Polynomial-time Extraction of the Maximum Clique Using Eigenvalue Relation

Research · December 2020

1 author:

Yasunori Ohto

6 PUBLICATIONS **15** CITATIONS

SEE PROFILE

Polynomial-time Extraction of the Maximum Clique Using Eigenvalue Relation

Yasunori Ohto, Independent
yasunori_ohto.asp8@dg7.so-net.ne.jp

2020 年 12 月 23 日

概要

本論文では, NP 完全問題の一つである k-Clique 問題の最適化バージョンである, 最大クリーク問題が多項式時間で解けることを証明する. まずはじめに, Input graph を Weighted graph S へ拡張する. ここで, S の Adjacency matrix における重複を含む -1 以下の固有値の個数を $\mathrm{k_m}(S)$ とすると, 最大クリークの頂点数は $\mathrm{k_m}(S)+1$ 以下となる. 次に, 重み $1/2$ の辺を適宜加えることで, 頂点削除時のグラフスペクトルを制御した, Weighted graph 集合を導入する. そして, この集合上において, 最大クリークサイズが変化しない頂点を削除していくことで, 最大クリークを一つ抽出する. このアルゴリズムの計算量は $O(n^8)$ となる. よって, NP 完全問題の一つが多項式時間で解けることから, Complexity class NP と P が等しいとなる.

1 Introduction

本論文では, NP 完全問題の一つである k-クリーク問題の最適化バージョンである, 最大クリーク問題が多項式時間で解けることを証明する. これにより, Complexity class NP と P が等しいことを示す.

従来, アルゴリズムの改良を行うことにより, 最大クリーク問題の計算量の下界を求めるアプローチが取られていた. これは, 計算量を $O(2^{f(n)})$ と仮定した上で, 木探索時のバックトラッキングのショートカットを効率的に行うことによる, $f(n)$ の下限を求める手法である [5]. しかし, アルゴリズムを複雑にすることでその計算量が小さくなるものの, その下限は不明だった. これに対して, グラフの隣接行列における固有値からグラフの性質を探る, グラフスペクトル理論 [7] がある. その一つとして, 対称行列とその主部分行列の間の固有値の関係を述べた Cauchy Interlacing Theorem を用いた研究がある [2]. しかしながら, グラフスペクトルを用いて NP 完全問題を多項式時間で厳密に解こうとする研究は存在しなかった. この理由は, 固有値が実数であり, オートマトンで正確に扱うことが不可能なためである. このことから, 我々はアルゴリズムで厳密に扱うことのできる, 固有値の個数を用いる方法を開発した. 本論文では, 最大クリークの一つを多項式時間で抽出する上で必要な証明と, 最大クリーク抽出アルゴリズムを擬似コードの形で与える.

本アルゴリズムの概要を以下に示す. まずはじめに, Input graph を辺に有理数の重みを持つ Weighted graph $S \in \mathscr{S}$ へ拡張する. ここで, S の Adjacency matrix A における重複を含む -1 以下の固有値の個数を $\mathrm{k_m}(S)$ とすると, Cauchy Interlacing Theorem により最大クリークの頂点数 k は $\mathrm{k_m}(S)+1$ 以下となる. このときの $\mathrm{k_m}(S)$ の計算量は $O(n^3)$ となる. 次に, 重み $1/2$ の辺を適宜加えることで, 頂点削除時のグラフスペクトルを制御した, Weighted graph 集合を導入する. そして, この集合上で T の最大クリークサイズが変化しない頂点を削除していくことで, 最大クリークを一つ抽出する. このアルゴリズムの計算量は $O(n^8)$ となる. 以上より, NP 完全問題の一つが多項式時間で解けることとなり, Complexity class NP と P が等しいことが示される.

本論文の残りの部分では, Section 2 において, 本論文で使用する定義を示す. Section 3 において, 最大クリークの一つを多項式時間で抽出する上で必要な証明と, 最大クリーク抽出アルゴリズムを擬似コードの形で与える. 最後に, Section 4 において, 本論文のまとめを行う.

2 Definition

この Section では, 本論文において用いる定義を示す.

Definition 2.1. 集合 V と V^2 の部分集合 E の組 (V, E) を Graph G とする. 有限集合 V の元を頂点とし, 頂点の個数を Graph のサイズを $0 < n = |V|$ とする. なお, V を $\{v_1, \ldots, v_n\}$ と整列しておく. 一方, 集合 E の元として, 頂点の組 (v_a, v_b), $v_a, v_b \in V$ を Graph の辺とする. また, 辺は向き付けされておらず, 多重辺を持たないとする. なお, $(v_a, v_a) \notin E$ とすることで, Graph G は一つの辺の両端が単一頂点となるループ辺を持たないとする.

Definition 2.2. Graph $G = (V, E)$ において, 関数 $w : E \to \mathbb{Q}_+$ を Graph G 上の辺の重みとする. これにより, Graph $S = (V, E, w)$ を Weighted graph とし, S の集合を $\mathscr{S}$ とする.

Definition 2.3. Weighted graph $S = (V, E, w)$, $n = |V|$ において, 次を満たす $n \times n$ の対称行列 A を S の Adjacency matrix を A とする. Matrix A の i 行, j 列の要素を $a_{i,j}$ とする. 頂点を $v_i, v_j \in V$, $v_i \neq v_j$ とする.

$$\begin{cases} (v_i, v_j) \in E & \text{if} \quad a_{i,j} = a_{j,i} = w((v_i, v_j)), \\ (v_i, v_j) \notin E & \text{if} \quad a_{i,j} = a_{j,i} = 0. \end{cases}$$

Definition 2.1 より, $(v_i, v_i) \notin E$ であることから, $a_{i,i} = 0$ となる. また, Definition 2.2 より, 関数 $w : E \to \mathbb{Q}_+$ となることから, A の要素は非負の有理数となる. よって, 対角成分が 0 となる有理数の非負値対称行列と Weighted Graph S はグラフ同型を除いて一対一関係となる.

Definition 2.4. $n \times n$ 正方行列 M において, 任意の $n - m$ 個の $i, \ldots, j$ 番目の行と列を除去した Submatrix を M の Principal submatrix P とする.

Definition 2.5. Weighted graph $S = (V, E, w)$ における Induced subgraph $S_i = (V_i, E_i, w_i)$ は, 頂点 $v_a, v_b \in V_i \subset V$ において, $(v_a, v_b) \in E$ のとき, かつそのときに限り, $(v_a, v_b, w_i) \in E_i$ とする. よって, S, S_i の Adjacency matrix をそれぞれ A, A_i とすると, S_i は S の Principal submatrix となる.

Definition 2.6. サイズ $1 < n$ の Weighted graph $S = (V, E, w)$ において, $E = V^2$ かつ任意の辺 $e \in E$ において $w(e) = 1$ となるとき, S を Complete graph K_n とする.

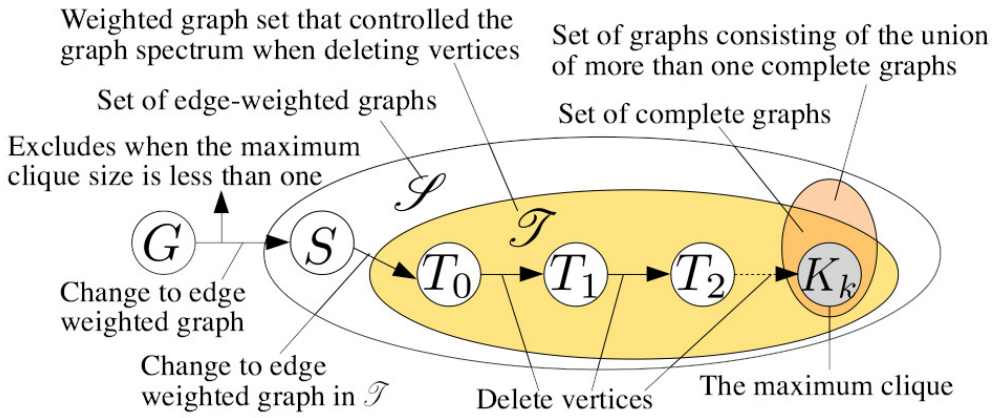

図 1　最大クリークを抽出する流れ.
まずはじめに, Input graph G から, 少なくとも 1 つの辺 e, $w(e) = 1$ を持つ Weighted graph S を作成する. 次に, S に重み付き辺を挿入することで, 頂点削除時の $\mathrm{k_m}(S)$ の変化を制御した, Weighted graph $T \in \mathscr{T} \subset \mathscr{S}$ へ変換する. 次に, 最大クリークサイズが変化しない頂点集合を削除していくことで, 最大クリークを一つ抽出する.

Definition 2.7. Weighted graph $S = (V, E, w)$ において, 頂点集合 $C \in V$ がクリークを形成するとは, C が誘導する S の Induced subgraph が Complete graph となることをいう. ここで, クリークを構成する頂点数をクリークサイズとする. なお, 複数存在するクリークにおいて, クリークサイズが最も大きいものを最大クリークと呼ぶ. このことから, 最大クリークは一つ以上存在する.

3　Proof

この Section では, 最大クリークの一つを多項式時間で抽出するアルゴリズムを構成する上で必要な証明を行ったのち, 当該アルゴリズムを擬似コードの形で与える.

Figure 1 において, 本アルゴリズムの戦略を示す. 本 Section では, この戦略が成立することを示す証明を行う. まずはじめに, Section 3.1 において, Input graph G から本論文におけるクリーク抽出対象である, 少なくとも 1 つの辺 e, $w(e) = 1$ を持つ Weighted graph S を作成する方法を示す. 次に, Section 3.2 において, S の最大クリークサイズを k として, S の Adjacency matrix A における重複を含めた -1 以下の固有値の個数 $\mathrm{k_m}(S)$ が $k \leq \mathrm{k_m}(S) + 1$ となること, $\mathrm{k_m}(S)$ が多項式時間で求まることを証明する. 次に, Section 3.3 において, S に重み $1/2$ の辺を適宜加えることで頂点削除時のグラフスペ

クトルを制御した, Weighted graph 集合 $\mathscr{T} \subset \mathscr{S}$ を導入する. 次に, Section 3.4 において, Weighted graph $T \in \mathscr{T}$ に対して, 最大クリークサイズが変化しない頂点集合が求まることを証明する. 最後に, Section 3.5 において, Input graph G に対して, 最大クリークサイズを変化させない頂点を同定したのち, これを削除する擬似コードを構成することで, 擬似コードの計算量が多項式時間となることを証明する.

3.1 Prepare

本 Section では, Input graph $G = (V, E)$ から Weighted graph S を作成する方法を示す. ここで, S は少なくとも 1 つの辺 e, $w(e) = 1$ を持つとする.

Input graph のサイズ $n = |V|$ が 0 のとき, $\emptyset$ を出力して終了する. $n \neq 0$ かつ $E = \emptyset$ のとき, Input graph G は独立頂点集合となることから, 最大クリークは任意の頂点となる. よって, 任意の頂点 $v \in V$ を取得したのち, $\{v\}$ を出力して終了する. アルゴリズムが終了しなかった時点で, $E \neq \emptyset$ となる. よって, G は少なくとも 1 つの辺を持つことになる. 以上より, 全ての辺 $e \in E$ において $w(e) = 1$ として, Weighted graph $S = (V, E, w)$ が残ることとなる.

3.2 Eigenvalue relation

この Section では, Weighted graph S の最大クリークサイズを k として, S の Adjacency matrix における重複を含めた -1 以下の固有値の個数 $\mathrm{k_m}(S)$ が $k \leq \mathrm{k_m}(S) + 1$ となることと, $\mathrm{k_m}(S)$ が多項式時間で求まることを証明する.

Proposition 3.1. サイズ $1 < n$ の Complete graph K_n の Adjacency matrix A における固有値は, 重複を含めると, 固有値 -1 が $n-1$ 個, 固有値 $n-1$ が 1 個となる.

Proof. Adjacency matrix A は正方実数行列であることから, その固有値 λ は, 行列式を $|A|$, 対角成分が 1, 他の成分が 0 となる単位行列 I を用いて, $|A - \lambda I| = 0$ を満たす実数となる. Adjacency matrix A は Definition 2.6 により対称行列となる. Adjacency matrix A はサイズ $1 < n$ であったことから, 重複を含めると n 個の実数からなる固有値を持つ. Adjacency matrix A は Definition 2.6 により, 対角成分が 0, その他の成分が 1 となる対称行列である. $\mathrm{rank}(A)$ を A の一次独立な行ベクトルの最大本数とする. このとき, 固有値 λ の重複度は $n - \mathrm{rank}(A - \lambda I)$ となる. ここで, -1, $n-1$ を $|A - \lambda I|$ に代入すると, いずれも 0 となる. また, $n - \mathrm{rank}(A - \lambda I)$ は $n-1$, 1 となることから, A

は 固有値 -1 を $n-1$ 個, 固有値 $n-1$ を 1 個持つ. 得られた固有値の数は, 重複を含めて n 個であり, これ以外の固有値を持たない. よって, Proposition 3.1 が成立する. □

行列 A を Weighted graph S の Adjacency matrix とする. Weighted graph S の最大クリークサイズを k とする. また, A の固有値のうち, -1 以下の重複を含めた個数を $\mathrm{k_m}(S)$ とする. このとき, k と $\mathrm{k_m}(S)$ の関係を示すために, 次の Theorem を用いる.

Theorem 3.2. (Cauchy Interlacing Theorem,[2]): A をサイズ n の実対称行列, B をサイズ $m < n$ の実対称行列とする. A の固有値を重複を含めて $\lambda_1 \geq \cdots \geq \lambda_n$ とする. 同様に B の固有値を $\mu_1 \geq \cdots \geq \mu_m$ とする. もし, B が A の Principal submatrix であれば, 任意の $0 < i \leq m$ において以下の Inequation が成立する.

$$\lambda_i \geq \mu_i \geq \lambda_{n-m+i}. \tag{1}$$

この Theorem から以下が成立する.

Lemma 3.3. サイズ $1 < n$ の Weighted graph S における Adjacency matrix A の最大固有値は 0 以上となる.

Proof. Weighted graph $S_0 = S$ とし, Weighted graph S_{i-1}, $0 < i < n$ から任意の頂点一つとこれに隣接する辺集合を削除した Weighted graph を S_i とする. よって, Weighted graph S_i は S_{i-1} の Induced subgraph となる. Weighted graph S_j, $0 \leq j < n$ における Adjacency matrix を A_j とする. これより, Adjacency matrix A_i は A_{i-1} の Principal matrix となる. A_{i-1} の固有値を $\lambda_1 \geq \cdots \geq \lambda_{i-1}$ とする. A_i の固有値を $\mu_1 \geq \cdots \geq \mu_i$ とする. Adjacency matrix A_j の最大固有値を ν_j とする.

S_{n-1} は頂点が一つの Weighted graph となることから, Adjacency matrix A_{n-1} の固有値は一つであり, その値は 0 となる. よって, A_{n-1} の最大固有値は $\nu_{n-1} = 0$ となる. Inequation 1 より, $\nu_{n-1} = 0 \leq \cdots \leq \nu_i \leq \cdots \leq \nu_0$ となる. この Inequation は Weighted graph S_{i-1} のとり方に依存しない. 以上より, Lemma 3.3 が成立する. □

Lemma 3.4. サイズ $1 < n$ の Weighted graph $S = (V, E, w)$ に対して, 任意の頂点 $v \in V$ とこれに隣接する辺集合 $E_a \subset E$ を除去した Weighted graph を $S' = (V', E') = (V - \{v\}, E - E_a, w)$ とする. このとき, $\mathrm{k_m}(S') = \mathrm{k_m}(S)$ もしくは, $\mathrm{k_m}(S') = \mathrm{k_m}(S) - 1$ が成立する.

Proof. Weighted graph S' は S の Induced subgraph となる. よって, S' の Adjacency matrix A' は S の Adjacency matrix A の Principal submatrix となる. A の固有値

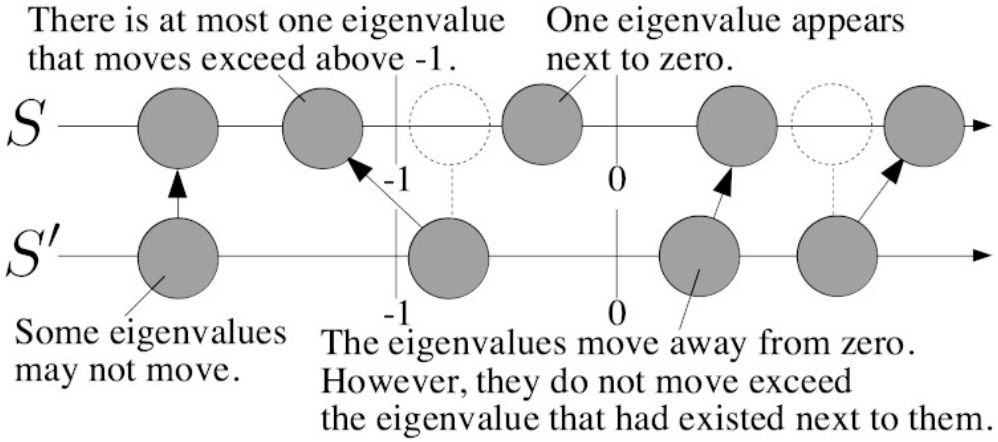

図 2　Weighted graph $S = (V_i, E_i, w)$ から任意の頂点 $v \in V$ と, これに隣接する辺を削除した Weighted graph S' におけるスペクトルの関係.
Weighted graph S' から S をみたとき, 全ての固有値はその場に留まるか, 0 から離れるように移動する. 固有値の移動において, 隣接する固有値を超えることはない. よって, -1 を超えて小さくなる固有値はたかだか一つとなる.

を $\lambda_1 \geq \cdots \geq \lambda_n$ とする. A' の固有値を $\mu_1 \geq \cdots \geq \mu_{n-1}$ とする. Lemma 3.3 より, Weighted graph A' の最大固有値は $0 \leq \mu_1$ となる. よって, $-1 < \mu_1$ となる.

$\mu_{i-1} > -1 \geq \mu_i$, $1 < i < n$ のとき, $\mathrm{k_m}(S') = n - i$ となる. Inequation 1 より, $\lambda_{i-1} \geq \mu_{i-1} \geq \lambda_i$ となる. ここで, $\mu_{i-1} > -1$ から $\lambda_{i-1} > -1$ となる. そして, λ_i の値によって, 以下の場合が成立する.

$$\begin{cases} \mu_{i-1} \geq \lambda_i > -1 & \text{if} \quad \mathrm{k_m}(S) = n - i = \mathrm{k_m}(S'), \\ \mu_{i-1} > -1 \geq \lambda_i & \text{if} \quad \mathrm{k_m}(S) = n - i + 1 = \mathrm{k_m}(S') + 1. \end{cases}$$

以上より, Lemma 3.4 が成立する.

$\mu_{n-1} > -1$ のとき, $\mathrm{k_m}(S') = 0$ となる. Inequation 1 より, $\lambda_{n-1} \geq \mu_{n-1} \geq \lambda_n$ となる. ここで, $\mu_{n-1} > -1$ から $\lambda_{n-1} > -1$ となる. そして, λ_n の値によって, 以下の場合が成立する.

$$\begin{cases} \mu_{n-1} \geq \lambda_n > -1 & \text{if} \quad \mathrm{k_m}(S) = 0 = \mathrm{k_m}(S'), \\ \mu_{n-1} > -1 \geq \lambda_n & \text{if} \quad \mathrm{k_m}(S) = 1 = \mathrm{k_m}(S') + 1. \end{cases}$$

以上より, Lemma 3.4 が成立する. □

Figure 2 において, Weighted graph $S_i = (V, E, w)$ から任意の頂点 $v \in V$ と, これに隣接する辺を削除した Weighted graph S' におけるスペクトルの関係を示す. Weighted graph S' から S をみたとき, 全ての固有値はその場に留まるか, 0 から離れるように移動

する. そして, 0 ないしは正負どちらか一方に新しく固有値が一つ加わる. 固有値の移動において, 隣接する固有値を超えることはない. よって, -1 を超えて小さくなる固有値はたかだか一つとなる.

以上より, 次の Theorem が成り立つ.

Theorem 3.5. Weighted graph S の最大クリークサイズ k は $\mathrm{k_m}(S)+1$ 以下となる.

Proof. Weighted graph S が サイズ k の Complete graph K_k のとき, Proposition 3.1 より, $\mathrm{k_m}(S)=\mathrm{k_m}(K_k)=k-1$ となる. よって, Theorem 3.5 が成立する.

Weighted graph S が Complete graph K_k ではないとする. Weighted graph $S_0=S$ とする. $S_{i-1}=(V_{i-1},E_{i-1},w)$, $0<i\leq n-k$ から, 最大クリークサイズを変更しない任意の頂点 $v_i\in V_i$ と, これに隣接する辺集合 $E'_i\in E_i$ を削除した Weighted graph を $S_i=(V_{i-1}-\{v_i\},E_{i-1}-E'_i,w)$ とする. よって, S_{n-k} はサイズ k の Complete graph K_k となる. Proposition 3.1 より, $\mathrm{k_m}(S_{n-k})=k-1$ となる. また, Lemma 3.4 より, $k-1=\mathrm{k_m}(S_{n-k})\leq\cdots\leq\mathrm{k_m}(S_0)=\mathrm{k_m}(S)$ となる. よって, $k\leq\mathrm{k_m}(S)+1$ となる. この Inequation は Weighted graph S_{i-1} のとり方に依存しない. 以上より, Theorem 3.5 が成立する. □

固有値は一般的には実数であり, チューリングマシンで正確に扱うことができない. しかし, -1 以下の固有値の個数を求めることは, 有理数の範囲内の計算で行うことができる. 有理数については二つの整数を用いて表現することが可能であり, その計算量は多項式時間となる.

Proposition 3.6. 有理数の四則演算は多項式時間で計算できる.

Proof. 有理数 $r\in\mathbb{Q}$ を a/b として, 二つの整数の組 $(a,b)\in\mathbb{N}^2$, $b\neq 0$ で表す. 二つの有理数 $r_1=(a_1,b_1)$, $r_2=(a_2,b_2)$ の四則演算を次に示す.

$$\begin{aligned}(a_1,b_1)\pm(a_2,b_2) &= (a_1b_2\pm b_1a_2,\, d_1d_2),\\ (a_1,b_1)\times(a_2,b_2) &= (a_1a_2,\, b_1b_2),\\ (a_1,b_1)\div(a_2,b_2) &= (a_1b_2,\, b_1a_2).\end{aligned}$$

計算結果において, a, b の最大公約数 m が 1 を超えるとき, $(a,b)\leftarrow(a/m,b/m)$ とする. 最大公約数を求めるユークリッドの互除法の計算量が $O(\log_2 n)$ となることから, 有理数の四則演算の計算量は多項式時間となる. よって, Proposition 3.6 が成立する. □

多項式時間アルゴリズム内で有理数の四則演算を用いても, 当該アルゴリズムが多項式時間で解けることから, 本論文では, 有理数の四則演算に伴う計算量を $O(1)$ として扱う.

Proposition 3.7. 多項式 $f(x) = \sum_{i=0}^{n} a_i x^i$ を $g(x) = \sum_{j=0}^{m} b_j x^j$, $0 < m < n$ で割るときの計算量は $O(n^2)$ となる.

Proof. 多項式 $f(x)$ を $g(x)$ で割るとき, 以下の計算をたかだか $n-m$ 回行うことになる.

$$f(x) - x^{n-m} \frac{a_n}{b_m}$$

$g(x)$ の項数はたかだか m 個であることから, $m(n-m)$ 回の計算を行うことになる. よって, 多項式の除算の計算量は $O(n^2)$ となる. 以上より, Proposition 3.7 が成立する. □

Weighted graph S における $\mathrm{k_m}(S)$ を求める多項式時間アルゴリズムを構成する. 以下に, この計算が多項式時間で行えることを示す.

Lemma 3.8. サイズ $1 < n$ の Weighted graph S の Adjacency matrix A の -1 以下の固有値の個数を求める関数 $\mathrm{k_m}(S)$ の計算量は $O(n^3)$ となる.

Proof. 関数 $w : E \to \mathbb{Q}_+$ が有理数の範囲に限定されていることから, Weighted graph S の Adjacency matrix A は, 有理数を要素として持つ対称行列となる. よって, A の特性方程式の係数は有理数となる.

まずはじめに, A の特性方程式の係数を求める. 有理数の四則演算の範囲で A を有理直交行列 P を用いた相似変換により, Frobenius normal form [3, 4] に変換する. ここで, 対角線ブロック行列の個数を m, 各々のブロック行列を F_i, F_i のサイズを n_i とする. 変換によって得られる Frobenius normal form は以下になる.

$$P^t A P \to \begin{pmatrix} F_1 & & & \\ & F_2 & & * \\ & & \ddots & \\ 0 & & & F_m \end{pmatrix}.$$

ここで, 各対角ブロック F_i は次のようになる.

$$F_i = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 & -a_{n_i-1} \end{pmatrix}.$$

A の特性多項式は, ブロック F_i の特性多項式 C_i の積 $\prod_{i=1}^{m} C_i$ となる. 特性多項式 C_i は, $a_0 + \sum_{k=1}^{n_i-1} a_k \lambda^k + \lambda^{n_i} = 0$ となる. そして, C_i は, 全て単根となる n_i 個の根を持

つ. ここで, $\sum_{i=1}^{m} n_i = n$ となる. このときの Frobenius normal form への変換にかかる計算量は $O(n^3)$ となる [6].

Strum Theorem [1] では, 代数方程式の根が重複を持たないとき, 任意の区間における根の個数を求めることができる. ここで, 各 F_i の特性多項式が重根を持たないことから, Strum Theorem を用いて, 各々の F_i における -1 以下の根の個数を求める. よって, 各 F_i における -1 以下の固有値の個数の和が A の -1 以下の固有値の個数となる.

サイズが n_i の特性多項式で始まる多項式の有限列を以下のように求める.

$$\begin{aligned} p_0(x) &= p(x), \\ p_1(x) &= p'(x), \\ p_j(x) &= -\mathrm{rem}(p_{j-2}(x), p_{j-1}(x)). \end{aligned}$$

ここで, $\mathrm{rem}(a, b)$ は a, b を多項式として, a を b で割ったときの剰余とする. 多項式列の最後の方程式 $p_{m_j}(x)$, $m_j \leq n_i$ は非ゼロの定数となる. $p_0(x), \ldots, p_{m_j}(x)$ を Strum 列の任意の多項式列とする. Proposition 3.7 より, 多項式の除算にかかる計算量は $O(n_i^2)$ となる. $p_j(x)$ から p_{j+1} を求めるさいに, 方程式の最高次数は 1 つ以上小さくなる. このことから, 方程式の数はたかだか n_i 個となる. よって, 多項式列を求めるための計算量は $O(n^3)$ となる.

次に, $\sigma_i(\alpha)$, $\alpha \in \mathbb{Q} + \infty$ を, ゼロを無視した, 多項式列の符号変化の数として定義する. ここで, $\sigma_i(\infty)$ は, 各多項式における一番高い次数を持つ項の係数の符号変化の数とする. よって, 重複を含めた -1 以下の根の個数は $n - (\sigma_i(-1) - \sigma_i(\infty))$ となる. n_i 次多項式を高次から順に計算すると, 計算量は $O(n_i)$ となる. 多項式の数 m_j は n_i 以下となるため, $\sigma_i(-1)$ の計算量は $O(n_i^2)$, $\sigma_i(\infty)$ の計算量は $O(n_i)$ となる. Weighted graph S の Adjacency matrix A の -1 以下の固有値の個数は, 各対角ブロック F_i における -1 以下の固有値の個数を足し合わせたものとなる. よって, $\mathrm{k_m}(S)$ は $n - \sum_{i=1}^{m}(\sigma_i(-1) - \sigma_i(\infty))$ となる.

以上より, Weighted graph S の Adjacency matrix A の -1 以下の固有値の個数 $\mathrm{k_m}(S)$ を求める関数の計算量は $O(n^3)$ となる. よって, Lemma 3.8 が成立する. □

3.3 Control of Weighted graph

この Section では, Weighted graph S に重み 1/2 の辺を適宜加えることで, 頂点削除時のグラフスペクトルを制御した, Weighted graph 集合 $\mathscr{T} \subset \mathscr{S}$ を導入する. そして, Weighted graph $S \in \mathscr{S}$ を $\mathscr{T}$ に変換するための関数 init_graph : $\mathscr{S} \to \mathscr{T}$ と, $T = (V, E, w) \in \mathscr{T}$ から頂点 $v \in V$ を削除するための関数 delete_vertex : $\{(T, v)\} \to \mathscr{T}$ を

導入する.

Corollary 3.9. サイズ $1 < n$ の Weighted graph S において, 頂点 $v_a, v_b \in V$, $v_a \neq v_b$, $(v_a, v_b) \notin E$ があるとする. 辺 $e' = (v_a, v_b)$, $0 < w(e') < 1$ として, Weighted graph $S' = (V, E \cup \{e'\}, w)$ とする. このとき, S と S' の最大クリークサイズは同じとなる.

Proof. Definition 2.7 より, S, S' のすべてのクリークに属する辺 e において $w(e) = 1$ となる. Weighted graph S' に追加した辺 e' の重み $w(e')$ は 1 ではないため, S と S' のすべてのクリークは同じとなる. よって, S と S' の最大クリークも同じとなり, 最大クリークサイズも同じとなる. 以上より, Corollary 3.9 が成立する. □

Lemma 3.10. サイズ $1 < n$ の Weighed graph を $S = (V, E, w)$ とする. 2 つの頂点 $v_a, v_b \in V$, $v_a \neq v_b$, $(v_a, v_b) \notin E$ において, 辺 e', $0 < w(e') < 1$ を加えた Weighted graph を $S' = (V, E \cup \{e'\}, w)$ とする. このとき, $\mathrm{k_m}(S')$ は $= \mathrm{k_m}(S) - 1$, $\mathrm{k_m}(S)$, $\mathrm{k_m}(S) + 1$ のいずれかとなる.

Proof. Weighted graph S から, 頂点 v_a ないしは v_b とそれに隣接する辺を除去した Weighted graph を S'' とする. Lemma 3.4 より, $\mathrm{k_m}(S'') = \mathrm{k_m}(S)$ or $\mathrm{k_m}(S) - 1$ かつ, $\mathrm{k_m}(S'') = \mathrm{k_m}(S')$ or $\mathrm{k_m}(S') - 1$ となる. よって, 以下が成り立つ.

$$
\begin{array}{lllll}
\mathrm{k_m}(S') & = & \mathrm{k_m}(S'') & = & \mathrm{k_m}(S) \quad \text{or,} \\
\mathrm{k_m}(S') & = & \mathrm{k_m}(S'') & = & \mathrm{k_m}(S) - 1 \quad \text{or,} \\
\mathrm{k_m}(S') & = & \mathrm{k_m}(S'') + 1 & = & \mathrm{k_m}(S) + 1 \quad \text{or,} \\
\mathrm{k_m}(S') & = & \mathrm{k_m}(S'') + 1 & = & \mathrm{k_m}(S) - 1 + 1.
\end{array}
$$

以上より, Lemma 3.10 が成立する. □

Weighted graph S に重み付き辺を挿入することで, 頂点削除時のグラフスペクトルを制御した, Weighted graph 集合 $\mathscr{T} \subset \mathscr{S}$ を導入する.

Definition 3.11. Weighted graph $T = (V, E, w) \in \mathscr{T} \subset \mathscr{S}$ は以下の 2 つの条件を満たすとする. 1 つ目の条件は, $w(e) = 1$ を満たす辺 $e \in E$ が存在する. 2 つ目の条件は, 2 頂点 $v_a, v_b \in V$, $(v_a, v_b) \notin E$ 間に辺 e', $w(e') = 1/2$ を挿入した Weighted graph T' を構成したとき, どのような頂点 v_a, v_b においても $\mathrm{k_m}(T') \geq \mathrm{k_m}(T)$ が成立する.

関数 init_graph : $\mathscr{S} \to \mathscr{T}$ を導入する. 入力となる Weighted graph $S = (V, E, w)$ のサイズを $1 < n$ とし, 少なくとも一つの辺 $e \in E$ が $w(e) = 1$ を満たすとする. そして, S に対して以下の操作を行うとする.

Algorithm 1: Weighted graph S を引数として取り, $\mathscr{T}$ に属する Weighted graph を出力する. Weighted graph S に辺 e, $w(e) = 1/2$ を挿入したグラフを S' とする. $\mathrm{k_m}(S') = \mathrm{k_m}(S) - 1$ となるとき, $S \leftarrow S'$ とする. そして, いずれの頂点間における辺の挿入によっても $\mathrm{k_m}(S)$ が小さくならない S を返す.

```
1  Function initialize_graph(S):
2      A ← Adjacency matrix of S
3      k_m ← k_m(S)
4      foreach v_i, v_j ∈ V such that (v_i, v_j) ∉ E do
5          S' ← graph which is added an edge with weight 1/2 to S
6          k'_m ← k_m(S')
7          if k'_m < k_m then
8              S ← S'
9              k_m ← k'_m
10         end
11     end
12     return S
```

Weighted graph $S_0 = S$ とする. Weighted graph $S_{i-1} = (V_{i-1}, E_{i-1}, w)$, $0 < i < j < n$ において, 任意の 2 つの頂点 $v_{a,i-1}, v_{b,i-1} \in V_{i-1}$, $v_{a,i-1} \neq v_{b,i-1}$, $(v_{a,i-1}, v_{b,i-1}) \notin E_{i-1}$ 間に, 辺 e_{i-1}, $w(e_{i-1}) = 1/2$ を挿入した Weighted graph を S_i とする. このとき, $\mathrm{k_m}(S_i) = \mathrm{k_m}(S_{i-1}) - 1$ が成立するとする. なお, Corollary 3.9 より, 辺 e_{i-1} を挿入しても, S の最大クリークサイズは変化しない.

Weighted graph S_0 において, 任意の 2 つの頂点 $v_{a,0}, v_{b,0} \in V_0$, $v_{a,0} \neq v_{b,0}$, $(v_{a,0}, v_{b,0}) \notin E_0$ 間に辺 $e_0 = (v_{a,0}, v_{b,0})$, $w(e_0) = 1/2$ を挿入したときの Weighted graph S'_0 とする. $\mathrm{k_m}(S'_0) \geq \mathrm{k_m}(S_0)$ となる場合, 関数 init_graph() は S'_0 を出力する.

$\mathrm{k_m}(S'_0) < \mathrm{k_m}(S_0)$ となる場合, Weighted graph $S_j = (V_j, E_j, w)$ とする. このとき, 任意の 2 つの頂点 $v_{a,j}, v_{b,j} \in V_j$, $v_{a,j} \neq v_{b,j}$, $(v_{a,j}, v_{b,j}) \notin E_j$ 間に辺 $e_j = (v_{a,j}, v_{b,j})$, $w(e_j) = 1/2$ を挿入したとき, $\mathrm{k_m}(S_j) \geq \mathrm{k_m}(S_{j-1})$ が成立するとする. このとき, 関数 init_graph() は S_{j-1} を出力する. なお, 前提より, S は $w(e) = 1$ となる辺 $e \in E$ を 1 つ以上持つことから, S は サイズ 2 の Complete graph K_2 を Subgraph として持つことになる. Complete graph K_2 の任意の頂点を削除したグラフ K_1 において, $\mathrm{k_m}(K_1) < \mathrm{k_m}(K_2)$ となる. よって, S_{j-1} は必ず存在する.

関数 init_graph : $\mathscr{S} \to \mathscr{T}$ に対する擬似コードを Algorithm 1 に示す. Line 3 において, $\mathrm{k_m}(S)$ を求める計算量は $O(n^3)$ となる. Line 4–11 において, 全ての頂点間において行う辺挿入テスト回数はたかだか S の頂点数の 2 乗回となる. Line 6 において, $\mathrm{k_m}(S')$ を求める計算量は $O(n^3)$ となる. よって, Algorithm 1 の計算量は $O(n^3 + n^2 \times n^3) = O(n^5)$ となる.

Corollary 3.12. Complete graph K_k, $1 < k$ において, $K_k \in \mathscr{T}$ となる.

Proof. Complete graph K_k は任意の 2 つ頂点間において辺を持つ. よって, `initialize_graph`(K_k) による辺の挿入はない. そして, Complete graph K_k の任意の頂点 v を削除した Weighted graph は Complete graph K_{k-1} となる. ここで, Proposition 3.1 より, $\mathrm{k_m}(K_{k-1}) = \mathrm{k_m}(K_k) - 1$ となる. 以上より, Corollary 3.12 が成立する. □

Lemma 3.13. $1 < m$ 個の Complete graph を $C_i = (V_i, E_i)$, $0 < i \leq m$, $\sum_{i=1}^{m} n_i = n$, $1 < n_i$ とし, Weighted graph $S = (\bigoplus_{i=1}^{m} V_i, \bigoplus_{i=1}^{m} E_i), w)$ とする. このとき, $S \notin \mathscr{T}$ となる.

Proof. Weighted graph S の Adjacency matrix を A, Complete graph C_i の Adjacency matrix を A_i とする. このとき, A は A_i を対角ブロックとして持つ行列となる. Proposition 3.1 より, 重複を含めた固有値 -1 の個数は $\sum_{i=1}^{m}(n_i - 1) = n - m$ 個となる.

任意の 2 つの Complete graph C_i, C_j, $i \neq j$ 間において, 任意の 2 つの頂点 $v_a \in V_i$, $v_b \in V_j$ 間に辺は存在しない ($(v_a, v_b) \notin E$). そこで, 辺 e', $w(e') = 1/2$ を S に挿入した Weighted graph を S' とする. また, S' の Adjacency matrix を A' とする.

Adjacency matrix A_i, A_j における線形従属となる行ベクトルの本数は, それぞれ, $n_i - 1$, $n_j - 1$ となる. 辺 e の挿入により, 行ベクトルはそれぞれ一つずつ減少する. よって, Adjacency matrix A' の重複を含めた固有値 -1 の個数は 2 つ減少して, $n - m - 2$ となる.

頂点 v_a, v_b のいずれかを削除した Weighted graph S'' とする. 頂点 v_a, v_b のどちらを削除しても, 重複を含めた固有値 -1 の個数は $n - m - 1$ 個となり, $\mathrm{k_m}(S'') = \mathrm{k_m}(S) - 1$ が成立する. よって, $\mathrm{k_m}(S') = \mathrm{k_m}(S)$ or $\mathrm{k_m}(S) - 1$ となる.

ここで, $\mathrm{k_m}(S') = \mathrm{k_m}(S)$ を仮定する. Inequation 1 より, Adjacency matrix A' の重複を含めた固有値 -1 の個数は $n - m$ or $n - m - 1$ 個となる. しかし, Adjacency matrix A' の重複を含めた固有値 -1 の個数は $n - m - 2$ であるため, $\mathrm{k_m}(S') = \mathrm{k_m}(S)$ の仮定が誤りだったことになる. よって, $\mathrm{k_m}(S') = \mathrm{k_m}(S) - 1$ となる.

以上より, Weighted graph S は辺 e', $w(e') = 1$ を挿入した Weighted graph を S' としたとき, $\mathrm{k_m}(S') = \mathrm{k_m}(S) - 1$ となる. よって, $S \notin \mathscr{T}$ となる. このことから, Lemma 3.13 が成立する. □

figure 3 において, 関数 `initialize_graph` : $\mathscr{S} \to \mathscr{T}$ における, Weighted graph S

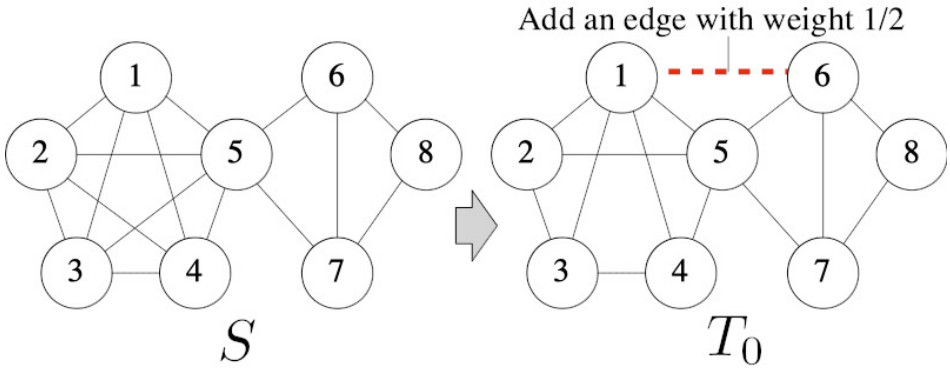

図 3　関数 initialize_graph: $\mathscr{S} \to \mathscr{T}$ において, Weighted graph S から $T_0 \in \mathscr{T}$ を作成する様子.
Weighted graph $S = (V, E, w)$ の頂点 v_1, v_6 間に, 重み $w(e) = 1/2$ の辺 e を加えた Weighted graph を T_0 とすると, $\mathrm{k_m}(S) = 5$, $\mathrm{k_m}(T_0) = 4$ となる. その他の頂点間に辺を加えてもさらに減少しないことから, 関数は T_0 を返す.

から $T_0 \in \mathscr{T}$ を作成する様子を示す. Weighted graph $S = (V, E, w)$ の頂点 v_1, v_6 間に, 重み $w(e) = 1/2$ の辺 e を加えた Weighted graph を T_0 とする. このとき, $\mathrm{k_m}(S) = 5$, $\mathrm{k_m}(T_0) = 4$ となる. T_0 の任意の頂点間において重み $w(e') = 1/2$ の辺 e' を加えた Weighted graph T_0' において, $\mathrm{k_m}(T_0') = \mathrm{k_m}(T_0) = 4$ となる. よって, $T_0 \in \mathscr{T}$ となり, 関数は T_0 を返す.

Weighted graph $T = (V, E, w) \in \mathscr{T}$ から頂点 $v \in V$ を削除するための関数 delete_vertex : $\{(T, v \in V)\} \to \mathscr{T}$ を導入する.

任意の頂点 $v \in V$ と v に隣接する辺を削除した Weighted graph を $T' = (V', E', w')$ とする. Lemma 3.4 より, $\mathrm{k_m}(T') = \mathrm{k_m}(T)$ or $\mathrm{k_m}(T) - 1$ が成立する. $\mathrm{k_m}(T') = \mathrm{k_m}(T) - 1$ のとき, 関数 delete_vertex(T, v) は T' を返す. $\mathrm{k_m}(T') = \mathrm{k_m}(T)$ のとき, 2 つの頂点 $v_a, v_b \in V'$, $(v_a, v_b) \notin E'$ において, 辺 $e'' = (v_a, v_b)$, $w'(e'') = 1/2$ を T' に挿入した Weighted graph を $T'' = (V'', E'' \cup \{e''\}, w'')$ とする. 任意の頂点 v_a, v_b について $\mathrm{k_m}(T'') = \mathrm{k_m}(T)$ が成立するとき, 関数 delete_vertex(T, v) は T' を返す. ある頂点 v_a,v_b について $\mathrm{k_m}(T'') = \mathrm{k_m}(T) - 1$ が成立するとき, 関数 delete_vertex(T, v) は T'' を返す.

Lemma 3.14. Weighted graph $T = (V, E, w) \in \mathscr{T}$, $v \in V$ において, delete_vertex$(T, v) \in \mathscr{T}$ が成り立つ.

Proof. Weighted graph $T_v = (V_v, E_v, w)$ を delete_vertex(T, v) とする.

Algorithm 2: Weighted graph $T = (V, E, w) \in \mathscr{T}$ において, 頂点 $v \in V$ とこれに隣接する辺を削除する. delete_vertex$(T, v) \in \mathscr{T}$ となるように, 辺 e, $w(e) = 1/2$ を適宜挿入する.

```
1  Function delete_vertex(T, v):
2      k_m ← k_m(T)
3      T ← graph which is deleted v of T
4      if k_m(T) = k_m then
5          foreach v_i, v_j ∈ V such that (v_i, v_j) ∉ E do
6              T' ← graph which is added e, w(e) = 1/2 to T
7              if k_m(T') < k_m then
8                  return T'
9              end
10         end
11     end
12     return G_s
```

$\mathrm{k_m}(T_v) = \mathrm{k_m}(T)$ のとき, 前提より, 辺の挿入によって $\mathrm{k_m}(T_v)$ が変化しないことから, $T_v \in \mathscr{T}$ が成立する.

$\mathrm{k_m}(T_v) = \mathrm{k_m}(T) - 1$ のとき, ある 2 つの頂点 $v_c, v_d \in V_v$, $(v_c, v_d) \notin E_v$ において, 辺 $e_v = (v_c, v_d)$, $w(e_v) = 1/2$ を T_v に挿入した Weighted graph を T'_v として, $\mathrm{k_m}(T'_v) = \mathrm{k_m}(T_v) - 1$ となると仮定する. このとき, 辺 e_v を T に追加した Weighted graph を T_w とすると, T_w から頂点 v とこれに隣接する辺を除去した Weighted graph が T'_v となる. Lemma 3.10 より, $\mathrm{k_m}(T_w) = \mathrm{k_m}(T) - 1$ or $\mathrm{k_m}(T)$ or $\mathrm{k_m}(T) + 1$ となる. $\mathrm{k_m}(T_w) = \mathrm{k_m}(T) - 1$ のとき, 前提 $T \in \mathscr{T}$ に反する. $\mathrm{k_m}(T_w) = \mathrm{k_m}(T)$ のとき, $\mathrm{k_m}(T'_v) = \mathrm{k_m}(T_w) - 2$ となり, Lemma 3.4 に反する. $\mathrm{k_m}(T_w) = \mathrm{k_m}(T) + 1$ のとき, $\mathrm{k_m}(T'_v) = \mathrm{k_m}(T_w) - 3$ となり, Lemma 3.4 に反する. よって, $\mathrm{k_m}(T'_v) = \mathrm{k_m}(T_v) - 1$ が成立する頂点 v_c, v_d が存在する仮定は成立しない. このことから, $T_v \in \mathscr{T}$ が成立する.

以上より, Lemma 3.14 が成立する. □

Weighted graph $T = (V, E, w) \in \mathscr{T}$, $v \in V$ における, 頂点 $v \in V$ を削除する関数 delete_vertex(T, v) に対する擬似コードを Algorithm 2 に示す. Line 2,4 において $\mathrm{k_m}(T)$ を求める計算量は $O(n^3)$ となる. Line 7 において $\mathrm{k_m}(T')$ を求める計算量は $O(n^3)$ となる. Line 5–10 において, 全ての頂点間において行う辺挿入テストはたかだか T の頂点数の 2 乗回となる. よって, Algorithm 2 の計算量は $O(n^3 + n^3 + n^2 \times n^3) = O(n^5)$ となる.

figure 4 において, 関数 delete_vertex: $\{(T, v) \,|\, T \in \mathscr{T}, v \in V\} \to \mathscr{T}$ において頂点を削除する様子を示す. Figure 3 内に示した Weighted graph $T_0 = (V_0, E_0, w)$ から頂点 v_5 を削除した Weighted graph を T_1 とする. このとき, $\mathrm{k_m}(T_1) = \mathrm{k_m}(T_0) = 4$ とな

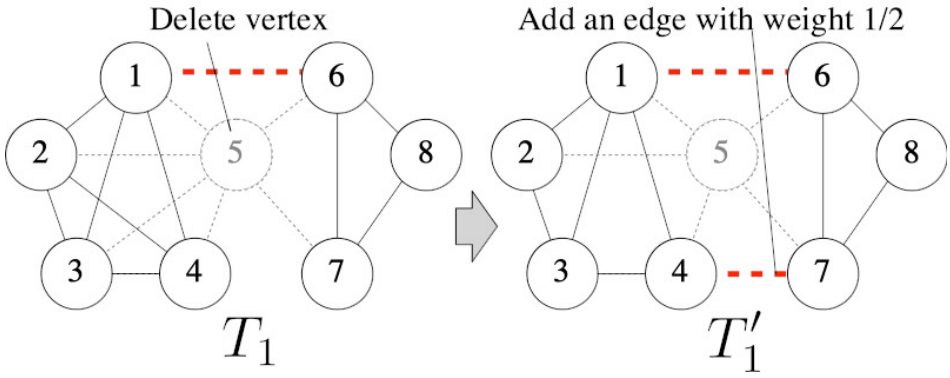

図 4　関数 delete_vertex: $\{(T, v) \,|\, T \in \mathscr{T}, v \in V\} \to \mathscr{T}$ において頂点を削除する様子.
Weighted graph T_0 (in Figure 3) から頂点 v_5 を削除した Weighted graph を T_1 とすると, $\mathrm{k_m}(T_1) = \mathrm{k_m}(T_0) = 4$ となる. ここで, T_1 において頂点 v_4, v_7 間に, 重み $w(e) = 1/2$ の辺 e を挿入した Weighted graph を T_1' とすると, $\mathrm{k_m}(T_1') = \mathrm{k_m}(T_1) - 1$ となる. よって, 関数は T_1' を返す.

る. ここで, T_1 において頂点 v_4, v_7 間に, 重み $w(e) = 1/2$ の辺 e を挿入した Weighted graph を T_1' とすると, $\mathrm{k_m}(T_1') = \mathrm{k_m}(T_1) - 1$ となる. よって, 関数は T_1' を返す.

3.4　Proof for extraction of a maximum clique

この Section では, Weighted graph $T \in \mathscr{T}$ に対して, 最大クリークサイズが変化しない頂点集合が求まることを証明する.

まずはじめに, Weighted graph $T = (V, E, w) \in \mathscr{T}$ と削除対象外頂点集合 V_p を入力として, 頂点集合 $V_a \subset V$ と T から V_a を除去した Weighted graph T_r を出力する関数 get_graph_keeping_km() を導入する. この関数は次を満たすとする.

任意の頂点 $v \in V - V_p$ において, $T' = \texttt{delete\_vertex}(T, v)$ として, $\mathrm{k_m}(T') = \mathrm{k_m}(T) - 1$ が成り立つとき, 関数 get_graph_keeping_km() は $(T, \emptyset)$ を返す. 一方, $\mathrm{k_m}(T') \neq \mathrm{k_m}(T) - 1$ が成り立つとき以下を行う. Weighted graph $T_0 = (V_0, E_0, w)$ を T とする. ある頂点 $v_{i-1} \in V_{i-1} - V_p$, $0 < i$ において, $T_i = \texttt{delete\_vertex}(T_{i-1}, v_{i-1})$, $\mathrm{k_m}(T_i) = \mathrm{k_m}(T_{i-1})$ を満たすとする. そして, 任意の頂点 $v_{j-1} \in V_{j-1} - V_p$, $i < j$ において, $T_j = \texttt{delete\_vertex}(T_{j-1}, v_{j-1})$, $\mathrm{k_m}(T_j) = \mathrm{k_m}(T_{j-1}) - 1$ が成り立つとする. このとき, 関数 get_graph_keeping_km() は $(T_{j-1}, \{v_{i-1} \,|\, 0 < i \leq j\})$ を返す. なお, 前提より, T は $w(e) = 1$ となる辺 $e \in E$ を 1 つ以上持つことから, T は サイズ 2 の Complete graph K_2 を Subgraph として持つことになる. Complete graph K_2 の任意

Algorithm 3: Weighted graph $T = (V, E, w) \in \mathscr{T}$ と削除対象外頂点集合 V_p を入力として, 頂点集合 $V_a \in V - V_p$ と, T から V_a に属する頂点全てとこれに隣接する辺を全て除去した Weighted graph $T_r = (V_r, E_r, w)$, $V_r = V - V_a$, $E_r \in E$ を返す. このとき, $\mathrm{k_m}(T_r) = \mathrm{k_m}(T)$ となる. また、T_r から任意の頂点 $v \in V_r - V_p$ とこれに隣接する辺を除去した Weighted graph を T'_r としたとき, $\mathrm{k_m}(T'_r) = \mathrm{k_m}(T_r) - 1$ となる.

```
1  Function get_graph_keeping_km(T = (V, E, w), V_p):
2      V_a ← ∅
3      k_m ← k_m(T)
4      foreach v ∈ V − V_p do
5          T' ← delete_vertex(T, v)
6          if k_m(T') = k_m then
7              T ← T'
8          end
9      end
10     return (T, V_a)
```

の頂点を削除したグラフ K_1 において, $\mathrm{k_m}(K_1) < \mathrm{k_m}(K_2)$ となる. よって, T_{j-1} は必ず存在する.

関数 get_graph_keeping_km() に対する擬似コードを Algorithm 3 に示す. Line 3 において $\mathrm{k_m}(T)$ を求める計算量は $O(n^3)$ となる. Line 4–9 において, 全ての頂点においてテストを行う回数は n 回となる. Line 5 において 関数 `delete_vertex`() にかかる計算量は $O(n^5)$ となる. Line 6 において $\mathrm{k_m}(T')$ を求める計算量は $O(n^3)$ となる. よって, Algorithm 3 の計算量は $O(n^3 + n \times (n^5 + n^3)) = O(n^6)$ となる.

Weighted graph $T \in \mathscr{T}$ において, $(T_r, V_a) =$ `get_graph_keeping_km`$(T, \emptyset)$ を求める. このとき, 以下の場合がある.

C1se 1: T_r が Complete graph のとき,
C2se 2: T_r が Complete graph ではなく, $V_a = \emptyset$ のとき,
C3se 3: T_r が Complete graph ではなく, $V_a \neq \emptyset$ のとき.

これらのケースについて, 最大クリークサイズに影響のない頂点集合が求まることを示す.

3.4.1 Case 1: T_r が Complete graph のとき

Case 1 のとき, 以下が成立する.

Lemma 3.15. Weighted graph T_r が Complete graph のとき, T_r は T の最大クリークとなる.

Proof. Weighted graph T が Complete graph のとき $T_r = T$ となる. よって, T_r は T の最大クリークとなる. このことから, Lemma 3.15 が成立する.

Weighted graph T が Complete graph ではないとき, Theorem 3.5 より, Weighted graph T の最大クリークサイズは $\mathrm{k_m}(T)+1$ となる. 前提より, T_{j-1} が Complete graph であることから, そのサイズは $\mathrm{k_m}(T_{j-1})+1$ となる. $\mathrm{k_m}(T) = \mathrm{k_m}(T_{j-1})$ より, T の最大クリークサイズは $\mathrm{k_m}(T_{j-1})+1$ となり, T_{j-1} のサイズと同じとなる. よって, T_{j-1} は T における最大クリークとなる. 以上より, Lemma 3.15 が成立する. □

3.4.2 Case 2: T_r が Complete graph ではなく, $V_a = \emptyset$ のとき

Case 2 のとき, 以下が成立する.

Lemma 3.16. Weighted graph $T_r = (V, E, w)$ が Complete graph ではなく, $V_a = \emptyset$ とする. T_r から頂点 $v \in V_r$ を削除した Weighted graph を $T'_r = \texttt{delete\_vertex}(T_r, v)$ として, $(T_s, V_b) = \texttt{get\_graph\_keeping\_km}(T'_r, \emptyset)$ としたとき, $0 < |V_b|$ となるような頂点 v が存在する.

Proof. 前提より $T_r = T$ となる. Definition 3.11 より, Weighted graph $T = (V, E, w) \in \mathscr{T}$ は少なくとも 1 つの辺 e, $w(e) = 1$ を持つ. よって, Weighted graph T のサイズが 2 以下のとき, T は Complete graph となることから, T のサイズは 3 以上となる.

Weighted graph T の最大クリークの一つを $C = (V_c, E_c, w)$ とし, その Adjacency matrix を A_c とする. ここで, Proposition 3.1 より, A_c の重複を含めた固有値 -1 は $|V_c| - 1$ 個, 固有値 $|V_c| - 1$ は 1 個となる.

任意の頂点 $v_0 \in V$, $v_0 \notin V_c$ について, Weighted graph T_d を $(V_c \cup \{v_0\}, E_c \cup \{(v_0, v_i) \in E \mid v_0 \in V\}, w)$ とし, この Adjacency matrix A_d とする. ここで, 前提より C が最大クリークであったことから, Weighted graph T_d は Complete graph ではない. 辺の重みは $0 < w((v_0, v_i)) \leq 1$ であることから, A_d の重複を含む -1 以下の固有値の個数は $|V_c| - 1$ であり, -1 より大きく 0 以下となる固有値の個数が 1, $|V_c| - 1$ 以上の固有値の個数が 1 となる. よって, $T_d \in \mathscr{T}$, $\mathrm{k_m}(T_d) = \mathrm{k_m}(C)$ となる.

$\mathrm{k_m}(T) = \mathrm{k_m}(C)$ と仮定すると, T から任意の頂点 v を削除した Weighted graph T' において, $\mathrm{k_m}(T') = \mathrm{k_m}(T) - 1$ を満たすという前提に反するため, $\mathrm{k_m}(T) > \mathrm{k_m}(C)$ となる.

以下に帰納法を用いる.

Case A: $\mathrm{k_m}(T) = \mathrm{k_m}(C) + 1$ のとき, C に頂点 v_0 を加えた Weighted graph T_d

において, $\mathrm{k_m}(T) = \mathrm{k_m}(C)$ であったことから, $v' \neq v_0$, $v', v_0 \notin V_c$ において $T_0 = \texttt{delete\_vertex}(T, v')$ としたとき, $T_1 = \texttt{delete\_vertex}(T_0, v_0)$, $\mathrm{k_m}(T_1) = \mathrm{k_m}(T_0)$ が存在する. そして, $j = |V| - |C|$ のとき, 任意の頂点 v_{j-1} が Complete graph C の頂点となるため, $T_j = \texttt{delete\_vertex}(T_{j-1}, v_{j-1})$, $\mathrm{k_m}(T_j) = \mathrm{k_m}(T_{j-1}) - 1$ となる.

Case B: $\mathrm{k_m}(T_p) = \mathrm{k_m}(C) + p$, $T_p \in \mathscr{T}$ において, $v'_p \neq v_{p,0}$, $v'_p, v_{p,0} \notin V_c$ として, $T_{p,0} = \texttt{delete\_vertex}(T_p, v'_p)$ としたとき, $T_{p,i_p} = \texttt{delete\_vertex}(T_{p,i_p-1}, v_{p,i_p-1})$, $\mathrm{k_m}(T_{p,i_p}) = \mathrm{k_m}(T_{p,i_p-1})$ および, $T_{p,j_p} = \texttt{delete\_vertex}(T_{p,j_p-1}, v_{p,j_p-1})$, $\mathrm{k_m}(T_{p,j_p}) = \mathrm{k_m}(T_{p,j_p-1}) - 1$ となるような $0 < i_p < j_p$ が存在するとする. このとき, $\mathrm{k_m}(T_{p,j_p-1}) = \mathrm{k_m}(C) + p - 1$ となる.

Weighted graph T において, Weighted graph T_p が T の Induced subgraph であり, $|V| = |V_p| + 1$ のとき, $\mathrm{k_m}(T) = \mathrm{k_m}(C) + p + 1$ を仮定する. $T'_{p,0} = \texttt{delete\_vertex}(T_p, v'_p)$ として, $T'_{p,i_p} = \texttt{delete\_vertex}(T'_{p,i_p-1}, v_{p,i_p-1})$, $0 < i_p < j_p$ とすると, $\mathrm{k_m}(T'_{p,j_p-1}) = \mathrm{k_m}(C) + p$ となる. しかし, Case B で導入した前提より, $\mathrm{k_m}(T_{p,j_p-1}) = \mathrm{k_m}(C) + p - 1$ であり, 一つ頂点を加えた Weighted graph T'_{p,j_p-1} もまた, $\mathrm{k_m}(T'_{p,j_p-1} = \mathrm{k_m}(C) + p - 1$ となる必要がある. よって, $\mathrm{k_m}(T) = \mathrm{k_m}(C) + p$ となる.

このことから, $\mathrm{k_m}(T) = \mathrm{k_m}(C) + p + 1$ であり, Weighted graph T_p が T の Induced subgraph のとき, $|V| > |V_p| + 1$ となる. よって, $v'_{p+1} \neq v_{p+1,0}$, $v'_{p+1}, v_{p+1,0} \notin V_c$ として, $T_{p+1,0} = \texttt{delete\_vertex}(T_{p+1}, v'_{p+1})$ としたとき, $T_{p+1,i_{p+1}} = \texttt{delete\_vertex}(T_{p+1,i_{p+1}-1}, v_{p+1,i_{p+1}-1})$, $\mathrm{k_m}(T_{p+1,i_{p+1}}) = \mathrm{k_m}(T_{p+1,i_{p+1}-1})$ および, $T_{p+1,j_{p+1}} = \texttt{delete\_vertex}(T_{p+1,j_{p+1}-1}, v_{p+1,j_{p+1}-1})$, $\mathrm{k_m}(T_{p+1,j_{p+1}}) = \mathrm{k_m}(T_{p+1,j_{p+1}-1}) - 1$ となるような $0 < i_{p+1} < j_{p+1}$ が存在する.

以上, Case A, Case B より, 任意の $0 < p$ において, Lemma 3.16 が成立する. □

Lemma 3.17. Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a = \emptyset$ とする. T_r から頂点 $v \in V_r$ を削除した Weighted graph を $T'_r = \texttt{delete\_vertex}(T_r, v)$ として, $(T_s, V_a) = \texttt{get\_graph\_keeping\_km}(T, \emptyset)$ としたとき, $0 < |V_a|$ となるような頂点 v が存在する. このとき, T と T_s の最大クリークサイズは同じとなる.

Proof. 前提より $T_r = T$ となる. 頂点集合 $V_d = V_a \cup \{v\}$ とする. 任意の頂点 $v' \in V_d$ において, $\mathrm{k_m}(\texttt{delete\_vertex}(T, v')) = \mathrm{k_m}(T) - 1$ となる. これは, 頂点集合 V_d の組み合わせによって, $\mathrm{k_m}(T)$ が減少することを示している. よって, V_d に属する頂点を削除しても, T の最大クリークサイズは変化しない. 以上より, Lemma 3.17 が成立する. □

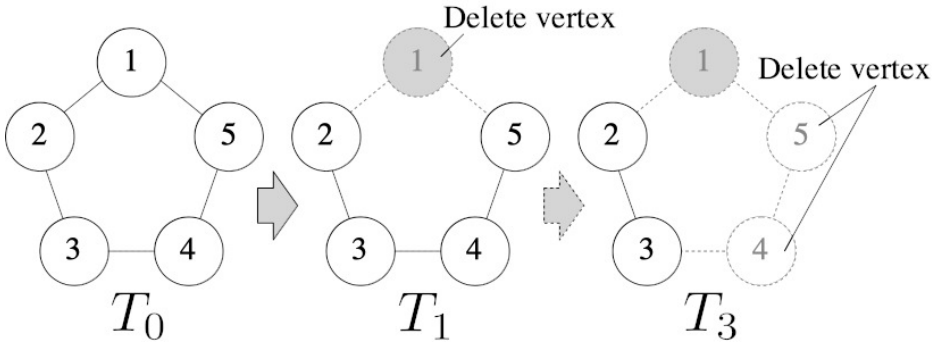

図 5　Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a = \emptyset$ のとき, 最大クリークサイズを変更しない頂点を削除する例.
Weighted graph T_0 は T_r の一例となる. Weighted graph T_1 = delete_vertex(T_0, v_1) において, get_graph_keeping_km$(T_1, \emptyset)$ は $(T_3, V_b = \{v_4, v_5\}$ を返す. よって, $0 < |V_b|$ となり, Lemma 3.17 から, Weighted graph T_0 と T_3 の最大クリークサイズは同じとなる.

Figure 5 において, Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a = \emptyset$ のとき, 最大クリークサイズを変更しない頂点を削除する例を示す. T_0 は Complete graph ではなく, T_0 の任意の頂点を削除した Weighted graph T_0' において, $\mathrm{k_m}(T_0') = \mathrm{k_m}(T_0) - 1$ となる. よって, Weighted graph T_0 は T_r の一例となる. T_0 から頂点 v_1 を削除した Weighted graph T_1 = `delete_vertex`(T_0, v_1) において, $\mathrm{k_m}(T_1) = \mathrm{k_m}(T_0) - 1$ となる. そして, `get_graph_keeping_km`$(T_1, \emptyset)$ は $(T_3, V_b = \{v_4, v_5\})$ を返す. よって, $0 < |V_b|$ となり, Lemma 3.17 から, Weighted graph T_0 と T_3 の最大クリークサイズは同じとなる.

Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a = \emptyset$ のとき, 最大クリークを 1 つ抽出する擬似コードを Algorithm 4 に示す. 頂点削除は Lemma 3.16, 3.17 に基づいて行う. Line 2–5 において, ループ数はたかだか $|V_r|$ となる. Line 3 において, 関数 `delete_vertex`() の計算量は $O(n^5)$ となる. Line 4 において, 関数 `get_graph_keeping_km`() の出力を (T_s, V_s) とすると, T_s の任意の頂点を削除したさいに $\mathrm{k_m}()$ が減少することから, 関数 `get_graph_keeping_km`() の入力条件を満たす. よって, $T_r \leftarrow T_s$ として計算を続行する. 関数 `get_graph_keeping_km`() の計算量は $O(n^6)$ となる. よって, 関数 get_graph_wo_va() の計算量は $O(n \times (n^5 + n^6)) = O(n^7)$ となる.

Algorithm 4: Weighted graph T_r が Complete graph ではなく, かつ $V_a = \emptyset$ のとき, 頂点削除により最大クリークサイズが変化しない頂点を削除していくことで, 最大クリークを出力する.

```
Function get_graph_wo_va(T_r = (V_r, E_r, w)):
    foreach v ∈ V_r do
        T'_r ← delete_vertex(T_r, v)
        (T_r, V_s) ← get_graph_keeping_km(T'_r, ∅)
    end
    return T_r
```

3.4.3 Case 3: T_r が Complete graph ではなく, $V_a \neq \emptyset$ のとき

Case 3 のとき, 以下が成立する.

Corollary 3.18. Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ とする. Weighted graph T_d を $(V_r \cup \{v\}, E_r \subset E, w)$, $v \in V_a$ とする. このとき, $T_d \in \mathscr{T}$ となる.

Proof. 頂点集合 $V_{a,0} = V_a$ とする. 頂点 $v_i \in V_{a,i}$, $0 < i \leq |V_a|$ において, $T_i =$ `delete_vertex`(T_{i-1}) とする. このとき, $\mathrm{k_m}(T_i) = \mathrm{k_m}(T_{i-1})$ となる. よって, Weighted graph T_d には辺 e, $w(e) = 1$ の挿入が行われていない. このことから, Corollary 3.18 が成り立つ. □

Lemma 3.19. Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ とする. Weighted graph T_d を $(V_r \cup \{v\}, E_r \subset E, w)$, $v \in V_a$ とする. `get_graph_keeping_km`$(T_d, \{v\}) = (T_s, V_s)$ において, $V_s = \emptyset$ のとき, 頂点 v の削除によって T_r の最大クリークサイズは変更しない.

Proof. Lemma 3.16 と Lemma 3.17 より, Weighted graph T_r には, 頂点削除による最大クリークサイズを変更しない頂点集合 V_b が存在する. しかし, V_b によって得られている $\mathrm{k_m}(T_r)$ を頂点 v の存在によって維持できないため, V_b に属する頂点を削除できない. このことから, T_d から頂点 v を削除しても, 最大クリークサイズは変化しない. よって, Lemma 3.19 が成立する. □

Figure 6 において, Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ のとき, 頂点削除によって最大クリークサイズを変更しない頂点を削除する例を示す. Weighted graph $T_0 = (V_0, E_0, w)$ はこの一例となる. まずはじめに,

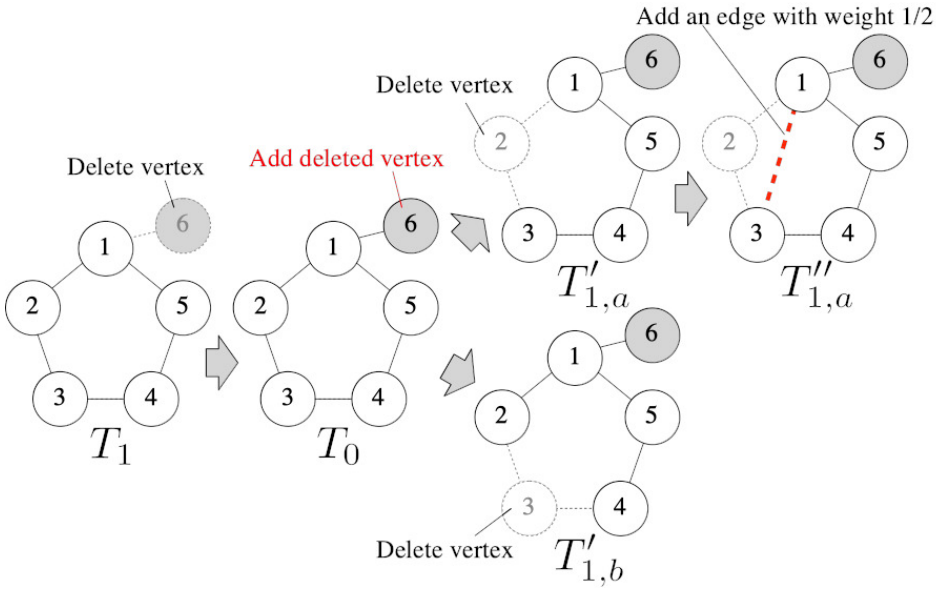

図 6　Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ のとき, 頂点削除によって最大クリークサイズを変更しない頂点を削除する例.
Weighted graph $T_0 = (V_0, E_0, w)$ はこの一例となる. まずはじめに, get_graph_keeping_km$(T_0, \emptyset)$ が (T_1, V_a), $T_1 = (V_1, E_1, w)$, $V_1 = V_0 - V_a$, $V_a = \{v_6\}$ を返しているとする. Weighted graph T_0 から頂点 $v_6 \in V_a$ を削除したとき, T_0 の最大クリークサイズが変化しないかを判断する. 頂点 $v_2 \in V_0 - V_a$ を削除した場合, Weighted graph は $T'_{1,a} = \text{delete\_vertex}(T_0, v_2)$ となる. このとき, $k_m(T'_{1,a}) = k_m(T_0) - 1$ となる. 一方, 頂点 $v_3 \in V_0 - V_a$ を削除した場合, Weighted graph は $T'_{1,b} = \text{delete\_vertex}(T_0, v_3)$ となる. このとき, $k_m(T'_{1,b}) = k_m(T_0) - 1$ となる. このように、頂点集合 V_a に属するいずれの頂点を削除しても, $\mathrm{k_m}()$ の結果が減少することから, get_graph_keeping_km$(T_0, \{v_6\})$ の戻り値は $(T_s, \emptyset)$ となる. よって, Lemma 3.19 により, 頂点 v_6 の削除によって T_0 の最大クリークサイズは変化しない.

`get_graph_keeping_km`$(T_0, \emptyset)$ が (T_1, V_a), $T_1 = (V_1, E_1, w)$, $V_1 = V_0 - V_a$, $V_a = \{v_6\}$ を返しているとする. この例では, $|V_a| = 1$ のため, T_1 に v_a を追加した Weighted graph は元の T_0 となる. Weighted graph T_0 から頂点 $v_6 \in V_a$ を削除したとき, T_0 の最大クリークサイズが変化しないかを判断するために, T_0 から v_6 以外の頂点を削除したときの $\mathrm{k_m}()$ の変化を確認する. 頂点 $v_2 \in V_0 - V_a$ を削除した場合, Weighted graph は $T'_{1,a} =$ `delete_vertex`(T_0, v_2) となる. 頂点 v_2 を削除しただけの Weighted graph $T'_{1,a}$ では, $\mathrm{k_m}(T'_{1,a}) = \mathrm{k_m}(T_0)$ となるが, 関数 `delete_vertex`() によって, 辺 e, $w(e) = 1/2$ が挿入されるため, $\mathrm{k_m}(T''_{1,a}) = \mathrm{k_m}(T_0) - 1$ となる. 一方, 頂点 $v_3 \in V_0 - V_a$ を削除した場合, Weighted graph は $T'_{1,b} =$ `delete_vertex`(T_0, v_3) となる. このとき,

$\mathrm{k_m}(T'_{1,b}) = \mathrm{k_m}(T_0) - 1$ となる．このように、頂点集合 V_a に属するいずれの頂点を削除しても, $\mathrm{k_m}()$ の結果が減少することから, $\texttt{get\_graph\_keeping\_km}(T_0, \{v_6\})$ の戻り値は $(T_0, \emptyset)$ となる．よって, Lemma 3.19 により, 頂点 v_6 の削除によって T_0 の最大クリークサイズは変化しない．

Lemma 3.20. Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ とする． Weighted graph T_d を $(V_r \cup \{v\}, E_r \subset E, w)$, $v \in V_a$ とする． $\texttt{get\_graph\_keeping\_km}(T_d, \{v\}) = (T_s, V_s)$ において, $V_s \neq \emptyset$ のとき, $T_s = \texttt{delete\_vertex}(T_s, v)$ とすると, $\texttt{get\_graph\_keeping\_km}(T_s, \emptyset) = (T_s, \emptyset)$ となる．

Proof. $\texttt{get\_graph\_keeping\_km}(T_s, \emptyset) = (T_t, V_t)$ として, $V_t \neq \emptyset$ と仮定する．このとき, 任意の頂点 $v_t \in V_t$ において, $\mathrm{k_m}(\texttt{delete\_vertex}(T_r, v_t)) = \mathrm{k_m}(T_r)$ となることから, T_r の任意の頂点を削除したとき $\mathrm{k_m}()$ が減少するという前提条件と異なる．よって, $V_t = \emptyset$ となる．以上より, Lemma 3.20 が成立する． □

Lemma 3.21. Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ とする． Weighted graph T_d を $(V_r \cup \{v\}, E_r \subset E, w)$, $v \in V_a$ とする． $\texttt{get\_graph\_keeping\_km}(T_d, \{v\}) = (T_s, V_s)$ において, $V_s \neq \emptyset$ のとき, 頂点集合 V_s に属する頂点の削除によって T_r の最大クリークサイズは変更しない．

Proof. Weighted graph T_r において頂点集合 V_s に属するすべての頂点を削除した Weighted graph を T'_r とする．このとき, $\mathrm{k_m}(T'_r) = \mathrm{k_m}(T_r) - 1$ となる．このとき, 頂点削除順番によらず, 最初の頂点を削除したときに $\mathrm{k_m}()$ が減少することから, 頂点集合 V_s の組み合わせによって, $\mathrm{k_m}()$ が変化することがわかる．また T'_r に頂点 v を加えた Weighted graph T''_r において, $\mathrm{k_m}(T''_r) = \mathrm{k_m}(T_r)$ となることから, 頂点集合 V_s に属する頂点の削除によって T_r の最大クリークサイズは変更しない．よって, Lemma 3.21 が成立する． □

Figure 7 において, Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ のとき, 最大クリークサイズを変更しない頂点を削除する例を示す． Weighted graph $T_0 = (V_0, E_0, w)$ はこの一例となる． まずはじめに, $\texttt{get\_graph\_keeping\_km}(T_0, \emptyset)$ が (T_1, V_a), $T_1 = (V_1, E_1, w)$, $V_1 = V_0 - V_a$, $V_a = \{v_6\}$ を返しているとする．この例では, $|V_a| = 1$ のため, T_1 に v_a を追加した Weighted graph は元の T_0 となる． ここで, $\texttt{get\_graph\_keeping\_km}(T_0, \{v_6\})$ の戻り値は (T'_3, V_s),

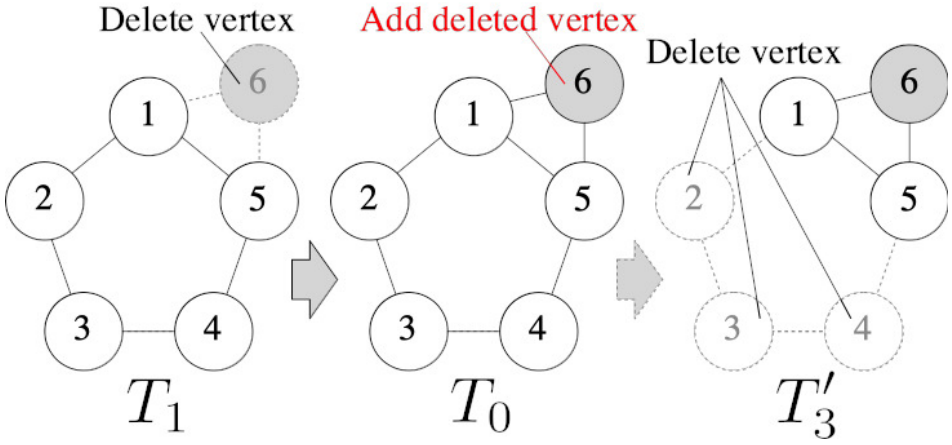

図 7　Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ のとき, 最大クリークサイズを変更しない頂点を削除する例.
Weighted graph $T_0 = (V_0, E_0, w)$ はこの一例となる. まずはじめに, get_graph_keeping_km$(T_0, \emptyset)$ が (T_1, V_a), $T_1 = (V_1, E_1, w)$, $V_1 = V_0 - V_a$, $V_a = \{v_6\}$ を返しているとする. この例では, $|V_a| = 1$ のため, T_1 に v_a を追加した Weighted graph は元の T_0 となる. ここで, get_graph_keeping_km$(T_0, \{v_6\})$ の戻り値は (T'_3, V_s), $V_s = \{v_2, v_3, v_4\}$ となる. このことから Lemma 3.21 により, Weighted graph T_0 から頂点集合 V_s に含まれる頂点を削除しても, T_0 の最大クリークサイズが変化しないことが分かる.

$V_s = \{v_2, v_3, v_4\}$ となる. 頂点集合 $V_s \neq \emptyset$ であることから, 頂点 v_6 を削除することで, 最大クリークサイズが変化する可能性がある. 一方, Lemma 3.21 により, Weighted graph T_0 から頂点集合 V_s に含まれる頂点を削除しても, T_0 の最大クリークサイズが変化しないことが分かる.

Weighted graph $T_r = (V_r, E_r, w)$ が Complete graph ではなく, $V_a \neq \emptyset$ のとき, 頂点削除により最大クリークサイズを 1 つ抽出する擬似コードを Algorithm 5 に示す. 頂点削除は Lemma 3.19, 3.21 に基づいて行う. Line 2–9 において, ループ数はたかだか $|V_r|$ となる. Line 4 において, 関数 `get_graph_keeping_km()` の計算量は $O(n^6)$ となる. この関数の出力 (T_s, V_a) において, T_s のいずれの頂点を削除しても $\mathrm{k_m}()$ が減少する. よって, T_s は関数 get_graph_wo_va() の入力条件を満たす. Line 6,10 において, 関数 get_graph_wo_va() の計算量は $O(n^7)$ となる. よって, 関数 get_graph_w_va() の計算量は $O(n \times (n^6 + n^7) + n^7) = O(n^8)$ となる.

Algorithm 5: Weighted graph T_r が Complete graph ではなく, かつ $V_a \neq \emptyset$ のとき, 頂点削除により最大クリークサイズを 1 つ抽出する.

```
1  Function get_graph_w_va(T_r = (V_r, E_r, w), V_a):
2      foreach v ∈ V_a do
3          T_d ← create graphT_r with v
4          (T_s, V_s) = get_graph_keeping_km(T_d, {v})
5          if V_s ≠ ∅ then
6              T_s ← get_graph_wo_va(T_s)
7              return T_s
8          end
9      end
10     T_r ← get_graph_wo_va(T_r)
11     return T_r
```

Algorithm 6: Input graph T から最大クリークサイズを変化させない頂点を削除することで, 最大クリークを一つ抽出する.

```
1  Function ExtractMaximumCliqueG = (V, E):
2      if V = ∅ then
3          return ∅
4      end
5      if E = ∅ then
6          v ← any vertex in V
7          return {v}
8      end
9      T ← initialize_graph(G)
10     (T_r, V_a)get_graph_keeping_km(T, ∅)
11     if T_r = Complete graph then
12         return vertex set of T_r
13     end
14     if V_a = ∅ then
15         T_r ← get_graph_wo_va(T_r)
16     else
17         T_r ← get_graph_w_va(T_r, V_a)
18     end
19     return vertex set of T_r
```

3.5 Extraction of a maximum clique

この Section では, Weighted graph $T \in \mathscr{T}$ に対して, 最大クリークサイズを変化させない頂点集合を同定して削除する擬似コードを提示し, その計算量が多項式時間となることを示す.

Theorem 3.22. Input graph G から最大クリークを多項式時間で一つ抽出する.

Proof. 最大クリークを一つ抽出する擬似コードを Algorithm 6 に示す. Line 2–9 において, 最大クリークを抽出する準備として, 少なくとも一つの辺 e, $w(e) = 1$ を持

つ Weighted graph を作成する．Line 2–4 において, Input graph G が頂点を持たないとき, $\emptyset$ を出力してアルゴリズムを終了する．Line 5–8 において, Input graph G が辺を持たないとき, V の任意の頂点 v を取得したのち, $\{v\}$ を出力してアルゴリズムを終了する．それ以外の場合において, Line 9 において, Weighted graph T を作成する．関数 `initialize_graph` (Algorithm 1) の計算量は $O(n^5)$ となる．Line 10 において, 関数 get_graph_wo_va() の計算量は $O(n^7)$ となる．Line 15 において, 関数 get_graph_w_va() の計算量は $O(n^8)$ となる．以上より, 本アルゴリズムの計算量は $O(n^5 + n^7 + n^8) = O(n^8)$ となる．よって, Theorem 3.22 が成立する． □

4 Conclusion

本論文では, NP 完全問題の一つである k-Clique 問題の最適化バージョンである, 最大クリーク問題が多項式時間で解けることを証明し, 当該アルゴリズムを擬似コードの形で与えた．アルゴリズムの計算量は $O(n^8)$ であり, 結果として Complexity class NP と P が等しいことが示された．

参考文献

[1] Strum's method for the number of real roots of a polynomial. `https://www.imsc.res.in/~knr/past/sturm/formal_notes.pdf`, 2016. [Online; accessed 06-May-2019].

[2] W. H. Haemers. Interlacing eigenvalues and graphs. *Linear Algebra and its applications*, 226:593–616, 1995.

[3] J. A. Howell. An algorithm for the exact reduction of a matrix to frobenius form using modular arithmetic. i. *mathematics of computation*, 27(124):887–904, 1973.

[4] J. A. Howell. An algorithm for the exact reduction of a matrix to frobenius form using modular arithmetic. ii. *Mathematics of Computation*, 27(124):905–920, 1973.

[5] K. K. Singh and A. K. Pandey. Survey of algorithms on maximum clique problem. *International Advanced Research Journal in Science, Engineering and Technology*, 2(2):18–20, 2015.

[6] A. Storjohann. An o (n 3) algorithm for the frobenius normal form. In *ISSAC*, volume 98, pages 101–104. Citeseer, 1998.

[7] X.-D. Zhang. The laplacian eigenvalues of graphs: a survey. *arXiv preprint arXiv:1111.2897*, 2011.