組合せ最適化の数理
計算困難問題への挑戦

柳浦 睦憲

1. はじめに

曾呂利新左衛門の逸話を知っている人は少なくないだろう．彼が太閤秀吉から褒美をもらう際，将棋版の最初の目に米粒を 1 つ置き，次の目に 2 つ，その次の目に 4 つと，順次 2 倍ずつ置いた分だけ欲しいと言い，秀吉ははじめは欲のないやつだと安請け合いしたが，途中で音をあげたというのである．秀吉が約束を守ったとすると，曾呂利新左衛門が手にする米粒の数は合計 $2^{81}-1$ 粒（約 4×10^{20} 合）となる．これを消費するには，1 億人が毎日 10 合（丼に 10 杯程度）食べたとしても，10 億年以上食べ続けなければならない．指数関数 2^n が n とともに急激に大きくなる様子が実感できるであろう．

組合せ最適化問題の多くは，有限の可能性を列挙してそれぞれを調べることで厳密に解くことができる．よって，一見簡単な問題に思えるが，列挙すべき可能性は通常指数通り存在し，上の逸話からも分かるように，この方法は問題の規模が大きくなると現実的ではない．しかし，

- コンビニエンスストアに効率よく商品を配達するようにトラックをスケジュールする，
- 工場の生産計画においてコストの最小化を図る，

というような，応用上重要な問題の多くが組合せ最適化問題として現れるため，現実的な解法が必要とされている．本稿では，そのような現実的解法を紹介する．

2. 組合せ最適化問題

最適化問題は一般的に以下のように表される．

$$\begin{array}{ll}\text{最小化} & f(x)\\ \text{制約条件} & x\in F.\end{array}\qquad(1)$$

$f:F\to R$（あるいは Z）（R（Z）は実数（整数）の集合）を**目的関数**，F を**実行可能領域**と呼ぶ．F は制約条件をみたす解の全体を表し，個々の解 $x\in F$ を**実行可能解**と呼ぶ．$f(x)$ を最小にする実行可能解を**最適解**と呼ぶ．そのような解を見つけることが最適化問題の目標である．F が組合せ的な構造を持つ場合，(1) は**組合せ最適化問題**（combinatorial optimization problem）と呼ばれる．以下に，代表的な組合せ最適化問題の例として，巡回セールスマン問題，ナップサック問題，整数計画問題を説明しておく．

2.1 巡回セールスマン問題

巡回セールスマン問題（traveling salesman problem，TSP）は，n 個の街の集合 $V=\{1,\ldots,n\}$ と街 i と j の間の距離 d_{ij} $(i,j\in V)$ が与えられたとき，全ての街をちょうど 1 度ずつ訪問したのち元に戻る巡回路のうち，総距離最小のものを求める問題である．TSP の解は n 個の街の訪問順序，すなわちそれらの順列 $\sigma:V\to V$ で与えられる．$\sigma(k)=i$ は，k 番目に訪れる街が i であることを意味する（便宜上 $\sigma(n+1)=\sigma(1)$ を仮定）．これを用いると，最小化すべき総距離は，

$$f_{\mathrm{TSP}}(\sigma) = \sum_{k=1}^{n} d_{\sigma(k),\sigma(k+1)}$$

と書ける．距離行列 (d_{ij}) は対称と仮定する．

2.2 ナップサック問題

0-1 ナップサック問題（0-1 knapsack problem）は，各要素 j に対するサイズ a_j と利得 c_j，およびナップサックの容量 b が与えられたとき，与えられた n 個の要素集合 $V = \{1, \ldots, n\}$ からいくつかを選び，選ばれた要素のサイズの合計がナップサックの容量を越えないという条件のもとで，利得の合計を最大化するという問題である．a_j, c_j および b は自然数であり，$a_j \leq b$ $(\forall j \in V)$ が成り立つと仮定する．決定変数 $z = (z_1, \ldots, z_n)$ を用意し，$z_j = 1$（0）ならば要素 j をナップサックに入れる（入れない）と解釈すると，問題は以下のように定式化できる．

$$\begin{array}{ll} \text{最大化} & f_{\mathrm{KP}}(z) = \sum_{j=1}^{n} c_j z_j \\ \text{制約条件} & \sum_{j=1}^{n} a_j z_j \leq b \\ & z_j \in \{0, 1\}. \end{array}$$

2.3 整数計画問題

係数 a_{ij}, b_i および c_j $(i = 1, \ldots, m,\ j = 1, \ldots, n)$ と集合 $J \subseteq V = \{1, \ldots, n\}$ が与えられたとき，以下の形に書ける問題を**整数計画問題**（integer programming problem, IP 問題）と呼ぶ．

$$\begin{array}{lll} \text{最小化} & f_{\mathrm{IP}}(z) = \sum_{j=1}^{n} c_j z_j & \\ \text{制約条件} & \sum_{j=1}^{n} a_{ij} z_j \geq b_i, & i = 1, \ldots, m \\ & z_j \geq 0, & j = 1, \ldots, n \\ & z_j : \text{整数}, & j \in J. \end{array}$$

$J = V$ ならば**全 IP 問題**，$J \neq V$ ならば**混合 IP 問題**という．$J = \emptyset$ のときは，とくに**線形計画問題**（linear programming problem, LP 問題）と呼ばれる．整数計画問題は一般的な形をしているので，ほとんどの組合せ最適化問題は自然な形でこの問題に定式化できる．

2.4 指数時間と多項式時間

指数関数が急激に増加することは，冒頭の逸話で説明した通りである．一般に，問題のデータを入力するのに要する領域量（文字数など）を n とするとき，アルゴリズムの計算時間が最悪の場合 $2^n, 3^n, n!$ のような関数に比例するものを指数時間アルゴリズムとよぶ．これに対し，計算時間が定数 k を用いて n^k に比例する関数で抑えられる場合，多項式時間アルゴリズムとよぶ．多項式時間のほうが指数時間よりも計算時間の増加が緩やかなので（実際，任意の定数 k と l に対し，n が十分大きければ $n^k < l^n$ が成り立つ），一般には多項式時間アルゴリズムのほうが望ましい．ある問題が **NP 困難**（NP-hard）であると，その問題に対する多項式時間アルゴリズムは存在しないと考えられている．（紙面の都合上，NP 困難性のきちんとした説明は省略するが，たとえば文献 2) などを参照のこと．NP 困難問題に対して多項式時間アルゴリズムが本当に存在しないかどうかは未解決で，P $\neq$ NP 予想として有名である．）上述の巡回セールスマン問題，ナップサック問題，整数計画問題の 3 つはいずれも NP 困難であることが知られている．一方，問題の条件の一部を制限した特殊な場合には多項式時間アルゴリズムが存在する場合もある．LP 問題はそのような例である．

3. 厳密解法

組合せ最適化問題に対して厳密な最適解を求める目的に用いられる代表的な手法として，**分枝限定法**（branch-and-bound method）と**動的計画法**（dynamic programming）がある[4]．分枝限定法や動的計画法は，問題の構造をうまく利用することによって，解の列挙を効率的に行う手法である．以下，これらの基本的な考え方を紹介する．具体例を用いたほうが分かりやすいので，0-1 ナップサック問題に対するアルゴリズムの構成例を表 1 の $n = 4$ のデータを用いて紹介する．

表 1　0-1 ナップサック問題の入力例

n	$=$	4,	b	$=$	9							
c_1	$=$	4,	c_2	$=$	5,	c_3	$=$	12,	c_4	$=$	14	
a_1	$=$	2,	a_2	$=$	3,	a_3	$=$	5,	a_4	$=$	6	

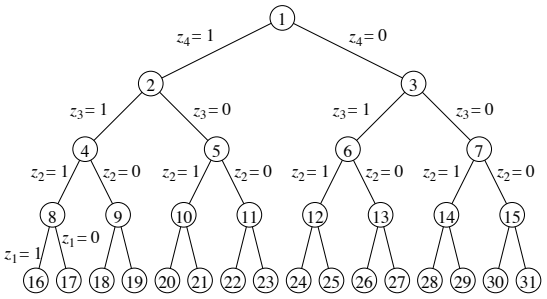


図 1　0-1 ナップサック問題に対する分枝木

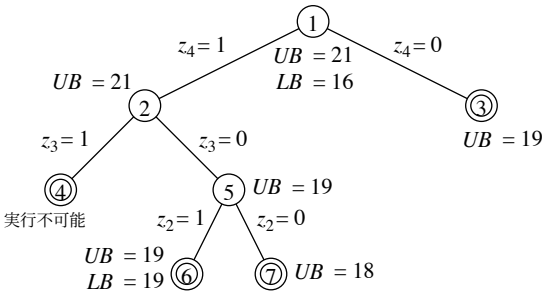


図 2　0-1 ナップサック問題に対する分枝限定法の進行

3.1 分枝限定法

分枝限定法は，一部の変数を固定しつつそれぞれの場合を調べ上げていくという考え方に基づいている．この過程は，図 1 のような図（分枝木と呼ぶ）を用いて表現できる．一番上の点 1 は元の問題に対応しており，その下の点 2 と 3 は，それぞれ変数 z_4 を 1 と 0 に固定した問題を表している．このように，一つの段がどれかの変数に対応しており，左（右）に下がることは対応する変数の値を 1（0）に固定することを意味する．よって，内部の点は，点 1 からその点へのパスに対応して変数の値を一部固定した問題（部分問題と呼ぶ）に対応しているが，部分問題もやはり 0-1 ナップサック問題である．最下段の各点は，$z_1, \ldots, z_4$ の全てが固定された 1 つの解に対応している（$2^4 = 16$ 個存在）．

分枝限定法では，図 1 のように全ての場合を列挙するわけではなく，その一部分だけを実際に生成する．ある部分問題に対して，(1) その最適値，または (2) その部分問題が元の問題の最適解を与えないことが分かると，その下の分岐（子孫と呼ぶ）を調べる必要がないと結論でき，探索を省略できる（終端するという）．これを限定操作と呼ぶ．

限定操作の例として，上界値テストを紹介しておく．制約「$z_j \in \{0, 1\}$」を「$0 \leq z_j \leq 1$」に緩和すると，LP 問題（LP 緩和問題と呼ぶ）が得られる．このとき，任意の実行可能解 z に対し，

$$f_{\mathrm{KP}}(z) \leq \lfloor \text{LP 緩和問題の最適値} \rfloor$$

が成り立つ．LP 問題の最適値は多項式時間で求めることができる．下界値についても，任意の実行可能解はその目的関数値が最適値の下界となるので，4 節で紹介する様々な近似解法やメタ戦略によって求めることが可能である．このように上下界値は容易に求まるが，これらを用いると，「ある部分問題の上界値が暫定値（すなわちこれまでに得られている最良の下界値）以下であれば，その子孫を調べる必要はない」ことが結論できる．以上が上界値テストである．（最小化問題の場合は上界と下界の役割が入れ替わる．）これ以外に，部分問題の最適解が求まった場合や，部分問題が実行不可能である場合も限定操作を適用できる．

図 2 に，表 1 の問題例に対する分枝限定法の進行の様子を示す．詳細は略すが，点番号 1, 2, 4, 5, 6, 7, 3 の順に探索を行った結果である．図中，UB は LP 緩和問題による上界値，LB は下界値を表す．点 1 における $LB = 16$ は，4.1 節の欲張り法による．点 4, 6, 7, 3 が終端された理由を示しておく．点 4 の部分問題は，$a_3 + a_4 = 11 > 9 = b$ より，実行不可能である．点 6 では，$a_2 + a_4 = 9 = b$ より自動的に $z_1 = 0$ が定まるため，部分問題の最適値が 19 であることが分かり，暫定値が 19 に更新される．点 7 と 3 の上界値はいずれも暫定値 19 以下である．結局，点 6 において求まった解 $z = (0, 1, 0, 1)$ が最適であり，最適値は 19 である．全ての解を列挙した場合，図 1 のように 31 個の点が生成されるが，限定操作によって列挙を省略することにより，図 2 では生成される点の数が 7 個と，大幅に削減されている．

一般に，より精度の高い上界値を求めることができれば，性能の向上が期待できる．この目的に

対して，元の問題の実行可能領域を変更することなく LP 緩和問題の実行可能領域を狭くする制約式（LP 問題の実行可能領域の一部を切り取る超平面であることからカットと呼ばれる）を追加することによって上界値の向上を図る方法も研究されており，**分枝カット法**（branch-and-cut method）と呼ばれている（例えば文献 5) 参照）．

3.2 動的計画法

動的計画法は，解の列挙にあたって，多くの場合を一つの状態にまとめることで列挙の手間を削減するというアイデアに基づいている．以下，0-1 ナップサック問題に対する構成例を紹介する．

$f^*(j,k)$ を，要素集合 $\{1,\ldots,j\}$ からいくつかを選んで容量 k のナップサックに詰める場合の最適値とする．すると，$f^*(j,k)$ は漸化式

$$f^*(1,k) = \begin{cases} -\infty, & k<0 \\ 0, & 0\le k<a_1 \\ c_1, & a_1\le k\le b \end{cases}$$

$$f^*(j,k) = \max\{f^*(j-1,k), f^*(j-1,k-a_j)+c_j\},\quad j=2,3,\ldots,n,\ k\le b$$

によって与えられる．第 1 式は，$j=1$ に対する初期条件（境界条件）を示している．3 つの場合は，それぞれ，$k<0$ は実現不可能であることに対応し，$0\le k<a_1$ は $z_1=0$ に，さらに $a_1\le k\le b$ は $z_1=1$ に対応している．第 2 式は，$j\ge 2$ の場合であり，max の中の第 1 項は $z_j=0$ に，第 2 項は $z_j=1$ に対応している．前者の場合は，要素集合 $\{1,\ldots,j-1\}$ からいくつかを選んで容量 k のナップサックに詰めたときの最適値がそのまま利用でき，後者の場合は，要素集合 $\{1,\ldots,j-1\}$ からいくつかを選んで容量 $k-a_j$ のナップサックに詰めたときの最適値に c_j を加えたものが最適になるという意味である．$j=1,\ldots,n$ の順にそれぞれ全ての k に対し $f^*(j,k)$ を求め，最終的に $f^*(n,b)$ を計算すれば元の問題の最適値が求まる．最適解を得るには，各 $f^*(j,k)$ の値が，$z_j=0$ と $z_j=1$ の 2 つの場合のいずれによって実現できるかを (n,b) から順にたどっていけばよい．

表 2　ナップサック問題に対する動的計画法の進行

	容量 k									
要素 j	0	1	2	3	4	5	6	7	8	9
1	0	0	4	4	4	4	4	4	4	4
2	0	0	4	5	5	9	9	9	9	9
3	0	0	4	5	5	12	12	16	17	17
4	0	0	4	5	5	12	14	16	18	19

アルゴリズムの計算手間は nb に比例する時間で抑えられる．b のように入力データの数値が直接計算時間に現れる場合は擬多項式時間と呼ばれ，多項式時間ではないが（b は $\log_2 b$ ビットで表現できることに注意），通常 b の値はそれほど大きくないので，n がある程度大きい場合は，単純に全ての解を列挙した場合の数 2^n より小さくなる．

漸化式において，$f^*(j,k)$ は，$\sum_{h=1}^{j} a_h z_h \le k$ をみたす 0-1 ベクトル $(z_1,\ldots,z_j)$ 全体を表す状態であると理解できる．この問題ではそのようなベクトルによって実現できる総利得の最大値のみが重要であるので，$f^*(j,k)$ の値でその結果を記憶しているわけである．すなわち，一つの状態に対応するたくさんのベクトルを $f^*(j,k)$ というデータに集約することで，列挙の数を減らし，計算効率を高めている．これが動的計画法の考え方である．

表 1 の問題例に対する漸化式の計算の様子を表 2 に示す．表の第 j 行 k 列は $f^*(j,k)$ の値を示している．第 4 行 9 列の値より最適値は $f^*(4,9)=19$ である．また，最適解は，$f^*(4,9)=f^*(3,3)+14$ より $z_4=1$，$f^*(3,3)=f^*(2,3)$ より $z_3=0$，$f^*(2,3)=f^*(1,0)+5$ より $z_2=1$，最後に $f^*(1,0)=0$ は $0\le k<a_1$ の場合に対応するから $z_1=0$ と計算でき，分枝限定法による結果と一致することが分かる．

なお，容量制約と目的関数の役割を入れ替えて，以下のような状態を考えることでも動的計画法を実現できる．すなわち，状態 $g^*(j,p)$ を，要素集合 $\{1,\ldots,j\}$ からいくつかを選んだ利得が丁度 p であるときのナップサックに詰める要素の合計サイズの最小値と定義して漸化式を立てるのである．詳細は練習問題としておこう．計算時間は $\sum_{j=1}^{n} c_j n$ に比例する手間で抑えられる．

3.3 厳密解法の発展

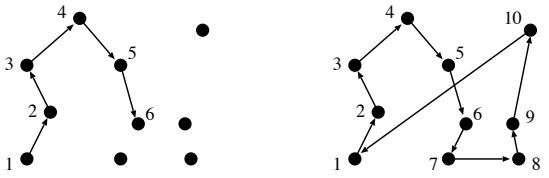
分枝限定法や動的計画法に基づく厳密解法の研究は，現在でも盛んに行われており，0-1 ナップサック問題の場合は要素数 1 万程度の問題例でも大抵あっという間に厳密に解けてしまう[*1)]．また，巡回セールスマン問題に対しては，どれだけ大きなサイズの問題例を厳密に解いたかに対する競争が活発に行われており，現在の世界記録は都市数 15112 である[*2)]．このように，NP 困難問題であっても問題によってはかなりのサイズまで厳密に解けることを知っておいて欲しい．

混合 IP 問題に対する分枝カット法に基づく汎用ソルバも販売されている．これらの性能は，問題構造に大きく依存してしまうが，問題によってはかなり大きなサイズまで扱える．また，分枝限定法や分枝カット法の計算を適当な時点で打ち切って，その時点での暫定解を出力すると，最適性の保証はなくなるが，よい解を速く求めるという目的には有効である．

4. 近似解法

応用の場面においては，厳密な最適解は必ずしも必要ではなく，よい解が速く求まれば十分である場合が多い．そこで，現実的な時間で良質の解を求めるために**近似解法**（approximate algorithm）や**発見的手法**（heuristics）が用いられる．

近似解法の基本戦略として，**欲張り法**（greedy method）や**局所探索法**（local search）が挙げられる．欲張り法は目的関数への貢献度を示す局所的な評価値に基づいて実行可能解を直接構成する方法，局所探索法は与えられた解を簡単な操作によって改善する手続きを反復する方法である．**メタ戦略**（metaheuristics; **メタ解法**，**メタヒューリスティクス**などとも呼ぶ）は，これら基本戦略よりも多少時間はかかっても，より良質の解を求める



(a) 構築中の解　　(b) 得られた巡回路

図 3　最近近傍法の実行例

ような解法の一般的枠組を与えるものである．以下，これらを紹介する．

4.1 欲張り法

欲張り法は例を示すほうが分かりやすいので，構成例を 2 つ紹介する．まず，0-1 ナップサック問題に対する欲張り法として，以下が挙げられる．各要素 j のサイズ a_j と利得 c_j に対し，c_j/a_j は単位サイズあたりの利得と解釈できるから，これの大きいものからナップサックに詰めていく．その際，ナップサックの残り容量よりもサイズが大きくて詰めることができない要素はスキップして次の要素を試すのである．

次に，巡回セールスマン問題に対する基本的な方法として，**最近近傍法**（nearest neighbor method）を紹介する．これは，適当な街から始め，まだ訪れていない街の中で現在の街に最も近い街へ移動する操作を，全ての街を訪れるまで反復する方法である．図 3 に最近近傍法の進行の様子を示す．距離は平面上のユークリッド距離とする．図中，街に付した番号は，訪問順序を表す．最近近傍法では，図のように，探索の最後の部分に遠い街が残されてしまう傾向があって，これが欠点とされている．

4.2 局所探索法

最適化問題 (1) において，実行可能解 $x \in F$ に対し，x に少しの変形を加えることによって得られる解集合 $N(x) \subset F$ を x の**近傍**（neighborhood）と呼ぶ．また，解 x から近傍 $N(x)$ 内の解を一つ生成するために x に加える変形操作を近傍操作と呼ぶ．局所探索法は，適当な解 $x \in F$ から始め，x の近傍 $N(x)$ 内に改善解 x' (すなわち最小化問題ならば $f(x') < f(x)$，最大化問題ならば

*1) http://www.diku.dk/~pisinger/codes.html

*2) http://www.keck.caam.rice.edu/tsp/などを参照．ただし，これは複数台の計算機を用いて長い時間をかけて得られた結果であり，実用的に解けたとは言いがたい．

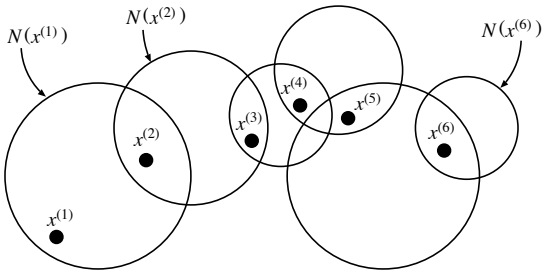


図 4　局所探索法の進行

$f(x') > f(x)$) があれば $x := x'$ と移動する操作を，近傍内に改善解が存在しなくなるまで反復する方法である．$N(x)$ 内に改善解が存在しない x を，近傍 N に関する**局所最適解**（locally optimal solution）と呼ぶ．図 4 に局所探索法の進行の様子を示す．図中 $x^{(k)}$ は k 番目の解である．

局所探索法が局所最適解に到達するまでの時間は，多項式時間である保証はないが，経験的にはたいていの場合決定変数の数に比例する程度といわれている．なお，簡単のため，局所探索法が探索の対象とする解を実行可能解に限定し，解は目的関数によって評価されるものとして説明を行ったが，一般にはこれに従う必要はなく，実行不可能解も含めて探索し，解の評価にも工夫を加えるほうが効果的な場合もある．

近傍 $N(x)$ 内の改善解は，一般には複数個存在するので，近傍をどのような順序で調べ，どの解を次の解として採用するかについては，様々な戦略（移動戦略という）が可能である．たとえば，

- 近傍 $N(x)$ 内をランダムな順序で調べて最初に見つかった改善解に移動する，
- $N(x)$ 内の解を全て調べて最良解に移動する，

などが代表的である．

近傍 $N(x)$ は，数学的には写像 $N : F \to 2^F$ であれば何でもよいが，性能の高い局所探索法を得るには，$|N(x)|$ が小さくて，$N(x)$ 内に改善解が存在する傾向が高くなるように定めることが重要である．これには，問題の性質を如何に取り込むかがポイントである．以下に巡回セールスマン問題に対する例を挙げる．巡回路 σ において隣り合う街のペア $\{\sigma(k), \sigma(k+1)\}$ を枝と呼ぶ．定

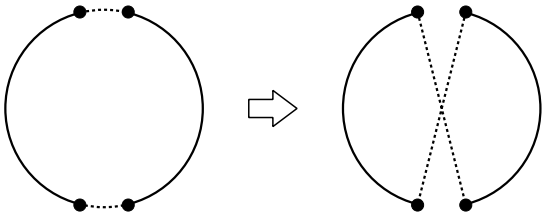


図 5　2-opt 近傍の近傍操作の例

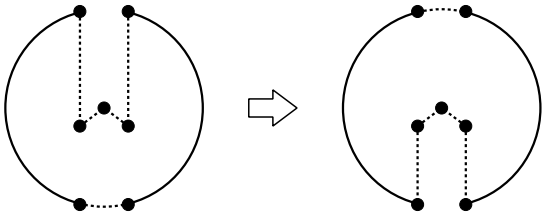


図 6　Or-opt 近傍の近傍操作の例

数 λ (≥ 2) に対して，現在の解から高々 λ 本の枝を入れ換えることにより得られる巡回路の集合を λ-opt 近傍と呼ぶ．2-opt 近傍の近傍操作の例を図 5 に示しておく．図中，破線は枝を，実線は枝が 1 本以上連続するもの（パスと呼ぶ）を表す．なお，3-opt 近傍は，σ のパスを他の位置に挿入することで得られる解を表すが，移動するパスに含まれる街の数を 3 以下に限定した場合は，とくに Or-opt 近傍と呼ばれている（図 6 参照）．この問題では，解に含まれる枝の距離が直接目的関数値に関わっているので，このように解の枝集合の変化が小さい近傍操作が有効である．

4.3 メタ戦略

近年，計算機性能が急速に向上したおかげで，欲張り法や局所探索法などの基本戦略は，高速に実現できるようになった．そのため，多少時間はかかっても，より精度の高い解を求める解法に対する要求が高まってきた．この目的を実現するための一般的枠組を提供しようとするのが，メタ戦略である．メタ戦略は，特定のアルゴリズムを指すのではなく，様々なアルゴリズムを含めた総称である．メタ戦略に含まれる代表的なものとして，**遺伝アルゴリズム**（genetic algorithm），**アニーリング法**（simulated annealing），**タブー探索法**（tabu search）などがある．遺伝アルゴリズムは

生物の進化に，アニーリング法は焼きなましの過程に着想を得た方法であるが，この他にも自然界の様々な現象にアイデアを得た方法が提案されており，このような自由な発想が応用できるところにメタ戦略の面白さがある．

メタ戦略は，(1) 過去の探索履歴を利用して新たな解を生成する，(2) 生成した解を評価し次の解の探索に必要な情報を取り出す，という操作の反復より成る．すなわち，生成された解のどのような情報を探索履歴として記憶するか，探索の履歴をどのように利用して新たな解を生成するか，に対する様々なアイデアの集合がメタ戦略であるといえる．

上記の 2 つのステップの最も基本的な形が局所探索法であり，ほとんどのメタ戦略は，局所探索法を基本としている．また，メタ戦略の多くは，「よい解同士は似通った構造を持っている」という概念に基づいて設計されている．例えば，局所探索によってよい解をいくつか求めた後，これらよい解と似通った構造を持つ解を集中的に探索することによって，周辺に潜んでいるよりよい解を発見しようとする方法がしばしば用いられる．このように，よい解の近くを集中的に探索しようとする考え方は，探索の**集中化**（intensification）と呼ばれ，メタ戦略の基本原理の一つである．一方，似通った構造の解を探索することに力を入れすぎると，同じ解を何度も探索してしまって，無駄が多くなる恐れもある．そこで，これまで生成してきた解とは構造の異なる解を生成することもときには必要である．この考え方は探索の**多様化**（diversification）と呼ばれ，メタ戦略のもう一つの基本原理である．これらの互いに相反する動作を如何にバランスよく組込むかがメタ戦略の成功の秘訣といえる．

メタ戦略の様々なアイデアを紹介するのは，紙面の都合上困難であるが，単純でしかも比較的高い性能が期待できる方法の一つとして，**反復局所探索法**（iterated local search）を紹介しておく．これは，過去の探索で得られた最良解 $x_{\rm best}$ にランダムな変形を加えたものを初期解として局所探索を行う操作を反復する方法である．初期解生成に用いるランダムな変形としては，適当な近傍の中からランダムに解を選ぶ方法が自然であるが，このための近傍に局所探索と同じ近傍を用いると，局所探索の 1 回の移動によってすぐに $x_{\rm best}$ に戻ってしまう可能性がある．これを避けるため，局所探索に利用する近傍より多少大きな近傍や，少し構造の異なる近傍を利用することが推奨される．例えば，巡回セールスマン問題で，局所探索法の近傍に 2-opt 近傍を用いたときには，図 7 に示すような double bridge 近傍をランダムな変形に利用するのが効果的であるといわれている（図 7 の 4 本の破線の枝は巡回路からランダムに選ぶ）．double bridge 近傍は，4-opt 近傍の特別な場合であるが，2-opt 近傍の操作を 2 回繰り返しても到達できないような解の集合になっており，$x_{\rm best}$ にすぐに逆戻りする現象を防ぐ効果があるのである．

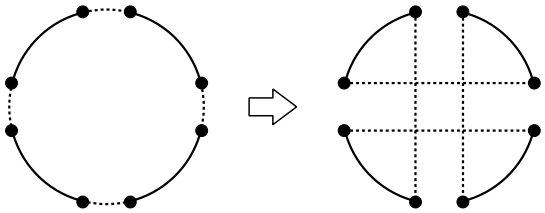


図 7　double bridge 近傍の近傍操作の例

4.4 近似解の精度

4 節で述べてきた様々な手法によって得られる解はどの程度よいのだろうか．これを議論するため，いくつか定義を行う．ある問題 Π の入力データ I に対し，アルゴリズム A によって得られる解の目的関数値を $A(I)$, 最適値を $OPT(I)$ と記す（簡単のため $OPT(I) > 0$ を仮定する）．このとき，これらの比 $A(I)/OPT(I)$ を Π の全ての入力データ I について考え，最小化問題ならばその最大値，最大化問題ならばその最小値を**近似度**（approximation ratio）と呼ぶ．近似度は，最小化問題ならば 1 以上，最大化問題ならば 1 以下となり，1 に近いほどよい．ある定数 ε (> 0) に対して，$|(\text{近似度}) - 1| \leq \varepsilon$ となることを保証できる近似アルゴリズムを，**ε-近似アルゴリズム**（ε-approximation algorithm）という．また，ε-近似

アルゴリズム A が，ε をパラメータとしてアルゴリズムの内部に含み，$\varepsilon\ (>0)$ を任意に指定できるとき，A は**近似スキーム**であるという．さらに，近似スキームが（ε を定数と見たとき）入力サイズに対する多項式時間アルゴリズムであるならば，**多項式時間近似スキーム**（polynomial time approximation scheme, PTAS）と呼び，また，計算時間が，入力サイズだけでなく，$1/\varepsilon$ に対しても多項式時間で抑えられる場合は，**全多項式時間近似スキーム**（fully polynomial time approximation scheme, FPTAS）と呼ぶ．

0-1 ナップサック問題に対する 4.1 節の欲張り法に対しては，近似度がいくらでも 0 に近づくような入力データが存在する．(たとえば，適当な定数 $\alpha > 1$ に対し，$n = 2$, $c_1 = a_1 = 1$, $c_2 = \alpha$, $a_2 = \alpha + 1$, $b = \alpha + 1$ を考えよ．）巡回セールスマン問題に対しては，距離行列が一般の場合は近似度 $2^{poly(n)}$（$poly(n)$ は n の任意多項式）ですら困難と考えられている．距離行列が三角不等式（任意の i, j, k に対し $d_{ij} \le d_{ik} + d_{kj}$）をみたすときは近似度 1.5 の多項式時間アルゴリズムが，さらにユークリッド距離のときは PTAS が知られている．しかし，この PTAS は計算時間が大きすぎて実用的ではなく，また，近似度 1.5（つまり誤差 50%!）というのは実際にはあまり意味のある上界とはいえない．一方，本稿で紹介した最近近傍法や λ-opt 近傍による局所探索法の近似度は理論的にはこれらに劣ることが示されているが，実際の性能は理論的上界よりもはるかによい．このように，近似度に対する理論的解析は必ずしも実際のアルゴリズムの性能を示す指針とはなっておらず，理論と現実の間にかなりのギャップがある．

理論的には FPTAS が理想的であるが，最後に 0-1 ナップサック問題に対する FPTAS の例を紹介しておこう．利得の最大値を $c_{\max} = \max_{j \in V} c_j$ と書く．与えられた $\varepsilon > 0$ に対して $K = \varepsilon c_{\max}/n$ とおき，各要素 j に対して $c'_j = \lfloor c_j/K \rfloor$ とする．この c'_j を利得として（a_j と b は元のデータ）3.2 節の動的計画法（詳しく説明した方ではなく，読者の宿題とした方）を適用し，得られた解 $\tilde{z}$ を出力する．まず，この方法の近似度が $1-\varepsilon$ 以上であることを示す．最適解の 1 つを z^* とする．利得 c'_j のもとでは解 $\tilde{z}$ は最適なので，$c_j/K - 1 < c'_j \le c_j/K\ (\forall j \in V)$ に注意すると

$$\begin{aligned} \sum_{j=1}^n c_j \tilde{z}_j &\ge K \sum_{j=1}^n c'_j z^*_j \\ &\ge \sum_{j=1}^n c_j z^*_j - nK \\ &= \sum_{j=1}^n c_j z^*_j - \varepsilon c_{\max} \\ &\ge (1-\varepsilon) \sum_{j=1}^n c_j z^*_j \end{aligned}$$

を得る．なお，最後の不等式は $\sum_{j=1}^n c_j z^*_j \ge c_{\max}$ による．最後に，動的計画法の計算時間は，$\sum_{j=1}^n c'_j n \le n^2 \max_{j \in V} c'_j = n^2 \lfloor c_{\max}/K \rfloor = n^2 \lfloor n/\varepsilon \rfloor$ に比例する時間で抑えられるので，n と $1/\varepsilon$ の多項式時間である．

5. まとめ

計算困難な組合せ最適化問題に対する様々な現実的アプローチを概観し，最後に近似精度に対する理論的成果にも触れた．最近では，後半に述べたメタ戦略や近似度に対する理論的解析の研究がとくに盛んである．局所探索法やメタ戦略については文献 1, 7)，近似度の理論的解析に関しては 3, 6) などを参照いただきたい．本稿が「組合せ最適化の研究を始めてみよう」と思うきっかけになれば幸いである．

参考文献

1) E.H.L. Aarts and J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, Chichester, 1997.
2) M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.
3) D.S. Hochbaum, ed., *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, Boston, 1997.
4) 茨木俊秀，組合せ最適化: 分枝限定法を中心として，産業図書，1983.
5) 茨木俊秀，福島雅夫，最適化の手法，共立出版，1993.
6) V.V. Vazirani, *Approximation Algorithms*, Springer, Berlin, 2001.
7) 柳浦睦憲，茨木俊秀，組合せ最適化: メタ戦略を中心として，朝倉書店，2001.

（やぎうら・むつのり，京都大学大学院情報学研究科）