整数/組合せ計画法の現状　その1

分枝限定法

整数計画法研究部会*

はじめに

昭和51年春に発足した整数計画法研究部会は，学会員諸賢の御支援のもとに第3年度の活動を行なっているが，本部会では昨年度以来，整数計画法および組合せ計画法のうち整数計画法に近い分野の最近の成果に関するサーベイを実施してきたので，以下6回にわたってその概要を報告する．

整数計画問題は，線形計画問題の変数の全部または一部が整数値をとる問題として定義されるもので，線形計画問題と並んでモデル能力のきわめて大きな問題群である．周知のとおり，この分野は線形計画法の成功のうえに築かれたものであって，現在でもその基本はかわっていないが，60年代末以来その流れは大きく変化した．すなわち初期においては，一般的な整数計画問題を一般的に解く方法の構成に力が注がれたのに対し，最近ではこの線上の研究はいずれかといえば整数計画問題の数学的構造の研究にシフトし，具体的なアルゴリズムの研究は，特殊な構造をもった実用上重要な問題に対するものに重点が移ってきたように思われる．これは整数計画問題の構造的研究の進歩によって，逆説的にその本質的むずかしさが明らかにされ，線形計画法におけるシンプレクス法のような，一般的かつ強力なアルゴリズムを構成することが原理的に困難であることが認識される一方で，問題の特殊構造やデータ構造をうまく利用することによって，効果的に解きうる問題群がいくつか発見されたためといってよい．とりわけ，整数計画問題の範疇に入る問題のうち，組合せ論的色彩の強い0-1変数問題群に対する解法の進歩は特筆すべきものがある．

このような新たな動きの中で，初期の過大評価とその反動として起こった整数計画法は役に立たないとする議論を乗り越えて，整数計画法は誕生以来約20年にして，やっとその正当な位置づけがなされつつあるといってよいのではないだろうか．

この総合報告は，整数計画法の最近の流れについてのサーベイを行なうものであるが，ここで全体の構成を説明しておこう．まず前半の3回では一般的解法の解説を行なう．すなわち第1回目の今回はもっとも実用的なアルゴリズムである分枝限定法とその計算機コードについてを述べ，次回は切除平面法を中心とする“構造的”アプローチを取り上げる．また第3回目は，ヒューリスティック法や感度分析など実用上重要なテクニックを紹介する．

一方，後半の3回は，特殊構造をもつ問題として，ナップサック問題，集合分割問題，巡回セールスマン問題などを2回にわたって取り上げ，最後に最近にぎやかな話題を提供している計算の複雑さについて解説する予定である．

1.　分枝限定法

分枝限定法（branch and bound method）は，主として一般的な整数計画問題や巡回セールスマン問題，スケジューリング問題，割当て問題，固定費問題等の組合せ論的な最適化問題に対して適用される列挙法の一つであり，最近では，多くの商用の数理計画システムにこの方法がインプリメントされて，実用規模の混合整数計画問題を一般的に解くことが可能となっている（たとえば，IBMのMPSXについては，[3，10]，CDCのOPHELIEについては，[8，18]，情報処理振興事業協会のIMPSについては，[16，17] 等を参照されたい）．分枝限定法に関する話題としては，その一般論（たとえば，[1，14]）や各種の組合せ論的最適化問題への適用例（たとえば[11]）に関する議論もあるが，本論ではこれらは割愛して，数理計画システムに採用されている，一般の混合整数計画問題を解くための分枝限定法がどのようなものであるかということに限定して話を進める．（なお，

* 岡本吉晴（三菱総研）　玉井哲雄（三菱総研）
今野　浩（筑波大）

0-1 問題に対する陰的列挙法については，後に項を改めて述べる予定である．)

1.1 基本アルゴリズム

混合整数計画問題に対する分枝限定法は Land-Doig [12]によって生み出されたものであるが，現在の数理計画システムでは基本的に彼らの方法を修正した Dakin [5]や Beale-Small [2]の流儀を採用している．

以下の説明では，つぎのような混合整数計画問題：

$$(P_0):\ \begin{aligned} &\text{最小化} && z=c^tx+d^ty \\ &\text{条　件} && Ax+Ey=b \\ & && x\geq 0 \\ & && l\leq y\leq u,\ y \text{ は整数} \end{aligned} \qquad (1.1)$$

を対象とする．ここで $c,\ x\in R^{n_1}$; $d,\ y,\ l,\ u\in R^{n_2}$, $b\in R^m$, $A\in R^{m\times n_1}$, $E\in R^{m\times n_2}$ で，$l\leq y\leq u$ などは，$l_j\leq y_j\leq u_j,\ j=1,\cdots,n_2$ をあらわすものとする．

(P_0)に対する分枝限定法は，まず(P_0)から y が整数でなければならないという条件を取り除いた線形計画問題 (C_0)（これを(P_0)の連続問題という）を解くことからはじまる．ここで(C_0)の最適解の y 成分がすべて整数であれば，明らかにその解は(P_0)の最適解となる．一方(C_0)の最適解の y 成分の中に整数でないものが含まれる場合は，整数値をもたない整数変数のいずれか1つ y_S を選んで，つぎのような2つの副問題(C_1)と(C_2)を解く：

副問題(C_1)：(C_0)において y_S 制約を $l_S\leq y_S\leq[y_S{}^0]$ に置きかえたもの

副問題(C_2)：(C_0)において y_S の制約を $[y_S{}^0]+1\leq y_S\leq u_S$ に置きかえたもの

（ここで，[ ]はガウス記号，$y_S{}^0$は整数変数y_Sの(C_0)での値である．）つぎに(C_1)か(C_2)の最適解の整数変数の中にその値が整数でないものがあると，さらに(C_1)か(C_2)を同様の2つの副問題に分解して解いてゆく．このように，副問題を図 1.1 のように木構造的に生成し，副問題の最適解が整数解になるか，副問題が実行不可能になるまで分枝（副問題の生成）を続けるのである．

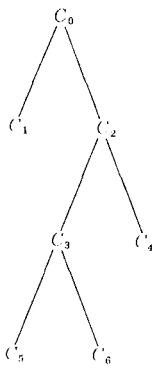

図 1.1 分枝限定法の列挙木

一般に，混合整数計画問題(P_0)に対する分枝限定法の列挙木のノード k には，線形計画問題(C_k)：

$$(C_k):\ \begin{aligned} &\text{最小化} && z=c^tx+d^ty \\ &\text{条　件} && Ax+Ey=b \\ & && x\geq 0 \\ & && l^k\leq y\leq u^k \end{aligned} \qquad (1.2)$$

の最適解が対応する．ノードが N 個生成されている場合，ノード k からの分枝は，(C_k)の最適解で整数値をとらない y 変数の成分 y_S を選択し，(C_k)の変数 y_S の有界制約条件 $l_S{}^k\leq y_S\leq u_S{}^k$ を，

(i)　$l_S{}^k\leq y_S\leq[y_S{}^k]$

(ii)　$[y_S{}^k]+1\leq y_S\leq u_S{}^k$

に変更した2つの副問題(C_{N+1})と(C_{N+2})を生成し（ここで $y_S{}^k$ は(C_k)の最適解における y_S の値である），この2つの副問題を解いて，その解をノード $N+1$, $N+2$ に対応させる（図 1.2）．

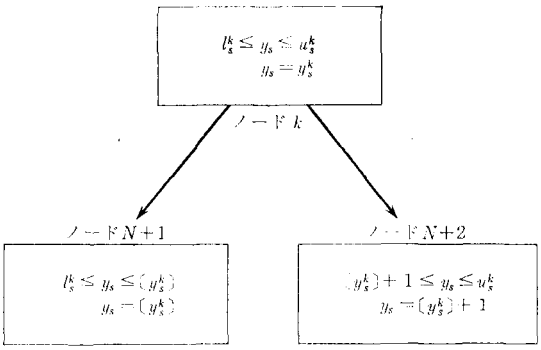

図 1.2 分枝限定法の分枝プロセス

以上のことを背景に分枝限定法の基本アルゴリズムを記そう．以下ではつぎのような記号を用いる：

$\bar{z}$：目的関数の上限（いままでに得られている整数解のうち，最良なものの目的関数値）

$\mathcal{N}$：未分枝ノードの集合

I：整数解の集合

z_k：ノード k の目的関数値

N：ノードの数

ステップ 0〈初期設定〉

$\bar{z}:=\infty$, $\mathcal{N}:=\phi$, $I:=\phi$, $N:=0$ として問題 (P_0) の連続問題(C_0)を解く．(C_0)の最適解のy成分が整数ならば $\bar{z}:=z_0$, $I:=I\cup$(ノード0)としてステップ5へゆく．そうでなければ $\mathcal{N}:=\mathcal{N}\cup$(ノード0)としてステップ1にゆく．

ステップ 1〈分枝ノードの選択〉

未分枝ノードの集合 $\mathcal{N}$ から1つのノード k を選び，$\mathcal{N}:=\mathcal{N}-$(ノード k)としてステップ2へゆく．$\mathcal{N}=\phi$ ならばステップ5へゆく．

ステップ 2〈分枝変数の選択〉

問題(C_k)の解の y 成分で，整数値をとらないもの y_S を選び，y_S の有界制約を(1.3),(1.4)のように変更した2つの副問題 (C_{N+1}), (C_{N+2}) を生成する．すなわち，

$$l_S{}^{N+1}:=l_S{}^k,\ u_S{}^{N+1}:=[y_S{}^k] \qquad (1.3)$$

$$l_S{}^{N+2}:=[y_S{}^k]+1,\ u_S{}^{N+2}:=u_S{}^k \qquad (1.4)$$

とおき，ステップ3へゆく．

ステップ3〈分枝計算〉

(C_{N+1}), (C_{N+2})を解き(これらの問題は(C_k)とはy_Sの上限もしくは下限が異なるだけであり，また(C_k)の最適基底もわかっているので右辺パラメトリック法または双対単体法で比較的容易に解けることに注意) ステップ4にゆく．

ステップ4〈ノード検査〉

副問題(C_{N+i})，すなわちノード$N+i(i=1,2)$を調べ，以下のような処理を行ない，$N:=N+2$としてステップ1にゆく．

a) $z_{N+i}\leqq\bar{z}$で，(C_{N+i})の最適解のy成分が整数の場合は$I:=I\cup$(ノード$N+i$)，$\bar{z}:=z_{N+i}$とし，未分枝ノードのうち$z_J>\bar{z}$を満たす$J\in\mathcal{N}$を$\mathcal{N}$から削除する．

b) $z_{N+i}\leqq\bar{z}$で(C_{N+i})の最適解のy成分が整数でない場合は$\mathcal{N}:=\mathcal{N}\cup$(ノード$N+i$)とする．

c) その他の場合，すなわち$z_{N+i}>\bar{z}$のときは何もしない．

ステップ5〈最終処理〉

$I\neq\phi$ならば，Iの中の最良のものが問題(P_0)の最適解となる．一方$I=\phi$ならば，(P_0)は実行可能解をもたない．

1.2 分枝限定法の計算戦略

分枝限定法の計算効率の良し悪しは，分枝プロセスが進むにつれて生成され解かれてゆく副問題の列挙木のサイズにもっとも大きく依存する．この列挙木のサイズの増大を抑えるためには，前節のアルゴリズムのステップ1における分枝ノードとステップ2の分枝変数の選び方が鍵となる．分枝限定法の計算戦略に関する数値実験の結果は，[3，4，10，13]等に報告されているが，本節ではそれらのうち，有効であると考えられているものをいくつか説明する．

(1) 分枝変数の選択

分枝変数の選択は全体に与える影響の大きな整数変数を優先するのが基本である．すなわち，なるべく目的関数値を大きく変化させる変数を分枝変数として選べば，分枝停止が早く起きる可能性が高く列挙木が大きくならないものと期待される．この線に沿った分枝変数選択の具体的ルールとしてはつぎのようなものが考えられる．

a) モデルの整数変数の重要度がわかっている場合は，あらかじめ分枝変数に選択優先順位をつけておき，これにしたがって分枝変数を選択する．たとえば，目的関数のコストの絶対値の大きい順序とか，モデル特有の変数の重要度の順序とかにしたがって，全順序関係で与えるのが1つの方法である．また，ある整数変数の値が定まると自動的に他の整数変数の値も限定されるという具合に，整数変数間に階層木構造的な関係がある問題に対しては，木構造の形で半順序的な優先順位を与えると有効である．

b) 各整数変数に対して，それを分枝変数として選んだ場合，その副問題の解がどの程度悪化するかを簡便な方法で計算し，もっとも目的関数値の変化が大きいと予測される整数変数を分枝変数として選ぶ．この解の悪化量をあらわすものを罰金(penalty)というが，これにもとづく分枝変数の選択ルールとしてはつぎの2つが有効であると考えられている．

① max-min ルール

非整数値をもつ整数変数y_jの添字集合をJとしたとき，

$$\max_{j\in J}\min(PNL_j,\ PNU_j) \tag{1.5}$$

を与える整数変数を分枝変数とする．

② max-max ルール

$$\max_{j\in J}\max(PNL_j,\ PNU_j) \tag{1.6}$$

を与える整数変数を分枝変数とする．

ここで，PNL_jおよびPNU_jは，それぞれ，整数変数y_jの値を$y_j{}^k$から$[y_j{}^k]$へ，$y_j{}^k$から$[y_j{}^k]+1$へ動かした時の罰金である．PNL_jを下方罰金(down penalty)，PNU_jを上方罰金(up penalty)というが，その定義式としてはつぎの2つがよく使われる：

(i) $$PNL_j=PCL_j\cdot f_j \tag{1.7}$$
$$PNU_j=PCU_j\cdot(1-f_j) \tag{1.8}$$

(ii) $$PNL_j=f_j \tag{1.9}$$
$$PNU_j=1-f_j \tag{1.10}$$

ここで$f_j=y_j{}^k-[y_j]$である．PCL_jとPCU_jは，それぞれ，下方罰金コスト(down penalty cost)，上方罰金コスト(up penalty cost)という．これにもいくつかの計算方法があるが，それらについて述べる前にmax-min ルールと max-max ルールの違いについて簡単に説明しておこう．

max-max ルールは，各分枝で，片方の分枝が目的関数を非常に悪くするように保つのに対し，max-min ルールは両方の分枝とも目的関数を悪くするようなものを選ぶ方法である．したがって，分枝の進行過程で，最初の整数解が見つかるまでは，max-max ルールを適用し，それ以後 max-min ルールにスイッチするのがよいであろう．なぜなら，最初の整数解が求まった時には，未分枝ノードの中に目的関数の値があまりよいものは残っていないから，これを max-min ルールで分枝させてゆくと，はやく分枝がストップして列挙木が大きくならないように保たれると考えられるからである．

罰金コストPCL_j，PCU_jの計算方法の1つとして，

擬似コスト(pseudo cost)と称するものがあるのでそれについて説明しよう[3,10]．ノードkから分枝変数y_Sによって，ノード$N+1$とノード$N+2$へ分枝し，y_Sの値がそれぞれ$[y_S^k]$と$[y_S^k]+1$になったものとして，分枝変数y_Sの罰金コストを，

$$PCL_S=|z_{N+1}-z_K|/f_S \tag{1.11}$$

$$PCU_S=|z_{N+2}-z_K|/(1-f_S) \tag{1.12}$$

のように定義する．この罰金コストは，以後の分枝で，整数変数y_Sが分枝変数に選ばれた場合の目的関数値の変化率の予測値としての意味をもっている．

分枝計算を双対単体法で行なう場合の，最初の1回目の反復での目的関数値の減少率を罰金コストとするのも1つの方法である[6]．整数変数$y_S(S\in J)$が分枝変数として分枝が起こると，y_Sの値を$[y_S^k]$にするほうの分枝（下方分枝）では，y_Sの有界制約が，$l_S^k\leq y_S\leq[y_S^k]$となり，ノードkでのy_Sの値は$[y_S^k]+f_S(=y_S^k)$となるから，ノードkの基底では，y_Sはf_Sだけの実行不可能量をもっている．ノードkの基底のp行にy_Sがはいっているとすると，双対単体法による下方分枝の最初の反復で，目的関数値は，

$$\min_j\{\alpha_{oj}f_S/\alpha_{pj}|\alpha_{pj}>0\} \tag{1.13}$$

だけ増加する．ここでα_{oj}はreduced cost，α_{pj}は単体表のp行j列要素である．この目的関数の変化率を採用して，

$$PCL_j=\min\{\alpha_{oj}/\alpha_{pj}|\alpha_{pj}>0\} \tag{1.14}$$

とする．同様に上方罰金コストPCU_jは，

$$PCU_j=\min\{\alpha_{oj}/|\alpha_{pj}||\alpha_{pj}<0\} \tag{1.15}$$

とする．このほかにもGomoryの切除平面を用いて，さらに強い罰金コストも考えられている[24]が，この種の罰金コストを計算するためには，かなりの手間がかかるのが欠点である．

(2) 分枝ノードの選択

基本アルゴリズムのステップ1で，未分枝ノードの集合$\mathcal{N}$から，つぎの分枝ノードを選ぶ場合，つぎの両極端な方法が考えられる：

方法1：もっとも最近に未分枝ノード集合$\mathcal{N}$に加わったものをつぎの分枝ノードとする（Depth First法）

方法2：未分枝ノードの集合$\mathcal{N}$からもっともよい目的関数値，あるいは評価値（未分枝ノードに対応する副問題内での最適整数解の目的関数値の予測値）をもつものを，つぎの分枝ノードとする(Breadth First法)

Depth First法では一般に$\mathcal{N}$のサイズはあまり大きくならないことが予想されるが，はじめのうちはよい整数解が得られることは期待できない．これに対しBreadth First法はよい整数解が得られることが期待される反面，集合$\mathcal{N}$のサイズが大きくなり，なかなか整数解が見つからない可能性がある．

以上のように，この2つの方法には，一長一短があるので，実際には両者を適当に折衷するのがよいようである．たとえば，ノードkが分枝ノードとなってノード$N+1$と$N+2$が生成された時に，そのつぎの分枝ノードを以下のように選択する戦略は一般にかなり有効である．

(a) ノード$N+1$と$N+2$の少なくとも一方が未分枝ノードとなる場合は，その基底状態や基底行列が使用可能の状態になっているので，なるべくこの2つのノードのどちらかをつぎの分枝ノードとするのが効率的である．ノード$N+1$と$N+2$がともに未分枝ノードとなる場合は，

(i) 目的関数値の小さいほう

(ii) 評価値の小さいほう

(iii) 分枝によって計算された擬似コスト(1.11)，(1.12)の小さいほう

等の基準で選択する．ここで一般にノードkの評価値E_kは，(1.11)，(1.12)式の擬似コストを用いて，

$$E_k=z_k+\sum_{j\in N_2}\min(f_j\cdot PCL_j,(1-f_j)PCU_j) \tag{1.16}$$

のように計算される．

またノード$N+1$と$N+2$があまりよい解でない場合は，それらをつぎの分枝ノードとはしないような条件を加えることも考えられる．たとえば，

$$E=\min_{k\in\mathcal{N}}E_k \tag{1.17}$$

として，

$$E_{N+i}\leq E+\rho(E-z_0) \tag{1.18}$$

でなければつぎの分枝ノードとしないというような条件が考えられている[10]．ここで，z_0は連続最適解の目的関数値，ρは適当な定数（たとえば0.5くらい）である．

(b) ノード$N+1$と$N+2$がともにつぎの分枝ノードになり得ない場合は，未分枝ノード$\mathcal{N}$の中を探してつぎの分枝ノードを選ばなければならない．これは，列挙木を逆もどりすることに相当するので，バックトラッキング(backtracking)という．この場合，

(i) まだ1つも整数解が見つかっていない場合は，もっとも最近$\mathcal{N}$に加わった未分枝ノード

(ii) 整数解が1つ以上得られていれば，目的関数値あるいは，評価値のもっとも小さい未分枝ノード

などを選択するのがよい．

1.3 特殊な制約式の取扱い

整数計画問題の典型的な例として，いくつかの代替案

のうちどれか1つを選択する問題がある．このような問題では代替案 j を選択するか否かを 0-1 変数 y_j, $j=1,\cdots,p$ で表現すると，y_j はつぎの条件

$$\left.\begin{aligned} &y_1+y_2+\cdots+y_p=1 \\ &y_j=0 \text{ または } 1,\ j=1,2,\cdots,p \end{aligned}\right\} \quad (1.19)$$

を満足する．このような関係にある 0-1 変数群は，特殊変数群(special ordered set)とよばれ，上式を SOS 制約，y_j, $j=1,\cdots,p$ を SOS 変数という．分枝限定法では，SOS 変数を，ひとまとめにして特別な取り扱いをしたほうが効率がよい([10, 18])のでそれについて説明しよう．

いま，SOS 変数 y_j に対して適当な重み $w_j>0$ を与えておく．簡単のため以下では，

$$w_j \geq w_{j+1},\ j=1,\cdots,p-1 \quad (1.20)$$

を仮定し，ノード k で特殊変数群 $y_1,\cdots,y_p$ は0-1整数値をとっていないものとする．このとき，SOS 変数 y_j の値を $y_j{}^k$ として，

$$w=w_1y_1{}^k+\cdots+w_py_p{}^k \quad (1.21)$$

とし，r を $w\geq w_{r+1}$ となる最初の変数番号とする．このとき，この特殊変数群を分枝変数(群)とする分枝はつぎのようにする．

(i) 副問題(C_{N+1})では $y_j=0$, $j=1,\cdots,r$ とする．

(ii) 副問題(C_{N+2})では $y_j=0$, $j=r+1,\cdots,p$ とする．

このように副問題を構成すれば，個々の SOS 変数に対して通常の副問題の構成法を行なうよりもずっと列挙木が小さくなり，効率アップが期待される．

SOS 変数に対して上記のような分枝を行なう場合，(1.11)，(1.12)式に対応する罰金コストは，変数 y_r に対してつぎのように計算できる．

$$PCL_r=|z_K-z_{N+1}|/\sum_{j\leq r} y_j{}^k \quad (1.22)$$

$$PCU_r=|z_K-z_{N+2}|/\sum_{j>r} y_j{}^k \quad (1.23)$$

したがって，(1.16) 式に対応するノード k の評価値 E_k も，

$$E_k=z_k+\min(PCL_r\sum_{j\leq r} f_j,\ PCU_r\sum_{j>0} f_j) \quad (1.24)$$

のようになる．

2. 混合整数計画法の計算機コード

前節で述べたようなアルゴリズムをインプリメントした計算機コードは，かなりの数にのぼる．ここでは，とくに汎用の数理計画システムの中で実現されているものを取り上げる．

代表的な数理計画システム――以下ではこれを MPS と総称する――では，ふつう混合整数計画問題を解くプログラム――これを以下 MIP と総称する――がくみこまれている．例を挙げれば，IBM の MPSX ないし MPSX/370, CDC の OPHELIE, SCICON の UMPIRE, 情報処理振興事業協会の IMPS, Bonner & Moore 社でつくられ多くの機種にインプリメントされている FMPS などであるが，これらのシステムは，いずれも分枝限定法をその解法として採用している．IMPS には，第3回で説明するヒューリスティック解法の1つである R-最適法も，利用者が選んで使える機能としてインプリメントされている．上に挙げた中では，FMPS は従来分枝限定法によらず，後に解説する Benders の分割法によっていたが，最近分枝限定法が組み込まれたとのことである．また，前節で述べた SOS 制約の特別な取扱いは，UMPIRE や MPSX/370 で採用されている．

これらの MPS は，線形計画問題(LP)を解く機能をその中心としている．MIP の機能は，近年のユーザー・ニーズの拡大とともに付加されてきた機能の一つである．LP に対しては単体法という問題の形によらず適用可能でしかも強力な手法があり，それにもとづいて汎用的で実用規模の問題を効率よく取り扱えるシステムをつくることが可能となったが，この LP に対する単体法に当るのが，MIP の場合少なくとも現在のところ，分枝限定法といういささかスマートさに欠ける手法になるようである．残念なことに分枝限定法は LP における単体法ほど強力ではない．現在の技術水準からすれば，MIP の問題に関しては，個々の問題に応じその特殊構造を利用した効率のよいプログラムを（それが分枝限定法を利用するものであっても，その問題用に書き直して）作成し適用するというのが，正道かも知れない．しかし，それでは実際に問題をかかえている一般ユーザーには繁雑で不便この上ない．そこで，少なくとも汎用性だけは充分にある分枝限定法が MPS に用意されて，それを注意深く上手に使うことにより実用上の問題もある程度は解かれているというのが，現状である．

分枝限定法が MPS で採用されている別の理由は，それが現在提案されているいくつかの手法の中では，実験的にも経験的にもすぐれていると認められていることはもちろんであるが，つぎのことも大きな理由であると思われる．すなわち，前節で説明したように，この分枝限定法という手法は，副問題としての LP 問題をくり返し解いていくという形をとっており，しかもその際，双対単体法または右辺パラメトリック法を用いて効率よく計算できるという仕組みになっているので LP を解く機能と連動させることがきわめて都合のよいという事情である．一方ユーザー側からすれば，LP の場合と同じ形式の入力データを用意すればよいことや，MPS のもつ豊

富なデータ管理機能（たとえば問題の保存や修正，解の保存や回復など．これらの機能がMPSの中でますます重要性を増してきている．）を利用できる点でMPSの中にMIP機能が含まれていることは便利である．

しかしそれでも，一般ユーザーにとってMPSは決して使いやすいものとは言えない．それは利用者の選択可能な計算戦略がきわめて多く，それらをどう使えばよいかわからない場合が少なくないからである．利用者の自由が制限されているほうがむしろ使いやすいということは，よく経験することである．計算過程の細部まで利用者に公開して選択させるのは，ある意味では，システムの自信のなさを示すものであるともいえるかも知れない．もちろん，どのシステムも標準的な使い方を用意しており，面倒なことをしたくないユーザーはそれを利用すればよいようになっている．しかし，たとえばLPの場合は，普通の問題に対してなら，そのようなシステムの用意した標準的な手続きによって充分効率よく解が得られるが，MIPについてはなかなかそういかないのである．したがって，IBMのMPSXのマニュアルに典型的に見られるように，システムの使用法に対しても，初級コース，中級コース，上級コースとわけた記述をして，簡単にすませたい人（解きたい問題がごく簡単だったり，計算機時間の予算をたくさんもっている人）には初級コースのやり方を，いろいろな機能を駆使してなんとかうまく解きたい人には，中級，上級コースのやり方を勧めるケースが多いようである．

もう1つの使いにくい理由は，問題が与えられたとき，どの程度の時間で解けるのか予測がつかないことで，簡単そうに見える問題でもかなりの計算時間を使って最適解はおろか一つの整数解すらみつからないということが起こりうる．これもMIPのもつ本質的なむずかしさからきている問題である．IMPSにインプリメントされているようなヒューリスティックな解法の存在価値はこの辺にあって，必ずしも最適解が求まらなくてもよいからある程度よい解をいくつかほしいとか，あまり長くない計算時間内にほしいとかいう目的には向いている．

ここで，MPSのMIPのとくに利用者側からみた形態がどのようになっているかを簡単に紹介しよう．代表的なものとしてIBMのMPSX/370と情報処理振興事業協会のIMPSを例にとる．システムの構成は両者ほぼ同じなので以下の説明は，断らない限りMPSX/370にもIMPSにもあてはまる．

問題は定まった形式の入力データで与える（この形式は両者で同じになっている）．その入力データを用いて，どのような戦略，どのような手順で最適化を実行するかについては，システムの有する制御言語を用いて指示することができる．システムは，この制御言語で書かれたプログラムを解読するコンパイラをもち，その解読結果にしたがって実行をすすめる．

実行の際に単位となるいくつかの手続きがある．利用者が制御言語によってよび出すことのできるのは，このレベルの手続きである．計算の戦略の指示，その他手続きに対しての必要な情報は，この手続きよび出しの際にパラメータとして指定するか，あるいは利用者のアクセス可能な定められたシステム変数群のうちの適当な変数に，直接値を代入することによって与えることができる．とくに実行の制御に関して，両システムとも有している特有な機能は割り込み処理機能である．各手続きは種々の条件で割り込みを起こし，全体の実行を統轄しているエクゼキュータに制御が返される．エクゼキュータは，この割り込みの種類に応じて利用者の指定した適当な処理を行なう．

MIP関係の主な手続きを，MPSX/370，IMPSそれぞれについて簡単に説明する．

［MPSX/370］

MIXSTART MIPの分枝限定法による最適化実行のための準備を行なう．また後述する手続きMIXSAVEによって保存された途中解から，実行を再開する場合にも，保存解を回復するために使われる．

計算戦略の選択に関するパラメータで重要なものとしては，整数変数の優先順位（これは前節の“分枝変数の選択”で述べた選択基準に関係する）のルールをここで指定することができる．

MIXFLOW 分枝限定法によって整数最適解の探索を行なう．ここではとくに，計算戦略の指定が豊富にある．それらは，

（i） 分枝ノードの選択に関するもの
（ii） 分枝変数の選択に関するもの
（iii） 第1分枝の選択に関するもの

などである．探索を適当に打ちきって効率をあげるために，目的関数がある値以下のノードについては，その先の探索を打ちきるという境界値の指定や，整数解がある数以上見つかったら，全体の探索をうちきるなどの指定もある．

MIXSAVE 分枝限定法による探索の現在の状態をファイルに保存する．

MIXSTATS 分枝限定法による探索の現在の状態に関する種々の情報をプリント出力する．

MIXFIX 整数変数を指定されたノードにおける整数値に固定した形に問題を書きかえる．これによって，連続変数に関して改めて最適化をやり直したり，感度解析を行なったりすることが可能になる．

MIXBOUND MIXFIX と同じように整数変数を整数値に固定するために，REVISE（問題ファイルを修正する機能をもった手続き）用のデータを出力する．MIXFIX との主な違いは現在の問題をその場で書きかえてしまわない点にある．

なお，これらの手続きを標準的に用いるための制御言語のシステムマクロとして OPTIMIX がある．

[IMPS]

STARTMIX 後述する手続き BBMIX および，HEUMIX による整数解探索の計算を行なうための準備をする．また保存された BBMIX の整数解探索の情報を回復し，そこから BBMIX の計算が展開できるようにする．機能的に MPSX/370 の MIXSTART と大体同じだが，とくに HEUMIX 処理のために初期整数解をデータとして読み込んで処理する機能を含む．

BBMIX

分枝限定法によって整数最適解の探索を行なう．MPSX/370 の MIXFLOW と大体同じ機能で，選択可能な計算戦略も，同じようなものがある．

HEUMIX

ヒューリスティックによる整数解探索を行なう．計算戦略の指定としては，

（i） 手法として 1-最適法（同時に動かす整数変数は1つだけ）と 2-最適法*（同時に2つの整数変数を動かす）との組合せに関するもの

（ii） 整数変数の優先順位に関するもの

などがある．

SAVEMIX

BBMIX による整数解探索の途中の情報をファイルに保存する．

またこれらの手続きを標準的に用いるための制御言語のシステムマクロとして OPTMIX 1，OPTMIX 2，OPTMIX 3 の3種類をもっている．

以上が代表的な MPS の形態の概要である．なお，ここに紹介したような MPS はいずれも1つの独立したソフトウエアのシステムという形態をとっている．しかし，とくに MIP の場合は，前にも述べたように，その問題ごとに特性を活かしたプログラムを工夫する必要があることが多いことなどから，実際上は利用者のかいたプログラムの中に，これらのシステムの機能を直接よび出して使用するという，パッケージ的な用い方が望ましい場合もあるであろう．その意味では，MPSX/370 が従来の MPSX 制御言語の他に PL/I プログラムでも制御ができるようになっていることは，この方向を意識したものと思われる．

* これについては第3回に詳しく解説する．

参考文献

[1] Balas, E., "A Note on the Branch-and Bound Principle", *Operations Research* **16**, pp. 442-445(1968)

[2] Beale, E. M. L. and R. E. Small, "Mixed Integer Programming by a Branch and Bound Technique", in *Proc. IFIP Congress*, Vol. 2 (W. A. Kalenich, ed.,), Spartan Press(1965)

[3] Benichou, M., et al., "Experiments in Mixed-Integer Linear Programming", *Math. Prog.* **1**, pp. 76-94(1971)

[4] Breu, R. and C.-A. Burdet, "Branch and Bound Experiments in Zero-One Programming", *Math. Prog. Study* **2**, pp. 1-50(1974)

[5] Dakin, R. J., "A Tree-Search Algorithm for Mixed Integer Programming Problems", *Computer Journal* **8**, pp. 250-255(1965)

[6] Driebeek, N. J., "An Algorithm for the Solution of Mixed Integer Programming Problems", *Management Science* **12**, pp. 576-587 (1966)

[7] Echols, R.E. and L. Cooper, "Solution of Integer Linear Programming Problem by Direct Search," *J. ACM*, **15**, pp. 75-84(1968)

[8] Forrest, J.J.H., J.P.H. Hirst and J.A. Tomlin, "Practical Solution of Large Mixed Integer Programming Problems with "UMPIRE"", *Management Science* **20**, pp. 736-773(1974)

[9] Garfinkel, R. S., and G. L. Nemhauser, *Integer Programming*, John Wiley & Sons, 1972

[10] Gauthier, J.-M. and G. Ribière, "Experiments in Mixed-Integer Linear Programming Using Pseudo-Costs", *Math. Prog.* **12**, pp. 26-47(1977)

[11] Lawler, E. L. and D. E. Wood, "Branch-and-Bound Methods : A Survey", *Operations Research*. **14**, pp. 699-719(1966)

[12] Land, A. H. and A. G. Doig, "An Automatic Method for Solving Discrete Programming Problms", *Econometrica* **28**, pp. 497-550(1960)

[13] Mitra, G., "Investigation of Some Branch and Bound Strategies for the Solution of Mixed Integer Linear Programs", *Math.*

Prog. **4**, pp. 155-170(1973)

[14] Mitten, L.G., "Branch-and-Bound Methods: General Formulation and Properties", *Operations Research* **18**, pp. 24-34(1970)

[15] Roy, B.,R. Benayoun and J. Tergny, "From S. E. P. Procedure to the Mixed Ophelie Program", in *Integer and Nonlinear Programming* (J. Abadie, ed.), American Elsevier(1970)

[16] 反町洋一，岡本吉晴，玉井哲雄 "数理計画システムの開発" 情報処理**15**, pp. 710-716(1974)

[17] Sorimachi, Y.,H. Mukawa, Y. Okamoto and T. Tamai, "Implementation of the Heuristic Mixed Integer Program to the Mathematical Programming System", in *Proceedings of IX International Symposium on Mathematical Programming*, Hungary(1976)

[18] Tomlin, J. A., "Branch and Bound Method for Integer and Non-Convex Programming", in *Integer and Nonlinear Programming* (J. Abadie, ed.), American Elsevier(1970)

[19] ——, "An Improved Branch-and-Bound Method for Integer Programming", *Operations Research* **19**, pp. 1070-1074(1971)

書評

書名 **Models of Preventive Maintenance**

著者 I. B. Gertsbakh

発行 North-Holland Publishing Co. 1977年 257ページ

装置が複雑・高度化しつつある現在，保全はますます重要なものとなってきている．ところが，保全に従事する人々が，ORによる最適保全をめざし数学の理論をはじめから学ぶ余裕は，現状では皆無といってよいであろう．本書はこの困難を解決すべく保全に対するシスティマティックな取扱いを行なっている．用いる手法はセミマルコフプロセス（SMP）であり，この最適化をめざすことによって保全への単純かつ明快な数学的アプローチを提示している．

本書は4章からなり，第1章で保全全般に関する概括ならびにSMPの明快な説明がなされている．元来，SMPというと，むずかしいというイメージがあるが，その点本書の説明は要を得ており，理解しやすい．第2章では，正常・故障の2状態をとる一般的なモデル——たとえば，Age Replacement, Block Replacement 等——がSMPを用いて解析されている．第3章は本書のメインテーマであり，システムの内部状態を直接把握しえない場合，すなわち代用特性にもとづくシステムの保全を扱っている．このパラメータを"prognostic parameter"とよび，これとシステムの真の状態との間に確率法則を与え，その最適化をはかっている．モデルの条件設定に甘さが見られるが，テーマとして非常に興味深いものである．その他，本章では，冗長ユニットの最適使用法，保全方法が多岐にわたるケース，k-out-of-n システム等に対する最適化，ならびにコントロールリミットポリシーの説明がなされている．第4章では，理論と実践とのギャップを埋めるべく，複雑なシステムにおける各要素のグループ化の方法ならびに多次元の特性をもつ場合の解析法を提示している．その他，グループ化した保全，大規模システムの保全，故障診断等のトピックスもあり，興味深い．

以上が本書のあらましであるが，SMPを用いて系統的に取り扱おうという意図はくめるが，その反面，取り上げた内容ならびに，そのモデルが平易になりすぎたきらいがあるのは残念である．研究者向け，というよりは，実務にたずさわっている人が実践に数学的アプローチを試みる時に有用と思われる．また卒研生あるいは，大学院修士課程における保全へのOR的アプローチの入門書として役立つであろう．ただ，例が多いのはよいが，取り上げた問題が本質的な点を指していないものも多く，単なるケーススタディ的なモデルが多いのが気になった．自分の捜しているモデルを見つけるには，各章の初めの概説に目を通し捜すとよい．本書は，最初にSMP（§1.3）のところを読めば，あとはどこからでも読めるようにできており，はじめから通読する必要はない．適宜，出くわした問題を求め，そこだけ読めば，事足りる本である．なお前述したように，新しいトピックスを数多く含んでおり，保全ならびに信頼性の分野に，他の領域からの手法の適用をなしうる話題も提供されている．

本書の考え方，ならびにSMPが保全に従事するORワーカに活用されることを願う．（鈴木和幸）