DEIM Forum 2022 G34-1

グラフデータベースに対する相関問合せ手法の高速化

直井　悠馬† 　真次　彰平†† 　塩川　浩昭†††

† 筑波大学情報学群情報科学類　〒 305-8577 茨城県つくば市天王台 1-1-1
†† 筑波大学理工情報生命学術院　〒 305-8577 茨城県つくば市天王台 1-1-1
††† 筑波大学計算科学研究センター　〒 305-8577 茨城県つくば市天王台 1-1-1
E-mail: †naoi@kde.cs.tsukuba.ac.jp , ††matsugu@kde.cs.tsukuba.ac.jp, †††shiokawa@cs.tsukuba.ac.jp

あらまし　相関問合せとはグラフデータベースの中からクエリと共起して出現する部分グラフを検出する処理であり，医療や生命科学分野で広く利用されている．大量のグラフから構成される大規模なデータベースに対する相関問合せは，グラフ構造の複雑さによって計算コストが大きくなるという問題がある．これまでグラフデータベースの中からクエリグラフとの相関値が最も高い k 個の部分グラフを求める Top-k 相関グラフ問合せ手法が提案されている．しかし，グラフデータベースを構成するグラフがより大規模になると相関判定のコストは大きくなるため，依然として実行に多くの時間がかかる．そこで本研究ではグラフ構造を要約したグラフを構築することで相関問合せのコストを削減する手法を提案する．実データを用いた評価実験を行い，提案手法が既存手法に比べて最大 6.1 倍高速であるとともに，高精度に Top-k 相関グラフを得られることを確認した．

キーワード　グラフ，相関問合せ，グラフマイニング，グラフデータベース

1 序　　論

グラフはオブジェクト間の関係を表現する際に利用される一般的なデータ形式であり，科学や生物学，生命科学分野，ソーシャルネットワークなどの分野で広く使われている [16,17]．グラフはデータオブジェクトを頂点とし，データオブジェクト間の関係を辺として表す．複数のグラフから構成されるグラフデータベースの需要が高まると共に，グラフデータベースの中から有用な情報を検出するグラフマイニングの問題が重要になる．これまでデータベースの中から頻出した部分グラフ構造を検出する頻出部分グラフ検出の研究 [3,5,11,12] が広く行われてきた．その中でも共起して出現する部分グラフ対を検出する相関部分グラフ検出の研究 [1,2,7,8] は近年注目を集めている．相関問合せはグラフデータベースとクエリグラフが与えられたとき，クエリグラフの出現分布に近い部分グラフを特定する問題である．共起して出現する部分グラフ対はデータベースの隠れた特性を示すことがあり，多くのアプリケーションで有用である．例えば，科学化合物のデータベースでは共起して出現する構造対を検出し，新たな知識発見の補助となる．また，Web サイトのログから共通の興味を持つユーザのパターンを検出し，ユーザの振る舞いをより把握することができるため，E コマースの分野でも応用することができる．

しかし，大規模なグラフデータベースに対する相関グラフ問合せにはいくつかの課題がある．1 つ目の課題はグラフデータベース内の全てのグラフの部分グラフが相関グラフの候補，すなわち候補グラフとなる点である．2 つ目の課題は相関尺度には逆単調性がなく，トランザクションデータにおける相関ルール検出で用いられる既存の枝刈り手法 [4] が適用できないという点である．最後の課題は相関グラフ判定には候補グラフの出現頻度を考えるため膨大なコストがかかるという点である．グラフデータベースにおいて，候補グラフの出現頻度を高速に計算するためにグラフ索引技術 [9] などがこれまで提案されているが，これらの手法を用いてもコストは依然として大きい．

これらの問題に対して，これまでユーザによって与えられた閾値より大きな相関値を持つ部分グラフを検出する手法 [2,6] や，相関値が最も高い k 個の部分グラフを検出する Top-k 相関問合せの手法 [1] が提案されてきた．これらの手法は探索する候補グラフの探索範囲をグラフデータベース全体ではなく，クエリグラフを含むグラフ集合のみに限定して探索する．これにより，探索する部分グラフ数を削減し，効率的に相関計算を行うことで相関グラフを検出することができる．Top-k 相関グラフ問合せは，ユーザの指定する相関値の閾値によって得られる相関グラフの数が変化する閾値ベースの手法に対して，ユーザが検出する相関グラフ数を直接制御することができる．しかしながら，相関グラフ問合せには依然として大きなコストがかかる．これは大規模なグラフデータベースでは候補グラフの数が大きくなり，相関値の計算のコストが大きくなることが原因である．ゆえに，既存の Top-k 相関グラフ問合せはより大規模なグラフデータベースに対する実行時間が大きくなるという問題がある．

この問題を解決するために，本研究では大規模なグラフデータベースに対する高速な Top-k 相関グラフ問合せ手法を提案する．提案手法はグラフデータベースの各グラフ構造を要約したグラフデータベースを構築することで探索するグラフ数を縮小させると共に，候補となる部分グラフの相関問合せの判定コストの削減を図る．また，提案手法ではグラフ要約によって生じる相関グラフの偽陽性・偽陰性に対応するため，乱択アルゴリズムに基づくランダムな要約グラフを利用する．これにより，獲得した相関部分グラフに対して検証を行うことで高精度な Top-k 相関グラフ問合せ手法を提案する．

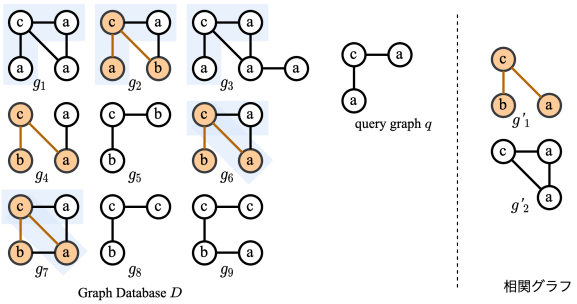

図 1: Top-k 相関問合せの例

本研究の貢献として，以下の 3 つが挙げられる．

- 提案手法は実際のタンパク質グラフデータベースに対して Top-k 相関問合せ処理が既存研究に比べて最大 6.1 倍高速であることを確認した．(5.3 節)
- 提案手法は高精度に Top-k 相関グラフを求めることができることを確認した．(5.3 節)
- 提案手法は評価実験においてパラメータ k，グラフデータベースのサイズを変えた場合にも，既存手法に比べて高速であることを確認した．(5.4 節，5.5 節)

本論文の構成は以下の通りである．2 節で前提となる知識と本研究で取り扱う問題について説明し，3 節では先行研究を紹介する．4 節で提案手法について述べ，5 節で実データを用いた性能評価結果を示す．最後に 6 節で本研究のまとめを述べる．

2 基本事項

2.1 前提知識

本研究で対象とするグラフはラベル付無向連結グラフ $g = (V, E, l)$ である．V は頂点集合，E は辺集合，l は各頂点と各辺にラベルを与えるラベル関数である．また N 個のグラフ $g_1, g_2, \ldots, g_N$ から構成されるグラフデータベースを $\mathcal{D} = \{g_1, g_2, ..., g_N\}$ とする．

2 つのグラフを $g = (V, E, l)$，$g' = (V', E', l')$ とするとき，$\forall(u, v) \in E$ について，$(f(u), f(v)) \in E', l(u) = l'(f(u)), l(v) = l'(f(v)), l(u, v) = l'(f(u), f(v))$ となる単射関数 $f : V \to V'$ が存在するとき，$g \subseteq g'$ と表記し，g は g' の部分グラフ同型であるという．また，本論文では $g \subseteq g'$ となるとき，g' は g のスーパーグラフと呼ぶ．

グラフ g の $\mathcal{D}$ での出現頻度の指標として支持度を $supp(g; \mathcal{D}) = \frac{|\mathcal{D}_g|}{|\mathcal{D}|}$ と定める．本論文では $\mathcal{D}$ が文脈上明らかであるときは，$supp(g; \mathcal{D})$ を $supp(g)$ と表記する．2 つのグラフ g_1, g_2 が同時に出現する割合である結合支持度は $supp(g_1, g_2) = \frac{|\mathcal{D}_{g_1} \cap \mathcal{D}_{g_2}|}{|\mathcal{D}|}$ と定める．支持度には逆単調性があり，$g \subseteq g'$ ならば，$supp(g') \leqq supp(g)$ となる．

2.2 問題定義

相関グラフ問合せとは $\mathcal{D}$ とクエリグラフ $q = (V_q, E_q, l_q)$ が与えられたとき，クエリと共起して出現する部分グラフを $\mathcal{D}$ から検出することである．本研究では 2 つのグラフの共起度合いを評価する相関尺度としてピアソン相関係数 [10] を採用し，定義は以下の通りである．

定義 1 (ピアソン相関係数)**.** 2 つのグラフ g_1, g_2 が与えられたとき，それらの相関値 $\phi(g_1, g_2)$ を以下のように定める．

$$\phi(g_1, g_2) = \frac{supp(g_1, g_2) - supp(g_1)supp(g_2)}{\sqrt{supp(g_1)supp(g_2)(1-supp(g_1))(1-supp(g_2))}}$$

相関値 $\phi(g_1, g_2)$ は $[-1, 1]$ の範囲の値をとる．また，$\phi(g_1, g_2)$ が正の値をとるとき，g_1 と g_2 の出現分布に正の相関があることを示す．したがって，正の相関がある 2 つのグラフ対はグラフデータベースの中で共起して出現すると言える．

相関値の例として，図 1 では g'_1，q のそれぞれの支持度は $supp(g'_1) = \frac{|\mathcal{D}_{g'_1}|}{|\mathcal{D}|} = \frac{4}{9}$，$supp(q) = \frac{|\mathcal{D}_q|}{|\mathcal{D}|} = \frac{5}{9}$ となり，結合支持度は $supp(g'_1, q) = \frac{3}{9}$ である．したがってこれらの相関値は，$\phi(g'_1, q) = \frac{\frac{3}{9} - \frac{4}{9} \cdot \frac{5}{9}}{\frac{4}{9} \cdot \frac{5}{9} \cdot (1 - \frac{4}{9}) \cdot (1 - \frac{5}{9})} = 0.35$ である．本研究で対象とする Top-k 相関グラフ問合せを以下のように定義する．

定義 2 (Top-k 相関グラフ問合せ)**.** グラフデータベース $\mathcal{D} = \{g_1, g_2, ..., g_N\}$，クエリグラフ q，整数 k が与えられたとき，相関値 $\phi(g, q)$ が最も高い k 個のグラフ g を求める．

Top-k 相関グラフ問合せの例として，図 1 では，g'_1, g'_2 のようなクエリグラフとの相関値が高いグラフ検出する．

3 先行研究 TopCor

本節では先行研究 [1] で提案された Top-k 相関グラフ問合せ手法 TopCor について述べる．Ke らは候補グラフ g がクエリグラフとの相関値が正の値を取るならば，g が $\mathcal{D}_q$ 内のグラフの部分グラフになっていることに着目し，グラフ相関問合せ処理を $\mathcal{D}_q$ 内の部分グラフに限定する手法 TopCor を提案した．$|\mathcal{D}_q|$ は $|\mathcal{D}|$ に比べて小さくなるため，TopCor は大規模なグラフデータベースに対しても効率的に相関グラフを探索することができる．

しかし，探索対象となるグラフを $\mathcal{D}_q$ に限定しただけでは，クエリグラフ q が $\mathcal{D}$ 内において頻出する場合等において依然として大きな計算時間を要することとなる．これは q の相関グラフの候補となる部分グラフを候補グラフとすると，$\mathcal{D}_q$ における各グラフの部分グラフを候補グラフとなるためである．そこで TopCor は，候補グラフの枝刈りの基準を設けることで探索対象となるグラフの削減を行う．本節では，TopCor が利用する諸定理と TopCor のアルゴリズムについて説明する．

3.1 TopCor における諸定理

本節では TopCor で用いる諸定理について説明する．$\mathcal{D}_q$ から候補グラフを探索するとき，探索時までに発見された Top-k 相関グラフを保持する優先度付きキュー Q_{cur} を考える．Q_{cur} は $g_Q \in Q_{cur}$ となる各 g_Q の相関値 $\phi(g_Q, q)$ によって降順にソートされている．Q_{cur} の k 番目の相関値を ϕ_{min} とすると，探索中に新たに発見された候補グラフ g は相関値 $\phi(g, q)$ が少なくとも ϕ_{min} を超えていなければ Top-k の相関グラフとはなり得ない．このとき，候補グラフ g の取りうる相関値の上界 $\phi_{\max}(g)$ を以下のように定めることができる．

$$\phi_{\max}(g) = \sqrt{\frac{(1 - \ supp(q))supp(g,q)}{supp(q)(1 - supp(g,q))}}$$

$supp(g,q) = supp(g;\mathcal{D}_q)\cdot supp(q)$ であることより，$\phi_{\max}(g)$ は $\mathcal{D}_q$ 内の候補グラフの支持度を計算することで求めることができる．ゆえに，$\phi_{\max}(g)$ は厳密な相関値 $\phi(g,q)$ よりも小さいコストで得ることができる．これより，候補グラフ g が $\mathcal{D}_q$ から検出された際には最大相関値 $\phi_{\max}(g)$ を計算することで，候補グラフが Q_{cur} に入るか判断することができる．候補グラフが Q_{cur} に入らない場合はその候補グラフは候補から排除することが可能である．この操作を枝刈りと呼び，その性質の詳細を以下の定理 1 に示す．

定理 1. 候補グラフ g は以下を満たすとき，$\phi(g,q) < \phi_{min}$ である．

$$\phi_{\max}(g) < \phi_{\min}$$

$\phi_{\max}(g)$ は 2 つの候補グラフ g_1, g_2 が与えられたとき，$supp\,(g_2,q) \leqq supp\,(g_2,q)$ ならば $\phi_{\max}\,(g_2) \leqq \phi_{\max}\,(g_1)$ である．ここで，支持度の逆単調性より 2 つの候補グラフ g_1, g_2 が $g_2 \subseteq g_1$ のとき，$supp(g_1,q) \leqq supp(g_2,q)$ が成り立つため，$\phi_{\max}(g_1) \leqq \phi_{\max}(g_2)$ である．したがって候補グラフ g が $\phi_{\max}(g) < \phi_{\min}$ となり定理 1 により枝刈りされる場合，g のスーパーグラフも全て枝刈りすることができる．これより，さらに以下の定理 2 が導出できる．

定理 2. 候補グラフ g の $\mathcal{D}_q$ における支持度 $supp(g;\mathcal{D}_q)$ が以下を満たすとき，$\phi(g,q) < \phi_{min}$ である．

$$supp(g;\mathcal{D}_q) < \varsigma\,(\phi_{\min}) = \frac{\phi_{\min}^2}{\phi_{\min}^2 \cdot supp(q) + 1 - supp(q)}$$

このように $supp(g;\mathcal{D}_q)$ の最小支持度閾値として，$\phi_{\min}$ の関数 $\varsigma(\phi_{\min})$ を定めることができる．また，支持度の逆単調性により g が枝刈りされるとき，g の全てのスーパーグラフについても枝刈りすることができる．最小支持度閾値 $\varsigma\,(\phi_{\min})$ は探索過程で Q_{cur} が更新され，$\phi_{\min}$ の値が増加すると $\varsigma\,(\phi_{\min})$ も増加することから，探索する範囲の縮小はより効率的となる．

他にも TopCor ではグラフの包含関係と支持度に基づき，効率的な相関計算を行うルールを作成している．

ルール 1. 候補グラフ g は，$q \subseteq g$ ならば $\phi(g,q) = \phi_{\max}(g)$

TopCor ではこのように効率的な相関計算を行うためのいくつかのルールを利用している．TopCor では $\mathcal{D}_q$ の中から候補グラフの探索は，1 つのエッジからなるグラフを深さ優先的に探索を進めていく．探索の過程で候補グラフが枝刈りが可能ならば，候補グラフとそのスーパーグラフを全て枝刈りする．

3.2 TopCor のアルゴリズム

TopCor は Algorithm 1 , Algorithm 2 , Algorithm 3 から構成される．Algorithm 1 では，初めに探索過程でクエリグラフとの相関グラフを格納する優先度付きキュー Q_{cur} を定義する．次に，探索は $\mathcal{D}_q$ に含まれるグラフの辺 1 つから構成される最小の部分グラフに対して，そのグラフの最大相関値と $\mathcal{D}_q$ での支持度が枝刈りの基準を満たすならば Algorithm 2 を呼び出す．深さ優先探索のプロセスが終了したとき，Q_{cur} に格納されている k 個のグラフを最終的に Top-k 相関グラフとして出力する．

Algorithm 2 では，引数のグラフに辺を 1 つ加える操作をしたグラフ集合に対して深さ優先的に探索を行う．このときに，上述した定理 1,2，ルールを用いて，候補グラフとそのスーパーグラフの枝刈りを行い，効率的に相関グラフ探索を行っていく．探索の過程で Q_{cur} が更新された場合は，$\phi_{\min}$ と $\varsigma\,(\phi_{\min})$ を更新することで，探索が進むたびに枝刈りの基準が厳しくなるようにする．また，クエリグラフのスーパーグラフになる候補グラフや厳密な相関値計算を必要としない候補グラフをルールによって選び，Algorithm 3 を用いて相関計算と探索を行う．

Algorithm 1 TOPCOR

Input: Graph database $\mathcal{D}$, a query q, and an integer k.
Output: The Top-k correlative graphs with respect to q.

```
Initialize an empty queue, Q_cur , of size k;
Obtain D_q;
Find the set of distinct edges, E, in D_q;
Set φ_min and ς(φ_min) initially as 0;
for each e ∈ E do do
    Obtain supp(e; D_q)
    if (φ_min ≦ φ_max(e) and ς(φ_min) ≦ supp(e; D_q)) then
        Invoke MineTopCor (e);
Output the graphs in Q_cur ;
```

Algorithm 2 MINETOPCOR(g)

```
if (q ⊆ g) then
    MineTopCor-1(g);
else
    Let G be the set of valid graphs that are grown from g
    by adding one edge to g;
    Compute supp(g'; D_q) for each g' ∈ G
    if g is a CFG then
        Compute supp(g);
        if (φ_min ≦ φ(g,q)) then
            push g into Q_cur and refine ς(φ_min);
    for each g' ∈ G do
        Apply Rule to determine whether to push g' into Q_cur;
        if supp(g) is computed and supp(g) == supp(g,q)
        then
            MineTopCor-1 (g');
        else
            MineTopCor(g')
        if supp(g) is not computed then
            Apply Rule to determine whether supp(g) needs to
            be computed;
    if (supp(g) is not computed and φ_min ≦ φ_max(g))
    then
        Compute supp(g);
        if (φ_min ≦ φ(g,q)) then
            push g into Q_cur and refine ς(φ_min);
```

Algorithm 3 MINETOPCOR-1(g)

```
if (φ_min ≦ φ_max(g)) then
    push g into Q_cur and refine ς(φ_min) ;
    Grow g by one edge;
    for each valid graph g' grown from g do
        MineTopCor-1(g');
```

3.3 TopCor の問題点

Top-k 相関グラフ問合せは探索すべき候補グラフ数が膨大にあることや相関値計算はコストが高いことが原因となり膨大な計算時間を必要とする．特に，候補グラフとクエリグラフと相関判定を行うための最悪計算量は $\mathcal{O}(|\bar{V}|! \cdot |\bar{V}|)$ である [15]．そのため，グラフデータベースの構成グラフ数が大きくなる場合や，構成グラフサイズが大きくなる場合において相関計算のコストが高くなる．候補グラフが確率 p でグラフデータベース $\mathcal{D}$ から生成されるとすると，TopCor の最悪計算量は $\mathcal{O}(|\bar{V}|! \cdot |\bar{V}| \cdot |\mathcal{D}| \cdot (|\bar{V}| + |\bar{E}|) \cdot p)$ である．タンパク質データベースなどの実際のグラフデータベースを考えた場合には頂点と辺の接続によってさらに計算量は大きくなるため，実世界のデータに対して既存手法を適用した場合でも，依然として膨大な計算時間を必要とする．したがって TopCor の問題点として，より大規模なグラフデータベースでは相関判定のコストが大きくなることから実行時間が大きくなるということが挙げられる．

4 提 案 手 法

本節では提案手法について説明する．本研究の目的は，大規模なグラフデータベースに対して高速に Top-k 相関グラフ問合せを行うことである．この目的に向けて提案手法ではグラフ要約と乱択アルゴリズムに基づく相関グラフ問合せ手法を提案する．本節の構成は次の通りである．4.1 節では提案手法のの概要について説明する．4.2 節では提案手法の事前処理部，4.3 節と 4.4 節では問合せ処理部についてそれぞれ述べる．

4.1 提案手法の概要

提案手法の基本アイディアは，データベース内の各グラフを要約することによってグラフサイズを小さくし相関判定のコストを小さくすることである．グラフ要約とは 2 つの頂点を 1 つの頂点に集約する操作を繰り返し行うことでグラフの頂点数と辺数を小さくすることである．グラフ要約の例を図 2 に示す．図 2 において，$\langle$ 頂点ラベル，辺ラベル，頂点ラベル $\rangle$ の接続を考えたとき，$\langle a, 1, a\rangle$ の接続をしている辺の両端の頂点を 1 つの頂点 $n1$ に集約している．同様に $\langle a, 1, b\rangle$，$\langle a, 1, c\rangle$ の接続も同じように $n2, n3$ へとそれぞれ集約をする．このとき，辺の両端の頂点のいずれかが集約された頂点であった場合は集約をしない．例えば，$\langle c, 1, n1\rangle$ の辺は集約を行わない．この集約する操作を辺がこれ以上集約できなくなるまで要約をすることで要約されたグラフが構築される．

要約されたグラフから構成されるグラフ集合を要約データベースとすると，要約データベース内のグラフはグラフサイズが小さいため，相関グラフ探索の際に生成する候補グラフ数が小さくなる．また，候補グラフの支持度の計算コストが小さくなる．このように要約データベースでの Top-k 相関グラフ問合せは高速に行うことができる．しかし，グラフ要約は Top-k 相関グラフ問合せに対して偽陽性・偽陰性となる結果をもたらす場合があるため，提案手法では乱択アルゴリズム [13] を利用することにより確率的に偽陽性・偽陰性となる結果を除外し，高速で高精度な Top-k 相関グラフ問合せを実現する．

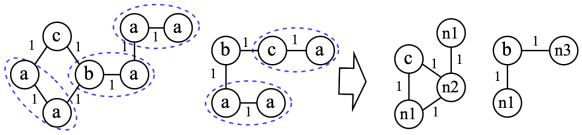

図 2: グラフ要約の例

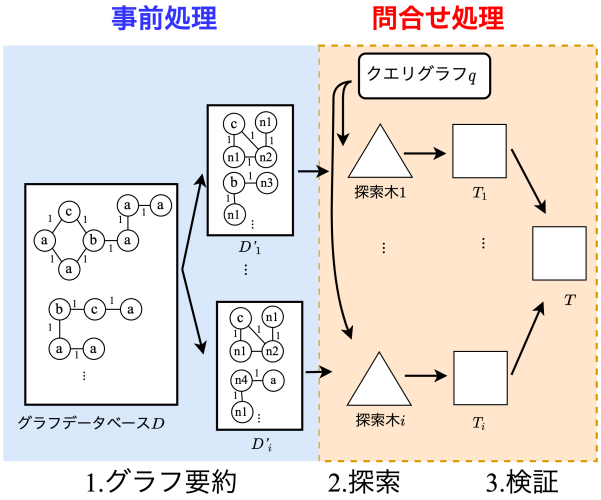

図 3: 提案手法概要

提案手法は以下の 3 つで構成される．

(1) グラフ要約： グラフデータベース $\mathcal{D}$ の各グラフに対してランダムに辺を集約した要約グラフデータベース $\mathcal{D}'$ を構築する．

(2) 探索： $\mathcal{D}'$ に対して TopCor を適用する．グラフ要約は Top-k 相関グラフ問合せに対して偽陽性・偽陰性となる結果をもたらす場合があるため，複数の要約グラフデータベースにおける TopCor の結果である k 個の要約された部分グラフからなる部分グラフ集合 T を複数生成する．

(3) 検証： 複数の T をグラフデータベース $\mathcal{D}$ と照合しグラフ復元を行い，正確な Top-k 相関グラフを求める．

4.2 提案手法フレームワーク

提案手法の概要を図 3 に示す．グラフデータベース $\mathcal{D}$ が与えられ，事前処理としてデータベースの各グラフについてグラフ要約行い，i 個の要約データベースを構築する．問合せ処理は，まず与えられたクエリグラフと要約データベース $D'_1, D'_2, \ldots, D'_i$ に TopCor を適用することで，相関グラフの探索を行う．次に探索で得られた相関グラフ集合 $T_1, T_2, \ldots, T_i$ について検証を行い，Top-k 相関グラフ T を得る．

グラフ要約，探索，検証をまとめたアルゴリズムを Algorithm 4 に示す．2，3 行目では $\mathcal{D}$ の要約を i 回数分だけ行う．その後，6 行目でそれぞれの要約グラフデータベースに対して TopCor を実行する．その結果をそれぞれ $T_1, T_2, \ldots, T_i$ に格納し，8 行目ではそれぞれの要約グラフデーベースに対する相関問合せで獲得した Top-k 相関グラフの検証を行う．9 行目で検証された最終的な結果，すなわち Top-k 相関グラフを出力する．

Algorithm 4 提案手法

Input: Graph database $\mathcal{D}$, a query q, and an integer k.
Output: The Top-k correlative graphs with respect to q.
1: Initialize an empty queue, Q_{cur} , of size k;
2: **for** $j = 1$ to i **do**
3: $(\mathcal{D}'_j, q'_j)$=SUMMARIZE$(\mathcal{D}, q)$;
4: **for** $j = 1$ to i **do**
5: T_j = TopCor$(\mathcal{D}'_j, q'_j)$;
6: VERIFICATE$(D'_1, D'_2, \ldots, D'_i, T_1, T_2, \ldots, T_i)$;
7: Output the graphs in Q_{cur};

4.3 グラフ要約

グラフ要約は $\mathcal{D} = \{g_1, g_2, ..., g_N\}$ について $\mathcal{D}$ 内の各グラフを要約したグラフデータベース $\mathcal{D}' = \{g'_1, g'_2, ..., g'_N\}$ を構築する．$\mathcal{D}$ が与えられたとき，グラフ要約はまず以下のように定義されるラベル対集合 R_j をサンプリングする．

定義 3 (ラベル対集合)**.** $\mathcal{D} = \{g_1, g_2, ..., g_N\}$ が与えられたとき，ラベル対集合を R_j とし，以下のように定める．

$$R_j = R_{j-1} \cup \{\langle l(u), l(u,v), l(v)\rangle | \sigma(E^{i-1}) = (u,v)\}$$

ただし，$R_0 = \emptyset$ であり，$\sigma(E^{i-1})$ は以下に定義するエッジ集合 E^{i-1} からランダムに 1 つのエッジを選択する関数である．

$$E^{i-1} = \bigcup_{g_j \in \mathcal{D}} \bigcup_{\substack{\langle l(u), l(u,v), l(v)\rangle \\ \in R_{i-1}}} E_j(\langle l(u), l(u,v) l(v)\rangle)$$

ただし，$E_j(\langle l(u), l(u,v) l(v)\rangle) = \{(u', v') \in E_j | l(u') \neq l(v') \neq l(u) \wedge l(u') \neq l(v') \neq l(v)\}$ である．
すなわち，R_j は (性質 1) R_j 内のラベル対は少なくとも 1 つ以上 $\mathcal{D}$ 内のいずれかのグラフで隣接しており，(性質 2) ラベル対の両端点は R_j 内の他のラベル対と重複しない，という 2 つの性質を満たした j 個のラベル対の集合である．提案手法は次の手順を R_j の要素がなくなるまで繰り返すことで，要約データベース $\mathcal{D}'$ を構築する．
(2-1) R_j からラベル対 r を取り出す．
(2-2) $\mathcal{D}$ 内の各グラフ g_i に対して，r を含む場合，それらのノードを 1 ノードに集約する．

ラベル対 r_i は，ある辺に関して隣接頂点のラベル 2 つとその頂点間の辺ラベルによって構成される．集約された 1 頂点のラベルはそれぞれのラベル対 r に対応して，辺と対応するラベル対が異なると集約された頂点は異なるラベルを持つ．

グラフ要約の例を図 4 に示す．まず，$\mathcal{D}$ からラベル対集合 R がサンプリングされる．このサンプリングは R の定義にしたがって行われる．次に $\mathcal{D}$ の各グラフ g に対して，R の最初のラベル対 $\langle a, 1, a\rangle$ を含む辺を全て 1 つの頂点に集約されたグラフを構築する．このとき，集約された頂点ラベルは *n1* となる．次に R の次の要素 $\langle a, 1, b\rangle$ についても同様の操作を行い，R の全ての要素についてグラフの集約操作を行う．この結果，$\mathcal{D}'$ が構築され，これを要約グラフデータベースとする．

最後にグラフ要約のアルゴリズム Algorithm 5 について説明する．グラフ要約はグラフデータベース $\mathcal{D}$ とクエリグラフ q が与えられ，これらを要約した $\mathcal{D}', q'$ を構築する．1 行目では $\mathcal{D}$ からラベル対集合 R をサンプリングする．ラベル対はグラフデータベースの各グラフの辺からサンプリングされる．2 行目以降では，各ラベル対に対して $\mathcal{D}$ の各グラフ g において，ラベル対と一致する辺の両端の頂点を 1 つの頂点に集約する．11 行目では $\mathcal{D}$ の全てのグラフについて要約が完了したら，同様の手順でクエリグラフも要約する．

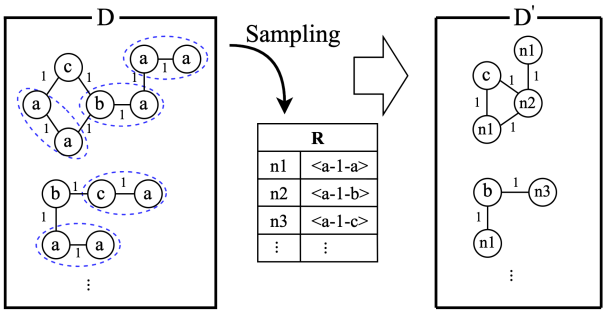

図 4: 要約データベース構築の例

Algorithm 5 SUMMARIZE$(\mathcal{D}, q)$ AND q'

Input: Graph database $\mathcal{D}$, a query q.
Output: Summarized Graph Databases $\mathcal{D}_i$
集約された頂点 : meta vertex
1: Sampling, $R = \{r_1, r_2, \ldots, r_j\}$;
2: **for** each $g \in \mathcal{D}$ do **do**
3: **for** each $r_i \in R$ do **do**
4: **for** each $e = (v_1, v_2) \in E$ do **do**
5: **if** v_1 and v_2 are not meta vertex
and ($l(v_1)$,$l((v_1, v_2))$,$l(v_2)$) == r_i **then**
6: Summarize v_1 and v_2 to v_i;
7: In the same way, summarized q to q';
8: Output $\mathcal{D}', q'$;

4.4 探　　索

本節では探索について説明する．要約データベース $\mathcal{D}'_1, \mathcal{D}'_2, \ldots, \mathcal{D}'_i$ から，対応するラベル対集合 $R_1, R_2, \ldots, R_i$ によって要約されたクエリグラフ $q'_1, q'_2, \ldots, q'_i$ に対する Top-k 相関グラフ $T_1, T_2, \ldots, T_i$ をそれぞれ検出する．このとき，i は乱択アルゴリズムによって複数回実行されるイテレーション回数である．提案手法では要約データベースにおける Top-k 相関グラフ問合せの処理には先行研究 TopCor [1] を利用する．

この要約グラフデータベース $\mathcal{D}'$ における TopCor による探索は $\mathcal{D}$ に対する探索に比べて探索する候補グラフ数は小さくなると考えられる．$\mathcal{D}'$ の探索時には, 要約されたグラフは頂点数と辺数が小さくなるため生成される候補グラフの数が小さくなり，支持度計算のコストも小さくなるため，候補グラフとクエリグラフの相関値計算のコストも小さくなると考えられる．したがって $\mathcal{D}'$ での Top-k 相関グラフの探索は $\mathcal{D}$ での Top-k 相関グラフ探索の時間に比べて小さくなると考えられる．

乱択アルゴリズムの導入： $\mathcal{D} = \{g_1, g_2, ..., g_N\}$ とクエリグラフ q が与えられたとき，ラベル対集合 R を用いて $\mathcal{D}$ 内の各グラフを要約したグラフデータベース $\mathcal{D}' = \{g'_1, g'_2, ..., g'_N\}$ と要約

されたクエリグラフ q' を構築する．このとき，$\mathcal{D}$ における q の Top-k 相関グラフ T と，$\mathcal{D}'$ における q' の Top-k 相関グラフ T' を考える．T と T' はそれぞれ k 個の部分グラフを含む集合である．ある部分グラフ g について R によって要約されたグラフを g' とすると，要約により以下の 2 つの場合が考えられる．

定義 4 (偽陰性)**.** $g \in T$ であるが，$g' \notin T'$ のとき，グラフ要約による偽陰性である．

定義 5 (偽陽性)**.** $g \notin T$ であるが，$g' \in T'$ のとき，グラフ要約による偽陰性である．

グラフ要約処理はランダムに辺を集約するため，集約の順序やラベル対集合 R の内容に依存して Top-k 相関グラフ問合せにおいて，上記の偽陽性・偽陰性を生じさせる場合がある．そこで提案手法では乱択アルゴリズムを導入することで確率的に偽陽性・偽陰性の排除を図る．具体的には，複数のラベル対集合 $R_1, R_2, \ldots, R_i$ を構築し，それぞれの集合をもとに要約データベース $\mathcal{D}'_1, \mathcal{D}'_2, \ldots, \mathcal{D}'_i$ を構築する．複数のラベル対集合によって要約されたデータベースは互いに異なるグラフデータベースとなる．以降の処理で，複数の要約データベースと同様に要約されたクエリグラフに対して Top-k 相関グラフ問合せを実行することで，偽陽性・偽陰性となる部分グラフを除外を図る．

4.5 検　　証

この節では検証について説明する．i 個の要約データベースに対して Top-k 相関グラフ問合せを行った $T_1, T_2, \ldots, T_i$ に対して正確な Top-k 相関グラフを求めるために検証を行う．$T_1, T_2, \ldots, T_i$ に含まれる k 個のグラフは要約データベースの部分グラフである．したがって，$T_1, T_2, \ldots, T_i$ を相関グラフとして検証するためには元々のグラフデータベース $\mathcal{D}$ におけるグラフの部分グラフに変換する必要がある．$T_1, T_2, \ldots, T_i$ に含まれる k 個の要約グラフは，$\mathcal{D}$ の各グラフにおける隣接リスト情報とラベル対集合 R に基づき要約を解除して復元する．

具体例として図 5 のように，グラフ g の要約グラフを g'，要約グラフデータベースに相関問合せをした結果獲得した T' とし，T' に含まれる部分グラフを g'' を考える．復元は g'' に対して，g の色付きの部分グラフを求める操作である．操作としては次の手順である．最初に g' のある頂点について集約された頂点であった場合は元々の 2 つの頂点を考えるため，R を参照する．2 頂点間の辺をつなげた後に g の隣接リスト情報から隣接する頂点を選択し辺をつなげる．集約された頂点ではない場合は隣接リストから隣接頂点をそのままつなげる．この操作を繰り返し行うことでグラフの復元を行う．このとき，$\mathcal{D}$ において復元したグラフのスーパーグラフとなるグラフを一意に求める．T' の全ての部分グラフに対して同様の操作をし，得られたグラフ集合に対して探索を行う．したがって，検証では復元したグラフとその部分グラフとスーパーグラフを含めて探索を行い，厳密な Top-k 相関グラフを求めるために検証を行う．

Algorithm 6 では，1 行目にグラフデータベースにおける index を保持する集合 *IDs*，検証するグラフ集合を G とする．2 行目以降では $T_1, T_2, \ldots, T_i$ のそれぞれの k 個のグラフに対して復元を行い，$\mathcal{D}$ において復元したグラフのスーパーグラフとなるグラフを一意に求める．このグラフの $\mathcal{D}$ における index を *IDs* に重複なく格納する．11 行目では TopCor を呼び出すが，対象となるグラフ集合 G は最大 $|k \cdot i|$ 個のグラフを含む．これはそれぞれの T でのグラフにはスーパーグラフや部分グラフなどのグラフ包含関係にあるものが存在するためである．このグラフ集合 G に対して TopCor を呼び出すことで，要約データベースにおいて求めた相関グラフの部分グラフとスーパーグラフの検証を行うことができる．最終的に Q_{cur} に格納されている k 個のグラフを出力し，これが Top-k 相関グラフとなる．

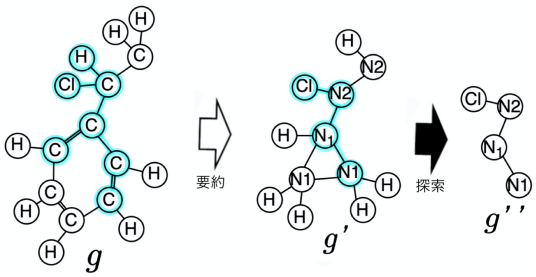

図 5: グラフ復元の図

Algorithm 6 VERIFICATE($\mathcal{D}'_1, \mathcal{D}'_2, \ldots, \mathcal{D}'_i, T_1, T_2, \ldots, T_i$)

Input: Summarized Graph databases $\mathcal{D}'_1, \mathcal{D}'_2, \ldots, \mathcal{D}'_i$, Graph sets $T_1, T_2, \ldots, T_i$.
Output: The Top-k correlative graphs with respect to q.
i : 反復回数
g.id : Index of Graph Database
1: Set IDs , G;
2: **for** j=1 to i **do**
3: **for** each $g' \in T_j$ do **do**
4: restore $g \leftarrow g'$;
5: $IDs \cap g.id$;
6: **for** each $id \in IDs$ do **do**
7: push $\mathcal{D}[id]$ into G;
8: Call TopCor(G,q,k);
9: Output the graphs in Q_{cur};

5 評 価 実 験

提案手法の計算時間と処理精度の評価を行うために，実データを用いて提案手法と既存研究 TopCor [1] の比較を行った．精度の評価指標には適合率を用いる．適合率は，提案手法が出力した k 個の相関部分グラフと，TopCor が出力した k 個の相関部分グラフが一致した割合とする．実験は以下の 4 つの点に着目して提案手法の有効性を検証する．

- 出現頻度の異なるクエリグラフに対する性能変化
- Top-k の値 k を変化させた場合における性能変化
- データベース $\mathcal{D}$ に含まれるグラフ数に対する性能変化
- $\mathcal{D}$ を構成する各グラフの大きさに対する性能変化

提案手法は事前計算部分である **(1) グラフ要約**と問合せ処理である **(2) 探索**と **(3) 検証**の 2 つに分けられる．5.1 節では提案手法における事前処理の実行時間を，5.2 節以降はパラメータを変化させた場合の問合せ処理の評価実験の結果を示す．各節における提案手法の要約グラフデータベースの数は 5 であり，5

表 2: 事前処理の実行時間

データセット	グラフ数	実行時間 [秒]	頂点集約率	辺集約率
nci10k	10,000	3.592	0.636	0.641
nci20k	20,000	7.969	0.645	0.652
nci40k	40,000	15.537	0.660	0.675
nci60k	60,000	25.387	0.659	0.667
nci4.2k	4,200	7.224	0.733	0.742

つの要約グラフデータベースに対して探索を行い，検証を行う．

5.1 実験環境とデータセット

提案手法のアルゴリズムは C++を用いて実装し，コンパイルオプションには-O2 を用いた．グラフは隣接リスト構造を用いて実装し，部分グラフ同型の判定には C++のライブラリ関数 [15] を用いた．全ての実験は Intel Xeon Platinum 8268 2.9 GHz，1 TiB RDIMM で構成される Linux サーバ上で行った．

実験に用いたデータセットは実際の癌や AIDS などのタンパク質のグラフデータベースである [14]．このグラフデータベースは原子を頂点とし，原子間の結合を辺として表現されている．データセット nci10k, nci20k, nci40k, nci60k はそれぞれ 10,000 , 20,000 , 40,000 , 60,000 個のグラフから構成される．またデータセット nci4.2k は 800,000 個のタンパク質構造のグラフデータベースから頂点数が 100 を超える 4,200 個のグラフから構成されるデータセットである．これらのグラフデータベースの詳細を表 1 に示す．実験に用いるクエリグラフは [0.001,0.005] , (0.005,0.01] , (0.01,0.03] , (0.03,1) の範囲の支持度を持つクエリグラフを利用し，それぞれ Q_1, Q_2, Q_3, Q_4 とする．

表 1: グラフデータセット

データセット	グラフ数	平均頂点数 $\lvert\bar{V}\rvert$	平均辺数 $\lvert\bar{E}\rvert$
nci10k	10,000	32.1994	32.4929
nci20k	20,000	34.6061	35.0607
nci40k	40,000	35.2219	35.7741
nci60k	60,000	35.5829	36.2005
nci4.2k	4,200	133.967	138.156

5.2 事前処理について

本節では提案手法における事前処理であるグラフデータベースの要約処理について実験を行う．提案手法における事前処理であるグラフデータベースの要約処理の実行時間を計測した結果を示す．それぞれのデータベースに対する事前処理の実行時間を表 2 に示す．頂点・辺集約率は要約前のグラフデータベースの平均頂点数・辺数を要約後のグラフデータベースの平均頂点数・辺数で割った数値である．表 2 より，グラフデータベースに対するグラフ要約にはグラフデータベースの構成グラフ数が大きくなる程，事前処理に時間がかかることがわかる．このとき，グラフ要約によって構成されるグラフの平均頂点数・辺数はおよそ 63 %から 74 %にまで集約された．事前処理は，グラフデータベースの構成グラフ数に比例して実行時間が大きくなるが，5.3 節以降で後述する問合せ処理の時間に比べて小さい．

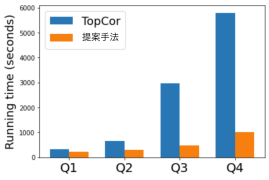

図 6: nci10k における問合せ処理時間比較

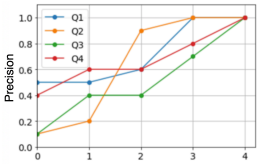

図 7: 適合率

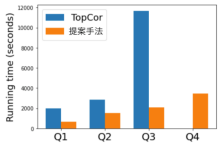

図 8: nci20k における問合せ処理時間比較

5.3 異なるクエリグラフによる影響

提案手法におけるクエリグラフを Q_1 から Q_4 まで変化させたときの問合せ処理の時間と TopCor の計算時間を比較した結果を図 6，8 に，nci10k における要約データベースの数を 1 から 5 まで変化させた際の適合率の推移を図 7 に示す．

図 8 において，TopCor はクエリ Q_4 では 4 時間以内に実行が終わらなかったので打ち切った．どのクエリグラフに対しても提案手法は TopCor よりも問合せ処理が高速であり，支持度が大きいほど提案手法が高速になることがわかる．図 6 の Q4 において，提案手法は TopCor よりも最大 6.1 倍高速に問合せ処理を行うことができた．これはグラフ要約によって相関問合せのの候補グラフの数が小さくなることや相関判定のコストが小さくなることが高速化の要因だと考えられる．

図 7 は提案手法で得られた Top-k 相関グラフの適合率が 100%になったときの適合率の推移を示す．提案手法は乱択アルゴリズムを用いて Top-k 検出を行うため，全ての実行に対して適合率が 100%になるとは限らない．クエリグラフの特性や要約のランダム性によるが，どのクエリグラフでもイテレーションを経る毎に適合率が上がる傾向にあることがわかる．

5.4 k による影響

提案手法と既存手法において，k の値を 10 から 300 まで変えて比較実験を行う．図 9，10 において実験に用いたデータベースはそれぞれ nci10k , nci20k であり，クエリグラフには支持度が Q1 に属するグラフを使用した．

図 9，10 より，どの k でも提案手法が高速であることが分かる．$k = 10$ から $k = 100$ までは高速化の倍率はあまり変わらなかったが，k が大きくなると徐々に高速化の倍率が上がり，図 9 において $k = 300$ で約 1.8 倍の高速に問合せ処理を行うことができた．k の値が上昇すると求める相関グラフが多くなるとともに，探索の過程での k 番目の相関値は小さくなるので，TopCor での枝刈りの基準は小さくなる．そのため，k の値が大きくなると実行時間が大きくなる．しかし，要約グラフデータベース内での探索は TopCor での探索に比べて候補グラフ数が小さくなるとともに，支持度計算のコストも元々のグラフデータベースに比べて小さいため，提案手法では k の値が大きく

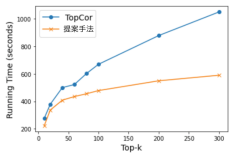

図 9: nci10k での k の変化

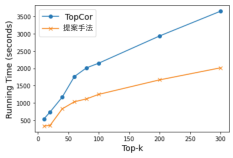

図 10: nci20k での k の変化

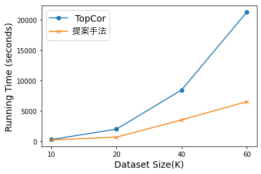

図 11: データベース数を変えた場合の問合せ処理時間

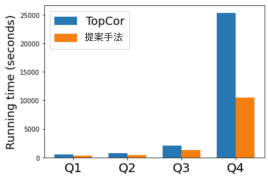

図 12: グラフサイズを変えた場合の問合せ処理時間

なっても問合せ処理の実行時間が抑えられると考えられる．

5.5 データベースのサイズによる影響

データセットのサイズを変えて提案手法と既存手法の比較実験を行う．データセットは nci10k から nci60k を使用した．クエリグラフは Q_1 に属するグラフを使用し，k=10 である．

図 11 より，データベースが含むグラフ数が大きくなると，提案手法はより高速に問合せ処理を行うことができると分かる．特に図 11 では nci60k のとき，提案手法は最大で 3.3 倍高速に処理することができる．これはデータベースに含まれるグラフ数が大きくなり，グラフデータベースが大規模になると，候補グラフ数が大きくなると共に支持度計算のコストが大きくなるのに対して，提案手法では要約データベースを考えることで候補グラフの探索する範囲を縮小し，相関問合せのコストを小さくしたためであると考えられる．

5.6 構成されるグラフサイズによる影響

データベースを構成するグラフサイズを大きくした場合の提案手法と既存手法の比較実験を行う．この実験で使用するデータベースは nci4.2k であり，クエリグラフには支持度が Q_1, Q_2, Q_3, Q_4 に属するグラフを使用した．

提案手法の問合せ処理時間と TopCor の実行時間を図 12 に示す．この結果より，大規模なグラフデータベースに対して Top-k 相関グラフ問合せを行うのは実行時間が多くかかることがわかる．提案手法と TopCor を比較すると，クエリグラフの支持度が大きくなるほど実行時間に差が生まれることがわかる．この実験では提案手法は Q_4 において最大 2.5 倍の高速であることが分かる．グラフサイズが大きなグラフデータベースにおける要約アルゴリズムは，nci10k のような比較的小さなグラフから構成されるグラフデータベースに比べて高速化率が上がると想定していたが，実際の実験では高速化率は nci10k に比べて小さかった．これは検証の段階では元々のグラフデータベースにおいて相関問合せを行うため，相関問合せのコストは TopCor と変わらないためだと考えられる．

6 結　　論

本研究では事前に要約したデータベースを構築することで相関判定コストを下げ，高速で高精度な Top-k 相関グラフ問合せ手法を提案した．実データを用いた評価実験により，提案手法は既存手法よりも高速に計算できることが示された．

今後の課題として，提案手法では要約による偽陽性・偽陰性に対応するために乱択アルゴリズムを利用したが，アルゴリズムのイテレーション回数に対してどの程度の確率で Top-k 相関グラフの一致率を保証することができるかという信頼区間の理論解析が課題となる．また，グラフ要約に関しても提案手法では集約されたノードは集約しないという要約手続きを行ったが，要約率を上げるために集約されたノードも集約することでさらに集約率を上げることも考えられる．これによって要約率が上がることで相関問合せのコストをさらに下げることができ，より高速な相関問合せが可能になると考えられる．

謝　　辞

本研究の一部は JST さきがけ（JPMJPR2033）による支援を受けたものである．

文　　献

[1] Yiping Ke, James Cheng, and Jeffrey Xu Yu. Top-k Correlative Graph Mining. In Proc. SIAM International Conference on Data Mining , Pages 1038-1049, 2009.
[2] Yiping Ke, James Cheng, and Wilfred Ng. Correlation search in graph databases. In Proc. ACM SIGKDD international conference on Knowledge discovery and data mining , Pages 390-399, 2007.
[3] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In Proc. IEEE International Conference on Data Mining , Pages 721, 2002.
[4] A. Inokuchi, T. Washio, and H. Motoda. An aprioribased algorithm for mining frequent substructures from graph data. In Proc. Principles of Data Mining and Knowledge Discovery , Pages 13-23, 2000.
[5] M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. In Proc. IEEE International Conference on Data Mining , 2001.
[6] Y. Ke, J. Cheng, and W. Ng. Efficient correlationsearch from graph databases. In Proc. IEEE Transactions on Knowledge and Data Engineering , Pagea 1601-1615 , 2008.
[7] Yiping Ke, James Cheng, Jeffrey Xu Yu Efficient Discovery of Frequent Correlated Subgraph Pairs. In Proc. IEEE International Conference on Data Mining , Pages 239-248 , 2009
[8] Arneish Prateek, Arijit Khan, Akshit Goyal and Sayan Ranu Mining Top-k Pairs of Correlated Subgraphs in a Large Network. In Proc. the Very Large Data Base Endowment , Pages 1511-1524 , 2020.
[9] J. Cheng, Y. Ke, and W. Ng. FG-Index: Towards verification-free query processing on graph databses. In Proc. ACM SIGMOD international conference on Management of data , Pages 857-872, 2007.
[10] H. Reynolds. The analysis of cross-classifications. The Free Press, New York, 1977
[11] C. Chen, C. X. Lin, M. Fredrikson, M. Christodorescu,X. Yan, and J. Han. Mining Graph Patterns Efficiently via Randomized Summaries. In Proc. the Very Large Data Base Endowment , Pages 742-753, 2009.
[12] M. Worlein, T. Meinl, I. Fischer, and M. Philippsen. A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. In Proc. Conference on Principles and Practice of Knowledge Discovery in Databases , Pages 392-403 , 2005.
[13] Rajeev Motwani, Prabhakar Raghavan Randomized algorithms First published 1995 , the Press Syndicate of the University of Cambridge , ISBN 0-521-47465-5
[14] NCI Databases. https://cactus.nci.nih.gov/ , (January 1st, 2022 Accessed)
[15] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. In Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence , Pages 1367-1372, 2004.
[16] Hiroaki Shiokawa, Toshiyuki Amagasa, Hiroyuki Kitagawa. Scaling Fine-grained Modularity Clustering for Massive Graphs. In Proc. the Twenty-Eighth International Joint Conference on Artificial Intelligence Main track. Pages 4597-4604, 2019.
[17] Hiroaki Shiokawa, Makoto Onizuka. Scalable Graph Clustering and Its Applications. Encyclopedia of Social Network Analysis and Mining , Springer , 2017