北海道大学工学部研究報告
第140号（昭和63年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No. 140 (1988)

代理制約式を用いた陰的列挙法の効率化に関する数値実験

姉崎　淳　大柳　俊夫　加地　郁夫

（昭和62年12月26日受理）

Computational Experience with Improving on Efficiency of Implicit Enumeration Method with Surrogate Constraints

Jun ANEZAKI, Toshio OHYANAGI and Ikuo KAJI

(Recieved December 26, 1987)

Abstract

In this paper an implicit enumeration method with surrogate constraints namely an algorithm for solving linear integer programming problems with zero-one variables is introduced and two heuristics for this algorithm are proposed. One heuristic suggests the opportunity of making surrogats constraints and the other suggests the selection of variables during branching operation.

Then a way of testing such heuristics is presented and its computational experience is given with consideration.

1．はじめに

整数計画問題の中で変数が0と1の二値しか取らない問題は0－1整数計画問題と呼ばれ，この問題に定式化される問題として，割り当て問題，施設配置問題などがある。この問題の解法はいろいろ提案されており，その中の一つにE. Balasによる陰的列挙法がある[1),2)]。この方法は分枝限定法を基礎とする探索的手法で，その探索は，解(実行可能解と実行不可能解)を陽的に列挙する分枝操作と陰的に列挙する限定操作により進められる。このとき，代理制約式と呼ばれる新しい制約式を加えながら探索を進めると，探索効率が高められることが知られている[3)]。

本報告は，一般の0－1整数計画問題を，代理制約式を用いた陰的列挙法を使ってより高速に解くためのヒューリスティックスの提案と，そのヒューリスティックスを実験的に検証するために行った数値実験について述べるものである。

まず代理制約式を用いた陰的列挙法を説明する。次に，今回提案する代理制約式作成の時機と分枝操作における変数の選択基準に関するヒューリスティックスについて説明し，それらを実験的に検証するための方法について述べる。また，数値実験で用いるテスト問題として，必ず最適解が存在するという性質を持つランダム0－1整数計画問題の発生方法を提案する。

そして最後に，今回提案したヒューリスティックスの妥当性を示す数値実験結果を示し，それについて考察する。

情報工学科　システム工学講座

2．代理制約式を用いた陰的列挙法

2.1 陰的列挙法のアルゴリズム

一般に0－1整数計画問題は，

$$\text{目的関数}\quad \sum_{j=1}^{n} c_j x_j \rightarrow \text{最小化}$$

$$\text{制約条件}\quad \sum_{j=1}^{n} a_{ij} x_j \geqq b_i \quad (i=1,\ 2,\ \cdots\cdots,\ m) \qquad (2.1)$$

$$x_j \in \{0,\ 1\} \quad (j=1,\ 2,\ \cdots\cdots,\ n)$$

と定式化され，この問題の解法の1つに陰的列挙法がある。陰的列挙法は，分枝限定法を基礎とする探索的手法で，最適解を探索する過程ですべての解(実行可能解と実行不可能解)を陰的，あるいは陽的に列挙する。その探索経路は図1に示すように2分木となる。この2分木を列挙木と呼ぶ。列挙木で，各ノードは陽的に列挙された1つの解を表わし，ノードとして現われない解は陰的に列挙された解となる。陰的列挙法では(2.1)の2^n個の解の大部分を陰的に列挙するので，効率的に最適解を求めることができる。

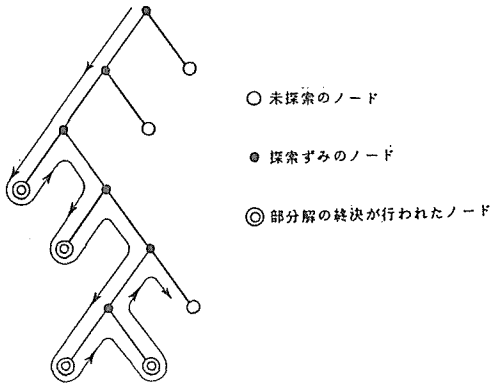


図1　探索経路の図式(列挙木)

最適解の探索は，変数の値を1つずつ0または1に固定して部分解を生成する分枝操作と，ある部分解に対して，もはや探索を続ける必要がないとわかった時に後戻りを行う限定操作の2つからなる。分枝操作において0または1に固定される変数を分枝変数と呼ぶ。また，分枝によって値が0または1に固定された変数とその変数の値の集合を部分解と呼び，部分解に含まれない変数を自由変数と呼ぶ。そして，ある部分解に対して自由変数の値を0または1とした解の集合を部分解の完備解と呼び，さらに，ある部分解に対して後戻りを行うことを部分解の終決と呼ぶ。部分解の終決は次の2つの場合に行われる。

(1)実行可能性による終決

ある部分解のすべての自由変数の値を0とした解が実行可能である場合には，部分解の完備解の中で，目的関数値が最小の解を発見したことになるので，探索は終決し後戻りが行われる。この際，その解のみを陽的に列挙し他の完備解は陰的に列挙される。

(2)実行不可能性による終決

ある部分解の完備解がすべて実行不可能である場合，あるいは実行可能解が存在してもその目的関数値が現在得られている実行可能解のうち目的関数値を最小にする解，すなわち暫定解の目的関数値よりも小さくないことが判明した場合，探索は終決し後戻りが行われる。

陰的列挙法では(1)，(2)を判断するためのテストが繰り返し行われる。

2.2 代理制約式の作成

陰的列挙法の探索過程では，いくつかの変数の値を1または0に固定して得られる部分問題を考える。第k番目に生成される部分解に対して部分問題は以下のように定義される。

$$\text{目的関数}\quad \sum_{j \in F^k} c_j x_j \rightarrow \text{最小化}$$

$$\text{制約条件}\quad \sum_{j\in F^k} a_{ij}x_j \geqq b_i^k \qquad (i \in M) \tag{2.2}$$
$$x_j \in \{0,\ 1\}$$

ただし，S^k は部分解に含まれる変数の添字集合，F^k は自由変数の添字集合，M は制約式の添字集合，そして

$$b_i^k = b_i - \sum_{j\in S^k} a_{ij}\, x_j$$

である。

代理制約式とはこの部分問題の各制約式に重み係数を与え，辺々加えあわせて作られる新たな制約式である。これを用いると，実行不可能性を判断するためのテストの効果が著しく高まることが知られている。代理制約式を作る方法はいろいろと提案されているが，ここでは Balas の方法を採用する。その方法は，重み係数を計算するために(2.2)式の部分問題の変数を $0 \leqq x_j \leqq 1$ とした連続緩和問題の双対問題

$$\text{目的関数}\quad \sum_{i\in M} u_i\, b_i^k - \sum_{j\in F^k} v_j \rightarrow \text{最大化}$$
$$\text{制約条件}\quad \sum_{i\in M} u_i\, a_{ij} - v_j \leqq c_j \tag{2.3}$$
$$u_i \geqq 0,\ i \in M,$$
$$v_j \geqq 0,\ j \in F^k$$

を解く。その結果つぎの 3 つの場合が生ずる。

(1)最適解が得られ，かつ最適値が暫定解の目的関数値より小さくなるか等しい場合

最適解 u^* を重み係数とし，算出された代理制約式を部分問題に追加して探索を続ける。

(2)最適解が得られ，最適値が暫定解の目的関数値より大きくなる場合

これ以上深く探索しても暫定解を更新できないので現在の部分解は終決する。

(3)非有界な解となる場合

双対問題が非有界であるから部分問題の緩和問題は解なしとなり，現在の部分問題は終決する。

このように重み係数の計算結果は，部分解の終決を判断するためのテストの一つとしても利用できる。

3．実験方法

3.1 効率化のための要点…2つのヒューリスティックス

陰的列挙法における探索を効率良く行うためには，探索中に生成される列挙木の拡大を抑えなければならない。このためには，早い段階で部分解の終決を行うことが必要である。

2.2 で述べた代理制約式を用いることにより，早い段階で部分解の終決を判断できることが知られている。しかし，代理制約式を作成するためには線形計画問題(Linear Programming, 以下 LP)を解く必要があり，そのためにかなり多くの計算を必要とする。従って，最適解を求めるまでに要する計算時間は LP を解く回数とそれぞれの LP のサイズに左右されることになる。そこで，本節 3.2 では探索終了までに解かれる LP の回数を減らすことと，解かれる LP のサイズを小さくすることを狙ったヒューリスティックを提案する。そのヒューリスティックとは，代理制約式の作成にある間隔を持たせることである。そして，その効果を調べる実験方法を示す。

また，早い段階で目的関数値の小さい暫定解を得た場合には，暫定解を更新できないという判

断による部分解の終決が早い段階で行われる。このとき，解の探索はヒューリスティックに行われるので，暫定解が得られる時期，およびその目的関数値は分枝変数の選択基準に依存することになる。そこで，本節3.3では第2のヒューリスティックとして，早い段階で目的関数値の小さい暫定解を得るために，複数個の選択基準を組み合わせて用いる方法を提案し，これを用いた場合の効果を調べる実験方法を示す。

なお，本実験では，プログラムの全実行時間からデータの入出力に要した時間を除いた時間を“実行時間”と呼ぶ。また，部分解を列挙した回数を“ステップ数”と呼び，これを列挙木の大きさの目安とする。

表1　25変数10制約式の問題に対するトレースの結果

	Case-1	Case-2	Case-3
探索終了までのステップ数	882	31979	29850
ＬＰの実行回数	859	–	29827
実行時間 (sec)	787	1490	31280
部分解が終決した回数	394	14602	13616
（内　訳）			
実行可能解発見の回数	23	23	23
ＬＰの結果による後戻り	371	–	–
他のテストによる後戻り	0	14579	13593

3.2　代理制約式作成に関する実験方法

代理制約式を用いた陰的列挙法プログラムを作成し，25変数10制約式のデータに対して行った探索過程のトレース結果を表1のCase-1に示す。あわせて，代理制約式を作成しない場合の結果を表1のCase-2に示す。

この実験結果からLPの結果による後戻りを行った場合には代理制約式を作成しない場合に比べて約半分の実行時間で探索を終了していることがわかる。この理由は，今回のプログラムではLP計算が実行時間の約9割を占めているにもかかわらず，探索終了までのステップ数が約1/35になったからである。つまり，LPを解くことは他のテストに比べ多くの計算を必要とするが，その反面，実行不可能性による後戻りが列挙木の浅い段階で効果的に行われ，ステップ数が大幅に減少するのである。

さらに，代理制約式を作成するが，LPの結果による後戻りを行わない場合の結果を表1のCase-3に示す。この結果を代理作約式を作成しない場合と比較すると，ステップ数は減少しているにもかかわらず，LPを行う回数が多いために20倍以上の実行時間を要していることがわかる。

これらの実験結果から，代理制約式作成の際に解かれるLPの結果により後戻りの判断を行うことは，非常に有効な手段であることがわかる。

しかし，代理制約式を作成しても，その結果が後戻りの判断に利用されない場合も多い。そこで，実行時間を短くするためには代理制約式を毎回作成するのではなく，その作成になんらかの基準で間隔をあけ，LPの計算にかかる負担を減らすという方策が考えられる。

本実験では，『解の探索が列挙木のある一定間隔の深さに達した場合に代理制約式を作成し，作成しないステップにおいては最も近いステップで作成された代理制約式を用いる』という代理制約式の作成方針を提案し，その妥当性の検証を行う。具体的には，列挙木の深さに応じた間隔の設定として，

(1)探索が浅い段階における間隔を大きく設定する

(2)探索が深く進んだ段階における間隔を大きく設定する

という2つの場合を考えて数値実験を行う。(1)は，探索が浅い段階のLPのサイズが大きいことに着目し，LPの計算にかかる負担を減らすことを狙っている。しかしこの場合，探索が深く進み，探索終了までに要するステップ数が多くなる恐れがある。また(2)は，LPの結果による後戻りの効果を重視し，探索を深く進めないことを狙っている。しかしこの場合，サイズの大きなLPを多く解かなければならないので，実行時間に対する寄与が小さくなることが考えられる。

なお，自由変数の選択基準は3.3で述べる基準(1)を用いた。

3.3 分枝変数の選択基準に関する実験方法

分枝変数の選択はヒューリスティックに行われるが，一般的には0—1計画問題の現実的な意味合いから最適解においてはおそらく1であろう，あるいは0であろうと見当をつけることのできる変数がある場合には，そのような変数を分枝変数として選択するのが有効とされている。しかし，本実験で取り扱っているランダムな問題のように，変数の値に対する見当がつけられない場合にも，探索を効率的に行うことを目的とした種々の選択基準が提案されている。それらはおおよそ次の3つに分類することができる[4]。

(1)実行可能性に基づく選択基準

なるべく早い時期に問題の実行可能解を得るように分枝変数を選択する。具体的には，第kステップで生成される部分問題(2.2)式に対し，

$$\theta_t = \sum_{i=1}^{m} \max\{0,\ b_i^k - a_{it}\} \qquad (t \in F^k) \tag{3.1}$$

とし，

$$\theta_t^* = \min_{t \in F^k} \theta_t \tag{3.2}$$

に対応する自由変数 x_t^* を1に固定する。

(2)実行不可能性に基づく選択基準

なるべく早い時期に実行不可能となるように分枝変数を選択する。この方法は目的関数値の小さい暫定解が得られた段解において，列挙すべき部分解の個数を少なく抑えることを目標としている。具体的には部分問題(2.2)式に対して，

$$\rho_t = \sum_{i \in M^+} \max\{0,\ a_{it}\} \tag{3.3}$$

(M^+ : $b_i^k \leqq 0$ となる制約式の添字集合，$t \in F^k$)

とし，

$$\rho_t^* = \max_{t \in F^k} \rho_t \tag{3.4}$$

に対応する自由変数 x_t^* を0に固定する。

(3)目的関数の係数に基づく選択基準

実行可能解が得られた時にその目的関数値ができる限り小さくなるように分枝変数を選択する。この方法を用いると部分問題の解の目的関数値を小さくすることができる。具体的には，

$$c_t^* = \min_{t \in F^k} c_t \tag{3.5}$$

に対応する自由変数 x_t^* を1に固定する。

ここで，どの選択基準が最も有効であるかが問題となる。LPの結果の判断により後戻りを行う場合には，“暫定解の目的関数値を改良する実行可能解は存在しない”という判断による後戻りが，非常に多く行われることが予備的実験により経験的に確かめられている[5]ので，なるべく早い段階で暫定解を発見し，かつ得られた暫定解が十分小さくなるように分枝変数を選択する方策が有効であると考えられる。本実験では(1)と(3)の選択基準を組み合わせた新しい選択基準を提案し，その効果を調べる。

2つの選択基準の組み合わせは，(1)，(3)の θ と c に対して重み係数 α_1，α_2 を用い，

$$\mu_t = \alpha_1 \theta_t / m + \alpha_2 c_t \qquad (t \in F^k) \tag{3.6}$$

とする。そして

$$\mu_t^* = \min_{t \in F^k} \mu_t \tag{3.7}$$

に対応する自由変数 x_t^* を1に固定するという方法を取る。ここで，(3.7)式の第1項を制約式の本数 m で割ることにより，2つの基準のスケーリングを図っている。

実験は，重み係数の比 α_2/α_1 を変化させ，そのときのステップ数と実行時間の変化を調べる。なお，本実験では代理制約式作成の間隔を1としている。

3.4 テスト問題と実験環境について

実行時間を測定するためのテスト問題としてランダムな問題を用いる。この問題は，(2.1)式の a_{ij}，c_j として，

$0 \leqq L_a \leqq a_{ij} \leqq U_a$ （L_a：下限値，U_a：上限値）
$0 \leqq L_c \leqq c_j \leqq U_c$ （L_c：下限値，U_c：上限値）

を満足する範囲内で発生させた整数一様乱数を用いる。また，b_i の値としては，$0 \leqq p \leqq n$ を満たす乗数を設定し，

$$b_i = \frac{\sum_{j=1}^{n} a_{ij}}{n} p \quad (i=1, 2, \cdots, m) \tag{3.8}$$

とする。乗数 p の値を変えることにより，実行可能解を発見するまでの探索レベルの深さを変えることができる。本実験では a_{ij} と c_j の値の上限値と下限値の対(L_a, U_a, L_c, U_c)として，(0, 50, 15, 35)と(0, 50, 0, 50)を用い，p の値を $0.4\,n \leqq p \leqq 0.6\,n$ の範囲に設定する。この問題は変数の値をすべて1にすることにより実行可能となるので必ず最適解を持つ。

なお，実験で使用した機種は富士通 FM16βFD II で，言語は SW-Pro FORTRAN77 を用いた。

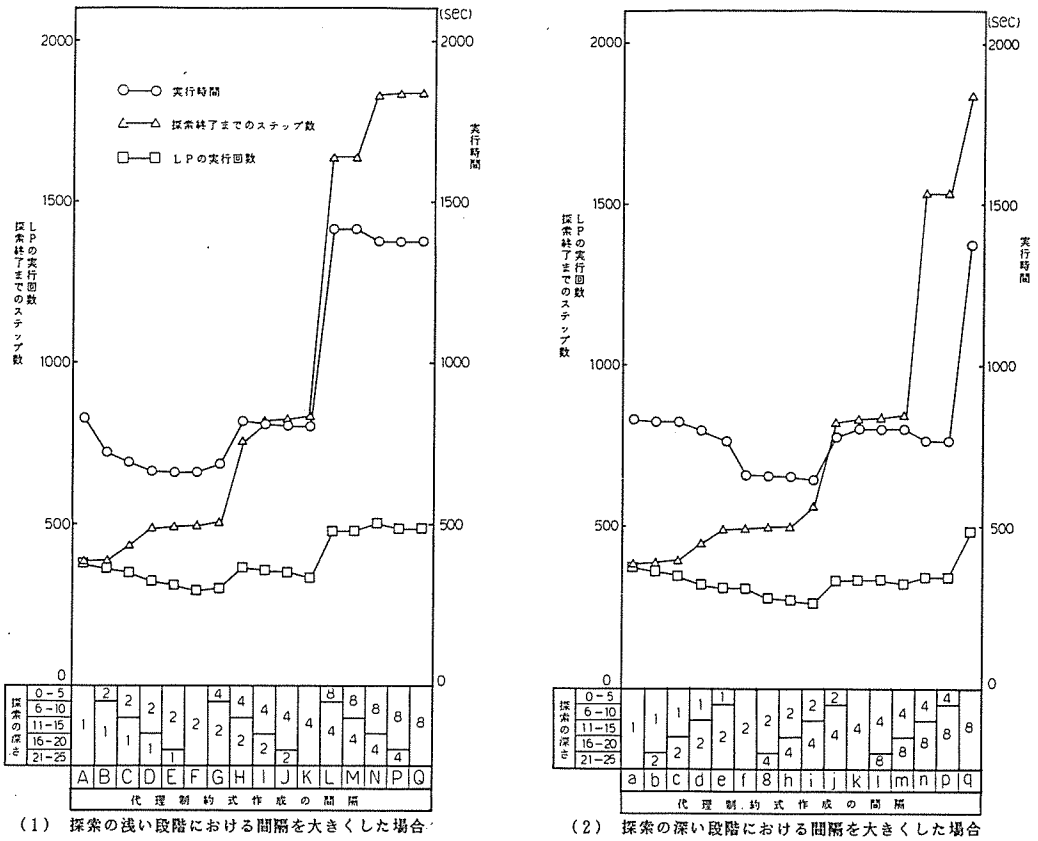
4．実験結果

4.1 代理制約式作成の間隔に関する数値実験結果

25変数15制約式の問題に対する実験結果を図2の(1)，(2)に示す。(1)は探索が浅い段階における代理制約式の作成間隔を大きくした場合で，(2)は，探索が深く進んだ段階における間隔を大きく設定した場合である。例えば，表中で代理制約式作成の間隔が A と a の場合にはすべての深さについて間隔を1(すべてのステップで代理制約式を作成することを意味する)と設定している。また，代理制約式作成の間隔が B の場合には，探索の深さが0から5のときは間隔を2，探索の深さが6から25のときは間隔を1と設定している。

この結果から，LP の実行回数と実行時間の増減の傾向は，ほぼ一致することがわかる。これは，実行時間の大部分を LP の計算が占めるからであると考えられる。また，代理制約式作成の間隔を大きくすることにより探索終了までのステップ数が増加することもわかる。これは，代理制約式の作成間隔を大きくした場合には，LP の結果による後戻りが探索の浅い段階で効果的に行われなくなるので，探索が不必要に深く進められるからであると考えられる。さらに，間隔を適切にした場合には，LP の実行回数が減少し実行時間が短縮されることもわかる。

間隔の設定が C と c の場合を比較すると，両者の LP の実行回数はほぼ一致しているが，C の



（1）　探索の浅い段階における間隔を大きくした場合

（2）　探索の深い段階における間隔を大きくした場合

図２　代理制約式の作成間隔に関する数値実験結果

実行時間が短くなっている。この理由は，探索が浅い段階の代理制約式の作成間隔を大きくした場合には，サイズの大きいLPを計算する負担が減るためだと考えられる。また，間隔の設定がＰとｐの場合を比較すると，ｐのLPの実行回数が少なくなっていて，その結果実行時間もまたｐの方が短くなっている。この理由は，探索の浅い代理制約式の作成間隔を小さくした場合には，LPの結果による探索の後戻りが探索の深い段階で有効に行われるからだと考えられる。

従って，（1）と（2）の方法にはそれぞれ一長一短があり，どちらかが有効であるかを決定することは難しい。

表２　代理制約式の作成間隔を２とした場合の実験結果

問題番号	0-1変数×制約式	代理制約式の作成間隔	ＬＰの実行回数	探索終了までのステップ数	実行時間 (sec)
1 a)	10×10	1	31	33	11
		2	18	35	8
2 b)	10×15	1	45	48	25
		2	24	49	14
3 b)	20× 5	1	148	153	97
		2	112	186	79
4 a)	20×10	1	141	149	146
		2	98	178	115
5 a)	20×15	1	390	412	299
		2	262	481	229
6 b)	25× 5	1	470	484	912
		2	303	572	224

問題番号につけたａ），ｂ）はランダム問題のタイプを示す
ａ）：$(L_a, U_a, L_c, U_c) = (50, 0, 50, 0)$
ｂ）：$(L_a, U_a, L_c, U_c) = (50, 0, 35, 15)$

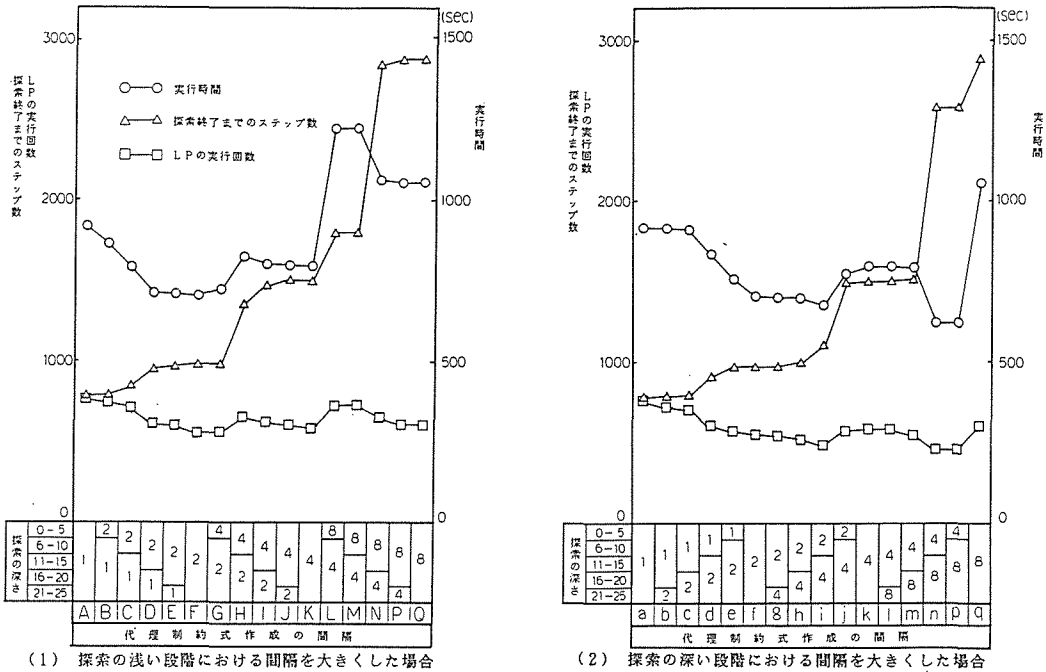


図3 代理制約式の作成間隔に関する数値実験結果

以上の結果から，実行時間を最も短くする代理制約式の作成間隔はLPのサイズとLPの結果による後戻りの効果の兼ね合いによって決められると考えられる。この問題に対しては，間隔を2とすることが比較的有効であると思われる。

次に，25変数10制約式の問題に対する実験結果を図3(1)，(2)に示す。この問題に対しても，代理制約式の作成に間隔を持たせることにより実行時間は短縮することがわかる。また，間隔を2とすることは有効であることもわかる。しかし，図3(2)に示すように，間隔の設定をnまたはpとすると，さらに実行時間が短縮することもある。

以上の結果から，あらゆる問題に対して最適な代理制約式の間隔を決定することは無理なようであるが，間隔を2とした場合に実行時間が短縮する可能性が高いと考えられる。そこで，間隔2に対する有効性を確かめるために，問題を変えて多くの実験を行った。実験結果の一部を表2に示すが，いずれの問題に対しても，間隔2を用いた場合にLPの実行回数が減少し，実行時間が短縮することがわかる。

4.2 分枝変数の選択基準に関する数値実験結果

25変数10制約式の問題に対する実験結果を図4に示す。この結果から，自由変数の選択基準として基準(1)と基準(3)を重み係数を用いて組み合わせて用いた場合には，基準(1)，(2)，(3)をそれぞれ単独に用いた場合に比べ，早い段階で最適解を発見できることがわかる。また，最適解を早い段階で発見した場合には，探索終了までに要するステップ数が減少し実行時間が短縮されていることもわかる。このことは，早い段階で目的関数値の小さい暫定解を得ることにより，“暫定解の目的関数値を更新する実行可能解は存在しない”という判断により後戻りが探索の浅い段階で行われることの裏付けとなる。

この問題に対しては，重み係数比 α_2/α_1 の値を1.0と設定した場合に効率良く探索を行っている。また，重み係数比 α_2/α_1 の値を1.0から小さくし，基準(1)の基準(3)に対する相対的な重みを増

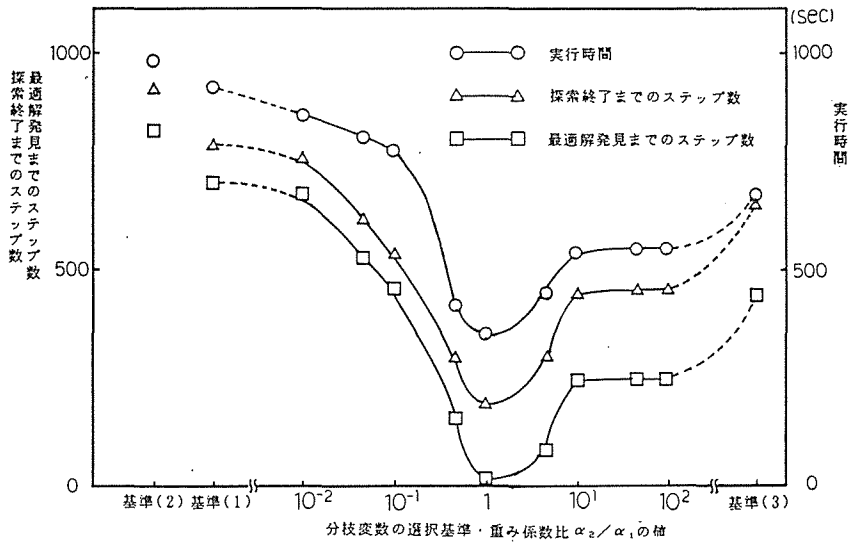


図4　分枝変数の選択基準に関する数値実験結果

すと，最適解発見までのステップ数，探索終了までのステップ数，および実行時間はいずれも増加する傾向にあり，さらに相対的な重みを増して $\alpha_2/\alpha_1=0.01$ とすると，基準(1)のみを用いた場合とほぼ同様な結果が得られる。

逆に，重み係数比 α_2/α_1 の値を 1.0 から大きくし，基準(3)の基準(1)に対する相対的な重みを増した場合にも，最適解発見までのステップ数，探索終了までのステップ数，および実行時間はいずれも増加する傾向にあり，今回の実験では行わなかったが α_2/α_1 の値を十分大きくすると，(3)のみを用いた場合とほぼ同様な結果が得られることが予想される。

以上の結果より“選択基準を単独で用いるよりも，組み合わせて用いた方が探索効率が良い”ことがわかる。

次に，制約式の本数を変更した 25 変数 5 制約式の問題に対する実験結果を図 5 に示す。この問題に対しても，分枝変数選択基準を組合せて用いることは有効であることがわかる。また，この場合にも重み係数比 α_2/α_1 の値を 1.0 とした場合に探索が効率良く行われており，選択基準に対してスケーリングを行ったことは有効であるといえる。

以上の結果から，重み係数比 $\alpha_2/\alpha_1=1.0$ を用いた分枝変数選択基準の重み付けが有効であると仮定して，さらに異なる問題に対して数値実験を行った。実験は，9 種類のサイズに対し，それぞれ乱数の種を変え 3 問ずつ問題を作成して行った。その結果，大多数の問題に対して重み係数比 $\alpha_2/\alpha_1=1.0$ を用いた場合に探索効率が最も良くなるという結果が得られた。しかし，表 3 の問題番号 1，2 に示すようにサイズの小さい問題に対しては分枝変数の選択基準を変えることによる効果が少ないことがわかった。また，表 3 の問題番号 4，5，6 に示すように，必ずしも最適解発見までのステップ数，あるいは探索終了までのステップ数は減少しないが，実行時間を短縮する問題があることもわかった。

4.3　その他の数値実験結果

これまでの実験結果から有効であることがわかった“代理制約式の作成間隔を 2 にする”および“分枝変数選択基準を組み合せる際の重み係数比 α_2/α_1 を 1.0 にする”を同時に用いた場合の実験結果を表 4 に示す。この結果から 2 つのヒューリスティックスを合わせて用いることによりさ

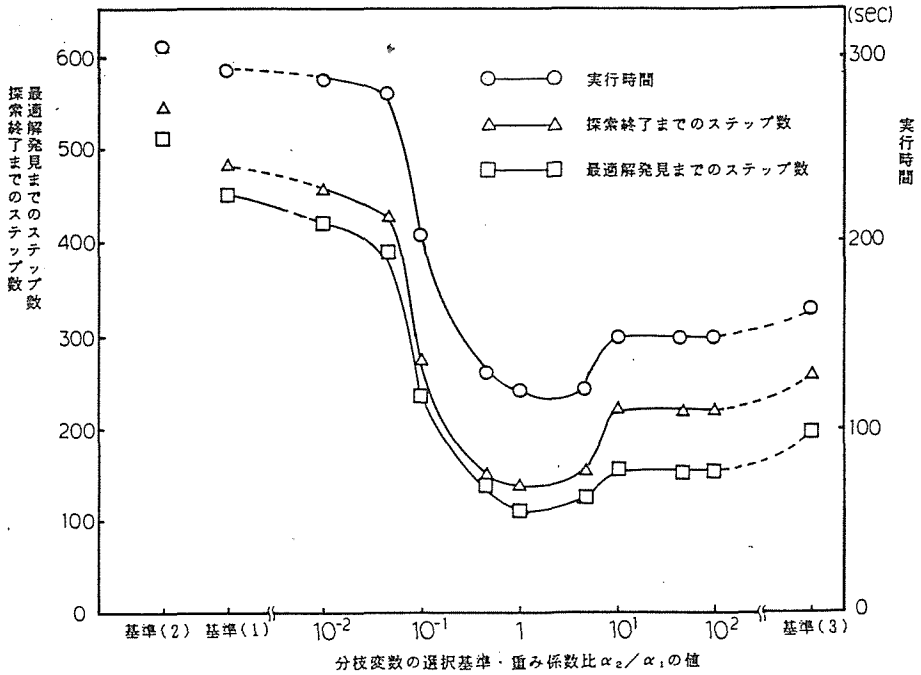


図5　分枝変数の選択基準に関する数値実験結果

表3　2つの選択基準を組み合わせて用いた場合の実験結果

問題番号	0-1変数×制約式	分枝変数の選択基準	最適解発見までのステップ数	探索終了までのステップ数	実行時間 (sec)
1 a)	10×10	基準(1)	11	33	11
		基準(3)	8	36	10
		α_2/α_1=1	8	36	10
2 b)	10×15	基準(1)	12	48	25
		基準(3)	8	42	21
		α_2/α_1=1	8	42	21
3 b)	20× 5	基準(1)	79	153	97
		基準(3)	619	633	252
		α_2/α_1=1	29	88	68
4 a)	20×10	基準(1)	84	149	146
		基準(3)	37	142	154
		α_2/α_1=1	39	132	119
5 a)	20×15	基準(1)	381	412	299
		基準(3)	25	106	146
		α_2/α_1=1	25	98	126
6 b)	25×15	基準(1)	123	386	824
		基準(3)	1068	1418	1681
		α_2/α_1=1	137	414	594

問題番号につけたa），b）はランダム問題のタイプを示す
a）：$(L_a, U_a, L_c, U_c) = (50, 0, 50, 0)$
b）：$(L_a, U_a, L_c, U_c) = (50, 0, 35, 15)$

らに実行時間は短縮されることがわかる。

また，問題のサイズと代理制約式作成の関係に対する数値実験を行ったところ，サイズの小さい問題に対しては代理制約式を用いずに解いた方が実行時間が短くなることがわかった。しかし，20変数以上の問題に対して代理制約式を各ステップごと用いて解いた場合には，これを用いずに解いた場合に比べ実行時間は約30%から60%短縮され，さらに，代理制約式を作成し，今回提案した2つのヒューリスティックスを用いて解いた場合には，代理制約式を用いたい場合に比べ実行時間は約50%から90%短縮されることがわかった。

表4　代理制約式の作成間隔を2，自由変数の選択基準を $\alpha_1/\alpha_2=1.0$ と設定した場合の探索終了までのステップ数と実行時間

問題番号	0-1変数×制約式	Case-1	Case-2	Case-3	Case-4
1 a)	10×10	33 step 11 sec	35 step 8 sec	36 step 10 sec	38 step 7 sec
2 b)	10×15	48 step 25 sec	49 step 14 sec	42 step 21 sec	43 step 12 sec
3 b)	20×5	153 step 97 sec	186 step 79 sec	88 step 68 sec	110 step 59 sec
4 a)	20×10	149 step 146 sec	178 step 115 sec	132 step 119 sec	160 step 96 sec
5 a)	20×15	412 step 299 sec	481 step 229 sec	98 step 126 sec	125 step 106 sec
6 b)	25×5	484 step 294 sec	572 step 224 sec	155 step 120 sec	191 step 95 sec
7 b)	25×10	784 step 912 sec	972 step 700 sec	186 step 351 sec	228 step 258 sec
8 b)	25×15	386 step 824 sec	494 step 656 sec	414 step 594 sec	494 step 457 sec

Case-1：代理制約式の作成間隔 1，自由変数の選択基準 基準(1)
Case-2：代理制約式の作成間隔 2，自由変数の選択基準 基準(1)
Case-3：代理制約式の作成間隔 1，自由変数の選択基準 $\alpha_1/\alpha_2=1.0$
Case-4：代理制約式の作成間隔 2，自由変数の選択基準 $\alpha_1/\alpha_2=1.0$

問題番号につけたa），b）はランダム問題のタイプを示す
a）：$(L_a, U_a, L_c, U_c) = (50, 0, 50, 0)$
b）：$(L_a, U_a, L_c, U_c) = (50, 0, 35, 15)$

5．おわりに

本報告では，0－1線形計画問題を代理制約式を用いた陰的列挙法を使って効率よく解くための2つのヒューリスティックを提案し，その妥当性を数値実験により検証した。

このうち第1のヒューリスティックは，多くの計算を要する代理制約式算出の負担を軽くするために，代理制約式の作成に間隔を持たせるものである。数値実験の結果，探索の後戻りの効果を失わない程度に代理制約式の作成に間隔を持たせた場合に実行時間が短縮されることがわかった。今回の実験では，多くの問題に対して一定間隔2とすると有効であることがわかった。

第2のヒューリスティックは，探索の早い段階で最適解を発見することを狙った分枝変数の選択基準である。今回提案した分枝変数の選択基準は，すでに提案されている2つの選択基準を重み係数を用いて組み合わせたものである。数値実験の結果，多くの問題に対して重み係数比 $\alpha_2/\alpha_1=1.0$ として選択基準を組み合わせることが有効であることがわかった。

文　献

1）E. Balas, “An Additive Algolithm for Solving Linear Programs with Zero-One Variables”: Operations Research, Vol. 13, No. 4, (1965), pp. 517-546.
2）D. R. Plane & C. McMillan, Jr. 著　黒田　他訳：“整数計画法入門“(昭47)，pp. 35-59, 培風館.
3）Fred Grover: “Surrogate Constraints”: Operations Research, Vol. 16 (1968), pp. 741-749.
4）今野　浩，鈴木久敏：“整数計画法と組合せ最適化“(昭57)，pp. 24-46, 日科技連.
5）久保祐二：“0－1計画法における陰的列挙法アルゴリズムの改良とその応用に関する研究“昭和61年度北海道大学電気工学科修士論文.
6）姉崎　淳，大柳俊夫，加地郁夫：“0－1計画問題の解法の効率化に関する数値実験について”，情報処理学会

第35回全国大会予稿集，(1987).
7）姉崎　淳，大柳俊夫，加地郁夫：“0－1計画問題の効率的解法に関する一提案“，昭和62年度電気関係学会北海道支部連合大会予稿集.