2005－AL－99　(10)
2005／1／20

新たな手法に基づく最大クリーク抽出の効率化

仲谷 洋幸, 富田 悦次

電気通信大学大学院 電気通信学研究科 情報通信工学専攻
〒 182-8585　東京都調布市調布ヶ丘 1-5-1
E-mail: {b10074, tomita}@ice.uec.ac.jp

あらまし　最大クリークを効率良く抽出するために，これまで筆者らは節点に対する適切な近似彩色と整列を援用した分枝限定アルゴリズムを提唱してきている．本稿では，同目的のために異なった手法を基にして他者から新しく考案されたアルゴリズムに対して前記の分枝限定手法を組合せ，その結果として同アルゴリズムを顕著に効率化し，提唱した分枝限定手法の有効性を明らかにした．

キーワード　最大クリーク, 組合せ最適化, 分枝限定法, 実験的評価

An Improved Algorithm for the Maximum Clique Problem Based on Another Approach

Hiroyuki NAKATANI, and Etsuji TOMITA

Department of Information and Communication Engineering
The University of Electro-Communications
Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, Japan

E-mail:{b10074, tomita}@ice.uec.ac.jp

Abstract　We have already presented an effective depth first search branch-and-bound algorithm for finding a maximum clique, where appropriate approximate coloring and sorting techniques of vertices are employed to prune the branches. We show here that such techniques can also be successfully applied to New algorithm for the same problem that is based on another approach so to remarkably improve its efficiency.

Key words　maximum clique, combinatorial optimization, branch-and-bound algorithm, experiments

1　はじめに

無向グラフ中の最大クリークを抽出する問題は典型的な NP 困難問題で難しいにもかかわらず, その重要性より, これまでに幾多の研究がなされている（文献 [1]〜[11]). こうして, 高速な最大クリーク抽出アルゴリズムが考案されたことにより, バイオインフォマティクス [12]〜[14], 画像処理 [15], 量子論理回路設計 [16], DNA 配列設計 [17], 薬剤設計問題 [18, 19] などへの新しい応用の成果が得られている．これらの応用を更に発展させるためには，最大クリーク抽出アルゴリズムの高速化が非常に重要な基本となる．そこで筆者らは, 深さ優先探索を基礎とし, それに近似彩色による分枝限定条件を加えることなどによって効率的な厳密最大クリーク抽出アルゴリズム MCQ 等を提唱してきている [9]. 一方, これらとは独立に発表されたアルゴリズム New[8] は部分グラフを拡大していくという手法を用いて, 従来の深さ優先探索とは別の枠組みを提唱した．しかし, このアルゴリズムは実行時間が極端に増大したり, 解が求まらないことがあったりと, そのままでは必ずしも他のアルゴリズムに勝っているとは言い難い．従って, 本稿では新しいアルゴリズム [8] を元にして, 深さ優先探索アルゴリズム MCQ における中心的手法の援用により, 新しい枠組みにおけるアルゴリズムに対しても, その顕著な高速化を達成した.

以下では, まず基礎となるアルゴリズムについて説明し, それから改良点について述べる．最後に計算機実験により今回提唱するアルゴリズムの有効性を確認した.

2 諸定義及び記法

本稿では特に断りのない限り以下のような定義と記法を用いる.

$G=(V,E)$ を V を節点集合, E を枝集合とする単純無向グラフであるとする. V は順序つき集合であり, i 番目の節点を $V[i]$ と表す. グラフ G における枝の存在割合 (枝密度)$Rate(G)$ は以下のように計算される.

$$\begin{aligned} Rate(G) &= \frac{\text{グラフ中の枝の本数}}{\text{節点全てが隣接する場合の枝の本数}} \\ &= \frac{2|E|}{|V|(|V|-1)} \end{aligned}$$

G において枝で結ばれている二つの節点は隣接していると言う. グラフ G において節点 v に隣接する節点の集合を $\Gamma(v)$ で表記し, $deg(v)=|\Gamma(v)|$ を v の次数と定義する.

$G=(V,E)$ において V の部分集合 S によって誘導される部分グラフは $G(S)=(S,E\cap(S\times S))$ と表され, 本稿では部分グラフと記述した場合, 誘導部分グラフを表す.

G におけるクリークとは全ての異なった節点 $x,y\in S$ について, $(x,y)\in E$ であるような部分グラフである. G における最大クリークとはその中で最大サイズのものを指す. グラフ中の全節点を独立節点集合に分割することを彩色と言う.

2.1 分枝限定

クリーク探索の全過程は, 全節点集合 V を根, 各時点における候補節点集合 U を頂点として各候補節点集合について親子関係にあるものを枝で結んだ探索木で表現される. この探索木における枝を分枝といい, 探索に要した枝数を分枝数という. アルゴリズムを効率化するためには, 探索領域を小さくする, 即ち, 分枝数を削減することが効果的であり, これを何らかの方法を用いて実現するのが分枝限定法である. 最大クリーク抽出において最も基本的な分枝限定条件は, 現在保持しているクリークを Q, 候補節点集合を U, 現在までに求まっている最大クリーク Q_{max} とすると,

$$|Q|+|U|\leq|Q_{max}|$$

で表され, このような条件が成り立つときはこれから先探索を行っても, 現在保持しているよりも大きいサイズの極大クリークが存在しないことが明らかであるため, 探索を打ち切ることができる.

気をつけなければならないのは, アルゴリズムによって枝一本当たりの処理時間が異なるということである. 分枝数を大幅に削減するために複雑な処理を行うと, 枝一本当たりの処理時間が増大し, 総合して実行時間も増大してしまう場合がある. 従って, ただ単に分枝数を削減すれば良いというわけではなく, 分枝限定のための処理にかかる時間とその分枝数の削減のトレードオフ問題を解決しなければならないのである.

3 基礎アルゴリズム [8]

3.1 概要

本稿の基礎アルゴリズムとなる [8] について述べる. 端的に言えば, このアルゴリズムは対象となるグラフの部分グラフの最大クリークを下界として分枝限定を行っていると言うことができる.

要素数が n の全節点集合 S において, 節点番号 $i(1\leq i\leq n)$ 以降の全節点を含む部分グラフを $S_i=\{S[i],S[i+1],\ldots,S[n]\}$ とする. 初期値 $i=n$ とし, S_i における最大クリークサイズ $c[S[i]]$ を求める. 次に $c[S[i]]$ を下界として, S_{i-1} における最大クリークサイズ $c[S[i-1]]$ を求める. 以下同様にしていき, $i=1$ となる時, すなわち S_1 における最大クリークサイズ $c[S[1]]$ を求めたとき, それが対象となるグラフの最大クリークサイズとなっている. ここで注意すべきは, $c[S[i-1]]=c[S[i]]$ *or* $c[S[i-1]]=c[S[i]]+1$ となっていることである. これは, S_{i-1} の節点数が S_i の節点数+1 であることから, 部分グラフの最大クリークサイズが更新されたとしても, 高々1 しか変わらないからである.

図 1 に, 擬似パスカル言語による具体的なアルゴリズムを示す.

3.2 関数 $c[S[i]]$

関数 $c[S[i]]$ は S_i における最大クリークサイズであり, 次の部分グラフ S_{i-1} における最大クリークサイズの下界である. よって, 候補節点集合 U における最大クリークの上界は, U が S_i に含まれているとすると, $c[S[i]]$ で与えられるため, 次のような分枝限定条件を考えることができる.

$$size+c[S[i]]\leq max$$

この分枝限定条件は従来のアルゴリズムでは得られない, アルゴリズム New 固有のものである.

3.3 アルゴリズム New の特徴

New の手順はすでに説明したが, ここでは New が実際に問題となるグラフに適用したときどのような特徴を持っているかについて述べる.

文献 [8] では, ランダムグラフと DIMACS のベンチマーク用グラフについてアルゴリズムの計算機実験を行っている. その結果を観察すると, DIMACS ベンチマークグラフの幾つかのグラフについて解を求めるまでに極端に時間がかかったり, または解が求まらないということがある. この理由としては, 部分グラフを拡張していくという手法をとっているためであると考えられる. なぜならば, 解となる節点が部分グラフに全て含まれて初めて, 最大クリークが発見されるため, 早い段階で部分グラフに含まれない場合には, 最大クリー

クがなかなか更新されないということがあるからである. つまり, どのような順序で節点が部分グラフに加えられるかというのは初期節点の並びで決定されるため, 初期節点の並びに原因があると言うことができる.

要するに, New は他のアルゴリズムに比べ, 初期節点の並びに対して敏感であるということである.

```
function clique(U,size)
    if U ≠ ∅ then
        if size > max then
            max := size;
            最大クリークの更新;
            found := true;
        fi
        return
    fi
    while U ≠ ∅ do
        if size + |U| ≤ max then
            return
        fi
        i := Min{j|U[j] ∈ U};
        if size + c[U[i]] ≤ max then
            return
        fi
        U := U \{U[i]};
        clique(U ∩ Γ(U[i]), size + 1);
        if found = true then
            return
        fi
    od
return
 function new
    max := 0
    for i := n downto 1 do
        found := false;
        clique(S_i ∩ Γ(S[i]), 1);
        c[S[i]] := max;
    od
return
```

図 1. アルゴリズム New

4 提唱するアルゴリズム

これより, 本稿で提唱するアルゴリズム New_MCQ について述べる. 最大クリーク抽出アルゴリズムを説明する上で重要なのは, 基礎となるアルゴリズム, 初期節点の並び, そして分枝限定条件の三項目である. 基礎となるアルゴリズムについてはすでに述べているので, 後の二つについて New_MCQ がどのような手法を用いているかを述べる.

4.1 初期節点の並び

初期節点の並びは最大クリーク抽出アルゴリズムにおいて非常に重要である. 初期節点の並びの変化は探索中の候補節点集合の並びに影響し, さらには実行時間に効果を及ぼす. New_MCQ は従来の MCQ 系統のアルゴリズムとは異なり, [8] のアルゴリズム New を基礎としているため, どのように初期節点の並びを決定するかは, すなわちどのように部分グラフ $S_n, S_{n-1}, ..., S_1$ を決定するかということになる. 単に元のグラフの最大節点番号から, 節点番号 i までを含む節点集合を S_i とするのが最も単純な方法であるが, 本稿では, 部分グラフに含まれる節点の次数が高いほど部分グラフにおける最大クリークサイズが大きいという傾向にあることを利用し, 以下のような決定法を用いた.

Step1. 初期設定：$U := S$,$S := \emptyset$,$i := 1$

Step2. U において, 次数が最小の節点を選び出し, 複数個存在する場合は隣接する節点の次数の和が最小となる節点 $S[i]$ を抽出.

Step3. $U := U \setminus \{S[i]\}$

Step4. $i := i + 1$

Step5. $i := n$ となるまで Step 2 から Step 4 を繰り返す.

このようにして処理し, 抽出されたものから順に並べ, これを改めて節点集合 S とする (一番最初に抽出された節点が一番最後に探索範囲に加えられることになる).

4.2 分枝限定条件の追加

New_MCQ では基礎となるアルゴリズム New の分枝限定条件に加え, アルゴリズム MCQ における近似彩色アルゴリズム NUMBER-SORT[9] による分枝限定条件を追加した. NUMBER-SORT は逐次的な近似彩色アルゴリズムであり, 彩色精度と実行時間のトレードオフを上手く解決しているため, 最大クリーク抽出アルゴリズムにおける分枝限定において重要な役割を果たす. ここではアルゴリズムのみを掲載する (図 2 を参照). $N[V[i]]$ は NUMBER-SORT によって節点 $V[i]$ に与えられた色番号, U は候補節点集合, C_i は番号 i を付けられた節点集合である. また, SORT を行わず, NUMBER の部分の処理のみを行うアルゴリズムを NUMBER と表記する.

```
procedure NUMBER-SORT(U, N)
begin
{NUMBER}
    maxno := 1;
    C_1 := ∅;    C_2 := ∅;
    while U ≠ ∅ do
        p := the first vertex in U;
```

```
        k := 1;
        while C_k ∩ Γ(p) ≠ ∅
            do  k := k + 1;  od
        if k > maxno then
            maxno := k;
            C_{maxno+1} := ∅;
        fi
        N[p] := k;
        C_k := C_k ∪ {p};
        U := U − {p};
    od
{SORT}
    i := 1;
    for k := 1 to maxno do
        for j := 1 to |C_k| do
            U[i] := C_k[j];
            i := i + 1;
        od
    od
end
```

図 2.NUMBER-SORT

候補節点集合を U, 現在までに見つかっている極大クリークの中で最大のものを Q_{max} とすると, NUMBER-SORT を用いることによって得られる分枝限定条件は,

$$|U|+\mathrm{Max}\{N[p]| \in U\} \leq |Q_{max}|$$

となる. このように, 今回は二つの分枝限定条件を用いているわけであるが, その際に気をつけなければならない点がある. それは, 双方の分枝限定条件によって刈られる枝の範囲が独立しているかということである. なぜならば, 片方の枝刈りの範囲がもう片方の枝刈りの範囲と重複する, もしくは最悪含まれてしまうと, 全く意味のないものになってしまうからである.

実際にそれぞれの分枝限定条件が独立性が高いかどうかを, 予備実験によって確認した (図 4, 5 を参照). グラフの横軸は探索の深さを表し, 縦軸は枝刈りを行った回数とする. 関数 $c[S[i]]$ については $c[S[i]]$ のみを用いた場合と, $c[S[i]]$ と NUMBER-SORT の両方を用いた場合を掲載した. この二つの値の差が要は二つの分枝限定が重複している回数を表す. 縦軸はあくまで枝刈りが何回行われたかを表すものであり, 実際に刈られた枝の本数を表しているわけではない. 一つ言えることは, 枝刈りが行われている深さが浅ければ浅いほど, 一回の枝刈りで多くの枝が刈られるということである. 二つの図を比較すると, 関数 $c[S[i]]$ は深さの比較的浅いところ, NUMBER-SORT は深さの深いところという様に, 分枝限定がなされる領域がはっきりと分かれているのが見て取れる. 同様の傾向が他のグラフについても確認できた. よって, 二つの分枝限定条件はかなり良く独立していることが示されたといえる.

4.3 アルゴリズム New_MCQ

ここで提唱するアルゴリズム New_MCQ について, 基礎アルゴリズム New との違いは分枝限定条件を追加したというのが一番大きい変更点であるが, その他の細かな変更点について述べる.

第一に, 探索の深さ 1 においては NUMBER-SORT のソートを行わない. これは [9] に根拠を置くもので, NUMBER-SORT のソートによって候補節点集合内の次数の並びが, 破壊されてしまわないようにするためである.

第二に, 部分グラフ決定の際に残りの部分グラフがサイズ $maximal$ の極大クリークになる場合があることを利用し, 元となるグラフの最大クリークの下界を与える (最低でも $maximal = 1$ は得られる). すると, 部分グラフ $S_n, \ldots, S_{n-maximal+1}$ については, すぐさま関数 $c[S[i]]$ の値を決定することができ, 実際に探索するのは $S_{n-maximal}, \ldots, S_1$ のみで良いことになる. このことから, 分枝数の削減と処理の軽減が期待できる. 以上の変更を施したアルゴリズムを図 3 に擬似パスカル言語で示す. なお, 節点集合 U が極大クリークとなっているか否かについては, まず各節点の次数を計算し, U が正則グラフであるかどうかを調べる. その後, 各節点の次数が $|U|-1$ かどうかを確認することにより判定を行っている.

```
function clique(U,size)
    if U = ∅ then
        if size > max then
            max := size;
            最大クリークの更新;
            found := true;
        fi
        return
    fi
    while U ≠ ∅ do
        p := Max{N[q] | q ∈ U} である節点;
        if size + N[p] ≤ max then return fi
        if size + Max{c[r] | r ∈ U}
                    ≤ max then return fi
        U := U \ {p};
        NUMBER-SORT(U ∩ Γ(p), N);
        clique(U ∩ Γ(p), size + 1);
        if found = true then return fi
    od
return
function New_MCQ
    U := S; S := ∅; max := 0;
    for i := 1 to n do
        if U が極大クリーク then
            for j := 1 to |U| do
                S[i] := U[j];
                i := i + 1;
            od
```

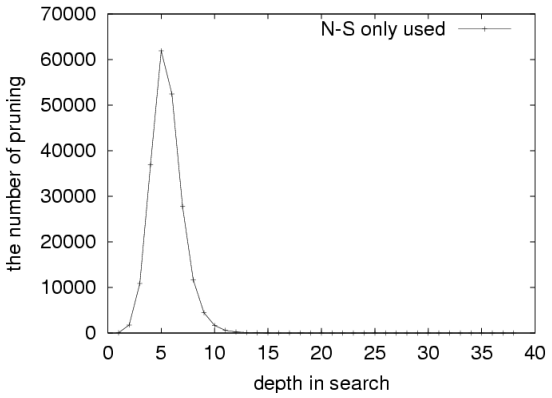

図 4: NUMBER-SORT の枝刈り

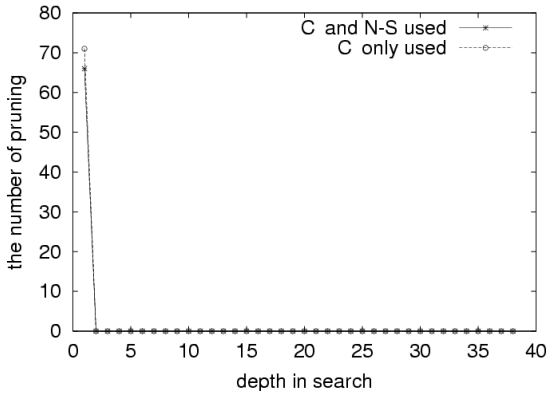

図 5: 関数 c[S[i]] の枝刈り

$maximal := |U|$;
goto Exit;
fi
$D_{min} := U$ において, 最小次数である
節点の集合;
$S[i] := D_{min}$ において, 隣接する節点の次数の
和が最も小さい節点;
$U := U \setminus \{S[i]\}$;
od
Exit:
for $i = n - maximal$ **downto** 1 **do**
$found := false$;
$S'_i := S_i \cap \Gamma(S[i])$;
NUMBER(S'_i, N);
clique$(S'_i, 1)$;
$c[S[i]] := max$;
od
return

図 3. アルゴリズム New_MCQ

5 計算機実験

以下のような計算機実験を行い提唱するアルゴリズムの有効性を確認した.

5.1 評価方法および実験環境

アルゴリズムの評価基準の一つとしては, 最悪時間計算量があるが, これはあくまで漸近的な判断材料であり, 実際の計算時間の優劣とは必ずしも一致しない. 従って, 比較対象となるアルゴリズムを実働化し, テストグラフについて計算機実験を行い, 実際に実行時間を比較することでアルゴリズムの相対評価とする.

一つ問題となるのは, 実行時間は計算機環境に大きく影響を受けるということである. 計算機の仕様やその使用状況, 更には実働化の際に用いるプログラミング言語やコンパイラなど多くの事柄に影響を受ける. これらのことを考慮に入れて, 正確にアルゴリズムの評価を行うためにできる限り共通の記述でプログラムを作成し, 実行時間の平均をとるなどした.

テストグラフは, 最大クリーク問題に対するベンチマークグラフとして多く用いられているランダムグラフ (シード 1〜10) と DIMACS のベンチマーク用グラフとする.

なお, ランダムグラフについてはシード 1 から 10 について文献 [9] と同一のものを対象とし, 十個のグラフの実行時間を平均したものを, DIMACS ベンチマークグラフについては各一回の実行時間を測定したものを掲載している. 比較対象のアルゴリズムとして, New_MCQ の基礎となっている New[8], 及び参考として MCQ[9] を用いる.

比較対象とするアルゴリズムの実行時間は各文献のものを今回の計算機環境に合わせて校正したものであり [9], また New の分枝数については文献 [8] には掲載されていなかったため, 自身で実働化し測定したものである. [8] において, DIMACS ベンチマークグラフについてなんらかの節点のシャッフルが行われているとしているが, その方法が明記されていないため, アルゴリズムを完全に再現することは不可能である. DIMACS ベンチマークグラフは特徴的なグラフであるため, その様な違いだけでも大きく結果に影響を及ぼす. よって, これについては分枝数を掲載していない.

今回実験を行った計算機環境は

- 使用計算機
  - CPU Pentium4 2.2GHz
  - メモリ 2GB
- 使用言語　C 言語　コンパイラ gcc -O2

であり, これは [9] と同一の実験環境となっている.

6 実験結果の考察

まず New と今回提唱した New_MCQ との比較であるが, ランダムグラフについては全般的に New_MCQ

の方が高速になっている．これは分枝数の大幅な削減が理由であると考えられる．NUMBER-SORT を新たに分枝限定条件として追加したことにより, 分枝限定効果が高まりこのような結果が得られたといえる．もちろん, 分枝数の枝一本あたりの計算時間は New の方が複雑な処理を行っていない分高速であるが, それを上回る分枝数の削減が New_MCQ ではなされているということである．

DIMACS ベンチマークグラフについても一部のグラフを除いてやはり, New_MCQ が高速化に成功している．特に, New では極端に時間がかかる, もしくは解が求まらないような幾つかのグラフについても New_MCQ は短時間で解を求めることに成功している．

これらのことから，今回提唱するアルゴリズム New_MCQ は New の大幅な性能向上を達成したといえる．

次に MCQ との比較であるが, ランダムグラフについては枝密度が低く規模の小さいグラフに関しては MCQ が勝るが, それ以外 ほぼ全般的に今回提唱するアルゴリズムが高速である．

DIMACS ベンチマークグラフに関しては, グラフの種類などによって得手不得手があり, 一概にどちらのアルゴリズムが優れているとは言い難いというのが現状である．得手不得手が生じる理由としては, グラフ中に存在する最大クリークの個数が大きく関係していると考えられる．DIMACS ベンチマークグラフは最大クリークの個数が極端に多い, もしくは少ないグラフが存在し, そのようなグラフについては New_MCQ は前に述べた New アルゴリズムの特徴を引き継いでいるためなかなか最大クリークを発見することができない場合がある．この点は今後の課題である．

7 まとめ

本稿では文献 [8] を基とすることで, 従来の深さ優先探索を基にしたアルゴリズムとは異なった枠組みのアルゴリズム提案した．即ち, それに深さ優先探索アルゴリズム MCQ において非常に有効であった NUMBER-SORT を追加し、改良を加えることによって大幅に性能を向上することに成功した．提唱したアルゴリズム New_MCQ と MCQ など [9, 11] に関しては，対象グラフに依存して実行時間の大小が異なり, 一概には優劣は定まらないが，筆者らが提唱した基本的な分枝限定手法 NUMBER-SORT が共に非常に有効であることを明らかにした．

謝辞 本研究は科学研究費補助金基盤研究 (B) の支援を受けている．

参考文献

[1] 富田悦次, 藤井利昭, “最大クリーク抽出の効率化手法とその実験的評価,” 電子通信学会論文誌 (D), vol.J68-D, no.3, pp.221-228 (1985).

[2] D. S. Johnson and M. A. Trick, ed., “Cliques, Coloring, and Satisfiability,” DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society (1996).

[3] E. Balas, S. Ceria, G. Cornuéjols, and G. Pataki, “Polyhedral methods for the maximum clique problem,” pp.11-28 in [2] (1996).

[4] J.-M. Bourjolly, P. Gill, G. Laporte, and H. Mercure, “An exact quadratic 0-1 algorithm for the stable set problem,”pp.53-73 in [2] (1996).

[5] E. C. Sewell, “A branch and bound algorithm for the atability number of a sparse graph,” INFORMS J. Comput. 10, pp.438-447 (1998).

[6] I.M.Bomze, M.Budinich, P.M.Pardalos and M.Pelillo, “The Maximum Clique Problem,” in: Ding-Zhu Du and P.M.Pardalos, ed., Handbook of Combinational Optimization, Supplement vol. A, Kluwer Academic Publishers, pp.1-74 (1999).

[7] T. Fahle, “Simple and fast: Improving a branch-and-bound algorithm for maximum clique, ” European Symp. on Algorithms 2002, LNCS 2461, pp.485-498 (2002).

[8] P. R. J. Östergård, “A fast algorithm for the maximum clique problem,” Discrete Appl. Math. vol.120, pp.197-207 (2002).

[9] E. Tomita and T. Seki, “An efficient branch-and-bound algorithm for finding a maximum clique,” Proc. DMTCS 2003, LNCS 2731, pp.278-289 (2003).

[10] V. Stix:“Target-oriented branch and bound method for global optimization, ” J. Global Optimization, vol.26, pp.261-277 (2003).

[11] 亀田宗克, 富田悦次, “最大クリーク抽出アルゴリズムの高速化と解析・評価,” 情報処理学会研究報告, 2004-AL-93, pp.33-40 (2004).

[12] D. Bahadur K. C. , T. Akutsu, E. Tomita, T. Seki, and A. Fujiyama, “Point matching under non-uniform distortions and protein side chain packing based on an efficient maximum clique algorithm,” Genome Informatics, 13, pp.143-152 (2002).

[13] T. Akutsu, M. Hayashida, E. Tomita, J. Suzuki, and K. Horimoto: Protein threading with profiles and constraints, Proc. IEEE Symp. on Bioinformatics and Bioengineering, pp.537-544 (2004).

表 1: ランダムグラフの平均実行時間 [sec] と分枝数

Graph			New	New_MCQ	MCQ	New	New_MCQ	MCQ
n	p	$\omega(G)$	実行時間			分枝数		
100	0.6	11-12	0.0035	0.0029	**0.0026**	16286	1085	**942**
	0.7	14-16	0.011	**0.0059**	0.007	72976	**1899**	2413
	0.8	19-21	0.1	**0.02**	0.026	652670	**4588**	6987
	0.9	29-32	1.04	**0.034**	0.066	7612372	**5714**	10854
	0.95	39-44	0.31	**0.019**	0.023	5403194	**2060**	2618
200	0.4	9	0.011	0.0092	**0.0061**	32236	3419	**2031**
	0.5	11-12	0.03	**0.024**	0.025	157437	**6532**	7900
	0.6	14	0.27	**0.085**	0.14	1316086	**23702**	38378
	0.7	18-19	4.75	**0.69**	1.13	24725249	**161895**	233495
	0.8	24-27	231.54	**9.76**	20.19	1208003946	**1699080**	3121042
	0.9	40-42		**465.47**	956.88		**45376270**	82947437
300	0.3	7-8	0.018	0.017	**0.0089**	43747	5644	**3127**
	0.4	9-10	0.074	0.043	**0.038**	288351	13727	**11561**
	0.5	12-13	0.32	**0.15**	0.24	1848988	**42968**	56912
	0.6	15-16	5.5	**1.45**	2.28	29756955	**347442**	473629
	0.7	19-21	179.71	**21.77**	37.12	1005018610	**4138716**	6214907
	0.8	28-29		**1147.3**	2140.1		**152215691**	252298931
500	0.2	7	0.03	0.024	**0.012**	54824	9959	**3990**
	0.3	8-9	0.13	0.078	**0.067**	400717	23266	**18829**
	0.4	11	0.94	**0.34**	0.48	2735754	**86157**	124059
	0.5	13-14	11.4	**3.75**	5.4	58212932	**904012**	1124209
	0.6	17	288.1	**59.9**	96.34	1484649635	**11655459**	16062634
	0.7	22-23		**3239.06**	5298.1		**494735984**	708601190
1000	0.1	5-6	0.1	0.049	**0.024**	59447	14963	**4101**
	0.2	7-8	0.33	0.25	**0.18**	896471	68548	**43380**
	0.3	9-10	2.58	**1.68**	1.85	13137913	**447022**	463536
	0.4	12	36.46	**13.98**	22.98	153945045	**3101858**	4413740
	0.5	15		**366.94**	576.36		**68089057**	89634336

[14] D. Bahadur K.C., E. Tomita, J. Suzuki, and T. Akutsu, "Protein side-chain packing problem: A maximum edge-weight clique algorithmic approach," J. Bioinformatics and Computational Biology (to appear).

[15] 堀田一弘, 富田悦次, 高橋治久, "最大クリーク抽出に基づく向きの変化に依存しない人物の顔検出," 情報処理学会論文誌 数理モデル化と応用, vol.44, no.SIG14 (TOM9), pp.57-70 (2003).

[16] 名久井行秀, 西野哲朗, 富田悦次, 中村知倫, "節点重み最大クリーク抽出に基づく量子論理回路の深さ最小化," 京大数解研講究録 1325, pp.45-50 (2003).

[17] S. Kobayashi, T. Kondo, K. Okuda, and E. Tomita, "Extracting globally structure free sequences by local structure freeness," Proc. DNA9, p.206 (2003).

[18] H. Xu and D. K. Agrafiotis, "Retrospect and prospect of virtual screening in drug discovery," Current Topics in Medicinal Chemistry, vol.2, no. 12, pp.1305-1320 (2002).

[19] P. Willett, "Similarity-based approaches to virtual screening," Biochem. Soc. Trans. vol.31, pp.603-606 (2003).

表 2: DIMACS ベンチマークグラフについての実行時間 [sec]

Graph				New	New_MCQ	MCQ
Name	n	p	$\omega(G)$	実行時間		
MANN_a27	378	0.99	126	>3500	**5.77**	8.49
MANN_a9	45	0.927	16	0.035	0.00033	**0.0002**
brock200_1	200	0.745	21	19.05	**2.38**	2.84
brock200_2	200	0.496	12	**0.018**	0.02	0.02
brock200_3	200	0.605	15	0.15	0.11	**0.09**
brock200_4	200	0.658	17	0.34	**0.14**	0.32
c-fat200-1	200	0.077	12	0.0035	0.00089	**0.00024**
c-fat200-2	200	0.163	24	0.0035	0.00155	**0.0005**
c-fat500-1	500	0.036	14	0.025	0.00505	**0.0011**
c-fat500-10	500	0.186	126	0.025	0.04	**0.0061**
hamming10-2	1024	0.99	512	**0.88**	2.06	1.82
hamming6-2	64	0.905	32	0.0035	0.00044	**0.0003**
hamming6-4	64	0.349	4	0.0035	0.00034	**0.0001**
hamming8-2	256	0.969	128	**0.014**	0.02	0.021
hamming8-4	256	0.639	16	0.3	**0.13**	0.34
johnson16-2-4	120	0.765	8	**0.095**	0.25	0.34
johnson8-2-4	28	0.556	4	0.0035	0.00008	**0.000022**
johnson8-4-4	70	0.768	14	0.0035	0.0016	**0.0009**
keller4	171	0.649	11	0.175	0.05	**0.045**
p_hat1000-1	1000	0.245	10	2.05	**0.82**	0.86
p_hat300-1	300	0.244	8	0.014	0.00943	**0.0053**
p_hat300-2	300	0.489	25	0.35	**0.06**	0.08
p_hat500-1	500	0.253	9	0.102	0.06	**0.04**
p_hat500-2	500	0.505	36	150.05	**0.79**	6.22
p_hat700-1	700	0.249	11	0.24	0.17	**0.16**
san1000	1000	0.502	15	**0.18**	1.13	10.06
san200_0.7_1	200	0.7	30	0.2	0.02	**0.017**
san200_0.7_2	200	0.7	18	0.014	**0.0061**	0.011
san200_0.9_1	200	0.9	70	0.095	**0.02**	2.3
san200_0.9_2	200	0.9	60	1.5	**0.03**	3.49
san400_0.5_1	400	0.5	13	**0.011**	0.05	0.04
san400_0.7_1	400	0.7	40	>3500	**0.67**	1.72
san400_0.7_2	400	0.7	30	177.6	2.33	**1.58**
sanr200_0.7	200	0.702	18	4.95	**0.57**	0.92
sanr400_0.5	400	0.501	13	2.32	**0.7**	1.47