区間木を用いた固有値に基づくグラフのインデクシング手法

永井　貴博† 片山　薫†† 石川　博††

† 東京都立大学工学部　〒 192–0397 東京都八王子市南大沢 1–1
†† 首都大学東京システムデザイン学部　〒 192-0397 東京都八王子市南大沢 1–1
E-mail: †takahiro-nagai@c.metro-u.ac.jp, ††{katayama,ishikawa}@eei.metro-u.ac.jp

あらまし　今日グラフは，情報学だけでなく化学や生物学など様々な分野で広く応用されている．グラフデータベースに蓄えられた大量のグラフ集合から，質問となる入力グラフを部分グラフとして持つものを見つけ出すことが重要となっている．部分グラフ同型判定問題は NP 完全に属するため，大量のグラフ集合に対する部分グラフ同型判定には膨大な計算コストを必要とする．よって，膨大な計算を実施する前に，他の手段を用いてグラフ集合から入力グラフを部分グラフとして持たないものを除くことが有効である．グラフの固有値を用いて，2 つのグラフ間における部分グラフ同型の関係を示す Cauchy のインタレース定理が知られている．このインタレース定理によって，グラフ集合の中のグラフから入力グラフを部分グラフとして持たないものを除くことができる．そこで本稿では，このインタレース定理を用いて，幾何学的データ構造である区間木を利用したインデクシング手法を提案する．さらに区間木インデクスを用いることによる有効性を示す．

キーワード　部分グラフ同型，インタレース定理，区間木

An Indexing Method for Graphs Based on Eigenvalues Using Interval–tree

Takahiro NAGAI†, Kaoru KATAYAMA††, and Hiroshi ISHIKAWA††

† Faculty of Engineering, Tokyo Metropolitan University Minamiosawa 1–1, Hachioji-shi, Tokyo, 192–0397 Japan
†† Faculty of System Design, Tokyo Metropolitan University Minamiosawa 1–1, Hachioji-shi, Tokyo, 192–0397 Japan
E-mail: †takahiro-nagai@c.metro-u.ac.jp, ††{katayama,ishikawa}@eei.metro-u.ac.jp

Abstract　Graph is widely applied in various fields like not only informatics but also chemistry and biology, etc. It is important to find out the graphs which contain the input subgraph quickly in graph database. A huge computational cost is needed for the subgraph isomorphism problem as it belongs to NP-complete. Removing the graphs don't contain input graph by another costless method is effective. The Cauchy's interlace theorem which shows the relation of subgraph isomorphism because of the eigenvalues in the graph is known for the problem. This theorem can be removed graphs don't contain the input graph as subgraph in graph set. We propose an indexing method using a geometrical data structure Interval-tree by using this interlace theorem. We also evaluate our proposition.

Key words　Subgraph Isomorphism, Interlace Theorem, Interval-tree

1. はじめに

データベース検索は，巨大で複雑な木構造データやグラフデータなどの出現によってますます困難になっている．グラフは情報学だけでなく，化学や生物情報学などの分野でも広く応用されている [1] [2] [3]．化学の分野においては，新しく発見もしくは合成された化学分子の構造や特性を調べ，分類して登録する必要がある．実際にアメリカのメディシンライブラリではユーザが分子を問い合わせ，検索するデータベースシステムがある．そしてグラフデータベースは化学や生物情報学だけにとどまらず，情報学の分野でもコンピュータビジョン [8] やパターン認識 [9] に使用されている．

しかし，2 つのグラフ間の包含関係を調べる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いる．大量なグラフデータを扱う際には膨大な計算コストを必要とする．Messsmer [4] らの提案する部分グラフ同型判定法といったグラフの構造を用いた研究もされており，計算コストの短縮が求められている．

そこで本稿では，グラフの構造ではなく，グラフの隣接行列表現から求まる固有値に着目し，固有値に基づく Cauchy のインタレース定理 [7] を部分グラフ同型判定問題に適用したインデクシング手法を提案する．インデクスとしては幾何学的データ構造である区間木 [10] を利用し，グラフの隣接行列表現から

求まる各固有値間を区間と見なして，すべての区間を木に蓄えてインデクスを作る．そのインデクスに対して入力グラフの固有値を問い合わせることにより，直接的にグラフ間の構造を比較することなく，インデクスを用いて固有値の比較をすることのみで, 部分グラフになるための必要条件を調べて枝刈りすることができる．木に蓄えられている区間が報告されなければ, そのことで数値比較をぜずに部分グラフを含む候補とはならないためである．一つの入力グラフに対して，様々な大きさの大量のグラフ集合から部分グラフにならない候補を枝刈りする際に有効であり，本稿では大量のグラフ集合に対して一つの入力グラフの関係を想定している．

2. グラフと固有値の関連事項

2.1 誘導部分グラフとその行列表現

グラフとはノード集合 V とエッジ集合 E からなる．図 1 に示すあるラベル付き有限無向グラフ $G = (V, E)$ を考える．G の頂点 $u, v \in G$ の間が枝で接続されているときには枝のラベルを，接続されていないときは 0 を，$V \times V$ の正方行列 A の (u, v) 成分にそれぞれ割り当ててできた行列はあるグラフ G の行列表現となる．対角成分には頂点のラベルがそれぞれ入っている．本稿では，この行列表現を隣接行列表現と呼ぶことにする．図 1 のように頂点 1 と頂点 2 が接続されていれば行列 A の $(1, 2)$ 成分と $(2, 1)$ 成分が a となり，頂点 1 と頂点 4 は連結ではあるが直接接続されていないので行列 A の $(1, 4)$ 成分と $(4, 1)$ 成分は 0 となる．上記からグラフの行列表現は正方行列であり対称行列でもある．

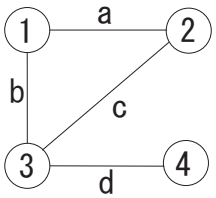

図 1　グラフ G

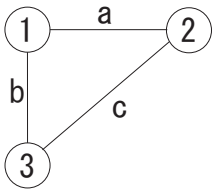

図 2　グラフ G の誘導部分グラフ H

$$G = \begin{bmatrix} 1 & a & b & 0 \\ a & 2 & c & 0 \\ b & c & 3 & d \\ 0 & 0 & d & 4 \end{bmatrix} \tag{1}$$

$$H = \begin{bmatrix} 1 & a & b \\ a & 2 & c \\ b & c & 3 \end{bmatrix} \tag{2}$$

グラフ G における誘導部分グラフ H は図 2 を指す．その隣接行列表現は式 (2) である．誘導部分グラフ H とはグラフ G においていくつかのノードを取り出したときに，それらのノードに関わるすべてのノードとエッジを取り出したグラフを言う．この誘導部分グラフ H はグラフ G の部分グラフである．さらに，グラフ G の部分グラフとしては図 3 も考えられる．しかし，これは誘導部分グラフではなく一般の部分グラフとなる．なぜなら，グラフ G からノード $1, 2, 3$ を取り出したにも関わらず，エッジ c がないためである．

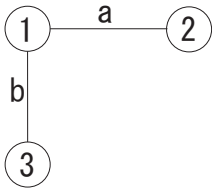

図 3　グラフ G の誘導部分グラフではない一般の部分グラフ X

$$X = \begin{bmatrix} 1 & a & b \\ a & 2 & 0 \\ b & 0 & 3 \end{bmatrix} \tag{3}$$

ここで，図 2 に示した誘導部分グラフ H の隣接行列表現はあるグラフ G の主部分行列 (Principal Submatrix) となるが，図 3 に示した一般の部分行列 (式 (3)) はあるグラフ G の部分行列とはならないことに注意する必要がある．その場合には，グラフの接続行列を利用した行列表現を考えることで同様に処理することができる [16].

グラフの固有値は隣接行列表現に変換後，求める必要がある．すべての議論は隣接行列表現から固有値が求まっているという前提で進めていく．

2.2 Cauchy のインタレース定理

隣接行列の固有値は必ず実数である．計算される固有値の数はグラフ G の頂点数である．以下の Cauchy のインタレース定理を示す．$n \times n$ の正方行列 A は

$$A = \begin{bmatrix} H & B \\ B^* & U \end{bmatrix} \tag{4}$$

H は $m \times m$ の A の主部分行列 (principal submatrix)，ただし $m < n$ である．正方行列 A と H の固有値の組を次のように表すと，

$$\begin{cases} Az_i = z_i\alpha_i, \quad i = 1, \ldots, n, \quad \alpha_1 \leqq \alpha_2 \leqq \cdots \leqq \alpha_n, \\ Hy_i = y_i\theta_i, \quad i = 1, \ldots, m, \quad \theta_1 \leqq \theta_2 \leqq \cdots \leqq \theta_m, \end{cases} \tag{5}$$

次の定理が成り立つ．

$$\alpha_k \leqq \theta_k \leqq \alpha_{k+(n-m)}, \qquad k = 1, \ldots, m, \tag{6}$$

この定理が意味することは，正方行列 A の固有値の最大値から最小値までの幅は H の固有値の最大値から最小値までの幅よりも広く，かつそれぞれの固有値の組において，A の固有値の範囲に H の固有値が含まれなければならないということである．関係式 (6) はあるグラフ G の部分グラフになるための必要条件であり，定理を満たしているからといって部分グラフになるとは限らない．この関係式 (6) を用いることによって，正方行列 A のグラフと H の誘導部分グラフの関係で部分同型判定を行うことができる．

3. 提案手法の概要

3.1 線形問い合わせの問題点

線形問い合わせとは先ほど示した Cauchy のインタレース定理 (式 (6)) を全て数値比較する方法である．すなわち大規模なグラフ集合に蓄えられているグラフの固有値関係をすべて調べる必要がある．大規模なグラフ集合の中には，入力グラフ Q より頂点数が少なく部分グラフとなりえないものも多く存在する．つまり，入力グラフ Q の固有値の最大値から最小値までの区間より狭い区間を持つ候補グラフが存在したら，そのグラフは部分グラフの候補となりえないということである．しかし線形問い合わせでは一度固有値比較を実施し，インタレース定理の示す範囲に入力グラフ Q の固有値を含んでいるか調べ，少なくとも一回は問い合わせることになり，グラフ集合に含まれるグラフ数を n とすると最悪の場合 $O(n)$ の計算コストが必要になってしまう．さらに，グラフが一対一の場合の問い合わせでは，入力グラフ Q の固有値の数だけ定理を調べる必要があり計算コストが大きい．よって，部分グラフを含む可能性のある候補グラフのすべての固有値を，線形に数値比較することなく枝刈りができるインデクスが必要である．

3.2 提 案 手 法

突き刺し問い合わせとは，与えられた点を含む区間をすべて求めることをいう．いくつかの区間があり，ある質問点が来たときに閉区間内に質問点があれば区間を返す．この突き刺し問い合わせが提案手法の基盤となっている．

次に提案手法の具体的な内容であるインデクスを用いた突き刺し問い合わせの簡単な例を以下に示す．

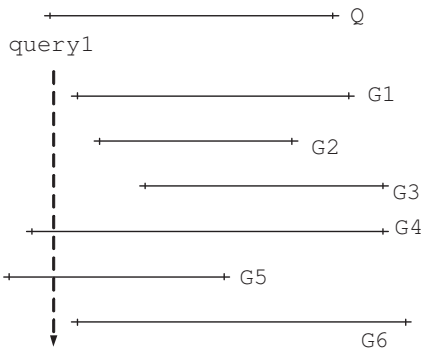

図 4　一次元上におけるインデクスを用いた突き刺し問い合わせ一回目

図 4，図 5 はインデクスを用いた突き刺し問い合わせの例である．大きなグラフ集合として $G_i, i = 1 \sim 6$ の固有値区間があり、質問の入力グラフ Q の固有値区間がある．図が示すように，入力グラフ Q の固有値 θ_i の最小固有値である $i = 1$ を質問点 $query1$ として区間木に問い合わせる．Cauchy のインタレース定理 (6) が示すとおり $\alpha_1 \leqq \theta_1 \leqq \alpha_{1+(n-m)}$, という関係が成り立ち，入力グラフ Q を部分グラフとして持つグラフ候補となるためにはグラフ G_i の最小固有値の右側に $query1$ がなければならない．図 4 の通りに入力グラフの Q 最小固有値である区間の左端を $query1$ として入力すると部分グラフの候補と

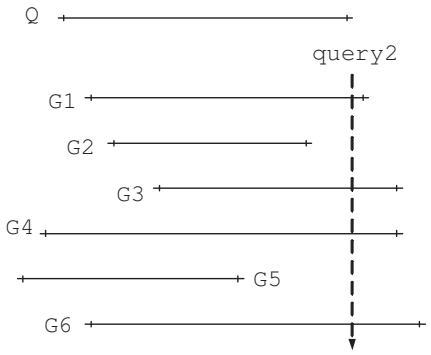

図 5　一次元上におけるインデクスを用いた突き刺し問い合わせ二回目

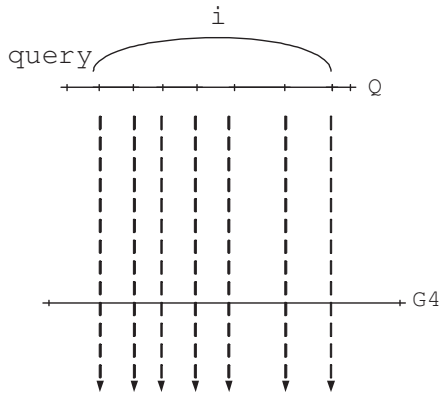

図 6　一次元上における突き刺し問い合わせ後の線形チェック

なる必要条件を満たしているのは，区間を返す G_4 と G_5 だけである．この 2 つ以外に部分グラフを持つ候補となるものはない．よって，1 回の問い合わせでは 2 回のチェックで候補を見つけることができた．次に図 5 において入力グラフ Q の最大固有値 θ_i の $i = m$(m:最大区間) を質問点 $query2$ として問い合わせを行う．すでに 1 回目の問い合わせで候補は G_4 と G_5 となることがわかっている．2 回目の $query2$ では G_1，G_3，G_4 と G_6 の候補が見つかる．よって 4 回のチェックによって候補の判断ができる．1 回目と 2 回目で共に候補となったのは G_4 だけになる．つまり入力グラフ Q を部分グラフとして持つ候補となるのは G_4 だけになる．グラフの固有値の最大値と最小値の 2 回の突き刺し問い合わせの後，入力グラフ Q の固有値が全てインタレース定理を満たしているか調べる必要があるので，2 番目に最小の固有値から昇順に調べていく．そのことを示したのが図 6 で，線形の問い合わせと同じ処理である．定理を満たさなくなるまで最大 i 回, つまり m-2 回だけ固有値を調べる必要がある．グラフ集合の固有値は 0.0 近傍ではほとんどのグラフが存在する．よって 0.0 付近の固有値で問い合わせを行ってしまうと，すべてのグラフが区間を返してしまうことになり，線形の場合とコストが変わらなくなってしまう．それに加え，インデクスである区間木をたどるコストを足す必要があるので計算コストはより大きくなる．よって報告される区間が少ない問い合わせをすることが有効である．大規模なグラフ集合を考えたときに，グラフ集合の中には様々の頂点数を持つグラフがあり，部分グラフでないものも多く存在する．このよう

インデクスを用いた問い合わせ

1. 入力グラフの最小固有値 $\theta_1 \in Q$ で区間木に問い合わせを行う
2. 報告される区間の数だけ以下を調べる
3. 区間の G_{id} を調べ，インタレース定理をチェックする
4. 定理を満たしていればフラグに 1，でなければ 0 を立てる
5. 次に入力グラフの最大固有値 $\theta_m \in Q$ で区間木に問い合わせを行う
6. 報告される区間の数だけ以下を調べる
7. 区間の G_{id} を調べ，インタレース定理をチェックする
8. インタレース定理を満たしていないかもしくはフラグが 0 である場合は何もしない
9. そうでなければ入力グラフ Q が部分グラフとなる可能性があるので入力グラフ Q の固有値数 $m-2$ だけ G_{id} ごとにインタレースを線形にチェックする
10. 入力グラフ Q のどの固有値 $\theta_n \in Q$ も定理を満たしているならば入力グラフを含む候補としてカウントする

図 7　インデクスを用いた問い合わせの流れ

な中で，部分グラフの候補でないものを少ないコストで枝刈りすることは十分に可能である．

3.3　提案手法の拡張

提案手法では，グラフの固有値の最大値と最小値の 2 回の突き刺し問い合わせの後，入力グラフの固有値の全てが定理を満たすか判定した．判定の途中で定理を満たさないことが分かればそれ以降調べなくてもよい．ただし，その時には固有値を昇順に調べていた．後の実験結果で示す通り，候補グラフの数が増えて飽和状態になる場合には，約 9 割近くをこの定理を調べる処理が占めている．提案手法の改善のためには，この判定方法の変更が重要である．

ここで実験的な観察から，定理を満たさなかった時の入力グラフの固有値は最大固有値もしくは最小固有値の近辺であることが多い．つまり最大固有値の直前で定理を満たさないこともあるので，既存の手法である固有値を昇順に調べた場合には最大固有値近辺で初めて定理を満たさないことが分かったとしても，それまでのコストは非常に大きい．よって，固有値を昇順ではなく，最大値と最小値の近くの両端から順次調べるように変更する．

3.4　提案手法の問い合わせの流れ

提案手法を用いた問い合わせの流れを図 7 に示す．

大きな処理としては，突き刺し問い合わせでの枝刈りとインタレース定理での固有値比較である．インデクスを用いて突き刺し問い合わせでの枝刈りを行った後に固有値比較をするので，効率的に入力グラフを部分グラフとして持つ可能性のある候補を見つけ出せる．

4.　関 連 研 究

4.1　部分グラフ同型判定

グラフ同型判定に関しての関連研究としては，グラフ集合をグラフの分解によって得られる特別なデータ構造に変換後，それぞれに対して部分グラフ同型判定を行って効率的に問題を解く Messmer [4] らの方法が挙げられる．また検索範囲を著しく減少させるバックトラックに基づくアルゴリズムが Ullmann [5] によって提案されている．集合論をもとにしたアルゴリズムを用いている Nauty library も速い同型判定を行うことができる．さらに，深さ優先探索を行う VF [6] はグラフ同型判定，部分グラフ同型判定を両方行える上に使用するメモリ量が少ないため，大きなグラフを扱うことのできる手法として知られている．ただ，いずれの方法も誘導部分グラフ，もしくは部分グラフといったグラフの構造を用いて同型判定をするため，計算コストは非常に大きい．

4.2　領域問い合わせインデクス

特徴空間を階層的な部分領域に分割し，データ構造を構築する．分割の方法は 2 種類あり，一つ目として，部分領域が重なりなく分割されたものに Kd-tree [11] や quad-tree [12] がある．Kd-tree はデータの分布に応じて分割を進めるのに対し，quad-tree はあらかじめ定められた比率で規則的に分割していく．二つ目は部分領域の重なりを許容するものである．その代表は R-tree [13] や R*-tree [14] がある．R*-tree は動的に木の平衡が維持可能なので高速な探索が可能である．R-tree は領域を囲む最小包囲長方形 (MBR) を用いて階層化していく．ただ重なりを許容しているので高次元になるほど効率的ではなくなる．むしろ高次元では階層化するよりも線形の方が効率がよい．そのことを用いているのが階層構造と線形構造を組み合わせた X-tree [15] である．

R-tree などの領域問い合わせは点データを対象としていたのに対し，ウィンドー問い合わせは線分，多角形，曲線などを対象としている．ウィンドー問い合わせの幾何学データ構造に関しては，与えられた点を含む区間をすべて求めるという突き刺し問い合わせにプライオリティー探索木や区分木などがある．プライオリティー探索木は 2 次元の領域をヒープ木に格納するとき，分割情報を用いて領域を分割していく．プライオリティー探索木は，データ変換を施すことによって突き刺し問い合わせを可能としている．区間木や区分木については動的化の研究が多くなされている．

5.　区間木を用いたインデクシング

本インデクシングのデータ構造として区間木 [10] を利用している．もともと区間木は地理情報などの分野で，与えられた点を含む区間を返す突き刺し問い合わせに利用されている．今回はグラフデータベースにある様々な大きさのグラフを区間木に蓄え，小さな入力グラフの固有値を入力している．具体的には，様々な大きさのグラフの各固有値間を区間と見立てて区間木に蓄え，入力グラフの固有値を点として入力して突き刺し問い合わせを行い，部分グラフ同型判定を調べるために固有値間の包含関係を調べるものである．この包含関係は式 (6) で示したインタレース定理を適用することである．

5.1　区間木の改良点

多数のグラフの固有値区間を区間木に蓄えるにあたっての改良点を以下に挙げる．

- それぞれの区間によるグラフ ID の保持

- 0.0 と −1.0 による重解処理

一つ目の ID の保持に関しては，区間木に蓄えられる一つのグラフは複数の固有値を持つ．複数の固有値を持つので，蓄えられる区間数も複数になる．よって，すべての区間の数値だけで木構造を構成するので区間の身元を示すために G_{id} をそれぞれ持っている．これは，問い合わせ時に一意にインタレース定理を比較できるようにするためである．

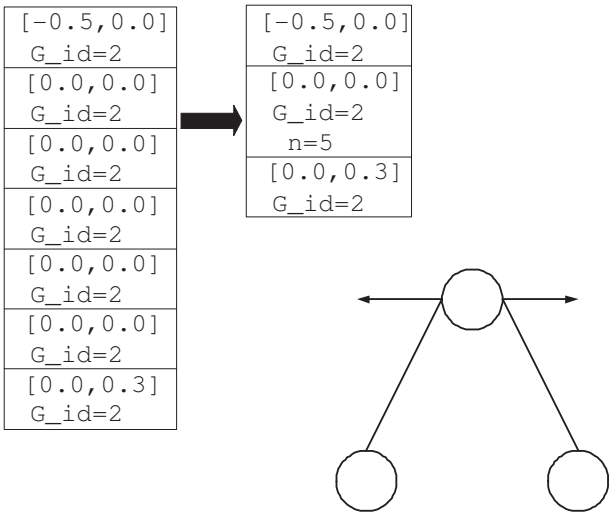

図 8　区間木のノードリストでの重解処理と ID の保持

次に二つ目の重解処理に関してである．グラフの隣接行列表現の固有値は，ラベル数が少ないときに，0.0 と −1.0 に多く重解を持つ場合があることが，実験から分かっている．グラフの頂点数が増大したときには重解の数も比例して増えている傾向にある．しかし，重解もインタレース定理では一つの固有値として見なされるので，区間木に蓄える必要がある．ここで問題なのが，以下に示す区間木構築のアルゴリズムを適用すると 0.0 もしくは −1.0 のノードに重解によるたくさんの区間が蓄えられてしまい，問い合わせ時に余分な計算コストを必要とする可能性がある．そこで，重解は同じ G_{id} 同士を一つのデータと考えることによりリストを短くし，問い合わせを工夫することによって余計なコストを省いている．

図 8 にリストに重解処理を施した結果を示す．具体的には，重解の 0.0 もしくは −1.0 は区間木を構築するとき自らがノードの値になる．図 8 の場合は 0.0 が木のノードになっている．0.0 のノードに付随しているリストには重解区間がたくさん蓄積されてしまうが，その数をグラフ集合におけるグラフ一つずつに対して重複区間を一つとして蓄えることで線形に問い合わせるコストを削減している．

具体的には，元々の区間数が 5 であったのに対して，処理をした後では区間数を 1 にして，そこに重解数を示す $n = 5$ の情報を保持している．これはグラフの固有値を扱うことを意識したものである．ただ注意すべきことは，グラフの固有値に必ず重解が発生するとは限らないということである．

5.2　区間木の構築

区間木は実数直線上の区間集合 I が与えられたとき，質問点 q を含むものを報告するデータ構造である．$I = [x_1, x'_1], [x_2, x'_2], \ldots, [x_n, x'_n]$ を実数直線上の閉区間集合とする．区間集合 I を木が平衡になるよう分割するには，区間集合 I のすべての端点 (区間の両端) のメディアンをノード X として木の節点に蓄える．つまり区間集合 I が n あるとすると端点の合計は $2n$ なのでメディアンは n 番目の端点となることわかる．そしてノード X の左右にはそれぞれ等しく区間の端点数があり、木の平衡は保たれている．ノード X の値 X_{mid} を含む区間集合 I_X を節点に蓄え，ノード X を含まずノード X の左にある区間の集合 I_{left} を左部分木に，ノード X を含まずノード X の右にある区間の集合 I_{right} を右部分木にそれぞれ蓄える．この操作を再帰的に行い，区間数が 0 になるまで木を構築する．上記の手順の概要を示したのが図 9 である．

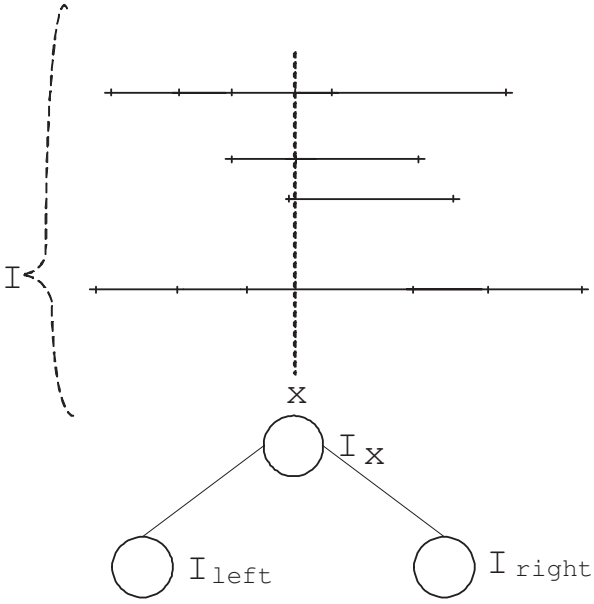

図 9　区間木での分割

区間データは木の各ノードに蓄えられていて，それは付随構造と呼ばれるリストに格納されている．付随構造は一つのノードに左右二種類がある．一つは左側リスト，もう一つは右側リストである．左右のリストは同じ区間集合であるが，違いは区間の左端点でソートされたものか右端点でソートされているかである．今，質問点 q がノード X の値より左にあるとすると，ノード X の左側リストを参照する．左側リストにある区間 $[x_j, x'_j] \in I_X$ が質問点 q を含むための必要充分条件は $x_j < q$ である．つまり，左端点が昇順にソートされていれば q を含まない区間まで順に調べていけばよいことになる．区間木のデータ構造を簡潔にまとめると，

- $I = 0$ の時，区間木は一つのノードだけからなる．
- そうでないとき，区間端点のメディアンを X_{mid} とする．

$$\begin{cases} I_{left} = [x_j, x'_j] \in I | x'_j < X_{mid} \\ I_{mid} = [x_j, x'_j] \in I | x'_j \leqq X_{mid} \leqq x'_j \\ I_{right} = [x_j, x'_j] \in I | X_{mid} < x_j \end{cases} \tag{7}$$

とする．さらに区間木は X_{mid} を蓄えるノード X を持ち，

ConstructIntervalTree(I)

1. もし I が空ならば
2. 空のノードを返す
3. そうでなければノード ν を作る
   区間端点の集合のメディアン X_{mid} を計算し，
   ν に蓄える
4. 区間集合 I_X を作り，I_X に対して 2 つの
   ソート列 L_{left} と L_{right} を構成する
5. 左の子ノード $lc(\nu)$ =ConstructIntervalTree(I_{left})
6. 左の子ノード $rc(\nu)$ =ConstructIntervalTree(I_{right})
7. ノード ν を返す

図 10　区間木構築アルゴリズム

QueryIntervalTree(ν, q)

1. もし ν がノードでなければ
2. もし $q < X_{mid}$ ならば
3. ノード ν の付随構造リスト L_{left} を先頭から調べ，
   q を含む区間をすべて報告する
4. q を含まない区間に到着したら終了する
5. QueryIntervalTree($lc(\nu), q$)
6. もし $q > X_{mid}$ ならば
7. ノード ν の付随構造リスト L_{right} を先頭から調べ，
   q を含む区間をすべて報告する
8. q を含まない区間に到着したら終了する
9. QueryIntervalTree($rc(\nu), q$)
10. もし $q = X_{mid}$ ならば
11. ノード ν の付随構造リスト L_{right} を先頭から調べ，
    q を含む区間をすべて報告する
12. q を含まない区間に到着したら
    QueryIntervalTree() を終了する．

図 11　区間木問い合わせアルゴリズム

– 区間集合 I_{mid} を二度蓄える．一度は区間の左端点に関してソートしたリスト L_{left} に蓄え，区間の右端点に関してソートしたリスト L_{right} にも蓄える

– ノード X の左部分木は集合 I_{left} の区間木になっている

– ノード X の右部分木は集合 I_{right} の区間木になっている

図 10 に区間木の構築のアルゴリズムを示す．

5.3　区間木での問い合わせ

問い合わせとは，質問点 q を含む区間をすべて報告することである．基本的な処理は木の根からノードを調べ，そのノードにある付随構造リストを検索して区間をすべて報告する．この処理を子ノードへと進みノードが空になるまで続けるが，あるノードの値と等しくなった場合には，そのノードの付随構造リストを検索して終了する．図 11 に正確な問い合わせアルゴリズムを示す．

5.4　インデクスについての処理

今回想定している状況は，大小さまざまの大規模なグラフ集合から，入力グラフ Q を含んでいる，入力グラフ Q が部分グラフとなるようなグラフ G を見つけることである．

大規模なグラフ集合にあるグラフ G はそれぞれグラフ ID である G_{id} を持っている．区間木に蓄えられている区間数は，各グラフの固有値すなわち頂点数 V_i から一つ少ない区間数の合計 $\sum_i (V_i - 1)$ である．その $\sum_i (V_i - 1)$ の数だけ区間が木に蓄えられているので，突き刺し問い合わせにより報告される区間のグラフの種類は様々である．そのグラフの種類を即時に判断し，インタレース定理を適用する必要がある．区間木に対して突き刺し問い合わせをして報告される区間は，区間が持つグラフ ID である G_{id} を元にグラフの種類識別を行っている．

5.5　区間木に関する計算コスト

問い合わせに関する時間は，木をたどる時間と付随構造のリストをたどる時間の和で表される．まず木をたどる時間は，木の深さの上界が $\log(n)$ なので $O(\log(n))$ になる．付随構造のリストは，各訪問するノードで費やす時間は $O(1 + k_v)$．ただし，k_v は報告される区間の個数である．訪問したすべてのノードについて k_v の和を取ると，k である．よって全体の問い合わせ時間は $O(\log(n) + k)$ となる．蓄えるべき区間数が増えたとき，付随構造のリストが短くなるように区間木を構成できればよいが，区間が重複して付随構造のリストが長くなると，問い合わせ時間はリストの長さに支配される．よって，区間が重複して $k \gg 1$ のとき，問い合わせ時間は $O(k)$ となることがわかる．

5.6　区間木インデクスの記憶領域

提案手法を行うにあたって必要な記憶領域は区間木インデクスの領域だけとなる．区間木に蓄えられる区間数を n とすると，$O(n \log(n))$ であるので，このインデクス手法では $O(n \log(n))$ の記憶領域を必要とする．

6.　評 価 実 験

線形問い合わせと提案するインデクシングを用いた問い合わせの処理時間を比較した．検索対象となるグラフ集合に含まれる元の数を変えたものと，元の数を一定にしてグラフの大きさを変えたものの二種類の実験を行った．グラフには，5 種類のラベルを無作為に付与した．

6.1　実 験 環 境

Pentium(R)4 プロセッサ 3GHz，メモリ 1GB，OS として WindowsXP を搭載した PC で測定を行った．インデクスの実装は C 言語を用いた．実験で使用するグラフデータはミネソタ大学の蔵持らの開発したグラフ生成ソフトを利用した．

6.2　実験結果と考察

図 12 は，グラフ集合の元を一定 (1000) とし，グラフ集合に含まれる元の平均頂点数を徐々に増やしていった場合の処理時間の変化を示している．このとき，入力グラフの頂点数は 100 である．表 1 に記したのは処理時間と候補グラフ数の関係である．グラフ集合に含まれる元の平均頂点数が入力グラフの頂点数以上になると，線形問い合わせに比べて提案手法が約 2 倍早い．しかし，グラフ集合の平均頂点数が入力グラフより少ない場合は提案手法が遅くなる．さらに表 1 から，候補グラフ数が半数を超えて全グラフの数に近づくと，提案手法が線形問い合わせに劣る．この場合，提案手法においても突き刺し問い合わせを行った後に，すべての固有値で定理を満たしているか判定するので，入力グラフを部分グラフとして持つ候補の数が大き

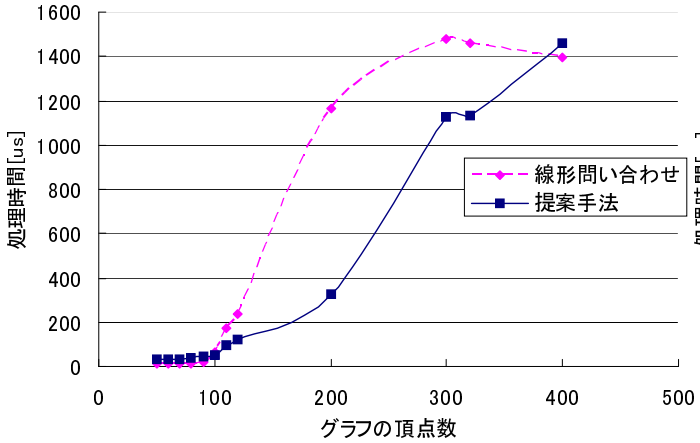

図 12　入力グラフの頂点数が 100，グラフ数が 1000，頂点数を変化させたときの問い合わせ処理時間

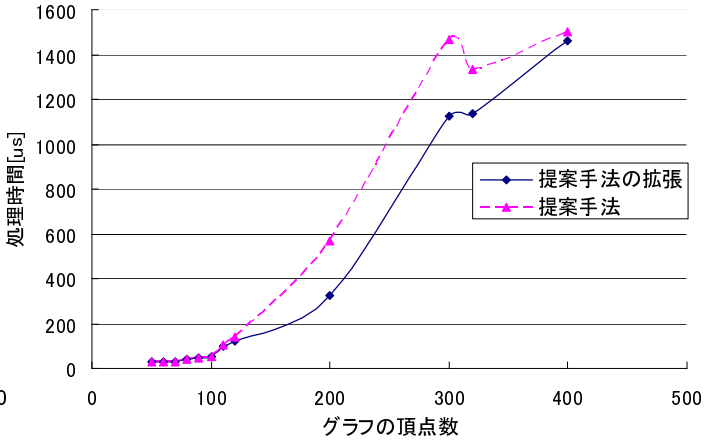

図 13　グラフ数が 1000 で一定で頂点数を変化させたとき，提案手法の拡張の問い合わせ処理時間

表 1　入力グラフの頂点数が 100，グラフ数が 1000，頂点数を変化させたときの問い合わせ処理時間と候補グラフ数の関係

平均頂点数	線形での時間 [us]	提案手法での時間 [us]	候補グラフ数
60	11	30	0
80	13	40	0
100	62	52	0
110	174	99	1
120	234	121	5
200	1167	326	125
300	1480	1124	488
400	1398	1459	892

くなれば線形と同等もしくはそれ以上となるからである．それでも，入力グラフを部分グラフとして持つ候補が約半分の場合では，十分な枝刈りが可能となった．平均頂点数が 400 を超えると，候補グラフ数が全体の 9 割近くになり，ほぼすべての区間を調べるので，処理時間が一定に近づくと考えられる．処理時間が一定に近づいた時の線形問い合わせと提案手法の差は，提案手法が行っている木をたどる時間だと考えられる．

図 13 は，提案手法の拡張として述べた，固有値の順番を昇順から最大固有値もしくは最小固有値に近い順番に変更した結果である．この場合には，拡張なしの提案手法より平均頂点数 100 以上の領域で 2 割程度改善した．しかし，頂点数 100 以下もしくは処理時間が一定に近づく場合は改善がみられない．このことから，頂点数 100 以下の領域は枝刈りのコストではなく，インデクスである区間木をたどるコストであると考えられる．

図 14 はグラフの平均頂点数を一定にして，グラフ数を変化させたときの結果である．グラフ集合に蓄えられているグラフの数を 100 から 10000 まで変化させるが，そのときの平均頂点数は 50 で一定である．また，入力グラフの頂点数をグラフ集合の平均頂点数より少ない 40 とした．線形問い合わせに比べてグラフ数が少ない場合は，わずかではあるが提案手法はインデクスをたどる時間を要するために劣っているが，グラフ数が大きくなると次第に提案手法が線形問い合わせを勝る．これは

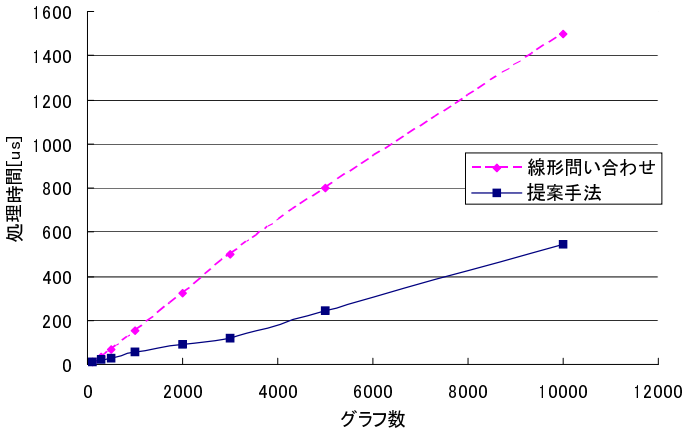

図 14　入力グラフの頂点数が 40 で，グラフ集合の頂点数が 50，グラフ数を変化させたときの処理時間

入力グラフの頂点数がグラフ集合の平均頂点数とほぼ等しいことから，突き刺し問い合わせだけで枝刈りできなかったグラフに対しても，定理を満たすかの判定が早く済んだためである．

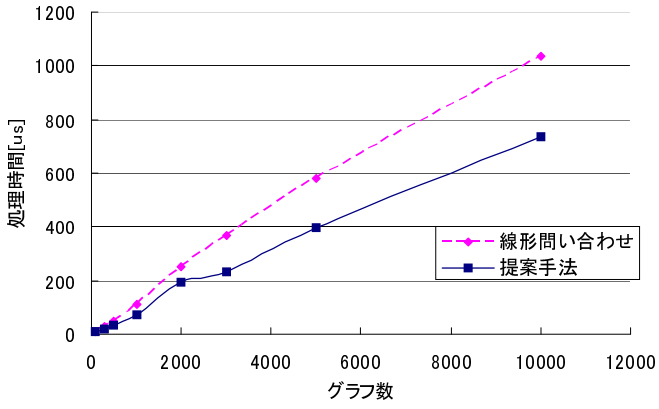

図 15　入力グラフの頂点数が 50，グラフ集合の頂点数が 50，グラフ数を変化させたときの処理時間

図 15 は，入力グラフの頂点数を 50 として，図 14 と同様の実験を行った結果である．この場合，入力グラフが部分グラフとなる可能性は低い．図 14 と同様にグラフ数が少ない段階ではインデクスをたどるコストが大きくなっているが，グラフ集合のグラフ数を大きくすると提案手法が勝っている．さらに図 14 の頂点数 40 と比較すると，突き刺し問い合わせによる枝刈りができない場合，線形に定理を満たすか調べるコストが大きくなるので，提案手法の傾きが大きくなっている．また，処理時間が依存しているのは，単にグラフ数だけではなく，処理結果の候補グラフ数の関係が大きな要因となっていることが考えられる．

図 14 と図 15 を総合的に考えると，入力グラフの頂点数がグラフ集合に含まれる元の平均頂点数と等しいか，もしくはそれ以下の大きさのグラフからなるグラフ集合を考えたとき，この提案手法が有効であることがわかる．さらに，入力グラフの頂点数がグラフ集合に含まれる元の平均頂点数以上になっても，突き刺し問い合わせだけで枝刈りができるので提案手法は有効である．

提案手法が既存の線形問い合わせに比べて劣っている場合を考察する．入力グラフとグラフ集合に含まれる元の平均頂点数の大小関係により，入力グラフを部分グラフとして持つ可能性のある候補の数が増え，グラフ集合の枝刈りができなかったときである．入力グラフがグラフ集合に含まれる元の平均頂点数と等しいもしくは大きいときに，突き刺し問い合わせによって多くの区間が返ってきてしまうと，その区間をすべて定理が満たさなくなるまで判定しなければならず，線形のコストに近づいてしまう．それに加え，グラフ数が少ない段階では木をたどる時間が大きく効いてしまい，線形問い合わせより遅くなると考えられる．

7. まとめと今後の課題

本稿では，大量のグラフ集合を対象とした部分グラフ同型判定問題に対し，区間木を用いてインタレースの定理を適用した固有値に基づくグラフのインデクシング手法を提案した．区間木による固有値でのインデクシング手法により，大量のグラフ集合から入力グラフを部分グラフとして持たないものを効率的に枝刈りする問い合わせ方法を提案した．突き刺し問い合わせによる提案手法は，入力グラフとグラフ集合に含まれる元の平均頂点数の関係を比べた時，入力グラフの頂点数が比較的少ない場合に固有値比較するだけで効率的に枝刈りができ，線形にインタレースを適用するよりも高速であるという結果を得た．

今後の課題としては，区間木のインデクス改善が挙げられる．区間木はノードに区間データとして線形リストを蓄えているのため，一つのノードにたくさんの区間が蓄えられると，区間の探索時間に大きな影響を与える．

また，動的に区間木の内容を変更することも考えられる．固有値区間が途中で変更された際に，木の平衡を保つには検討が必要である．

謝辞　実験用グラフデータ生成ソフトウェアを提供いただいたミソネタ大学蔵持道広氏に深く感謝いたします．また多くの貴重なコメントを頂いた査読者の方々に深く感謝いたします．本研究の一部は，(独) 日本学術振興会科学研究費補助金基盤研究 (B)(2)(課題番号:16300030) による．

文　献

[1] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H.Weissig, I. Shindyalob, and P. Bourne. The protein data bank. ”Nucleic Acids Research”, 2000

[2] National Library of Medichine http://chem.sis.nlm.nih.gov/chemidplus

[3] Xifeng Yan, Philip S. Yu, Jiawei Han,”Substructure Similarity Search in Graph Databases”, SIGMOD 2005 June 14-16,USA, 2001

[4] Bruno T. Messmer and Horst Bunke, ”Efficient Subgraph Isomorphism Detection: A Decomposition Approach”, IEEE Trans. On Knowledge and Data Eng., 12(2), 2000

[5] J.R. Ullmann, ”An Algorithm for Subgraph Isomorphism”, J.Assoc for computing Machinery, vol.23, pp.32–42, 1976

[6] Luigi.P. Cordella, Pasquale. Foggia, Carlo. Sansone, Mario, Vento, ”A (Sub)graph Isomorphism Algorithm for Matching Large Graphs”, IEEE Trans. on PATTERN ANALYSIS AND MACHINE INTELLIGENCE, vol.26, no.10, pp.1367-1372, 2004

[7] Beresford N. Parlett, ”The Symmetric Eigenvalue Problem”, siam, 1997

[8] E. Petraki and C. Faloutsos, ”Similarity searching in medical image detabases” ,Knowledge and Data Engineering, 9(3):435-447,1997

[9] S. Beretti, A. Bimbo, and E. Vicario, ”Efficient matching and indexing of graph models in content based retrieval”, IEEE Trans. on Pattern Analysis and Machine Intelligence, 23:1089-1105, 2001

[10] M. ドバーグ, M. ファン．クリベルド, M. オーバーマーズ, O. シュワルツコップ, ”コンピュータジオメトリ”, 近代科学社,2001

[11] J. L. Bently, ”Multidimensional binary search trees used for associative searching”, Commun.ACM,1975

[12] R. A. Finkel, J. L. Bentley, ”Quad trees: a data structure for retrieval on composite keys”, Acta Inform., 4:1-9,1974

[13] Guttman A., ”R-trees: A Dynamic Index Structure for Spatial Searching”, Proc.ACM SIGMOD Int. Conf. on Management of Data, pp.47–57,1984

[14] Beckmann N., Kriegel H.-P., Schneider R., Seeger B., ”The R*-tree: An Efficient and Tobust Access Method for Points and Rectangles”, Proc.ACM SIGMOD Int. Conf. On Management of Data, pp322–331,1990

[15] Stefan Berchtold, Daniel A. Kein, Hans-Peter Kriegel, ”The X-tree: An Index Structure for High-Dimensional Data”, Proc. 22th Conf. on VLDB,1996

[16] 長屋 未来, 片山 薫, 石川 博, ”大規模部分グラフを対象とした部分グラフ同型判定おける Interlace 定理の利用”, 電子情報通信学会第 17 回データ工学ワークショップ DEWS2006,2006.3