擬似クリークを列挙する多項式時間遅延アルゴリズム

宇野 毅明
国立情報学研究所 〒 101-8430 東京都千代田区一ツ橋 2-1-2 e-mail: uno@nii. jp

抄録: 与えられたグラフの密な部分構造を見つけ出す問題は，データマイニングやデータ工学を始めとする情報学の分野での基礎的な問題である．密な構造をモデル化したものとしてクリーク（完全グラフとなっている部分グラフ）があり，これはモデルの単純さと解法の構築しやすさから重宝されており，今までに多くの研究でクリーク，あるいは極大クリーク列挙が使われてきた．その一方で，近年，本来の動機であるグラフの密な構造を見つける，という問題を直接解こうという機運が高まりつつある．クリークだけでなく，クリークから枝を少し抜いたような，擬似クリークを見つけようというものである．擬似クリークの定義はいくつか考えられるが，ここでは枝数がクリークと比べて閾値以上の割合になっている部分グラフで定義する．本稿では，このように定義された擬似クリークを全て列挙する問題を考える．まず，この問題に対する分割統治法的アプローチが計算量的には絶望的であることを示し，続いて逆探索法に基づく多項式時間アルゴリズムを提案する．さらに，計算実験から実用上の効率の良さを示す.

A Polynomial Time Delay Pseudo Clique Enumeration Algorithm

Takeaki Uno
National Institute or Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, JAPAN, e-mail: uno@nii. jp

Abstract: The problem of finding dense structures of a given graph is a quite basic method in informatics including data mining and data engineering. One of a popular model to represent dense structure is a clique, which is a subgraph being a complete graph. Clique is widely used because of its simpleness and easiness to be solved. On the other hand, recently, solving the original problem of finding dense structures have arisen again. The problem is to find pseudo cliques, which are subgraphs obtained by removing small number of edges from cliques. There may be some kinds of definitions for a pseudo clique, but here we define it by a subgraph such that the ratio of whose edges compared to the clique is no less than a given threshold value. In this paper, we address the problem of enumerating all pseudo cliques for given a graph and a threshold value. We first show that it seems to be hard to obtain polynomial time algorithms by usual divide and conquer approaches. Then, we propose a polynomial time algorithm based on reverse search. We show the results of computational experiments showing efficiency of our algorithm for the practice.

1 はじめに

与えられたグラフから密な部分構造を見つけ出す問題を考えよう．つまり，枝を多く含む頂点部分誘導グラフを見つける問題である．例えば，関係ある項目の間に枝があるグラフや，似ているものが枝で結ばれているグラフでは密な構造は関係のある，似ている，あるいは活発な項目のグループを表すと考えられる．そのため，グラフから密な部分構造を見つ出す問題はデータマイニングやデータ工学を始めとする情報学の分野での基礎的な問題であり，密な構造を用いて問題を解く，という手法はとても一般的である．例えば，クラスタリング，コミュニティー発見，知識獲得，(web) 検索などで，多くの応用がある [1, 6, 8, 9, 10, 12, 13, 14, 20].

グラフの部分グラフで，完全グラフとなっているものをクリークとよぶ．クリークは密な構造を表す最も基礎的な構造である．クリークは数学的に良い性質をいくつも持ち，また，取り扱いも簡単であることから，多くの研究でクリークを発見する手法が用いられている．クリーク自体を密な部分構造として，あるいは密な構造が含むべき部分構造として扱っているのである．近年，データベース技術の向上，計算機の進歩，および列挙アルゴリズムの進展 [11, 16] により，クリーク，および極大なクリークの列挙は，ほぼ最適な時間で行えるようになった．クリーク列挙はツールとしてほぼ完成された状態にあるのである．

すると，次に，クリークよりももっと豊かなモデル化が行いたい，という要求が出てきた．確かにクリークは密な部分構造を表すが，とても疎なグラフではそれなりの大きさのクリークを含まずとも，密な構造と考えられる構造はある．また，データの不備により，本来クリークであるべき部分がクリークでなくなる可能性もあり，計算の安定性・信頼性からは，こういったエラーにも対応できるべきであろう．そのため，クリークから枝を少し抜いて得られるようなグラフ，つまり**擬似クリーク**を見つけようという機運が高まっている．

擬似クリークの定義はいくつか考えられる．１つには，ある閾値 θ を与え，クリークから θ 本以内の枝を抜いて得られるグラフ，と定義するものである．この定義のモデルとしての長所は，擬似クリークの任意の部分集合が擬似クリークになる，つまり擬似クリークの集合族が単調になる，という点である．これは，非常にきれいな性質であり，今までの研究で培われてきた単調な集合族に対する列挙手法をそのまま適用することで，効率良い列挙手法の開発が見込める．しかし，グラフの大きさが変化しても，失ってよい枝数の閾値が変化しないため，大きい部分グラフではほんの一部の枝を除去しただけで擬似クリークとならなくなる一方で，小さい部分グラフはほぼ全てが擬似クリークとなってしまうという欠点も持つ．

もう１つの定義は，枝の存在割合が閾値以上である部分グラフを用いるものである．ある部分グラフの枝数が k であり，同じ頂点数を持つクリークの枝数が m であるとき，k/m をその部分グラフの**枝密度**とよぶことにする．部分グラフの枝密度がある閾値以上であるとき，その部分グラフを擬似クリークと定義する．この定義では，擬似クリークの部分集合は擬似クリークとならないため，もはや単調性は成り立たないが，代わりに，大きさによらず同程度の密度のグラフが擬似クリークとなる，というメリットがある．小さい部分グラフの多くが擬似クリークとなる，ということも無い．

このような，含む枝の割合が大きい部分グラフは，密部分グラフ (dense subgraph) や重い部分グラフ (heavy subgraph), 最大枝部分グラフとよばれることもあるが，両者ともに最適化の言葉であり，例えば頂点数が k の部分グラフの中で最も枝数が多いものを密部分グラフとよぶ，というように定義されることが多く，最適解を示すことが多い．また，情報学の分野では「クリークのようなもの」という意味合いで用いられることが多いため，ここでは擬似クリークという異なる用語を用いている．ちなみに，頂点数 k の最大枝部分グラフを求める問題は，頂点数 k のクリークの存在判定問題を帰着できるため，NP 困難である．しかし，$k = \Theta(|V|)$ であれば，PTAS アルゴリズムが存在する [2]．また，枝に重みをつけた問題に対して，$O(|V|^{1/3-\epsilon})$ 近似アルゴリズムが知られているようである [4].

本稿では，後者の定義を採用する．与えられたグラフと閾値に対して，後者の定義で定義された擬似クリークを全て列挙する問題を扱う．この問題に対して難しさとアルゴリズム，および現実面での効率を論じていく．まず，2 節で記法を与えた後，3 節でこの問題に対する分割統治法的アプローチが計算量的には絶望的であることを，各反復で解くべき子問題が NP 完全となることを証明することで示す．続いて，4 節で擬似クリーク間に全ての擬似クリークを連結する隣接関係を定義し，逆探索法に基づく多項式時間アルゴリズムを構築する．さらに，5 節で計算実験の結果を示し，このアルゴリズムが実用上も良い性能を持つことを示す．6 節でこの論文をまとめる．

2 記法

$G = (V, E)$ を頂点集合 V と枝集合 E を持つ無向グラフとする．多重枝は持たないものとする．E の枝 e が頂点 u と v を結んでいるとき，$e = (u, v)$ と表記し，u と v は**隣接**するという．また，u と v を e の**端点**とよび，e は，u，v に**接続**するという．V の部分集合 U に対して，頂点集合を U，枝集合を両端点が U の頂点であるような E の枝の集合を U が**誘導する**グラフ，あるいは U の**頂点誘導グラフ**とよび，$G[U]$ と表記する．$G[U]$ の枝集合を $E[U]$ と表記する．また，G の部分グラフで，ある V の頂点部分集合が誘導するグラフになっているものを，単に頂点誘導グラフとよぶ.

G の頂点 v に対して，v に接続する枝の数，つまり v を端点とする枝の数を v の**次数**といい，$deg(v)$ と表記する．また，$deg_U(v)$ を v に隣接する U の頂点の数とする．また，グラフ G の頂点の中で最も大きな次数を持つ頂点の次数を G の**最大次数**といい，Δ と表記する．同様にして**最小次数**を定義する．

グラフ G の頂点部分集合 U に対して，U の誘導するグラフが完全グラフとなっている，つまり任意の U の頂点のペアが枝で結ばれているとき，U の誘導する部分グラフを**クリーク**とよぶ．この論文では，頂点

集合 U 自体もクリークとよぶ．大きさ 2 以上の頂点集合 U に誘導されるグラフに対して，その**枝密度** $p(U)$ を $|E[U]|$ / $(|U| \times (|U|-1)/2)$ で定義する．$|E[U]|$ は $G[U]$ の枝数，$(|U| \times (|U|-1)/2)$ は頂点数が $|U|$ のクリークの枝数であるため，枝密度は頂点誘導グラフの枝の存在割合となる．U が 1 つの頂点からなるとき，あるいは空集合であるときは，枝密度は 1 であると定義する．閾値 θ に対して，枝密度が θ 以上となる頂点誘導グラフを**擬似クリーク**とよぶ．U が 1 つの頂点からなるとき，あるいは空集合であるときは，定義から $G[U]$ は擬似クリークである．擬似クリークを誘導する頂点部分集合も擬似クリークとよぶ．本稿では，以下で定義される擬似クリーク列挙問題を扱う．

擬似クリーク列挙問題
入力: グラフ $G = (V, E)$，閾値 θ
出力: G の擬似クリークとなっている頂点部分集合全て

列挙アルゴリズムが，入力の大きさと出力の大きさの和に対して多項式時間で終了するとき，そのアルゴリズムは**出力多項式時間**とよばれる．特に，任意の 2 つの連続する出力の間にかかる計算時間が入力の大きさの多項式時間で抑えられるとき，**多項式時間遅延**とよばれる．多項式時間遅延であれば，計算時間は出力数に対して線形となり，出力数に対しては最適なアルゴリズムとなる．

3 分割統治法の困難性

列挙アルゴリズムを構築する際に最も良く使われる手法の 1 つが分割法である．一般的に説明すると，分割法とは，列挙すべき解の集合をいくつかの非空な集合に分割し，それぞれの分割を再帰的に分割していき列挙する，という手法である．これは，分割統治法，最適化で使われる分枝限定法と，同じアイディアに基づく解法である．擬似クリーク列挙問題のように，頂点部分集合を列挙する問題の場合，ある頂点を含むものと含まないものに解集合を分割し，それぞれを，非空でなければ子問題として再帰的に列挙する，という形で設計されることが多い．例えば全張木・パス・サイクル [15]，クリーク，安定集合 [18]，マッチング [5] など多くの部分構造がこのような形で列挙されている．

上記の方法で，擬似クリークを列挙する分割法のアルゴリズムを設計すると以下のようになる．なお，I は必ず解に含めなければならない頂点の集合，O は解に入れてはいけない頂点の集合である．

Algorithm BinaryPartition ($G = (V, E)$, I, O)
1: **If** I を含み，O の頂点を含まない擬似クリークが存在しない **then return**
2: $v := V$ の頂点のうちの 1 つ
3: **Call** BinaryPartition ($G = (V, E)$, $I \cup \{v\}$, O)
4: **Call** BinaryPartition ($G = (V, E)$, I, $O \cup \{v\}$)

このアルゴリズムのステップ 1 では，以下の判定問題を解く必要がある．

擬似クリーク存在判定問題
入力: グラフ G，頂点集合 I，頂点集合 O
出力: G に I の頂点を全て含み，O の頂点を 1 つも含まない擬似クリークが存在する場合 Yes，そうでなければ No

もし，この問題が多項式時間で解けるならば，アルゴリズムの計算量は多項式時間遅延となる．しかし，以下で述べるようにこの問題は NP 完全である．これは，このアプローチで多項式時間遅延アルゴリズムを構築することは，$P \neq NP$ であれば不可能であるということを示し，出力多項式時間アルゴリズムを得るのも難しいであろうという観察を与える．

定理 1 擬似クリーク存在判定問題は，たとえ O が空集合である場合でも NP 完全である．

Proof: 証明は，与えられたグラフ $G'=(V',E')$ が大きさ k のクリークを含むかどうかを判定する問題を帰着することで行う．この問題は NP 完全である [7].

グラフ G' に対して，グラフ $G=(V,E)$ を以下のようにして構築する．まず，頂点集合 V を，V' に大きさ $2|V|^2$ の頂点集合 U を加えたもの，つまり $V=V'\cup U$ とする．U から任意の頂点を 2 つ選び z_1,z_2 とする．枝集合の構築は，まず $E=E'$ とし，続いて V の各頂点 v に対して z_1,z_2 以外の U の頂点と v を結ぶ枝を張る．V の各頂点 v に対して，v と U の頂点のペアが $2|V|^2$ 存在するが，そのうちの $(|V|^2-1)/|V|^2$ に枝が張られている．また，U の頂点間に $(2|V|^2-1)\times(|V|^2-1)$ 本の枝を任意に張る．$G[U]$ の枝密度がちょうど $(|V|^2-1)/|V|^2$ となることを注意されたい．この結果，V の任意の頂点集合 S に対して，$G[U\cup S]$ の $G[S]$ に含まれない枝の含有割合を考えると，$(|V|^2-1)/|V|^2$ となっている．

S を V の部分集合とする．このとき，$G[S]$ がクリークであるとき，またそのときに限り $G[S]$ の枝密度は $(|V|^2-1)/|V|^2$ より大きくなる．よって，$G[U\cup S]$ の枝密度も，S がクリークであるときのみ，$(|V|^2-1)/|V|^2$ より大きくなり，その値は S の大きさにより一意に決まる．S が空集合，あるいは 1 つの頂点からなるときのみ，枝密度は $(|V|^2-1)/|V|^2$ と等しく，その後は S の大きさの増大に伴い，単調に大きくなる．S の大きさが k である場合の枝密度を式で書き下せば，

$$\frac{k(k-1)/2+(2|V|^2-1)\times(|V|^2-1)+k(2|V|^2-2)}{k+|V|^2}$$

となる．この値を $\theta(k)$ と表記する．V の頂点部分集合 S が大きさ k 以上のクリークであるとき，またそのときに限り，$G[U\cup S]$ の枝密度は $\theta(k)$ と同じか大きくなる．よって，大きさ k のクリークの存在を判定する問題は，擬似クリーク存在判定問題に帰着され，擬似クリーク存在判定問題は NP 完全である．∎

4 逆探索による多項式時間列挙

前節で示したとおり，良く使われる分割統治法によるアプローチは，擬似クリークに対して効果的でない．しかし，擬似クリーク自体は見つけにくいものではなく，ある種の隣接性を持ち，連結になっている．本節では，この連結性を示し，連結性を用いて効率良く列挙を行う，逆探索に基づいたアルゴリズムを提案する．まず，以下の補題により，擬似クリークの連結性を示す．

補題 1 頂点集合 U に対して，$G[U]$ での次数が最も小さい頂点の 1 つを v とする．このとき，$U\setminus\{v\}$ の枝密度は U の枝密度より同じか高い．

Proof: U の頂点のペアを 2 つのグループ F_1，F_2 に分ける．F_1 は v と v 以外の頂点のペア，F_2 は v 以外の頂点のペアである．また，$E_1=E[U]\cap F_1$，$E_2=E[U]\cap F_2$ とする．このとき，U の枝密度 $p(U)$ は，$|E_1|/|F_1|$ と $|E_2|/|F_2|$ の間の値をとることがわかる．$p(U)\times(|U|-1)$ は グラフ $G[U]$ の平均次数となるため，v の選び方より $|E_1|/|F_1|=deg_U(v)/(|U|-1)\le p(U)$ となる．上述の観察より，これは $G[U\setminus\{v\}$ の枝密度が $G[U]$ のそれよりも同じか大きいことを意味する．∎

以下，擬似クリークは頂点集合を指すものとする．頂点部分集合 $U\neq\emptyset$ に対して，頂点間の全順序 $\prec_U$ を，

$$u\prec_U v \quad\Leftrightarrow\quad deg_U(u)<deg_U(v) \ \text{ or } \ (deg_U(u)=deg_U(v) \ \text{ and } \ u\le v)$$

と定める．$v^*(U)$ を，U の頂点 v で，他の任意の頂点 u に対して $v\prec_U u$ が成り立つものとする．今，擬似クリーク $K\neq\emptyset$ に対して，K の**親**を $K\setminus\{v^*(K)\}$ で定義する．K の親が K' であるとき，K を K' の**子ども**とよぶ．上記補題 1 から，擬似クリークの親は擬似クリークである．また，親は子どもより頂点数が 1 小さい．よって，この親子関係が導出するグラフ，つまり擬似クリークの集合を頂点集合とし，親子の間に枝を張って得られるグラフは，空集合を根とする根付き木になる．この根付き木を深さ優先探索すれば，全ての擬似クリークを列挙することができる．このようにして列挙を行う手法は逆探索と呼ばれる [3]．深さ優先探索を行うには，子どもを列挙するアルゴリズムがあればよい．各擬似クリークに対してその子どもを列挙し，各子どもに対して再帰呼び出しを行って，その子孫を再帰的に列挙すればよい．その子どもの子孫を列挙した後は，次の子どもの子孫の列挙を行うのである．

子どもの列挙が１つあたり多項式時間でできれば，この逆探索アルゴリズムは多項式時間遅延となる．実際，擬似クリーク K の子どもは K に頂点を１つ加えて得られるため，K に各頂点を追加して子どもの候補を作り，候補が擬似クリークであり，かつその親が K であるかを確認する，という作業で全ての子どもを列挙できる．$K \cup \{v\}$ の親は，$v^*(K \cup \{v\})$ を求めることで得られ，これにかかる時間は $O(|E|)$ である．よって，各反復での計算時間は高々$O(|V||E|)$ となり，再帰の深さが $|V|$ を超えないことを合わせれば，多項式時間遅延アルゴリズムとなることが確認できる．以下に，この逆探索アルゴリズムを示す．

Algorithm EnumPseudoClique ($G = (V, E)$, K)
1: **Output** K
2: **For each** $v \notin K$ **do**
3: **If** $K \cup \{v\}$ が K の子ども **then** EnumPseudoClique ($G = (V, E)$, $K \cup \{v\}$)

実際には，これらの作業は $deg_K(v)$ の値がわかれば，短時間で行える．$K \cup \{v\}$ が擬似クリークであるためには，枝数がある数以上追加されなければならない．そのため，$deg_K(v)$ がある数以上であることが，$K \cup \{v\}$ が擬似クリークであるための必要十分条件となり，全ての頂点を $deg_K(v)$ の大きさによって分類し，大きさ順に非空なものを連結したリストで保管することで，$K \cup \{v\}$ が擬似クリークとなるような v は１つあたり定数時間で見つけられる．v を K に追加した際の deg_K の更新は，$O(deg(v))$ 時間で行える．

$K \cup \{v\}$ が擬似クリークである場合，もし $v = v^*(K \cup \{v\})$ ，つまり $G[K \cup \{v\}]$ の中で v が最小次数であり，かつ最小次数頂点の中で最小添え字であれば，$K \cup \{v\}$ は K の子どもとなる．この条件を，$deg_K(v)$ の大きさによって場合分けして考えると，以下のようになる．$d^*(K)$ を $G[K]$ の最小次数とする．

・ $deg_K(v) < d^*(K)$ の場合： $v = v^*(K \cup \{v\})$ となる．
・ $deg_K(v) > d^*(K) + 1$ の場合： $v = v^*(K \cup \{v\})$ となる．
・ それ以外の場合： $u < v$ かつ $deg_K(u) < deg_K(v)$ である $u \in K$ が存在せず、かつ $u \prec_K v$ が成り立つ任意の $u \in K$ が v に隣接していれば，$v = v^*(K \cup \{v\})$ となる．

$deg_K(v)$ の大きさで頂点を分類しておけば，ここでも計算の効率化が行える．3 番目の条件のチェックは，単純に行うと全ての頂点に対するチェックの時間が $O(|E|)$ となるが，以下の方法を使うことで，チェック時間を $O(d^*(K) \times \Delta)$ とすることができる．なお，K の頂点を $\prec_K$ の順に最初から $d^*(K)$ 個並べた列とする．
・ S の頂点 u を添え字の小さい順に選ぶ
・ 各頂点の印を nil とする
・ u が隣接する各頂点 w について，w の印が u 前の頂点であれば，印 u をつける．u が S の最初の頂点である場合は無条件でつける．
・ 印を更新した各頂点の印を，「S の中で現在の印の次にある頂点」に変更する

この操作が終了した後，各頂点には「隣接しない S の頂点で最も前にある頂点」が印として付く．そのため，チェックが単位時間で行えるようになる．$v^*(K)$ の最小添え字頂点よりも小さい添え字を持つものに関しては $deg_K(v) = d^*(K)$ である場合，S の頂点に隣接せずとも子どもを生成するが，この場合，必ず子どもになるため，チェックの必要はない．よって，チェックを行う必要のある頂点は，S のある頂点に隣接しているもののみである．上記のチェックを効率的に行うためには $deg_K(v)$ が等しい頂点のグループが添え字順に並べる必要があるが，これは保持する際に並び替えておくものとする．すると，データの更新の時間が $O(\Delta \log |V|)$ となるが，K に関する反復の計算時間は $O(\Delta(d^*(K) + \log |V|) + (K$ の子どもの数$))$ となる．(子どもの数) に関しては，その計算時間は各子どもが消費している，とみなせば，これは，$O(\Delta(\Delta + \log |V|)$ となる．よって，擬似クリークは１つあたり $O(\Delta(\Delta + \log |V|)$ 時間で列挙できることとなる．また，深さが奇数の反復では，再帰呼び出しの前に解の出力を行い，深さが偶数の場合は再帰呼び出しの後に解の出力を行う，という改良を行うと，3 つの連続する反復のうち必ず１つは解を出力するようになり，遅延時間もこの計算量となる．

定理 2 与えられたグラフ $G = (V, E)$ と閾値 $0 \leq \theta \leq 1$ に対して，G に含まれる全ての擬似クリークは $O(\Delta(\Delta + \log |V|)$ 時間遅延で列挙できる．

実際には，この計算量は現実の計算時間よりも大きく見積もりすぎている．現実的な応用問題では，疎なグラフから，あまり大きくない，小さめの擬似クリークを列挙する，という状況が多いと考えられる．そうでないと，擬似クリークの数が爆発してしまい，とても列挙できない．そのため，$d^*(K)$ は通常とても小さいと考えられる．また，$G[K]$ の最小次数頂点が G で大きな次数を持つ場合も考えがたく，この点でも計算量は，実際の計算時間と乖離すると考えられる．

5 計算実験

本節では，本稿で提案するアルゴリズムの，実際の性能を見るべく，計算実験の結果を示す．プログラムコードは C で組まれ，計算機は Pentium M 1.1 GHz，メモリ 256MB を搭載したノート PC，OS は Windows 上で動く Linux シミュレータである cygwin を用いた．実装したプログラムは上記のアルゴリズムをそのままコード化したものだが，子どものチェックに関する計算量の改良を行っていない．これは，現実問題は疎である場合が多いと思われるため，$deg_K(v) = d^*(K), d^*(K)+1$ となる頂点の数は非常に少ないと考えたためである．実際，計算実験でも非常に少なく，平均的にも高々10 個程度であった．

実験には，いくつかの方法でランダムに生成したグラフと，現実問題のグラフを使用した．１つ目は通常のランダムグラフで，全ての枝を等しい確率で発生させるものである．応用上，疎なデータが扱われることが多いだろうという観測から，確率は 0.1 とした．しかし，ランダムグラフは全体が均一で局所性がまったく無いため，現実のデータとは特性が大きく異なる．そこで，2 つ目として，局所的に密なグラフを用意した．これは，全ての頂点を並べた輪っか状の列を考え，各頂点から前後に r 個以内離れている頂点に対して等しい確率で枝をはるというものである．遠距離の頂点と結ばれていないことから，任意の部分が局所的であり，かつ密である，という意味で非常に均質である．実験では，$r = 20$，確率は 0.5 とした．3 つ目は，グラフの頂点次数の構造がパワー則（あるいは zip law，スケールフリー）となるように生成したランダムグラフである．パワー則とは，ある定数 Γ に対して，値が λ である確率が $1/\lambda^{\Gamma}$ であるような分布のことを指し，頂点の次数の分布がおおよそこのような分布になっているグラフは，スケールフリーグラフとよばれる．現実問題に現れるグラフの多くがスケールフリーグラフであることが知られている．スケールフリーグラフを生成する方法として，頂点を逐次的に追加していく方法が知られている．まず，頂点数が k であるクリークを出発点とし，頂点を１つずつ追加していく．各頂点を追加する際に，グラフの頂点から k 個を次数に比例した確率で選び，それらに対して枝をはる．このようにして作られたグラフはスケールフリーグラフになるが，現実問題が持つような局所的に密な構造は持たないため，クリークの平均的な大きさが現実問題より小さいようである．

それぞれの発生方法で頂点数を変化させて，いくつかのグラフを生成し，擬似クリークを，閾値を 0.9，0.8 とした場合の実験を行い，擬似クリーク数・計算時間・解 100 万個あたりの計算時間を計測した．既存の研究に，このような列挙問題を解き，実装を公開しているものが無いため，既存研究との比較は行わない．代わりに，安定した性能を示しているクリーク列挙アルゴリズムとの比較を行う．結果を以下のグラフに示す．

図 1 左が通常のランダムグラフに対する結果であり，横軸が頂点数，縦軸が計算時間とクリーク・擬似クリークの総数である．軸は両者とも対数である．図 1 右は局所的に密なランダムグラフ，図 2 左はスケールフリーグラフである．

どのグラフでも，上部右上がりの 3 本の線が擬似クリーク数とクリーク数である．閾値を下げるに従い，解の総数は増加している．ランダムグラフの場合，平均次数が大きくなることが原因と思われるが，３つの総数の比が頂点数の増加とともに大きくなっている．局所的に密なグラフでは一定であるが，スケールフリーグラフの場合，逆に頂点数が大きくなると比が小さくなっている点が興味深い．

擬似クリーク１つあたりの計算時間は，閾値によらずほぼ同じであり，また，クリーク列挙に比べてそれほど長いわけではない．クリーク列挙はほぼ最適な時間で行えるため，これは擬似クリーク列挙アルゴリズムが，実用上，十分最適に近い性能を発揮できるということを示しているだろう．

しかし，ランダムグラフにおいては，次数が大きくなるにつれ非常にゆっくりと増加し，スケールグラフにおいては大きく増加している．これは，擬似クリークの子どもの候補の数に対する実際の子どもの割合が減少することが原因と思われる．閾値が小さいと $deg_K(v) = d^*(K)$ を満たす頂点の数が増加し，子どもの候補の数が増えるが，スケールフリーグラフは局所性が高いため，このうちの一部しか子どもにならないものと思われる．

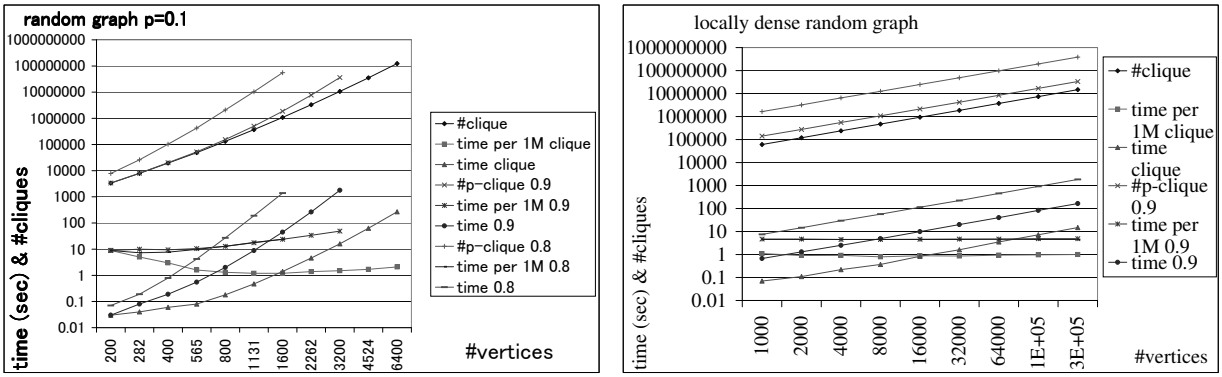

図 1: ランダムグラフ (左)，局所的に密なランダムグラフ (右) での計算結果

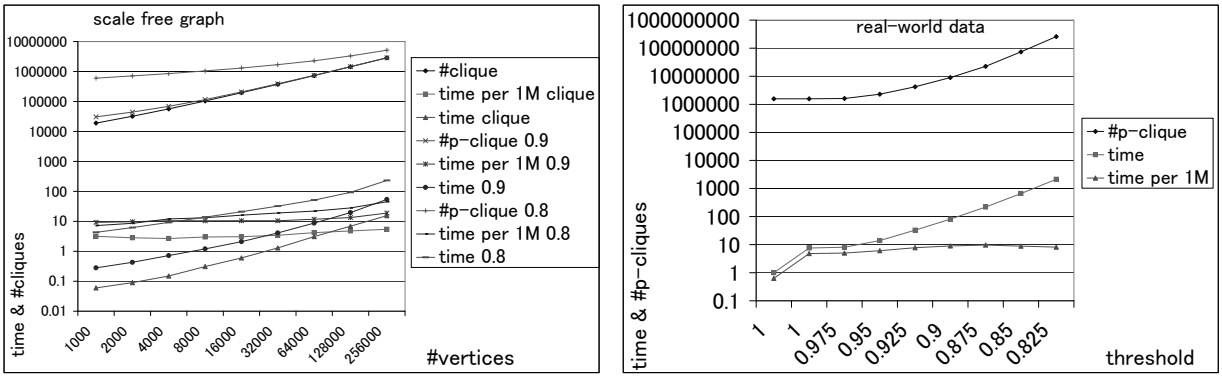

図 2: スケールフリーグラフ (左)，論文共著グラフ (右) に対する計算結果

次に，現実のデータで行った計算の結果を示す．用いたデータは論文の共著関係を表すネットワークで，頂点は人物を表し，枝は共著のある人物の間を結んでいる [19]．頂点数はおよそ 3 万，枝数は 12 万 5 千であり，スケールフリーである．頂点数を変化させて挙動を見ることができないため，このグラフでは閾値のみを変化させた．一番左の列はクリーク列挙の計算時間を示しており，擬似クリーク列挙がクリークに近い時間で列挙できていることがわかる．また，閾値の変化は，計算時間に対してあまり変化を与えないように見られた．

6 まとめ

本稿では，グラフから密な部分グラフを見つけ出す問題を扱った．頂点数が同じであるクリークに対する枝数で，部分グラフの枝密度を定義し，枝密度が閾値以上である部分グラフを擬似クリークと定義し，擬似クリークを全て見つける問題に取り組んだ．まず，この問題を通常の分割統治法によるアプローチで解いた場合，子問題に NP 困難である問題が現れることを示し，このアプローチで多項式時間アルゴリズムを構築することは容易ではないことを示した．続いて，任意の擬似クリークに対して，その擬似クリークに含まれ，頂点数がちょうど 1 だけ小さい擬似クリークが必ず存在することを示し，これから導かれる隣接関係を用いた探索により，1 つあたり $O(\Delta(\Delta + \log |V|)$ 時間であり，遅延時間もこれと等しいアルゴリズムを提案した．そして，計算実験により，実用上も最適に近い計算時間で擬似クリークが列挙できることを示した．

擬似クリークは，クリークを豊かにしたモデルとして注目されつつあるが，クリークより数が多くなる傾向があるという欠点もある．クリークの場合は極大クリークを見つけることでその解数を上手に減少させることができるが，擬似クリークは単調性を持たないため，極大元の定義が難しく，また効率的な列挙も難しいと思われる．この課題を解消するモデルの考案と，新たなアルゴリズムの構築が今後の課題となるであろう．

謝辞

当研究の一部は文部科学省科学研究補助費によって行われた.

参考文献

[1] J. Aslam, K. Pelekhov, and D. Rus, “A Practical Clustering Algorithms for Static and Dynamic Information Organization,” *ACM-SIAM Symposium on Discrete Algorithms*, ACM Press, (1999).

[2] S. Arora, D. Karger, and M. Karpinski, “Polynomial time approximation schemes for dense instances of NP-hard problems”, *Proc. 27th Ann. ACM Symp. on Theory of Comp.*, ACM, pp. 284–293 (1995).

[3] D. Avis and K. Fukuda, “Reverse Search for Enumeration”, *Discrete Applied Mathematics* **65**, pp. 21–46, (1996).

[4] U. Feige, G. Kortsarz, and D. Peleg, “The dense k-subgraph problem”, Unpublished manuscript, (1995),

[5] K. Fukuda and T. Matsui, “Finding All Minimum-Cost Perfect Matchings in Bipartite Graphs”, *Networks* **22**, pp. 461–468 (1992).

[6] K. Fujisawa1, Y. Hamuro, N. Katoh, T. Tokuyama, and K. Yada, “Approximation of Optimal Two-Dimensional Association Rules for Categorical Attributes Using Semidefinite Programming”, *Lecture Notes in Computer Science* **1721**, pp. 148–159 (1999).

[7] M. R. Garey, D. S. Johnson, and L. Stockmeyer, “Some Simplified NP-complete Problems”, *Annual ACM Symposium on Theory of Computing, Proceedings of the sixth annual ACM symposium on Theory of computing*, pp. 47–63, (1974).

[8] D. Gibson, R. Kumar and A. Tomkins, “Discovering Large Dense Subgraphs in Massive Graphs”, *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 721–732 (2005).

[9] H. Hu, X. Yan, Y. Huang, J. Han and X. J. Zhou, “Mining Coherent Dense Subgraphs Across Massive Biological Networks for Functional Discovery”, *Bioinformatics* **21**, pp. 213–221 (2005)

[10] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins “Extracting Large-Scale Knowledge Bases from the Web”, *Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 639–650 (1999).

[11] K. Makino and T. Uno, “New Algorithms for Enumerating All Maximal Cliques,” *SWAT 2004*, *Lecture Note in Computer Science* **3111**, pp. 260–272 (2004).

[12] 大久保好章, 原口誠, “疑似クリーク探索によるクラスター抽出”, 人工知能学会全国大会 (2005).

[13] 大久保好章, 原口誠, “擬似クリークに基づく近似的形式概念の抽出”, 人工知能学会全国大会 (2006).

[14] G. Palla, I. Derenyi, I. Farkas, T. Vicsek, “Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society,” *Nature*, **435** 7043, pp. 814–818,(2005)

[15] R. C. Read and R. E. Tarjan, “Bounds on Backtrack Algorithms for Listing Cycles, Paths, and Spanning Trees,” *Networks* **5**, 237-252 (1975).

[16] E. Tomita, A. Tanaka, and H. Takahashi, “The Worst-case Time Complexity for Generating All Maximal Cliques,” *COCOON 2004*, *Lecture Note in Computer Science* **3106**, pp. 161–170 (2004).

[17] T. Uno, “Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs”, *ISAAC 97*, *Lecture Note in Computer Science* **1350**, pp. 92–101 (1997).

[18] 宇野 毅明, “大規模グラフに対する高速クリーク列挙アルゴリズム”, 電気通信学会コンピュテーション研究会 (2003).

[19] S. Warner, “E-prints and the Open Archives Initiative,” *Library Hi Tech* **21**, pp.151–158 (2003).

[20] Y. Zhang, C. H. Chu, X. Ji, H. Zha, “Correlating Summarization of Multisource News with k Way Graph Biclustering”, *ACM SIGKDD Explorations Newsletter archive* **6**, pp. 34–42 (2004).