岡山理科大学紀要 第54号 A pp. 27-36(2018)

2次割当問題に対する局所探索法の結合アルゴリズムの特性

岡野 傑士・金原 一歩・片山 謙吾*

岡山理科大学大学院工学研究科情報工学専攻

* 岡山理科大学工学部情報工学科

(2018年10月31日受付、2018年12月6日受理)

1 はじめに

反復局所探索法や memetic アルゴリズムなどのメタ戦略アルゴリズムでは，局所探索法を繰り返し適用することで良好な解の探索を行う [5]．実用上重要な組合せ最適化問題の一つである 2 次割当問題 (Quadratic Assignment Problem, QAP) に対するメタ戦略アルゴリズムでは，一般的に局所探索法として 2-opt 局所探索法 (2-opt Local Search, 2LS) を利用するものが多い．一方，QAP に対する 2LS より広い近傍を巧妙に探索する局所探索法として，グラフ分割問題に対する Kernighan-Lin 法 [1] のアイデアに基づく局所探索法が Murthy らによって提案されている [2]．一般にこのアルゴリズムは可変深度探索法 (Variable Depth Search, VDS) や k-opt 局所探索法 (standard k-opt Local Search, sKLS) と呼ばれる．また，我々は VDS に基づく局所探索アルゴリズムとして，2LS と sKLS を結合するアルゴリズム [6] 及び sKLS の変形アルゴリズムである変形 k-opt 局所探索法 (variant k-opt Local Search, vKLS) [3] を示している．我々の行った実験 [3] [6] によると，これらの VDS に基づく局所探索アルゴリズムは，2LS より良好な解を算出しやすいという傾向を観測している．また，VDS に基づく局所探索アルゴリズムの結合アルゴリズムはほかにも様々考えられる．

そこで本研究では，QAP を対象として，2LS と KLS を結合したアルゴリズムとして 18 種類の性能や特性を調査する．また，比較実験を通して，アルゴリズム改良の手がかりや特性に関する知見等の獲得を試みる．

2 2次割当問題

本研究では，異なる 2 つのコスト行列から，複数存在する割当パターンのなかで，全体のコストが最小となる割当を求める問題を 2 次割当問題の定義とする．

n 個の点があり，各 2 点間の距離が $n \times n$ の行列 $\mathrm{W} = w_{i,j}$（$1 \leqq i, j \leqq n$）で与えられているとする．また，これとは別に n 個の各要素間の相互関係を表す $n \times n$ の行列 $\mathrm{F} = f_{i,j}$（$1 \leqq i, j \leqq n$）が与えられているとする．

2 次割当問題は式（1）で定義することができる．

$$C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_{\pi(i)\pi(j)} \tag{1}$$

- n : 全頂点 (要素) の数
- π : n 個の要素 {1,...,n} からなる順列で表される解
- w_{ij} : 頂点 i と頂点 j の間の距離
- $\pi(i)$: 頂点 i に割当られている要素
- $f_{\pi(i)\pi(j)}$: 頂点 i に割当られている要素から頂点 j に割当てられている要素間のコスト

QAP は式（1）を最小化する解 π を求める問題であり，問題サイズが n の場合, 解空間の大きさは $n!$である．

3 局所探索法

局所探索法 (Local Search, LS) は，ある操作を行うことで生成可能な解の集合である近傍内に改善解が存在しなくなるまで近傍探索を繰り返す方法である．

3.1 節では，一般的に使用される 2-opt 局所探索法について，3.2 節では，VDS に基づく局所探索法である k-opt 局所探索法について説明する．

3.1 2-opt 局所探索法

2-opt 局所探索法 (2-opt Local Search, 2LS) では 2-opt 近傍操作により局所探索を行う．問題サイズ n の問題例に対して，2-opt 近傍操作を用いて得られる近傍解の総数は $n(n-1)/2$ である．近傍操作によって生成された近傍解集合の中に，現在の解より良好な評価値を持つ近傍解は複数存在すると考えられる．その複数の改善解の中でどの解に移動するかの戦略として，最良移動戦略 (Best Improvement, BI) と即時移動戦略 (First Improvement, FI) が知られている [5]．本研究では，最良移動戦略に基づく 2-opt 局所探索法を BI2LS と示し，即時移動戦略に基づく 2-opt 局所探索法を FI2LS と示す．

3.1.1 最良移動戦略に基づく 2-opt 局所探索法

最良移動戦略 (Best Improvement, BI) は近傍解全ての中で最も良好な解への移動を行う．

図 1 に BI2LS の処理手順を疑似コードを用いて示す．このアルゴリズムは近傍探索及び解の移動の処理からなる．π は現在の解，g は最良のゲイン値を保持するものである．まずゲイン値 g を初期化する (Line 2)．近傍探索の処理では，近傍解を生成し，現在の解から変化量 $\delta_{i,j}$ を求める (Line 5)．その結果，現在保持しているゲイン値 g より良好な変化量の解を発見した場合 (Line 6)，その変化量を保持し (Line 7)，その解に移動するために交換する要素 i, j を $swap\_i$, $swap\_j$ として保持する (Line 8)．この近傍探索の処理を $n(n-1)/2$ 個の近傍全てを探索するまで繰り返す (Line 3〜11)．近傍探索の結果，保持しているゲイン値 g が改善していた場合，保持している要素 $swap\_i$, $swap\_j$ の交換により最良の近傍解へ移動する (Line 12)．以上の処理を近傍内に改善解がなくなる (近傍探索の結果，$g \leq 0$ を満たした) 場合，局所解に至ったとして現在の解 π を出力し，探索を終了する．

```
    procedure Best Improvement 2-opt Local Search (π)
    begin
 1      repeat
 2          g := 0;
 3          for i := 1 to n − 1
 4              for j := i + 1 to n
 5                  δ_{i,j} := SwapGain(i, j, π);
 6                  if δ_{i,j} > g then
 7                      g := δ_{i,j};
 8                      swap_i := i, swap_j := j;
 9                  endif
10              endfor
11          endfor
12          if g > 0 then π := SwapMove(swap_i, swap_j, π); endif
13      until g ≤ 0
14      return π;
    end;
```

図 1: QAP に対する最良移動戦略に基づく 2-opt 局所探索法

3.1.2 即時移動戦略に基づく 2-opt 局所探索法

即時移動戦略 (First Improvement, FI) は近傍を生成する過程で，評価値が良好な近傍解を発見した場合その解への移動を行う．

図 2 に FI2LS の処理手順を疑似コードを用いて示す．このアルゴリズムは近傍探索及び解の移動の処理からなる．ゲイン値 g，基準として選択可能な解の配置場所の集合 P_{out} を初期化する (Line 2)．まず，P_{out} からランダムに選ばれた配置場所 i を基準として選択し (Line 4)，P_{out} から選んだ配置場所 i を取り除く (Line 5)．解の要素集合 P_{in} を基準配置場所 i を取り除いた P_{out} で初期化する (Line 6)．これにより，P_{in} は基準 i の対として選択できる配置場所の集合となる．次に基準とした配置場所 i とランダムな配置場所 j の交換操作により現在解 π より改善するか確認する (Line 8〜10)．改善される場合，要素値 $\pi(i), \pi(j)$ の交換を行い (Line 11)，その交換により得られるゲイン値を保持する (Line 12)．この処理を P_{in} が空集合になる (ほかの配置場所全てに伴う近傍探索を行う) まで繰り返す (Line 7〜14)．同様に上述の一連の処理を P_{out} が空集合になるまで繰り返す (Line 3〜15)．以上の処理を改善解がなくなるまで近傍を探索する (Line 1〜16) ことで局所解に達する．

```
    procedure First Improvement 2-opt Local Search (π)
    begin
 1      repeat
 2          g := 0, P_out := {1, ..., n};
 3          repeat
 4              select i ∈ P_out randomly;
 5              P_out := P_out\{i};
 6              P_in := P_out;
 7              repeat
 8                  select j ∈ P_in randomly, P_in := P_in\{j};
 9                  δ_{i,j} := SwapGain(i, j, π);
10                  if δ_{i,j} > 0 then
11                      π := SwapMove(i, j, π);
12                      g := g + δ_{i,j};
13                  endif
14              until P_in = ∅;
15          until P_out = ∅;
16      until g ≤ 0;
17      return π;
    end;
```

図 2: QAP に対する即時移動戦略に基づく 2-opt 局所探索法

3.2 *k*-opt 局所探索法

可変深度探索法 (VDS) では，与えられた解に対して 2-opt 近傍操作を連鎖的に適用することで得られた解集合を改めて大きな近傍 (*k*-opt 近傍) として捉える [2]．このアルゴリズムを我々は標準 *k*-opt 局所探索法 (standard *k*-opt Local Search, sKLS) と呼ぶ．

3.2.1 標準 *k*-opt 局所探索法

図 3 に sKLS の処理手順を擬似コードを用いて示す．このアルゴリズムは外ループと内ループの 2 つの処理からなる．π は現在の解，π_{best} は内ループで見つかった最良解である．ゲイン値 g は内ループ処理前の解 π_{prev} と内ループ中で得られる解 π それぞれの評価値の差 $g = C(\pi_{prev}) - C(\pi)$ であり，文献 [4] による方法で計算する．g_{best} は π_{best} に対応している．

まず，現在の解を π_{prev} として保持し，要素集合 P を準備する (Line 2)．要素集合 P は内ループでの解のサイクリング [5] を防ぐためのものである．内ループでは，現在の解 π に対して，近傍操作によって最良の解 (得られる近傍解のゲイン値 $\delta_{i,j}$ が最大) となる二つの要素 i と j を選び，解のゲイン値を求める (Line 4)．選ばれた要素 i と j の交換により解を移動し (Line 5)，ゲイン値 g を更新する (Line 6)．π_{best} より良好な解であった場合，その解とゲイン値を保持する (Line 7)．それと共に，要素 i と j を P から削除する (Line 8)．これによって 1 度交換された要素は内ループが終了するまで交換の対象にならない．以上の内ループ処理を 2-opt 近傍操作を行うための要素のペアが集合 P に存在しなくなるまで繰り返すことで *k*-opt 近傍を探索する．内ループでは，毎回の繰り返しにおいて，2 つの要素が P から選ばれるため，内ループで生成する近傍解の個数は $n/2$ となる．外ループでは，内ループで連鎖的に生成された近傍解群の内，$g_{best} > 0$ となる最良の近傍解 π_{best} を次回の反復の際の初期解 π に設定する．もし，内ループの近傍操作で初期解を改善する解が得られなかった場合 ($g_{best} \leq 0$)，*k*-opt 近傍内に改善解が存在しなくなったと判断し，探索を終了する．

3.2.2 変形 *k*-opt 局所探索法

sKLS の近傍探索は制限付き最良移動戦略と言える．この探索により，sKLS は大きな近傍の小さな部分を効率的に探索することができる．しかし，sKLS によって得られる局所解は，sKLS に与えられる初期解に依存する．

vKLS では，即時移動戦略の考えに基づくランダム化を取り入れることで，初期解への依存性を減少させ，処理時間の大幅な増加なしにより良い解を得ることができる．

図 4 に QAP に対する vKLS の疑似コードを示す．まず，配置場所 i を Line 4 で P_{out} からランダムに選び，選ばれた配置場所 i を P_{in} 及び P_{out} からそれぞれ削除する (Line 5)．これは内ループでの 2-opt 近傍操作において，同じ解が生成されるサイクリング [5] の現象を抑制する役割を持つ．また，vKLS では要素値 $\pi(i)$ を基準として探索を行うため，要素値 $\pi(i)$ の配置場所を i_b として保持する (Line 5)．$\pi(i_b)$ を基準として，現在解 π に対して，2-opt 近傍操作によって得

```
    procedure standard k-opt Local Search(π)
    begin
1       repeat
2           π_prev := π, g := 0, g_best := 0, P := {1, ..., n};
3           repeat
4               find a pair of element i and j with max_{i,j∈P} δ_{i,j} := SwapGain(i, j, π);
5               π := SwapMove(i, j, π);
6               g = g + δ_{i,j};
7               if g > g_best then g_best := g, π_best := π; endif
8               P := P\{i, j};
9           until P = ∅;
10          if g_best > 0
11              π := π_best;
12          else
13              π := π_prev
14          endif
15      until g_best ≤ 0;
16      return π;
    end;
```

図 3: QAP に対する標準 k-opt 局所探索法

られる近傍解のゲイン値が最大となる，$\pi(i_b)$ と対となる要素値 $\pi(j)$ の配置場所 j を集合 P_{in} から選ぶ．このとき，配置場所 j は Line 8 で P_{in} から削除する．また，交換処理 (Line 8) により $\pi(i_b)$ の配置場所が変化するため，i_b を j に更新する (Line 8)．以上の内ループの主要処理を P_{in} が空集合になるまで繰り返し，k-opt 近傍探索を行う．なお，内ループにて良好な k-opt 近傍解が得られた場合（$g_{best} > 0$），P_{out} をリセットする (Line 11)．P_{out} が空集合となったとき，k-opt 近傍内に改善解が存在しなくなったと判断し，vKLS の探索を終了する (Line 13)．

```
    procedure variant k-opt Local Search(π)
    begin
1       P_out := {0, 1, ..., n − 1};
2       repeat
3           π_prev := π, g := 0, g_best := 0, P_in := {1, ..., n};
4           select i ∈ P_out randomly;
5           P_out := P_out\{i}, P_in := P_in\{i}, i_b := i;
6           repeat
7               find a node j with max_{j∈P_in} δ_{i_b,j} := SwapGain(i_b, j, π);
8               π := SwapMove(i_b, j, π), g := g + δ_{i_b,j}, P_in := P_in\{j}, i_b := j;
9               if g > g_best then π_best := π, g_best := g;
10          until P_in = ∅;
11          if g_best > 0 then P_out := {0, 1, ..., n − 1}, π := π_best
12          else π := π_prev;
13      until P_out = ∅;
14      return π;
    end;
```

図 4: QAP に対する変形 k-opt 局所探索法

4 2LS と KLS を結合するアルゴリズム

2-opt 局所解に対して 2LS より広い近傍を巧妙に探索する KLS を実行することで，高速化及び解の精度の向上が行えると考えられる [6].

以下では，本研究で調査を行う結合アルゴリズム 18 種類について説明する．まず，4.1, 4.2 節で単純結合するアルゴリズについて説明し，4.3, 4.5 節で 2LS の探索情報を利用するアルゴリズム，4.4 節で vKLS の探索の途中に 2LS の探索を行うアルゴリズムの説明を行う．

4.1 2LS と sKLS を単純結合するアルゴリズム

2LS では近傍の探索に $O(n^2)$ の時間がかかるのに対し，sKLS は近傍の探索に $O(n^3)$ の時間を要する．しかし，sKLS は 2LS より大きな近傍を巧妙に探索するため，2-opt 局所解からの探索であっても改善が見込める．2LS の局所解に対して sKLS を実行することで解の精度を維持したまま，探索時間を減少できる [6].

本研究では，2LS として BI2LS 及び FI2LS の 2 種類をそれぞれ導入した BI2LS+sKLS, FI2LS+sKLS を考える．

図 5 に 2LS+sKLS の処理手順を擬似コードを用いて示す．まず，上述した 2-opt 局所探索法である BI2LS もしくは FI2LS を実行し，2-opt 局所解を得る (Line 1)．次に得た 2-opt 局所解に対して上述の sKLS を実行し，standard k-opt 局所解を得る (Line 2)．この解を出力し，探索を終了する．

```
   procedure 2LS + sKLS(π)
   begin
1      π := 2-opt Local Search(π);
2      π := standard k-opt Local Search(π);
3      return π;
   end;
```

図 5: QAP に対する 2LS と sKLS を結合するアルゴリズム

4.2 2LS と vKLS を単純結合するアルゴリズム

上述の 2LS+sKLS と同様に 2-opt 局所解に対して vKLS を実行する方法である．同様に解の精度を維持したまま，探索時間が減少できると考えられる．図 5 の QAP に対する 2LS と sKLS を結合するアルゴリズムの Line2 の sKLS の処理の代わりに vKLS を導入する．

本研究では，2LS として BI2LS 及び FI2LS の 2 種類をそれぞれ導入した BI2LS+vKLS, FI2LS+vKLS を考える．

4.3 2LS の情報を利用して vKLS と結合するアルゴリズム

vKLS では即時移動戦略の考え方を取り入れ，基準を用いて探索を行う．しかし，現在の vKLS はランダムに基準となる要素を決めている．そこで，2LS の探索時に交換した要素を記憶し，その要素の交換回数を基準選択に利用する方法が考えられる．

本研究では，2LS として BI2LS 及び FI2LS の 2 種類をそれぞれ導入し，交換回数の昇順 (Ascending Order) 及び降順 (Descending Order) によって基準選択を行う以下の 4 種類を考える．

- BI2LS の交換回数の昇順によって基準選択を行う AscendingOrderBI2LS+vKLS(AOBI2LS+vKLS)
- BI2LS の交換回数の降順によって基準選択を行う DescendingOrderBI2LS+vKLS(DOBI2LS+vKLS)
- FI2LS の交換回数の昇順によって基準選択を行う AscendingOrderFI2LS+vKLS(AOFI2LS+vKLS)
- FI2LS の交換回数の降順によって基準選択を行う DescendingOrderFI2LS+vKLS(DOFI2LS+vKLS)

4.4 vKLS 探索ごとに 2LS を行うアルゴリズム

vKLS は 2LS や sKLS より大きい近傍を巧妙に探索する．そのため，2-opt 局所解に対して vKLS の近傍解への移動を 1 度行うだけで与えられたもとの 2-opt 局所解を脱出し，改善する可能性がある．そこで，2LS を行い，vKLS の近傍移動によって解が改善するたびに 2LS を再度実行する方法 2LS+vKLS_move が考えられる．

本研究では，2LS として BI2LS 及び FI2LS の 2 種類をそれぞれ導入した 2LS+vKLS_move として，BI2LS+vKLS_move, FI2LS+vKLS_move を考える．

4.5 vKLS 探索ごとに 2LS を行い，その情報を利用するアルゴリズム

上述の 2LS+vKLS_move では vKLS の基準選択をランダムに行っている．しかし，上述の 2LS の情報を利用して vKLS と結合するアルゴリズムと同様に 2LS の探索の情報を vKLS の基準選択に利用することができる．そこで，交換

回数をリセットし 2LS を探索し，その情報を利用する reset2LS+vKLS と 2LS で交換した情報を随時更新し，その情報を利用する overwrite2LS+vKLS の 2 パターンが考えられる．

本研究では，2LS として BI2LS 及び FI2LS の 2 種類をそれぞれ導入し，交換回数の昇順及び降順によって基準選択を行う以下の 8 種類を考える．

- 交換回数をリセットし BI2L を探索し，その情報を昇順で基準選択を行う resetAscendingOrderBI2LS+vKLS(resetAOBI2LS+vKLS)
- 交換回数をリセットし BI2L を探索し，その情報を降順で基準選択を行う resetDescendingOrderBI2LS+vKLS(resetDOBI2LS+vKLS)
- 交換回数をリセットし FI2L を探索し，その情報を昇順で基準選択を行う resetAscendingOrderFI2LS+vKLS(resetAOFI2LS+vKLS)
- 交換回数をリセットし FI2L を探索し，その情報を降順で基準選択を行う resetDescendingOrderFI2LS+vKLS(resetDOFI2LS+vKLS)
- BI2LS で交換した情報を随時更新し，その情報を昇順で基準選択を行う overwriteAscendingOrderBI2LS+vKLS(overwriteAOBI2LS+vKLS)
- BI2LS で交換した情報を随時更新し，その情報を降順で基準選択を行う overwriteDescendingOrderBI2LS+vKLS(overwriteDOBI2LS+vKLS)
- FI2LS で交換した情報を随時更新し，その情報を昇順で基準選択を行う overwriteAscendingOrderFI2LS+vKLS(overwriteAOFI2LS+vKLS)
- FI2LS で交換した情報を随時更新し，その情報を降順で基準選択を行う overwriteDescendingOrderFI2LS+vKLS(overwriteDOFI2LS+vKLS)

5 実験結果

上述した結合アルゴリズムの性能を評価するため，比較実験を行った．比較対象とするアルゴリズムは BI2LS, FI2LS, sKLS, BI2LS+sKLS, FI2LS+sKLS, vKLS, BI2LS+vKLS, FI2LS+vKLS, AOBI2LS+vKLS, DOBI2LS+vKLS, AOFI2LS+vKLS, DOFI2LS+vKLS, BI2LS+vKLS_move, FI2LS+vKLS_move, resetAOBI2LS+vKLS, resetDOBI2LS+vKLS, resetAOFI2LS+vKLS, resetDOFI2LS+vKLS, overwriteAOBI2LS+vKLS, overwriteDOBI2LS+vKLS, overwriteAOFI2LS+vKLS, overwriteDOFI2LS+vKLS の 22 種類である．各アルゴリズムはC++言語によりコード化し，コンパイラはオプション-O3 を付与した g++(Ver5.4.0) を使用した．計算機は CPU: Intel Xeon 3.60GHz，RAM: 8GB 上で実行した．対象とする問題例は QAP のベンチマーク問題例集である QAPLIB より取得した 48 例題とし，特徴により 4 つ (“type1” ～ “type4”) に分類した．

表 1 は各アルゴリズムを各問題例に対して 1000 回実行した各 type 及び全問題例の平均を算出した結果である．“time” の欄は，1000 回実行するのに要した時間 (秒) である．“best [%]” 及び “avg [%]” は，既知の最良解のコストからの最良及び平均誤差（1000 回）である．また，表 2 は各アルゴリズムを各問題例に対して 60sec 実行した各 type1 及び全問題例の平均を算出した結果であり，各アルゴリズムは，問題例データの読み込み時間を含まず，ランダムな初期解の生成時間を含め，各問題例に対して各局所探索法を制限時間内繰り返す．“#LS” の欄は，制限時間内での局所探索法を繰り返した回数である．“best [%]” 及び “avg [%]” は，既知の最良解のコストからの最良及び平均誤差（対応する “#LS”）である．なお，解の評価値の精度は式 (2) によって算出した．式 (2) 中の $C(\pi)$ は得られた解 π の評価値，$C(\pi^*)$ は最適解 (既知の最良解) の評価値を表している．

$$\text{解の評価値の精度 (\%)} = \frac{C(\pi) - C(\pi^*)}{C(\pi^*)} \times 100 \tag{2}$$

表 1 の従来法である BI2LS, FI2LS, sKLS, 及び vKLS (3.1, 3.2 参照) の結果から type1～3 においてより深く探索する KLS のほうが良好な結果を算出しているが，type4 では sKLS より FI2LS のほうが良好な結果を示している．また，2LS は type1, 2, 4 においては BI2LS より FI2LS のほうが良好な解を算出しているが，type3 は FI2LS より BI2LS のほうが良好な解を算出している．このことから，近傍探索の方法によって問題例や type の違いにより，探索性能は異なると考えられる．そのため，BI と FI の両方の探索を取り入れ vKLS は全問題平均してほかの従来法より良好な解を算出している．

結合アルゴリズムである BI2LS と BI2LS+sKLS (4.1 参照) 及び FI2LS と FI2LS+sKLS (4.1 参照) の比較によって，2-opt 局所解に対して sKLS を実行することで平均的に解の改善が見込めることがわかる．sKLS と BI2LS+sKLS 及び

表 1: 各局所探索法を 1000 回実行した平均結果

Instance	BI2LS			FI2LS			sKLS			vKLS			BI2LS+sKLS			FI2LS+sKLS		
Name	best	avg	time	best	avg	time	best	avg	time	best	avg	time	best	avg	time	best	avg	time
type1	2.185	4.922	1.219	2.161	4.761	1.347	1.838	4.083	10.052	1.175	3.353	64.170	1.814	4.234	5.837	1.843	4.165	5.388
type2	0.850	2.386	8.569	0.727	2.248	5.562	0.569	1.970	46.396	0.282	1.438	424.998	0.621	2.068	30.880	0.607	1.994	25.170
type3	0.663	8.234	0.513	0.525	8.521	0.483	0.206	6.831	3.665	0.018	4.319	30.493	0.539	7.085	1.785	0.377	7.289	1.934
type4	0.442	9.646	11.877	0.434	9.154	13.873	0.486	9.437	53.190	0.215	8.399	1,244.263	0.417	9.568	29.341	0.409	9.079	28.796
All_Avg	0.961	6.080	5.530	0.876	5.988	5.022	0.684	5.340	28.534	0.358	4.087	412.959	0.779	5.500	17.378	0.730	5.422	15.439

Instance	BI2LS+vKLS			AOBI2LS+vKLS			DOBI2LS+vKLS			FI2LS+vKLS			AOFI2LS+vKLS			DOFI2LS+vKLS		
Name	best	avg	time	best	avg	time	best	avg	time	best	avg	time	best	avg	time	best	avg	time
type1	1.267	3.498	64.723	1.283	3.480	74.200	1.368	3.530	64.982	1.291	3.453	62.476	1.146	3.457	71.809	1.269	3.506	63.025
type2	0.377	1.724	433.505	0.360	1.683	1,534.011	0.402	1.723	1,142.875	0.379	1.674	404.820	0.334	1.640	1,350.919	0.376	1.680	996.135
type3	0.005	4.755	26.263	0.033	4.730	31.561	0.005	4.790	22.565	0.007	4.788	26.671	0.028	4.727	32.103	0.008	4.745	23.351
type4	0.299	8.777	1,161.860	0.300	8.709	5,832.145	0.243	8.828	4,596.553	0.202	8.196	1,118.525	0.202	8.142	5,188.798	0.245	8.199	4,295.795
All_Avg	0.419	4.410	397.320	0.425	4.373	1,717.526	0.434	4.437	1,333.529	0.405	4.275	379.026	0.370	4.236	1,525.989	0.409	4.275	1,224.877

Instance	BI2LS+vKLS_move			resetAOBI2LS+vKLS			resetDOBI2LS+vKLS			overwriteAOBI2LS+vKLS			overwriteDOBI2LS+vKLS		
Name	best	avg	time	best	avg	time	best	avg	time	best	avg	time	best	avg	time
type1	1.472	3.602	44.021	1.435	3.594	46.099	1.421	3.595	43.809	1.389	3.596	47.392	1.440	3.597	43.984
type2	0.409	1.812	250.825	0.456	1.807	351.458	0.415	1.805	326.786	0.431	1.805	368.349	0.437	1.813	311.903
type3	0.005	4.857	25.210	0.005	4.847	26.094	0.005	4.864	24.273	0.006	4.858	29.451	0.005	4.878	22.053
type4	0.337	8.848	631.638	0.326	8.851	860.637	0.344	8.857	778.369	0.334	8.822	874.778	0.334	8.888	846.593
All_Avg	0.476	4.502	225.581	0.481	4.496	305.384	0.469	4.502	279.575	0.466	4.493	314.830	0.478	4.515	288.523

Instance	FI2LS+vKLS_move			resetAOFI2LS+vKLS			resetDOFI2LS+vKLS			overwriteAOFI2LS+vKLS			overwriteDOFI2LS+vKLS		
Name	best	avg	time	best	avg	time	best	avg	time	best	avg	time	best	avg	time
type1	1.209	3.533	45.207	1.341	3.530	47.083	1.252	3.543	42.787	1.211	3.521	47.577	1.217	3.559	44.221
type2	0.421	1.742	248.146	0.403	1.732	342.147	0.385	1.739	309.253	0.400	1.736	357.359	0.434	1.742	303.474
type3	0.018	4.906	25.457	0.061	4.886	26.267	0.018	4.900	23.890	0.057	4.892	29.934	0.022	4.908	22.413
type4	0.241	8.252	609.876	0.222	8.251	826.473	0.265	8.240	736.751	0.254	8.233	868.122	0.229	8.241	825.418
All_Avg	0.414	4.357	220.504	0.441	4.347	295.592	0.415	4.354	265.122	0.422	4.345	310.185	0.418	4.360	281.626

表 2: 各局所探索法を 60sec 実行した平均結果

Instance	BI2LS			FI2LS			sKLS			vKLS			BI2LS+sKLS			FI2LS+sKLS		
Name	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS
type1	1.531	4.951	359,176	1.372	4.758	432,112	1.331	4.085	30,699	0.955	3.355	28,485	1.267	4.261	84,352	1.224	4.170	87,500
type2	0.564	2.383	22,023	0.500	2.245	34,520	0.508	1.970	3,573	0.339	1.445	967	0.465	2.066	6,634	0.483	1.992	7,880
type3	0.030	8.255	403,110	0.027	8.510	487,922	0.058	6.833	54,001	0.000	4.220	26,550	0.026	7.051	143,847	0.052	7.271	142,638
type4	0.304	9.690	154,907	0.295	9.078	197,663	0.405	9.498	19,666	0.391	8.382	13,131	0.295	9.615	80,413	0.307	9.000	83,789
All_Avg	0.535	6.099	224,074	0.483	5.967	275,299	0.510	5.353	26,720	0.366	4.058	16,122	0.452	5.504	76,597	0.460	5.400	77,927

Instance	BI2LS+vKLS			AOBI2LS+vKLS			DOBI2LS+vKLS			FI2LS+vKLS			AOFI2LS+vKLS			DOFI2LS+vKLS		
Name	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS
type1	1.021	3.503	33,248	0.970	3.502	31,516	1.034	3.547	34,874	0.957	3.464	34,138	0.917	3.394	28,454	0.917	3.394	28,455
type2	0.471	1.725	1,070	0.499	1.674	669	0.504	1.724	784	0.407	1.679	1,138	0.436	1.319	367	0.451	1.320	367
type3	0.000	4.740	29,833	0.008	4.613	27,677	0.000	4.763	29,242	0.000	4.749	30,182	0.000	4.162	22,505	0.000	4.050	20,967
type4	0.456	8.796	19,025	0.598	8.705	16,780	0.589	8.841	17,142	0.281	8.190	19,168	0.701	8.076	7,533	0.681	8.039	7,173
All_Avg	0.434	4.411	19,233	0.465	4.339	17,687	0.474	4.435	18,884	0.365	4.265	19,553	0.454	3.945	13,583	0.455	3.905	13,060

Instance	BI2LS+vKLS_move			resetAOBI2LS+vKLS			resetDOBI2LS+vKLS			overwriteAOBI2LS+vKLS			overwriteDOBI2LS+vKLS		
Name	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS
type1	1.019	3.604	39,606	1.017	3.602	39,097	1.062	3.608	39,704	1.022	3.596	38,931	1.027	3.615	39,998
type2	0.489	1.810	1,564	0.536	1.800	1,404	0.527	1.797	1,442	0.493	1.795	1,341	0.508	1.811	1,504
type3	0.000	4.894	32,599	0.000	4.890	31,875	0.000	4.891	32,499	0.000	4.771	31,144	0.000	4.903	32,521
type4	0.407	8.892	22,769	0.402	8.891	22,224	0.474	8.892	22,495	0.487	8.860	21,144	0.465	8.921	22,628
All_Avg	0.429	4.521	22,166	0.442	4.517	21,696	0.463	4.517	22,061	0.447	4.472	21,207	0.448	4.532	22,169

Instance	FI2LS+vKLS_move			resetAOFI2LS+vKLS			resetDOFI2LS+vKLS			overwriteAOFI2LS+vKLS			overwriteDOFI2LS+vKLS		
Name	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS	best	avg	#LS
type1	1.057	3.537	39,514	0.961	3.536	38,704	0.952	3.548	40,172	1.019	3.534	38,804	0.980	3.549	40,023
type2	0.465	1.743	1,598	0.467	1.734	1,401	0.412	1.742	1,490	0.461	1.735	1,354	0.468	1.744	1,516
type3	0.000	4.893	32,590	0.000	4.893	31,648	0.000	4.894	32,619	0.000	4.903	31,083	0.000	4.888	32,303
type4	0.297	8.256	22,759	0.287	8.254	21,800	0.325	8.238	22,585	0.314	8.206	20,687	0.276	8.220	22,696
All_Avg	0.405	4.355	22,155	0.386	4.352	21,467	0.375	4.353	22,217	0.400	4.344	21,074	0.387	4.349	22,128

FI2LS+sKLS の比較により，2-opt 局所解に対して sKLS を実行することで平均して 2 倍弱の高速化が行えていることもわかる．

また，BI2LS+vKLS (4.2 参照) と AOBI2LS+vKLS, DOBI2LS+vKLS (4.3 参照) 及び FI2LS+vKLS (4.2 参照) と AOFI2LS+vKLS, DOFI2LS+vKLS (4.3 参照) では，各 2-opt 局所解から平均的に改善がなされている．しかし，vKLS と比較して，BI2LS+vKLS と FI2LS+vKLS は平均して若干の高速化がなされているが，2LS の情報を利用して vKLS と結合を行った AOBI2LS+vKLS, DOBI2LS+vKLS, AOFI2LS+vKLS, DOFI2LS+vKLS では全問題平均では 2 倍以上の時間がかかっている．これは vKLS の探索能力が高く，2-opt 局所解から FI の基準を用いる探索の動きにより局所解より脱出しやすいためと考えられる．しかし，高い探索能力ではあるものの，処理にかかる時間が多いため，結果として速度向上がなされなかったと考えられる．交換情報を調べたところ，交換回数が少ない箇所でも交換回数が多い箇所でもない，中間あたりの箇所を基準として用いた場合に改善しやすいことが分かった．このことから交換回数が少ない箇所から探索を行う AO でも，交換回数が多い箇所から探索を行う DO でも，探索が進むと改善が起こるたびに改善の見込みが薄くなった要素を基準として探査を行っていために速度が低下したと考えられる．しかし，ランダムに基準選択を行っている BI2LS+vKLS や FI2LS+vKLS より良好な結果を算出しており，また AOFI2LS+vKLS 及び DOFI2LS+vKLS では type1 以外の問題例において平均して vKLS より良好な結果を算出していることから基準を設けて探索を行うことは良好な解を得る上で一定の効果があると考えられる．

また，2LS を行い，vKLS の近傍移動によって改善するたびに 2LS を再度実行する BI2LS+vKLS_move と FI2LS+vKLS_move (4.4 参照) の結果に着目すると，2LS より改善しているものの同様に 2-opt 局所解に対して実行し，vKLS を局所解に至るまで実行する BI2LS+vKLS 及び FI2LS+vKLS よりも平均して精度が悪いことがわかる．このことから 2LS により探索を行うよりも，より広い探索を行う vKLS を実行し続けるほうが良好な解に至りやすいと考えられる．

また，vKLS 探索ごとに 2LS を行い，その情報を利用するアルゴリズムの 2LS を行うたびに交換回数をリセットする各手法 (4.5 参照) を 2LS の交換回数を随時更新する各手法 (4.5 参照) とそれぞれ比較すると，平均結果より昇順に利用する AO では，overwrite のほうが，降順に利用する DO では，reset のほうが若干ではあるが良好な結果を示している．しかし，表 2 では overwrite のほうが若干良好な結果を示した．よって，特徴的な傾向を観測するには至らなかった．

以上の傾向は表 2 でも同様に確認されたが，AOFI2LS+vKLS 及び DOFI2LS+vKLS は type2〜4 において vKLS より平均的に良好な解を算出した．本論文で調査した方法では，vKLS の性能を全問題例に対して改善することはできなかったが，問題例によっては改善する方法を確認した．また，FI2LS の局所解に対して vKLS を実行した場合には，特に type4 の問題例に対して改善を確認した．このことから探索の序盤はよりランダムに探索を行い，探索が進んだ場合，より大きな近傍のもとで広く探索することで良好な解の探索が行えると考えられる．

6 むすび

反復局所探索法や memetic アルゴリズムなどのメタ戦略アルゴリズムでは，局所探索法を繰り返し適用する．本論文では，代表的な組合せ最適化問題である 2 次割当問題を対象として，2-opt 局所探索法と可変深度探索法を結合したアルゴリズムとして 18 種類の性能や特性を調査した．また，比較実験の結果，探索の序盤はよりランダムに探索を行い，探索が進んだ場合，より大きな近傍のもとで広く探索することで良好な解の探索が行えると考えられる．

参考文献

[1] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, Vol. 49, pp. 291–307, 1970.

[2] K.A. Murthy, Y. Li, and P.M. Pardalos. A local search algorithm for the quadratic assignment problem. *Informatica*, Vol. 3, pp. 524–538, 1992.

[3] T. Okano, K. Katayama, K. Kanahara, and N. Nishihara. A local search based on variant variable depth search for the quadratic assignment problem. In *Proc. of IEEE 7th Global Conference on Consumer Electronics (GCCE 2018)*, pp. 32–35, 2018.

[4] É. D. Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, Vol. 3, No. 2, pp. 87–105, 1995.

[5] 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.

[6] 岡野傑士, 片山謙吾, 金原一歩, 西原典孝. 2 次割当問題に対する 2-opt 局所探索法と *k*-opt 局所探索法の結合. 平成 30 年度（第 69 回）電気・情報関連学会中国支部連合大会講演論文集 R18-18-05, 2018.

Performance Research of Hybrid Local Search Methods for Quadratic Assignment Problem

Takeshi OKANO, Kazuho KANAHARA and Kengo KATAYAMA*
Graduate School of Engineering,
Okayama University of Science,
**Department of Information and Computer Engineering,*
Faculty of Engineering,
1-1 Ridai-cho, Okayama, 700-0005, Japan

(Received October 31, 2018; accepted December 6, 2018)

Metaheuristic algorithms such as iterated local search and memetic algorithm search high-quality solutions by repeatedly applying local search and perturbation operators. In the quadratic assignment problem (QAP), a typical iterative improvement local search is called 2-opt local search (2LS) that is based on 2-opt neighborhood. A refined local search heuristic algorithm to the QAP is proposed by Murthy, Li and Pardalos. Their local search algorithm is based on the idea of variable depth search (VDS) shown by Kernighan and Lin. Since the VDS is generally called (variable) *k*-opt local search (KLS), we call their algorithm, standard KLS (sKLS) for the QAP. As a variant of the standard VDS (i.e., sKLS) shown by Murthy et al. for the QAP, we consider a sophisticated local search algorithm, called variant *k*-opt local search (vKLS).

In this paper, we investigate search properties of 18 types of hybrid local search algorithms that combine 2LS and KLS for the QAP.

Keywords: combinatorial optimization; quadratic assignment problem; local search; variable depth search.