メタヒューリスティクス事始め
―まずは局所探索法から―

梅谷　俊治，柳浦　睦憲

多くの組合せ最適化問題に対して厳密な最適解を効率よく求めることは困難であることが知られている．局所探索法はそのような問題に対する近似解法の基本的な戦略の 1 つであり，多くのメタヒューリスティクスは局所探索法にさまざまなアイデアを加えて拡張したものと位置づけることができる．本稿では，これからメタヒューリスティクスを設計する人を対象に，局所探索法の基本的な概念と効率的なアルゴリズムを実現するためのアイデアを具体例を交えながら解説する．

キーワード：組合せ最適化，局所探索法，メタヒューリスティクス

1. はじめに

現実社会に現れる組合せ最適化問題の多くに対して厳密な最適解を効率よく求めることは困難である．しかし，現実には，最適性の保証はなくとも十分に精度の高い解が現実的な計算時間で求まれば満足のいく場合が多い．近似解法はこのような目的に用いられる．

組合せ最適化問題に対する近似解法の基本的な戦略として局所探索法がよく知られている．局所探索法というと素朴で単純な手法と思われるかも知れない．しかし，局所探索法はその汎用性（多くの組合せ最適化問題に適用できる），容易性（アイデアが簡単で実装しやすい），実用性（簡単な実装でもある程度の性能が期待できる）などの利点により，大規模な組合せ最適化問題に対する強力な手法として広く利用されている．

また，より高度な近似解法を設計するための戦略としてメタヒューリスティクスがよく知られているが，その多くは局所探索法にさまざまなアイデアを加えて拡張したものと位置づけることができる．そのため，良いメタヒューリスティクスを設計するには，まず良い局所探索法を設計することが肝要となる．本稿では，これからメタヒューリスティクスを設計する人を対象に，局所探索法の基本的な概念と効率的なアルゴリズムを実現するためのアイデアを具体例を交えながら解説する．

うめたに　しゅんじ
大阪大学大学院情報科学研究科
〒 565–0871 大阪府吹田市山田丘 2–1
やぎうら　むつのり
名古屋大学大学院情報科学研究科
〒 464–8601 名古屋市千種区不老町

2. 組合せ最適化問題と局所探索法

最適化問題は一般的に以下のように表される．

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

f を目的関数，F を実行可能領域，制約条件を満たす解 $\boldsymbol{x} \in F$ を実行可能解と呼ぶ．$f(\boldsymbol{x})$ を最小にする実行可能解を最適解と呼び，そのような解の 1 つ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，問題 (1) は組合せ最適化問題と呼ばれる．そのような問題の一例として，巡回セールスマン問題がよく知られている．これは，n 個の都市とそれらの間の距離が与えられたとき，すべての都市をちょうど一度ずつ訪れて最初の都市に戻る巡回路のうち，総移動距離が最小のものを求める問題である．なお，本稿では都市間の距離は対称であるものとする（すなわち都市 i から j に移動する距離 d_{ij} が任意の i と j に対して $d_{ij} = d_{ji}$ を満たす）．

多くの組合せ最適化問題に対して，厳密な最適解を求めることは極めて困難であることが，計算の複雑さの理論により明らかにされてきた．NP 困難性はその代表例であり，巡回セールスマン問題をはじめとする多くの組合せ最適化問題が NP 困難であることが知られている．説明は省略するが，NP 困難問題の最適解を求めようとすると，最悪の場合すべての実行可能解を列挙するのと本質的に変わらない計算時間が必要であると予想されている（この予想が正しいかどうかは未解決で，P $\neq$ NP 予想として有名）．例えば，巡回セールスマン問題の解を列挙しようとすると，$(n-1)!$

通りあるが（最初に訪れる都市を固定しても一般性を失わない），これが現実的な方法でないことは，少し大きい n に対して実際に列挙法のプログラムを実行すればすぐにわかる．

現実には，最適性の保証はなくとも十分に精度の高い実行可能解が求まれば満足のいく場合が多く，近似解法はこのような目的に用いられる．局所探索法は組合せ最適化問題に対する近似解法の基本戦略である．局所探索法は，適当な実行可能解 $\boldsymbol{x} \in F$ から始めて，現在の解 $\boldsymbol{x}$ に少しの変形を加えて得られる解の集合 $N(\boldsymbol{x}) \subset F$ 内に改善解 $\boldsymbol{x}'$ (すなわち $f(\boldsymbol{x}') < f(\boldsymbol{x})$) があれば，現在の解 $\boldsymbol{x}$ から $\boldsymbol{x}'$ に移動する操作を反復する方法である．ここで，現在の解 $\boldsymbol{x}$ に変形を加える操作を近傍操作と呼び，それにより生成されうる解の集合 $N(\boldsymbol{x})$ を近傍と呼ぶ．現在の解 $\boldsymbol{x}$ の近傍 $N(\boldsymbol{x})$ 内に改善解がなければ局所探索法は終了する．このとき得られる解のように近傍内に改善解をもたない解を（その近傍の下での）局所最適解と呼ぶ．図 1 に局所探索法の進行の様子を示す．図中の各点 $\boldsymbol{x}^{(k)}$ は k 番目の解を表す．また，各点 $\boldsymbol{x}^{(k)}$ を中心とする点線で囲まれた円領域 $N(\boldsymbol{x}^{(k)})$ は $\boldsymbol{x}^{(k)}$ の近傍を表す．

例えば，巡回セールスマン問題では，解を辺（巡回路において隣り合う都市の対）の集合ととらえて，現在の解から辺を高々 λ 本交換して得られる解集合を近傍と定義する λ-opt 近傍 ($\lambda \geq 2$) がよく知られている．$\lambda = 2$ の場合の例を図 2 に示す．点が都市を，点対を結ぶ実線が辺を表し，それら 7 本の辺が巡回路を表す．図 2 では交差している 2 本の辺を入れ替えて新たな巡回路を得ている．

局所探索法はこのように非常に単純なアイデアに基づいている．しかし，その設計の自由度は大きく，以下に述べる探索空間，解の評価，近傍の定義，移動戦略などの要素を注意深く設計することで高性能なアルゴリズムを実現することが可能である．

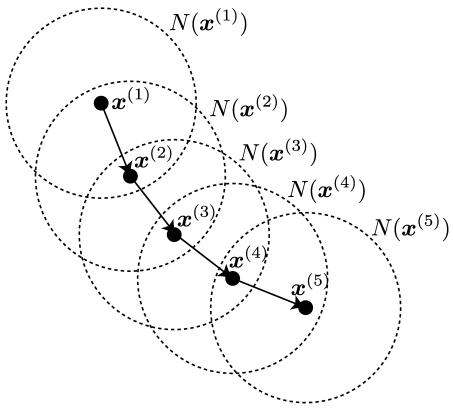


図 1　局所探索法の進行

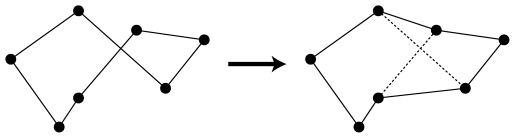


図 2　2-opt 近傍の近傍操作の例

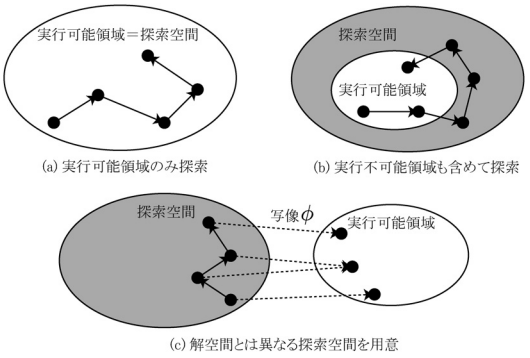


図 3　さまざまな探索空間

3. 探索空間と解の評価

探索の対象となる解全体の集合を探索空間と呼ぶ．初期解となる実行可能解を 1 つ見つけることと，近傍操作によって新たな実行可能解を生成することがともに容易であれば，実行可能領域をそのまま探索空間とすればよい（2 節では簡単のためそのような場合に限定して局所探索法の枠組みを説明した）．しかし，問題によっては実行可能領域とは異なる探索空間を定義するほうが有効な場合も多い．図 3 のように，探索方針は，(a) 実行可能領域のみを探索する，(b) 実行不可能解（すなわち $\boldsymbol{x} \notin F$ であるような解）も含めて探索する，(c) 解空間とは異なる探索空間を導入して，探索空間から解空間への写像 ϕ を用いて探索するの 3 通りに分類できる．図中の黒点は解を，実線の矢印は解の移動を表す．2 節の巡回セールスマン問題の例は (a) の一例である．以下に，(b) と (c) の例として，それぞれ集合分割問題と長方形詰込み問題に対する探索空間の設計例を紹介する．

3.1 集合分割問題

集合分割問題は，m 個の要素からなる集合 $M = \{1, 2, \ldots, m\}$，n 個の部分集合 $S_j \subseteq M$ ($j \in N = \{1, 2, \ldots, n\}$) とそれらのコスト c_j (> 0) が与えられたとき，集合 M のすべての要素 $i \in M$ をちょうど 1 回ずつ含む部分集合の組合せの中で，コストの総和が

最小となるものを求める問題である．係数 a_{ij} を，要素 i が部分集合 S_j に含まれていれば $a_{ij}=1$，含まれていなければ $a_{ij}=0$ と定義し，変数 x_j を，部分集合 S_j を選ぶときに $x_j=1$，選ばないときに $x_j=0$ を取る 0-1 変数として，集合分割問題を以下のように定式化できる．

$$\begin{aligned} &\text{最小化} && \sum_{j\in N} c_j x_j \\ &\text{制約条件} && \sum_{j\in N} a_{ij}x_j = 1,\ i\in M, \\ & && x_j\in\{0,1\},\ j\in N. \end{aligned} \tag{2}$$

この問題については，制約条件を満たす実行可能解 $\boldsymbol{x}$ が存在するかどうかを判定する問題自体が NP 完全であることが知られている．このように実行可能解を得ることすら難しい場合には，実行可能領域 F を探索空間とすることは現実的ではなく，一部もしくはすべての制約を緩和して，実行不可能解も探索の対象とする方法が有効である．集合分割問題では，例えば，すべての制約条件を緩和して 0-1 ベクトル全体の集合を探索空間とし，実行不可能解に対しては制約条件の違反度を表すペナルティに適当な重みをかけて目的関数に加えるペナルティ関数法がよく用いられる．具体的には，例えば制約式の左辺と右辺の差の絶対値をペナルティ，制約条件 i に対応する重みを $\alpha_i(>0)$ として，

$$\tilde{f}(\boldsymbol{x}) = \sum_{j\in N} c_j x_j + \sum_{i\in M} \alpha_i \left| \sum_{j\in N} a_{ij}x_j - 1 \right| \tag{3}$$

を解の評価関数とする．そして，近傍内に $\tilde{f}(\boldsymbol{x})$ の値がより小さい解があればそれに移動する操作を反復する．このとき，得られる局所最適解は実行可能解とは限らないので，現在の解 $\boldsymbol{x}$ とは別に探索中に評価した近傍解の中で最良の実行可能解を記憶しておき[1]，これを局所探索法の出力とする．

ペナルティ重み α_i が十分に大きければ実行可能解は得られやすくなるが，目的関数の小さな実行可能解を得るためにはペナルティ重みはあまり大きすぎないほうが良い傾向にある．しかし，ペナルティ重み α_i の調整は容易ではなく，適当な値を与えて局所探索法を 1 回適用するだけでは良い近似解はなかなか得られない．そこで，ペナルティ重みの更新と局所探索法を繰返し適用する方法がしばしば用いられる．例えば，直前の局所探索法で実行可能解が 1 つも得られなかった場合には，ペナルティ重みが小さすぎると判断して重みを増やす．このとき，得られた局所最適解 $\boldsymbol{x}$ における違反度が大きい制約ほど，そのペナルティ重みの増加量を大きくする．一方で，実行可能解が 1 つでも得られた場合には，ペナルティ重みが十分に大きいと判断してすべての制約の重みを減らすなどである．

3.2 長方形詰込み問題

長方形詰込み問題は，n 個の長方形の幅と高さおよび容器の幅が与えられたとき，長方形同士を重なりなく容器内に配置して，高さ（長方形の上辺の位置の最大値）を最小化する問題である．ただし，ここでは長方形を置く向きは決まっている（つまり回転は許されない）ものとする．

各長方形の配置座標は実数値を取り，また，それらを独立に定めたのでは重なりの排除が困難なことから，配置座標を直接探索の対象とすることは現実的ではない．そこで，実現可能な配置を表現するさまざまな手法が提案されている．

その 1 つに，順列を解の表現として，bottom-left 法（以下 BL 法）と呼ばれるアルゴリズムによって順列に対応する配置を計算する方法がある．BL 法は長方形を 1 つずつ配置していく方法で，各反復では次に置く長方形の座標を配置済みの長方形に重なりなく置ける最も低い位置の中の最も左に定める．図 4 に BL 法の配置例を示す．ここでは，左図に与えられた長方形 $(1,2,\ldots,6)$ をこの順に配置する例を考える．中央の図が長方形 3 を配置する場面を表しており，黒点が BL 法の条件を満たす位置（長方形 3 の左下の角を置く点）である．右図がすべての長方形を配置した結果を表している．

順列が変わると対応する配置も変わるので，順列を探索の対象とし，対応する配置の目的関数値を解の評価として局所探索を行う．探索空間は n 要素の順列全体であり，問題の実行可能領域とは全く別のものである．このような方法は，問題の決定変数を直接探索することが難しい問題に対してよく用いられる．なお，BL 法は簡潔でわかりやすい手法であるが，すべての順列

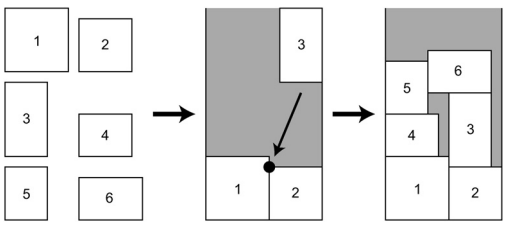


図 4 BL 法による配置例

[1] 実際に訪れた解だけではなくそれらの近傍の中で評価したすべての解を対象とする．

に対して BL 法を適用しても，それらの中に最適解が含まれない場合がある．一方で，探索空間の中に最適解に対応するものが含まれることを保証できる方法もいくつか知られている．

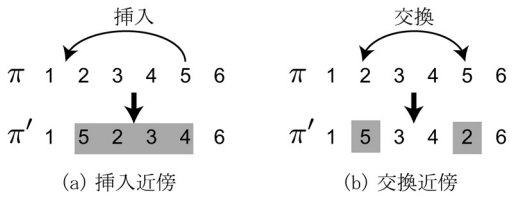
4. 近傍の定義と解の表現

近傍は局所探索法の設計において最も重要な要素の 1 つである．近傍内に改善解が含まれる可能性が高まるように，しかも近傍のサイズが大きくなりすぎないように設計する必要がある．

巡回セールスマン問題に対する近傍の例として，λ-opt 近傍を 2 節で紹介したが，改善解を探す計算量が大きくなりすぎないように，λ には通常 2 あるいは 3 程度の小さな定数が用いられる．また，この問題では，都市の訪問順序でも解（巡回路）を表現できる．このように順列 π で解を表せる問題に対しては，挿入近傍や交換近傍がよく用いられる（図 5）．挿入近傍は 1 つの都市を順列の他の位置に移動して得られる解集合，交換近傍は 2 つの都市の順列における位置を交換して得られる解集合である．このように，1 つの問題に対してさまざまな近傍を定義できるが，その中のどれを利用するか（複数でもよい）によって局所探索法の性能は変わる．

多くのメタヒューリスティクスは「良い解同士は似た構造を持っている」という近接最適性と呼ばれる概念に基づいて設計されている．これは，多くの問題に対して観測されている経験則である．近接最適性が成立していれば，良い解と似た解の中により良い解が見つかる可能性が高い．例えば，巡回セールスマン問題においては，巡回路の辺の長さ（つまり巡回路において隣り合う都市間の距離）の総和が巡回路長となるため，辺の選択が重要であるが，良い解同士には共通して含まれる辺が非常に多い傾向にあることが知られている．そのため，少数の辺を交換する 2-opt 近傍や 3-opt 近傍に基づく局所探索法によって良い解の周辺を集中的に探索できる．一方，巡回セールスマン問題に対しては，交換近傍はあまり有効ではない．交換近傍は，巡回路を構成する辺集合の中の 4 本の辺を一度に変更するものであり，2-opt 近傍や 3-opt 近傍と比較して，目的関数への影響が大きいことが原因と考えられる．このように，近接最適性の観点から，近傍操作の前後で解の評価値があまり大きく変化しないように設計するのが良いと言える．

近傍の大きさも重要なポイントである．λ-opt 近傍のようにパラメータによって近傍の大きさを設定できる近傍は多数存在し，通常は大きな近傍を用いることで局所探索法で得られる解の精度は向上する．小さな近傍の下では局所最適解であっても大きな近傍では局所最適でなくなる可能性が高く，近傍を大きくすることで小さな近傍の下での局所最適解から脱出できるからである．一方で，近傍を大きくすると，近傍内を探索するための計算時間が大きくなってしまうので，近傍の設計には解の精度と計算時間のトレードオフが重要である．また，小さな近傍と大きな近傍を組み合わせて用いることもしばしば有効である．例えば，2-opt 近傍に基づく局所探索法の後に続けて 3-opt 近傍に基づく局所探索法を実行するというように，まず小さい近傍で探索した後に大きい近傍を用いると，大きな近傍による探索にかかる計算時間を抑えつつ解の精度を上げる効果が期待できる．



(a) 挿入近傍　(b) 交換近傍

図 5　挿入近傍と交換近傍の例

5. 移動戦略

一般に，近傍内に改善解は複数存在する．よって，近傍内をどのような順序で調べ，どの改善解に移動するかによって，局所探索法の振る舞いは変わる．このルールを移動戦略と呼ぶ．近傍内の解をランダムもしくはある一定の順序で調べて，最初に見つかった改善解に移動する即時移動戦略と，近傍内のすべての解を調べたうえで最良の改善解に移動する最良移動戦略が代表的である．多くの場合，適当な初期解から始めて局所最適解に到達するまでの計算時間は，1 回の移動の手間の少ない即時移動戦略のほうが短く，最終的に得られる局所最適解の精度には大きな差はない．

即時移動戦略を用いる場合には，近傍内の解の探索順序が局所探索法の性能に影響を与えるので注意が必要である．例えば，与えられた変数の添字の順番に従って毎回添字の値の小さいほうから探索すると，実装は簡単であるが近傍の探索に偏りが生じてしまい，到達し難い局所最適解が生じる可能性がある．これを避けるには，例えば以下の方法がある．近傍内の解を一巡するランダムな順序を定めたリストを予め用意しておき，この順に従って探索を行う．そして，改善解が見

つかって解の移動を行ったのち，新たな解の近傍を探索する際には，リストの先頭からではなく，改善解を生成した近傍操作の次の候補から探索を行う．その際，リストを環状のリストとみなし（つまりリストの最後尾の次の候補はリストの先頭），改善解が見つからない限りリストを一周して近傍のすべてを調べる．このようにすることで，近傍内の探索の極端な偏りや添字による影響を避けることが可能となる．

6. 局所探索法の効率化

近傍内の改善解を発見するための探索を近傍探索と呼ぶ．局所探索法では計算時間の大部分が近傍探索に費やされるので，近傍探索の効率化はアルゴリズム全体の高速化に直結する．また，近傍探索を高速化することにより，同程度の計算時間でより大きな近傍を探索できるようになり，その結果，解の精度が向上するという効果も期待できる．ここでは，近傍探索の効率化を実現する方法として，(1) 解の評価値の計算を高速化する方法と，(2) 改善の可能性のない解の探索を省略する方法の 2 つを紹介する．これらを実現するためには，個々の問題の構造をうまく活用する必要があるが，基本的な考え方は多くの問題に共通している．

6.1 解の評価値の計算の高速化

局所探索法では，近傍操作によってごく少数の変数のみ値が変化するため，現在の解 $\boldsymbol{x}$ と近傍解 $\boldsymbol{x}' \in N(\boldsymbol{x})$ の間で値が変化した変数に関わる部分のみ再計算すれば，目的関数値の変化量 $f(\boldsymbol{x}') - f(\boldsymbol{x})$ を高速に計算できる場合が多い．例えば，巡回セールスマン問題では，巡回路の距離を一から計算するには都市の数 n に比例する計算時間を要するが，2-opt 近傍の近傍操作では，$f(\boldsymbol{x})$ に追加する 2 本の辺の距離を加えて，削除する 2 本の辺の距離を引くことで，$f(\boldsymbol{x}')$ を定数時間で計算できる．

別の例として，集合分割問題において (3) 式で定義した評価関数 $\tilde{f}(\boldsymbol{x})$ の変化量を計算する方法を紹介する．ここでは，現在の解 $\boldsymbol{x}$ とのハミング距離が 1 である解の集合を近傍とする 1 反転近傍を考える．1 反転近傍の解には，$x_j = 0 \to 1$ と反転して得られるものと，$x_j = 1 \to 0$ として得られるものの 2 種類があり，前者（後者）のみからなる近傍を追加近傍（削除近傍）と呼ぶことにする．変数の数 n と制約条件の数 m に加えて，制約条件の左辺に現れる係数 a_{ij} が 1 であるものの総数を σ と定義すると，何も工夫しなければ 1 つの近傍解 $\boldsymbol{x}' \in N(\boldsymbol{x})$ の評価関数値 $\tilde{f}(\boldsymbol{x}')$ を計算するのに $\mathrm{O}(\sigma)$ 時間かかる．

このように，評価関数値の計算に時間がかかる問題に対しては，(1) 近傍内の解の評価に必要な情報を補助記憶に持ち，(2) 現在の解が移動する際に補助記憶を更新する方法がしばしば有効である．局所探索法では，近傍内の解を評価する回数に比べて，現在の解が移動する回数ははるかに少ない場合が多いので，補助記憶の更新に多少時間がかかっても，全体では十分な高速化が実現できる．

現在の解 $\boldsymbol{x}$ からある変数 x_j の値を $x_j = 0 \to 1$ と反転して得られる近傍解 $\boldsymbol{x}'$ と $\boldsymbol{x}$ との評価関数値の差を $\Delta\tilde{f}_j^+(\boldsymbol{x}) = \tilde{f}(\boldsymbol{x}') - \tilde{f}(\boldsymbol{x})$ と定義し，$x_j = 1 \to 0$ と反転する場合についても同様に $\Delta\tilde{f}_j^-(\boldsymbol{x})$ と定義する．これらは制約条件 i の左辺値 $s_i(\boldsymbol{x}) = \sum_{j \in N} a_{ij} x_j$ を用いて

$$\Delta\tilde{f}_j^+(\boldsymbol{x}) = c_j + \sum_{i \in S_j} \alpha_i \left\{ |s_i(\boldsymbol{x})| - |s_i(\boldsymbol{x}) - 1| \right\}$$

$$\Delta\tilde{f}_j^-(\boldsymbol{x}) = -c_j + \sum_{i \in S_j} \alpha_i \left\{ |s_i(\boldsymbol{x}) - 2| - |s_i(\boldsymbol{x}) - 1| \right\} \quad (4)$$

と計算できる．ここで，制約条件 i の左辺値 $s_i(\boldsymbol{x})$ をあらかじめ計算して補助記憶に持てば，評価関数値の変化量 $\Delta\tilde{f}_j^+(\boldsymbol{x}), \Delta\tilde{f}_j^-(\boldsymbol{x})$ は各 j について $\mathrm{O}(|S_j|)$ 時間で計算できる．また，$x_j = 0 \to 1$ と反転して現在の解が $\boldsymbol{x}$ から $\boldsymbol{x}'$ に移動した際には，各制約条件 $i \in S_j$ に対して $s_i(\boldsymbol{x}') \leftarrow s_i(\boldsymbol{x}) + 1$ と計算すれば，補助記憶も $\mathrm{O}(|S_j|)$ 時間で更新できる．$x_j = 1 \to 0$ と反転する場合も同様に各制約条件 $i \in S_j$ に対して $s_i(\boldsymbol{x}') \leftarrow s_i(\boldsymbol{x}) - 1$ とすればよい．

さらなる高速化としては，各制約条件 i の左辺値 $s_i(\boldsymbol{x})$ に加えて，評価関数値の変化量 $\Delta\tilde{f}_j^+(\boldsymbol{x}), \Delta\tilde{f}_j^-(\boldsymbol{x})$ を直接補助記憶に持つ方法もある．この場合は，現在の解が移動した際の補助記憶の更新が複雑になるものの，各近傍解 $\boldsymbol{x}'$ を定数時間で評価できる．

6.2 近傍の縮小

局所探索法では，目的関数や制約条件の構造を利用して，評価関数値が改善するための必要条件を満たす近傍操作に探索を限定できる場合がある．以下に，巡回セールスマン問題における 2-opt 近傍の例を紹介する．

現在の巡回路から辺 $\{i_1, i_2\}$ と $\{i_3, i_4\}$ を除き，新たに $\{i_2, i_3\}$ と $\{i_4, i_1\}$ を加える 2-opt 近傍の操作を考える．この操作を，図 6 のようにまず辺 $\{i_1, i_2\}$ を削除して $\{i_2, i_3\}$ を追加し，その後辺 $\{i_3, i_4\}$ を削除して $\{i_4, i_1\}$ を追加する，と 2 段階に分けて考える（図の左の輪は巡回路を模式的に表し，その左右の弧はそれぞれ都市 i_1 と i_3 および i_2 と i_4 をつなぐ複数の都市を含む路を表す）．これを i_1 を始点とする辺交換の

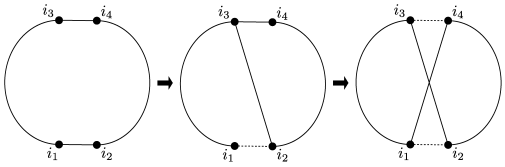


図 6　2-opt 近傍操作を 2 段階に分けた例

操作と呼ぶ．1 段階目で辺 $\{i_1, i_2\}$ と $\{i_2, i_3\}$ を決めると，2 段階目の辺の候補は $\{i_3, i_4\}$ と $\{i_4, i_1\}$ の一意に決まるため，1 段階目の組合せをすべて調べればよい．

このような 2-opt 近傍の操作を適用して改善解が得られるならば，$d_{i_2 i_3} + d_{i_4 i_1} < d_{i_1 i_2} + d_{i_3 i_4}$ が満たされる（d_{ij} は都市 i, j 間の距離）．これが成り立つためには，$d_{i_2 i_3} < d_{i_1 i_2}$ と $d_{i_4 i_1} < d_{i_3 i_4}$ のうち少なくとも一方が満たされているはずである．そこで，はじめに削除する辺 $\{i_1, i_2\}$ を決めたとき，追加する辺 $\{i_2, i_3\}$ の候補を $d_{i_2 i_3} < d_{i_1 i_2}$ を満たす辺のみに限定する．ただし，各 i_1 に対して接続する 2 本の辺両方の端点を i_2 の候補として辺交換の操作を考える．すると，$d_{i_2 i_3} \geq d_{i_1 i_2}$ かつ $d_{i_4 i_1} < d_{i_3 i_4}$ であるような近傍解は，i_3 を始点とする辺交換の操作において探索の対象となる．したがって，このように探索対象を制限しても，2-opt 近傍内の改善解を逃さないことを保証できる．通常は，探索で選ばれた巡回路に含まれる辺の距離はかなり短い場合が多いため，この方法によって近傍内で実際に評価する解の数を大幅に縮小できる．

1 段階目で削除する辺 $\{i_1, i_2\}$ に対して $d_{i_2 i_3} < d_{i_1 i_2}$ を満たす辺の候補 $\{i_2, i_3\}$ を効率良く列挙する方法として，都市 i に対して距離 d_{ij} の短い順に都市 j の番号を記憶した近傍リストを用いる方法が知られている．都市 i_2 に対するリストを前から順に走査することで $d_{i_2 i_3} < d_{i_1 i_2}$ を満たす辺のみを列挙できる．しかし，完全な近傍リストをすべての都市に対して準備するには $\mathrm{O}(n^2)$ の領域が必要となるため，通常は，適当なパラメータ γ $(0 \leq \gamma \leq n)$ を用意して，各都市 i に対して距離 d_{ij} の小さいほうから γ 番目までを近傍リストに記憶する．このような制限を加えると改善解を逃さない保証はなくなってしまうが，代表的なベンチマーク問題例などに対する計算実験により，n が相当大きい場合でも，γ の値は 20 程度で十分な性能が得られることが観測されている [2]．

局所探索法では，一度探索して改善が生じなかった近傍操作を，その後の解の移動によって改善の可能性が再び生じるまで探索の候補から除いても問題はない．このように探索の必要のない近傍操作にマークをつけることで無駄な探索を省略できることがある．例えば，集合分割問題における追加近傍では，ある時点で $\Delta \tilde{f}_j^+(\boldsymbol{x}) \geq 0$ であれば，少なくとも 1 つの制約条件 $i \in S_j$ の左辺値 $s_i(\boldsymbol{x})$ が変わるまで変数 x_j を探索の候補から除いても問題ないことがわかる．

巡回セールスマン問題における 2-opt 近傍では，don't-look bit と呼ばれるマークを利用した方法が知られている．まず，すべての都市に対してマークを 0 とする．都市 i_1 を始点とする辺交換の操作をすべて試しても改善解が得られなかった場合は，都市 i_1 のマークを 1 に変更する．その後，都市 i_1 に接続している 2 本の辺のうち少なくとも一方が巡回路から削除されたとき，i_1 のマークを 0 に戻す．このようなマークを用いて，マークが 0 である都市を始点とする辺交換の操作に探索を限定する．この方法は，近傍内に改善解があれば必ず見つけるという保証はないが，解の精度をほとんど悪化させることなく計算時間を大幅に短縮できることが観測されている．

7. おわりに

局所探索法は，組合せ最適化問題に対する近似解法の基本戦略であり，単純なアイデアに基づいている．しかし，その設計における自由度は大きく，各要素を注意深く設計することで高性能なアルゴリズムを実現しうる．最近では，実用的な解法としてより高度なメタヒューリスティクスが広く利用されている．その多くの基本要素である局所探索法の性能を高めることは，それを基礎とするメタヒューリスティックアルゴリズムの性能を高めることにつながる．局所探索法の実現方法について興味を持たれた方は文献 [1, 3, 4] などを参照いただきたい．

参考文献

[1] E. H. L. Aarts and J. K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, 1997.
[2] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study," in: E. H. L. Aarts and J. K. Lenstra, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, pp. 215–310, 1997.
[3] 久保幹雄，J. P. ペドロソ，『メタヒューリスティクスの数理』，共立出版，2009.
[4] 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.

