岡山理科大学紀要 第38号 A pp. 123-132(2002)

バイナリー2次計画問題に対する知識を導入した高速 k-opt 局所探索法の効果

河本　敬子・片山　謙吾*・成久　洋之*

岡山理科大学大学院工学研究科博士課程システム科学専攻

*岡山理科大学工学部情報工学科

(2002年11月1日　受理)

1　序論

組合せ最適化問題の多くはNP-困難であり，厳密な最適解を求めることは極めて困難であることが知られている．しかし，現実には，最適性の保証はなくとも，ある程度精度の高い解が求まれば，十分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 [20]．近似解法の基本戦略の一つである局所探索法（Local Search, LS）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，様々な組合せ最適化問題に対してある程度精度の高い解を比較的短時間に求められることが知られている．また，各組合せ最適化問題には代表的なLSが存在し，これらのLSに対してこれまでに多くの改良アルゴリズムが提案されてきた．

本論文では，バイナリー2次計画問題（Binary Quadratic Programming Problem, BQP）に対して知られているLSの中でも，最も有効な k-opt 局所探索法（k-opt 法）に焦点をあてる．この k-opt 法は，1960年代後期から70年代前期において，TSP及びGPPに対してLinとKernighanにより提案された局所探索法（LK法，KL法）のアイデア（可変深度探索, variable depth search）に基づき，近年BQPに対してMerzとFreislebenによって提案された [16]．

BQPに対する k-opt 法は，良好な k-opt 近傍解の探索を行う内ループと，その近傍解を評価する外ループによって構成され，最終的に良質な近似解を算出する．しかしながら，近傍解探索のプロセスは必ずしも効率的ではないため，内ループを強制的に終了させ近傍解探索を打切るためのパラメータが準備されている．文献 [16] における k-opt 法の内ループの繰返し回数は，最良解が見つかった時点から数えて m 回の内ループの繰返しにおいても最良解が更新されなければ内ループを打切るというパラメータ m によって決定されている．さらに，文献 [13] では文献 [16] の変形版として，外ループの繰返し回数が増すごとにパラメータ m の値を変動的に減少させている．しかしながら，より最適なパラメータ設定は対象とする各問題例の性質を取り入れた方法などが有効であると考えられる．

本論文の目的は，文献 [13] で提案した方法を拡張した高速 k-opt 法の提案である．greedyな初期解に対して従来法を試行し，最良解が得られたときの各外ループにおける内ループの繰返し回数について分析する．この分析から得られた知識をもとに内ループの繰返し回数の設定を行い，大幅な探索時間の効率化を行う高速 k-opt 法の設計を試みる．高速 k-opt 法の性能を確認するために，BQPの問題例に対してテストを行い，従来法との比較を行う．その結果，我々の高速 k-opt 法は従来法に比べ，解質の低下をできるだけ抑え，計算時間を短縮可能であることを示す．

2　バイナリー2次計画問題

バイナリー2次計画問題（BQP）とは，$n \times n$ の対称行列 $Q = (q_{ij})$ が与えられたとき，次の目的関数

$$\begin{aligned} f(x) &= x^t Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j, \\ & x_i \in \{0,1\}, \forall i = 1, \ldots, n, \end{aligned} \tag{1}$$

を最大化する解 x を求める問題である [5]．BQPは，NP-困難であり，数多くの応用例を有している．例として，CAD問題 [14]，マシンスケジューリング問題 [1]，capital budgeting and financial analysis 問題 [15],

traffic message management 問題 [6]，分子構造問題 [19] などがある．更に，BQP は様々な組合せ最適化問題と同等であることが知られている．そのような問題として，最大カット問題，最大クリーク問題，最大頂点パッキング問題，最小頂点被覆問題，最大独立集合問題などがある [3, 5, 8, 17, 18]．また，2 次ナップザック問題（QKP）や 2 次割当問題（QAP）を BQP として解いた研究 [2] もあり，BQP に対する解法の重要性を示す一例 [10] として考えられている．

3 BQP に対する k-opt 局所探索法

3.1 k-opt 局所探索法

最初に，局所探索法で利用される解の表現について記述する．BQP における解 x は，長さ n で構成される各ビット $x_i(i=1,\ldots,n)$ に 0 もしくは 1 を持つ，0-1 ビット表現である．また，0 と 1 のどのような組合せであっても実行可能解となるため，BQP の解空間 X のサイズは 2^n となる．

最も簡単な近傍は，解 x に対して，i 番目のビットを反転することによる 1-opt 近傍である．このとき，一度に生成可能な 1-opt 近傍解の数は n に等しく，1-opt 近傍を有する局所探索法は 1-opt 局所探索法（1-opt 法）と呼ばれる．1-opt 法は，各繰返しにおいて，現在解の目的関数 $f(x)$ と n 個からなる 1-opt 近傍解の $f(x')$ とのゲイン $g(=f(x')-f(x))$ が最大となる近傍解を生成しながら探索を進め，その最大となるゲインが 0 もしくは負となるときを終了条件とするものである．

k-opt 近傍 $N_{k-\mathrm{opt}}(x)=\{x'\in X|d_H(x,x')\leq k\}$ (d_H は二つの解のハミング距離を指す) は，k 個のビットを一度に反転することにより到達可能な解の集合と定義され，k の数に応じて指数的 ($|N_{k-\mathrm{opt}}|=n^k$) となる．

BQP に対する k-opt 近傍は，連続的な 1-opt 近傍操作によって到達可能な解集合とされる．具体的には，各ビットに対して 1 回のみ行うことを保証したビットの反転を連続的に行う．これは，最大のゲインを持つビットを連続的に反転することにより n 個の解集合を生成し，その n 個の解集合の中の最良解を k-opt 近傍解とするものである．よって，BQP に対する k-opt 近傍は，各繰返しにおいて 1〜k 個の可変的なビットの反転によって構成され，一般的に k の数を常に固定した完全 k-opt 近傍よりも効率的な探索を可能にする．

現在解 x における j 番目のビット ($x_j\in\{0,1\}$) を反転し得られる近傍解評価のためのゲイン値 g_j は，次式で求められ，$O(n)$ 時間を要する．なお，$\overline{x}_j=1-x_j$ である．

$$g_j=q_{jj}(\overline{x}_j-x_j)+2\sum_{i=1,i\neq j}^{n}q_{ij}x_i(\overline{x}_j-x_j) \tag{2}$$

式 (2) を用いて n 個の 1-opt 近傍解のゲイン値を計算するには，$O(n^2)$ 時間を要する．Merz らは，n 個の 1-opt 近傍解のすべてのゲイン値を線形時間で可能とするゲイン更新法を採用しており [16]，我々の k-opt 法でもこの更新法を取り入れた．

ここで，図 1 に示す BQP に対する k-opt 法の擬似コードをもとにアルゴリズムを説明する．各変数は，探索中に算出された最良解 x_{best} を保持するための x_{prev}，最良解 x_{best} のゲイン値 G_{max}，現在解 x のゲイン値 G，近傍解へ移動するために現在解に対して反転操作を実施し得るビット番号を保持する C である．内ループ中の処理は，C のすべてのビットを反転させることにより，生成された n 個の解の中から最大のゲイン G_{max} を有する x_{best} を k-opt 近傍解とする．次に，外ループでは，このような探索操作を良好な近傍解が生成できなくなるまで繰返す．

3.2 k-opt 局所探索法のパラメータ m について

BQP に対する k-opt 局所探索法の基本アルゴリズムは，内ループにおいて n 個の解の生成を巧妙に行い，その解集合の中から最良の解を選ぶ．しかしながら，探索時間の効率化を考えると，常に n 個の解を調べる必要はなく，実際的にはそれよりも比較的少ない個数の解を調べることで，(保証は無いが) 最良解を選び出

```
   procedure k-Opt-Local-Search(x, g)
   begin
 1   repeat
 2     x_prev := x, G_max := 0, G := 0, C := {1,...,n};
 3     repeat
 4       find j with g_j = max_{i∈C} g_i;
 5       G := G + g_j;
 6       x_j := 1 − x_j, C := C\{j};
 7       update gains g_i for all i;
 8       if G > G_max then G_max := G, x_best := x;
 9     until C = ∅;
10     if G_max > 0 then x := x_best
11     else x := x_prev;
12   until G_max ≤ 0;
13   return x;
   end;
```

図 1　k-opt 局所探索法

すことが可能になる．Merz [16] らは，内ループの終了条件（$C = \emptyset$ になるまで）を内ループで保持される x_{best} が見つかった時点から数えて m 回の内ループの繰返しにおいても最良解が更新されなければ，内ループを強制的に終了する方法に修正し，探索時間の短縮を試みた．これにより，パラメータ m を $m \ll n$ のように設定する場合は，基本アルゴリズムよりも探索時間の効率化が可能であることを述べている．Merz らは，この m を 100 とすることを推奨した．一方，k-opt 法にランダム性を与えた，片山らのランダム k-opt 法 [9, 11] でも同様のパラメータが存在し，$m = 50$ を推奨している．k-opt 法は，m の設定値によって，最終的に得られる解や探索に要する探索時間に影響を及し，無駄な探索時間を省くことができると考えられる．

3.3　k-opt 法における探索傾向の分析

本論文では，greedy に生成された初期解からスタートする k-opt 法に関して探索傾向を分析する．以下に，greedy な初期解を用いる理由を述べ，分析結果を示す．

近年，最適化問題に対する効率的な解法として，メタ戦略の研究が盛んである．遺伝的アルゴリズム (Genetic Algorithm, GA)，アニーリング法，タブー探索法やそれらの変形版などのアルゴリズムはメタ戦略と呼ばれ [20]，基本解法である LS などと自然界に存在するアイデアを結合させることによって，比較的短時間に高品質な近似解を算出可能である．

この一例として，GA に LS を組み込んだ遺伝的局所探索法 (Genetic Local Search, GLS) が挙げられる．GLS では，突然変異や交叉操作によって生成された解は，ランダム解よりも比較的良好な解となり，greedy な初期解と類似する [12] ものと考えることができる．LS はこの良好な解に対して探索を行うこととなる．よって，greedy な初期解の利用は，将来的にメタ戦略へ k-opt 法を導入する場合においてより実際的な効果を観測できるものと期待できる．なお，greedy 解法 [16] は，ランダム性を有しており，計算機上で与えるランダムの種などに応じて異なる近似解を生成可能である．greedy 解法により平均的に得られる解質は，以下に記述する問題例に対して，多くの場合に，既知の最良解から約 1%〜5%程度と比較的良い．

表 1 各外ループにおいて G_{max} を得たときの IteInLoop の平均

instances	*dens* (Q)	outer-loop									
		1	2	3	4	5	6	7	8	9	10
glov500-1	0.1	26	12	5	2	1					
glov500-3	0.5	30	11	5	2	1					
glov500-5	1.0	28	15	7	4	2	1				
kb-g01	0.1	41	20	13	6	2	2				
kb-g05	0.5	47	26	15	7	3	1	1			
kb-g10	1.0	62	32	20	11	7	4	1			
beas500-1	0.1	27	13	6	2	1					
beas500-5	0.1	24	10	7	3	1					
beas500-10	0.1	20	10	4	1						
beas1000-1	0.1	46	14	8	4	2	1				
beas1000-5	0.1	58	27	14	7	2	1				
beas1000-10	0.1	48	23	10	6	2					
beas2500-1	0.1	122	58	35	19	11	8	4	2	2	1
beas2500-5	0.1	118	56	34	18	13	6	4	2	1	1
beas2500-10	0.1	110	49	28	19	10	6	3	2	1	1

表 1 は，greedy な初期解に対する基本アルゴリズムの各外ループにおいて G_{max} を得たときの内ループの繰返し回数を示す．ここで，内ループの繰返し回数を IteInLoop と表記する．この結果は，100 回の実行による平均である．本実験では，BQP の良く知られたテスト問題 [4] から 500〜2500 変数までの大規模な 15 個の問題例を扱う．これらの問題例は，Glover らによる 500 変数の 3 個の問題例 (glov500) [7]，Kochenberger らによる 1000 変数の 3 個の問題例 (kb-g) [2]，Beasley による 500, 1000, 2500 変数の問題例 (beas500, beas1000, beas2500) [5] で，それぞれは 3 個の問題例を含む．全問題例は，ORLIB [4] より取得可能であり，これらは，問題サイズ，各問題例の行列 Q の density（$dens(Q)$:行列 Q において 0 以外の数値が含まれる割合）などの違いがある．

Merz らのパラメータ m を 100 とした方法での各外ループにおける IteInLoop は，G_{max} が更新されてから最大 100 回を必要とする．しかしながら，表 1 によると，各外ループにおいて G_{max} が更新されてから 100 回も内ループを行う必要はないと考えられる．例えば，glov500-1 の問題例における 1 回目の外ループの IteInLoop は 26，2 回目では 12 となっており，3 回目以降でも外ループの回数が増すごとに G_{max} を得るときの IteInLoop は，徐々に減少している．よって，G_{max} が更新されてから 100 回も内ループを追加する必要がないことが確認できる．また，問題例の特徴の違いに関係なく，他の問題例においても同様の傾向がみられる．但し，beas2500 の問題例においては 1 回目の外ループが 122，118，110 となっており，G_{max} を得るときの IteInLoop にもよるが，G_{max} が更新されてから 100 回の内ループの繰返しを行う必要があるかもしれない．

表 1 を更に分析すると，glov500 の各問題例に対する 1 回目の外ループにおいて G_{max} を得たときの IteInLoop の平均値は，26, 30, 28 となっており，いずれも，問題サイズ n の約 1/20 である．また，2 回目以降の各外ループにおいて G_{max} を得たときの IteInLoop の平均値は，前回の外ループにおける IteInLoop の約 1/2 であることが観測できる．また，他の各問題例群においても同様の探索傾向がみられた．このように，外ループの回数が増すにしたがって，G_{max} が得られたときの IteInLoop は問題例の特徴の違いに関係なく，平均的に減少する傾向が確認された．

これらの近傍探索の傾向をもとに，各外ループにおいて G_{max} が得られたときの IteInLoop 付近で内ループを打切れば，解質をできるだけ落とさずに探索時間を大幅に短縮できると考えられる．

4　近傍探索における知識を導入した高速 k-opt 局所探索法

文献 [13] の方法は，k-opt 近傍探索処理で生成される連続的な近傍解の候補をできるだけ有望なものだけに絞るための探索打切り判定によって，簡潔で効率的な処理を行っている．具体的には，内ループの繰返し回数をパラメータの設定値と外ループの繰返し回数によって変動させることで k-opt 法自体の探索時間を短縮可能としている．しかしながら，近傍解の候補をできるだけ絞るという意味では効率的とは言えない．パラメータの設定値と外ループの繰返し回数以外の有益な情報によって内ループの繰返し回数を制御できれば，より効率的な k-opt 法になると考えられる．本論文では，最も有効な情報として考えられる k-opt 法の近傍解探索のプロセスにおいて暫定的な解の変動傾向を考慮に入れた高速 k-opt 法を提案する．我々の提案法では，3.3 の近傍解探索の分析結果から得られた知識を利用した「内ループの基本繰返し回数（各外ループにおける最小限の内ループの繰返し回数）」を使用する．アルゴリズムの特徴としては，まず「内ループの基本繰返し回数」によって最小限の近傍解探索を行う．その後，過去のある時点から現時点までの探索状況から将来の探索状況を予測し，更に内ループを追加するか否かを判定する．この内ループの追加は，最小限の内ループの繰返し回数による近傍解探索の不足を防ぐ役割を担っている．これによって近傍解の生成の打切りを行い，解質をできるだけ落とさずに探索時間を大幅に短縮可能とする．

```
    procedure Fast-k-Opt-Local-Search(x, g)
    begin
 1    set L_basic
 2    repeat
 3      x_prev := x, G_max := 0, G := 0, C := {1,...,n}, L_pres := 0, L_gmax := 0, G_prev := 0;
 4      update L_basic;
 5      repeat
 6        L_pres + +;
 7        find j with g_j = max_{i∈C} g_i;
 8        G := G + g_j;
 9        x_j := 1 − x_j, C := C\{j};
10        update gains g_i for all i;
11        if (G > G_max) then G_max := G, x_best := x, L_gmax := L_pres;
12        if ((L_pres ≥ L_basic) and (G_prev ≤ G)) then break;
13        G_prev = G;
14      until C = ∅;
15      if (L_gmax > L_pres − p) then L_basic := L_pres + p, goto 5;
16      if (G_max > 0) then x := x_best, else x := x_prev;
17    until G_max ≤ 0;
18    return x;
    end;
```

図 2　近傍探索における知識を導入した高速 k-opt 法

ここで，図 2 に示す擬似コードをもとに，近傍探索における知識を導入した高速 k-opt 法のアルゴリズムを説明する．各変数は，内ループの基本繰返し回数 L_{basic}, 現在の内ループの繰返し回数 L_{pres}, G_{max} が得られたときの内ループの回数 L_{gmax}, 前回の内ループでのゲイン値 G_{prev}, ある区間における G の増減を調べるための変数 p である．L_{basic} の値は，次のように決定する．1 回目の外ループでの L_{basic} は問題サイズ n の 1/20，2 回目以降の外ループでは前回の外ループで用いた IteInLoop の 1/2 とした．

このアルゴリズムでは，まず内ループの基本繰返し回数として L_{basic} 回の内ループを繰返す。その後，12 行目の処理で現在の G と前回の内ループで得られた G_{prev} の比較を行い，G の増大があれば増大がみられ

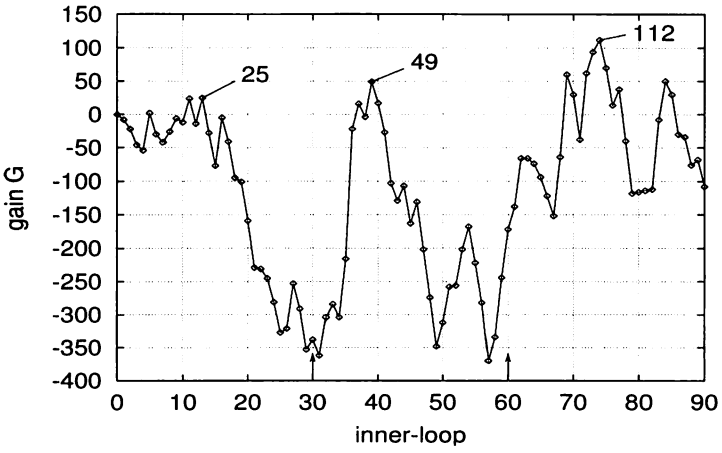

図 3 G_{max} が更新される可能性

なくなるまで内ループを繰返す．この処理によって，無駄な内ループの繰返しを大きく省くことができる．しかしながら，この時点から何回かの内ループを繰返すことによって G_{max} が更新される場合がある．そこで，G_{max} が更新される可能性を予測するために，現時点までの内ループにおける G の増減を着目する．

15 行目では，区間 $[L_{pres}-p, L_{pres}]$ における p 個の G の増減を調べる．その区間内で G_{max} が更新されていれば，今後，G_{max} が更新される可能性があると考え，現時点 L_{pres} から更に p 回の内ループを繰返す．ここでは，区間を決定するための変数 p の値を 30 とした．

次に，図 3 を用いて高速 k-opt 法の近傍解探索の様子を説明する．図 3 は，提案法を beas2500-8 の問題例に対して試行したときのゲインの増減を示す．ここでの試行回数は 1 回で，横軸は 5 回目の外ループにおける内ループの繰返し回数の一部（0〜90）を示し，縦軸は現在解のゲイン値 G を表す．

この例では，4 回目の外ループ終了時での L_{pres} は 60 であった．よって，図 3 に示した 5 回目の外ループにおける内ループの基本繰返し回数 L_{basic} は，その 1/2 の 30 となっている．矢印は，$L_{basic}30$，その現時点 $L_{pres}30$ から p 回追加した 60 を示す．

図 3 を例に挙げると，L_{basic} が 30 の時点までの G の値は，IteInLoop=13 の G の値が 25 であるピーク時から徐々に減少していることがわかる．したがって，内ループは 30 回の時点で打切っても良いと考えられるが，更に内ループを 9 回繰返した IteInLoop=39 のとき，G_{max} が 49 に更新されることが観測できる．これらの様子から，G が減少傾向にあっても G_{max} が更新される可能性を最大限に生かすために，現時点と現時点から p 回前の区間 $[L_{pres}-p, L_{pres}]$ の内ループにおいて G_{max} が得られていれば，現時点から更に p 回の内ループを繰返す処理を追加している．

5 数値実験

上で提案した近傍探索における知識を導入した高速 k-opt 法の効果を検討するために，ベンチマーク問題を用いて，m を 100 に固定した Merz らの k-opt 法との比較を行う．高速 k-opt 法の p の値は 10, 20, 30, 40, 50, 60, 70 について検討した．実験は，表 1 で扱った問題例を含め全 45 問題例を使用する．すべての実験は，NEC PC98-NX Mate (Pentium Ⅲ, 650MHz) 上で実行され，プログラム言語は C 言語である．

表 2 は，greedy な初期解に対する従来法と提案法の実験結果である．各欄は，既知の最良解値 (best-known)，m の値を従来通り 100 に固定，提案法に対して，1000 回の試行で得られた最良解 (best) と既知の最良解からのその解質 (%)，平均値 (avg) とその解質 (%)，全計算時間 t（秒）を示す．更に，高速 k-opt 法の有効性を検討するために，従来法に対する avg の解質差 (loss)，全計算時間の短縮率 (r-t) を示す．

例えば，beas500-6 の問題例における従来法による avg の解質は 0.489%，計算時間は 15 秒であるのに対

し，提案法では，avg の解質が 0.510%，計算時間は 2 秒となっており，0.021%の解質の低下で計算時間を 86.667%短縮できている．また，45 個の全問題例のうち 38 個は従来法と同等の最良解が得られた．さらに，kb-g03 の問題例の best においては従来法で得ることのできなかった既知の最良解を得たことが分かる．

表 3 は，45 個の全問題例と kb-g の 10 個の問題例に対する各手法の実験結果の比較である．各欄は，基本アルゴリズム（$C = \emptyset$），従来法（fixed $m = 100$），文献 [13] の外ループの繰返し回数が増すごとにパラメータ m の値を変動的に減少させる方法（m 減少率は 40%），提案法（高速 k-opt 法 $p = 30$）における平均の解質 avg(%)，全計算時間 time(s)，基本アルゴリズムと従来法に対する各手法の avg の解質差 loss(%)，全計算時間の短縮率 r-t(%) を示す．

結果から，全問題例における高速 k-opt 法の解質は，p の値が 10 のとき 0.571%，p の値が 20 のとき 0.531%，以後，0.514%, 0.504%, 0.498%, 0.496%, 0.495%となっており，p の値が増す毎に解の質が向上していることが分かる．そして，計算時間においては 21.622 秒, 26.089 秒, 31.822 秒, 40.733 秒, 44.800 秒, 54.756 秒, 61.556 秒となっており，p の値が大きくなるに従って，計算時間がかかることが分かる．

また，全問題例における基本アルゴリズムによる avg の平均の解質は 0.491%，計算時間は 615.489 秒であるのに対し，m 減少率 40%の場合，解質が 0.520%，計算時間は 34.067 秒となっており，0.029%の質の低下で計算時間を 94.465%短縮できている．これに対して，高速 k-opt 法 ($p = 30$) の場合では，解質が 0.514%，計算時間は 31.822 秒となっており，0.023%の質の低下で計算時間を 94.830%短縮できている．よって，高速 k-opt 法は m 減少率 40%の手法に比べ，短時間で高品質な解を得たことが分かる．さらに，高速 k-opt 法は従来法に比べ，0.023%の解の質の低下で計算時間を 50.570%短縮可能であることが分かる．また，kb-g の問題例における基本アルゴリズムと従来法に対する avg の解質差，全計算時間では，高速 k-opt 法は解質，計算時間ともに m 減少率 40%よりも優れており，解質は約 2/3 の改善を行うことができた．

6 結論

本論文では，バイナリー2次計画問題（BQP）に対する LS の中でも最も有効な k-opt 局所探索法（k-opt 法）に焦点をあて，k-opt 法の近傍解探索の分析を行った．greedy な初期解に対する k-opt 法の近傍解探索の分析を行った結果，最良解が得られたときの外ループにおける内ループの繰返し回数は問題サイズに依存し，外ループの繰返し回数が増すにしたがって一定の割合で減少することが分った．この分析から得られた知識を導入することで内ループの繰返し回数を決定すれば，より効率的な k-opt 法となると考えられる．そこで，各外ループにおける内ループの繰返し回数は，問題サイズと近傍解の生成状況に応じて決定する方法とした．このような内ループの繰返し回数によって近傍解生成の打切りを行うことで，k-opt 法の近傍解探索のプロセスにおける暫定的な解の変動傾向を考慮した大幅な探索時間の効率化手法である高速 k-opt 法を提案した．

その探索性能を確認するために，よく知られている BQP の問題例に対して greedy な初期解を用いた提案法を試行した結果，既存の効率化手法に比べ，平均的に解質をできるだけ維持しながら，探索時間を大幅に短縮できることを示した．よって，実用的な観点からメタ戦略に導入する場合にも，高速 k-opt 法は非常に有効であると考えられる．

今後は，高速 k-opt 法を GLS などの実際のメタ戦略に適用した場合の性能評価を検討予定である．

参考文献

[1] B. Alidaee, G.A. Kochenberger, A. Ahmadian, "0-1 quadratic programming approach foe optimal solution of two scheduling approach for the optimal solution of two scheduling problems", International Journal of Systems Science, vol.25, no.2, pp.401–408, 1994.

[2] M.M. Amini and B. Alidaee, and G.A. Kochenberger, "A scatter search approach to unconstrained quadratic binary programs", New Ideas in Optimization (eds, D. Corne, M. Dorigo and F. Glover), McGraw-Hill, pp.317–329, 1999.

[3] F. Barahona, M. Jünger, G. Reinelt, "Experiments in quadratic 0-1 programming", Mathematical Programming, vol.44, pp.127–137, 1989.

表 2 greedy な初期解に対する実験結果

instances	*dens* (Q)	best-known	fixed $m = 100$ best	(%)	avg	(%)	t(s)	高速 k-opt 法 best	(%)	avg	(%)	t(s)	loss(%)	r-t(%)
glov500-1	0.1	61194	61194	0.000	61059.066	0.221	6	**61194**	**0.000**	61055.125	0.227	6	0.006	0.000
glov500-2	0.25	100161	100161	0.000	99963.203	0.197	15	**100161**	**0.000**	99956.484	0.204	8	0.007	46.667
glov500-3	0.5	138035	138035	0.000	137688.484	0.251	37	**138035**	**0.000**	137679.578	0.257	11	0.006	70.270
glov500-4	0.75	172771	172771	0.000	172016.031	0.437	41	**172771**	**0.000**	172002.578	0.445	23	0.008	43.902
glov500-5	1.0	190507	190507	0.000	190017.625	0.257	60	**190507**	**0.000**	190002.734	0.265	26	0.008	56.667
kb-g01	0.1	131456	131441	0.011	130696.523	0.578	21	**131441**	**0.011**	130668.859	0.599	8	0.021	61.905
kb-g02	0.2	172788	172788	0.000	171795.234	0.575	59	**172788**	**0.000**	171741.203	0.606	23	0.031	61.017
kb-g03	0.3	192565	192534	0.016	189560.813	1.560	73	**192565**	**0.000**	189426.219	1.630	29	0.070	60.274
kb-g04	0.4	215679	215374	0.141	213336.891	1.086	108	**215374**	**0.141**	213220.156	1.140	46	0.054	57.407
kb-g05	0.5	242367	242367	0.000	240838.234	0.631	97	**242367**	**0.000**	240779.922	0.655	52	0.024	46.392
kb-g06	0.6	243293	243132	0.066	240444.313	1.171	145	242983	0.127	240283.859	1.237	71	0.066	51.034
kb-g07	0.7	253590	253561	0.011	249826.625	1.484	129	**253561**	**0.011**	249653.156	1.552	92	0.068	28.682
kb-g08	0.8	264268	264077	0.072	262086.281	0.826	150	**264077**	**0.072**	261981.453	0.865	71	0.039	52.667
kb-g09	0.9	262658	262624	0.013	259387.031	1.245	182	**262624**	**0.013**	259253.500	1.296	94	0.051	48.352
kb-g10	1.0	274375	274030	0.126	271032.438	1.218	215	**273964**	**0.150**	270892.281	1.269	98	0.051	54.419
beas500-1	0.1	116586	116586	0.000	115890.391	0.597	12	**116586**	**0.000**	115863.344	0.620	3	0.023	75.000
beas500-2	0.1	128339	128339	0.000	127963.680	0.292	9	**128339**	**0.000**	127958.906	0.296	4	0.004	55.556
beas500-3	0.1	130812	130812	0.000	130538.961	0.209	10	**130812**	**0.000**	130530.820	0.215	4	0.006	60.000
beas500-4	0.1	130097	130077	0.015	129672.227	0.327	10	**130077**	**0.015**	129654.563	0.340	5	0.013	50.000
beas500-5	0.1	125487	125487	0.000	124946.625	0.431	10	**125487**	**0.000**	124916.539	0.455	5	0.024	50.000
beas500-6	0.1	121772	121772	0.000	121175.992	0.489	15	**121772**	**0.000**	121150.859	0.510	2	0.021	86.667
beas500-7	0.1	122201	122201	0.000	121376.336	0.675	10	**122201**	**0.000**	121339.984	0.705	4	0.030	60.000
beas500-8	0.1	123559	123530	0.023	122926.336	0.512	10	**123530**	**0.023**	122900.727	0.533	4	0.021	60.000
beas500-9	0.1	120798	120798	0.000	120173.617	0.517	11	**120798**	**0.000**	120142.063	0.543	5	0.026	54.545
beas500-10	0.1	130619	130619	0.000	130259.305	0.275	14	**130619**	**0.000**	130248.070	0.284	5	0.009	64.286
beas1000-1	0.1	371438	371438	0.000	370718.688	0.194	25	**371438**	**0.000**	370691.969	0.201	15	0.007	40.000
beas1000-2	0.1	354932	354631	0.085	353504.250	0.402	33	**354631**	**0.085**	353443.531	0.419	8	0.017	75.758
beas1000-3	0.1	371236	371236	0.000	369663.281	0.424	31	**371236**	**0.000**	369581.438	0.446	6	0.022	80.645
beas1000-4	0.1	370675	370675	0.000	369705.844	0.261	21	**370675**	**0.000**	369653.344	0.276	7	0.015	66.667
beas1000-5	0.1	352760	352730	0.009	351584.000	0.333	29	**352730**	**0.009**	351526.969	0.350	12	0.017	58.621
beas1000-6	0.1	359629	359629	0.000	358467.375	0.323	37	**359629**	**0.000**	358401.281	0.341	10	0.018	72.973
beas1000-7	0.1	371193	371193	0.000	369812.781	0.372	35	**371193**	**0.000**	369741.344	0.391	10	0.019	71.429
beas1000-8	0.1	351994	351894	0.028	350711.656	0.364	32	351872	0.035	350645.031	0.383	11	0.019	65.625
beas1000-9	0.1	349337	349337	0.000	347868.656	0.420	34	**349337**	**0.000**	347800.813	0.440	16	0.020	52.941
beas1000-10	0.1	351415	351415	0.000	350170.125	0.354	26	**351415**	**0.000**	350108.969	0.372	9	0.018	65.385
beas2500-1	0.1	1515944	1515098	0.056	1510182.625	0.380	127	**1515088**	**0.056**	1509815.250	0.404	72	0.024	43.307
beas2500-2	0.1	1471392	1470386	0.068	1467646.000	0.255	110	**1470386**	**0.068**	1467485.375	0.266	55	0.011	50.000
beas2500-3	0.1	1414192	1413613	0.041	1410490.000	0.262	120	1413578	0.043	1410235.375	0.280	62	0.018	48.333
beas2500-4	0.1	1507701	1507701	0.000	1504721.375	0.198	107	**1507701**	**0.000**	1504539.000	0.210	72	0.012	32.710
beas2500-5	0.1	1491816	1491796	0.001	1488286.250	0.237	113	1491770	0.003	1488013.625	0.255	64	0.018	43.363
beas2500-6	0.1	1469162	1468319	0.057	1465333.875	0.261	117	1468293	0.059	1465043.500	0.280	71	0.019	39.316
beas2500-7	0.1	1479040	1478126	0.062	1473872.625	0.349	123	1477657	0.094	1473562.125	0.370	62	0.021	49.593
beas2500-8	0.1	1484199	1484199	0.000	1481679.250	0.170	91	**1484199**	**0.000**	1481520.125	0.180	43	0.010	52.747
beas2500-9	0.1	1482413	1482051	0.024	1479224.375	0.215	124	**1482051**	**0.024**	1478974.375	0.232	67	0.017	45.968
beas2500-10	0.1	1483355	1482544	0.055	1479378.375	0.268	113	**1482544**	**0.055**	1479117.875	0.286	61	0.018	46.018

表 3 全問題例と kb-g の問題例に対する各手法の実験結果の比較

	全問題例 (45 個)					kb-g の問題例 (10 個)				
			$C = \emptyset$		fixed $m = 100$			$C = \emptyset$		fixed $m = 100$
methods	avg(%)	time(s)	loss(%)	r-t(%)	loss(%) r-t(%)	avg(%)	time(s)	loss(%)	r-t(%)	loss(%) r-t(%)
$C = \emptyset$	0.491	615.489	——	——	—— ——	1.037	908.800	——	——	—— ——
fixed $m = 100$	0.491	64.378	0.000	89.540	—— ——	1.037	117.900	0.000	87.027	—— ——
m 減少率 40%	0.520	34.067	0.029	94.465	0.029 47.083	1.109	62.400	0.072	93.133	0.072 47.074
高速 k-opt 法 ($p = 10$)	0.571	21.622	0.080	96.487	0.080 66.414	1.193	36.500	0.156	95.984	0.156 69.042
高速 k-opt 法 ($p = 20$)	0.531	26.089	0.040	95.761	0.040 59.475	1.121	45.700	0.084	94.971	0.084 61.238
高速 k-opt 法 ($p = 30$)	0.514	31.822	0.023	94.830	0.023 50.570	1.085	58.400	0.048	93.574	0.048 50.466
高速 k-opt 法 ($p = 40$)	0.504	40.733	0.013	93.382	0.013 36.728	1.066	77.200	0.029	91.505	0.029 34.521
高速 k-opt 法 ($p = 50$)	0.498	44.800	0.007	92.721	0.007 30.411	1.054	86.700	0.017	90.460	0.017 26.463
高速 k-opt 法 ($p = 60$)	0.496	54.756	0.005	91.104	0.005 14.946	1.050	104.500	0.013	88.501	0.013 11.366
高速 k-opt 法 ($p = 70$)	0.495	61.556	0.004	89.999	0.004 4.383	1.046	122.800	0.009	86.488	0.009 -4.156

[4] J.E. Beasley, "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society, vol.41, no.11, pp.1069–1072, 1990.

[5] J.E. Beasley, "Heuristic algorithms for the unconstrained binary quadratic programming problem", Technical Report, Management School, Imperial College, UK, 1998.

[6] G. Gallo, P.L. Hammer, B. Simeone, "Quadratic knapsack programs", Mathematical Programming vol.12, pp.132–149, 1980.

[7] F. Glover, G.A. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs", Management Science, vol.44, no.3, pp.336–345, 1998.

[8] P.L. Ivănescu, "Some network flow problems solved with pseudo-Boolean programming", Operations Research, vol.13, pp.388–399, 1965.

[9] 片山謙吾, 成久洋之, "バイナリー 2 次計画問題に対する変形 *k-opt* 局所探索法", 電子情報通信学会論文誌 (A), vol.J84-A, no.3, pp.430–435, Mar. 2001.

[10] 片山謙吾, 谷昌史, 成久洋之, "バイナリー 2 次計画問題の地形解析と遺伝的局所探索の性能", 電子情報通信学会論文誌 (A), vol.J84-A, no.10, pp.1258–1271, Oct. 2001.

[11] K. Katayama, M. Tani, and H. Narihisa, "Solving large binary quadratic programming problems by effective genetic local search algorithm", Proc. of the 2000 Genetic and Evolutionary Computation Conference, Jul.8–12, Las Vegas, USA, pp.643–650, 2000.

[12] K. Kohmoto, K. Katayama, and H. Narihisa, "Empirical Knowledge of a Parameter Setting in k-opt Local Search for the Binary Quadratic Proggramming Problem", Proc. of the 2001 Seventeenth International Joint Conference on Artifical Intelligence (IJCAI-01), Aug 4–10, Seattle, USA, pp.21–26, 2001.

[13] 河本敬子, 片山謙吾, 成久洋之, "バイナリー 2 次計画問題に対する k-opt 局所探索法の効率化", 電子情報通信学会論文誌, vol.J85-D-1, no.3, pp.322–328, Mar. 2002.

[14] J. Krarup, P.M. Pruzan, "Computer aided layout design", Mathematical Programming Study, vol.9, pp.75–94, 1978.

[15] R.D. McBride, J.S. Yormark, "An implicit enumeration algorithm for quadratic integer programming", Management Science, vol.26, no.3, pp.282–296, 1980.

[16] P. Merz and B. Freisleben, "Greedy and local search heuristics for unconstrained binary quadratic programming", Journal of Heuristics, vol.8, no.2, pp.197–213, 2002.

[17] P.M. Pardalos, G.P. Rodgers, "A branch and bound algorithm for the maximum clique problem", Computers and Operations Research, vol.19, no.5, pp.363–375, 1992.

[18] P.M. Pardalos, J. Xue, "The maximum clique problem", Journal of Global Optimization, vol.4, pp.301–328, 1994.

[19] A.T. Phillips, J.B. Rosen, "A quadratic assignment formulation of the molecular conformation problem", Journal of Global Optimization, vol.4, pp.229-241, 1994.

[20] 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.

Effect of k-opt Local Search based on Knowledge for the Binary Quadratic Programming Problem

Keiko KOHMOTO, Kengo KATAYAMA* & Hiroyuki NARIHISA*

Graduate School of Engineering
**Department of Information & Computer Engineering*
Faculty of Engineering
Okayama University of Science
Ridaicho 1-1, Okayama 700-0005, Japan

(Received November 1, 2002)

Since many of combinatorial optimization problems are NP-hard, it is known that it is very difficult to calculate an optimal solution. However, actually, if a solution with accuracy high to some extent can be calculated even if there is no guarantee of optimal nature, it is sufficiently satisfactory in many cases. An approximate algorithm is used for such a purpose. It is known that local search (LS), which is one of the basic strategies of an approximate algorithm can calculate a solution with the accuracy high, which is comparatively short time to various combinatorial optimization problems.

In this paper, we focus on the most powerful the k-opt local search heuristic in LS known for the binary quadratic programming problem (BQP). The k-opt local search heuristic for BQP searches for the k-opt neighborhood solution in an inner-loop, and evaluates its neighborhood solution in an outer-loop. And finally the resulting solution becomes a local optimum. We analyze about an iteration number of the inner-loop when the best solution is obtained in each outer-loop. Based on the knowledge by this analysis, we set up an iteration number of the inner-loop, and propose fast k-opt local search to reduce the more search time. In comparison with the existent the efficiency enhancement technique, our proposal method shows that search time can be more reduced, maintaining an equivalent search performance.