岡山理科大学紀要 第45号 A pp. 63-70(2009)

2次割当問題に対する反復 k-opt 局所探索法の Kick 法の検討

北田 雅享・片山 謙吾*・南原 英生*・成久 洋之*
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

(2009年9月30日受付、2009年11月5日受理)

1. まえがき

2次割当問題(Quadratic Assignment Problem, QAP)[1]は，n 個の場所に n 個の施設を設置した場合，施設同士の移動コストを最小化する問題である．それぞれの場所同士を結ぶ距離行列 d，施設間を移動するためのフロー行列 f が予め与えられる．QAP の解表現を π とすると，π は要素数 n の順列で表すことができる．また，2地点間の距離は d_{ij}，2施設間のフローは $f_{\pi(i)\pi(j)}$ で表すことができる．以上より，QAP の目的関数は式(1)のように表すことができる．

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

QAP は，この目的関数を最小化する問題である．

QAP は，施設配置問題や LSI 配置問題，スケジューリング問題などの実用上重要な組合せ最適化問題として知られており，NP-困難[2]である．NP-困難な問題は，多項式時間で厳密解を算出するアルゴリズムは存在しないであろうと考えられている．従って，最適性の保証はなくとも，ある程度の精度の高い解(近似解)が求まれば，十分に満足のいく場合が多い．そこで，欲張り法や局所探索法などの近似解法，遺伝的アルゴリズムや反復局所探索法などのメタ戦略アルゴリズム[3]が多く用いられている．QAP に対して，現在までに様々なメタ戦略アルゴリズムの適用報告がされており，それらは良好な結果を示している[1) 7)]．メタ戦略アルゴリズムは基本構成として局所探索法を使用している．つまり局所探索法の改善が，そのままメタ戦略アルゴリズムの改善につながると考えられる．

最近我々は QAP に対して，可変深度探索法(Variable Depth Search, VDS)[4]のアイデアに基づく k-opt 局所探索法(k-opt Local Search, KLS)[5) 6)]を提案し，その有効性を示した．さらに，QAP に対する地形解析の結果[5]，QAP の多くの問題例の探索空間は大域最適解に対して，構造化されていることを確認している．

そこで我々は，KLS をメタ戦略アルゴリズムの一つである反復局所探索法(Iterated Local Search, ILS)の枠組みに導入した，反復 k-opt 局所探索法(Iterated k-opt Local Search, IKLS)[5) 6)]を提案している．ILS は単純な枠組みながら，過去の探索で得られた局所解に対してランダムな変形を加えたものを次回探索の初期解として，局所探索を反復する手法である．ILS の主要オペレータとして，局所探索法と Kick がある．Kick は局所解を脱出する操作であり，局所探索によって得られた局所解を効率的に脱出させることができれば，ILS の性能を向上させることが可能であると考えられる．そこで，本論文では QAP の地形解析の結果を考慮した Kick を示し，Kick に用いるパラメータの検証を行う．

2. QAP に対する反復 k-opt 局所探索法

本章では，QAP に対する反復 k-opt 局所探索法(IKLS)について述べる．まず，IKLS の概要を示した後，IKLS の主要な構成要素について具体的に説明する．

2.1 IKLS の基本アルゴリズム

IKLS は，過去の探索で得られた局所解に対してランダムな変形を加えたものを次回探索の初期解として，局所探索を反復する手法である．IKLS の流れを図1に示す．IKLS は探索の準備を行う初期プロセス

```
   procedure Iterated k-opt LocalSearch
   begin
 1    set a minimum α;
 2    generate a random solution π;
 3    π := k-opt LocalSearch(π); π_best := π;
 4    π_temp := π, π_best := π;
 5    repeat
 6       if restart = true
 7          then π := restart(π_best), set the minimum α;
 8          else π := Kick(π_temp, α); endif
 9       π := k-opt LocalSearch(π);
10       if C(π) < C(π_temp)
11          then π_temp := π, set the minimum α;
12          else increase α; endif
13       if C(π_temp) < C(π_best)
14          then π_best := π_temp endif
15    until terminate = true;
16    return π_best;
   end;
```

図1　QAP に対する IKLS

(Line1-Line4) と反復して解を探索するメインループ (Line5-Line15) の 2 つの部分から構成されている．初期プロセスでは，まず初期解としてランダム解を生成し π とする (Line2)．そして，π に対して KLS を適用し，現在の最良解 π_{best}，探索で発見された最良解 π_{temp} とする (Line3, Line4)．次にメインループでは，π_{temp} に対して Kick を適用し (Line8) π とし，KLS によって π を改善する (Line9)．最後に，π_{temp} および π_{best} を更新する (Line10-14)．これらを終了条件を満たすまで繰り返す (Line15)．探索終了後は，π_{best} を返して処理を終了する．以下に，本 IKLS の主要な構成要素である Local Search，Kick 法について詳しく述べる．

2.2 Local Search

IKLS で使用する Local Search として，我々が既に提案している QAP に対する k-opt 局所探索法 (KLS) を使用する．以下で KLS の概要について示す．

2.2.1 QAP に対する k-opt 局所探索法

本章では，QAP に対して我々が提案する k-opt 局所探索法 (KLS) について述べる．

KLS は，可変深度探索 (VDS) に基づいている．VDS は，与えられた解に対して比較的小さな近傍操作を連鎖的に適用することによって到達可能な解の集合を，新たに大きな近傍と捉える近傍探索のアイデアである．KLS は，各反復において現在の解に対して，2-opt 近傍操作を連鎖的に適用することで得られる解集合を，改めて大きな近傍と捉えることで，局所探索を行うアルゴリズムである．また，2-opt 近傍操作は図 2 に示すように，現在の解の 2 つの要素を交換して近傍解を生成する方法である．

KLS の擬似コードを図 3 に示す．KLS は外ループ (Line2-12) と内ループ (Line4-10) の 2 つのループから構成されている．まず，擬似コードで用いられている変数を説明する．π は現在保持している解である．また，π_{best} は KLS で見つかった最良解，π_{in} は内ループの探索で見つかった暫定的な最良解である．同様に，g は探索によって得られた解と探索を開始する前の解の評価値の改善幅 (ゲイン値) を表す．KLS の内ループでは，現在の解に対して任意の要素 i, j を交換し，そのゲイン値を求める (Line5-7)．そして，ゲイン値が改善ならば現在の解 π を π_{in} に，ゲイン値 g を g_{in} に保存する (Line8)．一度交換した要素は，内ループの処理が終わるまで交換禁止とする (Line9)．交換できる要素がなくなれば，内ループは終了する (Line10)．次に外ループでは，内ループで発見された最良解 π_{in} のゲイン値と現在保持している最良解 π_{best} のゲイン

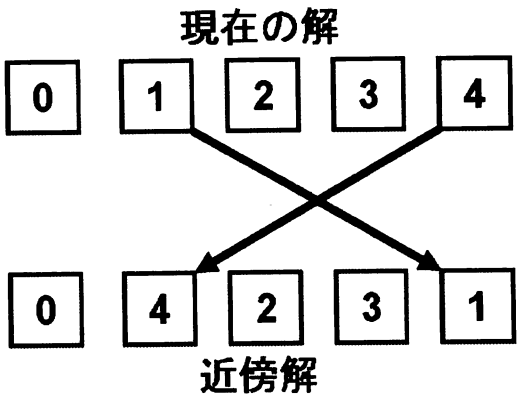

図 2　2-opt 近傍操作

```
   procedure k-opt LocalSearch(π)
   begin
1     π_best := π, g_best := ∞;
2     repeat
3        π := π_best, g := 0, g_in := ∞, P := {1,...,n};
4        repeat
5           find a pair of element i and j
                with min_{i,j∈P_in} δ_{i,j} := SwapGain(i, j, π);
6           π := SwapMove(i, j, π);
7           g = g + δ_{i,j};
8           if g < g_in then g_in := g, π_in := π; endif
9           P := P\{i, j};
10       until P = ∅;
11       if g_in < g_best then g_best := g_in, π_best := π_in; endif
12    until g_in > g_best ;
13    return π_best;
   end;
```

図 3　QAP に対する KLS

値を比較して，π_{in} のほうが良好であれば，π_{best} を更新する．この処理を解の改善がなくなるまで繰り返す．

2.3 Kick 法

本 IKLS で使用する Kick 法について述べる．Kick 法は，LS によって得られた局所解を脱出させる操作である．本研究で使用する Kick 法は，可変近傍探索（Variable Neighborhood Search，VNS）の考え方を取り入れた，2-opt 近傍に基づく Kick 法である．VNS は，最良解が更新されている場合は Kick によって小さな変化を，最良解が更新されない場合には，Kick によって徐々に大きな変化を加えていくという方法である．図 4 に本 IKLS で使用する Kick の擬似コードを示す．

kick は変数の初期化プロセス（Line1-3）と，解に変化を加えるメインループ（Line4-9）から構成されている．初期化プロセスでは，まず変化を加える解を π とする（Line1）．次に，メインループで使用する順列 P を初期化し（Line2），Kick によって加える変化の大きさ（Kick_size）を決定する（Line3）．Line3 で使用している α は VNS のアイデアを取り入れたもので，IKLS によって解が改善されている間は小さい値，改善されなければ徐々に大きな値となる．Kick_size は，

$$n \times \text{Kick\_min} \times \alpha \leq \text{Kick\_size} \leq n \times \text{Kick\_max} \times \alpha$$

の範囲からランダムに決定される．また，Kick_min と Kick_max の関係は

```
   procedure Kick (π_temp, α)
   begin
1     π := π_temp;
2     P := {1, ..., n};
3     Kick_size :=
            GetRandom(n × Kick_min × α ≤ Kick_size ≤ n × Kick_max × α);
4        repeat
5           find a pair of element i and j with i ≠ j ∈ P ;
6           π := SwapMove(i, j, π);
7           P := P\{i, j};
8           Kick_size := Kick_size - 1;
9        until Kick_size = 0;
10    return π;
   end;
```

図 4　Kick 法の擬似コード

$$0.1 \leq \text{Kick\_min} \leq \text{Kick\_max} \leq 0.9$$

である．メインループでは，順列 P の中から，2 つの要素を選択し，解 π を入れ替える（Line5-6）．入れ替える要素の重複を避けるため，入れ替えた要素は P から削除する（Line7）．Line8 で Kick_size を 1 減らす．これらの処理を Kick_size が 0 になるまで繰り返す．

2.4 restart 処理

本 IKLS では，探索の多様性を維持するために restart 処理を行っている．ある程度探索が進むと，現在発見している最良解の近くにそれ以上良好な解が発見できない場合がある．restart 処理は，そのような場合に Kick よりも大きな変化を解に加え，新たな領域から探索を始めるための処理である．本 IKLS では，問題サイズ × 80 % 解構造を変化させるように設定している．

3. IKLS の Kick 法の性能評価実験

本章では 2.3 節で示した Kick_size のランダム選択のパラメータ，Kick_min および Kick_max について検討を行う．また，処理時間の経過に対する最良解の変化の様子を調べ，探索の進行速度について考察する．

3.1 実験詳細

実験は Kick_min および Kick_max を 0.1，0.2，0.3，0.4，0.5，0.6，0.7，0.8，0.9 の 9 段階，計 45 パターンのパラメータ値の設定を用いて，有効なパラメータ設定を検討する．また，IKLS で使用する α は初期値を 0.1 とし，0.1 ずつ 10 段階で変化するように設定した．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[8] より文献[1] で使用されている els19，chr25a，bur26a，nug30，kra30a，ste36a，tai60a，tai80a，tai100a，sko100a，tai60b，tai80b，tai100b，tai150b，tho150，tai256c の 16 例題を用いる．また QAPLIB では，tai256c が最大の問題サイズとなっている．実験は，各問題サイズごとに計算打ち切り時間を設定し 10 回試行する．また，最適解（既知の最良解）が出力された場合もその試行は打ち切りとする．表 1 に各問題例の最適解と打ち切り処理時間（s）を示す．打ち切り処理時間は文献[1] に基づいて定めている．性能評価は，得られた解の評価値の平均精度を用いる．解の評価値の精度は，式 (2) によって求めている．

$$\text{解の評価値の精度 (\%)} = \frac{\text{改善解の評価値} - \text{最適解 (既知の最良解) の評価値}}{\text{最適解 (既知の最良解) の評価値}} \times 100 \tag{2}$$

IKLS は C 言語によってコード化し，使用コンパイラは，gcc で最適化オプション-O3 を付加した．全ての実験は，Hewlett-Packard 社の計算機 HP xw4300 Workstation CPU：Pentium4 3.4GHz，4GB RAM，OS：Fedora Core 5 上で行う．

表1　使用問題例と打ち切り処理時間

instance		
name	opt	time(s)
els19	17212548	5
chr25a	3796	15
bur26a	5426670	20
nug30	6124	20
kra30a	88900	20
ste36a	9526	30
tai60a	7205962	90
tai80a	13511780	180
tai100a	21052466	300
sko100a	152002	300
tai60b	608215054	90
tai80b	818415043	180
tai100b	1185996137	300
tai150b	498896643	600
tho150	8133484	600
tai256c	44759294	1200

3.2 実験結果

結果を表2〜表10に示す．それぞれの表の instance 欄の name には問題例名を示し，問題例名に含まれる数字は問題サイズを表している．なお，sum は各結果の合計値で小さくなるほど良い結果である．Kick_max の下には Kick_max で使用した値を示した．結果は全て%で表記している．<0.01 は 0 ではないが，0.01 よりも小さい値であるということである．また太字で示した値は，結果が良好な10パターンである．全パターンにおいて，各問題例に対する解の評価値の平均精度は同様の傾向が得られた．小さいサイズの問題例では，ほぼ全てのパターンで最適解が出力されている．sum に大きな影響を与える問題例として tai60a，tai80a，tai100a が挙げられる．これらの問題例は精度が 1.18〜1.94 とパラメータの組み合わせで大きな差が出ている．また，ste36a，tai60b，tai80b，tai100b の精度は最適解到達回数に応じて大きく異なる結果が得られた．以上より，sum で良好な結果を示したパラメータ設定は，全ての問題例に対して比較的良好な精度を示したものであるということがいえる．

また，sum の結果に探索の進行状況が関係するかを調べるために，sum が最良となるパラメータ値設定（Kick_min = 0.9，Kick_max = 0.7）と sum が最悪となるパラメータ値設定（Kick_min = 0.4，Kick_max = 0.9）の tai60a の処理時間に対する最良解の変化を調べ，以下に考察する．図5，図6に tai60a の処理時間に対する最良解の変化を示す．どちらの図も横軸は処理時間（time），縦軸は解の評価値（cost）である．また，実線は各試行に対する最良解の変化，破線は最適解を表している．どちらも探索の序盤で急速に最良解が更新されている．しかし，図5では探索終盤にも多く最良解が更新されているのに対し，図6では中盤をすぎたあたりから最良解の更新が少なくなっている傾向が観測できた．

表2　Kick_min=0.1 の結果

instance	Kick_max								
name	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
els19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
chr25a	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bur26a	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nug30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
kra30a	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ste36a	0.34	0.17	0.15	0.08	0.01	0.00	0.01	0.01	0.00
tai60a	1.52	1.55	1.51	1.46	1.27	1.39	1.31	1.44	1.42
tai80a	1.50	1.40	1.55	1.31	1.33	1.30	1.40	1.41	1.41
tai100a	1.41	1.30	1.41	1.38	1.31	1.37	1.36	1.36	1.39
sko100a	0.28	0.25	0.23	0.16	0.12	0.10	0.10	0.12	0.13
tai60b	0.06	< 0.01	< 0.01	0.00	0.00	< 0.01	0.00	0.00	< 0.01
tai80b	0.63	0.23	0.42	0.39	0.28	0.16	0.21	0.06	0.07
tai100b	0.44	0.14	0.48	0.12	0.12	0.53	0.05	0.07	0.09
tai150b	0.93	0.98	0.86	0.82	0.83	0.72	0.76	0.74	0.57
tho150	0.67	0.50	0.40	0.35	0.34	0.30	0.28	0.32	0.28
tai256c	0.12	0.10	0.10	0.10	0.10	0.11	0.11	0.12	0.12
sum	7.98	6.64	7.11	6.17	5.72	5.98	5.58	5.64	**5.46**

4. むすび

本論文では QAP に対する反復 k-opt 局所探索法（IKLS）を提案し，IKLS の主要な処理である Kick 法のパラメータ値について検討を行った．その結果，Kick_min は 0.3〜0.5，Kick_max は 0.5〜0.7 程度に設定した場合に比較的良好な結果を示した．また，処理時間に対する最良解の更新状況については，良好な結果を示すパラメータ値は探索終盤でも最良解が更新されている傾向が観測できた．

表 3　Kick_min=0.2 の結果

instance	Kick_max							
name	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
els19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
chr25a	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bur26a	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nug30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
kra30a	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ste36a	0.08	0.07	0.01	0.00	0.01	0.00	0.00	0.00
tai60a	1.47	1.44	1.35	1.41	1.28	1.37	1.38	1.36
tai80a	1.47	1.43	1.41	1.38	1.34	1.29	1.36	1.42
tai100a	1.45	1.45	1.32	1.32	1.34	1.35	1.42	1.33
sko100a	0.19	0.19	0.16	0.16	0.11	0.12	0.13	0.12
tai60b	< 0.01	0.00	0.00	0.00	< 0.01	0.00	0.00	0.00
tai80b	0.24	0.29	0.48	0.19	0.17	0.33	0.07	0.18
tai100b	0.29	0.20	0.13	0.27	0.11	0.09	0.07	0.13
tai150b	0.94	0.89	0.80	0.69	0.60	0.65	0.70	0.50
tho150	0.49	0.35	0.52	0.39	0.35	0.29	0.26	0.28
tai256c	0.09	0.10	0.10	0.09	0.10	0.11	0.13	0.12
sum	6.70	6.41	6.28	5.89	**5.40**	5.60	5.53	**5.45**

表 4　Kick_min=0.3 の結果

instance	Kick_max						
name	0.3	0.4	0.5	0.6	0.7	0.8	0.9
els19	0.00	0.00	0.00	0.00	0.00	0.00	0.00
chr25a	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bur26a	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nug30	0.00	0.00	0.00	0.00	0.00	0.00	0.00
kra30a	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ste36a	0.03	0.01	0.00	0.00	0.01	0.00	0.00
tai60a	1.40	1.30	1.35	1.23	1.33	1.39	1.46
tai80a	1.47	1.28	1.39	1.38	1.37	1.42	1.34
tai100a	1.38	1.30	1.25	1.35	1.29	1.36	1.42
sko100a	0.19	0.16	0.09	0.10	0.11	0.11	0.14
tai60b	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tai80b	0.34	0.36	0.19	0.27	0.06	0.25	0.14
tai100b	0.54	0.28	0.31	0.26	0.08	0.09	0.14
tai150b	0.71	0.58	0.76	0.66	0.69	0.68	0.77
tho150	0.40	0.34	0.29	0.29	0.29	0.23	0.31
tai256c	0.09	0.12	0.09	0.12	0.14	0.13	0.14
sum	6.55	5.72	5.71	5.66	**5.37**	5.65	5.85

表 5　Kick_min=0.4 の結果

instance	Kick_max					
name	0.4	0.5	0.6	0.7	0.8	0.9
els19	0.00	0.00	0.00	0.00	0.00	0.00
chr25a	0.00	0.00	0.00	0.00	0.00	0.00
bur26a	0.00	0.00	0.00	0.00	0.00	0.00
nug30	0.00	0.00	0.00	0.00	0.00	0.00
kra30a	0.00	0.00	0.00	0.00	0.00	0.00
ste36a	0.01	0.00	0.00	0.00	0.00	0.00
tai60a	1.29	1.18	1.26	1.25	1.33	1.48
tai80a	1.34	1.27	1.31	1.28	1.44	1.41
tai100a	1.32	1.31	1.30	1.34	1.40	1.37
sko100a	0.16	0.12	0.12	0.10	0.11	0.15
tai60b	0.16	0.00	0.00	0.00	0.00	0.00
tai80b	0.42	0.06	0.23	0.04	0.18	0.20
tai100b	0.37	0.40	0.09	0.09	0.11	0.07
tai150b	0.84	0.60	0.63	0.58	0.54	0.56
tho150	0.37	0.36	0.30	0.27	0.29	0.28
tai256c	0.12	0.14	0.14	0.15	0.16	0.17
sum	6.39	**5.43**	**5.38**	**5.11**	5.56	5.68

表 6　Kick_min=0.5 の結果

instance	Kick_max				
name	0.5	0.6	0.7	0.8	0.9
els19	0.00	0.00	0.00	0.00	0.00
chr25a	0.00	0.00	0.00	0.00	0.00
bur26a	0.00	0.00	0.00	0.00	0.00
nug30	0.00	0.00	0.00	0.00	0.00
kra30a	0.00	0.00	0.00	0.00	0.00
ste36a	0.00	0.00	0.00	0.00	0.00
tai60a	1.34	1.26	1.33	1.36	1.39
tai80a	1.22	1.28	1.39	1.53	1.42
tai100a	1.23	1.31	1.35	1.33	1.49
sko100a	0.10	0.11	0.11	0.11	0.13
tai60b	< 0.01	< 0.01	< 0.01	0.00	< 0.01
tai80b	0.17	0.34	0.24	0.08	0.16
tai100b	0.23	0.13	0.08	0.08	0.07
tai150b	0.73	0.47	0.54	0.59	0.57
tho150	0.28	0.27	0.28	0.31	0.26
tai256c	0.16	0.15	0.17	0.17	0.18
sum	**5.45**	**5.33**	**5.48**	5.57	5.67

表 7　Kick_min=0.6 の結果

instance	Kick_max			
name	0.6	0.7	0.8	0.9
els19	0.00	0.00	0.00	0.00
chr25a	0.00	0.00	0.00	0.00
bur26a	0.00	0.00	0.00	0.00
nug30	0.00	0.00	0.00	0.00
kra30a	0.00	0.00	0.00	0.00
ste36a	0.00	0.00	0.00	0.03
tai60a	1.37	1.31	1.53	1.60
tai80a	1.35	1.45	1.67	1.71
tai100a	1.47	1.69	1.67	1.80
sko100a	0.10	0.13	0.15	0.15
tai60b	< 0.01	0.00	< 0.01	< 0.01
tai80b	0.32	0.14	0.08	0.13
tai100b	0.12	0.10	0.12	0.12
tai150b	0.56	0.44	0.67	0.63
tho150	0.31	0.28	0.29	0.28
tai256c	0.17	0.18	0.19	0.20
sum	5.77	5.72	6.37	6.65

表 8　Kick_min=0.7 の結果

instance	Kick_max		
name	0.7	0.8	0.9
els19	0.00	0.00	0.00
chr25a	0.00	0.00	0.00
bur26a	0.00	0.00	0.00
nug30	0.00	0.00	0.00
kra30a	0.00	0.00	0.00
ste36a	< 0.01	0.01	< 0.01
tai60a	1.54	1.79	1.86
tai80a	1.81	1.82	1.84
tai100a	1.80	1.83	1.74
sko100a	0.15	0.16	0.16
tai60b	< 0.01	< 0.01	< 0.01
tai80b	0.23	0.17	0.07
tai100b	0.12	0.11	0.12
tai150b	0.66	0.63	0.71
tho150	0.30	0.33	0.36
tai256c	0.18	0.20	0.19
sum	6.78	7.03	7.04

表 9　Kick_min=0.8 の結果

instance	Kick_max	
name	0.8	0.9
els19	0.00	0.00
chr25a	0.00	0.18
bur26a	0.00	0.00
nug30	0.00	0.00
kra30a	0.00	0.00
ste36a	0.00	0.05
tai60a	1.89	1.92
tai80a	1.85	1.87
tai100a	1.87	1.85
sko100a	0.20	0.22
tai60b	0.01	0.02
tai80b	0.15	0.08
tai100b	0.13	0.23
tai150b	0.76	0.87
tho150	0.39	0.38
tai256c	0.18	0.20
sum	7.42	7.87

表 10　Kick_min=0.9 の結果

instance	Kick_max
name	0.9
els19	0.00
chr25a	0.37
bur26a	0.00
nug30	0.00
kra30a	0.00
ste36a	0.13
tai60a	1.94
tai80a	1.90
tai100a	1.79
sko100a	0.24
tai60b	0.02
tai80b	0.16
tai100b	0.22
tai150b	0.76
tho150	0.38
tai256c	0.20
sum	8.11

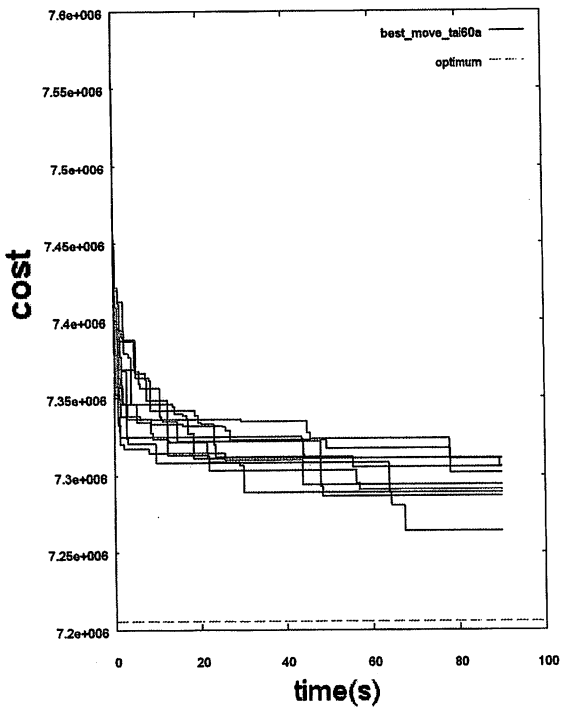

図 5 tai60a の最良解の変化（Kick_min= 0.4, Kick_max=0.7）

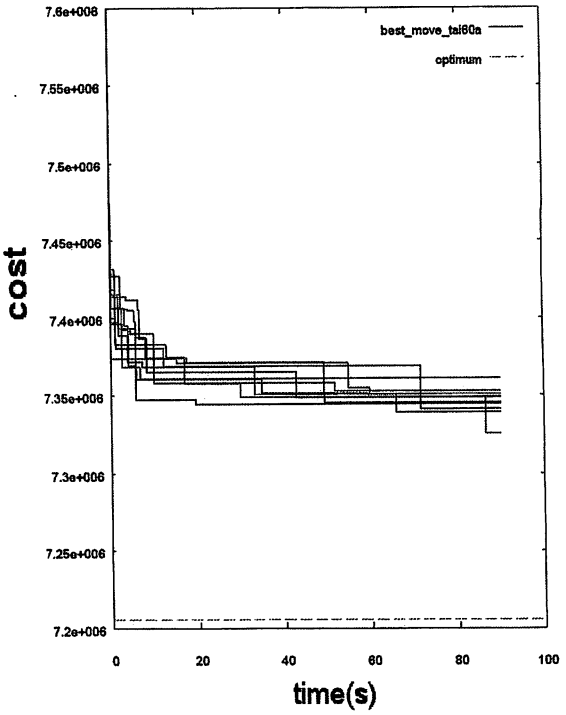

図 6 tai60a の最良解の変化（Kick_min= 0.9, Kick_max=0.9）

以下では，今後の課題・検討事項について記述する．

(1) 今回検討を行わなかった α の増減について検討する必要がある．

(2) 各問題例によって，Kick_min および Kick_max の最適な値が異なるので，適応的にそれらを調整できるようにする必要がある．

参考文献

1) Merz, P., Freisleben, B., "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem," IEEE Transactions on Evolutionary Computation, Vol. 4, No. 4, pp.337–352, 2000.

2) M. Garey, and D. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, New York, 1979.

3) 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.

4) S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," Operations Research, vol.21, pp.498–516, 1973.

5) 北田雅享，片山謙吾，成久洋之，南原英生，"2 次割当問題に対する反復 k-opt 局所探索法の性能," 2009 総合大会講演論文集, DS-1-4, pp.S-27–28, 愛媛, Mar. 17–20, 2009.

6) 北田雅享，片山謙吾，成久洋之，南原英生，"2 次割当問題に対する反復 k-opt 局所探索法の検討," 平成 20 年度電気・情報関連学会中国支部第 59 回連合大会講演論文集, p.391, 鳥取, Oct. 25, 2008.

7) T.Stützle, "Iterated local search for the quadratic assignment problem," Technical Report AIDA-99-03, FG Intellektik, FB Informatik, TU Darmstadt, Darmstadt, Germany, 1999.

8) http://qaplib.uwaterloo.ca/inst.html

Analysis of Kick of Iterated k-opt Local Search for Quadratic Assignment Problem

Masataka KITADA, Kengo KATAYAMA*, Hideo MINAMIHARA* and Hiroyuki NARIHISA*

Graduate School of Engineering,
**Department of Information and Computer Engineering, Faculty of Engineering,*
Okayama University of Science.
1-1 Ridai-cho, Kita-ku, Okayama 700-0005, Japan.

(Received September 30, 2009; accepted November 5, 2009)

Many metaheuristic algorithms are based on local search. One of the most effective local search algorithms is known to be variable depth search(VDS) for combinatorial optimization problems. In this paper we present an effective iterated k-opt local search (IKLS), for the quadratic assignment problem (QAP). To show the effectiveness of kick of IKLS, we compare the performance of kick parameters. The results indicated that suitable parameter values depend on the QAP instances.

Keywords: combinatorial optimization; quadratic assignment problem; iterated k-opt local search; kick.