局所探索法とその拡張—タブー探索法を中心として

Local Search and its Extensions—Focusing on Tabu Search

野々部 宏司＊　柳浦 睦憲＊＊

＊法政大学デザイン工学部，〒 102-8160 東京都千代田区富士見 2-17-1
＊＊名古屋大学大学院情報科学研究科，〒 464-8603 名古屋市千種区不老町
＊Faculty of Engineering and Design, Hosei University, 2-17-1 Fujimi, Chiyoda, Tokyo 102-8160, Japan
＊＊Graduate School of Information Science, Nagoya University, Furocho, Chikusaku, Nagoya 464-8603, Japan
＊E-mail: nonobe@hosei.ac.jp
＊＊E-mail: yagiura@nagoya-u.jp

キーワード：局所探索（local search），組合せ最適化（combinatorial optimization），タブー探索法（tabu search），アニーリング法（simulated annealing），移動戦略（move strategy）
JL 0006/08/4706-0493

1. はじめに

厳密な最適解を効率よく求めることが困難である組合せ最適化問題に対し，良質の解を少ない計算時間で探索するための基本戦略として，**局所探索法**（local search）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しれないが，汎用性（解の実行可能性と目的関数値が計算できれば適用可能），容易性（アイデアが簡単で基本的なものなら手軽に実装可能），実用性（基本的な局所探索法でもある程度の性能を実現可能）といった利点があることから，組合せ最適化における手軽で強力なツールとして欠かせないものとなっている．しかし現実には，単純な局所探索法のみでは満足のいく結果が得られず，より高性能なアルゴリズムが必要となることも多い．この目的のために，近年では**メタヒューリスティクス**（metaheuristics）と呼ばれる手法がよく用いられている．

メタヒューリスティクスとは，最適化問題（とくに組合せ最適化問題）に対する実用的な探索手法を設計するための一般的な枠組みを与えるものである．これには数多くの手法やアイデアが含まれるが，その中には，局所探索法を基本として，その性能を高めるための手法と位置付けられるものが多数存在する．代表的なものとしては，GRASP 法，反復局所探索法，アニーリング法，タブー探索法などが挙げられる．本稿では，これらの手法の基礎となる局所探索法について解説した後，とくにアニーリング法とタブー探索法を取り上げて，そのアイデアを紹介する．

ところで，メタヒューリスティクスには，遺伝アルゴリズムやアント法，memetic アルゴリズムなど，複数の解を保持しながらそれらを集団として改善していくタイプの手法も存在し，総称して多点探索などと呼ばれる．これらの手法も，その内部に局所探索法（もしくは同様の考え方）を組み込んでいることが多く，その意味で局所探索法の拡張と見なすことも可能であるが，本稿では割愛する．多点探索型のアルゴリズムについては本特集の他の解説[10), 11), 14), 16)]で詳しく説明されているので，そちらを参照頂きたい．

2. 局所探索法

アニーリング法やタブー探索法などを用いて高性能なアルゴリズムを実現するためには，まず，その基礎となる局所探索法を適切に設計することが重要である．本節では，局所探索法の基礎について，具体例を交えながら説明する．

簡単な例から始めよう．巡回セールスマン問題は，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 1 度ずつ訪れて出発地に戻る巡回路のうち，総移動距離最小のものを求める問題である．都市を点，距離をそれらの間のユークリッド距離として，巡回路において隣接する 2 都市を辺で結ぶとき，図 1 の左の巡回路をその右のものへと少し修正すると，総移動距離は短くなる．適当な巡回路から始め，このような小さな修正を可能な限り繰り返すことで，ある程度よい巡回路にたどり着くと予想できる．このような手法を一般に局所探索法，あるいは反復改善法，山登り法などと呼ぶ．

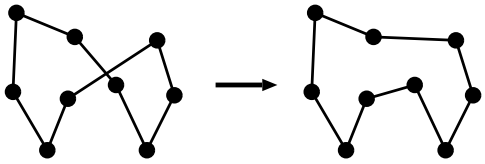

図 1　巡回路の修正例
交差している 2 本の辺を取り除き，新たに 2 本追加することで，総移動距離が減少する．

最適化問題は一般的に以下のように表される．

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

ここで，$f : F \to R$（R は実数の集合）を**目的関数**，F を**実行可能領域**と呼ぶ．F は制約条件を満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**と呼び，その 1 つを見つけることが最適化問題の目標である．とくに F が組合せ的な構造をもつ場合，(1) は組合せ最適化問題と呼ばれる．

ある解 x に加える小さな修正を**近傍操作**，そのような操作により得られる解の集合 $N(x)$ を**近傍**（neighborhood）と呼ぶ．局所探索法は，適当な初期解から始め，現在の解 x よりもよい解 x' が近傍 $N(x)$ 内に存在すれば $x := x'$ と置き換える操作を，可能な限り繰り返す方法である．近傍内によりよい解が存在しない解は**局所最適解**と呼ばれ，局所最適解が得られた時点で，局所探索法は終了する．

2.1 近傍

近傍は局所探索法の設計において最も重要な要素のひとつである．近傍内に改善解が含まれる可能性が高まるように，かつ，近傍のサイズが大きくなりすぎないよう設計することが望ましい．以下に，例をいくつか挙げる．

巡回セールスマン問題では，都市の訪問順序で解（巡回路）を表現できる．このように順列 σ 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 2）．挿入近傍は 1 つの都市を順列の他の位置に移動することで得られる解集合，交換近傍は 2 つの都市の順列における位置を交換することで得られる解集合である．

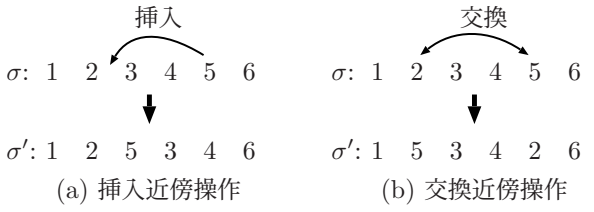

図 2 挿入近傍と交換近傍の近傍操作の例
1〜6 は都市番号を表し，σ と σ' はそれぞれ近傍操作適用前と適用後の順列を表す．

また，解を辺の集合と捉え，辺に着目した近傍を定義することもできる．よく知られているものとして，**λ-opt 近傍** ($\lambda \geq 2$) と **Or-opt 近傍**がある．λ-opt 近傍は，現在の解から，辺を高々 λ 本交換することによって得られる解集合であり，通常 $\lambda = 2$ か 3 が用いられる．図 1 は 2-opt 近傍の近傍操作の例である．この図からも分かる通り，2-opt 近傍操作は，訪問順序を表す順列 σ に対し，その連続する一部分を反転する操作に対応する（ただし，順列の最初と最後はつながっていると考える）．また，4-opt 近傍操作は，交換近傍操作を特別な場合として含む．Or-opt 近傍は，現在の巡回路において連続する 3 つ以下の都市を他の位置に挿入することによって得られる解集合である．図 3 に近傍操作の例を示す．図中，点線は 1 本の辺，波線は複数の辺からなるパスを表す．これは挿入近傍操作の自然な拡張であり，また，3-opt 近傍操作の特別な場合となっている．

どの近傍が効果的であるかは問題によるが，巡回セールスマン問題に対しては，2-opt 近傍や 3-opt 近傍が効果的で，交換近傍はあまり有効ではない．これは，目的関数値（総移動距離）が，辺集合によって定まることを考えれば直感的には理解しやすい．交換近傍操作は，巡回路を構成する辺集合の中の 4 本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．

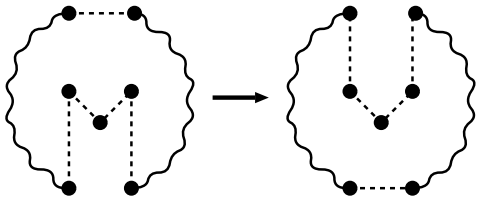

図 3 Or-opt 近傍の近傍操作の例
連続する 3 つ以下の都市を他の位置に挿入（点線は 1 本の辺，波線は複数の辺からなるパスを表す）．

なお，局所探索法では，「近傍内に改善解が存在するかどうかを判定し，存在するならばその 1 つを求める」という処理を繰り返し行わなくてはならない．そのため，この計算を高速に行えるように近傍を設計したり，データ構造を工夫したりすることが重要である．また，現在の解を x として，近傍解 $x' \in N(x)$ の目的関数値 $f(x')$ を計算する際，$f(x')$ を直接計算するのではなく，x の目的関数値との差 $f(x') - f(x)$ を考えることで，計算を高速化できることが多い．例えば，巡回路の長さを一から計算するには都市数 n に比例する時間を要するが，2-opt 近傍操作の場合は，$f(x)$ に追加する 2 本の辺の長さを加え，削除する 2 本の辺の長さを引くことで，$f(x')$ を定数時間で計算できる．

2.2 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探索し，どの改善解に移動するかによって，局所探索の振舞いは異なる．これを定めるルールを**移動戦略**（move strategy）という．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いき，改善解が見つかった時点で残りの解を見ることなくその解に移動する**即時移動戦略**と，近傍内のすべての解を調べた上で最良のものに移動する**最良移動戦略**の 2 つが代表的である．これらを比べると，多くの場合，即時移動戦略のほうが高速であり，最終的に得られる局所最適解の精度には大きな差がない傾向にある．一方，近傍内の解の評価値を表として記憶しておき，解の移動の際にその更新を高速に行うといったデータ構造の工夫が可能な場合，最良移動戦略のほうが有利な場合もある．

2.3 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**（search space）という．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な場合には，実行可能領域をそのまま探索空間とすればよい（1 つめの条件は，初期解を得るために必要）．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がある．以下，そのような例を 2 つ紹

介しよう．

（1） 一般化割当問題

一般化割当問題は，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 $I = \{1, 2, \ldots, m\}$ に割当てるとき，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．それぞれの仕事を必ずいずれか 1 つのエージェントに割当てなくてはならず，また，各エージェントに割当てられた仕事の総資源要求量が，そのエージェントの利用可能資源量を越えないようにしなくてはならない．仕事 j をエージェント i に割当てるときに値 1，そうでないときに値 0 をとる 0-1 変数 x_{ij} を用いて，一般化割当問題は以下のように定式化できる．

$$\text{最小化} \quad cost(x) = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \qquad (2)$$

$$\text{制約条件} \quad \sum_{j \in J} a_{ij} x_{ij} \leq b_i, \quad \forall i \in I \qquad (3)$$

$$\sum_{i \in I} x_{ij} = 1, \quad \forall j \in J \qquad (4)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in I,\ \forall j \in J. \qquad (5)$$

この問題において，実行可能領域 F は制約 (3)〜(5) を満たす 0-1 ベクトル x 全体の集合であるが，そのような x が存在するか否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なく，一部の制約を緩和し，実行不可能解も探索の対象とする方法が有効である．一般化割当問題の場合には，例えば，資源制約 (3) を緩和し，制約 (4) および (5) を満たす 0-1 ベクトル全体の集合を探索空間とする方法が考えられる．この場合の近傍には，

- シフト近傍: ある 1 つの仕事 j に対して，その割当て先 i を別のエージェント i' に変更（$x_{ij} = 1, x_{i'j} = 0$ を $x_{ij} = 0, x_{i'j} = 1$ に変更）することで得られる解の集合
- 交換近傍: ある 2 つの仕事 j_1 と j_2 に対して，それらの割当て先 i_1 と i_2 を交換（$x_{i_1 j_1} = x_{i_2 j_2} = 1, x_{i_1 j_2} = x_{i_2 j_1} = 0$ を $x_{i_1 j_1} = x_{i_2 j_2} = 0, x_{i_1 j_2} = x_{i_2 j_1} = 1$ に変更）することで得られる解集合

などがよく用いられる．

このように実行可能解と実行不可能解が混在する探索空間を用いる場合，目的関数値によって解の評価を一律に行うことは適切ではない．実行不可能解に対しては，制約の違反度をペナルティとして，これに適当な重みをかけて目的関数に加える方法がよく用いられる．例えば，一般化割当問題の例では，制約 (3) のそれぞれの i に対応する重みを α_i (≥ 0) として，ペナルティつきのコスト

$$cost(x) + \sum_{i \in I} \alpha_i \cdot \max \left\{ \sum_{j \in J} a_{ij} x_{ij} - b_i, 0 \right\}$$

を解の評価に用いるなどである（この値が小さいほど評価が高い）．このとき，ペナルティ重み α_i を十分大きくすれば，近傍探索において実行可能解が優先的に選択されることになるが，有効な探索を行うためにはペナルティ重みはあまり大き過ぎないほうがよい傾向にあり，この調整は難しい．そこで，制約の厳しさの程度や目的関数値とのバランスに応じてペナルティ重みを探索の進行とともに適応的に調整する方法がしばしば効果を発揮する（3.4 節参照）．

（2） パッキング問題

n 個の矩形の幅と高さ，および容器の幅が与えられたとき，矩形同士を重複なく容器内に配置し，高さ（矩形の上辺の位置の最大値）の最小化を図る問題を考える．ただし，ここでは矩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．

各矩形の配置位置には無限の可能性があり，また，それらを独立に定めたのでは重複の排除が困難であることから，配置座標の組合せを直接探索の対象とすることは現実的でない．そこで，実行可能な配置を表現する様々な手法が提案されている．その一つに，順列を解表現とし，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，矩形を 1 つずつ配置していく方法で，各反復では，次に置く矩形の位置を，配置済みの矩形に重複なく置ける最も低い位置の中の最も左に定める．図 4 に順列 $(1, 2, \ldots, 6)$ に対する BL 法の動作例を示す．

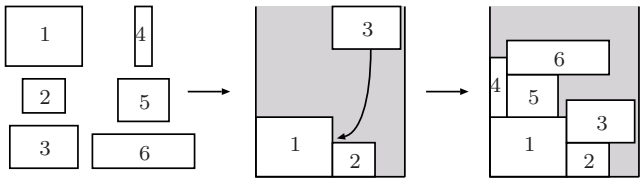

図 4　BL 法による配置例
順列にしたがって矩形を 1 つずつ，配置済みの矩形に重ならない最も低い位置の中の最も左に配置．

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のである．（実は，この方法では最適配置に対応する順列が存在しない場合がある．最適配置に対応する解表現の存在を保証する表現法も知られている．）探索空間は n 要素の順列全体の集合であり，問題の実行可能領域とは全く異なる．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．

探索空間の例をいくつか挙げたが，これらは，(a) 実行可能領域のみを探索する，(b) 実行不可能領域も含めて探

索する，(c) 解空間とは異なる探索空間を導入し，探索空間から解空間への写像 π を用いて探索を行う，の 3 通りに分類できる．巡回セールスマン問題の例は (a)，一般化割当問題の例は (b)，パッキング問題の例は (c)（写像 π は BL 法で定義される）にそれぞれ対応する．

3. 局所探索法の拡張手法

通常，局所最適解の中には精度の低いものも存在するため，局所探索法を 1 度適用しただけでは，そのような解を出力して探索が終了してしまう危険性がある．これを軽減する工夫として，異なる初期解を用いて局所探索法を何度も繰り返し実行する方法（多スタート局所探索法と呼ばれる）や，局所探索法自体の性能を高めるため，移動戦略や評価関数を前節で述べた基本的なものから変更するなどのアイデアが種々提案されている．これらを採り入れた手法は一般にメタヒューリスティクスと呼ばれる．

3.1 改悪解への移動

局所探索法は解の改善を繰り返し行う方法であり，解の改善ができなくなったとき，すなわち，局所最適解が 1 つ得られた時点で探索は終了する．しかし，多くの問題において，局所最適解の周囲にはさらによい解が潜んでいる可能性が高いことが経験的に知られている．そこで，そのような解を見つけるため，改悪解への移動を許し，局所最適解で探索が停止しないようにする方法が考えられる．以下に，例を 2 つ挙げる．

（1） アニーリング法

アニーリング法（simulated annealing）の移動戦略は，現在の解 x の近傍 $N(x)$ 内から解 x' をランダムに 1 つ選び，それが改善解であれば確率 1 で，改善解でない場合でも，その評価値に応じた確率（遷移確率と呼ばれる）で x から x' へ移動する，というものである．この操作を 1 反復とし，解の移動が行われなかった場合には，再び x を基点として同様の操作を繰り返す．遷移確率は，x と x' の評価値をそれぞれ $f(x)$ と $f(x')$，それらの差を $\Delta = f(x') - f(x)$，さらに $t\ (>0)$ をパラメータとして，$e^{-\Delta/t}$ で定義される（x' が改悪解であれば，$\Delta > 0$ となり遷移確率の値は 1 未満となる）．パラメータ t は**温度**と呼ばれ，温度が高いほど改悪解への移動が起きやすい．ここで,「温度」という用語は，アニーリング法が物理現象の焼きなましを模擬したものであることに由来する．探索の初期の段階では，温度 t を高めに設定することで探索空間をある程度自由に動きまわれるようにし，反復回数が増えるにつれて t を徐々に下げていくことで，その動きを改善解への移動のみに制限していく．

温度の更新は，通常，ある一定の反復回数（例えば近傍サイズの数倍）ごとに行い，その更新方法は冷却スケジュールと呼ばれる．冷却スケジュールとしてはいろいろな方法が提案されているが，パラメータ $\beta\ (0 < \beta < 1)$ を用意し，$t := \beta \cdot t$ とする方法（幾何冷却法と呼ばれる）が簡便で実用的とされている．

アニーリング法では，探索を永遠に継続することが可能であるため，探索の終了条件をあらかじめ定めておく必要がある．これには,「温度がある一定値を下回った」,「近傍解への移動（もしくは改善解への移動）が起きない反復が一定回数続いた」などの条件がよく用いられる．

（2） タブー探索法

タブー探索法（tabu search）は，現在の解 x から，近傍 $N(x)$ 全体，もしくはその一部の中で最良の解 x' に移動する，という戦略を基本とする．x' が改悪解であっても解の移動が強制的に行われるため，局所最適解で探索が終了することなく，解の移動が継続される．しかし，現在の解 x が局所最適解である場合，x から近傍解 x' に移動した後，同様の操作で $N(x')$ 内の最良の解を求めると，再び元の x に戻ってしまう可能性がある．一般に，いくつかの解を経由して元の解に戻ってしまう現象のことをサイクリングと呼ぶ．タブー探索法では，**タブーリスト**（tabu list）と呼ばれる解集合 T を定義し，これに含まれる解（禁止解と呼ばれる）への移動を禁止することで，サイクリングを防止する．ここで，タブーリストは，探索の状況に応じて随時更新される．

タブーリスト T の定義方法としてはいろいろ考えられる．例えば，最近探索した解の集合を T として，これを直接記憶しておく方法もその一つである．しかし，この方法は効果的な探索を行う上であまり有効でないことが多い．巡回セールスマン問題の 2-opt 近傍を例に考えてみよう．局所最適解において，巡回路の一部が図 5 (a) のようになっているとする．このとき，2-opt 近傍操作により (b) が得られるが，総移動距離は変化しない．逆に，2-opt 近傍操作によって (b) から (a) を得ることもできる．いま，このような部分構造が巡回路に k 箇所あるとすると，総移動距離を変化させることなく 2-opt 近傍操作で互いに行き来できる 2^k 個の局所最適解が存在することになる．このような状況において，これらの局所最適解をすべて探索することは，解空間のある一部分の探索に多くの計算時間を費やすことになり効率的とは言えない．ある程度の探索が行われたところで別の領域に探索を移すことが望ましいが，最近探索した解を禁止解とするだけでは，これを実現することは難しい．

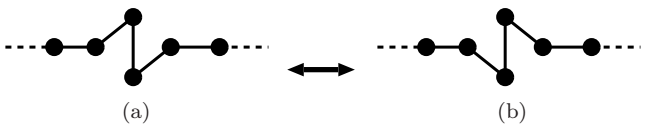

図 5 2-opt 近傍操作によって移動距離が変化しない部分構造の例

そこで，通常は，過去に実行した解の移動の特徴を記憶

しておき，これを用いてタブーリストを定義する方法が用いられる．上の例では，2-opt 近傍操作によって巡回路から削除された辺を「禁止辺」として記憶しておき，その後の探索で，禁止辺を再び解に追加するような近傍操作を禁止する方法が考えられる．つまり，探索済みの解自体への移動を禁止するのではなく，過去に実行した解の変更を元に戻すような近傍操作全般を禁止するのである．この方法によれば，上の例の場合，k 回の反復で 2^k 個の解すべてが禁止解となり，その後，探索が別の領域に移ることが期待できる．なお，タブーリストを定義するために用いられる近傍操作の特徴（この例では禁止辺）を属性と呼ぶ．また，記憶している属性集合が禁止解の集合 T と同じ役割を果たすことから，属性集合自体をタブーリストと呼ぶことが多い．

ところで，解の移動を行うたびにその属性をタブーリストに追加しつづけると，いずれ近傍内の解がすべて禁止解となり，探索を継続することができなくなってしまう．そこで，ある属性をタブーリストに追加したとき，解の移動を一定回数行った後に，その属性をタブーリストから取り除くことにする．属性をタブーリストに保持しておく期間は**タブー期間**（tabu tenure）と呼ばれる．

属性を用いてタブーリストを定義する場合，よい解が近傍内にありながら，それを禁止解として探索対象から除外してしまう可能性がある．そこで，ある近傍解 x' が禁止解であっても，x' が十分によい解であり，x' への移動によってサイクリングが発生しないと考えられる場合には，例外的に x' への移動を許可する場合が多い．禁止解への移動を許可するかどうかの判断基準は**特別選択基準**（aspiration criterion）と呼ばれ,「過去の探索で得られた最良解よりもよい解が得られる」などが代表的である．

一般に探索型のアルゴリズムにおいては，似通った構造をもつ解を集中的に探索すること（探索の**集中化**）と，ときどきは，一時的に解の構造を大きく崩し，未探索の領域に探索を移すこと（探索の**多様化**）の相反する 2 つの動作をバランスよく行うことが重要である．しかし，タブー探索法では，禁止解以外の最良解へ移動することが基本戦略であるため，大幅な改悪を伴う解の移動があまり行われず，集中化のみが強調される傾向にある．探索済みの解を直接タブーリストとする方法は，この傾向が強く現れやすい戦略といえる．一方，属性に基づく方法は，解の移動を強く制限することも可能であり，結果として多様化の実現にも利用できる．

それぞれの近傍操作に対して，何を属性とすればよいかは一概には言えないが，大規模な問題例に対しては，多様化の傾向を高めるために，より多くの近傍操作を禁止する属性が有効となることがある．一般化割当問題のシフト操作を例に考えてみよう．ある 1 つの仕事 j の割当て先を i から i' に変更した場合に，タブーリストに記憶する属性としては，(i) 仕事 j とエージェント i, i' の 3 つ組，(ii) 仕事 j とエージェント i の組，(iii) 仕事 j などが考えられる．それぞれ，タブーリストに保持されている間，(i) 仕事 j の割当先を i' から i に戻すこと，(ii) 仕事 j の割当先を i に戻すこと，(iii) 仕事 j の割当先を変更することを禁止することになり，禁止解の数は，(i) が最も少なく，(iii) が最も多い．現在の解の周辺をくまなく探索するためには (i) もしくは (ii) が適切であるが，大規模な問題例に対しては，解の移動方向をある程度制限し，解空間全体を探索できるように (iii) が適していると考えられる．

タブー期間の値は，集中化と多様化のバランスに影響を与え，探索の性能を大きく左右する最も重要なパラメータの一つであるため，予備実験を行いながら注意深く設定する必要がある．また，問題規模や属性の選び方などによって効果的な値は大きく変わり得る．そこで，3.4 節で紹介するように，探索の状況に応じて適応的にタブー期間を調整する手法も提案されている．

以上，本節では改悪解への移動を許す方法として代表的な 2 つを簡単に紹介したが，その考え方は大きく異なる．アニーリング法は，改悪の可能性をランダム性に委ねたやや受け身的な方法であるのに対し，タブー探索法は，自ら移動方向を制御しながら探索を推し進めていくという意味でより能動的な方法であるといえる．そのため，タブー探索法は，この制御方法をうまく設計できれば非常に高性能なアルゴリズムを実現できる一方で，この設計を誤ると，単純な多スタート局所探索法よりも性能が劣ってしまうこともある．以下では，局所探索法の拡張手法として，とくにタブー探索法の性能を高めるために有用な手法について述べる．

3.2 候補リスト戦略

近傍探索において，現在の解を改善する可能性がない近傍操作を探索の候補から除外したり，改善の見込みのありそうなものに限定したりすることで，探索の効率を高めることができる．これは，局所探索法の高速化に欠かせないテクニックであるが，タブー探索法においては単なる高速化以上の意味をもつ重要な考え方であり，**候補リスト戦略**（candidate list strategy）などとも呼ばれる．前節で述べたように，タブー探索法では集中化が強調されやすい．この傾向は，属性やタブー期間の調整によってある程度解消できるが，候補リスト戦略を積極的に取り入れることも有用である．ここでは一例として，問題構造に依存しない汎用的な手法である**限定選択戦略**（aspiration plus strategy）を紹介する．まず，評価値についての閾値を設定しておき，それよりもよい評価値をもつ近傍解の発見を目標とする．（現在の解の評価値を閾値に設定すれば，改善解を見つけることが目標となるが，この閾値を探索履歴に基づいて適応的に設定することも可能である.）そして，近傍解をラン

ダムな順序で調べていき，閾値を満たす解が見つかった時点から，ある一定個数の近傍解を探索し，それまでに見つかった最良の解に移動するのである．ただし，調べる近傍解の個数が多すぎる，もしくは少なすぎることのないように，あらかじめ個数に関する上下限を設定しておく．そして，上の方法で探索した結果，調べた解の個数が下限に満たない場合には下限に達するまで探索を続け，逆に，探索の途中で上限に達した場合にはその時点で探索を打ち切ることとする．

探索空間が 2.3 節の最後に述べた (c) に分類され，写像 π が単射でない場合，解 x に変更を加えても，その像 $\pi(x)$ が変化するとは限らない．例えばパッキングの例では，順列を変更しても，BL 法を適用して得られる配置は変わらないことがある．このような状況では，多くの場合，現在の解と同じ像をもつ近傍解を探索対象から除外することが効果的と考えられる．これを実現する近傍をうまく設計できない場合でも，候補リスト戦略によって効果的な探索を実現できる可能性がある．

3.3 長期メモリ

タブー探索法の基本的な考え方として，過去の探索履歴を積極的に活用するというものがある．タブーリストもその一つであるが，より長期的な情報は**長期メモリ**（long term memory）と呼ばれ，これを活用する手法が提案されている．ここでは代表的なものとして，頻度メモリを紹介する．これは，ある変数が解の移動によって変更された頻度や，ある変数が特定の値をとっていた頻度を探索の全期間にわたって記録しておき，これを探索の集中化や多様化の制御に活用するものである．例えば，ある一部の変数のみが高い頻度で変更されている場合には，探索が一部の領域に偏っていると判断できる．変更頻度の高い変数の値を変更しにくくすることで探索の多様化を促す必要があり，これは，近傍探索において解の評価値を計算する際，頻度に応じたペナルティを加えることで実現できる．

3.4 パラメータの自動調整

通常，メタヒューリスティクスに基づくアルゴリズムには，探索の挙動を制御するためのパラメータがいくつか含まれる．その中には，個々の問題例のタイプや規模によって適正値が大きく変わるものも存在するため，通常，予備実験が必要不可欠となる．そこで，探索中，パラメータの値を自動調整する手法が提案されている．以下にこの例を 2 つ簡単に紹介する．いずれの手法も，探索状況を考慮した適応的なものであり，パラメータの調整手間を軽減するだけでなく，アルゴリズムの性能を大幅に向上させる効果をもつ．

（1） タブー期間の自動調整

タブー探索法におけるタブーリストの主な役割は，サイクリングを防止し，ある程度の多様化を実現することであるが，タブー期間が短すぎる場合，この効果はあまり期待できない．逆に，タブー期間が長すぎる場合には，近傍探索の対象範囲が過剰に制限され，有望な解が解候補から除かれてしまう．そこで，以下の自動調整方法が考えられる．

- 3.3 節の長期メモリを利用して探索が一部の領域に偏っていないかを随時チェックし，探索が偏っていると判断した場合にはタブー期間を長くする，
- 特別選択基準による例外ルールを設けておき，これが適用されたときには過剰な制限が行われていると判断し，タブー期間を短くする．

（2） ペナルティ重みの自動調整

2.3 節の（1）で述べたペナルティ重み α_i を例にとって説明しよう．ある一定反復ごとに α_i の値を更新するものとして，その更新方法は，前回の更新以降，実行可能解が 1 つでも得られたかどうかに基づいて決定される．もし実行可能解が 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，重みを増やす．このとき，すべての α_i $(i \in I)$ を一律に増やすのではなく，前回の更新以降に得られた解の中で評価値が最良のものを x^* として，x^* による違反度が大きい制約ほど，そのペナルティ重みを大きく増加させる．一方，実行可能解が 1 つでも得られた場合には，ペナルティ重みは十分大きいと判断する．そして，以後の探索で本来の目的関数による影響が相対的に高まるように，ペナルティ重みを小さくするのである．

4. おわりに

局所探索法は，アイデアは非常に単純であるが，アルゴリズム設計における自由度が大きく，様々な工夫を行うことができる奥の深い枠組みである．組合せ最適化問題に対する実用解法として，より高度な枠組みであるメタヒューリスティクスの利用が増えてきてはいるが，実際に解くべき問題に直面したとき，まずは基本的な局所探索法を注意深く実装し，その性能を確かめてみることが重要であろう．そして，その結果に満足できない場合には，より高度な手法を，適宜その実装手間を考慮しながら組み込んでいくことが現実的であると思われる．なお，メタヒューリスティクスの考え方に基づいてアルゴリズムを実装するためのライブラリやフレームワークがいくつか提案されている（EasyLocal++[4] HotFrame[9] iOpt[3] Localizer++[13] など）．これらを活用することで，アルゴリズム開発の手間を大幅に削減できる可能性がある．

局所探索法の拡張手法については，本稿で紹介したもの以外にも数多くのアイデアが提案されている．代表的な手法については，文献 1), 2), 5), 7), 8), 12), 15), 18), 19) などに解説があるので，興味のある読者はこれらの文献を参照いただきたい．とくに文献 19) では，3.2~3.4 節で述べた手法を含め，メタヒューリスティクスの性能を向上させるための方法が詳しく解説されている．一方，局所探索法の基礎に関しては，文献 17) に平易な解説がある．また，

文献 20) には，局所探索法の性質についての考察が，数値実験の結果とともに示されている．タブー探索法の詳細については，文献 6) を参照されたい．この本には，タブー探索法の基本的な考え方だけでなく，高度な手法やタブー探索法の適用例が多数まとめられている．

謝辞．本稿の執筆にあたって貴重なコメントを頂いた今堀慎治，橋本英樹の両氏に感謝します．

（2009 年??月??日受付）

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） E.K. Burke and G. Kendall, eds., *Search Methodologies: Introductory Tutorials in Optimizing and Decision Support Techniques*, Springer, New York, 2005.
3） R. Dorne and C. Voudouris, HSF: The iOpt's framework to easily design meta-heuristic methods, in: M.G.C. Resende and J.P. de Sousa, eds., *Metaheuristics: computer decision-making*, pp. 237–256, Kluwer Academic Publishers, Norwell, MA, 2004.
4） EasyLocal++ Home Page, `http://tabu.diegm.uniud.it/EasyLocal/`
5） F. Glover and G.A. Kochenberger, eds., *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, 2003.
6） F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
7） T.F. Gonzalez, ed., *Handbook of Approximation Algorithms and Metaheuristics*, Chapman & Hall/CRC in the Computer & Information Science Series, 2007.
8） H.H. Hoos and T. Stützle, *Stochastic Local Search—Foundations and Applications*, Morgan Kaufmann/Elsevier, San Francisco, 2005.
9） HotFrame—Heuristic OpTimization FRAMEwork, `http://www1.uni-hamburg.de/IWI/hotframe/hotframe.html`
10） 石亀 篤司，Particle Swarm Optimization—群れでの探索—，計測と制御，47-6，pp. 459–465（2008）．
11） 片山 謙吾，石渕 久生，Memetic アルゴリズム，計測と制御，47，pp. 487–492（2008）．
12） 久保 幹雄，メタヒューリスティクス，室田（編），離散構造とアルゴリズム IV，pp. 171–230，近代科学社，1995.
13） L. Michel and P. van Hentenryck, Localizer++: An open library for local search, Technical Report CS-01-02, Brown University, 2001.
14） 大林 茂，進化計算による多目的最適化と工学設計，計測と制御，47，pp. 480–486（2008）．
15） P.M. Pardalos and M.G.C. Resende, eds., *Handbook of Applied Optimization*, Oxford University Press, New York, 2002.
16） 筒井 茂義，アントコロニー最適化手法，計測と制御，47，pp. 466–472（2008）．
17） 柳浦 睦憲，局所探索法—反復改善に基づく最適化の基本戦略，オペレーションズ・リサーチ，52，pp. 538–542（2007）．
18） 柳浦 睦憲，茨木 俊秀，組合せ最適化問題に対するメタ戦略について，電子情報通信学会論文誌，J83-D-I，pp. 3–25（2000）．
19） 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.
20） M. Yagiura and T. Ibaraki, Local Search, in: P.M. Pardalos and M.G.C. Resende, eds., *Handbook of Applied Optimization*, pp. 104–123, Oxford University Press, New York, 2002.

［著 者 紹 介］

野々部宏司君

2000 年 3 月京都大学大学院情報学研究科数理工学専攻博士後期課程修了．現在，法政大学デザイン工学部准教授．メタヒューリスティクス，スケジューリングアルゴリズムの研究等に従事．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．

柳浦睦憲君

1993 年 3 月京都大学大学院工学研究科数理工学専攻修士課程修了．現在，名古屋大学大学院情報科学研究科准教授．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．