情報処理学会論文誌 Vol. 50 No. 8 1780–1788 (Aug. 2009)

最小拘束問題の分枝限定法アルゴリズム

片岡靖詞[†1] 坂森義成[†2]

与えられた 0-1 行列に対し，列を並べ替えることで，各行において 1 ができるだけ連続して現れるようにする問題を最小拘束問題と呼ぶことにする．最小拘束問題を厳密に解くことは難しく，商用ソフトのように線形計画法をベースにした分枝限定法では，例題ほどのサイズでさえも解くことが困難である．最小拘束問題には動的計画法によるアルゴリズムも知られており，小さな問題では効率良く解けるが，メモリの消費が指数的に増大するため，問題が大きくなるとたちまち限界に達する．しかし，動的計画法の考え方は，行数が小さい場合であれば，下界値を求めるために利用することもできる．本論文では，動的計画法をもとにした下界値を用いて，分枝限定法による厳密解法を開発している．さらに最適解の必要条件に基づいた枝刈りや下界値の強化の工夫も取り入れている．

A Branch-and-bound Algorithm for the Minimum Binding Problem

Seiji Kataoka[†1] and Yoshinari Sakamori[†2]

Given a 0-1 matrix, the *minimum binding problem* is about finding a sequence of the columns in which 1's appear as consecutively as possible. This problem is so hard to solve exactly that LP-based branch-and-bound algorithms cannot solve even text-like instances. A dynamic programming algorithm that can solve some smaller instances is known but it consumes exponential memories. The dynamic programming algorithm, however, is applicable to obtain lower-bounds if the number of rows is relatively small. Hence, using the DP-based lower-bound, we develop a branch-and-bound algorithm. We also propose some strategies to prune the enumerating tree and to obtain tighter lower bounds.

†1 防衛大学校情報工学科
Department of Computer Science, National Defense Acacemy
†2 陸上自衛隊
Japan Ground Self Defense Force

1. はじめに

行集合を M（$|M| = m$），列集合を N（$|N| = n$）とする 0-1 行列 $A = [a_{ij}]$（$a_{ij} \in \{0, 1\}$，$i \in M$，$j \in N$）を考える．ただし，すべてが 0 の行およびすべてが 0 の列はないものとする．最小拘束問題（MBP: Minimum Binding Problem）とは，列を適当に並べ替えることで，各行において 1 ができるだけ連続して現れるようにする問題である．ある行を左から右へ見ていくとき，最初に 1 が現れてから，最後に 1 が現れるまでを “1 の幅” と呼ぶことにし，本論文では評価値として 1 の幅の総和を最小にすることを考える．**図 1** において，左は与えられた行列，右は最適に列を並べ替えた行列である．このとき，評価値は 36 から 24 に改善される．

もし，すべての行において 1 の幅の中に 0 が存在しないように列を並べることができるとき，その行列は “連続する 1 の性質（concecutive ones property）” を持つといい，そのような行列は完全ユニモジュラであることが知られており，数理的にも優れた性質を持つ[10),11)]．MBP は与えられた行列が連続する 1 の性質からどれだけ遠ざかっているかを示す指標にもなる．Booth[1)] は，与えられた 0-1 行列が連続する 1 の性質を持つか否かを判定する多項式オーダのアルゴリズムを開発している．このアルゴリズムを適用すると，与えられた行列が連続する 1 の性質を持っている場合は，具体的な列の並べ方を答えることができる．しかし，そうでない場合には，良い解を答えることはあっても，最適性を保証することはできない（より正確には，連続する 1 の性質は，列の先頭と末尾とがつながって環状になっている場合も含めているので，Booth のアルゴリズムでは，良い解すら得られない場合も少なくない）．

MBP は基礎的な問題であるが，様々な適用例も考えられる．行を工具・列を仕事とすると，MBP は FMS の工具マガジンのスケジューリング問題ととらえることができる．また，行を俳優・列を映画の場面とすると，撮影計画問題ととらえることもできる．Kataoka ら[9)] は，行を教員・列を学生とすることで，論文審査の適切な順序を計画する例をあげて MBP を紹介している．

MBP に関連する研究として，時間割計画問題や最適な順列を求めるスケジューリング問題などが思い浮かぶ．時間割計画問題とは，クラス編成問題[2),3),5)] や看護師スケジューリング問題[8)]，乗務員スケジューリング問題[7)] などを含む．時間割計画問題では，行列を時間割に・各要素をクラス（看護師・乗務員）に見立て，各要素を個々に入れ替えながら最適な時間割を組む問題である．一方，MBP では列の要素を保持したまま，列ごとに入れ替えな

place col.	1 1	2 2	3 3	4 4	5 5	6 6	7 7	8 8	9 9	10 10	ones-range
1	0	1	1	1	0	0	1	1	0	0	7
2	1	0	0	0	1	0	1	1	0	1	10
3	1	0	0	0	1	1	1	0	1	0	9
4	1	0	1	1	0	1	0	1	0	1	10

place col.	1 2	2 3	3 4	4 8	5 7	6 1	7 5	8 10	9 6	10 9	ones-range
1	1	1	1	1	1	0	0	0	0	0	5
2	0	0	0	1	1	1	1	1	0	0	5
3	0	0	0	0	1	1	1	0	1	1	6
4	0	1	1	1	0	1	0	1	1	0	8

図 1　最小拘束問題の例：与えられた 0-1 行列（左）と最適に列を並べ替えたもの（右）
Fig. 1　An example of MBP: a 0-1 matrix (left) and its optimal sequence (right).

ければならないところが大きく異なる．また時間割計画問題では，一般に複雑な条件が絡み合い，実行可能解を 1 つ見つけることさえも難しく，ほとんどが何らかの近似解法の研究であるが，MBP ではどのような列の並びでも実行可能解となる．したがって，MBP の厳密解を求めるために，時間割計画問題の研究やアルゴリズムが直接的に適用できる可能性は低い．

もう一方の関連研究であるスケジューリング問題は，様々なタイプの目的関数や制約式，あるいはコスト，処理時間，資源，納期などの数値データを含んでいる[4),6)]．しかし，MBP の記述に必要なのは 0-1 行列だけであり，この単純さがかえって MBP を厳密に解くことを難しくしている．具体的な数値データがないということは，線形計画法をベースとした分枝限定法において，多少の列の固定や交換—特に行列の中ほどにおいて—を行っても，上下界値に明確な変化が生じないため，列挙木の無駄な枝を刈るための十分な情報が得られない．そして，同じ評価値を持つ多くの解を徒に探索してしまう現象が生じる．一例として，Kataoka ら[9)] は，図 1 の例題を数理計画問題として定式化し，商用ソルバ XpressMP を適用したところ，20,000 秒もの実行時間を要したと報告している．近年，計算機やソルバは著しく発展しているため，再実験を Dell Dimension 8300 Pentium(R)4 3.2 GHz 上で（以降の実験もこの計算機を使用）Xpress-MP（version 2005B）を用い，さらに Kataoka らよりも改善・強化した定式化で実行したところ，同じ例題であれば 3.6 秒で解くことができた．しかしながら，適当に 2 行追加して 4 行 12 列の問題にするだけで 10 分以上の時間を要した．充実したソルバは日進月歩の世界ではあるが，同様の戦略では，一気に倍以上のサイズが解ける可能性は低い．

MBP の計算量やアルゴリズムに関しては，著者らの調べた限りにおいては，NP-困難か否かも明確ではなく，Kataoka らの動的計画法以外に，そのままアルゴリズムを適用できるような類似研究も見当たっていない．こうしたことからも，MBP は応用例も豊富な基礎的問題であり，単純ながらも新規性のある問題であるととらえている．

ここからは，MBP の解法について述べるが，2.1 節の途中までは Kataoka ら[9)] から抜粋してまとめたものである．しかし，2.2 節以降の展開のために必要なアルゴリズムや補題が現れるので，本論文だけでの完結性を高めるためにも，重複記述をする（証明は省略する）．

Kataoka らは MBP に対する動的計画法のアルゴリズムを開発し，DP-MBP と呼んでいる．列集合 S（$\subseteq N$）に属する列（以降 “S の列” と記述）を行列の左端から位置 $|S|$ までに割り当て，$f(S)$ と $\Delta f_i(S,j)$ を次のように定義する．

$f(S)$　は，S の列を位置 1 から位置 $|S|$ までに割り当てたときの MBP の最適値．ただし，$N \setminus S$ の列が後に続くので，位置 $|S|+1$ 以降に 1 を含む行では，位置 $|S|$ は 1 の幅の中にある．

$\Delta f_i(S,j)$　は，S の列を位置 1 から位置 $|S|$ までに割り当て，位置 $|S|+1$ に列 j（$\in N \setminus S$）を割り当てるとき，第 i 行において，位置 $|S|+1$ が 1 の幅の中であれば 1，1 の幅の中になければ 0 をとる関数．ただし，$N \setminus (S \cup \{j\})$ の列が後続することに注意する．

$\Delta f_i(S,j)$ は式 (1) のように算定できる．

$$\Delta f_i(S,j) := \begin{cases} 1 & a_{ij} = 1 \text{ のとき} \\ 1 & 0 < \sum_{s \in S} a_{is} < \sum_{s \in N} a_{is} \text{のとき} \\ 0 & \text{それ以外のとき} \end{cases} \tag{1}$$

したがって，DP-MBP の漸化式は式 (2) のようになる．

$$f(S) := \min_{j \in S} \left\{ f(S \setminus \{j\}) + \left[\sum_{i \in M} \Delta f_i(S \setminus \{j\}, j) \right] \right\} \tag{2}$$

初期状態は $f(\emptyset) := 0$ であり，最適値は $f(N)$ である．

DP-MBP は $O(2^n)$ のメモリと計算の手間を要するため，ほんの数列であっても問題のサイズを小さくすることは有効である．ここで，列 j_1 と列 j_2（$j_1 \neq j_2$）が，$a_{ij_1} = a_{ij_2}$（$\forall i \in M$）のとき，これらの列は “同一パターン” を持つと呼ぶことにする．

補題 1　MBP の最適解において，同一パターンの列は連続して現れる．　■

列 j と同一パターンを持つ列の数を $p(j)$ とする．補題 1 より，$p(j)$ 本の列はまとめて 1 つの代表列として縮約し，式 (2) において，N を縮約した問題の列集合とし，大括弧の項に $p(j)$ を掛ければ，縮約問題を解くことができる．

表 1 は DP-MBP の計算機実験結果である．Kataoka らにも同様の実験結果があるが，後

表 1　DP-MBP の計算機実験結果（秒）
Table 1　The computational results of DP-MBP (sec.).

		sparse (25%)				middle (50%)				dense (75%)			
m	n	#	ave	min	max	#	ave	min	max	#	ave	min	max
	20	10	0.03	0.00	0.08	10	0.01	0.02	0.58	10	0.00	0.00	0.05
5	25	10	0.11	0.02	0.24	10	0.58	0.16	1.61	10	0.03	0.00	0.25
	30	10	0.42	0.02	1.61	10	4.15	0.52	13.31	10	0.04	0.00	0.25
	20	10	1.22	0.09	3.56	10	2.06	0.09	5.49	10	0.18	0.00	0.63
7	25	10	16.05	0.44	67.70	10	59.54	3.55	200.74	10	2.73	0.02	12.05
	30	10	115.52	8.30	585.52	9	833.93	43.56	—	10	7.85	0.05	25.03

ほどの分枝限定法の実験と計算機環境を同じにするために，再実験したものである．プログラムは C 言語で記述し，Visual C++ 6.0 でコンパイルし，いくつかの m，n や 0-1 行列の 1 の密度に対して実行した．問題は同一パターンを持つ列をまとめて縮約しているので，実際に解いている列の数は n 以下であることに注意する．各パラーメタにつき 10 回試行し，#は 10 回のうち 1 時間内に解けた問題数と CPU 時間の平均・最小・最大値である．

表 1 より，DP-MBP は商用ソルバを直接適用するよりも圧倒的に優れている．また行列が密または疎であるときには，同一パターンが出現しやすく，縮約のために効果的に解ける．しかし，DP-MBP は，実行時間および $f(S)$ の値を記憶するメモリ確保に 2^n の影響を直接受ける．したがって，本実験で使用した計算機環境では，特殊な設定などをしない限り，整数値 $f(S)$ を確保できるサイズの限界が 2^{31} までであり，縮約しても 32 列以上になる問題を解くことはできない．

このような現状があるため，本論文では，DP-MBP では解決不可能なサイズの問題でも解くことができる分枝限定法のアルゴリズム開発に主目的を置いている．

2. 下界値戦略

2.1 基 本 戦 略

下界値を求めるにあたって，1 の幅の開始だけをできるだけ遅くする最遅拘束開始問題（LSBP: Latest Start Binding Problem）を考える．同様に 1 の幅の終了だけをできるだけ早くする最早拘束終了問題（FFBP: Fastest Finish Binding Problem）も考えられる．このとき，問題 P の最適値を opt(P) とすれば，式 (3) の大小関係が成立し，opt(MBP) の下界値を与えることができる．

$$mn - \mathrm{opt(LSBP)} - \mathrm{opt(FFBP)} = mn - 2\ \mathrm{opt(LSBP)} \leq \mathrm{opt(MBP)} \tag{3}$$

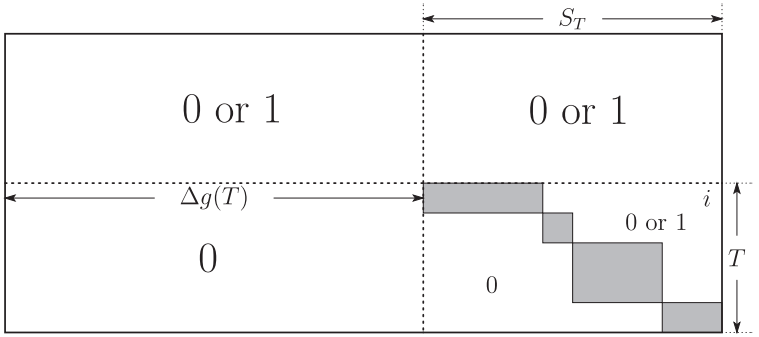

図 2　DP-LSBP の過程，行集合 T，列集合 S_T，増分 $\Delta g(T)$
Fig. 2　The process of DP-LSBP, T, S_T and $\Delta g(T)$.

等号の成立は，LSBP と FFBP に本質的な違いがないことから明らかであり，今後は LSBP のみを考える．

Kataoka らは，LSBP が列ではなく行を並べ替える問題に帰着できることを示し，動的計画法のアルゴリズム DP-LSBP を開発した．行集合 T（$\subseteq M$）に属する行（以降 “T の行” と記述）に対して，次を定義する．

$g(T)$　は，T の行と N の列で構成される行列に対する LSBP の最適値．

$\Delta g(T)$　T の行と N の列で構成される行列における 0 列の数 ($= |\{j|a_{ij} = 0, \forall i \in T\}|$)．

このとき，DP-LSBP の漸化式は，式 (4) のようになる．

$$g(T) := \max_{i \in T} \{g(T \setminus \{i\})\} + \Delta g(T) \tag{4}$$

初期状態は $g(\emptyset) := 0$ であり，最適値は $g(M)$ である．

ここまでが Kataoka らの概要であるが，以降の節において列に関して制約を加えた LSBP を考えるため，ここで必要な定義などを行っておく．

DP-LSBP は，**図 2** のように T の行を行列の下に集めて考えると分かりやすい．このとき，T の行の中に 1 が含まれている列集合を S_T（$:= \cup_{i \in T}\{j|a_{ij} = 1,\ j \in N\}$）とする．DP-LSBP は，$T$ を拡張しながら，T の行と S_T の列で構成される部分行列に対し，随時 LSBP の最適値を求めていることにほかならない．動的計画法の最適性の原理より，$T \setminus \{i\}$ の行までは最適な行の並びになっており，最後に行 i が追加されて T になるとき，列集合 $\{j|a_{ij} = 1, a_{tj} = 0, \forall t \in T \setminus \{i\}\}$ が追加されて S_T となる．このとき，追加される行と列

集合で対角状にブロックが構成される（図 2 の陰影部）．追加される列集合が空のときは，直前行のブロックを i 行目にコピーしてブロックを構成する．$\Delta g(T)$ の正当性は，LSBP の最適解において，ブロック内の成分はすべて 1 であることから導かれる．そうでなければ，LSBP の値がさらに改善できるからである．また，増分（$(g(T) - g(T \setminus \{i\}) = \Delta g(T)$）は，新たに追加されたブロックの左側にある 0 の数となるので，$\Delta g(T)$ は，S_T を用いて，式 (5) のように再定義できる．

$$\Delta g(T) := n - |S_T| = |\{j | a_{ij} = 0, \forall i \in T\}| \tag{5}$$

DP-LSBP も $O(2^m)$ のメモリと計算時間を要するが，行の数が比較的小さい場合であれば簡単に解け，MBP の下界値を求めることができる．次節以降では，分枝限定法で用いる下界値を得るために，列に関して制約を加えた LSBP を考える．列に制約がついた問題でも基本的には式 (4) による動的計画法で解くことができる．制約がつくことで改訂が必要な部分は，増分を計算するための $\Delta g(T)$ の項，つまり式 (5) だけである．

2.2　分枝限定法のための下界値計算

通常，MBP のような問題で分枝限定法を考えるとき，いくつかの列に対して割り当てる（割り当てない）位置を固定しながら下界値を計算をする．しかし，MBP では，たとえば割り当てる位置を固定した列によっては，それらの列の間で他の列をどのように割り当てても上下界値の違いがほとんど生じなくなることもあるため，列挙木を刈り取れないという現象が起こってしまう．そこで本論文では，いくつかの列に対し，前半部あるいは後半部のどこかに割り当てられていればよいという緩い割当ての概念を導入する．緩い割当てだと当初の下界値はあまり良くないが，割当てが進むにつれ，次第に上下界値の違いが生じるようになり，結果的には列挙木を大きく刈り取ることができる．また，目的関数値に影響を与えないような近接位置での列交換などは，同じ子問題として扱えるので，列挙木そのものも小規模にできる．

与えられた行列の中央部の位置 $\bar{k} := \lceil n/2 \rceil$ を境界（$\bar{k}$ は前半部最後の位置）として，S_f を前半部に割り当てる列の集合（前半列集合），S_ℓ を後半列集合とする．ただし，$S_f \cup S_\ell \subseteq N$ かつ $S_f \cap S_\ell = \emptyset$ である．このように列を分配した LSBP を LSBP($S_f|S_\ell$) と記述する．下界値を計算するために FFBP($S_f|S_\ell$) が必要になるが，このときは LSBP($S_\ell|S_f$) を考え，境界を $\lfloor n/2 \rfloor$ とすればよい．

図 3 は，DP-LSBP が行集合 T まで進行している状況を示している．このとき，前半列集合のうち S_T に含まれる列集合を S_{f_T}（$:= S_f \cap S_T$，図 3 では $\{j_1, j_2\}$），後半列集合のうち $N \setminus S_T$ に含まれる列集合を S_{ℓ_T}（$:= S_\ell \setminus S_T$，図 3 では $\{j_3, j_4, j_5\}$）とする．このとき，LSBP($S_f|S_\ell$) の動的計画法における $\Delta g(T)$ を評価したいが，理解しやすくするため，LSBP($S_f|\emptyset$) と LSBP($\emptyset|S_\ell$) に分けて考える．

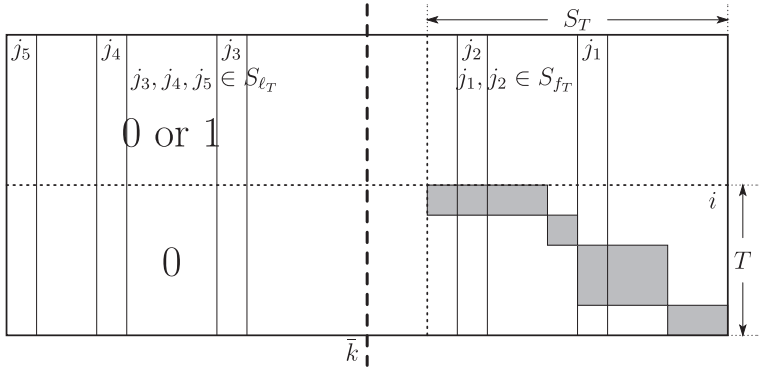

図 3　$|S_T| \leq n - \bar{k}$ のときの前半列集合 $S_{f_T} \subseteq S_T$ と後半列集合 $S_{\ell_T} \subseteq N \setminus S_T$
Fig. 3　$S_{f_T} \subseteq S_T$ and $S_{\ell_T} \subseteq N \setminus S_T$ when $|S_T| \leq n - \bar{k}$.

LSBP($S_f|\emptyset$) では（図 3 において列 j_3，j_4，j_5 を無視する）$|S_T| \leq n - \bar{k}$ のとき，前半に割り当てるべき列集合 S_{f_T} が空でないときに注意を要する．動的計画法の最適性の原理より $T \setminus \{i\}$ までの行が最適に並んでいるので，ここに行 i を追加するとき，増分を最大にするには，**図 4** のように S_{f_T} の列を $\bar{k}$ の位置から前半に向かって配置しなければならないので，$\Delta g(T) := \bar{k} - |S_{f_T}|$ となる．

このとき，もし列 j_2 が行 i に関与しないとき，たとえば図 3 で j_1 の左隣列が j_2 だったとき，a_{ij_1} や a_{ij_2} の値次第で $\Delta g(T)$ の評価法はもっと複雑な式になる．しかし，動的計画法の最適性の原理により，$\Delta g(T)$ が $\bar{k} - |S_{f_T}|$ よりも大きく評価できるような行 i は，最後に T に入って最適解を構成することはなく，もっと早い段階で T に入ってくる．したがって，$\Delta g(T) := \bar{k} - |S_{f_T}|$ のままでも，最適値は正しく求められる．

一方，LSBP($\emptyset|S_\ell$) では（図 3 において列 j_1，j_2 を無視する），S_T の列を割当て済みのときに，まだ割り当てられていない後半列集合 S_{ℓ_T} があり，その列数が後半部に残っているスペースを超えてしまう $|S_T| + |S_{\ell_T}| > n - \bar{k}$ のときに注意を要する．このときは，無理にでも S_{ℓ_T} の列を後半部に入れざるをえず，しかも LSBP として拘束開始時刻を最も遅くするには，位置 $\bar{k} + 1$ から後半部に向かって S_{ℓ_T} の列を割り込ませて配置するしかない（**図 5**）．

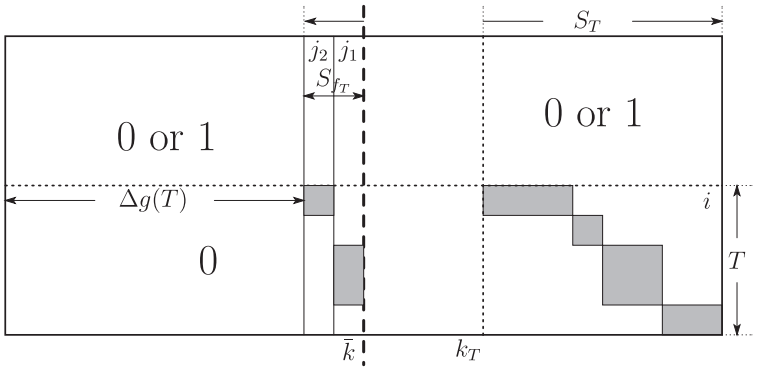

図 4　LSBP($S_f|\emptyset$) の T における増分 $\Delta g(T)$
Fig. 4　The $\Delta g(T)$ for T on LSBP($S_f|\emptyset$).

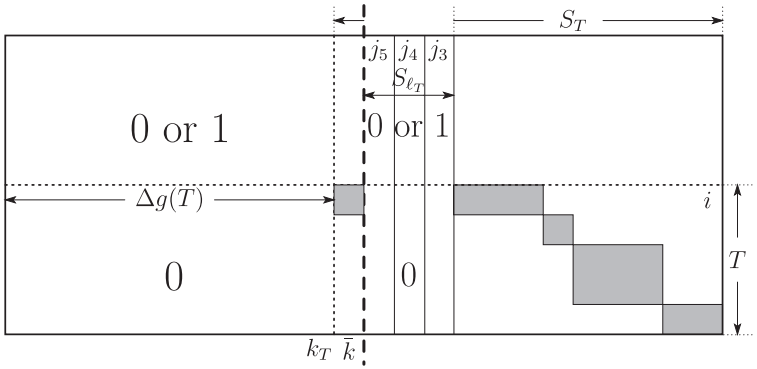

図 5　LSBP($\emptyset|S_\ell$) の T における増分 $\Delta g(T)$
Fig. 5　The $\Delta g(T)$ for T on LSBP($\emptyset|S_\ell$).

前半列集合・後半列集合が両方とも非空の場合でも，これまでの議論から DP-LSBP をもとに，増分 $\Delta g(T)$ を改訂するだけで LSBP($S_f|S_\ell$) も解くことができる．実際にプログラミングするときには，T の行を集めたりするのではなく，各 T に行 i を追加する際に考慮すべき列（$\{j|a_{ij}=1,\ a_{tj}=0,\ t\in T\}$）の割当て開始位置 k_T の値に注意しながら，以下の手順に従って $\Delta g(T)$ を定める．$\Delta g(T)$ の定め方は，k_T が後半か前半かによって大きく場合分けされ，当初 k_T は後半部にあり，$k_T := n - |S_T| + |S_{f_T}|$ で与える．S_T が拡張され，k_T が前半部に入るときは，以下の手順に従って $k_T := n - |S_T| - |S_{\ell_T}|$ に更新する．

- k_T が後半部にある場合
  - $S_{f_T} = \emptyset$ かつ $k_T - \bar{k} \geq |S_{\ell_T}|$ のとき，前半部に割り当てるべき列はなく，S_{ℓ_T} を後半部に割り当てる十分なスペースもあるので，増分はオリジナルのまま $\Delta g(T) := n - |S_T|$ でよい．
  - $S_{f_T} = \emptyset$ かつ $k_T - \bar{k} < |S_{\ell_T}|$ のとき，境界 $\bar{k}$ から後半部に S_{ℓ_T} の列を割り当てなければならないので，$\Delta g(T) := n - |S_T| - |S_{\ell_T}|$ となり，この後 k_T は前半部に入り，$k_T := n - |S_T| - |S_{\ell_T}|$ とする．
  - $S_{f_T} \neq \emptyset$ かつ $k_T - \bar{k} \geq |S_{\ell_T}|$ のとき，境界 $\bar{k}$ から前半部に S_{f_T} の列を割り当てなければならないので，$\Delta g(T) := \bar{k} - |S_{f_T}|$ となる．
  - $S_{f_T} \neq \emptyset$ かつ $k_T - \bar{k} < |S_{\ell_T}|$ のとき，境界 $\bar{k}$ から前半部に S_{f_T} の列を，後半部に S_{ℓ_T} の列を割り当てなければならないので，$\Delta g(T) := n - |S_T| - |S_{\ell_T}|$ となり，この後 k_T は前半部に入り，$k_T := n - |S_T| - |S_{\ell_T}|$ とする．

位置 k_T が前半部に入るまで動的計画法が進行したときには，S_{f_T} の列はすでに前半部に割当て済みで，残りの前半列集合 $S_f \setminus S_{f_T}$ の列は自動的に前半部に入ってくるので，新たな問題は起こらない．このときは，まだ割り当てられてない後半列集合 S_{ℓ_T} が空か否かで場合分けをすればよい．

- k_T が前半部にある場合，
  - $S_{\ell_T} = \emptyset$ のとき，DP-LSBP をそのまま適用すればよいので，増分はオリジナルと同じ $\Delta g(T) := n - |S_T|$ である．
  - $S_{\ell_T} \neq \emptyset$ のとき，S_{ℓ_T} の列を境界 $\bar{k}$ の後に割り当てなければならないので，増分は $|S_{\ell_T}|$ だけシフトして，$\Delta g(T) := n - |S_T| - |S_{\ell_T}|$ となる．

以上をまとめると，LSBP($S_f|S_\ell$) を解くための動的計画法は，式 (4) を基調とし，増分を式 (6) のように改訂すればよい．

$$\Delta g(T) := \begin{cases} \bar{k} - |S_{f_T}| & k_T - \bar{k} \geq |S_{\ell_T}| \text{ かつ } S_{f_T} \neq \emptyset \text{ のとき} \\ n - |S_T| - |S_{\ell_T}| & k_T - \bar{k} < |S_{\ell_T}| \text{ のとき} \\ n - |S_T| & \text{それ以外のとき} \end{cases} \tag{6}$$

3. 分枝限定法

2 章において，列を前半部あるいは後半部のどこかに割り当てる分枝ルールを提案し，下界値の求め方について説明した．しかし，この分枝方法は，列を特定の位置には固定しない緩い割当て方法であるため，もしすべての列が最適に前半部・後半部に分けられていたとしても，そのときの下界値はときとして最適値よりも小さくなる可能性があり，分枝限定法がうまく機能しなくなる．本章では，このような欠点を解決し，分枝限定法を完全なものにする．

3.1 最適解になりえない列割当て

列の割当てを進めていくと，最適解にはなりえない列割当てをしてしまうことがある．このような場合は列挙木を刈り取ることができる．P_j を列 j（$\in N$）と同一パターンを持つ列集合とする．特に P_f，P_ℓ，P_{f_T}，P_{ℓ_T} と記述するとき，それらは $P_X := \bigcup_{j \in S_X} P_j$ を意味する．ただし，X は，それぞれ f，ℓ，f_T，ℓ_T である．

最適解になりえない列割当て 1：同一パターンを持つ 2 つの列に対し，列 j_1 を前半部，列 j_2 を後半部に割り当てるとき，補題 1 からこのパターンを持つ列集合の列は，境界 $\bar{k}$ をまたいで割り当てられていなくてはならない．したがって，さらに別の同一パターンを持つ列 j_3，j_4 があり，これらが $j_1, j_3 \in S_f$，$j_2, j_4 \in S_\ell$ かつ $P_{j_1} = P_{j_2} \neq P_{j_3} = P_{j_4}$ となっているときは，最適解を得ることはなく，列挙木を枝刈りすることができる．

最適解になりえない列割当て 2：前半列集合 S_f および後半列集合 S_ℓ を割り当てるとき，補題 1 から実際には前半部に $|S_f|$ 本よりも，後半部には $|S_\ell|$ 本よりも多い列が割り当てられる可能性がある．たとえば**図 6** では，j_1 が後半に割り当てる列，j_2，j_3，j_4，j_5 が前半に割り当てる列であり，しかも j_1，j_2，j_3 は同一パターン（$P_{j_1} = P_{j_2} = P_{j_3}$）であり $|P_{j_1}| = 5$ とする．また $|P_{j_4}| = 2$，$|P_{j_5}| = 3$ とする．このとき，P_{j_1} は前半列，後半列両方を含んでいるので境界をまたいで割り当て，P_{j_4} と P_{j_5} の列は連続させて前半に割り当てなければならないので，その結果，少なくとも 7 本の列が前半部に割り当てられることになる．

$Least_f$ を S_f が与えられたときに，前半部に割り当てられるべき列の最小数とする．まずどのパターンが境界をまたいで割り当てられるかを考える．そのようなパターンを $\bar{P}$ とする．もし，図 6 のように，$P_f \cap P_\ell \neq \emptyset$ のとき，$\bar{P}$ は $P_f \cap P_\ell$ であり唯一に定まる．また，$P_f \cap P_\ell = \emptyset$ のとき，$\bar{P}$ は後半部に最も突き出して割り当てられるパターン，すなわち $\bar{j} := \arg\max_{j \in S_f} \{|P_j \setminus S_f|\}$ とすると $\bar{P} := P_{\bar{j}}$ である．したがって，$Least_f$ は，次のように算定できる．

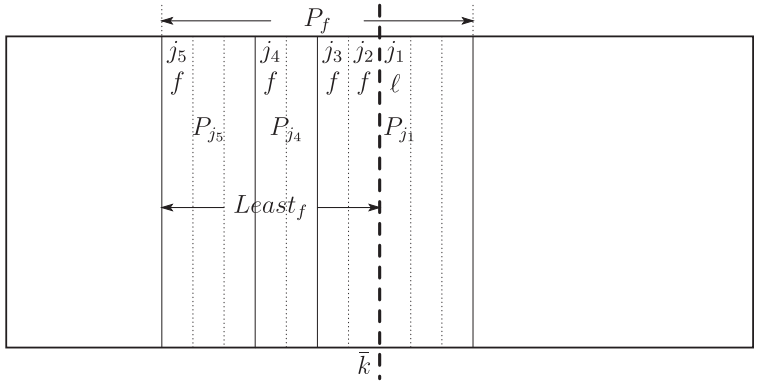

図 6　P_f の列の割当てと $Least_f$
Fig. 6　The assignment of P_f and $Least_f$.

$$Least_f := |P_f| - |\bar{P} \setminus S_f| \tag{7}$$

同様に，$Least_\ell$ についても，$P_f \cap P_\ell \neq \emptyset$ であれば，$\bar{P} := P_f \cap P_\ell$ であり，$P_f \cap P_\ell = \emptyset$ であれば，$\bar{j} := \arg\max_{j \in S_\ell} \{|P_j \setminus S_\ell|\}$ とすると，$\bar{P} := P_{\bar{j}}$ である．したがって，$Least_\ell$ も次のように算定できる．

$$Least_\ell := |P_\ell| - |\bar{P} \setminus S_\ell| \tag{8}$$

このときに，$Least_f > \bar{k}$ または $Least_\ell > n - \bar{k}$ であれば，このような列割当てからは最適解が得られることはなく，列挙木を枝刈りすることができる．

3.2 解の確定

もし，$Least_f = \bar{k}$（かつ $Least_\ell \leq n - \bar{k}$）または $Least_\ell = n - \bar{k}$（かつ $Least_f \leq \bar{k}$）が成り立っている場合は，すべての列が前半部または後半部のどちらに割り当てられるかが確定したことになる．しかし，ここでの分枝ルールは，各列を特定の位置に固定してはいないので，これだけの制約からは解を確定することができない．

このような場合，DP-MBP を前半部・後半部それぞれ個別に適用して最適解を確定する．DP-MBP は $O(2^n)$ の計算量であるため，それぞれの問題の列数がほぼ半分になっていれば，単純な算定でも列数が 20 を超えると，まともに DP-MBP を適用するよりも 1/1000 以下の手間で済む．また半分にすることで，ゼロ行もいくつか発生しうるので，さらに効率良く解を確定することができる．

まず，すべての列に対し前半部・後半部のどちらの割当てになるか S_f^N，S_ℓ^N（$S_f^N \cup S_\ell^N = N$，

$S_f^N \cap S_\ell^N = \emptyset$) を確定する．$Least_f = \bar{k}$ のとき，$S_f^N := P_f \setminus (\bar{P} \setminus S_f)$ そして $S_\ell^N := N \setminus S_f^N$ とする．一方，$Least_\ell = n - \bar{k}$ のとき，$S_\ell^N := P_\ell \setminus (\bar{P} \setminus S_\ell)$ そして $S_f^N := N \setminus S_\ell^N$ とする．

このとき，前半部・後半部それぞれにおいて，DP-MBP を適用して最適値 $f(S_f^N) + f(S_\ell^N)$ を得るためには，初期値と増分項を前半部と後半部とで次のように改訂すればよい．

前半部の場合，$S \subseteq S_f^N$ とし，初期値は $f(\emptyset) := 0$ である．そして増分項を次のようにする．

$$\Delta f_i(S, j) := \begin{cases} 1 & a_{ij} = 1 \text{ のとき} \\ 1 & 0 < \sum_{s \in S} a_{is} < \sum_{s \in N} a_{is} \text{のとき} \\ 0 & \text{それ以外のとき} \end{cases} \tag{9}$$

後半部の場合，$S \subseteq S_\ell^N$ とし，初期値は $f(\emptyset) := m - |\{i | \sum_{j \in S_\ell^N} a_{ij} = \sum_{j \in N} a_{ij}\}| - |\{i | \sum_{j \in S_f^N} a_{ij} = \sum_{j \in N} a_{ij}\}|$ とする．そして増分項は次のようにする．

$$\Delta f_i(S, j) := \begin{cases} 1 & a_{ij} = 1 \text{ のとき} \\ 1 & 0 < \sum_{s \in S \cup S_f^N} a_{is} < \sum_{s \in N} a_{is} \text{のとき} \\ 0 & \text{それ以外のとき} \end{cases} \tag{10}$$

3.3 パターンを考慮に入れた LSBP ($S_f|S_\ell$)（PCLSBP ($S_f|S_\ell$)）

最後に，$Least_f < \bar{k}$ かつ $Least_\ell < n - \bar{k}$ のときは，LSBP($S_f|S_\ell$)（と FFBB($S_f|S_\ell$)）を解くことで下界値を得ることができる．このとき，同一パターンの連続出現性を考慮に入れると，より強い下界値が期待できる．この問題をパターンを考慮に入れた LSBP($S_f|S_\ell$)（PCLSBP($S_f|S_\ell$)）と呼ぶことにする．基本的に PCLSBP($S_f|S_\ell$) は LSBP($S_f|S_\ell$) であり，S_{f_T} を P_{f_T} に，S_{ℓ_T} を P_{ℓ_T} に置き換えればよい．P_{ℓ_T} の列を後半に割り当てるスペースがないときは，たかだか 1 パターンが連続出現性を失うが，MBP の下界値を得ることが目的なので問題ない．このスペースに余裕があるときは，前半部に $Least_{f_T}$ だけ突き出すことに注意すればよい．

ここでも，動的計画法進行中において，後続するブロックの構成開始位置 k_T に注意しながら，後半部にあるときは $k_T := n - |S_T| + |P_{f_T}|$ を用い，前半部に入ると $k_T := n - |S_T| - |P_{\ell_T}|$ を用いる．これらのことをまとめると，PCLSBP($S_f|S_\ell$) を解くときの増分項 $\Delta g(T)$ は式 (11) のようになる．

BB-MBP($S_f|S_\ell$)
/* meaningless restriction 1 */
If there exist four columns $j_1, j_1' \in S_f$, $j_2, j_2' \in S_\ell$ such that $P_{j_1} = P_{j_2} \neq P_{j_1'} = P_{j_2'}$, then return.
/* meaningless restriction 2 */
Compute the values $Least_f$ and $Least_\ell$ by using (7) and (8).
If $Least_f > \bar{k}$ or $Least_\ell > n - \bar{k}$, then return.
/* obtaining a feasible solution */
If $Least_f = \bar{k}$, let $S_f^N := P_f \setminus (\bar{P} \setminus S_f)$ and $S_\ell^N := N \setminus S_f^N$.
Else if $Least_\ell = n - \bar{k}$, let $S_\ell^N := P_\ell \setminus (\bar{P} \setminus S_\ell)$ and $S_f^N := N \setminus S_\ell^N$.
　Obtain $f(S_f^N)$ by DP-MBP of (2) and (9), and obtain $f(S_\ell^N)$ by DP-MBP of (2) and (10).
　The upper-bound is $z_U := f(S_f^N) + f(S_\ell^N)$.
　If $\tilde{z} > z_U$, then improve the current best solution and value, $\tilde{z} := z_U$.
　Return.
/* lower-bound */
(Both $Least_f < \bar{k}$ and $Least_\ell < n - \bar{k}$ are satisfied.)
Solve PCLSBP($S_f|S_\ell$) and PCFFBB($S_f|S_\ell$) (We solve PCLSBP($S_\ell|S_f$) instead of PCFFBB($S_f|S_\ell$).) by DP-LSBP of (4) and (11).
The lower-bound is
　$z_L := mn - \text{opt(PCLSBP}(S_f|S_\ell)) - \text{opt(PCFFBP}(S_f|S_\ell))$ by (3).
If $\tilde{z} \leq z_L$, then the current branching node is pruned, and return.
/* branching */
Choose a column $j \in N \setminus (S_f \cup S_\ell)$.
Call BB-MBP($S_f \cup \{j\}|S_\ell$).
Call BB-MBP($S_f|S_\ell \cup \{j\}$).
Return.

図 7　分枝限定法アルゴリズム BB-MBP($S_f|S_\ell$)
Fig. 7　The branch-and-bound algorithm: BB-MBP($S_f|S_\ell$).

$$\Delta g(T) := \begin{cases} \bar{k} - Least_{f_T} & k_T - \bar{k} \geq |P_{f_T}| - Least_{f_T} + |P_{\ell_T}| \\ & \quad \text{かつ } S_{f_T} \neq \emptyset \text{ のとき} \\ n - |S_T| - |P_{\ell_T}| & k_T - \bar{k} < |P_{f_T}| - Least_{f_T} + |P_{\ell_T}| \text{ のとき} \\ n - |S_T| & \text{それ以外のとき} \end{cases} \tag{11}$$

以上で分枝限定法を完成させるすべての準備が整った．**図 7** にそのアルゴリズム BB-MBP($S_f|S_\ell$) を示す．メイン関数において，BB-MBP($\emptyset|\emptyset$) を呼ぶ．アルゴリズム進行中の最良値 $\tilde{z}$ は大域領域にあるものとする．

表 2　BB-MBP の計算機実験結果（秒）
Table 2　The computational results of BB-MBP (sec.).

		sparse (25%)				middle (50%)				dense (75%)			
m	n	#	ave	min	max	#	ave	min	max	#	ave	min	max
	30	10	5.51	1.31	20.20	10	3.88	0.09	19.84	10	6.91	0.17	31.41
	35	10	12.30	3.63	28.47	10	23.94	1.97	133.25	10	14.48	0.23	60.45
5	40	10	18.43	6.95	44.34	10	43.53	5.19	192.77	10	33.62	0.55	196.59
	45	10	52.93	17.34	143.53	10	67.20	8.30	238.72	9	45.24	0.33	—
	50	10	81.62	21.91	327.09	10	107.17	14.09	391.67	8	190.64	0.39	—
	30	8	958.70	106.16	—	9	116.49	8.33	—	10	307.18	0.64	2,296.00
7	35	2	2,513.71	1,516.34	—	8	529.42	10.19	—	9	172.53	10.22	—
	40	0	—	—	—	6	1,483.36	118.92	—	9	444.82	5.05	—
	30	4	2,559.99	1,638.97	—	7	555.69	125.48	—	10	717.13	12.84	2,025.59
9	35	0	—	—	—	4	1,585.47	463.81	—	8	861.54	1.22	—
	40	0	—	—	—	0	—	—	—	5	854.94	144.42	—

3.4　計算機実験と結果

BB-MBP の計算機実験結果を**表 2** に示す．パラメータなどの表記は，DP-MBP の結果である表 1 に準じている．

本論文内では詳しい議論を省略しているが，分枝限定法に入る前に，局所探索による近似解法を適用して上界値を求めている．本実験程度のサイズでは，その実行時間は 1 秒にも満たないため，CPU 時間にも含めていない．また，単純な局所探索法では，時間をかけても精度の良い解は得られず，有効な解の更新は分枝限定法の中で行われている．列の選択方法については，1 の密度の違いにより試行してみたが，明確な違いが観察されなかったので，ここでは問題の生成時につけた列番号順に従っている．

動的計画法では，縮約後の問題サイズが列数 30 を超えると解けないので，分枝限定法では $n \geq 30$ を実験対象とする．CPU 時間は，10 回の試行のうち，1 時間以内に解けたものを対象に平均値を示している．

動的計画法である DP-MBP では，縮約後の列数が 30 以上になると不可能だったものが，分枝限定法である BB-MBP では解くことに成功している．ただし，列数が 30 のときには動的計画法の方が効率が良く，分枝限定法・動的計画法どちらを使えばよいかは，縮約後の問題サイズを見てから適用法を選ぶべきであろう．

分枝限定法と動的計画法の違いとして，BB-MBP は行列が疎なときには効率良く解けない．これは，使用している下界値戦略が非ゼロ要素を一方の側に寄せて計算するものであるため，行列が疎だと下界値として弱くなってしまうからである．また，LSBP の最適値は，列数が多くなってもそれほど大きくならない傾向があるので，下界値としては相対的に良くなっていく．反対に行列が密なときには，行列の中ほどで列を入れ替えても目的関数に及ぼす影響が少ない．したがって，$m = 5$ のときでは，疎な問題の方が，密な問題よりも効率良く解けていることが観察される．

動的計画法にしても分枝限定法にしても，行の数の影響は大きい．行の数が多くなると，同一パターンが現れる可能性も低くなり，問題の縮約や同一パターンを考慮した下界値の強化の効果が薄れるため，実行時間にも指数的な増大の影響を受ける．

4. 結　　論

最小拘束問題は，単純であり応用の幅も広く興味深い問題であるが，厳密に解くことは非常に難しい．本論文では，Kataoka ら[9] をもとに分枝限定法のアルゴリズムを開発した．そこでの分枝ルールは，いくつかの列を前半部あるいは後半部に割り当てるものである．この分枝ルールによって，局所的な列の入れ替えによる同一目的関数を持つ解を無駄に探索することはなくなるが，列を特定の位置に固定するものでないため，そのままでは分枝限定法として機能しない．

分枝限定法として完全なものにするために，最適解になりえない列割当ての条件や，早期にすべての列を前半部・後半部に分類できることを判定して半分ずつ解く方法や，パターンを考慮した下界値強化法などを提案した．

計算機実験の結果，本研究の分枝限定法では，列挙木の多くを刈り取り，中規模サイズの問題を解くことに成功した．しかし行数が多いときなどでは，この問題を厳密に解くことは難しいと考えられる．

参 考 文 献

1) Booth, K.S. and Lueker, G.S.: Testing for the Consecutive Ones Property, Interval Graphs and Graph Planarity Using *PQ*-Tree Algorithms, *Journal of Computer and Systems Sciences*, Vol.13, pp.335–379 (1976).
2) Burke, E.K., McCollum, B., Meisols, A., Petrovic, S. and Qu, R.: A Graph-Based Hyper-Heuristic for Educational Timetabling Problems, *European Journal of Operational Research*, Vol.176, pp.177–192 (2007).
3) Dammark, A., Elloumi, A. and Kamoun, H.: Classroom Assignment for Exam Timetabling, *Advances in Engineering Software*, Vol.37, pp.659–666 (2006).
4) Hoogeveen, J.A. and van de Velde, S.L.: Annotated Bibliographies in Combinatorial Optimization, *Sequencing and Scheduling*, Dell'Amico, M., Maffioli, F. and

Martello, S. (Eds.), Wiley (1997).
5) Head, C. and Shaban, S.: A Heuristic Approach to Simultaneous Course/Student Timetabling, *Computers and Operations Research*, Vol.34, pp.919–933 (2007).
6) Reinelt, G.: Research and Exposition in Mathematics 8, *The Linear Ordering Problem – Algorithms and Applications*, Hofmann, H.H. and Wille, R. (Eds.), Verlag (1985).
7) IBM Tokyo Research Laboratory: Air-Crew Scheduling (online). http://www.research.ibm.com/trl/projects/optsim/opt/ACS
8) Ikegami, A. and Niwa, A.: A Subproblem-Centric Model and Approach to the Nurse Scheduling Problem, *Mathematical Programming*, Vol.97, pp.517–541 (2003).
9) Kataoka, S. and Kajiya, M.: Dynamic Programming and Lower Bound Approaches to the Minimum Binding Problem, *International Journal of Systems Science*, Vol.35, pp.629–636 (2004).
10) Nemhauser, G.L. and Wolsey, L.A.: *Integer and Combinatorial Optimization*, Wiley (1988).
11) Schrijver, A.: *Theory of Linear and Integer Programming*, Wiley (1986).

(平成 20 年 9 月 25 日受付)
(平成 21 年 5 月 13 日採録)

片岡　靖詞（正会員）

昭和 60 年早稲田大学理工学部工業経営学科卒業，昭和 62 年早稲田大学大学院理工学研究科修士課程修了，平成 2 年早稲田大学大学院理工学研究科博士課程単位取得後退学．平成 5 年学位取得，博士（工学）．平成 2 年より防衛大学校情報工学科助手・講師を経て，現在，准教授．各種組合せ最適化問題のアルゴリズム開発に従事．日本オペレーションズ・リサーチ学会会員．

坂森　義成

平成 9 年防衛大学校情報工学科卒業．同年陸上自衛隊入隊，幹部候補生学校卒業後，第 5 通信大隊勤務．平成 12〜14 年防衛大学校理工学研究科前期課程，工学修士．その後，補給統制本部勤務，技術研究本部勤務を経て，平成 21 年から補給統制本部に勤務．