固有値を利用したインデクス手法の改良

三井　良太† 　片山　 薫†

† 首都大学東京 システムデザイン研究科
E-mail: †mitsui-ryota@sd.tmu.ac.jp, ††kaoru@tmu.ac.jp

あらまし　高橋らは，グラフデータベースに対するインデクス手法としてグラフを行列で表現し，その固有値と Interlace 定理を用いたインデクスを提案した．本稿ではグラフを枝，頂点ラベルを用いて分割しその固有値をインデクスの構築と探索に用いることでインデクス構築時間，探索時間と候補グラフ数の改善を行う．加えてデータベース中のグラフ同士の関係をすべて調べることで探索時間の高速化を図った．

キーワード　インデクス，データ構造，グラフデータ処理，半構造データ

Ryota MITSUI† and Kaoru KATAYAMA†

† Graduate School of System Design,Tokyo Metropolitan University　Asahigaoka 6–6, Hino-shi, Tokyo,191–0065 Japan
E-mail: †mitsui-ryota@sd.tmu.ac.jp, ††kaoru@tmu.ac.jp

1. はじめに

グラフは要素とそれらの関係性などの複雑な構造を簡潔に表すことができ, XML ドキュメント [11] やたんぱく質, 化合物 [12] などを表現するデータ構造として広く利用されている．あるグラフが他のグラフに含まれているかどうかを判定する問題はデータマイニング [6] [9] や化合物データベース [8]，パターン認識 [3] など様々な分野に応用されている重要な問題である．

いずれの分野においても，データベースから欲しいグラフを効率よく検索することが求められる．データベースからのグラフ問い合わせを以下のように表現する．グラフデータベース $D = \{g_1.g_2, \cdots, g_n\}$ と問い合わせグラフ q が与えられたとき, q を部分グラフとして含む全てのグラフ $g_i(g_i \in D)$ を発見する．あるグラフが他のグラフに含まれているかどうかを判定する問題は部分グラフ同型性判定問題と呼ばれ, NP 完全であることが知られている．そのため扱うデータが大きくなるに従って部分グラフ同型同型判定には膨大なコストが必要となる．そこで効率よくデータベースに対する問い合わせを行うため様々なインデクス手法が研究されている．

高橋ら [16] は，グラフを行列で表現しその固有値と Interlace 定理 [5] を用いて，グラフデータベースから問い合わせグラフを含むグラフを効率よく検索するためのインデクスとその利用方法を提案した．Interlace 定理とは, ある実対称行列 A の固有値と A の任意の主部分行列の固有値の間に常にある関係が成立することを示した定理である．行列 A が B の主部分行列であるとき,A の固有値と B の固有値は Interlace 定理を必ず満たし, このとき A の固有値は B の固有値を Interlace するという．しかし,A が B の主部分行列でなくても A の固有値が B の固有値を Interlace してしまう場合がある．またあるグラフを後述する接続行列を用いた行列で表した行列の主部分行列は, 元のグラフの部分グラフを表す．したがってグラフを接続行列を用いた行列で表現し Interlace 定理を用いることで部分グラフでないものを見つけることができる．高橋らは木構造のインデクスを提案した．木のノードそれぞれはデータベースに含まれるグラフを接続行列を用いた行列で表したその固有値を持つ．インデクスは, 子ノードが親ノードを Interlace するように構築する．そうすることであるノードに格納されている固有値と問い合わせのグラフの固有値が Interlace しない場合，そのグラフからはノードまでのグラフは問い合わせグラフと含まないことが分かるため探索を効率よく行える．

宮奥ら [15] はグラフを頂点のラベルで分割し，その固有値をインデクスの探索に用いることで，高橋らのインデクス手法の改良を行った．Interlace 定理を用いたグラフ除去は 2 つのグラフのグラフサイズの差が大きい場合判定精度が低い．グラフをラベルで分割しグラフサイズの差を小さくすることで Interlace 定理を用いた判定の精度が向上する．

本稿では宮奥らの手法の改良として枝ラベル, 頂点ラベルを用いたグラフ分割を提案する．この結果,2 つのグラフサイズの差を更に小さくすることができ, 宮奥らの手法と比較し判定精度が向上した．また行列の次数が大きい場合, その接続行列を用いた行列の固有値の計算には膨大なコストがかかる．本提案で扱う行列が小さくなったことでより大きなグラフを処理でき

るようになった．

提案した手法の評価のため，グラフジェネレータで生成されるグラフデータを用いた実験を行った．実験は部分グラフ問い合わせについて宮奥らの手法と GCode [14] と比較し，インデクスの構築時間，問い合わせ時間，候補グラフ数の 3 つの観点から評価を行った．実験により，提案手法が比較対象に比べ有効であるケースを発見した．

2. 関 連 研 究

2.1 グラフインデクス

グラフデータベースに関する多くのインデクス手法が提案されてきた．パスを用いたインデクスに GraphGrep [9] や 1-index [4], A(k)-index [7],D(k)-index [1] などがある．パスを用いる利点としてインデクスのサイズを前もって知ることができる点である．しかしパスによる表現ではグラフの構造的特長が保持されず，パスの数が膨大になってしまいインデクスの性能が低下してしまうという問題があった．

頻出部分グラフを用いるインデクス手法に gIndex [13] や FG-Index [2] がある．gIndex は頻出部分グラフのなかでも冗長なものを除去したものを用いてインデクスを作成する．扱うデータが同じような分布の場合, 扱うデータベースサイズが大きくなってもインデクスのサイズが大きく変化しない特徴がある．FG-Index もグラフデータベース中の頻出部分グラフをインデクス構築に用いる．問い合わせグラフが頻出部分グラフである場合, インデクス探索によって正解グラフ集合を得ることができる特徴がある．正解グラフ集合とは部分グラフ同型性判定を終えたグラフ集合である．頻出部分グラフを用いたインデクスの場合，頻出部分グラフを予め求めておく必要がある．その際，グラフの頂点数や枝数が大きくなると頻出部分を検索するのに大きなコストが必要となる．よって，頻出部分グラフを用いたインデクスはグラフサイズが大きい場合には適さない．

また GCoding は我々の提案と同じく行列の固有値を用いたインデクスである．グラフの任意の頂点から, その点に隣接する頂点と枝で木を作る．このとき頂点から何個先の頂点までの木を作るかを与える必要があり, この値を大きく取ると正確にグラフ除去を行えるが処理に時間がかかる．この木を隣接行列で表現し, その固有値を用いて GCode と呼ばれるデータを作成する．そのデータをバランス木に格納し GCodingTree と呼ばれるインデクスを構築する．グラフから木を作りその隣接行列の固有値を用いることで誘導部分グラフだけでなく部分グラフでないものを除去することができる．

本研究では性能評価の際, 頻出部分グラフを用いるインデクスよりも大きなグラフを処理できる GCode を比較対象とする．

2.2 部分グラフ同型性判定

部分グラフ同型判定問題は NP 完全であることが知られており, 効率的に処理するためのアルゴリズムが提案されてきた．Ulmann の手法 [10] や VF2 [6] が挙げられる．

Ullmann の手法は，効果的な先見関数とバックトラックと呼ばれる手続きを利用して検索領域を減らす手法である．同型性判定のために，2 つのグラフの各頂点のペアから状態空間木を作成する．その木にバックトラックを基としたアルゴリズムを適用しながら，逐次的にノードを判定していくことで木の検索する範囲を減少させることができる．VF2 も同様にグラフを木構造を用いて表現し, 探索を行う．2 つのグラフの枝の間のマッピングを State Spase Representation でと呼ばれる状態で表し，その状態を更新していくことで 2 つのグラフの間の同型写像を作成する．VF2 は Ullman の手法に比べ使用するメモリ量が少ないという利点があり大規模なグラフを処理することができる．本提案ではインデクス探索から候補グラフを得た後, 部分グラフ同型性判定に VF2 を用いる

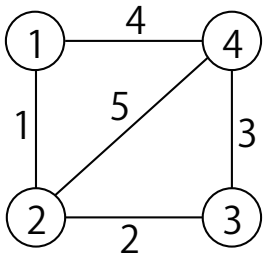

図 1　Graph g_1

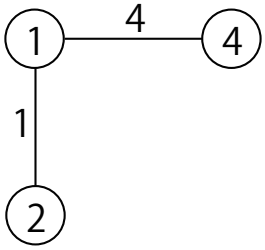

図 2　Graph g_2

3. 準　　備

本稿において，議論の対象はラベル付の連結無向グラフに限定する．以降，ラベル付の連結無向グラフをグラフと呼ぶこととする．

3.1 部分グラフ

グラフ g を $(V, E, L_v, L_e, F_v, F_l)$ と定義する．ここで $V = \{v_1, v_2, \ldots v_n\}$ は頂点の集合であり，$E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j (i, j = 1, 2, \cdots n)\}$ は枝の集合である．また，L_v を頂点ラベルの集合，L_e を枝ラベルの集合とする．$F_v : V \to L_v$, $F_e : E \to L_e$ 各頂点と枝にラベルを割り当てる関数とする．

[**定義** 1]（部分グラフ）2 つのグラフ $g = (V, E, L_v, L_e, F_v, F_l)$ と $g' = (V', E', L'_v, L'_e, F'_v, F'_l)$ が与えられたとき，以下の条件を満たす単射 $f : V' \to V$ が存在すれば g' は g の部分グラフであるという．g' が g の部分グラフであることを $g' \subseteqq g$ と表現する．

（1）　$(f(u), f(v)) \in E$.

（2）　$F'_v(u) = F_v(f(u))$, $F'_v(v) = F_v(f(v))$.

（3）　$F'_e(u, v) = F_e(f(u), f(v))$.

図 1, 図 2 に部分グラフの例を示す．g_2 は g_1 の部分グラフである．

3.2 Interlace 定理

本稿ではグラフを行列で表現しその固有値をもちいてグラフ除去を行う．グラフ除去に用いる Interlace 定理について述べる．

2 つの実数の列 $\alpha_1 \leqq \alpha_2 \leqq \cdots \leqq \alpha_n$ と $\beta_1 \leqq \beta_2 \leqq \cdots \leqq \beta_m$ について考える．ただし $m < n$ とする．以下の式が成り立つとき，後者の数列は前者の数列を Interlace するという．

$$\alpha_i \leqq \beta_i \leqq \alpha_{i+(n-m)} (i = 1, \ldots, m) \tag{1}$$

［**定理** 1］ $S^TS = I$ を満たす $n \times m$ 行列と A, B をそれぞれ n 次,m 次の対称行列とする．B が $B = S^TAS$ を満たすとき B の固有値は A の固有値を Interlace する．

3.3 接続行列を用いたグラフの表現

我々は以下に定義する行列表現を用いる．

［**定義** 2］ (接続行列を用いたグラフの表現)

$$R := \begin{pmatrix} P & N \\ N^t & Q \end{pmatrix}$$

N, P, Q はそれぞれ $|V| \times |E|, |V| \times |V|, |E| \times |E|$ の行列であり以下の通りに定義される．

$$N_{ij} := \begin{cases} 1 & (v_i, v_k) = e_j \in E, \\ 0 & otherwise, \end{cases}$$

$$P_{ij} := \begin{cases} F_v(v_i) & i = j, \\ 0 & otherwise, \end{cases}$$

$$Q_{ij} := \begin{cases} F_e(e_i) & i = j, \\ 0 & otherwise, \end{cases}$$

N は接続行列である．

グラフ g_s と g_l が与えられ g_s が g_l の部分グラフであるとき, g_s と g_l の接続行列を用いた行列 M_{g_s}, M_{g_l} が $M_{g_l} = S^TM_{g_s}S$ を満たし，$S^TS = I$ を満たす S が存在する．このとき M_{g_s} の固有値が M_{g_l} の固有値を Interlace する．

つまり g_s の固有値が g_l の固有値を Interlace しなければ g_s は g_l の部分グラフではない．

以降あるグラフ g_s の固有値が g_l の固有値を Interlace することを $g_s \subseteq_{\overline{=}_i} g_l$ と表す．

［**例** 1］ g_1, g_2 を接続行列を利用した行列表現を用いると以下の行列になる． g_2 は g_1 の部分グラフなので $S^TS = I, M_{g_2} =$

$$M_{g_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 3 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

$$M_{g_2} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 4 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 4 \end{bmatrix}$$

$S^TM_{g_1}S$ を満たす S が存在する．S を以下に示す．

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.4 インデクスを用いたグラフ問い合わせ

n 個のグラフで構成されるグラフデータベース $D = \{g_1, g_2, \cdots, g_n\}$ と問い合わせグラフ q が与えられたとする．グラフ問い合わせとは D の中から q を部分グラフとして含むグラフをすべて見つける処理である．問い合わせの結果得られるグラフ集合は $D_q = \{g : g \in D, q \subseteq g\}$ である．

またインデクスを用いたグラフ問い合わせは以下の 2 ステップで行われる．

（1） グラフデータベース D からインデクスを作成する. 本稿では D 中のグラフを行列で表現し, その固有値を用いることでインデクスを構築する．

（2） 問い合わせグラフ q と (1) で構築したインデクスを用いて, 候補グラフの集合 C_q を得る. 候補グラフとは, サブグラフ問い合わせに於いてはクエリグラフを部分グラフとして含む可能性のあるグラフ．この後 C_q に含まれるグラフに対して, 実際に部分グラフ同型性判定をし,q を含む正解グラフ集合 D_q を得る．本稿では部分グラフ同型性判定に VF2 を用いる．

4. 従 来 手 法

4.1 高橋らのインデクス手法

高橋らは Interlace 定理を用いたインデクスとして Interlace-Tree 提案した．以下に示す 3 つの数列についての命題を用いてインデクスを構築，探索する．

［**命題** 1］ 3 つの数列 α, β, γ が与えられ, まず β が α を Interlace し,γ が β を Interlace するとき γ は α を Interlace する．

上記の命題から以下のことが言える．

［**補題** 1］ 3 つの数列 α, β, γ が与えられ, β が α を Interlace し,γ が α を Interlace しないとき γ は β を Interlace しない．

グラフデータベース D に含まれるグラフについて固有値を計算しその固有値を多分木に格納する．補題 1 よりグラフデータベース D に含まれるグラフ g と問い合わせグラフ q が $q \not\subseteq_{\overline{=}_i} g$ であるならば D 中の $g' \subseteq_{\overline{=}_i} g$ であるすべての g' は q と $q \not\subseteq_{\overline{=}_i} g$ である．そのため $g' \subseteq_{\overline{=}_i} g$ であるすべての g' の格納されるノードを探索する必要はない．このことから InterlaceTree を用いることでクエリグラフの固有値とデータベース中のグラフの固有値の比較回数を減らすことができる．

4.2 宮奥らのグラフ分割手法

Interlace 定理を利用したグラフ除去を行う場合,2 つのグラフのサイズの差が大きくなると部分グラフでない候補グラフが多くなってしまう．そこで宮奥らはグラフを頂点ラベルを用いたグラフ分割を提案した．グラフサイズの差を小さくすることで Interlace 定理による判定の制度を向上させた．宮奥らの頂点ラベルを用いたグラフ分解の定義を以下に示す．

［**定義** 3］ グラフ $g = (V, E, L_v, L_e, F_v, F_l)$ について頂点ラベル l_i^v で分割されたグラフを $g[l_i^v] = (V', E', L'_v, L'_e, F'_v, F'_l)$ で

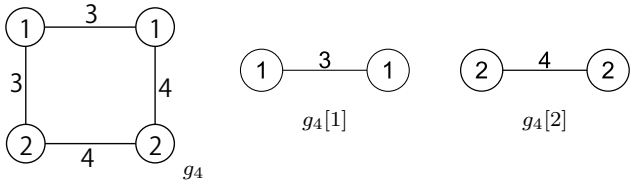

図 3　グラフ g_4 と頂点ラベルを用いて分割されたグラフ

表し，$g[l_i^v]$ は以下の条件を満たす.

- $V' = \{v \in V | F'_v(v) = l_i^v\}$,
- $E' = \{e = (v, v') \in E | F'_v(v) = F'_v(v') = l_i^v\}$,
- $L'^v = \{l_i^v\}$,
- $L'^e = \{F_e(e) \in L^e | e \in E'\}$
- F'_v, F'_e はそれぞれ F_v, F_e 機能を制限したもの.

[**例** 2]　図 3 に g_4 とその分割グラフを示す.

宮奥らはあるグラフ g とその分割グラフ $g[l_i^v]$ を用いて Interlace 定理による判定を行い, 高橋らのインデクス手法にくらべ判定の精度を向上させることができた．あるグラフ g_s と g_l が与えられ,g_s が g_l の部分グラフであるときラベル i で分割した $g_s[l_i^v]$ は $g_l[l_i^v]$ の部分グラフになっている．つまり g_s が g_l の部分グラフかどうかを調べたいときに g_s が g_l の固有値を比較するだけでなく $g_s[l_i^v]$ が $g_l[l_i^v]$ を Interlace するかどうかを調べる．$g_s[l_i^v] \not\subseteq_i g_l[l_i^v]$ であれば分割グラフの固有値を調べるだけで g_s が g_l の部分グラフでないことがわかる．問題点として扱うグラフが増え, 固有値計算時間が増加する点である.

5. 提 案 手 法

5.1　頂点, 枝のラベルを用いたグラフ分割手法

宮奥らの手法改善のため枝ラベルを用いた分割を以下に定義する.

[**定義** 4]　グラフ $g = (V, E, L_v, L_e, F_v, F_l)$ について枝ラベル l_i^v で分割されたグラフを $g[l_i^e] = (V', E', L'_v, L'_e, F'_v, F'_l)$ で表し，$g[l_i^v]$ は以下の条件を満たす.

- $V' = \{v \in V | e = (v, v') \in E'\}$,
- $E' = \{e \in E | F'_e(e) = l_j^e\}$,
- $L'^v = \{F_v(v) \in L^v | v \in V'\}$,
- $L'^e = \{l_j^e\}$
- F'_v, F'_e はそれぞれ F_v, F_e 機能を制限したもの.

グラフ $g[l_j^e][l_i^v] = (V'', E'', L''^v, L''^e, F''_v, F''_e)$ は枝ラベル $[l_j^e]$ で分解し，頂点ラベル $[l_i^v]$ で分解されたグラフを意味する.

以降, あるグラフ g のすべての頂点, 枝ラベルで分割されたグラフの固有値の集まりを $AE(g)$ と表す.

[**例** 3]　図 4 に g_4 とその分割グラフを示す.

分割グラフを用いた Interlace 定理を用いた判定を以下に定義する.

[**定義** 5]　グラフ $g_1 = (V_1, E_1, L_1^v, L_1^e, F_1)$ の分割グラフ $g_1[l_i^e]$，$g_1[l_i^e][l_j^v]$，$g_2 = (V_2, E_2, L_2^v, L_2^e, F_2)$ の分割グラフ $g_2[l_m^e]$，$g_2[l_m^e][l_n^v]$ が与えられ，以下の条件を満たすことを $AE(g_1) \subseteq_{ia} AE(g_2)$ と表す.

（1）　$L_1^v \subseteq L_2^v$，$L_1^e \subseteq L_2^e$

（2）　$Eig(g_1[l^e][l^v])$ が $Eig(g_2[l^e][l^v])$ すべての $l^e \in L_1^e$, $l^v \in L_1^v$ において Interlace する

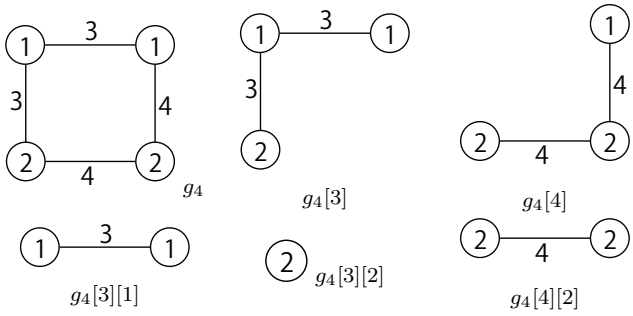

図 4　グラフ g_4 とその枝, 頂点ラベルを用いて分割されたグラフ

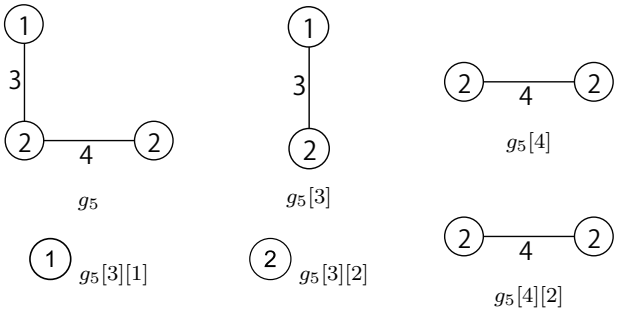

図 5　グラフ g_5 とその枝, 頂点ラベルを用いて分割されたグラフ

g_1 の固有値が g_2 の固有値を Interlace しない場合，g_1 が g_2 の部分グラフでないことは明らかである．g_1 と g_2 の同じラベルで分割したグラフが Interlace しない場合，そのことからも g_1 が g_2 の部分グラフでないことが判る．したがって分割グラフを Interlace 定理の判定に用いることで扱うデータが小さくなったことによる処理時間の軽減と，2 つのグラフサイズの差が小さくなることによる Interlace 定理によるグラフ除去性能の向上させることができる.

[**例** 4]　図 5, 図 6 に g_5 と g_6，その分割グラフを示す．g_5 は g_4 の部分グラフである．$g_5[3][1]$,$g_5[3][2]$,$g_5[4][2]$ はそれぞれ $g_4[3][1]$,$g_4[3][2]$,$g_4[4][2]$ を Interlace する.

一方，g_6 と g_4 を比較したときに，$g_6[3][2] \not\subseteq_i g_4[3][2]$ である．このことから g_4 と g_6 の固有値を比較することなく g_6 は g_4 の部分グラフでないことが判る.

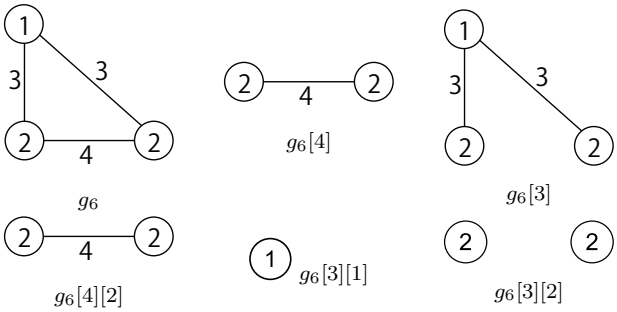

図 6　グラフ g_6 とその枝, 頂点ラベルを用いて分割されたグラフ

5.2　インデクス手法

高橋らの InterlaceTree の改良としてグラフデータベース中のすべてのグラフ同士の関係を調べたインデクスを提案する．すべてのグラフ同士の関係を調べることで InterlaceTree に比

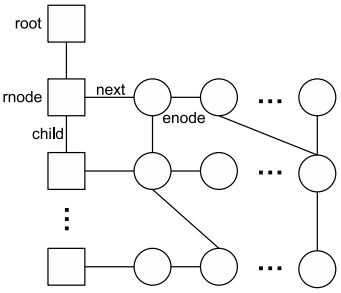

図 7　インデクス構築例

Algorithm 1 ConstructIndex(D)
Input: グラフデータベース D
Output: インデクス
1: *root* ノード作成
2: **for** each $g \in D$ **do**
3: 　SortNode(*root*,g)
4: **end for**
5: CreateEdges(*root*)

ベインデクスの枝の増えたデータ構造になる．InterlaceTree の特徴からあるノードのグラフを問い合わせグラフが Interlace しない場合, そのノードから葉ノードまでのグラフはすべて Interlace 定理による判定を行わずに問い合わせグラフを含まないことが判る．InterlaceTree よりも枝が増えているため, 提案のインデクスは Interlace 定理による判定を行わなければならないノードが減り, 探索を高速に行うことができる．

5. 2. 1　インデクスの構築

図 7 にインデクスの構築例を示す．インデクスは *rnode* と *enode*2 種類のノードで構成される．それぞれ図 7 のように *child*，*next* の 2 方向に枝が伸びる．あるノード *nod* の *child* 方向に接続された i 番目のノードを $child(nod)_i$，*next* 方向に接続されるノードを $next(nod)$ と表すこととする．まずインデクス構築のためデータベース中のすべてのグラフをその頂点と枝の和 $|V+E|$ の順に整理する．*rnode* は $|V+E|$ の値を持ち,$next(rnode)$ はその $|V+E|$ サイズのグラフ g と $AE(g)$ が *next* 方向に *enode* として順次格納される．$Num(g)$ は g の $|V+E|$ を表す．$child(rnode)$ には *rnode* よりも小さな $|V+E|$ を指定し, 同様に *next* 方向にグラフを格納する．次に *enode* に枝を張る．ある *rnode* に連なる *enode* よりも $V+E$ が小さい *rnode* に連なるすべての *enode* と Interlace 定理の条件式を判定する．Interlace した場合 $child(enode)_i$ とする．この処理をすべての *rnode* について行う．

アルゴリズム 1，2，3，4 にインデクス構築のアルゴリズムを示す.

[**例** 5]　グラフ g_1, g_2, g_3 についてインデクスを構築する．それぞれ以下のような関係になっている．

（1）　$AE(g_1) \subseteq_{ia} AE(g_2)$
（2）　$AE(g_1) \subseteq_{ia} AE(g_3)$
（3）　$AE(g_2) \not\subseteq_{ia} AE(g_3)$
（4）　グラフ g_1, g_2, g_3 の頂点と枝の数の和 $|V+E|$ はそれぞれ 4, 4, 3

インデクス構築例を図 8 に示す．2 つの *rnode* にそれぞれ 4, 3 の値を持ちその *next* 方向にグラフを格納したノードを持つ.

Algorithm 2 SortNode(*nod*,g)
Input: インデクスのある *rnode nod*, グラフ g
Output: インデクス
1: $AE(g) \leftarrow$ComputeDecomposedEigenvalue(g)
2: **if** *nod* が $child(nod)$ を持たない **then**
3: 　*cnod* を $child(nod)$ として作成する
4: 　g の頂点と枝の数の和,$Num(g)$ を *cnod* に格納する
5: 　*nnod* を $next(cnod)$ として作成する
6: 　*nnod* に $AE(g)$ を格納する
7: **else if**　$Num(g_{nod} > Num(g)$ **then**
8: 　SortNode($child(nod)$,g)
9: **else if**　$Num(g_{nod} < Num(g)$ **then**
10: 　*cnod* を $child(nod)$ かつ $parent(child(nod))$ として作成する
11: 　g の頂点と枝の数の和,$Num(g)$ を *cnod* に格納する
12: 　*nnod* を $next(cnod)$ として作成する
13: 　*nnod* に $AE(g)$ を格納する
14: **else if**　$Num(g_{nod}) == Num(g)$ **then**
15: 　$next(cnod(next))$ がなくなるまで読み進め, そのノードの *next* として *nnod* を作成する
16: 　*nnod* に $AE(g)$ を格納する
17: **end if**

Algorithm 3 CreateEdges(*nod*))
Input: インデクスのある *rnode nod*
Output: インデクス
1: $enode \leftarrow next(child(nod))$
2: **while** *enode* が存在する **do**
3: 　$rnode \leftarrow child(child(nod))$
4: 　**while** *rnode* が存在する **do**
5: 　　$enode2 \leftarrow next(rnode))$
6: 　　**while** *enode*2 が存在する **do**
7: 　　　**if** $AE(g_{enode2}) \subseteq_{ia} AE(g_{enode})$ **then**
8: 　　　　*enode*2 を $child(enode)$ とする
9: 　　　**end if**
10: 　　　$enode2 \leftarrow next(enode2)$
11: 　　**end while**
12: 　　$rnode \leftarrow child(rnode)$
13: 　**end while**
14: 　$enode \leftarrow next(enode)$
15: **end while**
16: CreateEdges($child(nod)$)

Algorithm 4 ComputeDecomposedEigenvalue(g)
Input: $g = (V, E, L_v, L_e, F_v, F_l)$,
Output: 頂点と枝ラベルで分割されたグラフの固有値の集合 $AE(g)$
1: **for** $j = 1$ **to** $|L^e|$ **do**
2: 　**for** $i = 1$ **to** $|L^v|$ **do**
3: 　　add $Eig(g[l^e_j][l^v_i])$ to $AE(g)$
4: 　**end for**
5: **end for**
6: **return**　$AE(g)$

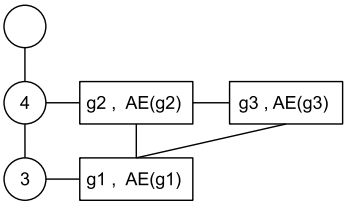

図 8　インデクス構築例

Algorithm 5 ProcessQuery(q)

Input: グラフデータベース D

Output: 候補グラフ集合 C_q

1: ScanIndex($root$,q)

Algorithm 6 ScanIndex(nod,g)

Input: インデクスのある $rnode$ nod, 問い合わせグラフ q

Output: 候補グラフ集合 C_q

```
1: while nod が存在する do
2:    enode ← next(nod)
3:    while enode が存在する do
4:       if q⊆_ia enode then
5:          g_enode を C_q へ格納
6:       else
7:          enode のすべての子ノードを探索対象から除去
8:       end if
9:       enode ← next(enode)
10:   end while
11:   nod ← child(nod)
12: end while
```

$AE(g_1)\subseteq_{ia} AE(g_2)$, $AE(g_1)\subseteq_{ia} AE(g_3)$ であるので g_1 を格納したノードは g_2 と g_3 の子ノードとなる．

5.2.2　インデクスの探索

生成したインデクスに対する問い合わせ方法について述べる．$root$ を始点としてインデクスを探索していく．ある $rnode$ に連なる $enode$ すべてと問い合わせグラフ q について Interlace 定理の条件式を判定する．補題 1 よりインデクスのあるノードに含まれるグラフ g と問い合わせグラフ q が $q\not\subseteq_i g_{nod}$ であるならばすべての $child(nod)_i, (i = 1, \cdots)$ は q を含まない．本インデクスは高橋らのインデクスに比べあるノードの $child(nod)$ が多い．したがって $q\not\subseteq_i g_{nod}$ となったとき調べなくてよいノードが多くなるため処理を高速化できる．ある $rnode$ に連なる $enode$ すべてと条件式の判定が終われば $child(rnode)$ についても同様の処理をする．この際, 上記の処理で問い合わせグラフを含まないことがわかっているノードの Interlace 定理の条件式の判定は飛ばす．これを $rnode$ がなくなるまで行う．インデクスへの問い合わせアルゴリズムをアルゴリズム 5，6 に示す．

［**例** 6］　問い合わせグラフ q が与えられする．構築で示した例のインデクスを探索する．q と g_1,g_2,g_3 それぞれ以下のような関係になっている．

（1）　$AE(q)\not\subseteq_{ia} AE(g_1)$
（2）　$AE(q)\subseteq_{ia} AE(g_2)$
（3）　$AE(q)\not\subseteq_{ia} AE(g_3)$

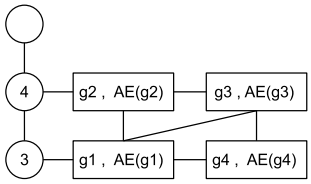

図 9　g_4 を追加したインデクス

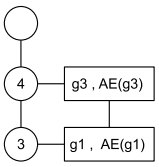

図 10　g_2 の削除されたインデクス

まず $root$ 直下の $rnode$ に連なる $enode$ に含まれるグラフと Interlace 定理の条件式を判定する．$AE(q)\subseteq_{ia} AE(g_2)$ なので g_2 を候補グラフ集合 C_q へ．つぎに g_3 と q を調べる．$AE(q)\not\subseteq_{ia} AE(g_3)$ であるので g_3 は q を部分グラフとして含まない．また $child(g_3)$ である g_1 も q を部分グラフとして含まない．これで 4 の $rnode$ の探索は終了したので 3 の $rnode$ に連なる $enode$ を調べる．しかし g_1 は g_3 との判定で部分グラフとして q を含まないことが判っているので Interlace 定理の条件式の判定を行わない．$rnode$ がこれ以上ないので探索を終了する．

5.2.3　インデクスへのグラフの追加と削除

提案のインデクスへのグラフの追加と削除について述べる．提案のインデクスはグラフの追加と削除が可能である．グラフを追加する場合は構築と同じ手順で行えばよい．ただし, インデクス中のグラフ全てと Interlace 定理の条件式を判定する必要があるので, グラフ数が大きいとコストが大きくなる可能性がある．インデクスに格納されているグラフを削除する場合は削除したいグラフの格納されているノード nod を探索し, $next(nod)$ が存在しない場合は nod を削除すればよい．$next(nod)$ が存在する場合は $next(nod)$ を nod を $next$ にもっていたノードの $next$ に指定する．

［**例** 7］　例 11 のインデクスに g_4 を追加する．ただし g_4 は以下の条件を満たす．

（1）　$AE(g_4)\not\subseteq_{ia} AE(g_2)$
（2）　$AE(g_4)\subseteq_{ia} AE(g_3)$
（3）　g_4 の頂点と枝の数の和 $|V + E|$ は 3

g_4 の頂点と枝の数の和 $|V + E|$ は 3 なので g_1 の $next$ に接続する．$AE(g_4)\not\subseteq_{ia} AE(g_2)$ であるので g_2 とは枝を張らない．$AE(g_4)\subseteq_{ia} AE(g_3)$ であるので g_4 を g_3 の子ノードとする．g_4 の追加されたインデクスを図 9 に示す．

［**例** 8］　例 11 のインデクスから g_2 を格納するノードを削除する．まず $root$ から g_2 を格納するノードを探す．このノードは $next$ 方向にノードを持つので 4 を格納する $rnode$ と g_3 を格納するノードを接続し g_2 を格納するノードを削除する．図 10 に g_2 を削除したインデクスを示す．

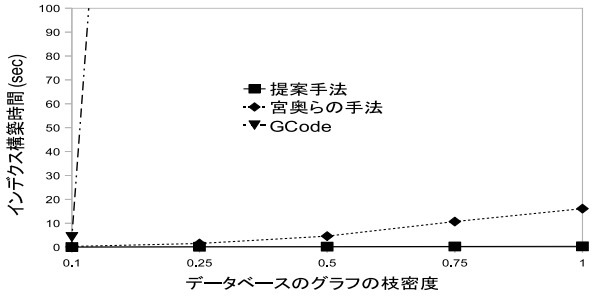

図 11　インデクス構築時間 (sec)

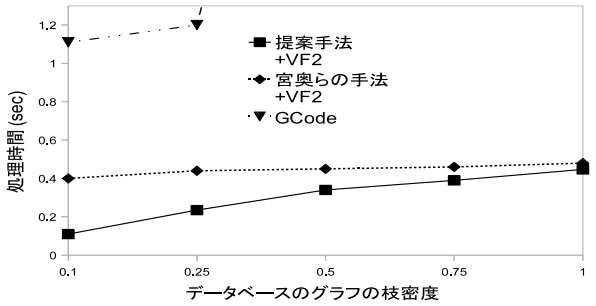

図 12　処理時間 (sec)

6. 実験と評価

提案したインデクスの性能を評価し, 提案手法が有効な条件を検証する．実験環境は Core i7 2.80GHz,16GB RAM 上で行う．OS は Windows 7 professional を用いる．

実験データとして, 頂点数, 枝数, ラベル数を設定した人工グラフデータをランダムに生成して実験に用いる．データベースのグラフと問い合わせのグラフを作成しインデクスを構築し問い合わせ処理を行う．データベースのグラフ, 問い合わせグラフはともにランダムに 1000 個作成する．問い合わせ時間は 1000 個の処理時間の合計を用いる．候補グラフ数は 1000 個のグラフの平均をデータとして扱う．インデクス構築時間, インデクス探索時間, 候補グラフ数の 3 つの観点からインデクスを評価する．

6.1　枝密度の変化

比較対象は宮奥らの InterlaceTree と GCode を用いる．データベースのグラフの平均頂点を 20, 枝の密度を 0.1 から 1 まで変化させる．枝密度 0.1 は張りうる枝の数の 10 %を, 枝密度 1 は張りうる枝すべて張った状態を指す．問い合わせグラフの平均頂点を 10, 枝密度は 0.5 を用いる．部分グラフ同型性判定には VF2 を用いる．インデクス構築時間, インデクス探索と部分グラフ同型性判定に掛かった処理時間, 候補グラフ数をそれぞれ図 11, 図 12, 図 13 に示す.

図 11 から提案手法が宮奥らの手法,GCode と比べ高速に, インデクスを構築できていることが判る．GCode はグラフを深さを指定し木で表し, その隣接行列を扱う．そのためグラフの枝の密度が大きくなれば木が複雑になり処理に時間がかかる．宮奥らと我々の扱う行列表現のサイズも $|V+E|$ であり枝が増えれば行列のサイズも大きくなり, その固有値の計算には時間

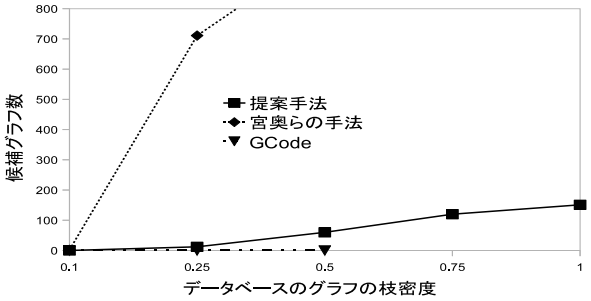

図 13　候補グラフ数

がかかってしまう．しかし,GCode よりも高速に処理できているため枝の多い複雑なデータに対して提案手法が有効であるといえる．また提案手法と宮奥らの手法を比較すると宮奥らの手法に比べ小さい行列を大量に扱う．先も述べたとおり n 次の行列の固有値の計算コストは $O(n^2)$ である．つまり大きなグラフの固有値を一つ求めるよりも小さい行列をいくつか扱ったほうが高速に固有値を計算できる．そのため宮奥らの手法に比べ高速にインデクスを構築できた．図 12 にインデクス探索時間を示す．インデクスで扱うデータが複雑なほどその探索にも時間がかかる．提案手法と宮奥らの手法はインデクス探索で得られた候補グラフ集合に対して VF2 で部分グラフ同型性判定をかけた時間も加えてある．提案手法が正解グラフを得るまでの時間がもっとも速いことが判る．なお, 枝密度が増えるにつれ提案手法と宮奥らの手法の探索時間が近づいているが枝の密度は 1 が最大なので逆転することはない．図 14 からインデクスのグラフ除去性能は GCode が最も高い．しかし処理に膨大な時間がかかるので正解グラフを得られる時間は提案手法が勝っている．また GCode に関しては枝密度が 0.75 以降はインデクス構築に時間がかかり実験を行えなかったが, 枝密度 0.5 までの傾向から提案手法が GCode に比べ高速に正解グラフを得られるといえる．

6.2　グラフの大きさの変化

扱うグラフの行列表現では行列のサイズが頂点と枝の和 $|V+E|$ となる．扱うグラフの $|V+E|$ の平均を 1000 から 10000 まで変化させ, 結果がどのように変化するかを観る．$|V|:|E|=3:7$ でグラフを作成した．$|V+E|$ =1000 であれば $|V|$ =300,$|E|$ =700 である．問い合わせグラフの $|V+E|$ の平均は 500 とした．枝密度を変化させる実験よりも大きなグラフを扱っているため宮奥らの手法と GCode では処理に膨大な時間がかかるため提案手法のみの実験とする．

図 14 のようにインデクス構築時間が増加する．これは固有値計算コストが $O(n^2)$ であることに依存する．図 15 からインデクスを探索する時間は線形時間で行えることがわかる．また $|V+E|$ を増やしていくにつれ問い合わせグラフサイズとデータベースのグラフのサイズの差が大きくなるので候補グラフ数は増えてしまう．

7. 結　　論

本稿では, グラフデータベースに対するサブグラフ問い合わ

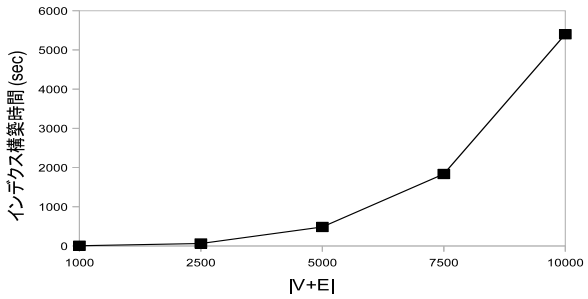

図 14　インデクス構築時間 (sec)

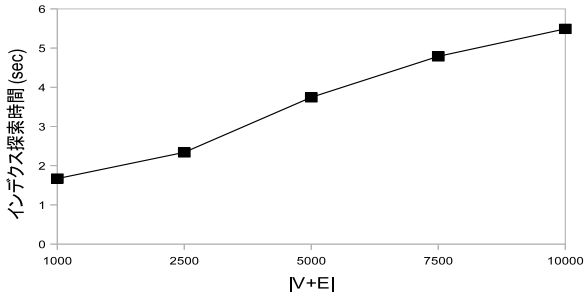

図 15　インデクス探索時間 (sec)

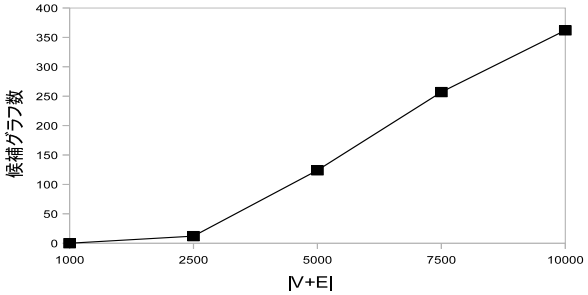

図 16　候補グラフ数

せの効率化ために, 頂点と枝を用いたグラフ分割と Interlace 定理を用いたインデクスの改良を行った.

実験の結果, 提案のインデクスがインデクス構築時間, 探索時間, グラフ除去性能で宮奥らの手法よりも優れていることを確認できた. GCode との比較ではインデクスのグラフ除去性能では劣っているもののインデクス構築時間と部分グラフ同型性判定まで行ったの時間で GCode よりも提案が高速に処理できることを確認できた.

また我々の用いている接続行列を用いたグラフの行列表現は行列の次数が頂点と枝の数の和になる. 頂点数 n のグラフが張りうる枝の数は $n(n-1)/2$ である. そのため頂点数が大きくなり枝が密に張られたグラフを行列で表現すると非常に大きくなり固有値計算に膨大な時間が掛かる. 今後の課題としてこの問題解決のためグラフの新しい行列表現の提案が必要であると考える.

文　献

[1] Q. Chen, A. Lim, and K.W. Ong. D (k)-index: An adaptive structural summary for graph-structured data. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 134–144. ACM, 2003.

[2] J. Cheng, Y. Ke, W. Ng, and A. Lu. Fg-index: towards verification-free query processing on graph databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 857–872. ACM, 2007.

[3] M.F. Demirci, R.H. van Leuken, and R.C. Veltkamp. Indexing through laplacian spectra. *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 312–325, 2008.

[4] A. Fiat. Online algorithms: The state of the art (lecture notes in computer science). 1998.

[5] W.H. Haemers. Interlacing eigenvalues and graphs. *Linear Algebra and its Applications*, Vol. 226, pp. 593–616, 1995.

[6] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. *Principles of Data Mining and Knowledge Discovery*, pp. 13–23, 2000.

[7] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 129–140. IEEE, 2002.

[8] E.G.M. Petrakis and A. Faloutsos. Similarity searching in medical image databases. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 9, No. 3, pp. 435–447, 1997.

[9] D. Shasha, J.T.L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 39–52. ACM, 2002.

[10] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, Vol. 23, No. 1, pp. 31–42, 1976.

[11] H. Wang, J. Li, J. Luo, and H. Gao. Hash-base subgraph query processing method for graph-structured xml documents. *Proceedings of the VLDB Endowment*, Vol. 1, No. 1, pp. 478–489, 2008.

[12] P. Willett, J.M. Barnard, and G.M. Downs. Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, Vol. 38, No. 6, pp. 983–996, 1998.

[13] X. Yan, P.S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 335–346. ACM, 2004.

[14] L. Zou, L. Chen, J.X. Yu, and Y. Lu. A novel spectral coding in a large graph database. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pp. 181–192. ACM, 2008.

[15] 宮奥祥多, 片山薫. ラベルによるグラフ分割を用いた固有値に基づくグラフ索引手法. 第 2 回データ工学と情報マネジメントに関するフォーラム, 兵庫,2010,3, 3 2010.

[16] 高橋俊介, 片山薫. グラフ索引のための interlace 定理によるグラフの包含関係を考慮したデータ構造の提案. 情報処理学会第 2 回データ工学と情報マネジメントに関するフォーラム", 宮崎,2008,3, 3 2008.