2$ （5.12）

〔証明〕 定理 5.1 の証明は付録 4 (1)に示す．

定理 5.1 より，$C(y, \mathrm{F})$ の下界を評価する手続きが次のように与えられる．

〔手続き 5.1〕（$C(y, \mathrm{F})$ の下界）

**入力**：最小被覆問題 $P(y, \mathrm{F})$

**出力**：$C(y, \mathrm{F})$ の下界 $\mathrm{L}(y, \mathrm{F})$

**方法**：

Step 1：$\mathrm{F}(\mathbf{x})$ に対して，式（5.6）で与えられる $T$ を構成する；$s \leftarrow 0$，$k \leftarrow 0$ として，Step 2 へ；

Step 2：(i) $s \geq m$ ならば，Step 3 へ；

(ii) $s < m$ ならば，まず，$k \leftarrow k + 1$ とし，次に，$s \leftarrow s + t_{(k)}$ として，Step 2 へ；

Step 3 : $L(y, F) \leftarrow k + y(\mathbf{x})$ の定数項；手続き終了.

次に，簡単な例題を示し，この例題に対して，$L(y, F)$ の値を求めてみる.

〔例題 5.1〕

ブール制約条件：

$$
\begin{aligned}
F(\mathbf{x}) = &(x_1 \vee x_3 \vee x_7) \cdot (x_2 \vee x_6 \vee x_{10}) \cdot \\
&(x_1 \vee x_3 \vee x_8) \cdot (x_2 \vee x_4 \vee x_8) \cdot \\
&(x_4 \vee x_5 \vee x_{10}) \cdot (x_3 \vee x_6 \vee x_9) \cdot \\
&(x_4 \vee x_7 \vee x_8) \cdot (x_1 \vee x_8 \vee x_{10}) \cdot \\
&(x_3 \vee x_5 \vee x_9) \cdot (x_4 \vee x_6 \vee x_{10}) = 1
\end{aligned}
\tag{5.13}
$$

のもとで，

$$y(\mathbf{x}) = x_1 + x_2 + \cdots + x_{10} \tag{5.14}$$

を最小にする $\mathbf{x}$ を求めよ.

まず，式（5.15）に対する $T$ を求める.

$$
\begin{aligned}
&t_1 = |\{1, 3, 8\}| = 3, , \quad t_2 = |\{2, 4\}| = 2, , \\
&t_3 = |\{1, 3, 6, 9\}| = 4, \quad t_4 = |\{4, 5, 7, 10\}| = 4, , \\
&t_5 = |\{5, 9\}| = 2, , \quad t_6 = |\{2, 6, 10\}| = 3, , \\
&t_7 = |\{1, 7\}| = 2, \quad t_8 = |\{3, 4, 7, 8\}| = 4, , \\
&t_9 = |\{6, 9\}| = 2, \quad t_{10} = |\{2, 5, 8, 10\}| = 4,
\end{aligned}
$$

$$\therefore \ T = (4, 4, 4, 4, 3, 3, 2, 2, 2, 2)$$

$$t_{(1)} + t_{(2)} = 8 < 10,$$

$$t_{(1)} + t_{(2)} + t_{(3)} = 12 \geq 10$$

であることより，$L(y, F) = 3$ を得る.

例題 5.1 の最適解に対する目的関数の値 $C(y, F)$ は，3 である．よって，この例題に対して，$L(y, F) = C(y, F)$ であることが知られる.

## 5.4 問題の分割

第 2 章，第 4 章でも述べたように，Branch-and-Bound アルゴリズムの計算時間は,問題の分割方法によって大きな影響を受ける．この節では，目的関数の下界および上界を用いた，問

題の効果的な分割方法について述べる。

最小被覆問題 $P(y, \mathbf{F})$ に含まれる変数の1つに，値0および1を与えることによって，$P(y, \mathbf{F})$ の2つの部分問題を得ることができる。以下では，問題の分割を行なう際に値が割り当てられる変数を，分割変数と呼ぶことにする。

次の節で示す最小被覆問題を解くアルゴリズムでは，$\mathbf{F}(\mathbf{x})$ 中に最も多く現われる変数，すなわち，式（5.6）の $T$ の第1要素 $t_{(1)}$ に対応する変数を分割変数として選択する。$\mathbf{F}(\mathbf{x})$ 中に最も多く現われる変数を分割変数として選択するのは，以下の理由による。

$y(\mathbf{x})$, $\mathbf{F}(\mathbf{x})$ に含まれる変数 $x_j$ に値 $v$ を代入して得られる式を，それぞれ $y(x_j=v)$, $\mathbf{F}(x_j=v)$ で表わす。また，$\mathbf{F}(\mathbf{x})$ の項数に $y(\mathbf{x})$ の定数項を加えた値は，明らかに $C(y, \mathbf{F})$ の値の上界を与える。この値を，以下では $\mathrm{U}(y, \mathbf{F})$ で表わす。すなわち，

$$\mathrm{U}(y, \mathbf{F}) \triangleq m + y(\mathbf{x})\text{の定数項} \qquad (5.15)$$

〔定理 5.2〕

$\mathbf{F}(\mathbf{x})$ 中に最も多く現われる変数のうちの任意の1つを $x_j^*$ とすると，次の2つの不等式が成り立つ。

(i) $\mathrm{L}(y(x_j^*=0), \mathbf{F}(x_j^*=0)) \geq \mathrm{L}(y(x_j=0), \mathbf{F}(x_j=0))$

$$\text{for} \quad 1 \leq \forall j \leq n \qquad (5.16)$$

(ii) $\mathrm{U}(y(x_j^*=1), \mathbf{F}(x_j^*=1)) \leq \mathrm{U}(y(x_j=1), \mathbf{F}(x_j=1))$

$$\text{for} \quad 1 \leq \forall j \leq n \qquad (5.17)$$

〔証明〕　定理 5.2 の証明は，付録 4 (2)に示す。

定理 5.2 は，次の2点を主張している。

(1) $P(y, \mathbf{F})$ に含まれる各変数に値0を割り当てて得られる $n$ 個の部分問題の目的関数の下界の最大値は，変数 $x_j^*$ に値0を割り当てたとき得られる。

(2) $P(y, \mathbf{F})$ に含まれる各変数に値1を割り当てて得られる $n$ 個の部分問題の目的関数の上界の最小値は，変数 $x_j^*$ に値1を割り当てて得られる。

以上の理由により，$\mathbf{F}(\mathbf{x})$ 中に最も多く現われる変数 $x_j^*$ を分割変数として選択することにする。

また，変数 $x_j^*$ によって分割された2つの部分問題のうち，$x_j^*$ に1を割り当てて得られた部分問題の方が，$x_j^*$ に0を割り当てて得られた部分問題よりも有望な部分問題であることが，定理 5.2 から知られる。

## 5.5 最小被覆問題のアルゴリズム

この節では，最小被覆問題の性質を利用した，問題の縮小の方法について述べ，次に，最小被覆問題を解くアルゴリズムについて述べる．

### 5.5.1 問題の縮小

$C(y, \mathrm{F})$の値をcとすると，最小被覆問題$P(y, \mathrm{F})$の最適解の個数$|S^*(y, \mathrm{F})|$は最大$\binom{n}{c}$個となりうる．また，実用的見地から考えても，全ての最適解を求める必要がある場合は比較的少ないと思われる．よって以下の議論では，最適解のうちの1個のみを求める場合について考える．

最小被覆問題を解くアルゴリズムの効率を向上させる上で，定理5.1から導びかれる次の2つの性質は重要である．

まず，式(5.6)で与えられる$T$で，$t_{(1)}$に対応する変数を$x_j{}^*$とする．この時，次の命題が成り立つ．

$$C(y(x_j{}^*=0), \mathrm{F}(x_j{}^*=0)) \geq C(y(x_j{}^*=1), \mathrm{F}(x_j{}^*=1)), \text{ if } t_{(1)}=m \tag{5.18}$$

この命題が成り立つことは，定理5.1の(1)より明らかである．この命題より，以下のことが知られる．すなわち，$\mathrm{F}(\mathbf{x})$に含まれる変数のなかに，$t_j{}^*=m$である変数$x_j{}^*$が存在すれば，解の最適性をそこなうことなく，部分問題$P(y(x_j{}^*=0), \mathrm{F}(x_j{}^*=0))$の処理を中止してよい．

次に，$\mathrm{F}(\mathbf{x})$の項数が2である場合について考える．すなわち，$\mathrm{F}(\mathbf{x})$は次式で表わされる．

$$\mathrm{F}(\mathbf{x})=\mathrm{F}_1(\mathbf{x})\cdot\mathrm{F}_2(\mathbf{x}) \tag{5.19}$$

$$\mathrm{F}_1(\mathbf{x})=a_{11}\cdot x_1 \vee a_{12}\cdot x_2 \vee \cdots \vee a_{1n}\cdot x_n \tag{5.20}$$

$$\mathrm{F}_2(\mathbf{x})=a_{21}\cdot x_1 \vee a_{22}\cdot x_2 \vee \cdots \vee a_{2n}\cdot x_n \tag{5.21}$$

この時，$y(\mathbf{x})$の定数項をcとすると，次の2つの命題が成り立つ．

(1) $C(y, \mathrm{F})=c+1$ if $t_{(1)}=2$ (5.22)

(2) $C(y, \mathrm{F})=c+2$ if $t_{(1)}=1$ (5.23)

これらの2つの命題が成り立つこともまた，定理5.1より明らかである．これらの命題より，以下のことが知られる．まず，$m=2$の場合，$\mathrm{F}_1(\mathbf{x})$，$\mathrm{F}_2(\mathbf{x})$に共通に含まれる変数$x_j{}^*$が存在すれば，$x_j{}^*=1$として得られる解は$P(y, \mathrm{F})$の最適解を与える．また，これは，式(5.18)の命題で$m=2$の場合でもある．

次に，$\mathrm{F}_1(\mathbf{x})$，$\mathrm{F}_2(\mathbf{x})$に共通に含まれる変数が存在しない場合には，$\mathrm{F}_1(\mathbf{x})$，$\mathrm{F}_2(\mathbf{x})$に含まれる変数をそれぞれ1つずつえらび，これらを$x_{j_1}$，$x_{j_2}$とする．このとき，$x_{j_1}=1$，$x_{j_2}=1$として得られる解は，$P(y, \mathrm{F})$の最適解を与える．

$m \geq 3$以上の場合にも，同様の性質を用いて部分問題の縮小を試みることも可能である．し

かし，$m \geq 3$ 以上の場合には，各 $F_i(\mathbf{x})$ に含まれる変数の組み合せがかなり多くなるので，上記のようにして問題の縮小を行なうことが得策であるかどうかはやや疑問である．よって，ここでは，このような場合には問題の縮小は陽には行なわないことにする．

さらに，$F(\mathbf{x})$ 中に，単一の変数 $x_j$ のみからなる項 $F_i(\mathbf{x})$ が存在すれば，制約条件が $F(\mathbf{x}) = 1$ であることから，変数 $x_j$ に値 1 を代入してよい．

以下に示す最小被覆問題のアルゴリズムでは，以上のような問題の縮小のみを行なうことにする．

### 5.5.2 アルゴリズム

最小被覆問題を解くアルゴリズムを以下に示す．

〔アルゴリズム 5.1〕

**入力**: 最小被覆問題 $P(y, F)$；

**出力**: $P(y, F)$ の最適解 $\mathbf{x}_{opt}$；

$C(y, F)$ の値　Copt；

**記法**: S を，未解決な部分問題に対応するベクトル $\mathbf{x}$ を要素とするプッシュダウン・スタックとする．S の先頭に要素 $\mathbf{x}$ をプッシュダウンする操作を，$S \Leftarrow \mathbf{x}$ で表わす．また，S の先頭から要素を 1 個ポップアップして $\mathbf{x}$ に代入する操作を，$\mathbf{x} \Leftarrow S$ で表わす．さらに，S は，アルゴリズムの実行前に空 ($\varnothing$) に初期化されているとする．

さらに，変数 $x_j$ の値が未定であることを，$x_j = u$ で表わす．

**方法**:

Step 1：〔初期設定〕

$S \Leftarrow (u, u, \cdots, u)$，Copt $\leftarrow \infty$ とする；　Step 2 へ；

Step 2：〔バックトラック〕

(i)　$S = \varnothing$ ならば，Step 7 へ；

(ii)　$S \neq \varnothing$ ならば，$\mathbf{x}_0 \Leftarrow S$ として，　Step 3 へ；

Step 3：〔部分問題の簡単化*〕

$F(\mathbf{x}_0)$ を簡単化して得られる式を $F'(\mathbf{x}_0)$ とする；

---

*）ここでの $F(\mathbf{x}_0)$ の簡単化とは，次の 2 つの操作を意味する．

(1) 単一の変数 $x_j$ のみからなる項 $F_i(\mathbf{x}_0)$ が存在すれば，変数 $x_j$ に値 1 を代入する．

(2) 同一項の除去

これらの 2 つの操作は，第 3 章で述べたハッシングの技法を用いて高速で実行することができる．

(i) $L(y(x_0),\ F'(x_0)) \geq Copt$ または，

$F'(x_0) \equiv 0$ ならば，Step 2へ；

(ii) $F'(x_0) \equiv 1$ ならば，Step 6へ；

(iii) それ以外の時，Step 4へ；

Step 4：〔問題の縮小〕

(i) $t_j{}^* = m$ である変数 $x_j{}^*$ が $F'(x_0)$ 中に存在すれば，$x_j{}^* \leftarrow 1$ としてStep6へ；

(ii) $t_j{}^* = m$ である変数 $x_j{}^*$ が $F'(x_0)$ 中に存在しない場合，$m = 2$ であれば，

$F'_1(x_0)$，$F'_2(x_0)$ から，各々1個ずつ変数をえらび $x_{j_1}$，$x_{j_2}$ とする；$x_{j_1} \leftarrow 1$，

$x_{j_2} \leftarrow 1$ としてStep 6へ；

(iii) それ以外の時，Step 5へ；

Step 5：〔問題の分割〕

$F'(x_0)$ 中に最も多く現われる変数を1つえらび，これを $x_j{}^*$ とする；

$x_0 \equiv (\tilde{x}_1,\ \tilde{x}_2,\ \cdots,\ \tilde{x}_n)$ とする；

$S \Leftarrow (\tilde{x}_1,\ \tilde{x}_2,\ \cdots,\ \tilde{x}^*_{j-1},\ 0,\ \tilde{x}^*_{j+1},\ \cdots,\ \tilde{x}_n)$；

$x_0 \leftarrow (\tilde{x}_1,\ \tilde{x}_2,\ \cdots,\ \tilde{x}^*_{j-1},\ 1,\ \tilde{x}^*_{j+1},\ \cdots,\ \tilde{x}_n)$

として，Step 3へ；

Step 6：〔最適解の候補の更新〕

$x_0$ 中の $x_j = u$ である全ての $j$ について，

$x_j \leftarrow 0$ とする；

$Xopt \leftarrow x_0$，$Copt \leftarrow y(x_0)$ として，Step 2へ；

Step 7：〔アルゴリズム終了〕

停止．

このアルゴリズムは，最初にみつけられた最適解を1つだけ出力して停止する．もし，全ての最適解が必要であれば，Step3, Step4, Step6を修正すればよい．

### 5.5.3 例題とその解法

次に，5.3で示した例題5.1が，アルゴリズム5.1で解かれる過程を少しくわしく追ってみる．以下の解法で，Stage $i$ ($i = 0,\ 1,\ \cdots$) は，原問題を分割して得られる $i$ 番目の部分問題の処理を行なっていることを表わす．

例題5.1の制約条件式 $F(x)$ は，次式のブール関数で与えられる．

$$F(\mathbf{x}) = (x_1 \vee x_3 \vee x_7)\cdot(x_2 \vee x_6 \vee x_{10})\cdot(x_1 \vee x_3 \vee x_8)\cdot(x_2 \vee x_4 \vee x_8)\cdot$$
$$(x_4 \vee x_5 \vee x_{10})\cdot(x_3 \vee x_6 \vee x_9)\cdot(x_4 \vee x_7 \vee x_8)\cdot(x_1 \vee x_8 \vee x_{10})\cdot$$
$$(x_3 \vee x_5 \vee x_9)\cdot(x_4 \vee x_6 \vee x_{10})$$

〔例題の解法〕

Stage 0 :

Step 1 : $S \Leftarrow (u, u, u, \cdots, u)$, $\mathrm{copt} \leftarrow \infty$ ;

Step 2 : $S = ((u, u, u, \cdots, u))$

$\mathbf{x}_0 \leftarrow (u, u, u, \cdots, u)$ ;

Step 3 : $F'(\mathbf{x}_0) = F(\mathbf{x})$, $y(\mathbf{x}_0) = x_1 + x_2 + \cdots + x_{10}$ ;

$T = (4, 4, 4, 4, 3, 3, 2, 2, 2, 2)$,

$L(y(\mathbf{x}_0), F'(\mathbf{x}_0)) = 3$ ;

Step 4 : ——

Step 5 : $t_3 = t_{(1)} = 4$ であるから, $j^* \leftarrow 3$,

$S \Leftarrow (u, u, 0, u, u, u, u, u, u, u)$,

$\mathbf{x}_0 \leftarrow (u, u, 1, u, u, u, u, u, u, u)$,

Stage 1 :

Step 3 : $F'(\mathbf{x}_0)$

$= (x_2 \vee x_6 \vee x_{10})\cdot(x_2 \vee x_4 \vee x_8)\cdot(x_4 \vee x_5 \vee x_{10})\cdot$
$(x_4 \vee x_7 \vee x_8)\cdot(x_1 \vee x_8 \vee x_{10})\cdot(x_4 \vee x_6 \vee x_{10})$

$y(\mathbf{x}_0) = 1 + x_1 + x_2 + x_4 + x_5 + \cdots + x_{10}$

$T = (4, 4, 3, 2, 2, 1, 1, 1, 0)$

$L = (y(\mathbf{x}_0), F'(\mathbf{x}_0)) = 3$ ;

Step 4 : ——

Step 5 : $t_4 = t_{(1)} = 4$ であるから, $j^* \leftarrow 4$,

$S \Leftarrow (u, u, 1, 0, u, u, u, u, u, u)$,

$\mathbf{x}_0 \leftarrow (u, u, 1, 1, u, u, u, u, u, u)$ ;

Stage 2 :

Step 3 : $F'(\mathbf{x}_0)$

$= (x_2 \vee x_6 \vee x_{10})\cdot(x_1 \vee x_8 \vee x_{10})$

$y(\mathbf{x}\ ) = 2 + x_1 + x_2 + x_5 + x_6 + \cdots + x_{10}$

$T = (2, 1, 1, 1, 1, 0, 0, 0)$

$L(y(x_0), F'(x_0)) = 3$

Step 4 : $t_{10} = m = 2$ であるから, $x_{10} \leftarrow 1$ ;

Step 6 : Xopt $\leftarrow$ ( 0, 0, 1, 1, 0, 0, 0, 0, 0, 1)

Copt $\leftarrow$ 3 ;

Stage 3 :

Step 2 : $S = ((u, u, 1, 0, u, u, u, u, u, u)$

$(u, u, 0, u, u, u, u, u, u, u))$,

$x_0 \leftarrow (u, u, 1, 0, u, u, u, u, u, u)$ ;

Step 3 : $L(y(x_0), F'(x_0))$ の値は, Stage 1 で処理された, $x_0 = (u, u, 1, u, u, \cdots, u)$ に対する $L(y(x_0), F'(x_0))$ の値すなわち 3 以上である。よって, 明らかに,

$$x_0 = (u, u, 1, 0, u, u, u, u, u, u)$$

に対しても, Copt = 3 であることから,

$$L(y(x_0), F'(x_0)) \geq \text{Copt}$$

であることが知られる ;

Stage 4 :

Step 2 : $S = ((u, u, 0, u, u, u, u, u, u, u))$

$x_0 \leftarrow (u, u, 0, u, u, u, u, u, u, u)$

Step 3 : Stage 3 の場合と同様の理由により,

$L(y(x_0), F'(x_0)) > \text{Copt}$

であることが知られる ;

Stage 5 :

Step 2 : $S = \emptyset$ ;

Step 7 : 停　止 ;

得られた最適解は,

Xopt = ( 0, 0, 1, 1, 0, 0, 0, 0, 0, 1),

Copt = $y$ (Xopt ) = 3

である。

アルゴリズム5.1によって例題5.1が解かれる過程は，図5.1に示す木で表わされる．同図で円内の数字は，先に述べた解法のStageの番号を表わす．

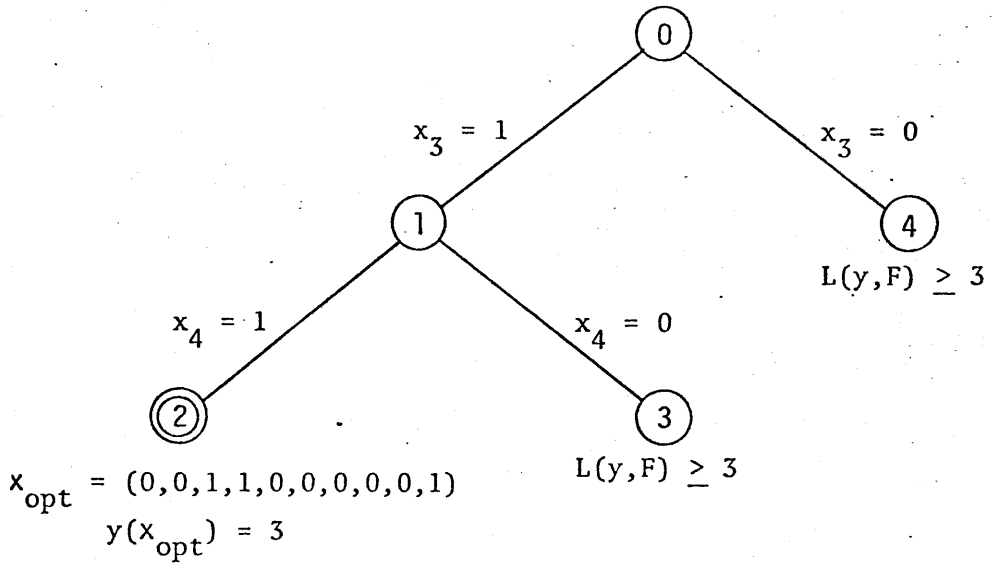



図5.1　例題5.1の解かれる過程

## 5.6　実験およびその結果

この節では，次の3種類の実験とそれらの結果について述べる．

(1)　実験1：手続き5.1を用いた目的関数の下界の評価の正確さの測定．

(2)　実験2：手続き5.1を用いた目的関数の下界および，分割変数の選択方法の効率の測定．

(3)　実験3：アルゴリズム5.1と阿部のアルゴリズム（文献(1)）との効率の比較

### 5.6.1 アルゴリズムの実現と問題の作製方法

アルゴリズム5.1をFORTRAN Nによって記述し，名古屋大学工学部情報工学科のFACOM 230/38上で実現した．実現に要した記憶容量は，問題の作製および入出力関係のルーチンを含めて約54 KBである．

以下に述べる実験で用いられた問題は，次の3つのパラメータを指定して，計算機の一様乱数を用いて発生された．

$n$：問題に含まれる変数の個数；

$m$：制約条件の個数（F(x)の項数）；

g：各条件に含まれる変数の個数．

### 5.6.2 実験1

手続き5.1を用いて得られる目的関数の下界の評価方法の正確さを調べるために，以下の実験を行なった．

まず，$n$，$m$，gの値をそれぞれ変化させ，各$n$，$m$，gの値に対して，各場合10問づつの最小被覆問題$P(y, F)$を発生させた．次に，これらの問題の目的関数の下界$L(y, F)$を手続き5.1を用いて求め，その値と，$P(y, F)$の最適解に対する目的関数の値$C(y, F)$とを比較した．

手続き5.1を用いて得られる目的関数の下界の正確さを，次式によって評価することにする．

$$L(y, F)/C(y, F) \qquad (5.24)$$

表5.1に，L，$C$およびL/$C$の10問題についての平均値を示す．同表で，$n$および$m$の値は問題の規模を，gは問題の複雑さをそれぞれ表わしていると考えられる．同表より，Lの正確さに関して，以下のことが知られる．

表5.1　手続き5.1による目的関数の下界の評価方法の正確さ

| g | n, m | 30,30 | 40, 40 | 50, 50 | 60, 60 | 70, 70 |
|---|---|---|---|---|---|---|
| 3 | L | 6.6 | 8.3 | 9.9 | 12.2 | 14.8 |
| | $C$ | 9.3 | 12.0 | 15.2 | 18.4 | 21.5 |
| | L/$C$ | 0.713 | 0.694 | 0.652 | 0.664 | 0.689 |
| 4 | L | 4.8 | 6.4 | 7.7 | 9.4 | 10.8 |
| | $C$ | 7.0 | 10.1 | 12.3 | 14.6 | 16.8 |
| | L/$C$ | 0.685 | 0.635 | 0.627 | 0.644 | 0.643 |
| 5 | L | 4.1 | 5.3 | 6.4 | 7.5 | 8.8 |
| | $C$ | 6.2 | 8.4 | 10.2 | 12.2 | 14.1 |
| | L/$C$ | 0.666 | 0.634 | 0.629 | 0.615 | 0.625 |

L：手続き5.1によって与えられる下界
$C$：目的関数の最適値
L/$C$：手続き5.1によって与えられる下界の正確さ

(1) 全体として，手続き 5.1 を用いて得られた目的関数の下界は，最適解に対する目的関数の値の，かなり良い下界を与えている．

(2) L（$y$，F）の正確さは，問題の規模（$n$，$m$）の変化に対して，かなり安定である．

(3) L（$y$，F）の正確さは，問題の複雑さ（g）が増すと，いくぶん減ずる傾向がある．

以上の結果より，手続き 5.1 による目的関数の下界の評価方法は，十分実用に耐え得ると考えられる．

### 5.6.3 実 験 2

アルゴリズム 5.1 では，次の 2 つの評価手続きが用いられている．

(i) 目的関数の下界を評価する手続き 5.1

(ii) $t_{(1)}$ に対応する変数を分割変数としてえらぶ手続き

これらの手続きの有効さをそれぞれ調べるために，以下の 2 つの実験を行なった．

(1) まず，$n = 30, m = 30$, g = 3 の問題を 10 問題発生させた．次に，これらの問題を，上記の 2 つの評価手続きを用いる場合と用いない場合の合計 4 通りの方法で解かせた．ただし，手続き 5.1 を用いない場合には，目的関数の下界は，値 1 が割り当てられた変数の個数によって評価した．また，$t_{(1)}$ に対応する変数を分割変数として用いない場合には，分割変数は，その部分問題に含まれる変数の中から，計算機の一様乱数を用いてランダムにえらばれた．

これらの 4 通りの場合について，アルゴリズム 5.1 の計算時間 T，および問題の分割の回数 B の平均値を上記の 10 問題に対して求め表 5.2 に示す．

同表より，両評価手続きの有効さについて，以下のことが知られる．

表 5.2 目的関数の下界を評価する手続きおよび分割変数を選択する手続きの有効さ

(a) 計算時間

| 分割変数＼下界 | 手続き 5.1 | 値 1 を割り当てられた変数の個数 |
|---|---|---|
| $\max_j$ を与える $x_j$ | 0.984 | 39.3 |
| ランダムに選んだ場合 | 15.7 | 362.0 |

(b) 問題の分割の回数

| 分割変数＼下界 | 手続き 5.1 | 値 1 を割り当てられた変数の個数 |
|---|---|---|
| $\max_j(t_j)$ を与える $x_j$ | 36.4 | $2.00 \times 10^3$ |
| ランダムに選んだ場合 | 699 | $2.00 \times 10^4$ |

(i) 分割変数として，$t_{(1)}$ に対応する変数を選択した場合，手続き 5.1 を用いて目的関数の下界を評価することにより，計算時間は約 1/40，問題の分割回数は約 1/55 にそれぞれ改善される．

(ii) 手続き 5.1 を用いて目的関数の下界を評価した場合，$t_{(1)}$ に対応する変数を分割変数として選択することにより，計算時間は約 1/16 に，問題の分割回数は 1/19 に，それぞれ改善される．

(iii) 両手続きを共に用いることにより，計算時間は約 1/370 に，また問題の分割回数は約 1/550 にそれぞれ改善される．

(2) 第 4 章でも述べたように，一般に，評価手続きの複雑さが増すに従がって，その手続き自体の計算時間も増す傾向がある．そのため，より複雑な評価手続きを用いることが，必ずしもアルゴリズム全体の計算時間の短縮に貢献するとは限らない．*

そこで，目的関数の下界の評価および分割変数の選択を以下のようにして行なうことにより，評価関数の複雑さとアルゴリズム全体の計算時間との関係を調べた．

Branch-and-Bound 法で問題が解かれる過程は，図 5.1 に示したような木で表わされる．この木の各節点には，それぞれ次のようにして，部分問題が対応づけられる．まず，木の根 $P_0$ には，解くべき問題が対応づけられる．また，一般に，節点 $P_i$ が節点 $P_i$ の子の一つであることは，$P_j$ に対応する部分問題が $P_i$ に対応する部分問題を直接分割して得られた部分問題の一つであることを表わしている．このとき，木の各節点 $P_i$ のレベルを，$P_0$ から $P_i$ に至るパスの長さによって定義し，この値を $level(P_i)$ で表わす．

分割変数の決定および目的関数の下界の評価は，次のようにして行うことにする．

---

＊） 以下で述べる実験の方針は，第 4 章で述べた，評価関数の強力さと計算時間に関する考察に基づく．

まず，$d$（$1 \leq d \leq n$）を与えられたパラメータとして，部分問題のレベルが0，$d$，$2d$，…，$\lfloor n/d \rfloor \times d$であれば，部分問題に含まれる変数を，F（x）中への出現回数によって順序付けておく．また，同時に，手続き5.1を用いて，その部分問題の目的関数の下界の値をも求めておく．

レベルが0，$d$，$2d$，…，$\lfloor n/d \rfloor \times d$でないような部分問題での分割変数の決定および目的関数の下界の評価は，次のようにして行なう．部分問題$P_i$のレベルを$\ell$とする．また，$P_0$から$P_i$に至るパス上で，レベルが$\lfloor \ell/d \rfloor \times d$である部分問題を$P_j$とする．（図5.2参照）

この時，$P_i$での分割変数を，$P_i$に含まれる変数の中で，$P_j$のF（x）に最も多く含まれていた変数とすることにする．また，$P_i$の目的関数の下界を，$P_i$の目的関数の下界とすることにする．

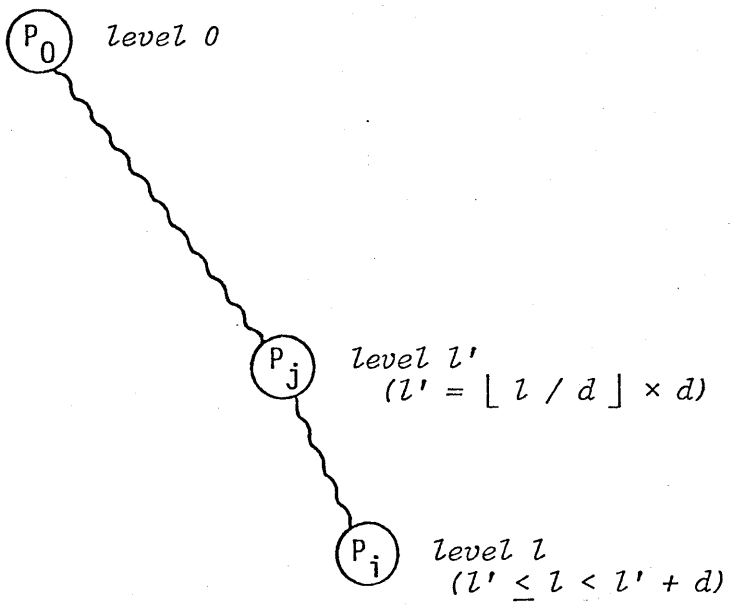



図5.2　部分問題PiとPjの関係

このようにして各部分問題の分割変数の選択および目的関数の下界の評価を行なうと，$d=1$の場合には，アルゴリズム5.1での分割変数の選択方法および目的関数の下界の評価と等価になる．また，$d=n$の場合には，分割変数は，$P_0$のF（x）中での各変数の出現回数によってのみ選択されることになり，また，目的関数の下界は，$P_0$の目的関数の下界をそのまま用いることになる．

よって，パラメータ$d$の値が増せば増すほど，分割変数の選択および目的関数の評価手続きはともに複雑さが減じ，その結果，これらの評価手続きの強力さも同時に減ずると考えられる．

$n = 30$, $m = 30$, $g = 3$ として発生された10問題について，上記の方法でパラメータ $d$ の値を変化させた場合の，アルゴリズムの計算時間および問題の分割回数を図5.3に示す．

同図より，アルゴリズムの計算時間および分割回数は，パラメータ $d$ の値の増加に伴なって急激に増すことが知られる．よって，上記のアルゴリズムではパラメータ $d$ の値は1が最適であると考えられる．（$d = 1$ の場合，上記のアルゴリズムは，アルゴリズム5.1と等価である）

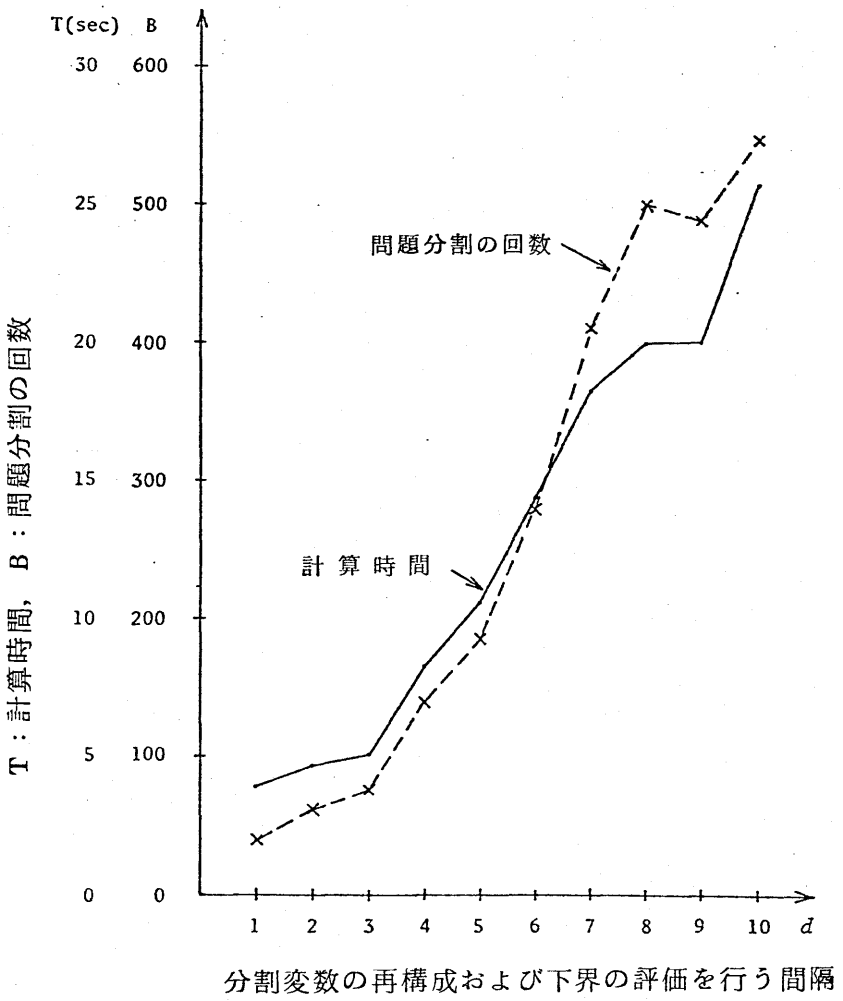



**図5.3　パラメータdの値と計算時間および問題分割の回数の関係**

### 5.6.4　実　験　3

実験3では，アルゴリズム5.1と阿部のアルゴリズム[1]との計算時間の比較を同一計算機上で行なう．阿部のアルゴリズムは，アルゴリズム5.1と同様に，Branch-and-Bound法にもとづいた解法であり，FORTRANで記述されている．

まず，$n$，$m$，gの値をそれぞれ変えて，各10問題ずつの最小被覆問題を発生させ，次に

これらの問題を両アルゴリズムで解いて計算時間を測定した．

表 5.3 に両アルゴリズムの計算時間の平均値と標準偏差を示す．同表より，以下のことが知られる．

(i) $n$，$m$ の値が大きくなるほど，すなわち，解かれる問題の規模が大きくなるほど，アルゴリズム 5.1 の計算時間の，阿部のアルゴリズムの計算時間に対する比の値は小さくなる．

表 5.3　アルゴリズム 5.1 と阿部のアルゴリズムとの計算時間の比較

| アルゴリズム | | アルゴリズム 5.1 | | 阿部の解法 | | 比 |
|---|---|---|---|---|---|---|
| n, m | g | $\mu_I$(sec) | $\sigma_I$(sec) | $\mu_A$(sec) | $\sigma_A$(sec) | $\mu_I/\mu_A$ |
| 10, 10 | 3 | 0.058 | 0.018 | 0.066 | 0.009 | 0.879 |
| | 4 | 0.059 | 0.020 | 0.104 | 0.024 | 0.567 |
| | 5 | 0.054 | 0.028 | 0.135 | 0.044 | 0.400 |
| 20, 20 | 3 | 0.239 | 0.101 | 0.318 | 0.117 | 0.752 |
| | 4 | 0.347 | 0.186 | 0.864 | 0.254 | 0.402 |
| | 5 | 0.287 | 0.147 | 1.22 | 0.264 | 0.235 |
| 30, 30 | 3 | 0.981 | 0.310 | 1.74 | 0.605 | 0.564 |
| | 4 | 0.941 | 0.191 | 4.23 | 1.75 | 0.222 |
| | 5 | 1.91 | 1.15 | 9.94 | 3.05 | 0.192 |
| 40, 40 | 3 | 4.55 | 2.77 | 6.19 | 3.28 | 0.735 |
| | 4 | 12.3 | 5.56 | 45.9 | 19.5 | 0.268 |
| | 5 | 12.5 | 8.25 | 83.1 | 34.4 | 0.150 |
| 50, 50 | 3 | 22.0 | 9.05 | 36.6 | 19.5 | 0.601 |
| | 4 | 35.7 | 25.6 | 268 | 105 | 0.133 |
| | 5 | 50.3 | 28.3 | 632 | 171 | 0.0795 |
| 60, 60 | 3 | 105 | 59.6 | 210 | 76.1 | 0.500 |
| | 4 | 158 | 97.0 | —— | —— | —— |
| | 5 | 157 | 87.6 | —— | —— | —— |
| 70, 70 | 3 | 488 | 432 | 1394 | 708 | 0.350 |
| | 4 | 592 | 298 | —— | —— | —— |
| | 5 | 955 | 838 | —— | —— | —— |

$\mu$：平均値　　$\sigma$：標準偏差

(ii) g の値が大きくなるほど，すなわち，解かれる問題の複雑さが増すほど，アルゴリズム 5.1 の計算時間の阿部のアルゴリズムの計算時間に対する比の値は小さくなる．

以上の結果から，アルゴリズム5.1は，従来の解法の一つである阿部のアルゴリズムと比べて，大規模かつ複雑な最小被覆問題の解法として適していると考えられる．

## 5.7 むすび

本章では，最小被覆問題を擬似ブール計画問題として定式化を行なった．つぎに，制約条件式の性質を用いて目的関数の下界を評価する方法を提案した．また，分割変数の選択方法について考察を行ない，最小被覆問題を解くアルゴリズムを示した．計算機の一様乱数を用いて発生された最小被覆問題を多数解かせた結果，本章で提案した，目的関数の下界の評価方法および分割変数の選択方法は，ともにアルゴリズムの計算時間を効果的に短縮することが知られた．

本章で扱かった最小被覆問題は，式（5.4）のように各変数の係数（重み）が全て1であるような目的関数の最小化の問題である．一方，目的関数：

$$\begin{cases} y'(\mathbf{x}) \triangleq \sum_{i=1}^{n} a_i x_i, \\ a_i > 0, \quad \text{for} \quad 1 \leq i \leq n \end{cases} \tag{5.25}$$

を式（5.3）の制約条件のもとで最小化する問題は，重みつき最小被覆問題と呼ばれる．

重みつき最小被覆問題をBranch-and-Bound法で解く場合には，次式で与えられる目的関数の下界を用いることができる．

$$\min \{ a_i \mid 1 \leq i \leq n \} \times \mathrm{L}(y, \mathrm{F}) \tag{5.26}$$

ここに，L（$y$，F）は，重みつきでない最小被覆問題$P$（$y$，F）の目的関数の下界であり，手続き5.1で与えられる．

また，重みつき最小被覆問題を解く場合には，分割変数は，目的関数の係数$a_i$およびF(x)中での各変数の出現回数の情報にもとづいて選択した方が良いであろう．

以上の考察にもとづく，重みつき最小被覆問題の解法を明らかにするのは今後の課題である．

# 第6章　Branch-and-Boundアルゴリズムの並列化とその評価

## 6.1 まえがき

Branch-and-Bound法は，組合せ論的最適化問題の一般的解法の一つとして，さまざまな分野に適用されてきた.[14] しかし，これらの問題の多くは，整数計画問題などの，NP－完全な問題のクラス[1] に属する．そのため，大規模な問題を解くためには，現在の高速計算機を用いてもなお，多くの計算時間を必要とする．したがって，計算時間をさらに短縮するためには，新たな方法を導入する必要がある．

高い処理能力を持つ計算機システムを実現するための方法の一つとして，MIMD*形計算機システムが提案されてきた.[3,5] また，近年におけるLSI技術の発達にともない，いくつかのシステムがハードウェアとして実現されてきた.[4,6]この形式の計算機システムでは，各処理装置は，たがいに通信しながら，独立に計算を行なう．このような計算機システムの上で動作する並列アルゴリズムの構成方法とその性質を明らかにすることは，単により大規模な組合せ論的最適化問題を解くことを可能にするのみならず，将来の計算機システムの発展に対しても寄与するところは大きい．

本章では，MIMD形の計算機システム（以下では，マルチプロセッサ・システムと呼ぶ）に適した，並列形Branch-and-BoundアルゴリズムPDFA（Parallelized Depth－First Algorithm）を提案する．さらに，$k$－値をとる$n$個の変数によって記述される最適化問題をp台の処理装置を持つマルチプロセッサ・システムでPDFAを用いて解く場合について，次の2つの性質を明らかにする．すなわち，(i)アルゴリズムを実現するために必要な記憶容量は$O(p\times k\times n)$でよい，(ii)処理装置の台数pを適当にえらべば，システムの計算時間は，1台の処理装置で同じアルゴリズムを実行する場合の1／p以下になる．

なお，(ii)については，Fullerらの最近の実験により，整数計画問題をBranch-and-Boundアルゴリズムを用いてMIMD形計算機システムで解いた時の計算時間が，実際に1/p以下になる例が確認されている.[6]

以下本章では，初めに，6.2で並列形Branch-and-Boundアルゴリズムを実現するための，マルチプロセッサ・システムについて述べる．次に，6.3では，アルゴリズムの並列化の方針を示し，さらに，並列化されたアルゴリズムPDFAを記述する．次に，6.4では，PDFAの性質を明らかにする．さらに，6.5では，PDFAの計算時間をシミュレーションによって調べ

---

*）Multiple－Instruction Stream－Multiple－Data Streamの略

た結果を示し，6.4での結論との比較を行なう．最後に，6.6では，並列形Branch-and-Boundアルゴリズムの実現上の問題点，および今後の課題について述べる．

## 6.2 マルチプロセッサ・システムのモデル

この節では，並列形Branch-and-Boundアルゴリズムを実現するために必要な，マルチプロセッサ・システムのモデルについて述べる．

本章で考察の対象とするマルチプロセッサ・システムは図6.1に示すような，MIMD形計算機システムである．すなわち，システムは次の3種類の構成要素からなる：

(i) p台の処理装置（$p \geq 1$）

(ii) 各処理装置に接続された，専用記憶装置

(iii) 各処理装置から共通に参照できる，共有記憶装置*

以下では，マルチプロセッサ・システムに関して，次の仮定を置く：

〔仮定6.1〕 p台の処理装置は全て能力が等しい．

〔仮定6.2〕 それぞれの処理装置は互いに独立に計算を行なうことができる．

〔仮定6.3〕 各処理装置間の通信は，共有記憶装置を通じてのみ行なわれるとし，各処理装置間の直接の割り込みなどを用いては行なわない．

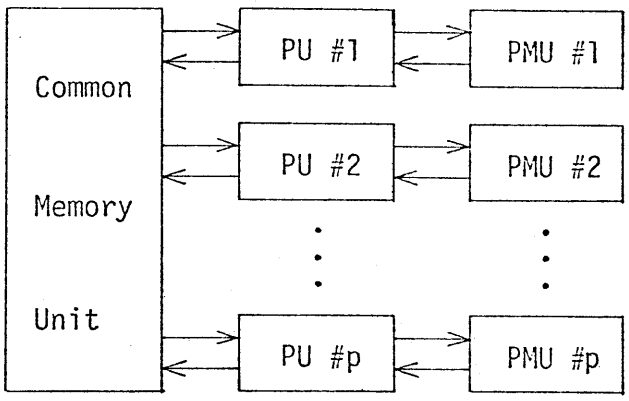



図6.1 マルチプロセッサ・システムのモデル

PU : Processing Unit（処理装置）
PMU : Private Memory Unit（専用記憶装置）
CMU : Common Memory Unit（共有記憶装置）

---

*）各処理装置と共有記憶装置との結合の方式としては，スイッチ・マトリックスによる方法，バス結合による方法などが提案されている．（3, 4, 6）

## 6.3 Branch−and−Boundアルゴリズムの並列化

この節では，Branch-and-Boundアルゴリズムの並列化の方針を示し，前節で述べたマルチプロセッサ・システムの上で動作する，並列形Branch-and-BoundアルゴリズムPDFAを記述する．

なお，以下では，$k$ −値をとる$n$個の変数によって記述される最小化問題 $P_0$ を解く場合について考察する．

### 6.3.1 諸定義および記法

問題$P_0$ がBranch-and-Boundアルゴリズムによって解かれる過程を，木$B=(V, E)$で表わす．ここに，$V$ および$E$はそれぞれ，木の節点の集合および枝の集合とする．$V$ の各要素には，問題$P_0$ の部分問題の1つが1対1に対応する．とくに，木$B$の根には問題$P_0$ が対応する．また，$E$ 中の枝$(P_i, P_j)$は，$P_j$ が$P_i$ を分割* して得られる部分問題の1つであることを表わす．また，木$B$の根$P_0$ から節点$P_i$ に至るパスの長さを$P_i$ のレベルと呼び，$level(P_i)$で表わす．

次に，木$B$の葉節点の集合を$L$で表わす．$L$の各要素は，次のいずれかの部分問題の1つに対応する：

(i) 全ての変数に値が割り当てられた部分問題

(ii) 制約条件を満足しない部分問題

次に，部分問題$P_i$ に対して，その最適解が与える目的関数の値を，次式の$cost(P_i)$によって定義する．*

$$cost(P_i) \triangleq \begin{cases} P_i \text{ の目的関数の値} & \text{if } P_i \in L \\ \min\{cost(P_j) \mid (P_i, P_j) \in E\} & \text{if } P_i \notin L \end{cases} \qquad (6.1)$$

さらに，$cost(P_i)$の下界を与える関数を$\ell b(P_i)$で表わす．ここに，$\ell b(P_i)$は次式が成り立つように定める．**

$$\ell b(P_i) = cost(P_i), \quad \text{for } P_i \in L \qquad (6.2)$$

$$\ell b(P_i) \leq \ell b(P_j), \quad \text{for } (P_i, P_j) \in E \qquad (6.3)$$

---

*) 問題の分割は，問題に含まれる変数の1つに，値を割り当てることによって行なわれるとする．

**) $P_i$ が制約条件を満たさない時には，$P_i$の目的関数の値および$\ell b(P_i)$の値はともに∞と定義する．

つぎに，$h$ を $V$ 上の実数値関数とする．ただし，$h$ は次の条件を満足すると仮定する．

$$h(Pi) \neq h(Pj) \quad \text{if} \quad i \neq j \quad \text{for} \quad Pi, Pj \in V \tag{6.4}$$

関数 $h$ は，通常ヒューリスティック関数と呼ばれるものである．さらに，関数 $h$ が次の条件を満たすとき $h$ は関数 $\ell b$ と矛盾しない（consistent）と言われる．

$$h(Pi) < h(Pi) \quad \text{if} \quad \ell b(Pi) < \ell b(Pj) \tag{6.5}$$

以下では，関数 $h$ は関数 $\ell b$ と矛盾しないとする．

$I$ を $V$ の部分集合とするとき，$\mathrm{L}h(I)$ を，関数 $h$ に関する $I$ の最小元とする．すなわち，次式が成り立つ．

$$h(\mathrm{L}h(I)) = \min\{h(Pi) \mid Pi \in I\} \tag{6.6}$$

逐次的な Branch-and-Bound アルゴリズムで，$I$ をアルゴリズムの実行途中での，未解決な部分問題の集合とする．$I$ 中から，アルゴリズムで次に処理すべき部分問題を選択する関数をサーチ関数と呼ぶことにする．サーチ関数 $s_{\mathrm{D}}$ を次のように定義する．

〔定義 6.1〕（サーチ関数 $s_{\mathrm{D}}$ ）

$I \subseteq V$ とする．$I$ 中で，$level$ の値が最大である要素の集合を $Il\,max$ とする．すなわち，

$$Il\,max \triangleq \{Pi \mid level(Pi) = \max\{level(Pj) \mid Pj \in I\}\} \tag{6.7}$$

このとき，関数 $s_{\mathrm{D}}(I)$ を次式で与える．

$$s_{\mathrm{D}}(I) \triangleq \mathrm{L}h(Il\,max) \tag{6.8}$$

サーチ関数として $s_{\mathrm{D}}$ を用いる Branch-and-Bound アルゴリズムは，Depth-First ストラテジに従うと言われる．

### 6.3.2 アルゴリズムの並列化の方針

Branch-and-Bound アルゴリズムの実行途中での，未解決な部分問題の集合を $I$ で表わす．$I$ 中の，相異なる，どの 2 つの要素の対も先祖－子孫の関係にはない．*

---

*）このような要素の対を，独立な要素の対と呼ぶ．また，集合中の相異なる要素の対が全て独立であるとき，その集合を独立な集合と呼ぶことにする．上記の $I$ が独立な集合であることは明らかである．

よって，$I$ の各要素は，独立に処理が可能である．そこで，前節で述べたマルチプロセッサ・システムの各処理装置に，同一の処理手続きを与え，$I$ の要素を1つずつ割り当てて処理させることにする．

この時，マルチプロセッサの各処理装置の間で共用される情報は，次の3種類である：

(i) 未解決な部分問題の集合 $I$

(ii) 最適解の候補に対する目的関数の値

(iii) 部分問題を処理中の処理装置の台数

これらの情報のうち，(iii)は，後に述べる並列形Branch-and-Boundアルゴリズムで，各処理装置が停止すべきか否かを判定するために用いられる．

また，上記の3種類の情報は，各処理装置の間で排他的に参照されなければならない．この排他制御は，たとえば，セマフォア[2]を用いて実現される．

### 6.3.3 並列形Branch-and-Boundアルゴリズム

上で述べた方針に従って並列化された，Branch-and-Boundアルゴリズムを，以下のように記述する．

アルゴリズム中で，各処理装置によって共通に参照される変数は次の4つである．

$I$ ：未解決な部分問題の集合，

Xopt ：最適解の候補の集合，

Copt ：Xoptに対する目的関数の値，

act ：部分問題の処理を実行中の処理装置の台数

これらの変数は，各処理装置の間で，排他的に参照されると仮定し，以下のアルゴリズム中では，排他制御の手順は示さない．*

アルゴリズムPDFA(Parallelized Depth-First Algorithm)を以下のように与える．

〔アルゴリズム6.1〕(PDFA)

**入力**：最小化問題 $P_0$ ；

**出力**：$P_0$ の全ての最適解の集合Xopt，およびXoptに対する目的関数の値Copt ；

ただし，$P_0$ が制約条件を満たす可能解を持たなければ，Xopt $=\emptyset$，Copt $=\infty$が出力される．

---

*) 排他制御の手順は，文献(11)に示す．

方法：

Phase 1：〔初期設定〕

$I \leftarrow \{P_0\}$, Xopt $\leftarrow \emptyset$, Copt $\leftarrow \infty$, act $\leftarrow 0$ とする．Phase 2へ；

Phase 2：

p台の処理装置で並列に，それぞれ，次の手続き6.1を実行する；

全ての処理装置が停止したとき，アルゴリズムは終了する．

〔手続き6.1〕

方法：

Step 1：〔処理装置が停止すべきか否かの判定〕

(i) $I = \emptyset$ かつ act $= 0$ ならば，停止する；

(ii) $I = \emptyset$ かつ act $> 0$ ならば，Step 1へ；

(iii) $I \neq \emptyset$ のとき，Step 2へ；

Step 2：〔部分問題の選択〕

act $\leftarrow$ act+1, $Pi \leftarrow s_D(I)$,

$I \leftarrow I - \{Pi\}$ として，Step 3へ；

Step 3：〔部分問題の評価〕

(i) $Pi \in L$ ならば，Step 6へ；

(ii) $Pi \notin L$ ならば，Step 4へ；

Step 4：〔目的関数の下界を用いた部分問題の除去〕

(i) $\ell b(Pi) >$ Copt ならば，Step 7へ；

(ii) $\ell b(Pi) \leq$ Copt ならば，Step 5へ；

Step 5：〔部分問題の分割〕

$Pi$ を分割して得られる $k$ 個の部分問題を $I$ に加える；Step 7へ；

Step 6：〔最適解の候補の更新〕

$$\text{Xopt} \leftarrow \begin{cases} \text{Xopt} & \text{if } cost(Pi) > \text{Copt} \\ \text{Xopt} \cup \{Pi\} & \text{if } cost(Pi) = \text{Copt} \\ \{Pi\} & \text{if } cost(Pi) < \text{Copt} \end{cases}$$

Copt $\leftarrow$ min(Copt, $cost(Pi)$)

としてStep 7へ；

Step 7 :〔部分問題 $P_i$ の処理の終了〕

act ← act − 1として，Step 1へ．

上記のアルゴリズム6.1は，全ての最適解を出力する．最適解の1つだけを求めるアルゴリズムは，手続き6.1のStep 4およびStep 6を修正することにより得られる．

また，このアルゴリズムで，処理装置の台数pを明示する必要があるときには，p−PDFAように記す．PDFAの種々の性質については，次節で述べる．

### 6.3.4 例題とその解法

図6.2に示される2値変数問題を考える．同図で，各節点の円内の数字は，対応する部分問題の番号を表わす．また，円の右側には，その部分問題の目的関数の下界（$\ell b$）の値を示す．さらに，木の葉節点の下には，対応する部分問題の目的関数（$cost$）の値を示す．

図6.2の問題を，(a) 1−PDFA，(b) 2−PDFA，および，(c) Best−Boundサーチを用いる逐次形Branch−and−Boundアルゴリズムで，それぞれ解いた時に，アルゴリズムで処理される部分問題とその順序を図6.3に示す．ここに，各部分問題の処理時間は，部分問題および処理装置によらず一定であると仮定する．

(a)，(b)，(c)の各アルゴリズムの計算時間をそれぞれ$T_1$，$T_2$，$T_b$とすると，図6.2の問題に対しては，

$$T_b < 2 \times T_2 < T_1 \qquad (6.9)$$

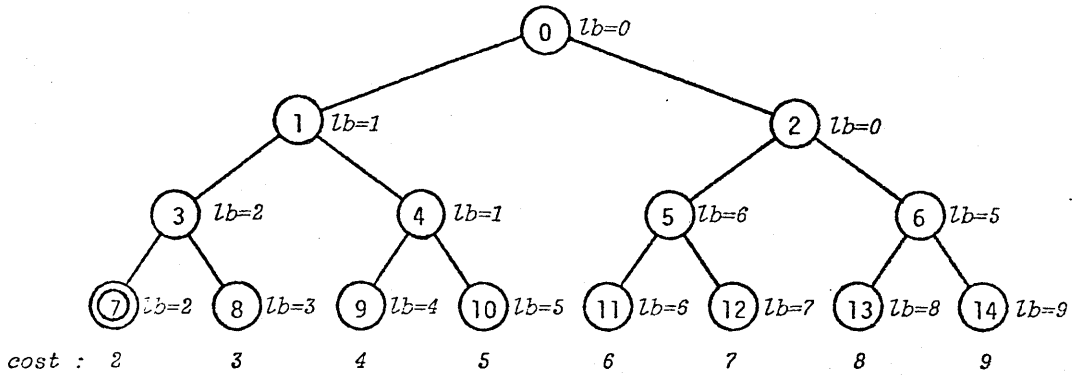



図6.2 木で表現された例題

| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PU #1 | 0 | 2 | 6 | 13 | 5 | 11 | 1 | 4 | 9 | 3 | 7 |

(a) 1-PDFA

| Time | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| PU #1 | 0 | 2 | 4 | 7 |
| PU #2 | - | 1 | 3 | 8 |

(b) 2-PDFA

| Time | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| PU #1 | 0 | 2 | 1 | 4 | 3 | 7 |

(c) Best-bound search

図6.3　1－PDFA，2－PDFAおよびBest－Boundサーチを用いる逐次形アルゴリズムによつて処理される部分問題とその処理順序

であることが知られる．このように，p－PDFAの計算時間は1－PDFAの計算時間の1/p以下になることが予想される．

## 6.4　PDFAの性質

この節では，前節で述べたPDFAの実現のために必要な記憶容量，および，PDFAの計算時間について考察する．

### 6.4.1　PDFAの実現に必要な記憶容量

PDFAの必要とする記憶容量に関して，次の定理が成り立つ．

〔定理6.1〕　p－PDFAの実行途中の任意の時点における，未解決な部分問題の集合$I$の個数は，次式を満足する．

$$|I| \leq p \times \{ (k-1) \times n + 1 \} \tag{6.10}$$

〔証明〕　証明は，付録5(1)に示す．

一方，逐次形のBranch-and-Boundアルゴリズムで，Depth-Firstサーチを用いた場合の$|I|$の値に関しては，

$$|I| \leq (k-1) \times n + 1 \tag{6.11}$$

が成り立つから[7]，p−PDFAの｜$I$｜の値の上限は，Depth-First サーチを用いる逐次形のBranch-and-Boundアルゴリズムの場合のp倍であることが知られる．

Depth-First サーチ以外のサーチ・ストラテジ，たとえば，Breadth-First サーチ，Best-Bound サーチ，および，Heuristic サーチを用いる逐次形のBranch-and-Boundアルゴリズムでは，｜$I$｜の値の上限は，いずれも，$k^n$ である．よって，これらのサーチ・ストラテジを用いる並列形Branch-and-Boundアルゴリズムでも，｜$I$｜の上限はやはり$k^n$となって，必要な記憶容量に関してはPDFAに劣ることは明らかである．

### 6.4.2 PDFAの計算時間

PDFAの計算時間の解析を行なうために，次の仮定を置く．

〔仮定 6.4〕　一つの部分問題の処理に要する計算時間，すなわち，PDFAの手続き6.1のStep 2からStep 6までの計算時間は，部分問題および処理装置によらず，一定である．

この仮定のもとでは，p−PDFAの計算時間T(p)は，同時に処理されている部分問題の組を1単位として，全ての処理装置が停止するまでに，処理される部分問題の組の個数によって評価される．* p−PDFAで処理される部分問題の総数を，以下ではK(p)で表わし，これをp−PDFAの計算量と呼ぶ．K(p)が処理装置の台数pよりも十分に大きければ，任意のp(p>1)について，次の関係が成り立つと考えてよい．

$$T(p) \approx K(p) / p \qquad (6.12)$$

ここで，もし，適当なpについて，

$$K(p) < K(1) \qquad (6.13)$$

が成り立てば，T(1)≈K(1)および式(6.12)から，そのpについて，

$$T(p) < T(1) / p \qquad (6.14)$$

が成り立つ．すなわち，適当なpの値に対して，p−PDFAの計算時間は，1−PDFAすなわち

---

*) ここでは，アルゴリズム6.1のPhase1，および，手続き6.1のStep1，Step 7の計算時間は，手続き6.1のStep 2〜Step 6の計算時間と比較して，十分に小さいと仮定する．これらの計算時間が無視できないほど大きい場合についての対策に関しては，あとがきで述べる．

Depth-First サーチを用いる逐次形のBranch-and-Boundアルゴリズムの計算時間の1/p以下になる．

式（6.13）が成り立ちうることは，次の理由による．

第一に，p-PDFAによって最初にみつけられる可能解による目的関数の値は，pの値が増大するにつれ，最適解による目的関数の値に漸近してゆく．*

第二に，最適解による目的関数の値により近い値を与える可能解が早くみつけられるほど，手続き6.1のStep 4で取り除かれる部分問題の個数が増大する．しかも，この機会は，pの値の増大とともに増加すると考えられる．

逆に，pの値が増大しすぎると，アルゴリズムは，Breadth-First サーチを並列化した場合に近づくから，最初の可能解がみつけられるまでに処理される部分問題の個数は，大きなpの値について最大 $p \times n + 1$ となり，かえって増大する．

以上の理由により，K(p)の値を最小にし，しかもK(p)＜K(1)をみたすようなpが存在すると考えられる．

以下では，並列化によるアルゴリズムの計算量K(p)の減少を加速効果と呼ぶことにする．** p台の処理装置を用いたときの加速効果の大きさA(p)を次式で定義する．

$$A(p) \triangleq K(1)/K(p) \qquad (6.15)$$

また，A(p)の最大値をA* で表わす．すなわち，

$$A^* \triangleq \max_p A(p) \qquad (6.16)$$

## 6.5 シミュレーションとその結果

この節では，6.4.2で述べた仮定6.4のもとで，PDFAの性質をシミュレーションによって調べ，6.4.2で述べた，PDFAの計算時間についての考察が妥当であることを示す．

実験は，計算機上での乱数を用いて発生された，2種類の問題の組がPDFAによって解かれる過程を，シミュレートするものである．一方の問題の組は，文献(7)で提案されている最小化問題のモデルに準じて発生された．また，他の一つの問題の組は，最小被覆問題である．

以下では，これらの問題の組による実験を，それぞれ，実験1，実験2と呼ぶ．

---

*）　p-PDFAではじめにみつけられる可能解による目的関数の値をZpとするとき，Z(p+1)＜Z(p)となるための必要十分条件は，付録5(2)に示す．

**）　諺「三人寄れば文殊の智慧」にちなみ，「文殊効果」とも呼ばれる．

### 6.5.1 実　験　1

実験1では，具体的な最小化問題を指定しないで，問題に対応する木の性質を与え，これにもとづいて，問題をPDFAで解く過程をシミュレートする．

実験の目的は，次の2つである．

(i) 加速効果が現実に存在することを確かめる．

(ii) 加速効果の大きさが，解かれる問題の規模および用いられる評価関数の正確さによって，どのように変化するかを知る．

#### 6.5.1.1 問題の発生方法

次のようにして，木$B=(V, E)$の各節点に対して，$cost$ および $\ell b$ の値を割り当てることにより，2値$n$変数の最小化問題を発生させた．

(1) $cost$ の割り当て：

初めに，$P_0$ のコストを1.0とし，一般に，ある節点$Pi$の2つの子節点のコストをそれぞれ，$cost(Pi)$および$cost(Pi)+\alpha$とする．ただし，$\alpha$は，$\lambda$を与えられたパラメータとして，その確率密度関数$f(\alpha)$が次式で与えられる確率変数である．

$$f(\alpha) \triangleq \begin{cases} \lambda e^{-\lambda\alpha} & \text{if } \alpha > 0 \\ 0 & \text{if } \alpha < 0 \end{cases} \qquad (6.17)$$

(2) $\ell b$ の割り当て：

初めに，$P_0$の目的関数の下界を0.0とする．さらに，一般に，ある節点$Pi$の2つの子節点を$Pi_0$, $Pi_1$とするとき，$Pi_0$，$Pi_1$の$\ell b$の値を以下のようにして定める．

(i) $level(Pi)=n-1$の時，

$$\ell b(Pij) \triangleq cost(Pij) \text{ for } j=0,1 \qquad (6.18)$$

(ii) $level(Pi)<n-1$の時，$\ell b(Pij)$の値を区間：

$$[\ \ell b(Pi),\ \ell b(Pi)+2\Delta ij\ ] \qquad (6.19)$$

での一様乱数の実現値によって定める．ここに，$\Delta ij$は次式で与えられる．

$$\Delta ij \triangleq \frac{cost(Pij)-\ell b(Pi)}{n-level(Pi)} \qquad (6.20)$$

(3) $h$ の割り当て：

ヒューリスティック関数$h$の値の与え方は，次の2通りを試みた．

タイプ1：$h(Pi) \triangleq \ell b(Pi)$　　(6.21)

タイプ2：$h(Pi)$を，$\sigma$を与えられたパラメータとして，次式で与えられる正規乱数の実現値とする．

$$N(cost(Pi), \{(n-level(Pi)) \times \sigma\}^2) \qquad (6.22)$$

上記のようにして発生された問題に対しては，$Pi$の子節点$Pi_0$，$Pi_1$について次の性質が成り立つと考えられる．

(i) $\lambda$の値が増すにつれて，$cost(Pi_1)$と$cost(Pi_0)$の値の差は小さくなる．一方，$cost(P_0) - \ell b(P_0)$は1に正規化されているので，$\lambda$の値が増すにつれて，$\ell b(Pi) \leq cost(P_0)$をみたす部分問題，すなわち，アルゴリズム6.1で必ず処理される部分問題の個数は増す．

(ii) (i)と同様の理由により，$\lambda$の値が増すにつれて，$\ell b(Pi)$は$cost(Pi)$の，より厳密な下界を与える．よって，$\lambda$の値が増すにつれて，$\ell b$のヒューリスティック関数としての正確さも増す．

(iii) タイプ2のヒューリスティック関数を用いた場合，$\sigma$の値が小さいほど，$h(Pi)$の値は$cost(Pi)$の値に近づく傾向がある．すなわち，$\sigma$の値が小さいほど，$h$は$cost$に関するより正しい情報を与えると考えられる．

### 6.5.1.2 実験方法および結果

上記のようにして発生された問題を，仮定6.4のもとで，PDFAによって解く過程をシミュレートし，PDFAの計算量を測定した．以下では，p-PDFAの計算量の20問題についての平均値をK(p)で表わす．

#### (1) タイプ1のhを用いた時の計算量

$h$をタイプ1のヒューリスティック関数とした時の，処理装置の台数pと計算量K(p)との関係を図6.4および図6.5に示す．図6.4では，$\lambda$を0.2に固定し，変数の個数$n$の値は10から60まで変化させた．また，図6.5では，$n$の値を30に固定し，$\lambda$を0.1から0.5まで変化させた

なお，両図および以下の図では，次式で与えられるK(p)の下限を2点鎖線（－・・－）で表わす．*

---

*）ここでは、アルゴリズムの計算量K(p)は，手続き6.1のstep1で，$I = \emptyset$かつact＞0であるために“待ち”状態におかれている処理装置の計算量をも1単位として数えている．

$$\min(K^*,\ p\times(n+1)) \tag{6.23}$$

ここに，$K^*$ は，次式で与えられる部分問題の要素の個数である．

$$K^* \triangleq \left|\{ P_i \mid P_i \in v,\ \ell b(P_i) \leq \mathrm{cost}(P_0) \}\right| \tag{6.24}$$

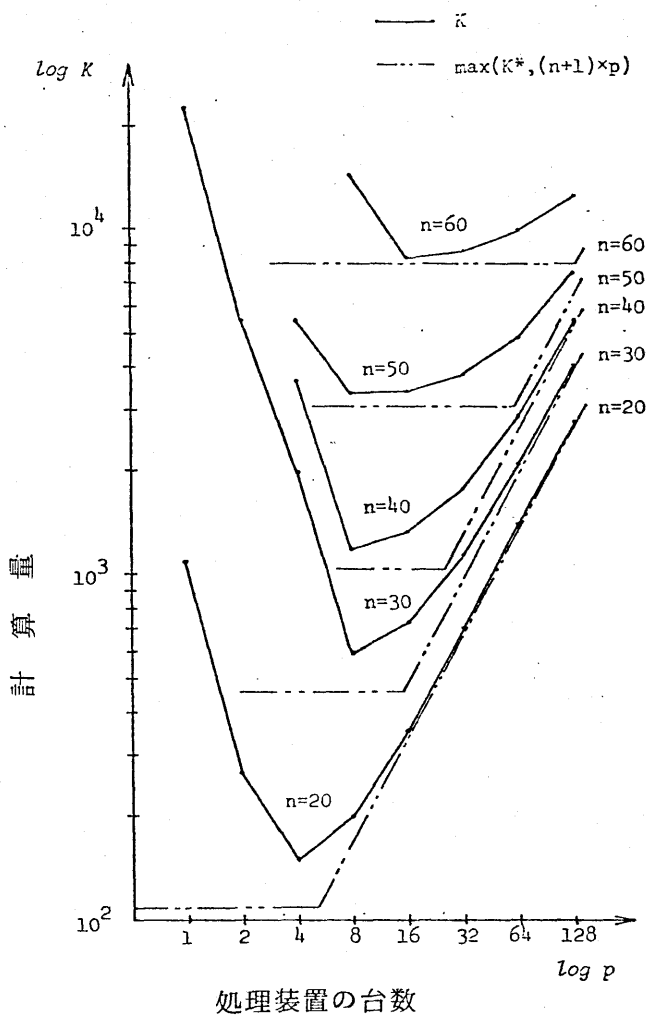



図6.4　PDFAの計算量（h=$lb$，λ=0.2）

$n$　：変数の個数
K　：計算量
max（$K^*$，（$n$+1）×p）：計算量の下限

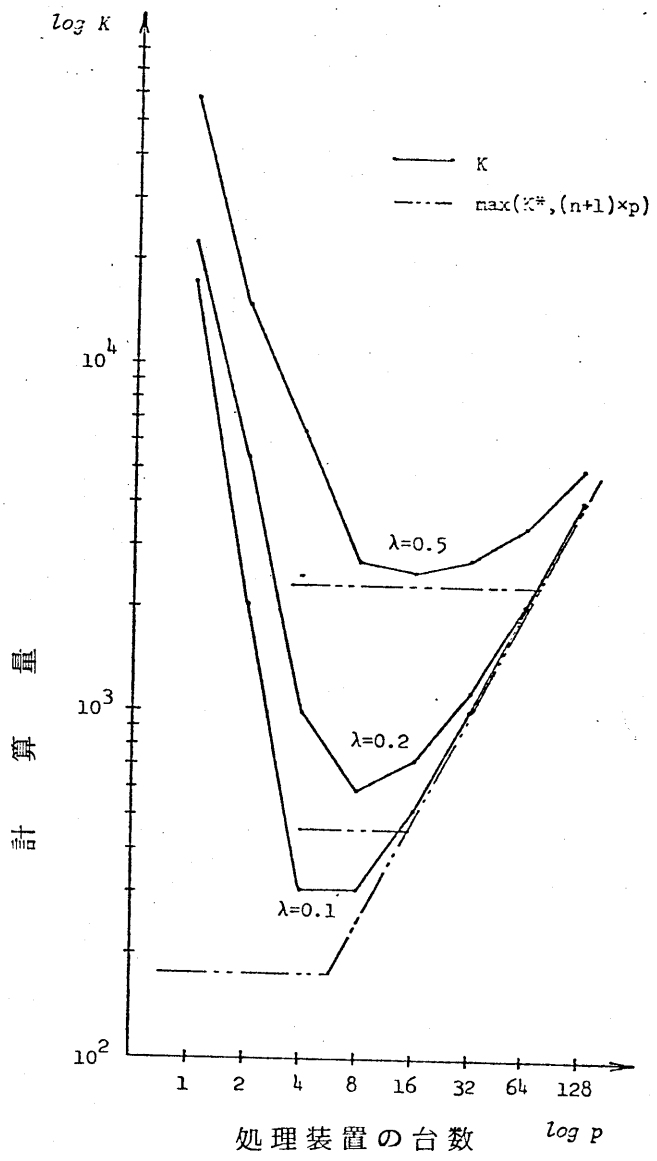



図6.5　ＰＤＦＡの計算量（h＝$lb$, n＝30）

λ：問題の規模を表わすパラメータ
K：計算量
max(K*, ($n$+1)×p)：計算量の下限

また，式（6.23）で$n+1$の項は，木$B$の根$P_0$と，木の最もレベルの大きい葉節点を結ぶパス上の節点の個数を表わしている．

両図より，適当なpの値（たとえば，図6.5で，$n=20$のとき，$2 \leq p \leq 32$）に対して，K(p)＜K(1)であること，すなわち，加速効果が現われていることが確認される．

また，図6.4より，$n$の値が大きくなるほど，加速効果の最大値$A^*$の値が増すことが知られる．

(2) タイプ2のhを用いた時の計算量

タイプ2の$h$を用いた時のK(p)のpに対する変化のようすを図6.6に示す．この図では，$n$を30に固定し，$\sigma$を0.01から0.2まで変化させた．

同図より，$\sigma$の値が大きいほど，すなわち，ヒューリスティック関数の正確さが減ずるほど，加速効果が顕著に現われることが知られる．

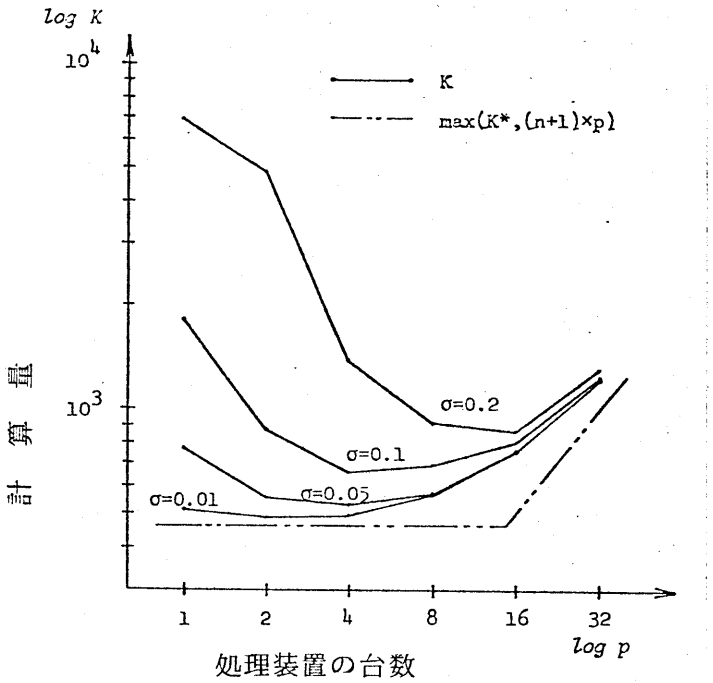



図6.6　PDFAの計算量（n＝30,λ＝0.2）

$\sigma$：ヒューリスティック関数の正確さを表わすパラメータ
K：計算量
max：(K*, (n+1)×p)：計算量の下限

### 6.5.2 実験 2

具体的に与えられた最小被覆問題をPDFAで解く過程をシミュレートし，現実の問題においても加速効果が現われることを示す．ただし，ここでは，最適解のうちの1つだけを求めることにする．

#### 6.5.2.1 問題の発生方法

変数の個数，制約条件の個数をともに30とし，一つの条件に含まれる変数の個数の分布範囲$\delta$を変化させる．すなわち，$\delta$＝〔a，b〕ならば，一条件当りの変数の個数が区間〔a，b〕

で一様に分布するように変数の個数を定めた．

### 6.5.2.2　実験方法および結果

上記の方法で発生された最小被覆問題を，仮定6.4のもとでPDFAによって解く過程をシミュレートし，PDFAの計算量を測定した．ただし，目的関数の下限は，値"1"を割り当てられた変数の個数で与えられる．また，ヒューリスティック関数は，目的関数の下界の値をそのまま用いる．

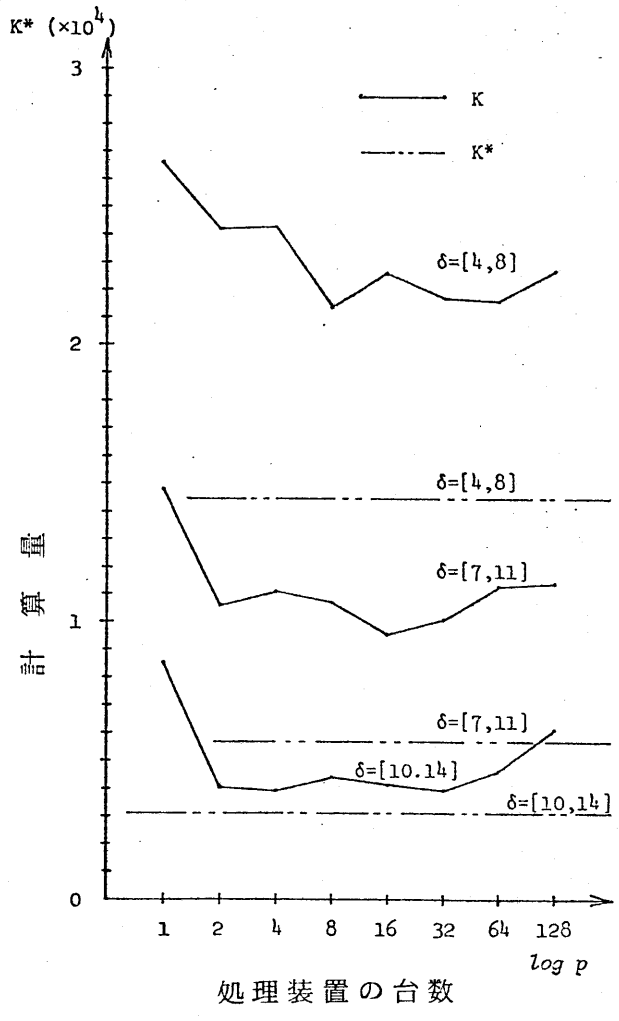



図6.7　PDFAの計算量（n=30,λ=0.2）

δ：一条件当りの変数の個数の分布範囲
K：計算量
K*: 計算量の下限

図6.7に，$\delta=[4,\ 8]$，$[7,\ 11]$，$[10,\ 14]$の3組,各20問題についてのK(p)の値を示す．同図より，実際に生成された問題に対しても加速効果が現われることが知られる．

## 6.6 むすび

本章では，マルチプロセッサ・システムの上で実現される，並列形Branch-and-Boundアルゴリズム，PDFAを提案し，PDFAを実現するために必要な記憶容量の評価，および，その計算時間についての予備考察を行なった．さらに，これにもとづいて，シミュレーションによるアルゴリズムの効率の評価を行なった．その結果，処理装置の台数(p)を適当にえらべば，アルゴリズムの計算時間は，処理装置が1台の場合の1／p以下になることが明らかにされた．また，並列化によるアルゴリズムの計算時間のこのような短縮を加速効果と名づけた．

この加速効果は，一般に，並列形Branch-and-Boundアルゴリズムで，Depth-Firstサーチまたは，一般のHeuristicサーチを用いた場合には現われるが，Best-Boundサーチおよび Breadth-First サーチを用いた場合には期待できない．

加速効果の存在は，本章ではシミュレーションのみにより確かめられているが，Fullerらによって実現されたマルチプロセッサ・システム(Cm*)上で，整数計画問題をBranch-and-Bound法で解くことによって，同様の効果が存在することが確認されている．[6]

本章で扱かったマルチプロセッサ・システムでは，各処理装置は，共有記憶装置を通してゆるやかに結合されている．ここで提案された並列形Branch-and-Boundアルゴリズムでは，部分問題の処理手続き(手続き6.1)を全処理装置にわたって一様に与えることができるので，プログラム分割をさけることができる．しかし，共有記憶装置上の，未解決な部分問題などの情報の参照の競合の問題は避け得ないものである．

Fullerらの実験によれば，並列化の効果は，整数計画問題に対して特に顕著に現われている．その第一の理由は，プログラム分割の容易さであると考えられる.記憶装置に対する競合がどのように対処されているかについての具体的な内容は不明であるが，次のような対策も有効であると考えられる．

(i) 計算が進行し，未解決な部分問題の集合$I$の要素の個数がある値をこえたら，$I$をr個の部分集合$I_1$，$I_2$，…，$I_r$に分割する．同じく，p台の処理装置もp／r台づつのr群に分け，第$i$処理装置群に対して$I_i$を当り当てる．以降，各$I_i$は共有記憶装置上の独立した領域に格納され，第$i$処理装置群によってのみ処理される．また，各$I_i$の要素の個数の均衡が保た

れるように注意しながら計算を行なう。

(ii) 各処理装置で処理される部分問題（$Pi$）のレベルがある値をこえたら，$Pi$ の子孫は $I$ に加えず，$Pi$ を処理した処理装置の専用記憶装置中に置き，以後，この処理装置で $Pi$ から生成される全ての部分問題の処理を行なう。

したがって，本研究の今後の課題としては，

(i) 共有記憶装置をめぐる競合の程度の解析と，上記の対策の有効さの検討，

(ii) 処理装置の同期に関する仮定 6.4 をゆるめ，各部分問題の処理に要する計算時間が，部分問題ごとに異なる場合についての，システムの動作の解析，

などが考えられる。

# 第7章　必要な記憶容量を自由に制御できる Branch-and-Bound法のサーチ・ストラテジ

## 7.1 まえがき

Branch-and-Bound法は，DP（Dynamic Programming）とともに，組合せ論的問題の一般的解法の一つとして，多くの分野に適用されてきた．

効率的なBranch-and-Boundアルゴリズムを設計するためには，次の3つの問題点に対する方策を明らかにする必要がある．

(i) 与えられた部分問題をどのように部分問題に分割するか．

(ii) 部分問題に対する目的関数の下界をどのようにして求めるか．

(iii) 与えられた問題を分割して得られた，未解決な部分問題の中から，次に処理すべき部分問題をどのようにして選択するか．

これらの3つの問題点のうち，(i)および(ii)については，第2章および第4章で詳しく述べた．本章では，(iii)の問題点について述べる．

Branch-and-Boundアルゴリズムで，与えられた問題を分割して得られた未解決な部分問題の中から，次に処理すべき部分問題を選択する手続きは，サーチ関数とよばれる．Branch-and-Boundアルゴリズムで従来用いられてきた代表的なサーチ関数は，次の4つである．

(i) Depth-Firstサーチ関数

(ii) Best-Boundサーチ関数

(iii) Heuriticサーチ関数

(iv) Breadth-Firstサーチ関数

$k$－値をとる$n$個の変数によって記述される組合せ最適化問題を，Branch-and-Boundアルゴリズムで解く場合，アルゴリズム中で保持される未解決な部分問題の個数の上限は，用いられるサーチ関数の種類によって，次のように変化する．まず，Depth-Firstサーチ関数を用いた場合には，上記の上限は$(k-1)\times n+1$である．また，Best-Bound，Henristic，Breadth-Firstの各サーチ関数を用いた場合には，上記の上限は$k^n$である．

よって，後者の3つのサーチ関数を用いた場合には，計算機の記憶容量の不足のために，大規模な問題を解くことが困難である場合が多い．一方，Depth-Firstサーチ関数を用いた場合には，アルゴリズムの実現に必要な記憶容量は最小となるが，強力な評価関数が得られない場

合には，アルゴリズム中で処理される部分問題の個数が非常に多くなり，そのためにぼう大な計算時間が必要となる可能性がある．

そこで，必要な記憶容量が制御でき，かつ，アルゴリズム中で処理される部分問題の個数が，Depth-First サーチ関数を用いた場合よりも削減されるサーチ関数を開発することが必要となる．茨木によって提案されたDepth-$m$サーチ関数[2,3]を用いた場合，アルゴリズムの実現に必要な記憶容量は，$m$をパラメータとして，$O(n^m)$である．しかも，この場合，アルゴリズム中で処理される部分問題の個数は，その下界へ漸近するという性質があるので，上記の目的は，ある程度みたされる．しかし，与えられた記憶容量を効率良く使用できるように$m$の値を決定するためには，解かれる問題に対する経験的な知識が必要である[2]ので，必要な記憶容量を完全に制御することは困難である．

本章では，必要な記憶容量を自由に制御できる，2つのサーチ関数を提案し，これらの関数の性質を明らかにする．さらに，これらの関数を用いた場合の，アルゴリズムの効率を調べるための実験の結果について述べる．

以下，初めに，7.2では，以下の議論のために必要ないくつかの定義を与え，また，記法を示す．次に，7.3では，Branch-and-Boundアルゴリズムを記述する．次に，7.4では，従来用いられてきたサーチ関数のいくつかの性質とそれらの問題点について述べる．次に，7.5では，必要な記憶容量を自由に制御できる2つのサーチ関数を提案し，さらに，7.6では，これらのサーチ関数の性質を明らかにする．最後に，7.7では，これらのサーチ関数を用いた場合の，アルゴリズム中で処理される部分問題の個数，および，アルゴリズムの計算時間を調べるために行なった実験の結果について述べる．

## 7.2 諸定義および記法

この節では，以下で述べる手続きおよびアルゴリズムのための，定義および記法を与える．

以下では，$k$値をとる$n$個の変数によって記述される最小化問題$P_0$を考える．まず最小化問題$P_0$を解くアルゴリズム$A$を，次の6項組で定義する．[4]

$$A \triangleq \langle B,\ L,\ z,\ b,\ h,\ s \rangle \qquad (7.1)$$

ここに，$A$の各構成要素は次のようである．

$B=(V,E)$を，問題$P_0$が分割される過程を表わす，節点$P_0$を根とする深さ$n$の$k$進木とする．ここに，$V$は木$B$の節点の集合を，また$E$は枝の集合とする．

$V$の各要素$P_i$を，問題$P_0$を何回か分割して得られる部分問題$P_i$に対応づける．また，$P_i$，

$P_j$ を $V$ の要素とする時，$E$ の要素 $(P_i, P_j)$ は，部分問題 $P_j$ が部分問題 $P_i$ を直接分割して得られる部分問題の一つであることを現わす.*

$L$ を，木 $B$ の葉節点の集合とする．$L$ は，値が割り当てられていない変数を陽に含まない部分問題，および制約条件を満たさない部分問題からなる．

部分問題 $P_i \in V$ のレベルを，$P_0$ から $P_i$ に至る枝の個数によって定義する．また，その値を $\ell(P_i)$ で現わす．

$z(P_i)$ を，部分問題 $P_i$ の最適解の値を与える関数とする．すなわち，$z(P_i)$ を次式で与える．

$$z(P_i) \triangleq \begin{cases} P_i \text{ の目的関数の値}^{**} \ ; \ \text{if } P_i \in L \\ \min \{ z(P_j) \mid (P_i, P_j) \in E \} \ ; \ \text{if } P_i \in L \end{cases} \qquad (7.2)$$

$b(P_i)$ を，$z(P_i)$ の下界を与える関数とする.*

ただし，関数 $\ell$ は，次式を満足すると仮定する．

$$b(P_i) = z(P_i) \qquad \text{if} \quad P_i \in L \qquad (7.3)$$

$$b(P_i) \leq b(P_j) \qquad \text{for} \quad (P_i, P_j) \in E \qquad (7.4)$$

$h$ を，$V$ から実数の上への写像とする．$h$ は，ヒューリステック関数と呼ばれる．以下では，$h$ に関して次式が成り立つと仮定する．

$$h(P_i) \neq h(P_j) \qquad \text{if} \quad i \neq j \qquad (7.5)$$

$s$ を，未解決な部分問題の集合の中から，アルゴリズム $A$ で次に処理すべき部分問題を選択する関数とする．以下では，$s$ をサーチ関数と呼ぶ．

Branch-and-Bound 法で従来用いられてきた代表的なサーチ関数は，次の 4 つである．

(i) Depth-First サーチ関数 : $s_D$

(ii) Best-Bound サーチ関数 : $s_{BB}$

(iii) Heuritic サーチ関数 : $s_H$

(iv) Breadth-First サーチ関数 : $s_{BF}$

以下では，$I \subseteq V$ を，アルゴリズム $A$ の実行途中のある時点での，未解決な部分問題の集合とする．また，$Lh(I)$ を，集合 $I$ 中で，関数 $h$ の値を最小にする要素を値とする関数とする．

このとき，上記の 4 つのサーチ関数は，それぞれ，以下のように定義される．

---

*) 部分問題の分割は，部分問題中に含まれる変数に値を割り当てることによって行なわれるとする．

**) 部分問題 $P_i$ が制約条件を満たさない時には，$z(P_i) = \infty$ ，$b(P_i) = \infty$ と定義する．

) サーチ関数 $s_D$ ：

$$D \triangleq \{ P_i \mid P_i \in I, \ \ell(P_i) = \max \{ \ell(P_j) \mid P_j \in I \} \} \tag{7.6}$$

義するとき，$s_D(I)$ を次式で与える．

$$s_D(I) \triangleq Lh(\{P_i \mid P_i \in D, \ b(P_i) = \min \{ b(P_i) \mid P_j \in D \}\}) \tag{7.7}$$

) サーチ関数 $s_{BB}$：

サーチ関数 $s_{BB}$ を次式で定義する．

$$s_{BB}(I) \triangleq Lh(\{P_i \mid P_i \in I, \ b(P_i) = \min\{b(P_j) \mid P_j \in I\}\}) \tag{7.8}$$

サーチ関数 $s_H$ ：

サーチ関数 $s_H$ を次式で定義する．

$$s_H(I) \triangleq \mathrm{Lh}(I) \tag{7.9}$$

サーチ関数 $s_{BF}$ ：

$$D' \triangleq \{ P_i \mid P_i \in I, \ \ell(P_i) = \min \{\ell(P_j) \mid P_j \in I \}\} \tag{7.10}$$

義するとき，$s_{BF}(I)$ を次式で与える．

$$s_{BF}(I) \triangleq Lh(\{P_i \mid P_i \in D', \ b(P_i) = \min \{ b(P_j) \mid P_j \in D' \}\}) \tag{7.11}$$

Branch－and－Bound アルゴリズム

の節では，前節の定義および記法を用いて，Branch-and-Bound アルゴリズムを一般的にする．以下では，与えられた最小化問題 $P_0$ の全ての最適解を求めるアルゴリズム $A_a$ ，び，最適解の一つを求めるアルゴリズム $A_s$ を与える．

アルゴリズム 7.1 〕（アルゴリズム $A_a$ ）

：最小化問題 $P_0$

：$P_0$ の全ての最適解 Xopt ；

Xopt に対する目的関数の値 Zopt ；

ただし，$P$ が制約条件を満足する可能解を持たない時には，Xopt $= \emptyset$，Zopt $= \infty$ が

出力される；

記法：$I \subseteq V$をアルゴリズムの実行途中で生成された未解決な部分問題の集合とする；

方法：

Step 1：〔初期設定〕

$I \leftarrow \{P_0\}$，Xopt $\leftarrow \emptyset$，Zopt $\leftarrow \infty$とする；

Step 2へ；

Step 2：〔部分問題の選択〕

(i) $I = \emptyset$ならば，Step 7へ；

(ii) $I \neq \emptyset$ならば，$P_i \leftarrow s(I)$，$I \leftarrow I - \{P_i\}$として，Step 3へ；

Step 3：〔部分問題の評価〕

(i) $P_i \in L$ならば，Step 6へ；

(ii) $P_i \notin L$ならば，Step 4へ；

Step 4：〔目的関数の下界による枝刈り〕

(i) $b(P_i) >$ Zopt ならば，Step 2へ；

(ii) $b(P_i) \leq$ Zopt ならば，Step 5へ；

Step 5：〔部分問題の分割〕

$P_i$を分割して得られる$k$個の部分問題を，$P_{i_1}, P_{i_2}, \cdots, P_{i_k}$とする；

$I \leftarrow I \cup \{P_{i_1}, P_{i_2}, \cdots, P_{i_k}\}$として，Step 2へ；

Step 6：〔最適解の候補の更新〕

$$I \leftarrow \begin{cases} I & ; \text{ if } z(P_i) > \text{Zopt} \\ I \cup \{P_i\} & ; \text{ if } z(P_i) = \text{Zopt} \\ \{P_i\} & ; \text{ if } z(P_i) < \text{Zopt} \end{cases}$$

$$\text{Zopt} \leftarrow \min(z(P_i), \text{Zopt})$$

として，Step 2へ；

Step 7：〔アルゴリズム終了〕

停止．

$P$の最適解の一つを求めるアルゴリズムは，次のように与えられる．

〔アルゴリズム 7.2〕（アルゴリズム$A_S$）

入力：最小化問題$P_0$；

出力: $P_0$ の最適解の一つ　Xopt;

Xopt に対する目的関数の値 Zopt;

ただし, $P_0$ が制約条件を満足する可能解を持たない時には, Xopt $=\lambda$ (null),

Zopt $=\infty$ が出力される;

記法: $I \subseteq V$ をアルゴリズムの実行途中で生成された未解決な部分問題の集合とする;

方法:

Step 1 :〔初期設定〕

$I \leftarrow \{P_0\}$, Xopt $\leftarrow \lambda$, Zopt $\leftarrow \infty$ とする; Step 2へ;

Step 2 :〔部分問題の選択〕

(i) $I = \emptyset$ ならば, Step 7へ;

(ii) $I \neq \emptyset$ ならば, $Pi \leftarrow s(I)$, $I \leftarrow I - \{Pi\}$ として, Step 3へ;

Step 3 :〔部分問題の評価〕

(i) $Pi \in L$ ならば, Step 6へ;

(ii) $Pi \notin L$ ならば, Step 4へ;

Step 4 :〔目的関数の下界による枝刈り〕

(i) $b(Pi) \geq$ Zopt ならば, Step 2へ;

(ii) $b(Pi) <$ Zopt ならば, Step 5へ;

Step 5 :〔部分問題の分割〕

$Pi$ を分割して得られる $k$ 個の部分問題を, $Pi_1, Pi_2, \cdots, Pi_k$ とする;

$I \leftarrow I \cup \{Pi_1, Pi_2, \cdots, Pi_k\}$ として, Step 2へ;

Step 6 :〔最適解の候補の更新〕

$$I \leftarrow \begin{cases} I & \text{if } z(Pi) \geq \text{Zopt}, \\ Pi & \text{if } z(Pi) < \text{Zopt}, \end{cases}$$

$$\text{Zopt} \leftarrow \min(z(Pi), \text{Zopt}),$$

として, Step 2へ;

Step 7 :〔アルゴリズム終了〕

停止.

アルゴリズム7.1とアルゴリズム7.2の相異は, Step 1, Step 3およびStep 6である.
れらの点を除いて, 両アルゴリズムの各Stepは同一である.

## 7.4　従来のサーチ関数の性質とその問題点

この節では，7.2節で定義された，Branch-and-Bound法の代表的な4つのサーチ関数の性質とその問題点を明らかにする．以下では，サーチ関数$s$を用いるBranch-and-Boundアルゴリズム$A$を$A(s)$で表わす．

### 7.4.1　アルゴリズムの計算時間

アルゴリズム$A$a($A$s)の各Stepの1回当りの計算時間は，処理される部分問題によらず，それぞれ一定であると仮定する．また，アルゴリズムのStep 3の1回当りの計算時間は，他のStepの1回当りの計算時間と比較して十分大きいと仮定する．

アルゴリズム$A$で問題$P_0$を解くときに，アルゴリズムのStep 3で処理される部分問題の個数をT($A$)で表わす．上記の仮定のもとでは，アルゴリズム$A$の計算時間は，$t$を$P_0$のみに依存する定数として，$t \times$T($A$)で与えられる．よって，以下では，アルゴリズム$A$の計算時間をT($A$)によって評価することにする．

Branch-and-Boundアルゴリズムでは，T($A$)の値は，一般に，解かれる問題$P_0$の性質に強く依存するため，その厳密な理論的評価は困難である．

ここでは，T($A(s)$)について，次の点についてだけ述べる．

まず，集合KaおよびKsを次式で定義する．

$$\mathrm{Ka} \triangleq \{P_i \mid P_i \in V,\ b(P_i) \leq z(P_0)\} \tag{7.12}$$

$$\mathrm{Ks} \triangleq \{P_i \mid P_i \in V,\ b(P_i) < z(P_0)\} \tag{7.13}$$

この時，以下の関係が成り立つことは容易に知られる．

$$\mathrm{T}(A\mathrm{a}(s)) \geq |\mathrm{Ka}| \quad \text{for all } s \tag{7.14}$$

$$\mathrm{T}(A\mathrm{s}(s)) \geq |\mathrm{Ks}| \quad \text{for all } s \tag{7.15}$$

$$\mathrm{T}(A\mathrm{a}(s_{\mathrm{BB}})) = |\mathrm{Ka}| \tag{7.16}$$

式(7.14)および式(7.16)より次のことが知られる．すなわち，全ての最適解を求める場合，アルゴリズム$A$でBest-Boundサーチ関数$s_{\mathrm{BB}}$を用いた時に，Step 3で処理される部分問題の個数は，他のサーチ関数を用いた時にStep 3で処理される部分問題の個数の下限を与える．よって，Best-Boundサーチ関数は，全ての最適解を求める場合には，計算時間(T($A$))に関して最適である．

### 7.4.2　必要な記憶容量

アルゴリズム$A(s)$を実現するために必要な記憶容量を，アルゴリズムの実行途中での，未

解決な部分問題の集合の要素の個数 $|I|$ の最大値で定義し，この値を $S(A(s))$ によって表わすことにする．

各サーチ関数を用いた時の $S(A(s))$ の値に関して以下のことが知られる．

$$S(A(s)) \leq k^n \quad \text{for } s = s_{BB},\ s_H,\ s_{BF} \tag{7.17}$$

$$S(A(s_D)) \leq (k-1)\cdot n+1 \tag{7.18}$$

式（7.17）の右辺は，木 $B$ が完全木である場合の $B$ の葉節点の集合 $L$ の要素の個数を表わしている．また式（7.18）の右辺は，木 $B$ が完全木である場合の $B$ の根 $P_0$ から任意の葉節点 $Pi$ に至るパス：

$$P_0 = Pi_0,\ Pi_1, \cdots,\ Pi_n = Pi \tag{7.19}$$

を考えた時の，$Pi_0,\ Pi_1, \cdots,\ Pi_{(n-1)}$ の子節点の集合から，集合 $\{Pi_1,\ Pi_2, \cdots,\ Pi_{(n-1)}\}$ を除いて得られる集合 $Ii$ の要素の個数を表わしている．すなわち，

$$N(j) \triangleq |\{P_k \mid P_k \in Ii,\ \ell(P_k) = j\}| \tag{7.20}$$

と定義すると，次式を得る．

$$N(j) = \begin{cases} 0 & ;\ \text{if } j=0 \\ k-1 & ;\ \text{if } 1 \leq j \leq n-1 \\ k & ;\ \text{if } j = n \end{cases} \tag{7.21}$$

よって，次式が成り立つ．

$$\begin{aligned} |Ii| &= \sum_{j=1}^{n} N(j) \\ &= (k-1)\times(n-1)+k \\ &= (k-1)\times n+1 \end{aligned} \tag{7.22}$$

式（7.17），式（7.18）より，以下のことが知られる．Depth-First サーチ関数 $s_D$ をいるアルゴリズム $A$ で問題 $P_0$ を解く場合に必要な記憶容量は，$O(k\times n)$ である．一方，est-Bound サーチ関数 $s_{BB}$，Heuristic サーチ関数 $s_H$，および Breadth-First サーチ関 $s_{BF}$ を用いた場合に必要な記憶容量は，ともに $O(k^n)$ である．よって，$s_D$ 以外のこれらのーチ関数を用いて大規模な問題を解く場合には，ぼう大な記憶容量を必要とする．

以上の理由から，大規模な組合せ論的最適化問題を Branch-and-Bound アルゴリズムで解時には，ほとんどの場合，サーチ関数として $s_D$ が用いられている．

記憶容量の不足の問題を解決する1つの方法として，茨木によって Depth-$m$ サーチが提案れている．[2,3]

このサーチ関数は，$s_D$ と $s_H$ との中間的な性質を持ち，パラメータ $m$ の値を1とした場合は $s_D$ と，また，$m \geq k^n$ の場合には $s_H$ とそれぞれ等価である．また，Depth－$m$ サーチ

を用いた時に必要な記憶容量は$O(n^m)$であるので，このサーチ関数を用いることによって記憶容量の不足の問題は部分的には緩和される．

しかしながら，与えられた記憶領域を有効に使用できるようにパラメータ$m$の値を決定するためには，解かれる問題のクラスに対する経験的な知識が必要である．よって，Depth－$m$サーチを用いたとしても，記憶容量の不足の問題は完全には解決されたとはいえない．

次節では，必要な記憶容量を自由に制御できる，2つのサーチ関数について述べる．

## 7.5　必要な記憶容量を自由に制御できるサーチ関数

この節では，アルゴリズムを実現するために必要な記憶容量を自由に制御できる，2つのサーチ関数，pdfsおよびslimを提案する．

### 7.5.1　サーチ関数pdfs

第6章で述べた，並列形Branch-and-BoundアルゴリズムPDFAでは，処理装置の台数をpとすると，アルゴリズムの実現に必要な記憶容量は$O(p \times k \times n)$であることが知られている．しかも，p台の処理装置を持つマルチプロセッサ・システムでPDFAによって問題を解く時の計算時間が，pの値を適当にえらべばp＝1の場合の計算時間の1/p以下になることは，6章で示した．

以下に述べるサーチ関数pdfs(parallel depth-first search)を用いるアルゴリズムは，p台の処理装置を持つマルチプロセッサ・システムの上でPDFAによって問題が解かれる過程を，逐次的な処理装置の上で模倣する．

〔手続き7.1〕(サーチ関数pdfs)

**入力**：未解決な部分問題の集合$I$；

**出力**：$Pi^* \in I$；

**記法**：$Q$をFIFOキューとする．$Q$の最後に$Pi$を追加する操作を$Q \Leftarrow Pi$で表わす．また，$Q$の先頭から要素を1個取り出し$Pi$に代入する操作を$Pi \Leftarrow Q$で表わす．$Q$は，アルゴリズムの初期設定(アルゴリズム$A$のStep 1)で$\emptyset$に初期化されているとする．また，pを$Q$の長さの上限を与えるパラメータとする．

**方法**：

Step 1：〔$Q$が空か否かの判定〕

$I' \leftarrow I$とする．

(i) $Q = \emptyset$ならば，Step 2へ；

(ii) $Q \neq \emptyset$ならば，Step 3へ；

Step 2：〔$Q$の生成〕

$q \leftarrow \min(p, |I'|)$とする；

$j = 1, 2, \cdots, q$について，

$P_i \leftarrow s_D(I'), I' \leftarrow I' - \{P_i\},$

$Q \Leftarrow P_i$ ；

を実行する；Step 3へ；

Step 3：〔部分問題の決定〕

$P_i^* \Leftarrow Q$ とする；手続き終了

### 7.5.2 サーチ関数slim

以下に示すサーチ関数は，未解決な部分問題の集合$I$の各レベルに属す要素の個数を与えらたパラメータ$w$以下に抑えることによって，$|I|$の最大値が常に$w \times n$以下に保たれるよに，$I$の要素を選択する．

まず，$I$のレベル$j$に属す部分問題の集合$I(j)$とする．すなわち，

$$I(j) \triangleq \{P_i \mid P_i \in I, \ell(P_i) = j\} \qquad (7.23)$$

た，$I(j)$の要素の個数をN($j$)とする．すなわち，

$$N(j) \triangleq |I(j)|$$

サーチ関数slim(space limited search)を，以下のように与える．

〔手続き7.2〕(サーチ関数slim)

入力：未解決な部分問題の集合$I$；

出力：$P_i^* \in I$

方法：$w$を，この手続きによって選ばれる部分問題$P_i$を分割して得られる$k$個の部分問題を$I$に加えた後での，$I$の各レベルに属する要素の個数の上限を表わすパラメータとする．

手法：

Step 1：〔初期設定〕

$I' \leftarrow I$とする；Step 2へ；

Step 2：〔部分問題の選択〕

$P_i \leftarrow s_{BB}(I')$ ；$j \leftarrow \ell(P_i)$ ；

$I' \leftarrow I' - \{P_i\}$ とする；

(i) $N(j+1) < w-k$ ならば，Step 3へ；

(ii) $N(j+1) > w-k$ ならば，Step 2へ；

Step 3：〔手続き終了〕

$P_i^* \leftarrow P_i$ とする；　手続き終了．

## 7.6　サーチ関数pdfsおよびslimの性質

この節では，前節で定義された2つのサーチ関数pdfsおよびslimの性質と，これらのサーチ関数を用いるアルゴリズムを実現するために必要な記憶容量などについて考察する．

### 7.6.1　pdfsの性質

サーチ関数pdfsに関して，次の定理が成り立つ．

〔定理7.1〕

(i) $p=1$ のとき，pdfsは，サーチ関数 $s_D$ と等価である．

(ii) $p \geq k^n$ のとき，pdfsは，サーチ関数 $s_{BF}$ と等価である．

〔証明〕　定理7.1の証明は，付録6(1)に示す．

上記の定理により，pが $1<p<k^n$ では，pdfsはDepth-Firstサーチ関数とBreadth Firstサーチ関数との中間的な性質を持つと考えられる．

すでに，第6章で述べたように，PDFAに関しては，処理の並列化による加速効果によって，pを適当にえらべば，アルゴリズムの計算量が，$p=1$ の場合よりも減少することが明らかにされている．よって，アルゴリズム7.1およびアルゴリズム7.2で，サーチ関数としてpdfsを用いた場合のアルゴリズムの計算時間は，パラメータpの値を適当にえらべば $p=1$ の場合よりも短縮されると考えられる．

また，pdfsを用いた時に，アルゴリズムの実現に必要な記憶容量に関しては，PDFAの場合と同様に，次の定理が成り立つ．

〔定理7.2〕

アルゴリズム7.1およびアルゴリズム7.2において，サーチ関数としてpdfsを用いた時の，アルゴリズムの実行途中で生成される未解決な部分問題の集合の要素の個数 $|I|$ は，次式を満足する．

$$|I| \leq p \times \{ (k-1) \times n + 1 \} \qquad (7.23)$$

定理7.2は，第6章の定理6.1と同様にして証明されるので，証明は省略する．

定理7.2より，pdfsで，pの値を変化させることにより，アルゴリズムを実現するために必要な記憶容量が自由に制御できることが知られる．

### 7.6.2 slimの性質

$I$を，アルゴリズムの実行途中の未解決な部分問題の集合とする．また，$I$の要素の中で，レベル$i$に属する部分問題の集合を$I(i)$とする．

さらに，集合$S$を次式で与える．

$$S \triangleq \{ i \mid 0 \leq i \leq n,\ I_{(i)} \neq \emptyset,\ |I_{(i+1)}| \leq w - k \} \qquad (7.26)$$

また，$I_{(i)} \neq \emptyset$をみたす$i$の最大値および最小値を，それぞれ$i_{max}$, $i_{min}$とする．

このとき，サーチ関数slimに関して，次の定理が成り立つ．

〔定理7.3〕

(i) $w = k$のとき，

$$i \notin S \quad \text{for} \quad i_{min} \leq {}^{\forall}i < i_{max} \qquad (7.27)$$

であれば，

$$\mathrm{slim}(I) = s_D(I) \qquad (7.28)$$

が成り立つ．

(ii) $w \geq k^n + k$ならば，

$$\mathrm{slim}(I) = s_{BB}(I) \quad \text{for} \quad {}^{\forall}I \subseteq V \qquad (7.29)$$

が成り立つ．

〔証明〕　定理7.3の証明は，付録6(2)に示す．

定理7.3より，サーチ関数slimは，$k \leq w < k^n + k$の時，Depth-Firstサーチ関数とBest-Boundサーチ関数との中間的な性質を持つと考えられる．よって，パラメータ$w$の値を増すことにより，サーチ関数slimを用いるBranch-and-Boundアルゴリズムの計算時間は，Best-Boundサーチを用いる場合の計算時間に漸近してゆくと考えられる．

7.4.1で述べたように，全ての最適解を求めるアルゴリズム$Aa$では，Best-Boundサーチを用いることによって計算時間が最小化される．よって，$w$の値が増すのにしたがって，アルゴリズムの計算時間は減少すると考えられる．

また，サーチ関数slimを用いる時に，アルゴリズムの実現に必要な記憶容量に関しては，次の定理が成り立つ．

〔定理7.4〕

アルゴリズム7.1およびアルゴリズム7.2でサーチ関数としてslimを用いた時の，アルゴリズムの実行途中で生成される未解決な部分問題の集合の要素の個数$|I|$は，次式を満足する．

$$|I| \leq w \times n \qquad (7.30)$$

〔証明〕　定理7.4の証明は付録6(3)に示す．

定理7.4より，slimを用いた時もまた，$w$の値を変化させることにより，アルゴリズムの実現に必要な記憶容量を自由に制御できることが知られる．

## 7.7　実験および結果

この節では，サーチ関数pdfaおよびslimを用いるBranch-and-Boundアルゴリズムの計算時間を実験をとおして測定し，前節で述べたアルゴリズムの計算時間に関する予備考察の結果と比較する．

### 7.7.1　問題の作製方法

実験に用いられた問題は，以下に示すパラメータを変えながら，計算機の一様乱数を用いて発生された最小被覆問題である．

(i)　問題に含まれる変数の個数$n$．

(ii)　制約条件の個数$s$．

(iii)　一つの制約条件中に含まれる変数の個数の分布範囲$\delta$．ここに，$\delta=[a, b]$のとき，一条件当りの変数の個数が，区間$[a, b]$で一様に分布するように，変数の個数を定める．

以下の実験では，$n=30$；$s=30$；$\delta=[4, 8]$，$[7, 11]$，$[10, 14]$，$[13, 17]$の4種類各20問題を対象とした．

### 7.7.2 実験方法および結果

上で述べた方法を用いて発生された最小被覆問題を，アルゴリズム7.2のサーチ関数として関数pdfs，slimを用いてそれぞれ解かせ，アルゴリズム中で処理された部分問題の個数（アルゴリズム7.2のStep 3の実行回数），および，アルゴリズムを実行するのに要した計算時間（CPU時間）をそれぞれ測定した．ただし，この実験では，最適解の1つだけを求めることにした．

以下では，アルゴリズム7.2のサーチ関数としてpdfsおよびslimを用いた時に，アルゴリズム中で処理された部分問題の個数の20問題についての平均値をそれぞれ$T_p$，$T_s$で，また，アルゴリズムを実行するのに要した計算時間（CPU時間）の平均値をそれぞれ$C_p$，$C_s$で表わす．

図7.1に$T_p$，$T_s$の値を示す．また，図7.2に$C_p$，$C_s$の値を示す．両図より，以下のことが知られる．

i) サーチ関数pdfsを用いた場合，パラメータpの値に対する$T_p$，$C_p$の値は，ほぼ下に凸である．すなわち，pの値の増加に伴い$T_p$，$C_p$の値は，いったん減少し，次に増加する傾向がみられる．これは，7.6.1で述べた，pdfsを用いるアルゴリズムの計算時間についての予備考察の結果と一致する．

ii) サーチ関数としてslimを用いた場合，$T_s$，$C_s$の値は，パラメータ$w$の値の増加に伴って単調に減少し，その下限（式（7.15）の$|K_s|$の値）に漸近する．これも，また，7.6.2で述べたslimを用いるアルゴリズムの計算時間についての予備考察の結果と一致する．

iii) パラメータpおよび$w$の値が等しい場合*，すなわち，ほぼ同一の記憶容量が与えられた時，pおよび$w$の値が比較的小さい場合には，$T_p$の値は$T_s$の値よりも小さくなる傾向がある．逆に，pおよび$w$の値が十分大きい場合には，$T_p$の値は$T_s$の値よりも大きくなる．

iv) パラメータpと$w$の値が等しくかつ，pと$w$の値がともに比較的小さい場合，たとえば，図7.2でp，$w \leq 64$では，$C_p$の値は$C_s$の値よりも小さい．逆に，p，$w$の値が比較的大きい場合には，たとえば，p，$w > 128$では，Cpの値はCsの値よりも大きくなる傾向がある．

---

*）7.6節で示したように，アルゴリズムの実行途中での$|I|$の値の上限は，pdfsを用いた場合，$p \times \{(k-1) \times n + 1\}$．またslimを用いた場合，$w \times n$である．よって，$k = 2$の時，nが十分大きければ，両者の値はほぼ等しい．

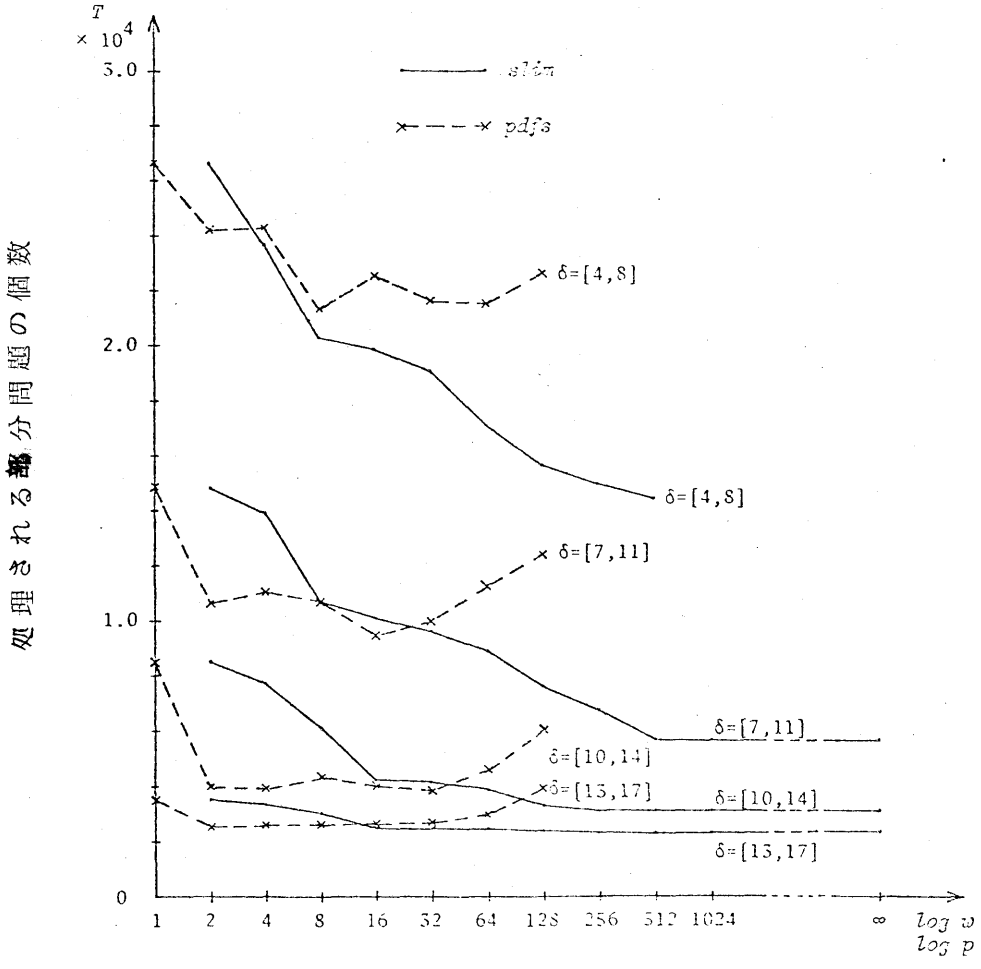
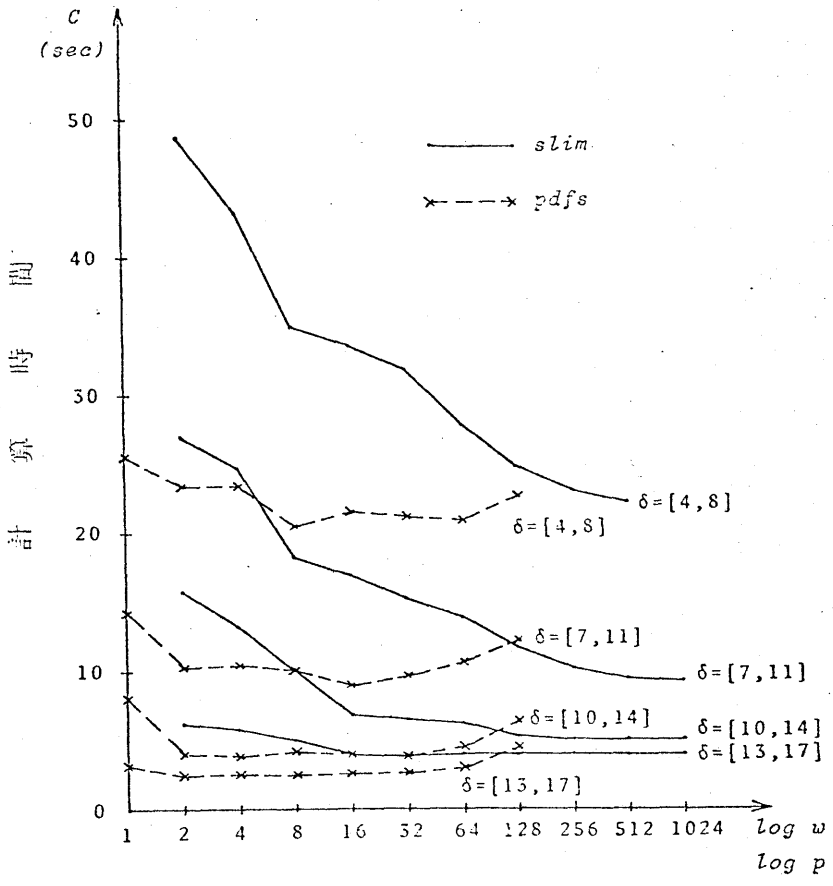



図7.1　slimおよびpdfsを用いるアルゴリズムで処理される部分問題の個数

δ：一条件当りの変数の個数の分布範囲




slimおよびpdfsのパラメータ

**図7.2　slimおよびpdfsを用いるアルゴリズムの計算時間**

δ：一条件当りの変数の個数の分布範囲

v)　pdfaでパラメータpの値を適切にえらぶことにより，Tp，Cpの値は，p＝1の場合と比較して，削減（短縮）できる．

vi)　slimでパラメータ$w$の値が十分大きいとき，すなわち，十分な記憶容量が与えられれば，Ts，Csの値は，$w=k$の場合と比較してかなり削減（短縮）される．

以上の結果より，Branch-and-Boundアルゴリズムで用いられるサーチ関数としては，使

用可能な記憶容量が比較的少ない場合にはpdfsが，また，十分に多い記憶容量が与えられた場合にはslimが，それぞれ適していると結論される．

### 7.7.3 考　察

先に示した図7.1および図7.2で，パラメータpおよび$w$の値が等しい時，Ts < Tpであるにもかかわらず，Cs > Cpとなる場合が多く存在する．この現象は，1部分問題当りのアルゴリズムの計算時間（Cp/TpおよびCs/Ts）が，用いられるサーチ関数によって異なることに起因している．

Branch-and-Boundアルゴリズムで用いられるしゅじゅのサーチ関数は，いずれも，未解決な部分問題の集合$I$の中から1つの要素を選択する．よって，この$I$を計算機の記憶装置の上で，どのような形で記憶し，かつ，$I$の中からどのようにして部分問題を選択するかが，サーチ関数自体の計算時間に大きな影響を及ぼす．よく知られているように，Depth-Firstサーチ関数およびBreadth-Firstサーチ関数は，それぞれLIFOスタックおよびFIFOキューを用いることによって実現される．[8] 一方，これら以外のサーチ関数を実現するために用いられる方法は，多くの場合，$I$を順序集合として取り扱い，その最小元をサーチ関数の値とする方法である．

この節で示した実験では，サーチ関数pdfs，slimをそれぞれ以下のようにして実現した．

まず，pdfsでは，$I$の各レベルに属する要素からなる部分集合を，それぞれ，リンク付きリストとして実現した．ここに，リンク付きリスト中の要素は，目的関数の下界の値の小さい順にならべられる．また，pdfsの値は，空でない最大のレベルを持つリストの先頭の要素とする．また，$I$中に要素を加える時には，その要素の属するレベルに対するリンク付きリストの中に，そのリストの各要素が$b$の値が増大する順にならぶように挿入する．よって，$I$から要素を1つとり出す操作は定数時間で実行できる．また，$I$に要素を1個追加するのに必要な計算時間は，$O(k \times p)$である．*

次に，slimでは，$I$をbalanced tree[6]として実現した．ただし，slimの値を計算するためには，手続き7.2のStep 2で$N(j+1) \leq w-k$を満足する要素がみつかるまで，$I$の要素を順に調べる必要がある．そこで，balanced treeの各節点に，順序集合$I$中の一つ前の要素と一つ後の要素を指すリンクを設け，上記の操作が容易に実行できるようにした．

---

*）各リンク付きリストの要素の個数は$k \times p$をこえることはない．

$I$をこのような両方向リンクのついたbalanced treeとして実現することにより，$I$への要素の追加は$O(\log_2|I|)$で実行することができる．一方，$I$を単なる線形リスト又は両方向リストとして実現した場合には，$O(|I|)$の計算時間が必要である．また，$I$から$N(j+1) \leq w-k$をみたす最小の要素を取り出すのに必要な計算時間は，$I$を上記のようなデータ構造を用いて実現した場合も，リンク付きリストとして実現した場合も，ともに$O(|I|)$である．よって，上記のデータ構造は，$I$への要素の追加の容易さの点で単純なリンク付きリスト構造よりもすぐれている．

## 7.8 むすび

本章では，必要な記憶容量を自由に制御できる，2つのサーチ関数pdfsおよびslimを提案し，これらの関数の性質を明らかにした．

これらのサーチ関数を用いた場合，従来最も良く使われてきたサーチ関数の1つであるDepth Firstサーチ関数と比較して，関数自体の計算時間が増加することは避けられない．しかし，これらのサーチ関数を用いることにより，アルゴリズム中で処理される部分問題の個数は，場合によりかなり削減される．

よって，これらのサーチ関数は，1部分問題当りの計算時間がサーチ関数の計算時間と比較して十分に大きい場合には特に有効である．さらに，使用可能な記憶容量が比較的少ない場合はpdfsが，また，使用可能な記憶容量が十分多い場合にはslimがそれぞれ適している．

また，サーチ関数slimでは，未解決な部分問題の集合$I$の各レベルに属する要素の個数を一律に定数$w$以下に制限している．この制限をより一般化して，要素の個数の上限を各レベルごとに$w_i$とした場合には，$|I|$の値の上限は$\sum_{i=1}^{n} w_i$となることは明らかである．このようにして一般化されたslimを用いた場合のアルゴリズムの効率の検討は，今後の課題である．

また，本章で提案されたサーチ関数を用いた場合と，たとえば，茨木によって提案されたDepth－mサーチ関数などを用いた場合の，アルゴリズムの効率の比較も今後の課題である．

なお，本章で示した実験には，名古屋大学大型計算機センターのFACOM 230/75が用いられた．また，実験に用いられたアルゴリズムはFORTRAN IV-Hで記述された．

# 第8章　あとがき

本論文では，整数計画問題の組合せ論的解法を構成するために必要な，さまざまな問題点を，いろいろな角度から検討し，それらに対する方策を明らかにしてきた．すなわち，第2章および第4章では，擬似ブール計画法のアルゴリズムで用いられる，いくつかの評価関数を提案し，それらについての理論的考察を行ない，さらに，アルゴリズムを実現して，その効率を実験的に調べた．また，第3章では，擬似ブール計画問題を記述する擬似ブール関数およびブール関数を，計算機内部で効率的に表現する方法，および，ハッシングの技法を用いた高速の数式処理アルゴリズムを導いた．また，第5章では，第4章までで得られた，汎用の擬似ブール計画法のアルゴリズムを問題向きに修正することによって，最小被覆問題を効率良く解くアルゴリズムが構成できることを示した．

次に，第6章では，将来の計算機システムの発展の一つの形態である，マルチプロセッサ・システムに着目し，マルチプロセッサ・システムに適した，Branch-and-Bound アルゴリズムの構成方法とその性質を明らかにした．さらに，マルチプロセッサ・システムの処理装置の台数の増加が，アルゴリズムの計算時間に及ぼす影響について，シミュレーションの結果にもとづいて考察を行なってきた．

最後に第7章では，Branch-and-Bound アルゴリズムの実現に必要な記憶容量について考察し，必要な記憶容量を自由に制御できる，2つのサーチ関数を提案した．さらに，これらのサーチ関数を用いるアルゴリズムの計算時間に関する予備考察の結果を，実験結果と比較した．

本論文で未解決のまま残された問題点，および今後の研究課題については，各章ごとにまとめに記した．以下では，本論文全体をしめくくる意味で，今後の研究の方向として，特に重要と思われる点について概観することにする．

(1)　本論文では，0－1計画問題の一般的定式化の1つとして，擬似ブール計画法について考察してきた．本論文で定式化された，擬似ブール計画法のアルゴリズムでは，制約条件がブール関数の形で与えられている．不等式条件によって与えられる制約条件を，ブール制約条件に変換するためには，一般には，かなりの計算量を必要とする．したがって，与えられる問題の形式によっては，不等式条件のまま処理を行なった方が良い場合もあると考えられる．

制約条件が不等式または等式で与えられる問題に対する効率的解法を明らかにするのは，今後の課題の1つであるが，これは，第4章に示したアルゴリズムをもとにして，比較的容易に実現されると考えられる．

(2) 本論文で示した整数計画法のアルゴリズムは，いずれも，最適解の一つまたは全部を出力する．実用的見地からは，最適解に準ずる可能解（準最適解）が求められれば十分である場合も多いと考えられる．[4]

準最適解を求める場合には，次の2通りの異なる要請が考えられる．

(i) 準最適解を得るまでに要する計算時間がある範囲以下であること．

(ii) 準最適解による目的関数の値と，最適解による目的関数の値との差が，相対的あるいは絶対的に，ある範囲内にあること．

準最適解を得るアルゴリズムの構成に関しては，いくつかの方法が考えられるが，本論文の第7章で示した2つのサーチ関数のいずれを用いても，(i)の要請を満たすアルゴリズムは容易に構成できると考えられる．これらのサーチ関数を用いて準最適解を得るアルゴリズムを構成すること，および，その効率について検討することは，今後の課題である．

(3) 本論文で示した整数計画法のアルゴリズムは，第5章の最小被覆問題の解法を除いて，いずれも，いわば汎用の解法である．これらのアルゴリズムを基礎とすれば，対象とする問題の性質を利用することによって，第5章で示したような，問題向きの効率的解法が得られると期待される．最小被覆問題以外の組合せ問題に対する問題向きアルゴリズムの構成と，従来の解法との効率の比較は，今後の課題である．

(4) 第6章で述べた，並列形Branch-and-Boundアルゴリズムでは，マルチプロセッサ・システムの共有記憶装置上に置かれた未解決な部分問題に対するアクセスの競合の問題は，本質的に避け得ないものである．よって，処理装置の台数が非常に多い場合には特に，アクセスの競合の問題に対する対策の良否が，システムの効率を左右する要因となると考えられる．

したがって，競合の程度の解析と，それに対する適切な対策を検討することは，並列形Branch-and-Boundアルゴリズムの実用化のための急務である．第6章のまとめで述べた方策は，かなり有効に競合を緩和すると期待されるが，その検証は今後の課題である．

(5) 最後に，本論文で述べてきた整数計画問題と，人工知能の分野で取扱われている問題の類似性に着目すると，次のような問題が考えられる．すなわち，人工知能の分野での，定理の自動証明，各種の数学プログラム，各種のゲーム，および，一般問題解決システムなどで取扱われる問題は，組合せ的な問題であると見なすことができる．[1, 3, 5] さらに，これらの問題が発見的方法によって解かれていることを考慮すると，これらの問題の多くは，計算機のみによって解かれるよりも，むしろ，人間－機械系によって，対話形式で解かれるべきであると考えられる．

このような形式で問題の処理を行なうためには，問題を記述する言語およびその処理系を開

発する必要がある．この目的に用いられる言語の持つべき特徴としては，まず，問題の記述能力が高いことがあげられる．すなわち，問題の記述が人間工学的観点からも容易であること，再帰的表現を用いうること，および，集合，木，グラフ，リンク付きリスト，スタック，キュー，数式などのデータ構造を扱えることが必要である．

また，その言語の処理系は，再帰的手続きの実行，各種のデータ構造の処理，ユーザとの対話などが効率良く実現できることが必要である．

このような言語およびその処理系の実現は，興味深い問題であるが，まだ，未解決の問題も多いように思われる．このような言語およびその処理系の開発もまた，今後の課題である．

# 謝　　辞

筆者が，卒業研究から始めて現在に至るまでの長い間研究を続けることができたのは，以下の先生方の暖かい御指導と御援助によるものであり，ここに感謝の意の表します．

名古屋大学　福村晃夫教授には，研究上の御指導と御援助のみならず，終始精神的な面での御援助も賜わった．ここに衷心より感謝の意を表します．

本学　本多波雄教授には，東北大学に御在籍中より今日に至るまで，御多忙の中を熱心な御指導と御討論を賜わった．ここに衷心より感謝の意を表します．

また，未熟な筆者を，研究の初期から今日に至るまで，終始懇切な御指導と御鞭達を賜わった，本学　吉田雄二助教授に，深く感謝致します．

また，日頃研究上の諸問題に関して，さまざまな角度から，御意見をいただき，御討論を賜わった，三重大学　稲垣康善教授，静岡大学　阿部圭一郎助教授，本学　羽賀隆洋助手，および，杉野花津江助手に，心から感謝の意を表します．

さらに，日頃熱心に御討論いただき，貴重な御意見を賜わった，名古屋大学情報工学科福村研究室および電子工学科本多研究室の諸先輩ならびに諸賢に，心から感謝の意を表します．

# 参 考 文 献

## 第 1 章の参考文献


(1) Balas,E.: An Additive Algorithm for Solving Linear Programs with Zero-One Variables, Oper. Res., Vol. 13, pp.517-546 (1965)

(2) Bellman,R.E.: *Dynamic Programming*, Prinston University Press (1957)

(3) Deo,N.: *GRAPH THEORY with Applications to Engineering and Computer Science*, Prentice-Hall (1974)

(4) Gomory,R.E.: An Algorithm for Integer Solutions to Linear Programs, in *Recent Advances in Mathematical Programming*, R. L. Graves and P. Wolfe eds, McGraw-Hill (1963)

(5) Greenberg,H.: *Integer Programming*, Academic Press (1971)

(6) Hoffer,J.A.: An Integer Programming Formulation of Computer Data Base Design Problems, Information Science, Vol. 11, pp.29-48 (1976)

(7) Hu,T.C.: *Integer Programming and Network Flows*, Addison-Wesley (1969)

(8) Lawler,E.L. et al.: Branch-and-Bound Methods: A Survey, Oper. Res, Vol. 14, No. 4, pp.699-719 (1966)

(9) Papaioannou,S.G.: Optimal Test Generation in Combinatorial Networks by Pseudo-Boolean Programming, IEEE, Trans. on Comp., Vol. c-26, No. 6, pp.553-560 (1977)


## 第 2 章の参考文献


(1) Nilsson,N.J.: *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill (1971)

(2) 吉田，稲垣，福村：Branch and Bound 法にもとづく擬似ブール計画法のアルゴリズム，電子通信学会雑誌，第 50 巻第 10 号，pp.231-238 (1967)

(3) Hammer,P.L. and Peled,U.N.: On the Maximization of a Pseudo-Boolean Function, Journal of the ACM, Vol. 19,No. 2, pp.265-282 (1972)

(4) 吉田，稲垣，福村：擬似ブール計画法における不等式条件の処理について，電子通信学会創立 50 周年記念全国大会予稿集，講演番号 22 (1967)

(5) 今井，吉田，福村：目的関数と制約条件式の特徴を利用した擬似ブール計画法のアルゴリ

ズムについて，電子通信学会，オートマトンと言語研究会資料，AL74-46，pp. 75-86 (1975)

(6) 稲垣，福村：制約条件を持つ擬似ブール計画法について，電子通信学会雑誌，第50巻第6号，pp.47-54 (1967)

第3章の参考文献

(1) Knuth,D.E.: *The Art of Computer Programming*, Vol. 3, Sorting and Searching, Chap. 6.4, Addison-Wesley (1973)

(2) 今井，吉田，福村：目的関数と制約条件式の特徴を利用した擬似ブール計画法のアルゴリズムについて，電子通信学会オートマトンと言語研究会資料AL74-46，pp.75-86 (1975)

(3) 今井，吉田，福村：ハッシングの技法を用いた数式処理アルゴリズムとその擬似ブール計画法への応用，情報処理，第18巻第7号，pp.639 - 647 (1977)

4) 吉田，稲垣，福村：Branch and Bound法にもとづく擬似ブール計画法のアルゴリズム，電子通信学会雑誌，第50巻第10号，pp.231-238

5) 高木：初等整数論講義，共立出版 (1931)

第4章の参考文献

1) 今井，吉田，福村：Branch and Bound法にもとづく擬似ブール計画法のアルゴリズムにおけるheuristicsの良さの評価について，電子通信学会，オートマトンと言語研究会資料AL76-35，pp. 9 - 17 (1976)

2) 今井，吉田，福村：目的関数と制約条件式の特徴を利用した擬似ブール計画法のアルゴリズムについて，電子通信学会，オートマトンと言語研究会資料AL74-46，pp. 75-86 (1975)

3) 今井，吉田，福村：ハッシングの技法を用いた数式処理アルゴリズムとその擬似ブール計画法への応用，情報処理，第18巻第7号,pp.639 - 647 (1977)

4) Hammer,P.L. and Peled,U.N.: On the Maximization of a Pseudo-Boolean Function, Journal of the ACM, Vol. 19, No.2, pp.256-282 (1972)

5) Nilsson,N.J.: *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill (1971)

## 第５章の参考文献


(1) 阿部，福村：最小被覆問題を解くアルゴリズム，情報処理，Vol 10, NO. 4, pp 227 − 234 (1969)

(2) Balas,E. and Padberg,M.: Set Partitioning: A Survey, SIAM Review, Vol. 18, No. 4, pp.710-761 (1976)

(3) Deo,N.: *GRAPH THEORY with Applications to Engineering and Computer Science*, Prentice-Hall (1974)

(4) 今井，吉田，福村：最小被覆問題の擬似ブール計画法による解法，電子通信学会論文誌 Vol 60 −D , No. 10, pp 891 − 893 (1977)

(5) Phister,M.,Jr.: *Logical Design of Digital Computers*, John Wiley & Sons (1958)

(6) Slagle,J.R. et al.: A New Algorithm for Generating Prime Implicants, IEEE, Trans. on Comp., Vol. c-19, No. 4, pp.304-310 (1970)


## 第６章の参考文献


(1) Aho,A.V. et al.: *The Design and Analysis of Computer Algorithms*, Addison-Wesley (1974)

(2) Dijkstra,E.W.: Co-operating Sequential Processes, in *Programming Languages*, Genuys,F. ed., Academic Press (1968)

(3) Enslow,P.H.,Jr.: *Multiprocessors and Parallel Processing*, A Wiley-Interscience publication (1974)

(4) Fennel,R.D.: Parallelism in Artificial Intelligence Problem Solving: A Case Stydy of Hearsay II, IEEE, Trans. on Comp., Vol. c-26, No.2, pp.98-111 (1977)

(5) Flynn,M.J.: Some Computer Organizations and their Effectiveness, IEEE, Trans. on Comp., Vol. c-21, No. 9, pp.984-960 (1972)

(6) Fuller,S.H. et al.: Multi-Microprocessors: An Overview and Working Example, Proc. of IEEE, Vol. 66, No. 2, pp.216-228 (1978)

(7) Ibaraki,T.: Theoretical Consideration on Computational Performance of Branch-and-Bound Algorithms, presented at IX Int. Symp. on Math. Prog., held in Budapest, Hungary (1976)

(8) 今井，吉田，福村：目的関数と制約条件式の特徴を利用した擬似ブール計画法のアルゴリズムについて，電子通信学会，オートマトンと言語研究会資料 AL 74 − 46,

pp.75-86 (1975)

(9) 今井，吉田，福村：Branch-and-Bound法にもとづく擬似ブール計画法のアルゴリズムにおけるheuriticsの良さの評価について，電子通信学会，信学技報AL76－35，pp.9-17 (1976)

(10) 今井，吉田，福村：最小被覆問題の擬似ブール計画法による解法，電子通信学会，論文誌 Vol 60－D, No. 10, pp.891-893 (1977)

(11) 今井，吉田，福村：Branch-and-Bound形アルゴリズムの並列化とその評価，昭和52年度情報処理学会全国大会（第18回）講演番号327，pp.645-646 (1977)

(12) 今井，吉田，福村：Branch-and-Boundアルゴリズムの並列化とその評価，電子通信学会，信学技報AL78－8, pp.67-76 (1978)

(13) Larson,R.E.: Parallel Processing Algorithms for the Optimal Control of Nonlinear Dynamic Systems, IEEE, Trans. on Comp., Vol. c-22, No.8, pp.777-786 (1973)

(14) Lawler,E.L. et al.: Branch-and-Bound Method: A Survey, Opr. Res., Vol. 14, No. 4, pp.699-719 (1966)

## 第7章の参考文献

(1) Ellwein,L.B.: A Flexible Enumeration Scheme for Zero-One Programming. Oper. Res., Vol. 22, No. 1, pp.144-150 (1974)

(2) Ibaraki,T.: A Generalization of Depth-First Search in Branch-and-Bound Algorithms, working paper (1975)

(3) Ibaraki,T.: Theoretical Consideration on Computational Performance of Branch-and-Bound Algorithms, presented at IX Int. Symp. on Math. Prog., held in Budapest, Hungary (1976)

(4) Ibaraki,T.: Theoretical Comparisons on Search Strategies on Branch-and-Bound Algorithm, Int. Journal of Computer and Information Science, Vol. 5, No. 4, pp. 315-344 (1976)

(5) 今井，吉田，福村：必要な記憶容量を自由に制御できるBranch-and-Bound法のサーチ・ストラテジについて，情報処理学会第19回全国大会予稿集3E－10，pp.359-360 (1978)

(6) Knuth,D.E.: *The Art of Computer Programming*, Vol. 3, Sorting and Searching, Addison-Wesley (1973)

(7) Lawler,E.L. et al.: Branch-and-Bound Methods: A Survey, Oper. Res., Vol. 14, No. 4, pp.699-719 (1966)

(8) Nilsson,N.J.: *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill (1971)

(9) Tuan,N.Ph.: A Flexible Tree-Search Method for Integer Programming Problems, Oper. Res., Vol. 19, No. 1, pp.115-119 (1971)

## 第８章の参考文献

(1) Banerji,R.B.: *Theory of Problem Solving - An Approach to Artificial Intelligence -*, American Elsevier Publishing Inc. (1969)

(2) Lauriere,J.: A language and a Program for Stating and Solving Combinatorial Problems, Artificial Intelligence, Vol. 10 (1978)

(3) Nilsson,N.J.: *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill (1971)

(4) Sahni,S.: Approximate Algorithms for the 0/1 Knapsack Problem, Journal of the ACM, Vol. 22, No. 1, pp.115-124 (1975)

(5) Slagle,J.R.: *Artificial Intelligence: The Heuristic Programming Approach*, McGraw-Hill (1971)

(6) Wells,M.B.: Aspects of Language Design for Combinatorial Computing, IEEE, Trans. on Elec. Comp., Vol. EC-13, No. 8, pp.431-438 (1964)

# 付　　録

## 付録1　（第2章の付録）

### (1)　定理2.1の証明について

定理2.1を証明するために，まず次の補助定理を示す．

〔補助定理A 1.1〕

式（2.3）で与えられるF(x)について$S_{ij}$を次のように定義する．

$$S_{ij} \triangleq \{ x \mid x \in \{1,0\}^n,\ x_j = \overline{\varepsilon}_{ij} \},\ \text{for } 1 \le i \le s,\ j \in I_i \tag{A1.1}$$

ここに，$I_i$は，式（2.14）で与えられる．

この時，次式が成り立つ．

$$\bigcap_{1 \le i \le s} \bigcup_{j \in I_i} S_{ij} = S(F) \tag{A1.2}$$

〔証　明〕

$\xi = (\xi_1, \xi_2, \cdots, \xi_n)$，とすると，次の命題が成り立つ．*

$$\begin{aligned}
&\xi \in \bigcap_{1 \le i \le s} \bigcup_{j \in I_i} S_{ij} \\
\Longleftrightarrow\ & 1 \le \forall i \le s,\ \exists j \in I_i \text{ such that } \xi_j = \overline{\varepsilon}_{ij} \\
\Longleftrightarrow\ & 1 \le \forall i \le s,\ F_i(\xi) = 0 \\
\Longleftrightarrow\ & F(\xi) = 0 \\
\Longleftrightarrow\ & \xi \in S(F)
\end{aligned} \tag{A1.3}$$

よって式（A 1.2）が成り立つ．　Q.E.D.

〔定理2.1の証明〕

ここでは，式（2.16）が成り立つことのみを示す．式（2.17）が成り立つことは式（2.16）の場合と同様にして示すことができる．

以下の証明は，$y(x)$, F(x) に含まれる変数の個数$n$についての帰納法によって行なわれる．

(i) $n = 0$の時，$y(x)$は定数関数である．よって inf($y$,F) の定義より，明らかに式（2.16）が成り立つ．

(ii) $n \le r$の時，式（2.16）が成り立つと仮定すると，$n=r+1$の時にも式（2.16）が成り立つことを示す．

(a) F(x) ≡ 0 の場合

inf($y$,F) の定義より，次式を得る．

$$\inf(y,F) = \min_{1 \le j \le r+1} \left[ \min_{v \in \{0,1\}} \inf(y(x_j = v), 0) \right] \tag{A1.4}$$

---

*）　⟺ は，対等を表わす．

式（A 1.4）の右辺で，$y(x_j=v)$ はたかだか $r$ 個の変数しか含まない．よって，$S(F)=\{0,1\}^{r+1}$ であること，および帰納法の仮定により，次式を得る．

$$\begin{aligned}
&\inf(y,F)\\
&= \min_{1\le i\le s}\ [\ \min_{v\in\{0,1\}}\ [\ \min\{y(x_j=v)\,|\,x\in\{0,1\}^r\}]]\\
&= \min\{y(x)\,|\,x\in\{0,1\}^{r+1}\}\\
&= \min\{y(x)\,|\,x\in S(F)\}
\end{aligned}$$

よって，$F(x)\equiv 0$ の時にも，式（2.16）が成り立つ．

(b) $F(x)\not\equiv 0$ の場合：

$$S_{ij} \triangleq \{x\,|\,x\in\{0,1\}^r,\ x_j=\overline{\varepsilon}_{ij}\},\ \text{for } 1\le i\le s,\ j\in I_i \tag{A1.6}$$

と定義すると，補助定理A 1.1より，次式を得る．

$$S(F) = \bigcap_{1\le i\le s}\ \bigcup_{j\in I_i} S_{ij} \tag{A1.7}$$

よって，次式を得る．

$$\begin{aligned}
&\min\{y(x)\,|\,x\in S(F)\}\\
&= \max_{1\le i\le s}\ [\ \min\{y(x)\,|\,x\in \bigcup_{j\in I_i} S_{ij}\}]\\
&= \max_{1\le i\le s}\ [\ \min_{j\in I_i}\ [\min\{y(x)\,|\,x\in S_{ij}\}]]
\end{aligned} \tag{A1.8}$$

ここで，式（A 1.7）より，

$$S(F) = \bigcap_{1\le i\le s}\ \bigcup_{j\in S_{ij}} (S_{ij}\cap S(F)) \tag{A1.9}$$

が成り立つことに注意すると，式（A 1.8）は次のように書き換えられる．

$$\begin{aligned}
&\min\{y(x)\,|\,x\in S(F)\}\\
&= \max_{1\le i\le s}\ [\ \min_{j\in I_i}\ [\min\{y(x)\,|\,x\in S_{ij}\cap S(F)\}]]
\end{aligned} \tag{A1.10}$$

さらに，$S_{ij}$ の定義により，次式を得る．

$$\begin{aligned}
&\min\{y(x)\,|\,x\in S(F)\}\\
&= \min\{y(x_j=\overline{\varepsilon}_{ij})\,|\,x\in\{0,1\}^r,\ F(x_j=\overline{\varepsilon}_{ij}) = 0\}
\end{aligned} \tag{A1.11}$$

式（A 1.11）で，$y(x_j=\overline{\varepsilon}_{ij})$，$F(x_j=\overline{\varepsilon}_{ij})$ は，ともに，たかだか $r$ 個の変数しか含まない．

よって，帰納法の仮定により，次式が成り立つ．

$$\begin{aligned}
&\inf(y(x_j=v),F(x_j=v))\\
&= \min\{y(x_j=v)\,|\,x\in\{0,1\}^r,\ F(x_j=\overline{\varepsilon}_{ij}) = 0\}
\end{aligned} \tag{A1.12}$$

式（A 1.10），式（A 1.11）および式（A 1.12）より，次式を得る．

$$\begin{aligned}
&\min\{y(x)\,|\,x\in S(F)\}\\
&\ge \max_{1\le i\le s}\ [\ \min_{j\in I_i}\ \inf(y(x_j=\overline{\varepsilon}_{ij}),F(x_j=\overline{\varepsilon}_{ij})]
\end{aligned} \tag{A1.13}$$

よって，$F(x) \not\equiv 0$ の場合にも式（ 2.16 ）が成り立つことが示された． Q.E.D.

(2) 定理 2.2 の証明について

定理 2.2 を証明するために，次の関数を定義する．

$$\sigma(Y_i(x)) \triangleq \begin{cases} 1 & \text{if } Y_i(x) \not\equiv 0 \\ 0 & \text{if } Y_i(x) \equiv 0 \end{cases} \tag{A1.14}$$

式（ 2.5 ）で各 $Y_i(x)$ がブール関数であることから，関数 $\sigma$ と $Y_i(x)$ について，次の不等式が成り立つ．

$$\begin{aligned} &0 \le \sigma(Y_i(x_j=v)) \le \sigma(Y_i(x)) \le Y_i(x) \le 1, \\ &\text{for } 1 \le {}^\forall i \le m,\ 1 \le {}^\forall j \le n,\ {}^\forall v \in \{0,1\} \end{aligned} \tag{A1.15}$$

〔定理 2.2 (i)の証明〕

以下では，式（ 2.23 ）が成り立つことを示す．式（ 2.24 ）が成り立つことは，式（ 2.23）の場合と同様にして示される．以下の証明は，$k$ についての数学的帰納法によって行なわれる．

(i) $k = 0$ の場合：

まず，$k=0$ の場合に，式（ 2.23 ）が成り立つことを示す．

(a) $y(x)$ が定数関数の場合：

が定数関数の場合，明らかに，

$$lb_1(y,F) = lb_0(y,F) \tag{A1.16}$$

である．よって式（ 2.23 ）が成り立つ．

(b) $F(x) \equiv 0$ の場合：

式（ A 1.15 ）より，次の命題が成り立つ．

$$\begin{aligned} &a_i \le a_i\sigma(Y_i(x_j=v)) \le 0 \quad \text{if } a_i < 0, \\ &\text{for } 1 \le {}^\forall i \le m,\ 1 \le {}^\forall j \le n,\ {}^\forall v \in \{0.1\} \end{aligned} \tag{A1.17}$$

$$\begin{aligned} &a_i \ge a_i\sigma(Y_i(x_j=v)) \ge 0 \quad \text{if } a_i > 0, \\ &\text{for } 1 \le {}^\forall i \le m,\ 1 \le {}^\forall j \le n,\ {}^\forall v \in \{0.1\} \end{aligned} \tag{A1.18}$$

式（ 2.19 ），式（ A 1.17 ），式（ A 1.18 ）より，次式を得る．

$$\begin{aligned} & lb_0(y(x_j=v),0) \\ = \ & a_0 + \sum_{1\le i\le m}^{-} a_i\sigma(Y_i(x_j=v)) \\ \ge \ & a_0 + \sum_{1\le i\le m}^{-} a_i \\ = \ & lb_0(y,0) \end{aligned} \tag{A1.19}$$

式（ 2.21 ）に $k=1$ を代入することにより，次式を得る．

$$lb_1(y,0) = \min_{1\le j\le n} \left[ \min_{v\in\{0.1\}} lb_0(y(x_j=v),0) \right] \tag{A1.20}$$

式（A 1.19）および式（A 1.20）より次式を得る．

$$lb_1(y,0) \geq lb_0(y,0) \tag{A1.21}$$

よって，F(x) ≡ 0 の場合にも式（2.23）が成り立つことが示された．

(c) F(x) ≠ 0 の場合：

F(x) ≡ 0 の場合と同様にして，次式を得る．

$$\begin{aligned} & lb_0(y(x_j=\bar{\varepsilon}_{ij}),F(x_j=\bar{\varepsilon}_{ij})) \\ = \ & a_0 + \sum_{1\leq i\leq m}^{-} a_i\sigma(Y_i(x_j=\bar{\varepsilon}_{ij})) \\ \geq \ & a_0 + \sum_{1\leq i\leq m}^{-} a_i \\ = \ & lb_0(y,F) \end{aligned} \tag{A1.22}$$

式（2.21）に $k=1$ を代入することにより，次式を得る．

$$lb_1(y,F) = \max_{1\leq i\leq s} \left[ \min_{j\in I_i} lb_0(y(x_j=\bar{\varepsilon}_{ij}),F(x_j=\bar{\varepsilon}_{ij}))\right] \tag{A1.23}$$

式（A 1.22）および式（A 1.23）より，次式を得る．

$$lb_1(y,F) \geq lb_0(y,F) \tag{A1.24}$$

よって，F(x) ≠ 0 の場合にも式（2.23）が成り立つことが示された．

以上により，$k=0$ の場合に式（2.23）が成り立つことが示された．

(ii) 次に，$k=r(\geq 0)$ の時，式（2.23）が成り立つと仮定して，$k=r+1$ の時にも式（2.23）が成り立つことを示す．

(a) $y(x)$ が定数関数の場合：

$y(x)$ が定数関数の場合，明らかに，

$$lb_{r+2}(y,F) = y(x) = lb_{r+1}(y,F) \tag{A1.25}$$

が成り立つ．よって式（2.23）が成り立つ．

(b) F(x) ≡ 0 の場合：

定義 2.5 より，次式を得る．

$$lb_{r+2}(y,0) = \min_{1\leq j\leq n} \left[ \min_{v\in\{0,1\}} lb_{r+1}(y(x_j=v),0)\right] \tag{A1.26}$$

$$lb_{r+1}(y,0) = \min_{1\leq j\leq n} \left[ \min_{v\in\{0,1\}} lb_{r}(y(x_j=v),0)\right] \tag{A1.27}$$

帰納法の仮定により，

$$lb_{r+1}(y,F) \geq lb_r(y,F) \tag{A1.28}$$

が成り立つことから，次の命題が成り立つ．

$$lb_{r+1}(y(x_j=v),0) \geq lb_r(y(x_j=v),0),$$
$$\text{for } 1 \leq {}^{\forall}j \leq n,\ {}^{\forall}v \in \{0,1\} \tag{A1.29}$$

式（A 1.26），式（A 1.27）および式（A 1.29）より，明らかに，次式が成り立つ．

$$lb_{r+2}(y,0) \geq lb_{r+1}(y,0) \tag{A1.30}$$

よって，F(x) ≡ 0 の場合にも式（2.23）が成り立つ．

(c) F(x) ≢ 0 の場合：

F(x) ≡ 0 の場合と同様に，定義 2.5 より，次式を得る．

$$lb_{r+2}(y,\mathrm{F}) = \max_{1\leq i\leq s}\ [\ \min_{j\in I_i} lb_{r+1}(y(x_j=\bar{\varepsilon}_{ij}),\mathrm{F}(x_j=\bar{\varepsilon}_{ij}))] \tag{A1.31}$$

$$lb_{r+1}(y,\mathrm{F}) = \max_{1\leq i\leq s}\ [\ \min_{j\in I_i} lb_{r}(y(x_j=\bar{\varepsilon}_{ij}),\mathrm{F}(x_j=\bar{\varepsilon}_{ij}))] \tag{A1.32}$$

F(x) ≡ 0 の場合と同様の理由により，次の命題が成り立つ．

$$lb_{r+1}(y(x_j=\bar{\varepsilon}_{ij}),\mathrm{F}(x_j=\bar{\varepsilon}_{ij})) \geq lb_r(y(x_j=\bar{\varepsilon}_{ij}),\mathrm{F}(x_j=\bar{\varepsilon}_{ij})),$$
$$\text{for } 1 \leq {}^{\forall}i \leq s,\ {}^{\forall}j \in I_i \tag{A1.33}$$

式（A 1.31），式（A 1.32），および式（A 1.33）より，明らかに次式が成り立つ．

$$lb_{r+2}(y,\mathrm{F}) \geq lb_{r+1}(y,\mathrm{F}) \tag{A1.34}$$

よって，F(x) ≢ 0 の場合にも式（2.23）が成り立つことが示された．

以上により，$k=r+1$ の場合にも式（2.23）が成り立つことが示された．Q.E.D.

〔定理 2.2 (ii)の証明〕

$k \geq n$ の場合には，定義 2.4 および定義 2.5 より，$lb_k(y,\mathrm{F})$ と $\inf(y,\mathrm{F})$ および $ub_k(y,\mathrm{F})$ と $\sup(y,\mathrm{F})$ はそれぞれ等価である．よって，明らかに式（2.25）および式（2.26）が成り立つ．Q.E.D.

付録２（第３章の付録）

(1) 式（3.20）の導出について

表3.1に定義される演算は，$\varepsilon_{ij}$と$x_j$に与える値$c_j$との組に対する$\tilde{\varepsilon}_{ij}$の値を与える．xの内部表現$(\xi,\theta)$および$F_i(x)$の内部表現$(\sigma_i, \tau_i)$から，$\tilde{F}_i(x)$の内部表現$(\tilde{\sigma}_i, \tilde{\tau}_i)$を得る関数をそれぞれαおよびβとする．すなわち，

$$\begin{cases} \tilde{\sigma}_i = \alpha(\xi, \theta, \sigma_i, \tau_i) & \text{(A2.1)} \\ \tilde{\tau}_i = \beta(\xi, \theta, \sigma_i, \tau_i) & \text{(A2.2)} \end{cases}$$

$\tilde{\sigma}_i, \tilde{\tau}_i, \xi, \theta, \sigma_i, \tau_i$の同一桁を考えると，αおよびβはともに4変数のブール関数と考えることができる．

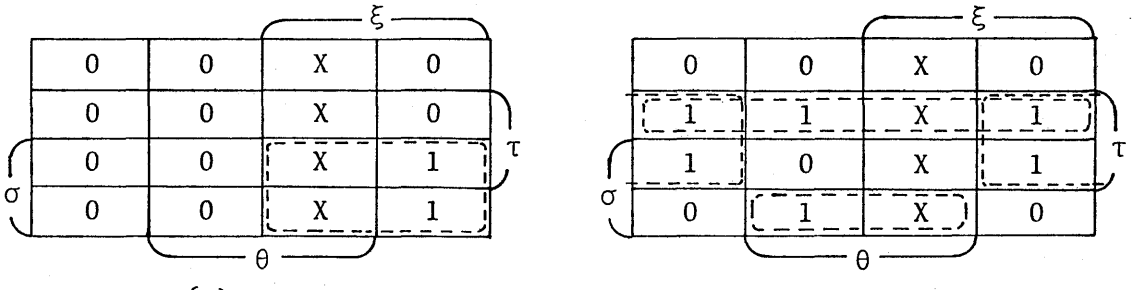
関数α,βを決定するため，Veitch - Karnaugh 図表を表3.1をもとにしてつくり，図A 2.1に示す．

図A 2.1より，関数αおよびβは，それぞれ次式で与えられる．

$$\alpha(\xi, \theta, \sigma_i, \tau_i) = \xi \wedge \sigma_i \qquad \text{(A2.3)}$$

$$\begin{aligned} &\beta(\xi, \theta, \sigma_i, \tau_i) \\ &= (\neg\theta) \wedge \tau_i \vee (\neg\sigma_i) \wedge \tau_i \vee \theta \wedge \sigma_i \wedge (\neg\tau_i) \\ &= \neg(\theta \wedge \sigma_i) \wedge \tau_i \vee \theta \wedge \sigma_i \wedge (\neg\tau_i) \\ &= (\theta \wedge \sigma_i) \oplus \tau_i \end{aligned} \qquad \text{(A2.4)}$$

図　A2.1　関数αおよびβのVeitch - Karnaugh 図表




(a) 関数α　　(b) 関数β

(2) 式（3.24）の導出について

上記の場合と同様に考えて，$Y_i(x)$の内部表現$\eta_i$およびxの内部表現$(\xi, \theta)$から$\tilde{Y}_i(x)$の内部表現$\tilde{\eta}_i$を得る関数をγとする．関数γを決定するため，表3.2よりVeitch-Karnaugh図表を作り，図A 2.2に示す．

図A 2.2より，関数γは次式で与えられる．

$$\gamma(\xi, \theta, \eta_i) = \xi \wedge \eta_i \tag{A2.5}$$

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | X | 0 |
| η | * | 0 | X | 1 |

ξ　θ

図　A2. 2　　関数γのVeitch - Karnaugh 図表

（図中の＊は $Y_i(x) = 0$ となることを表わす）

(3)　式（3. 21），式（3. 22）の導出について

$F_i(x) = \mathop{\mathrm{H}}_{j=1}^{n} x_j^{\varepsilon_{ij}}$ とすると，次の命題が成り立つ

(i)
$$\begin{aligned}
[F_i(x) = 0] &\Longleftrightarrow (\exists j)[1 \le j \le n \wedge x_j^{\varepsilon_{ij}} = 0] \\
&\Longleftrightarrow (\exists j)[1 \le j \le n \wedge \varepsilon_{ij} = f] \\
&\Longleftrightarrow (\exists j)[1 \le j \le n \wedge (\sigma_i)_j = 0 \wedge (\tau_i)_j = 1] \\
&\Longleftrightarrow [(\neg\sigma_i) \wedge \tau_i = 0]
\end{aligned} \tag{A2.6}$$

ここに，$(\sigma_i)_j$ および $(\tau_i)_j$ はそれぞれ $\sigma_i$ の第 $j$ 桁および $\tau_i$ の第 $j$ 桁の値を表わす．

(ii)
$$\begin{aligned}
[F_i(x) = 1] &\Longleftrightarrow (\forall j)[1 \le j \le n \rightarrow x_j^{\varepsilon_{ij}} = 1] \\
&\Longleftrightarrow (\forall j)[1 \le j \le n \rightarrow \varepsilon_{ij} = t] \\
&\Longleftrightarrow [\sigma_i = 0 \wedge \tau_i = 0]
\end{aligned} \tag{A2.7}$$

ここに，→　は含意を表わす．

(4)　式（3. 25），式（3. 26）の導出について

$Y_i(x) = \mathop{\mathrm{H}}_{j=1}^{n} x_j^{\delta_{ij}}, \tilde{Y}_i(x) = \mathop{\mathrm{H}}_{j=1}^{n} x_j^{\tilde{\delta}_{ij}}$ とすると，次の命題が成り立つ．

(i)
$$\begin{aligned}
[\tilde{Y}_i(x) = 0] &\Longleftrightarrow (\exists j)[1 \le j \le n \wedge \delta_{ij} = 1 \wedge c_j = 0] \\
&\Longleftrightarrow (\exists j)[1 \le j \le n \wedge (\eta_i)_j = 1 \wedge (\xi)_j = 0 \wedge (\theta)_j = 0] \\
&\Longleftrightarrow [(\neg\xi) \wedge (\neg\theta) \wedge \eta_i \ne 0] \\
&\Longleftrightarrow [\neg(\xi \vee \theta) \wedge \eta_i \ne 0]
\end{aligned} \tag{A2.8}$$

(ii)
$$\begin{aligned}
[\tilde{Y}_i(x) = 1] &\Longleftrightarrow (\forall j)[1 \le j \le n \rightarrow \tilde{\delta}_{ij} = 0] \wedge [\tilde{Y}_i(x) \ne 0] \\
&\Longleftrightarrow [\neg(\xi \vee \theta) \wedge \eta_i = 0] \wedge [\tilde{\eta}_i = 0]
\end{aligned} \tag{A2.9}$$

付録 3 （第 4 章の付録）

## 積項間の含意の関係の判定について

4.3 節に示した，目的関数の下界を評価する手続き（手続き 4.1 ）では，与えられた 2 つの積項 $Y_i(x)$，$Y_j(x)$ の間に，$Y_i(x) \rightarrow Y_j(x)$ の関係が成り立つか否かの判定を行なっている．式（ 4.5 ）で与えられる擬似ブール関数 $y(x)$ を構成する積項 $Y_i(x)$ は，変数の否定を含んでいない．よって，$Y_i(x)$, $Y_j(x)$ に含まれる変数の集合をそれぞれ $K_i$，$K_j$ とすると，次の関係が成り立つ．

$$[Y_i(x) \rightarrow Y_j(x)] \Longleftrightarrow [K_i \supseteq K_j] \tag{A3.1}$$

ここに，$\Longleftrightarrow$ は対等を表わす．

第 3 章と同じように，各 $Y_i(x)$ に次式で定義される整数 $\eta_i$ を対応させる．

$$\eta_i \triangleq \sum_{j \in K_i} 2^{j-1} \tag{A3.2}$$

の 2 進数表現を $\boldsymbol{\eta}_i$ で表わす．また，2 進数の各桁（ビット）ごとの論理和，否定をそれぞれ，$\vee$, $\neg$ で表わすと，式（ A 3.1 ）の右辺は，次式で置きかえられる．

$$[K_i \supseteq K_j] \Longleftrightarrow [(\neg \boldsymbol{\eta}_j) \vee \boldsymbol{\eta}_i = 1] \tag{A3.3}$$

ここに，1 は，2 進表現の全ての桁が 1 であるような整数を表わす．さらに，2 進数の各桁（ビット）ごとの含意を $\rightarrow$ で表わすと，

$$(\boldsymbol{\eta}_j \rightarrow \boldsymbol{\eta}_i) = ((\neg \boldsymbol{\eta}_j) \vee \boldsymbol{\eta}_i) \tag{A3.4}$$

であることから，結局，次式を得る．

$$[Y_i(x) \rightarrow Y_j(x)] \Longleftrightarrow [\boldsymbol{\eta}_j \rightarrow \boldsymbol{\eta}_i = 1] \tag{A3.5}$$

上記のようにして，積項間の含意の関係は，計算機のビット処理機能を用いて容易に実現される．

付録 4（第5章の付録）

(1) 定理 5.1 の証明について

(i) 式（5.11）が成り立つことは自明である．

(ii) 式（5.12）の対偶を考えて，

$$\sum_{i=1}^{k-1} t_{(i)} \geq m \quad \text{if} \quad C(y,\mathrm{F}) < k \tag{A4.1}$$

を示すことにする．

まず，

$$C(y,\mathrm{F}) = r < k \tag{A4.2}$$

と仮定する．

次に，$S^*(y,\mathrm{F})$ の任意の要素を，

$$\mathrm{x}^* = (x_1^*, x_2^*, \cdots, x_n^*) \tag{A4.3}$$

とする．式（A 4.2）より，適当な 1 − 1 写像，

$$\pi: \{j \mid 1 \leq j \leq n\} \rightarrow \{j \mid 1 \leq j \leq n\} \tag{A4.4}$$

が存在して，次式が成り立つことは明らかである．

$$\begin{cases} x_{\pi(j)} = 1 & \text{for } 1 \leq j \leq r \\ x_{\pi(j)} = 0 & \text{for } r+1 \leq j \leq n \end{cases} \tag{A4.5}$$

ここで，式（5.1）のブール関数 F(x) に対して，変数 $x_j$ が含まれる項 $\mathrm{F}_i$ の添字の集合を $T_j$ で表わす．すなわち，

$$T_j \triangleq \{i \mid 1 \leq i \leq m,\ a_{ij} = 1\} \tag{A4.6}$$

集合 $T_j$ について，次式が成り立つ．

$$\left| \bigcup_{j=1}^{r} T_{\pi(j)} \right| \leq \sum_{i=1}^{r} |T_{\pi(j)}| \tag{A4.7}$$

また，$T = (t_{(1)}, t_{(2)}, \cdots, t_{(n)})$ で，

$$t_{(j)} \geq t_{(j+1)} \quad \text{for } 1 \leq j \leq n-1 \tag{A4.8}$$

であることから，次式が成り立つ．

$$\sum_{j=1}^{r} |T_{\pi(j)}| = \sum_{j=1}^{r} t_{\pi(j)} \leq \sum_{j=1}^{r} t_{(j)} \tag{A4.9}$$

さらに，$\mathrm{x}^*$ が $S^*(y,\mathrm{F})$ の要素であることから，次式が成り立つ．

$$\left| \bigcup_{j=1}^{r} T_{\pi(j)} \right| = m \tag{A4.10}$$

式（A 5.6），式（A 5.8），式（A 5.9）より，次式を得る．

$$\sum_{j=1}^{r} t_{(j)} \geq m \tag{A4.11}$$

Q.E.D.

(2) 定理5.2の証明について

(i) 式(5.16)の証明

式(5.1)，式(5.2)で与えられるブール関数 F(x) は，変数の否定を含まない．よって，$F(x_j=0), 1 \le j \le n$ の項数は，吸収律等による簡約化を行なわないとすると，F(x) の項数 $m$ に等しい．

F(x) に対する $T$ を，

$$T = (t_{(1)}, t_{(2)}, \cdots, t_{(n)}),$$
$$t_{(j)} \ge t_{(j+1)} \quad \text{for } 1 \le j \le n-1 \tag{A4.12}$$

とすると，変数 $x_j$ のえらばれ方より，次式が成り立つ．

$$t_j^* = t_{(1)} \tag{A4.13}$$

$F(x_j^* = 0)$ に対する $T$ を $T'$，$F(x_j=0)$ に対する $T$ を $T''$ とする．ここに，$T', T''$ を次のように表わす．

$$T' = (t'_{(1)}, t'_{(2)}, \cdots, t'_{(n-1)}) \tag{A4.14}$$

$$T'' = (t''_{(1)}, t''_{(2)}, \cdots, t''_{(n-1)}) \tag{A4.15}$$

$F(x_j = 0)$ での変数 $x_j$ 以外の変数に対する $t_j$ の値は，F(x) での $t_j$ の値と同じであることから，$t'_{(j)}$，$t''_{(j)}$ に関して，次式が成り立つ．

$$t'_{(j)} = t_{(j+1)} \quad \text{for } 1 \le j \le n-1 \tag{A4.16}$$

$$t''_{(j)} \le t_{(j+1)} \quad \text{for } 1 \le j \le n-1 \tag{A4.17}$$

よって，次式が成り立つ．

$$t''_{(j)} \le t'_{(j)} \quad \text{for } 1 \le j \le n-1 \tag{A4.18}$$

一方，

$$k' = L(y(x_j^* = 0),\ F(x_j^* = 0)) \tag{A4.19}$$

$$k'' = L(y(x_j = 0),\ F(x_j = 0)) \tag{A4.20}$$

とすると，手続き5.1の性質より，次式が成り立つ．

$$\sum_{j=1}^{k'-1} t'_{(j)} < m \le \sum_{j=1}^{k'} t'_{(j)} \tag{A4.21}$$

$$\sum_{j=1}^{k''-1} t''_{(j)} < m \le \sum_{j=1}^{k''} t''_{(j)} \tag{A4.22}$$

ところが，式(A4.18)より，明らかに，

$$\sum_{j=1}^{r} t'_{(j)} \le \sum_{j=1}^{r} t''_{(j)} \quad \text{for } 1 \le r \le n-1 \tag{A4.23}$$

であるから，次式が成り立つ．

$$k' \le k''$$

すなわち，

$$L(y(x_j^* = 0), F(x_j^* = 0)) \leq L(y(x_j = 0), F(x_j = 0)), \text{ for } 1 \leq j \leq n \qquad (A4.24)$$

が成り立つ．Q.E.D.

(ii) 式（5.17）の証明

式（5.1），式（5.2）で与えられるブール関数 F(x) に対して，$F(x_j = 1)$の項数を$m_j$で表わすことにすると，$m_j$ は次式で与えられる．

$$m_j = m - t_j \qquad (A4.25)$$

一方，変数 $x_j^*$ のえらばれ方より，

$$t_j^* = t_{(1)} \geq t_{(j)} \quad \text{for } 1 \leq j \leq n \qquad (A4.26)$$

が成り立つ．

また，$U(y, F)$ の定義により，

$$U(y(x_j = 1), F(x_j = 1)) = m_j \qquad (A4.27)$$

である．式（A 4.25），式（A 4.26）より，明らかに，

$$m_j^* \leq m_j \quad \text{for } 1 \leq j \leq n \qquad (A4.28)$$

であることから，次式を得る．

$$U(y(x_j^* = 1), F(x_j^* = 1)) \leq U(y(x_j = 1), F(x_j = 1)), \text{ for } 1 \leq j \leq n \qquad (A4.29)$$

Q.E.D.

付録５（第６章の付録）

(1) 定理６．１の証明について

PDFA の実行途中での，未解決な部分問題の集合 $I$を，次式で与えられる，互いに排反な $p$ 個の部分集合 $I_1, I_2, \cdots, I_p$ に分割する．

$$I_i \triangleq \{P_j \mid P_j \in I, P_j \text{は処理装置} PU_i \text{によって処理された部分問題の直接の子孫}\} \tag{A5.1}$$

ただし，アルゴリズムの初期状態では $I=\{P_0\}$ であるが，この場合には $P_0 \in I_1$ とすれば，一般性を失なわない．

次に，$I_i$ 中の，レベル $j$ にある部分問題の個数を $N_{ij}$ で表わす．

PDFA では，各処理装置が処理する部分問題は，その部分問題がえらばれた時点で，$I$ 中では最もレベルが大きい部分問題である．よって $N_{ij}$ に関して，次式が成り立つ．

$$N_{ij} \leq \begin{cases} k & ; \text{if } j = n \ \vee \ N_{i,(j-1)} = 0 \\ k-1 & ; \text{if } j \neq n \ \wedge \ N_{i,(j-1)} \neq 0 \end{cases} \tag{A5.2}$$

よって，

$$\begin{aligned} |I_i| &= \Sigma_{j=1}^{n} N_{ij} \\ &\leq (k-1) \times (n-1) + k \\ &= (k-1) \times n + 1 \end{aligned} \tag{A5.3}$$

さらに，$I_i, (1 \leq i \leq p)$が互いに排反な集合であることにより，次式を得る．

$$\begin{aligned} |I| &= \Sigma_{i=1}^{p} |I_i| \\ &\leq p \times \{(k-1) \times n + 1\} \end{aligned} \tag{A5.4}$$

Q.E.D.

(2) $Z_{(p+1)} < Z_p$ であるための必要十分条件について

アルゴリズムの実行途中での，未解決な部分問題の集合を，

$$I \triangleq \{P_i \mid 1 \leq i \leq q\} \tag{A5.5}$$

とする．ここに，$P_i$に関して，次の２つの条件のいずれかが成り立つ．

(i) $level(P_i) < level(P_j)$ if $i < j$ (A5.6)

(ii) $level(P_i) = level(P_j) \ \wedge$
$h(P_i) < h(P_j)$ if $i < j$ (A5.7)

次に，関数 $select_p : I \rightarrow P(I)$を次式で定義する．（$P(I)$ は集合$I$のべき集合を表わす）

$$select_p(I) \triangleq \begin{cases} I & ; \text{if } |I| \leq p \\ \{P_i \mid 1 \leq i \leq p\} & ; \text{if } |I| > p \end{cases} \tag{A5.8}$$

次に，関数 $sons : P(V) \to P(V)$ を次式で定義する．

$$sons(I) \triangleq \{P_j \mid (P_i, P_j) \in E,\ P_i \in I\} \tag{A5.9}$$

さらに，$V$の部分集合の列 $trail_p(P_0)$ を次式で定義する．

$$trail_p(P_0) \triangleq Q_0^p\ Q_1^p\ \cdots\ Q_n^p \tag{A5.10}$$

ここに，$Q_i^p$ は，次式で与えられる．

$$Q_i^p \triangleq \begin{cases} \{P_0\}; & \text{if } i = 0 \\ select_p(sons(Q_{(i-1)}^p)); & \text{if } i \geq 1 \end{cases} \tag{A5.11}$$

$p$-PDFA によって，はじめにみつけられる可能解による目的関数の値 $Z_p$ は，次式で定義される．

$$Z_p \triangleq \min\{cost(P_i) \mid P_i \in Q_n^p\} \tag{A5.12}$$

次に，木$B$の根 $P_0$ から節点$P_i$ に至るパス上の節点の列を，

$$path(P_i) = P_{i_0}\ P_{i_1}\ \cdots\ P_{i_l} \tag{A5.13}$$

で表わす．ここに，

$$P_{i_0} = P_0,\ P_{i_l} = P_i \tag{A5.14}$$

である．

このとき，$Z_p$ に関して，次の定理が成り立つ．

〔定理A 5.1〕

$$Z_{(p+1)} < Z_p \tag{A5.15}$$

であるための必要十分条件は，

(i) $cost(P_i) < Z_p$ (A5.16)

をみたす $P_i$ が $Q_n^{p+1}$ 中に存在し，かつ，この $P_i$ について，

(ii) $h(P_{i_l}) > \max\{h(P_u) \mid P_u \in Q_l^i\}$ (A5.17)

をみたす$P_{i_l}$ が $path(P_i)$ 中に存在することである．

〔証明〕（十分条件）

$P_i \in Q_n^{p+1}$および，式（A 5.16）より，明らかに，

$$Z_{(p+1)} \leq cost(P_i) < Z_p \tag{A5.18}$$

が成り立つ．

（必要条件）

$Q_n^{p+1}$ の部分集合$Q'$を次式で与える．

$$Q' \triangleq \{P_i \mid P_i \in Q_n^{p+1},\ cost(P_i) < Z_p\} \tag{A5.19}$$

このとき，$Z_{(p+1)} < Z_p$ であることから，明らかに$Q' \neq \phi$ である．

次に，$Q'$ の全ての要素について

$$h(P_{i_l}) \leq \max\{h(P_u) \mid P_u \in Q_l^p\},\ \text{for}\ 1 \leq {}^{\forall}l \leq n \tag{A5.20}$$

と仮定すると，$Z_{(p+1)} \geq Z_p$ となることを示す．

仮定および式（A 5.8）より，次式を得る．

$$P_{i_l} \in Q_l^p = select_p(sons(Q_{l-1}^p)),\ \text{for}\ {}^{\forall}P_{i_l} \in path(P_i),\ {}^{\forall}P_i \in Q' \tag{A5.21}$$

よって，次式を得る．

$$P_i \in Q_n^p,\ \text{for}\ {}^{\forall}P_i \in Q' \tag{A5.22}$$

式（A 5.22）より，

$$cost(P_i) \geq Z_p,\ \text{for}\ {}^{\forall}P_i \in Q' \tag{A5.23}$$

しかるに，式（A 5.19）より，

$$Z_{(p+1)} = \min\{\ cost(P_i) \mid P_i \in Q'\} \tag{A5.24}$$

であるから，式（A 5.23）および式（A 5.24）より，

$$Z_{(p+1)} \geq Z_p \tag{A5.25}$$

を得る．これは，$Z_{(p+1)} < Z_p$ と矛盾する．

この矛盾は式（A 5.20）の仮定によりもたらされた．

従って，

$$h(P_{i_l}) > \max\{\ h(P_u) \mid P_u \in Q_l^p\ \} \tag{A5.26}$$

を満足する $P_{i_l}$ が，$path(P_i), P_i \in Q'$ 中に存在しなければならない．Q.E.D.

付録６（第７章の付録）

(1) 定理 7. 1 の証明について

(i) まず，$p=1$の場合，手続き 5. 1 の Step 1 では常に$Q=\phi$であることを，手続き 5. 1 が実行される回数についての帰納法によって示す．

手続き 7. 1 を$i$回目に実行を行なう時の$Q$を$Q^{(i)}$で表わす．$Q$は，手続き 7. 1 の最初の実行に先だって$\phi$に初期化されているので，明らかに，

$$Q^{(1)} = \phi \tag{A6.1}$$

である．次に，$i \geq 1$の時，$Q^{(i)} = \phi$ならば，$Q^{(i+1)} = \phi$となることを示す．

手続き 7. 1 を$i$回目に実行する時，仮定により，$Q^{(i)} = \phi$であるので，手続き 5. 1 を$i$回目に実行する時には，Step 2 が実行される．このとき，$p=1$ であることから，Step 2 の実行が終了した時点では，$|Q|=1$である．次に，Step 3が実行されると，$Q$の要素の個数は 1 だけ減じられるので，再び$Q=\phi$となる．アルゴリズム 7. 1 およびアルゴリズム 7.2 では，手続き 7. 1 以外の場所で$Q$に対する操作は行なわれないので，手続き 7. 1 が $i+1$ 回目に実行される時にも$Q^{(i+1)}=\phi$ であることは明らかである．

よって，任意の$i \geq 1$について，

$$Q^{(i)} = \phi \tag{A6.2}$$

であることが知られる．

上で述べたように，$p=1$ の場合には，手続き 7. 1 が実行されるたびごとに，Step 2 で $s_D(I)$ の値が$Q$に代入され，その結果 Step 3 で$P_i^*$には，常に，$s_D(I)$ の値が代入されることになる．よって，$p=1$ の場合，$pdfs$ は $s_D$ と等価である．Q.E.D.

(ii) まず，$p \geq k^n$の場合，手続き 7. 1 によって選択される部分問題 $P_i$ は，常に，$I$ 中で最もレベルが小さい部分問題であることを示す．

手続き 7. 1 のStep2の$i$回目の実行が終了した直後の$Q$の値を$Q^{(i)}$で，また，その時の$I$ の値を$I^{(i)}$で表わす．

アルゴリズム 7. 1 およびアルゴリズム 7. 2 が実行されている任意の時点で，$I$の要素の個数は，明らかに，たかだか$k^n$ 個である．よって，任意の $i \geq 1$ について，次式が成り立つ．

$$P_j \in Q^{(i)} \quad \text{iff} \quad P_j \in I^{(i)} \tag{A6.3}$$

次に，任意の$i \geq 1$について，次式が成り立つことを示す．

$$l(P_j) = i - 1 \quad \text{for } {}^{\forall}P_j \in I^{(i)} \tag{A6.4}$$

まず，$i = 1$ の時，明らかに，

$$I^{(1)} = \{ P_0 \} \tag{A6.5}$$

$$l(P_0) = 0 \tag{A6.6}$$

である．次に，$i \geq 1$について，式（A 6.4）が成り立つと仮定する．仮定より，$I^{(i)}$の任意の要素 $P_j$ について，$l(P_j) = i-1$ である．一方，アルゴリズム 7.1 およびアルゴリズム 7.2 の Step5 で $P_j$ を分割して得られる部分問題 $P_{j_1}, P_{j_2}, \cdots, P_{j_k}$ について，

$$l(P_{j_r}) = l(P_j) + 1 = i \tag{A6.7}$$

が成り立つので，$I^{(i)}$ の全ての要素が分割された時点で，$I$ 中の要素のレベルは全て $i$ である．また，その時，$Q = \phi$ となるので，手続き 7.1 の Step2 が実行されることになる．よって，

$$l(P_j) = i \quad \text{for } {}^{\forall}P_j \in I^{(i+1)} \tag{A6.8}$$

が成り立つことになる．よって，任意の $i \geq 1$について式（A 6.4）が成り立つ．

手続き 7.1 の Step 2によって$Q^{(i)}$が構成されてから次に $Q^{(i+1)}$ が構成される直前までは，手続き 7.1 は $Q^{(i)}$ の要素を先頭から 1 個ずつ出力する．よって，手続き 7.1 によって選択される部分問題 $P_i{}^*$ は，常に，$I$ 中で最もレベルが小さい部分問題である．

しかも，$Q^{(i)}$ の各要素は，関数 $h$ の値の小さい順に並べられているから，手続き 7.1 によって選択される部分問題 $P_i{}^*$について，次式が成り立つことは明らかである．

$$h(P_i{}^*) = \min\{h(P_j) \mid P_j \in I, \quad l(P_j) = \min\{l(P_r) \mid P_r \in I\}\} \tag{A6.9}$$

よって，式（A 6.9），式（7.9）より，$p \geq k^n$ の場合，$pdfs$ は $s_D$ と等価である．

Q.E.D.

(2) 定理 7.3 の証明について

(i) $w = k$ のとき，式（7.27）が成り立つとすると，式（7.26）より，次式が成り立つ．

$$S = \{ i_{max} \} \tag{A6.10}$$

すなわち，

$$l(P_i) = i_{max}, \text{ for } {}^{\forall}P_i \in I \tag{A6.11}$$

であるから，関数 $s_D$ の定義（式（7.6）および式（7.1））および，手続き 7.2 の性質により，

$$slim(I) = s_D(I) \tag{A6.12}$$

が成り立つことは明らかである．Q.E.D.

(ii) アルゴリズムの実行途中での$I$の相異なる任意の2つの要素を$P_i$, $P_j$とすると，$P_i$と$P_j$は，互いに，先祖－子孫の関係にはない．一方，$V$の部分集合の族の中で，上記の条件をみたす部分集合の族を$\Gamma$とすると，明らかに次式が成り立つ．

$$|G| \leq k^n \quad \text{for } {}^{\forall}G \in \Gamma \tag{A6.13}$$

よって，$I \in \Gamma$であることから，

$$|I| \leq k^n \tag{A6.14}$$

が成り立つ．

よって，$w \geq k^n + k$であれば，手続き7.2のStep 2で，全ての$j$について，

$$N(j+1) \leq w - k \tag{A6.15}$$

の条件が満たされるので，

$$slim(I) = s_{BB}(I) \tag{A6.16}$$

が成り立つ．Q.E.D.

### (3) 定理7.4の証明について

手続き7.2によって$I$中からえらばれる部分問題を$P_i^*$とする．$P_i^*$のレベルを$j$とすると，手続き7.2のStep 2の条件から，$I$のレベル$j+1$に属す部分問題の個数$N(j+1)$は次式を満足している．

$$N(j+1) \leq w - k \tag{A6.17}$$

一方，アルゴリズム7.1およびアルゴリズム7.2のStep5で$P_i^*$を分割して得られる$k$個の部分問題のレベルは，全て$j+1$である．よって，これらの$k$個の部分問題を$I$に加えた後の$N(j+1)$は，次式を満たす．

$$N(j+1) \leq w \tag{A6.18}$$

よって，アルゴリズムが実行されている任意の時点で，次式が成り立つ．

$$N(j) \leq w, \text{ for } 1 \leq {}^{\forall}j \leq n \tag{A6.19}$$

また，アルゴリズムの性質から，$N(j) > 0$をみたす$j$, $(1 \leq j \leq n)$が存在すれば$N(0) = 0$であるから，アルゴリズムが実行されている任意の時点で，

$$\begin{aligned} |I| &= \Sigma_{j=0}^{n} N(j) \\ &\leq w \times n \end{aligned} \tag{A6.20}$$

が成り立つ．Q.E.D.