システム/制御/情報，Vol. 50, No. 9，pp. 350–356，2006

「『堅く柔らかく … 数理アプローチ再訪』特集号」　　解　説

分枝限定法 ── さらなる計算効率の希求 ──

柳浦 睦憲[†]・野々部 宏司[‡]

1　はじめに

京都大学総合博物館の企画展に「工学の面白さを一般の人に伝えるようなものを出せないか」と頼まれ，一目見て直感的に理解できるものがよいと考え，当時研究室で取り組んでいた矩形パッキング問題に対するメタ戦略アルゴリズムのデモを展示することになった．さらに，コンピュータの画面上のデモだけよりも，実際に触れることのできるものをということで，問題の難しさを自分の手で体験できるような木製パズルを作ろうということになり，第1図のようなパズルができあがった．なるべく隙間なく詰めることがこのパズルの目的であるが，このパズルの特徴は，詰め込む矩形の組合せを自由に選べるところにあり，その選び方によって答えが変わる．この点が普通のジグソーパズルのように答えが予め決まっているパズルとの大きな違いである．ただし，選んだ矩形の組合せによっては隙間が残る配置が最適かもしれず，そのような場合には，最適値を知らない限り自分の出した答えが最適であることを確認できず達成感が得られない．そこで，このパズルの最適解を求めるためのアルゴリズムも作ろうということになった．その概略については後述するが，このように，近似解ではなく「厳密な最適解」が必要になることがある．この他にも，例えば，選挙区の区割問題 [22] のようにルールやシステム性能の限界を見極めるために，厳密解が必要になることがある．

第 1 図　木製のパッキングパズル

厳密な最適解を得るための戦略としては**分枝限定法**（branch-and-bound method）と**動的計画法**（dynamic programming）が代表的であるが，本稿では前者に焦点を当てる．分枝限定法は，計算困難な組合せ最適化問題の実用的解決法として広く利用されてきた．今日，大規模な問題例に対しては**局所探索**（local search）や**メタ戦略**（メタヒューリスティクス，metaheuristics）[26] などの近似解法が主流になってきているが，問題の規模やタイプによっては厳密解を求めてしまう方が速いことがある．また，分枝限定法の探索を途中で打ち切れば近似解法としても利用でき，この場合，最適値は分からなくとも最適値の下界を知ることができる（最小化問題を仮定している）．このような観点からも，分枝限定法は組合せ最適化の基本戦略としてなくてはならないものであるが，計算機の急速な進歩とアルゴリズムの発展のおかげで厳密解法が有効に働く規模が飛躍的に大きくなったこともあり，近年再び脚光を浴びている．

本稿では，まず分枝限定法の考え方を手短かに概観したのち，アルゴリズムを設計する際の注意点について述べるとともに，分枝限定法の発展形である分枝カット法と分枝価格法の基本的なアイデアを説明する．さらに，矩形パッキング問題への応用例や，分枝限定法に関連するその他の話題をいくつか紹介する．

2　分枝限定法の基礎

分枝限定法は，問題をいくつかの小規模な問題に分割し，その全てを解くことで等価的に元の問題を解くという考え方に基づいている．具体例を用いる方が分かりやすいので，以下，**一般化割当問題**（generalized assignment problem, GAP）を対象として説明を行うことにする．GAP は，与えられた n 個の仕事 $J=\{1,2,\ldots,n\}$ を m 個のエージェント $I=\{1,2,\ldots,m\}$ に割当てたとき，割当に伴うコストの総和を最小化する問題である．仕事 $j\in J$ をエージェント $i\in I$ に割当てたときのコスト c_{ij} と資源の要求量 a_{ij} (≥ 0)，および各エージェント $i\in I$ の利用可能資源量 b_i (>0) が入力として与えられる．ただし，これらの値はすべて整数値であるとする．各仕事は必ずいずれか 1 つのエージェントに割当てられなくてはならず，また，各エージェントに割当てられた仕事の資源要求量の総和は，そのエージェントの利用可能資源量を越えてはならない．仕事 j をエージェント i に割当

[‡] 名古屋大学大学院情報科学研究科
[‡] 法政大学工学部
Key Words: branch-and-bound, branch-and-cut, branch-and-price, combinatorial optimization

てるときに値 1，そうでないときに値 0 を取る 0-1 変数 x_{ij} を用いて，問題を以下のように定式化できる．

$$\min \quad f(x)=\sum_{i\in I}\sum_{j\in J}c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in J}a_{ij}x_{ij}\le b_i, \qquad \forall i\in I, \tag{2}$$

$$\sum_{i\in I}x_{ij}=1, \qquad \forall j\in J, \tag{3}$$

$$x_{ij}\in\{0,1\}, \qquad \forall i\in I,\ \forall j\in J. \tag{4}$$

小規模な問題への分割は，例えば，ある 1 つの変数の値を 0 または 1 に固定し，それぞれの場合を個別に考察することによって実現できる．このように問題を分割する操作を**分枝操作**（branching operation）という．分枝操作を繰り返し行うことで，すべての場合を列挙することができるが，その過程は**生成木**と呼ばれる根付き木を用いて表現できる．生成木において，根は元の問題に対応し，その 2 つの子はそれぞれある変数の値を 0 または 1 に固定した 2 つの場合に対応する．その他の節点も同様である．よって，内部の節点は根からその節点への路に対応していくつかの変数の値を 0 か 1 に固定した問題に対応し，葉は全ての変数の値が定まった解に対応する．一部の変数の値が固定された問題を**部分問題**（partial problem）と呼び，この例の場合，部分問題もやはり GAP である．

分枝限定法では，実際に全ての葉を列挙するわけではなく，その一部分だけを生成する．生成木のうち実際に生成された節点のみから成るものを**探索木**（search tree）という．ある部分問題（節点）に対して，(i) その最適値，(ii) その部分問題が実行不可能であること，(iii) その部分問題の最適解が元の問題の最適解とはならないこと，のいずれかが分かれば，その部分問題をこれ以上調べる必要はなく，その下の節点（子孫と呼ぶ）の探索を省略できる（**終端**する（terminate）という）．これを**限定操作**（bounding operation）と呼ぶ．探索の過程で，まだ終端されていない部分問題を**活性**（active）であるといい，全ての部分問題が終端されたとき，つまり活性部分問題がなくなったとき，分枝限定法は厳密な最適解を与える（もしくは実行可能解が存在しないことを証明する）．

限定操作の例としては，**下界値テスト**が一般的である．制約 (4)「$x_{ij}\in\{0,1\}$」を「$0\le x_{ij}\le 1$」[1]に緩和した**線形計画**（linear programming, LP）問題（LP 緩和問題と呼ぶ）を考えると，この LP 緩和問題の最適値の小数部を切り上げた値は，GAP の最適値の**下界**（lower bound）を与える．すなわち，GAP の任意の実行可能解 x に対し，

$$f(x)\ge\lceil\ \text{LP 緩和問題の最適値}\ \rceil$$

が成り立つ．なお，LP 緩和問題の最適値は多項式時間で求まる．一方，**上界**（upper bound）値については，近似解法等を用いて実行可能解を 1 つでも得ることができれば，その目的関数値が最適値の上界となる．また，探索が生成木の葉に到達したときにも実行可能解が得られる場合がある．これまでの探索で得られている最良の上界値を**暫定値**（incumbent value）という．このようにして得られた上下界値を用い，「ある部分問題の下界値が暫定値以上であれば，その子孫を調べる必要はない」ことが結論できる．その部分問題の子孫を全て調べても，暫定値よりもよい目的関数値を持つ解は見つからないからである．以上が下界値テストである．（最大化問題の場合は上界と下界の役割が入れ替わる．）

なお，計算時間の都合などにより活性部分問題が残っている状態で探索を終了し，その時点での暫定値を出力する方法もしばしば用いられる．このとき，出力した値が最適である保証はなく，分枝限定法を近似解法として用いることになるが，この場合でも，活性部分問題の下界値の最小値が元の問題の下界値となり，出力値の精度を測る指標としてこれを用いることができる．

3 アルゴリズム設計の勘所

分枝限定法に基づいたアルゴリズムを設計する際，性能を大きく左右する重要なポイントがいくつか存在する．本節ではその基本的なものを紹介する．

まず分枝操作の実現方法について，前節では 1 つの変数 x_{ij} を 0 か 1 に固定する方法を紹介したが，このとき，どの変数を選ぶかが重要となる．例えば，LP 緩和を用いる場合は，その最適解 $\bar{x}$ において $\bar{x}_{ij}$ の値が 0.5 に近い変数で分枝するなどの戦略が広く用いられる．また，別の分枝操作も色々考えられる．例えば，あるエージェント集合 $I'\subset I$ に対して $\sum_{i\in I'}x_{ij}=1$ の場合と $\sum_{i\in I\setminus I'}x_{ij}=1$ の場合の 2 つに分割するなどである．この方法を採る場合，I' と $I\setminus I'$ の両方に $\bar{x}_{ij}>0$ なる変数が含まれるように集合 I' を選ぶとよい．分枝操作で生成される 2 つの部分問題のどちらにおいても $\bar{x}$ はその LP 緩和問題の実行可能解とはならず，よりよい下界値が得られる可能性が高まるからである．

一般に，よりよい下界値を得ることは，部分問題が最適解を含まない場合に下界値テストによって終端される可能性が高まり，探索木の節点数を減らす効果がある．しかし，よい下界値を得るために各節点で多くの計算時間を費やしたのでは分枝限定法全体の性能向上にはつながらない．したがって，よりよい下界値をできるだけ高速に得ることが望ましいが，通常，下界値の精度と計算速度の間にはトレードオフがあり，ここにどのような方法を用いるかが分枝限定法の成功の鍵といえる．

前節で LP 緩和問題を用いる方法について説明したが，LP 問題を解くにはある程度計算コストがかかるため，より手軽な方法として**ラグランジュ緩和**（Lagrangian relaxation）を用いる方法も広く利用されている．具体

[1] GAP の場合「$x_{ij}\ge 0$」と緩和したものも等価である．

的な方法は色々考えられるが，GAP の場合は，容量制約 (2) を緩和するものと割当制約 (3) を緩和するものの 2 つがよく知られている．制約 (2) に対するラグランジュ乗数ベクトルを $u=(u_1,u_2,\ldots,u_m)\in R_+^m$（$R_+$ は非負実数全体の集合）として，前者は以下のように定義できる．

$$\begin{aligned} L^{\mathrm{rec}}(u)=\min & \sum_{i\in I}\sum_{j\in J}(c_{ij}+u_ia_{ij})x_{ij}-\sum_{i\in I}u_ib_i \\ \text{s.t.} & \sum_{i\in I}x_{ij}=1, \quad \forall j\in J, \\ & x_{ij}\in\{0,1\}, \quad \forall i\in I,\ \forall j\in J. \end{aligned} \tag{5}$$

この問題の最適解 x は，各 j に対して $c_{ij}+u_ia_{ij}$ を最小にする i を i_j^* として

$$x_{ij}:=\begin{cases} 1, & i=i_j^*, \\ 0, & i\neq i_j^*, \end{cases} \tag{6}$$

と簡単に定まる．後者の問題は，制約 (3) に対するラグランジュ乗数ベクトルを $v=(v_1,v_2,\ldots,v_n)\in R^n$（$R$ は実数全体の集合）として，以下のように定義できる．

$$\begin{aligned} L^{\mathrm{asgn}}(v)=\min & \sum_{i\in I}\sum_{j\in J}(c_{ij}-v_j)x_{ij}+\sum_{j\in J}v_j \\ \text{s.t.} & \sum_{j\in J}a_{ij}x_{ij}\le b_i, \ \forall i\in I, \\ & x_{ij}\in\{0,1\}, \quad \forall i\in I,\ \forall j\in J. \end{aligned} \tag{7}$$

問題 (7) は m 個の独立した **0-1 ナップサック問題**（0-1 knapsack problem）に分割できる．0-1 ナップサック問題は NP 困難であるが，現実的に最適解を得ることが GAP よりも容易であることが知られている (例えば [19])．それでもやはり $L^{\mathrm{rec}}(u)$ に対して最適な x を求める場合よりも計算コストは高い．任意の $u\in R_+^m$ と $v\in R^n$ に対して $L^{\mathrm{rec}}(u)$ と $L^{\mathrm{asgn}}(v)$ は $f(x)$ の下界を与えるが，できるだけ大きな下界を得ることが望ましい．$L^{\mathrm{rec}}(u)$ や $L^{\mathrm{asgn}}(v)$ を最大にする u や v を求める問題は**ラグランジュ双対**（Lagrangian dual）問題と呼ばれ，この問題に対する実用的な近似解法として**劣勾配法**（subgradient method）[7,14] を利用するのが一般的である．なお，最適な乗数 u^* と v^*，および LP 緩和問題の最適値 LP^* に対しては，

$$L^{\mathrm{asgn}}(v^*)\ge L^{\mathrm{rec}}(u^*)=LP^*$$

が成立することが知られており，多くの場合 $L^{\mathrm{asgn}}(v^*)$ の方がよりよい下界を与える．このように，下界の精度と計算コストの間にトレードオフがある場合，いろいろ試して検証することが推奨される．

下界値の精度を上げる有効な手段のひとつに，**妥当不等式**（valid inequality）を用いる方法がある．ある線形不等式制約を元の問題の任意の実行可能解が満たすとき，その制約を妥当不等式あるいは**切除平面**（cutting plane）と呼ぶ．LP 緩和問題の実行可能解の中にはそのような制約を満たさないものも有り得るので，LP 緩和問題に妥当不等式をいくつか追加することで，よりよい下界値が得られる可能性がある．このアイデアを分枝限定法に組み込んだものを総称して**分枝カット法**（分枝切除法，branch-and-cut method）と呼ぶ [11,23]．GAP に対する簡単な例を紹介しておこう（より高度な方法については [20,21] などを参照）．GAP の LP 緩和問題に対する最適解を $\bar{x}$ とし，あるエージェント i において $0<\bar{x}_{ij}<1$ を満たす j が存在するとする．この i に対し，$\bar{x}_{ij}>0$ を満たす j を a_{ij} の降順に整列し，その結果を $j_1,j_2,\ldots$ とする（すなわち $a_{ij_1}\ge a_{ij_2}\ge\cdots$）．$\sum_{l=1}^{k-1}a_{ij_l}\le b_i<\sum_{l=1}^{k}a_{ij_l}$ を満たす k に対し $J_i=\{j_1,j_2,\ldots,j_k\}$ と定義すれば，$\sum_{j\in J_i}x_{ij}\le|J_i|-1$ は妥当不等式となる．よって，$\bar{x}$ がこの制約を満たしていない（すなわち $\sum_{j\in J_i}\bar{x}_{ij}>|J_i|-1$ が成り立つ）場合，この制約を追加することで下界値が上がる可能性が高い．

よい上界値を早い時点で得ることも下界値テストをより強力なものにするのに効果的である．例えば初期暫定値として高い精度のものを得るために，メタ戦略などの高度なアルゴリズムを組み込むこともしばしば行われる（GAP に対するメタ戦略については，例えば [27] を参照）．分枝を開始した後も，各部分問題において，簡便な方法を用いて上界値計算を行うことも多い．この実現方法としては，下界値計算にラグランジュ緩和を用いる場合に，そこから得られる情報を利用する**ラグランジュ近似解法**（Lagrangian heuristics）や，LP 緩和問題を利用する場合に，その最適解に多少の修正を加えて元の問題の実行可能解を得ようとする方法などが代表的である．

活性部分問題の中から次の探索対象としてどれを選ぶかの戦略（**探索法**（search strategy）と呼ぶ）も分枝限定法の性能に大きな影響を与える．**深さ優先探索**（depth-first search）は，最後に生成された部分問題を優先する方法であり，実装が容易であるためしばしば利用される．この戦略では，根から現在探索中の節点に至る路上の節点の兄弟のみに活性節点が現れるため，記憶容量が小さくて済むという利点がある．別の戦略として，活性部分問題それぞれに対して最適値の推定値を計算しておき，その値が最小のものを優先する方法もよく知られており，**最良優先探索**（best-first search）あるいは**発見的探索**（heuristic search）と呼ばれる．この戦略を用いることで，よい上界値を早く見つけられる可能性が高まるため，とくに，探索を途中で打ち切り分枝限定法を近似解法として利用する場合に有効である．また，計算終了までに調べる部分問題の個数を低く抑えられることも期待できる．しかし，深さ優先探索に比べ多くの記憶容量を必要とする傾向にあり，実装もやや複雑になる．なお，最適値の推定値として緩和問題の最適値（下界値）を用いることも可能であり，この場合，**最良下界探索**（best-bound search）とも呼ばれる。

簡単な考察に基づき，最適性を失うことなく一部の

変数の値を固定できることがあり（**変数固定**，variable fixing），その実現法を**釘付けテスト**（pegging test）などと呼ぶ．探索の前処理として用いることが多いが，部分問題の縮小にも利用できる．例えば，GAP の例で緩和問題 (5) を用いる場合を考える．便宜上 $c_{ij}(u)=c_{ij}+u_i a_{ij}$ と定義しておく．いま，ある部分問題においてその緩和問題の最適値 $L^{\mathrm{rec}}(u)$ は暫定値より小さいとする．この緩和問題の最適解 x は式 (6) で与えられるが，仮にある仕事 j に対して変数 $x_{i^*_j,j}$ の値を 0 に固定したとすると，緩和問題の最適値 $L^{\mathrm{rec}}(u)$ は，$\min_{i\in I\setminus\{i^*_j\}} c_{ij}(u)-c_{i^*_j,j}(u)$ だけ増加することが容易に計算できる．したがって，この値を $L^{\mathrm{rec}}(u)$ に加えたものが暫定値以上になる場合，この部分問題において $x_{i^*_j,j}=0$ を満たす解を探索する必要はないことが結論づけられ，$x_{ij}=1$ $(i=i^*_j)$，$x_{ij}=0$ $(i\neq i^*_j)$ に固定することができる．同様に $x_{ij}=1$ $(i\neq i^*_j)$ と固定した場合の増加量は $c_{ij}(u)-c_{i^*_j,j}(u)$ で与えられるため，これを $L^{\mathrm{rec}}(u)$ に加えたものが暫定値以上になる場合，$x_{ij}=0$ に固定できる．なお，この例のような上下界に基づく釘付けテストが有効に働くには，上下界がある程度近いことが必要である．

問題の定式化の仕方にはいろいろあるが，部分構造の列挙に基づいた方法もしばしば有効である．一例として GAP を**集合分割問題**（set partitioning problem）に定式化する方法を紹介しておく．各エージェント $i\in I$ に対し，仕事集合 $J'\subseteq J$ の中で資源制約 $\sum_{j\in J'} a_{ij}\le b_i$ を満たすもの全てを（仮想的に）番号づけし，その添字集合を C_i と書くことにする．各 $k\in C_i$ に対応する仕事集合を $J^k_i\subseteq J$ とし，それらの仕事をエージェント i に割当てたときのコストを $d^k_i=\sum_{j\in J^k_i} c_{ij}$ とする．また，$j\in J^k_i$ $(j\notin J^k_i)$ ならば $\delta^k_{ij}=1$ (0) と定める．これらを用いて GAP を以下のように定式化できる．

$$\min \quad \sum_{i\in I}\sum_{k\in C_i} d^k_i y^k_i \tag{8}$$

$$\text{s.t.} \quad \sum_{i\in I}\sum_{k\in C_i} \delta^k_{ij} y^k_i = 1, \quad \forall j\in J, \tag{9}$$

$$\sum_{k\in C_i} y^k_i = 1, \qquad \forall i\in I, \tag{10}$$

$$y^k_i\in\{0,1\}, \qquad \forall k\in C_i,\ \forall i\in I. \tag{11}$$

もちろん C_i の全ての要素を実際に列挙するのは現実的ではなく，探索に必要と思われるものだけをうまく生成する手立てが必要となる．これを実現する方法として，制約 (11) を $0\le y^k_i\le 1$ に緩和した LP 問題の場合には**列生成**（column generation）法 [8,9] を用いることができる．列生成法では，まず，各 C_i を十分小さな $C'_i\subset C_i$ に置き換えた LP 問題を考え，その最適解 $\bar{x}$ を求める．このとき，制約 (9) と (10) の双対変数を $\bar{v}_j$ $(j\in J)$ および $\bar{w}_i$ $(i\in I)$ として，条件 $\sum_{j\in J'}(c_{ij}-\bar{v}_j)<\bar{w}_i$ および $\sum_{j\in J'} a_{ij}\le b_i$ を満たす (i,J') の組が存在すれば，J' を C'_i に追加し再び LP 問題を解くことによって通常は解を改善できる．逆にそのような (i,J') が存在しなければ，$\bar{x}$ は元の（C'_i ではなく C_i で定義される）LP 問題の最適解となるので，そのような解を得るには，C'_i に適当な仕事集合 J' を追加しながら繰り返し LP 問題を解けばよい．ここで，上述の条件を満たす (i,J') を求める（あるいは存在しないことを結論づける）問題をいかに効率よく解くかがアルゴリズム設計において重要となる．GAP の例においては，それぞれの $i\in I$ に対し，条件 $\sum_{j\in J'} a_{ij}\le b_i$ のもとで $\sum_{j\in J'}(c_{ij}-\bar{v}_j)$ を最小にする J' を求める問題，すなわち 0-1 ナップサック問題を解けばよい．これは，**被約費用**（reduced cost）が最小の変数 y^i_k を求めることに相当し，一般に**価格付け問題**（pricing problem）と呼ばれる．ところで，列生成法自体は LP 問題を解くものであり整数解を得るものではない．そこで，整数解を求めるために分枝限定法と組合せる方法が考えられ，これを**分枝価格法**（branch-and-price method）と呼ぶ [2]．分枝価格法を設計するときには，価格付け問題の構造をよく考えた上で分枝操作を定める必要がある．例えば GAP の場合，ある変数 y^k_i の値を 0 または 1 に固定する方法が考えられるが，y^k_i の値を 0 に固定した部分問題においては条件 $J'\neq J^k_i$ を考慮することになり，その結果，価格付け問題を通常の 0-1 ナップサック問題として定式化できなくなる．そのため，分枝操作には元の定式化における変数 x_{ij} を用いることで価格付け問題の構造を保つ方法が提案されている [24]．なお，列生成法とともに切除平面を分枝限定法に組み込むことも可能であり，この場合は**分枝価格カット法**（branch-and-price-and-cut）と呼ばれる [1]．

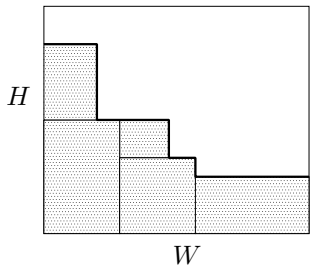
4 矩形パッキング問題への適用例

ここまで GAP を例に説明を行ってきたが，少々タイプの異なる問題への適用例として矩形パッキング問題への適用例を紹介しよう [17]．この問題には様々なバリエーションがあるが，ここでは以下を考える．n 個の矩形 $R=\{1,2,\ldots,n\}$ が与えられ，矩形 $i\in R$ の幅と高さをそれぞれ w_i, h_i とする．このとき，すべての矩形を重なりなく幅 W, 高さ H の長方形領域に隙間なく配置することができれば yes，できなければ no を返す．このように隙間のない配置のことを完全パッキングと呼び，完全パッキングが存在するためには，$\sum_{i=1}^n h_i w_i = HW$ でなくてはならない．なお，この問題は最適化問題ではなく実行可能性を判定する決定問題である．また，等式や不等式による定式化を前提としないという点でも，GAP の例とは異なる．しかし，分枝操作により生成木を定義し，限定操作により探索木の節点数を減らすという分枝限定法の基本的な考え方に相違はない．

本節で紹介する分枝限定法は，分枝操作により，ある 1 つの矩形の配置を固定するものである．したがって，生成木の深さ d の節点は，d 個の矩形の配置が固定された部分問題に対応する．分枝操作の具体的な実現方法と

しては例えば以下の方法が考えられる．いま，ある部分問題においていくつかの矩形の配置が定まっているものとする．ここで，まだ矩形が置かれていない領域（以下，未配置領域と呼ぶ）に着目すると，完全パッキングに隙間は存在しないので，残りの矩形で未配置領域を埋め尽くさなくてはならない．そこで，未配置領域の中で最も下に位置し，さらに最も左にある点を考え，そこにどの矩形を置くかによって分枝する．このとき，他の矩形と重なることなく，かつ長方形領域からはみ出すことなく配置することのできる矩形に対してのみ分枝すればよく，そのような矩形が存在しない場合には部分問題を終端することができる．分枝の結果，完全パッキングが得られれば yes を出力して分枝限定法全体が終了する．また，完全パッキングを発見することなくすべての部分問題が終端されれば答えは no である．

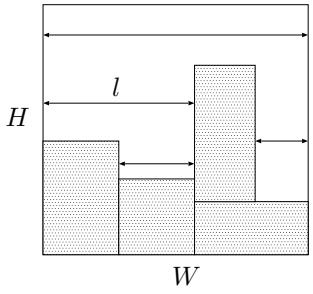
他には，分枝操作で矩形を 1 つ配置するとき,「矩形が置かれている領域と未配置領域が 1 本の右下がりの階段状の境界線によって分離される」という条件（第 2 図参照）を満たす範囲ですべての可能性を考え，それぞれについて子節点を生成する方法も可能である．詳細は省略



第 2 図　1 本の右下がりの階段状の境界線によって 2 分された長方形領域の例

するが，列挙する矩形の配置をこのように限定しても，答えが yes の場合には完全パッキングが得られることが証明できる．また，この分枝操作が探索木の節点数を減らすのにしばしば有効であることが数値実験により確かめられている．ただし，この分枝操作で定義される生成木には，同じ部分問題に対応する複数の異なる節点が存在する可能性があり，無駄な探索を行わないための工夫が必要になる．

限定操作には，以下のテストが利用可能である．ここでも,「完全パッキングに隙間は存在しない」という性質を活用することになる．ある部分問題において，まだ配置が固定されていない矩形集合を $R' \subset R$ とする．このとき，未配置領域に横方向の長さがちょうど l の隙間が存在するとすれば（第 3 図参照），l は，残りの矩形 $i \in R'$ の横の長さ w_i を組合せて実現できる長さでなくてはならない．つまり，条件 $\sum_{i \in R''} w_i = l$ を満たす矩形集合 $R'' \subseteq R'$ が存在しなくてはならず，存在しない場合には，残りの矩形を用いて隙間を完全に埋めることはできないと結論づけられる．隙間の縦方向の長さに対



第 3 図　残りの矩形で埋めるべき隙間の例（隙間の長さは横方向のみ図示）

しても同様のテストを行うことができ，条件を満たす矩形集合 R'' が存在しない l が 1 つでも存在すれば，部分問題を終端することができる．なお，条件を満たす矩形集合 R'' の存在性判定問題は**部分和問題**（subset-sum problem）に帰着される．この問題は NP 困難であることが知られているが，実際には動的計画法を用いて比較的高速に解くことができる．

5　関連の話題

最後に，関連する話題をいくつか紹介しておこう．分枝限定法に基づく方法が広く用いられている問題の一つに**混合整数計画**（mixed integer programming, MIP）問題がある．その汎用性から MIP 問題を解くアルゴリズムのニーズは高く，商用のものを含め多数のソフトウェアが存在する．近似解法として利用されることも多く，今もなおアルゴリズムの改良が活発に行われている．MIP 問題に対する分枝限定法を近似解法として用いることを念頭に置いた技法として（とは言え，分枝限定法の厳密性を失うものではない），**局所分枝**（local branching）と呼ばれる分枝規則が提案されており [6]，商用のパッケージにも組み込まれている．これは，発見的手法としてよく用いられる局所探索法のアイデアを取り入れたものであり,「よい解の近くにはよい解が存在する」という仮説に基づいている．n 変数の MIP 問題を考え，そのうち n' $(<n)$ 個が 0-1 変数であるとしよう．2 つの解 x_1 と x_2 の距離を，両者で値が異なる 0-1 変数の数と定義しておく．局所分枝は，暫定解 x に対して，x との距離が k $(<n')$ 以下である場合と $k+1$ 以上である場合の 2 つを考え，分枝する方法である（それぞれ不等式を 1 つ追加することで実現可能）．そして，前者を先に探索することで，より早い段階での暫定値更新を期待するものである．なお，局所分枝の枠組みは柔軟であり，解の間の距離を定義し，暫定解からの距離に関する不等式制約を扱うことができれば，0-1 変数を含まない MIP 問題や別の特定の問題に対しても適用可能である．

分枝限定法の効率化のために，主に人工知能の分野で研究されている**制約プログラミング**（constraint programming）において中心的役割を果たす**制約伝播**（constraint propagation）の考え方を利用することも可

能である．すなわち，前処理として，あるいは部分問題において，複数の制約を論理的に考察することで探索すべき解空間（探索空間）を縮小する方法である．縮小の結果，ある変数の値域が空になれば部分問題を実行不可能として終端することもできる．基本的に前述の変数固定と目指すところは同じであると言える．スケジューリング問題に対する簡単な例を紹介しておこう（より詳しくは例えば [4] を参照）．1台の機械で複数の作業 V を処理するものとする．ただし，作業の中断は許されず，2つの作業を同時に処理することもできない．各作業 $i \in V$ は処理時間 p_i を持ち，開始時刻 x_i は，$e_i \le x_i \le l_i$ を満たす変数であるとする．このとき，ある作業集合 $U \subseteq V$ とそれに含まれる作業 $i \in U$ に対して，条件

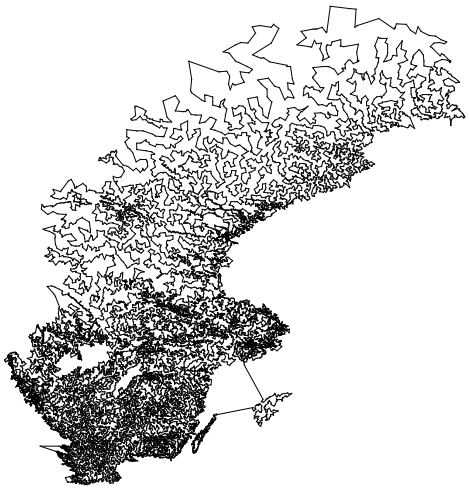
$$\max_{u \in U \setminus \{i\}, v \in U, u \neq v} (l_v + p_v - e_u) < \sum_{j \in U} p_j \qquad (12)$$

が成立すれば，作業 i は U に含まれる他のどの作業 $j \in U \setminus \{i\}$ よりも先に処理されなくてはならず（すなわち $x_i + p_i \le x_j$），この制約を満たさない解は探索する必要がないことを結論できる．(12) 式の左辺は，i 以外の作業の開始可能時刻 e_u の最小値から納期 $l_v + p_v$ の最大値までの時間であり，i を最初に行わない場合に U 内の作業に費やせる処理時間の上界を与えるからである．このとき，例えば $e_j < e_i + p_i$ が成立していたとすれば，変数 x_j の値域は縮小されることになるが，この結果，別の U と i の組が式 (12) を満たして新たな制約が導かれる可能性もある．このように制約を伝播させていくことで，探索空間の縮小を連鎖的に行えることもある．なお，これらの操作を前処理としてだけ利用するか，分枝後もそれぞれの部分問題に対して適用するかは，条件判定に要する計算時間とその効果とのバランスを考慮して決定する必要がある．

代表的な組合せ最適化問題である**巡回セールスマン問題**（traveling salesman problem, TSP）については，都市数 n の大きな問題例に対して厳密な最適解を求める競争が世界レベルで行われている．最近では，$n = 33{,}810$ のもの（TSPLIB の `pla33810`）が厳密に解けたとの記録がある [3].[1] このような記録更新は，切除平面に対する工夫による貢献が大きい（TSP に対する切除平面については [16,28] などに説明がある）．これよりも少々規模は小さいが，スウェーデンの 24,978 都市を巡回する最適巡回路を図 4 に示しておく．ただし，これらの記録は多数の計算機を用いて非常に長い時間をかけて得られたものであり，これほどの規模の問題例が実用的に解けるわけではないことを断っておく．

TSP の一般化で，より実用的な問題である**配送計画問題**（vehicle routing problem, VRP）にも分枝限定法はしばしば適用されてきた．VRP は，複数の配送車で

[1] http://www.tsp.gatech.edu/などを参照.



第 4 図　スウェーデンの 24,978 都市を巡回する最適巡回路

顧客をまわるとき，配送車の台数や総移動距離を最小化する問題である．ただし，顧客の時間枠制約や配送車の容量制約などの付加的な制約が課せられることも多く，その結果，実行可能なルートが制限されるような場合には，3 節で述べた集合分割問題としての定式化を行うことで分枝価格法が有効に働く（例えば [25] など）．集合分割問題への定式化は，VRP の他，航空機の乗務員スケジューリング [10] や人員スケジューリング [5] などに対してもよく行われる．また，これらの問題は，集合分割問題の等号制約を不等号制約にした**集合被覆問題**（set covering problem）として定式化されることも多い．

6　おわりに

本稿では，分枝限定法の基礎とその発展形，および関連の話題について概説した．主に GAP と矩形パッキング問題を取り上げて基本的なアイデアを紹介したが，この他にも様々な工夫を取り入れることが可能であり，その結果かなりの規模の問題例が厳密に解けるようになってきている．(問題タイプにもよるが，GAP では 0-1 変数で 4000 変数程度のものが，矩形パッキング問題では矩形数 50 程度のものが，実用的な時間で厳密に解けている [17,21].) なお，分枝限定法の基本部分に関して詳しい説明がある和書として [15,18] を挙げておくので，興味のある読者は参照して頂きたい．ところで，分枝限定法以外にも，組合せ最適化に対する厳密解法としては動的計画法が古くから知られている．これらはともに列挙をスマートに行う方法であるが，近年，列挙にも切除平面にも基づかない**整基底法**（integral basis method）と呼ばれる手法が注目されており，今後の発展が期待されている [12,13]．このように，厳密解法の研究は今も発展し続けており，本稿で取り上げていない話題も多数あるが，本稿が厳密解法の研究を志す読者の一助となれば幸いである．最後に，本稿の執筆にあたり有益なコメントをいただいた茨木俊秀，藤重悟，永持仁，梅谷俊治，今堀慎治の諸氏に謝意を表する．

参 考 文 献

[1] C. Barnhart, C.A. Hane and P.H. Vance, Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems, *Operations Research*, 48 (2000) 318–326.

[2] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, Branch-and-price: Column generation for solving huge integer programs, *Operations Research*, 46 (1998) 316–329.

[3] W. Cook, D. Espinoza and M. Goycoolea: Computing with domino-parity inequalities for the TSP, working paper available at http://www2.isye.gatech.edu/˜mgoycool/

[4] U. Dorndorf, *Project Scheduling with Time Windows: From Theory to Applications*, Springer, 2002.

[5] A.T. Ernst, H. Jiang, M. Krishnamoorthy and D. Sier, Staff scheduling and rostering: A review of applications, methods and models, *European Journal of Operational Research*, 153 (2004) 3–27.

[6] M. Fischetti and A. Lodi, Local branching, *Mathematical Programming*, 98 (2003) 23–47.

[7] M.L. Fisher, The Lagrangian relaxation method for solving integer programming problems, *Management Science*, 27 (1981) 1–18.

[8] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem, *Operations Research*, 9 (1961) 849–859.

[9] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting-stock problem — part II, *Operations Research*, 11 (1963) 863–888.

[10] B. Gopalakrishnan and E.L. Johnson, Airline crew scheduling: State-of-the-art, *Annals of Operations Research*, 140 (2005) 305–337.

[11] M. Grötschel and O. Holland, Solution of large-scale symmetric travelling salesman problems, *Mathematical Programming*, 51 (1991) 141–202.

[12] U.-U. Haus, M. Köppe and R. Weismantel, The integral basis method for integer programming, *Mathematical Methods of Operations Research*, 53 (2001) 353–361.

[13] U.-U. Haus, M. Köppe and R. Weismantel, A primal all-integer algorithm based on irreducible solutions, *Mathematical Programming*, 96 (2003) 205–246.

[14] M. Held and R.M. Karp, The traveling salesman problem and minimum spanning trees: Part II, *Mathematical Programming*, 1 (1971) 6–25.

[15] 茨木，組合せ最適化—分枝限定法を中心として，産業図書，1983.

[16] 茨木，福島，最適化の手法，共立出版，1993.

[17] 劔持，今道，野々部，柳浦，永持，矩形パッキングに対する厳密解法，信学技報 COMP2005-2 (2005) 5–14.

[18] 今野，鈴木（編），整数計画法と組合せ最適化，日科技連出版社，1982.

[19] S. Martello, D. Pisinger and P. Toth, Dynamic programming and strong bounds for the 0-1 knapsack problem, *Management Science*, 45 (1999) 414–424.

[20] R.M. Nauss, Solving the generalized assignment problem: An optimizing and heuristic approach, *INFORMS Journal on Computing*, 15 (2003) 249–266.

[21] R.M. Nauss, The generalized assignment problem, in: J.K. Karlof, ed., *Integer Programming: Theory and Practice*, CRC Press, 2006, 39–55.

[22] 根本，堀田，公平な小選挙区制のための数理モデル，システム/制御/情報，49 (2005) 78–83.

[23] M. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Review*, 33 (1991) 60–100.

[24] M. Savelsbergh, A branch-and-price algorithm for the generalized assignment problem, *Operations Research*, 45 (1997) 831–841.

[25] M. Savelsbergh and M. Sol, Drive: Dynamic Routing of Independent Vehicles, *Operations Research*, 46 (1998) 474–490.

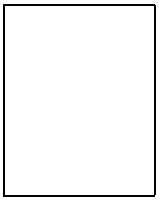
[26] 柳浦，茨木，組合せ最適化—メタ戦略を中心として，朝倉書店，2001.

[27] M. Yagiura and T. Ibaraki, Generalized Assignment Problem, in: T.F. Gonzalez, ed., Handbook of Approximation Algorithms and Metaheuristics, Chapman & Hall/CRC in the Computer & Information Science Series, to be published.

[28] 山本，久保，巡回セールスマン問題への招待，朝倉書店，1997.

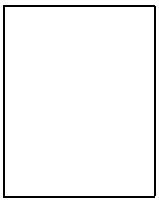
著 者 略 歴

柳浦 睦憲（やぎうら むつのり）



1991 年 3 月京都大学工学部数理工学科卒．1993 年 3 月同大学院工学研究科数理工学専攻修士課程了．1994 年 3 月同博士後期課程中退．同年 4 月京都大学工学部助手．2000 年 4 月京都大学大学院情報学研究科講師．2005 年 10 月名古屋大学大学院情報科学研究科助教授となり現在に至る．メタヒューリスティクスの研究等に従事．京都大学博士（工学）．日本オペレーションズ・リサーチ学会，情報処理学会，電子情報通信学会，スケジューリング学会，INFORMS, ACM などの会員．

野々部 宏司（ののべ こうじ）



1995 年 3 月京都大学工学部数理工学科卒．1997 年 3 月同大学院工学研究科数理工学専攻修士課程修了．2000 年 3 月同大学院情報学研究科数理工学専攻博士後期課程修了．同年 4 月京都大学大学院情報学研究科助手．2005 年 4 月法政大学工学部講師．2006 年 4 月助教授となり現在に至る．メタヒューリスティクス，スケジューリングの研究等に従事．京都大学博士（情報学）．日本オペレーションズ・リサーチ学会，スケジューリング学会，情報処理学会，INFORMS などの会員．