Vol.2018-AL-166 No.5
2018/1/28

根付きクリークマイナーアルゴリズムの設計と実験による評価

大塚 広夢[†1,a)]　玉木 久夫[†1,b)]　牧井 慶太朗[†1,c)]

概要：根付きクリークマイナー問題とは，グラフ G とその頂点集合 R が与えられた時に，G 上に R で根付けされるクリークマイナーが存在するかを判定する問題である．この問題は，Bodlaender と Koster によって導入された，木幅に対し安全なセパレータの十分条件において重要な応用を持つ．本稿では，まず根付きクリークマイナーが存在する必要条件について述べ，厳密アルゴリズムの設計を行い，これら必要条件とアルゴリズムの実験的評価について述べる．

キーワード：安全なセパレータ，木分解，根付きクリークマイナー，バックトラック，動的計画法

1. はじめに

R をグラフ G の頂点集合とする．G の R で根付けされたクリークマイナーとは，対毎に隣接し，対毎に素な連結集合の族で，どの要素も R の頂点をちょうどひとつ含むようなものをいう．根付きクリークマイナー問題とは，そのような族が存在するかの判定を行う問題である．

この問題は木幅計算と安全なセパレータの概念を通して次のように関連している．木幅に対する安全なセパレータとは，そのセパレータをクリークに補完してもグラフの木幅が変わらないようなセパレータであり，木幅計算の対象をより小さいグラフインスタンスに分解するために用いる事ができる．根付けされたクリーク問題は，安全なセパレータの十分条件を与えるため重要である．[1]

木分解は例えばグラフの木幅をパラメータとした固定パラメータアルゴリズムの設計などに利用できるため，素早く最適な木分解を求めることは非常に重要である．しかし，最適な木分解を求めることは NP 困難であり，その木分解の導出を早める上で重要な根付きクリークマイナー問題も同様に NP 困難である．

PACE(Parameterized Algorithms and Computational Experiments Challenge)2017 の玉木らの提出 [2] では，クリークマイナーによる十分条件を用いた安全なセパレータの検出を，発見的解法によって行なっている．本稿ではその根付きクリークマイナー問題に対する必要条件を導入し，厳密解法としてバックトラックによるものと木分解上の動的計画法を設計した．この必要条件の効果と，厳密解法と先の発見的解法の比較を行い，その有用性を見て行く．

†1 現在，明治大学
a) ohtsuka_yume@cs.meiji.ac.jp
b) tamaki@cs.meiji.ac.jp
c) makii@cs.meiji.ac.jp

2. 準備

グラフを G，頂点集合を $V(G)$，辺集合を $E(G)$ とする．ある頂点 $v \in V(G)$ に対し，その隣接頂点集合を $N_G(v) = \{u \in V(G) \mid \{u, v\} \in E(G)\}$，閉隣接頂点集合を $N_G[V] = N_G(v) \cup \{v\}$ とする．ある頂点集合 $U \subseteq V(G)$ に対し，$G[U]$ を U によって誘導される部分グラフとし，U の G 上の閉隣接頂点集合を $N_G[U] = \bigcup_{v \in U}(N_G[v])$，$U$ の G 上の隣接頂点集合を $N_G(U) = N_G[U] \setminus U$ とする．

全ての頂点対が互いに隣接するグラフを**完全グラフ**といい，頂点集合 $U \subseteq V(G)$ が G において**クリーク**であるとは $G[U]$ が完全グラフである時をいう．また，$K(U)$ を U を頂点集合とする完全グラフとする．

頂点集合 C が**連結**であるとは，全ての $u, v \in C$ に対して，$G[C]$ 上に u と v の間のパスが存在するときをいい，そのような C を**連結集合**と呼ぶ．またグラフ G において，$V(G)$ が連結である時，G を**連結グラフ**と呼び，極大で連結な部分グラフのことを**連結成分**という．

頂点集合 $R \subseteq V(G)$ 上の任意のグラフを H とする．R の各頂点で定義される G の互いに素な連結集合の族 $\{C_v\}_{v \in R}$ について，各頂点 $v \in R$ に対して $R \cap C_v = \{v\}$ が成立し，かつ各辺 $\{u, v\} \in E(H)$ に対して $x \in C_v, y \in C_u$ を満たすある辺 $\{x, y\} \in E(G)$ が存在するとき，グラフ H は G の R で**根付けされるマイナー**であるという．ここで，族 $\{C_v\}_{v \in R}$ を H の**マイナーモデル**という．R で根

付けされるあるマイナー H について，そのマイナーモデル $\{C_v\}_{v\in R}$ が $\bigcup_{v\in R} C_v = V(G)$ を満たし，さらにそのマイナーモデルと G の辺集合によって定義される集合 $\{\{u,v\} \mid \{x,y\} \in E(G), x \in C_u, y \in C_v\}$ が $E(H)$ と等しいとき，マイナー H を**フルマイナー**であるという．G の R で根付けされるマイナー H が**クリークマイナー**であるとは，H が完全グラフであるときをいう．

提案するアルゴリズムは，以下で述べる木分解（tree decomposition）を用いる．

定義 1 G の**木分解** $(T, \{X_t\}_{t\in V(T)})$ とは，木 T と T の各ノード $t \in V(T)$ によって定義される頂点集合 $X_t \in V(G)$ の族で，以下の条件を満たすものである．

(1) $\bigcup_{t\in V(T)} X_t = V(G)$,
(2) 任意の辺 $\{v,w\} \in E(G)$ に対して，$v,w \in X_t$ を満たす $t \in V(T)$ が存在し，
(3) 任意の頂点 $v \in V(G)$ に対して，ノードの集合 $\{t \mid \in X_t\}$ は T において連結である．

ここで，族 $\{X_t\}_{t\in V(T)}$ の各要素を**バッグ**と呼ぶ．木分解 T の**幅**とは，全ての T のノード $t \in V(T)$ における X_t の大きさの最大値から 1 減じた値であり，G の**木幅** $\mathrm{tw}(G)$ とは，G の全ての木分解の幅の最小値である．

3. 必要条件

この章では，根付きクリークマイナーが存在するための必要条件について述べる．

グラフを G，その頂点集合を $R \subseteq V(G)$ とする．R 中の異なる 2 頂点 u,v の全ての組み合わせに対し，以下のいずれかを満たすとき，R は G において**クリキッシュ**であるという．

(1) $u,v \in E(G)$.
(2) $u,v \in N_G(C)$ を満たす $G[V(G) \setminus R]$ の連結成分 C が存在する．

補題 1 G に $R \subseteq V(G)$ で根付けされるクリークマイナーが存在するならば，R は G においてクリキッシュである．

証明. $\{C_v\}_{v\in R}$ を G の R で根付けされるクリークマイナーのマイナーモデルとし，R が G においてクリキッシュでないと仮定する．異なる 2 頂点 $u,v \in R$ で，$\{u,v\} \notin E(G)$ かつ $u,v \in N_G(C)$ を満たす $G[V(G) \setminus R]$ の連結成分 C は存在しないようなものについて考える．マイナーの定義より，$x \in C_u$ かつ $y \in C_v$ を満たす辺 $\{x,y\} \in E(G)$ が必ず存在し，u と v の間に内部頂点に R の頂点を含まないようなパスが存在することを意味する．したがって，$u,v \in N_G(C)$ を満たす $G[V(G) \setminus R]$ の連結成分 C が存在しないことに矛盾する．よって，R はクリキッシュである． □

クリークを含むグラフの木分解に関して，以下の補題が知られている．

補題 2 G の頂点集合 $R \subseteq V(G)$ とする．$G[R]$ が完全グラフであるならば，G の任意の木分解において R を含むバッグが存在する．

補題 2 を用いて，次の補題が導ける．

補題 3 G の R で根付けされるクリークマイナーが存在するならば，$\mathrm{tw}(G) \geq |R| - 1$ が成立する．

証明. $\{C_v\}_{v\in R}$ を G の R で根付けされるクリークマイナーのマイナーモデルとし，$(T, \{X_t\}_{t\in V(T)})$ を G の最適な木分解とする．ここで，$(T, \{X_t\}_{t\in V(T)})$ の幅は $\mathrm{tw}(G)$ と等しい．$(T, \{X_t\}_{t\in V(T)})$ から，次の操作によって $(T, \{X'_t\}_{t\in V(T)})$ を得る．各 X'_t について，$X'_t := X_t$ とする．さらに，各頂点 $v \in R$ について，$X_t \cap C_v \neq \emptyset$ を満たす $t \in V(T)$ について $X'_t := (X'_t \setminus C_v) \cup \{v\}$ とする．このとき，各頂点 $v \in R$ について，$v \in X_t$ ならば，かつそのときに限り，$v \in X'_t$ であるから，$\{t \mid v \in X'_t\}$ は T において連結である．さらに，$\bigcup_{t\in V(T)} X'_t = R$ であり，クリークマイナーの存在より，各辺 $\{u,v\} \in E(K(R))$ について，$x \in C_u$ かつ $y \in C_v$ を満たす辺 $\{x,y\} \in E(G)$ が存在するから，木分解 $(T, \{X_t\}_{t\in V(T)})$ において $x,y \in X_t$ が存在する．よって，$u,v \in X'_t$ である．$(T, \{X'_t\}_{t\in V(T)})$ は $K(R)$ の木分解であり，上の操作で幅は増加しないから，その幅は $\mathrm{tw}(G)$ 以下である．補題 2 より，$\mathrm{tw}(K(R)) \geq |R| - 1$ であるから，$\mathrm{tw}(G) \geq |R| - 1$ である． □

4. アルゴリズム

この章ではグラフ G と頂点集合 $R \subseteq V(G)$ が与えられた時，グラフ G 上に R で根付けされるクリークマイナーが存在するかを判定するアルゴリズムとしてバックトラックによるものと木分解上の動的計画法の二つを示す．

4.1 バックトラック

この節では根付きクリークマイナー問題の解法の一つとしてバックトラックアルゴリズムの設計を行う．各頂点 $v \in V(G) \setminus R$ について R の各頂点に対して定義される G の互いに素な連結集合の族 $\{C_r\}_{r\in R}$ のどれに含めるかをバックトラックによって網羅的に決定していく．またここではどの $\{C_r\}_{r\in R}$ にも含まれていない頂点は頂点集合 U に属しているものとする．

アルゴリズムの設計にあたり，以下の二つの補題を利用する．

補題 4 $R' \subseteq R$ とする．G 上に R で根付けされるクリークマイナーが存在する時，G に R' で根付けされるクリークマイナーが必ず存在する．

証明. マイナーモデルを $\{C_v\}_{v\in R}$ としたとき，$\{C_v\}_{v\in R'}$

は明らかに R' で根付けされるクリークマイナーのマイナーモデルである．□

補題 5 $R' = \{r_1, r_2, \ldots, r_{i-1}\} \subset R$ とし，G 上に R' で根付けされるクリークマイナーが存在するとする．G に R で根付けされるクリークマイナーが存在するならば，連結集合 C_{r_i} を決定した時，$|N(C_{r_i}) \cap U| \geq |R| - i$ が成立する．

証明． $m = |N(C_{r_i}) \cap U| < |R| - i$ とする．まだ連結集合を決定していない頂点 $r \in (R \setminus R')$ の個数は $|R| - i$ 個存在するが，C_{r_i} が頂点 r のうち隣接できるのは高々 m 種類である．従って C_{r_i} と隣接できない頂点 $r \in (R \setminus R')$ が少なくとも一つは存在するので，その連結成分 C_{r_i} をもつマイナーはクリークマイナーになり得ない．よって $|N(C_{r_i}) \cap U| \geq |R| - i$ となる必要がある．□

これらの補題をバックトラックアルゴリズムの枝刈りの条件として利用する．R の部分集合 $R' = \{r_1, r_2, \ldots, r_{i-1}\}$ とし，$r \in R'$ について連結集合 C_r が決定しているとする．新たに頂点 r_i の連結集合 C_{r_i} を決定した際，$R' \cup \{r_i\}$ で根付けされるクリークマイナーが G 上に存在しなければ補題 4 より枝刈りが行える．また $|N(C_{r_i}) \cap U| \geq |R| - i$ でなければ補題 5 により枝刈りを行え，無駄な探索を抑えることができる．

以上の枝刈りを行うバックトラックアルゴリズムを非決定性アルゴリズムを用いることで説明する．実際には全てのパターンを探索し，いずれのパターンでも YES と答えるケースがなければ答えは NO となる．

以下のアルゴリズムでは，ある頂点 v を C_r または $C_{\bar{r}}$ に加えて行く．ここで，v を $C_{\bar{r}}$ に加えることは v を C_r に加えないことを意味する．これを行うことで，再び探索候補の中に同一頂点が表れた際に重複した探索を行わないで済むようになる．また，R 上には全順序 $<$ が定義されているものとする．

このアルゴリズムによって作成された連結集合の族 $\{C_r\}_{r \in R}$ はグラフ G のマイナーモデルで，YES と答えた際にはクリークマイナーとなっている．逆に，クリークマイナーが存在する時，そのマイナーモデル通りに連結集合を構成するアルゴリズム上のパスが確かに存在するのでこのアルゴリズムは YES と答え得る．

4.2 木分解上の動的計画法

ここでは，グラフ G とその頂点集合 $R \subseteq V(G)$，G の木分解 T が与えられるとする．ここで，T は R を含むバッグを持つと仮定する．G の R で根付けされるクリークマイナーが存在するかを判定する木分解上の動的計画法を示す．

根ノード $r \in V(T)$ において，$X_r = R$ である次のように定義される好適な木分解を用いる．

定義 2 木分解 T が以下の条件を満たすとき，T は**好適** (*nice*) であるという．

任意の葉ノード $l \in V(T)$ について，$X_l = \emptyset$ であり，すべての葉でないノード $t \in V(T)$ は，次のいずれかである．

導入ノード: ノード t はひとつの子 t' を持ち，$X_t = X_{t'} \cup \{v\}$ である $v \notin X_{t'}$ が存在する．このとき頂点 v はノード t で**導入される**といい，t を**導入ノード**と呼ぶ．

忘却ノード: ノード t はひとつの子 t' を持ち，$X_t = X_{t'} \setminus \{v\}$ である $v \in X_{t'}$ が存在する．このとき v はノード t で**忘却される**といい，t を**忘却ノード**と呼ぶ．

結合ノード: ノード t はふたつの子 t_1, t_2 を持ち，$X_t = X_{t_1} = X_{t_2}$ である．このとき t を**結合ノード**と呼ぶ．

好適な木分解 T とそのノード $t \in V(T)$ に対して，$V_t = \bigcup_{t' \in V(T_t)} X_{t'}$ とする．ここで，T_t は t を根とする T の極大な部分木とする．

よく知られているように，R を含む木分解から好適な木分解への変換は可能である．以下では，根ノード r について，$X_r = R$ である好適な木分解 T とする．

各 $t \in V(T)$ について，$\mathcal{H}_t$ を $G[V_t]$ の X_t で根付けされる全てのフルマイナーの集合とする．全ての葉ノード $l \in V(T)$ に対して，$\mathcal{H}_l$ は空のグラフのみを含む．アルゴリズムでは，木分解 T の各ノード t に対して，以下で述べるノードの種類に応じたその子供間の漸化式を用いて，$\mathcal{H}_t$ を構築する．

ここで，$R = X_r$ かつ $G = G[V_r]$ より，以下の補題が明らかに成立する．

補題 6 $K(X_r) \in \mathcal{H}_r$ ならば，かつそのときに限り，G の R で根付けされるクリークマイナーが存在する．

Algorithm 1 グラフ G 上に R で根付けされるクリークマイナーが存在するかを判定する非決定性アルゴリズム

```
C_0 ← V(G)
for 各頂点 r ∈ R を昇順に do
  C_r ← C_r ∪ {r}
  S ← N_G(r)
  while S ∩ C_0 ≠ φ do
    頂点 v ∈ S ∩ c_0 を選ぶ
    S ← S \ {v}
    頂点 v を C_r または C_r̄ のどちらに加えるかを非決定的に選択する
    if 頂点 v を C_r に加えたとき then
      S ← S ∪ N_G(v)
    end if
  end while
  if S ∩ C_q = φ となるような q < r が存在する then
    fail
  end if
  if |S ∩ C_r̄| < |{q ∈ R | q > r}| then
    fail
  end if
  C_0 ← C_0 ∪ C_r̄
end for
answer YES
```

以下より，忘却ノード，導入ノード，結合ノードの順で述べる．

4.2.1 忘却ノード

忘却ノード $t \in V(T)$ とそのひとりの子供 $t' \in V(T)$，忘却頂点 $v \in V(G)$ とする．ここで，$X_t = X_{t'} \setminus \{v\}$ である．

補題 7 X_t 上のグラフ H が $G[V_t]$ の X_t で根付けされるフルマイナーであるならば，かつその時に限り，$G[V_{t'}]$ の $X_{t'}$ で根付けされるフルマイナー H' が存在する．ここで，H' は，H' 上で忘却頂点 v と隣接する頂点 u に対して，辺 $\{u,v\}$ を u に縮約することによって H が得られるグラフとする．

証明. $G[V_t]$ の X_t で根付けされるフルマイナー H とし，そのマイナーモデル $\{C_x\}_{x\in X_t}$ とする．H はフルマイナーであり $V(G[V_t])$ を分割するから，$v \in C_u$ を満たす頂点 u が存在する．C_v を $G[C_u \setminus \{u\}]$ の v を含む連結成分とする．族 $\{C'_y\}_{y\in X_{t'}}$ を次のように定義する: 各 $y \in X_{t'} \setminus \{u\}$ に対して，$C'_y = C_y$，$C'_u = C_u \setminus C_v$，$C'_v = C_v$ とする．得られた族 $\{C'_y\}_{y\in X_{t'}}$ は互いに素であり，各 $x \in X_{t'}$ について $C'_x \in \{C'_y\}_{y\in X_{t'}}$ は連結であり，$C'_x \cap X_{t'} = \{x\}$ である．さらに，この族は $V(G[V_t])$ を分離する．H' を $X_{t'}$ 上のグラフで，$\{x,y\} \in E(H')$ であるならば，かつその時に限り，$G[V_t]$ において C'_x と C'_y の間に辺が存在するグラフとする．すると，H' は $\{C'_y\}_{y\in X_{t'}}$ をマイナーモデルとして持つフルマイナーであり，H は H' から $\{u,v\}$ を u に縮約することに得られるグラフである．

逆は自明である． □

4.2.2 導入ノード

導入ノード $t \in V(T)$ とそのひとりの子供 $t' \in V(T)$，導入頂点 $v \in V(G)$ とする．ここで，$X_t = X_{t'} \cup \{v\}$ である．

補題 8 X_t 上のグラフ H が $G[V_t]$ の X_t で根付けされるフルマイナーであるならば，かつその時に限り，$N_H(v) = N_{G[V_t]}(v)$ かつ $H[X_{t'}]$ は $G[V_{t'}]$ の $X_{t'}$ で根付けされるフルマイナーである．

証明. $G[V_t]$ の X_t で根付けされるフルマイナー H のマイナーモデル $\{C_x\}_{x\in X_t}$ とする．$N_{G[V_t]}(v) \subseteq X_t$ であるから，$C_v \cap X_t = \{v\}$ を満たす連結集合 C_v は $\{v\}$ である．したがって，$H[X_{t'}]$ はマイナーモデルとして $\{C_x\}_{x\in X_{t'}}$ を持つ $G[V_t \setminus \{v\}]$ のフルマイナーであり，かつ $N_H(v) = N_{G[V_t]}(v)$ である．

逆を示すために，H を $X_{t'}$ 上のグラフで，$N_H(v) = N_{G[V_t]}(v)$ を満たすものとし，$G[V_{t'}]$ の $X_{t'}$ で根付けされるフルマイナー $H[X_{t'}]$ が存在すると仮定する．さらに，$\{C_x\}_{x\in X_{t'}}$ を $H[X_{t'}]$ のマイナーモデルとする．連結集合 $C_v = \{v\}$ とする．$G[V_t]$ において C_v と C_x に辺が存在するならば，かつその時に限り，$x \in N_{G[V_t]}(v) = N_H(v)$ である．よって，$\{C_x\}_{x\in X_t}$ は H のマイナーモデルであり，したがって，H は $G[V_t]$ の X_t で根付けされるフルマイナーである． □

4.2.3 結合ノード

結合ノード $t \in V(T)$ とそのふたりの子供 $t_1, t_2 \in V(T)$ とする．

補題 9 X_t 上のグラフ H が $G[V_t]$ の X_t で根付けされるフルマイナーであるならば，かつその時に限り，$G[V_{t_1}]$ の X_{t_1} で根付けされるフルマイナー H_1 と $G[V_{t_2}]$ の X_{t_2} で根付けされるフルマイナー H_2 で，$E(H) = E(H_1) \cup E(H_2)$ を満たすものが存在する．

証明. $G[V_t]$ の X_t で根付けされるフルマイナー H が存在するとし，そのマイナーモデル $\{C_x\}_{x\in X_t}$ とする．ここで，$X_t = X_{t_1} = X_{t_2}$ であることに注意する．$i = 1,2$ に対して，$\{C_x \cap V_{t_i}\}_{x\in X_{t_i}}$ は明らかに $G[V_{t_i}]$ のマイナーモデルである．H はフルマイナーであるから，$\{x,y\} \in E(G[V_{t_i}])$ かつ $x \in C_{iu}, y \in C_{iv}$ を満たすならば，$\{u,v\} \in E(H)$ である．したがって，$\{C_{ix}\}_{x\in X_{t_i}}$ をマイナーモデルとしてもつ $G[V_{t_i}]$ のフルマイナー H_i が存在する．ここで，$i = 1,2$ である．

逆を示すために，$G[V_{t_i}]$ の X_{t_i} で根付けされるフルマイナー H_i が存在するとして，そのマイナーモデル $\{C_{ix}\}_{x\in X_{t_i}}$ とする．ここで，$i = 1,2$ とする．X_t 上のグラフ H を $E(H) = E(H_1) \cup E(H_2)$ を満たすものとする．$\{x,y\} \in E(G[V_t])$ で，$\{x,y\} \in E(G[V_{t_i}])$ とする．H_1 と H_2 はフルマイナーであるから，$x \in C_{iu}, y \in C_{iv}$ ならば，$\{u,v\} \in E(H_i) \subseteq E(H)$ が存在する．したがって，$\{C_{1x} \cup C_{2x}\}_{x\in X_t}$ をマイナーモデルとする $G[V_t]$ の X_t で根付けされるフルマイナーが存在する． □

5. 性能評価

PACE2017 Track:A Treewidth[3] の公開インスタンス 100 個に対して玉木らによって提出されたプログラム [2] を動かし，その中で出てきた根付きクリークマイナー問題のうち，厳密に解く意義のあると思われる頂点数が元グラフの半分以下のものをインスタンスとして作成した．我々の提案した必要条件とアルゴリズムをこれらのインスタンスに対して適用することで実験的評価を行った．

実験を行うにあたり，各インスタンスに対し以下の補題を用いて問題分割を行う．

補題 10 G のクリキッシュ $R \subseteq V(G)$ とし，$G[V(G) \setminus R]$ の連結成分の族 $\mathcal{C}$ とする．さらに，$\mathcal{C}$ 上の
(1) 同一である，または，
(2) 隣接頂点の共通部分が G のクリークでない，
という 2 項関係の m 個の推移閉包の同値類 $\mathcal{C}_1, \ldots, \mathcal{C}_m$ とする．このとき，G の R で根付けされるクリークマイナーが存在するならば，かつそのときに限り，1 以上 m 以下の

各整数 i について，$G[N_G[\bigcup_{C\in\mathcal{C}_i} C]]$ の $N_G(\bigcup_{C\in\mathcal{C}_i} C)$ で根付けされるクリークマイナーが存在する．

証明． G の R で根付けされるクリークマイナーのマイナーモデル $\{C_v\}_{v\in R}$ とする．1 以上 m 以下の整数 i とし，$U=\bigcup_{C\in\mathcal{C}_i} C$ とする．このとき，$\{C_v\cap N_G[U]\}_{v\in N_G(U)}$ は $G[N_G[U]]$ の $N_G(U)$ で根付けされるクリークマイナーのマイナーモデルと主張したい．そのために，任意の 2 つの異なる頂点 $u,v\in N_G(U)$ とする．u と v が隣接しているならば主張は成立するから，u と v は隣接してないとする．$u,v\in N_G(C)$ を満たす全ての連結成分 $C\in\mathcal{C}$ は，u と v が隣接していない事実より，$C\subseteq U$ である．よって，$x\in C_u\cap N_G[U]$ かつ $y\in C_v\cap N_G[U]$ を満たす $\{x,y\}\in E(G[N_G[U]])$ が存在する．さらに，各 $v\in N_G(U)$ について，$C_v\cap N_G[U]$ は $G[N_G[U]]$ において連結であるから，したがって，$G[N_G[U]]$ の $N_G(U)$ で根付けされるクリークマイナーが存在する．

逆を示すために，1 以上 m 以下の整数 i とし，$\mathcal{C}_i$ について，$U=\bigcup_{C\in\mathcal{C}_i} C$ とする．$G[N_G[U]]$ の $N_G(U)$ で根付けされるクリークマイナーの存在を仮定する．$\mathcal{C}_i,\ldots,\mathcal{C}_m$ が互いに素であることと，クリークマイナーの存在より，各 $u,v\in N_G(U)$ について，u から v への内部頂点に R の頂点を含まないパスが存在する．したがって，G の R で根付けされるクリークマイナーが存在する． □

実験環境は次の通りである．CPU: Intel Core Xeon 2.20GHz RAM: 1024GB, OS: Ubuntu 14.04 LTS, Trusty Tahr, プログラミング言語: Java 1.8, JVM: 1.8.0_131, 最大ヒープサイズ: 24GB.

5.1 必要条件による効果

生成された根付きクリークマイナー問題のインスタンス 73780 個に対し，各時間ごとに 3 節で述べた必要条件を満たさなかったインスタンスがどの程度存在したかを表 1 に示す．なお，finish,cond,cliquish,tw はそれぞれ必要条件の判定が終わったものの個数，必要条件のうちいずれかの必要条件を満たさなかったものの個数，クリキッシュの条件により必要条件を満たさなかったものの個数，木幅の条件により必要条件を満たさなかったものの個数を表すものとする．また，制限時間は各必要条件の判定にかけた時間である．

表 1 必要条件の効果

時間 (ms)	finish	cond	cliquish	tw
1	0	0	0	0
10	21471	6290	32	6284
100	66335	18831	235	18810
1000	71578	19222	269	19195
10000	73551	19266	284	19229
50000	73780	19269	284	19232

50000ms かけることでどのインスタンスに対しても必要条件の判定が終了した．表 1 より，100ms 程度で 9 割のインスタンスで判定が終わっており，最終的に必要条件を満たさなかったもののほとんどが 100ms で検出できているため，必要条件の判定に 100ms 程度なら時間をかける意味があると言える．

内訳を見てみると今回生成されたインスタンスについてはクリキッシュによる条件で判定できたものは少なかった．他のグラフに対してもこの傾向が表れるかは調べてみる価値があるだろう．逆に木幅による条件で問題を削減できた例は非常に多かった．よって厳密な木分解が行えるような小さな問題に対しては木幅を求める意味は十分にあると言える．

5.2 発見的解法 vs 本アルゴリズム

5.1 節の必要条件を満たした 54511 インスタンスに対し，今回提案した厳密解法の性能を測るために，玉木らのアルゴリズム [2] によって利用されてる発見的解法と比較した．10ms,100ms,10000ms の制限時間を設け，いくつ YES と答えられたかをそれぞれ表 2,3,4 に示す．厳密アルゴリズムにかける時間は制限時間を半分に分け，バックトラックと木分解上の動的計画法に割り振り，どちらか一方でも YES と答えたものをカウントした．また，木分解上の動的計画法にかかった時間には木分解を行う時間も含まれている．

表 2 発見的解法と厳密解法の比較 (制限時間 10ms)

頂点数	インスタンス数	発見的解法	厳密解法
$[0,10]$	33764	31691	32748
$[11,25]$	9949	5866	5199
$[26,50]$	4391	1434	766
$[51,100]$	3249	1257	649
$[101,200]$	1819	492	41
$[201,\infty)$	1339	394	62
$[0,\infty)$	54511	41134	39465

表 3 発見的解法と厳密解法の比較 (制限時間 100ms)

頂点数	インスタンス数	発見的解法	厳密解法
$[0,10]$	33764	31692	32827
$[11,25]$	9949	5882	7482
$[26,50]$	4391	1891	1376
$[51,100]$	3249	1547	861
$[101,200]$	1819	798	40
$[201,\infty)$	1339	619	76
$[0,\infty)$	54511	42429	42383

表より，発見的解法は 100ms で YES と答えるものは全て終わっていることがわかる．厳密解法の結果を見てみると，頂点数が少ない時は時間内に終わるものは多いが頂点数が 25 程度でも完全に終わる事が難しいことが分かる．それよりも頂点数が多いものについてはアルゴリズムが終

表 4　発見的解法と厳密解法の比較 (制限時間 10000ms)

頂点数	インスタンス数	発見的解法	厳密解法
$[0, 10]$	33764	31692	32827
$[11, 25]$	9949	5882	7614
$[26, 50]$	4391	1891	1512
$[51, 100]$	3249	1547	973
$[101, 200]$	1819	798	83
$[201, \infty)$	1339	619	112
$[0, \infty)$	54511	42429	43121

了せず，結果を正しく判定できないものが多くなり発見的解法に負けてしまっていることが分かる．

次に，発見的解法では YES と答えられなかったが厳密解法では YES と答えられた個数を表 5 に示す．ここで使われている時間は先ほど同様二つの提案アルゴリズムに半分ずつ割り当て，どちらか一方でも YES と答えられたものをカウントしている．

表 5　厳密解法のみが YES と答えたもの

頂点数	10ms	100ms	10000ms
$[0, 10]$	1119	1135	1135
$[11, 25]$	360	1320	1732
$[26, 50]$	54	297	337
$[76, 100]$	6	83	117
$[101, 200]$	1	5	24
$[201, \infty)$	1	1	12
$[0, \infty)$	1541	2841	3357

表 4 では僅差で厳密解法が発見的解法に合計数で勝利していたが，表 5 を見てみると発見的解法で YES と答えられなかった見逃しを多く発見できていることが分かる．このことから発見的解法では見つけることのできなかった根付きクリークマイナーがまだまだ多く残されていることが分かる．

5.3　バックトラック vs 木分解上の動的計画法

次に，バックトラックアルゴリズムと木分解上の動的計画法の比較を行う．5.1 節の必要条件を満たした 54511 インスタンスに対し，各制限時間ごとにバックトラックと木分解上の動的計画法が YES と答えた数，NO と答えた数をそれぞれ表 6,7 に示す．なお，表中に現れる bt,btonly,tdp,tdponly はそれぞれバックトラックアルゴリズムが答えた個数，バックトラックアルゴリズムのみが答えた個数，木分解上の動的計画法が答えた個数，木分解上の動的計画法のみが答えた個数を表すものとする．

木分解上の動的計画法では 10ms では実行が終わりきらないものが多いが 100ms 程度かければ多くの問題が解けるようになることが分かる．逆にバックトラックは比較的早い段階で実行が終わるものは多いが，そのあと時間をかけても解けるようになる問題はそこまで多くはならない．総じて見てみると木分解上の動的計画法の方が解ける問題の数が多く，特に NO と答えたものについては顕著で，バックトラックは NO と答えるのが苦手だということが分かる．

表 6　YES

time(ms)	bt	btonly	tdp	tdponly
10	40643	12406	28646	409
100	41685	886	41774	975
1000	42156	268	42770	882
10000	42509	293	42908	692

表 7　NO

time(ms)	bt	btonly	tdp	tdponly
10	2473	1367	1286	180
100	2947	10	4337	1400
1000	3269	18	4580	1329
10000	3465	2	6434	2794

6.　終わりに

今回実験に用いたインスタンスでは，100ms 必要条件に用いることで 25%のインスタンスを削減できたため，必要条件による効果は大きかったと言える．一方，厳密解法と発見的解法について，根付きクリークマイナーが検出できた個数で比較すると，厳密解法にかかる時間が大きかったため，その優位性を強く示すことはできなかった．しかし表 4,5 より，発見的解法では YES と答えられたなかったものの中でも 100ms 程度かけるだけで $23(2841/(54511-42429)\times 100)$%もの根付きクリークマイナーが検出できた．そのため，発見的解法で NO と答えたものに対し厳密解法を使うことで，見逃した根付きクリークマイナーを検出するのが有効であると言える．

グラフサイズの大きい問題については，厳密解法では時間がかかりすぎて検出するのが難しいため，発見的解法の検出精度を上げることが肝要である．今回利用した発見的解法の精度を上げるほか，他の発見的解法を考え，元の発見的解法とを組み合わせることで根付きクリークマイナーの見逃しを少なくすることが考えられる．また，発見的解法で全ての根付きクリークマイナーを検出することは難しいので，特にグラフサイズの小さな問題で見逃したものについては厳密解法で発見するのが重要である．しかし本稿の厳密解法では 25 頂点程度で実行が終了するのに時間がかかるインスタンスが多く存在した．厳密解法の実行速度を良くするために，バックトラック，木分解上の動的計画法の両アルゴリズムについて，無駄な探索を抑えるために枝刈り条件を他にも見つけるといった改善が必要になるだろう．

参考文献

[1] Bodlaender, Hans L., and Arie MCA Koster. “Safe separators for treewidth.” Discrete Mathematics 306.3 (2006):

337-350.
[2] GitHub repository for TCS-Meiji on PACE2017 Track A submission. https://github.com/TCS-Meiji/PACE2017-TrackA.
[3] The Parameterized Algorithms and Computational Experiments Challenge. https://pacechallenge.wordpress.com/pace-2017/.