経路を用いた高速なサブグラフ編集距離問合せ

堀内 美聡† 佐々木勇和† 肖 川† 鬼塚 真†

† 大阪大学大学院情報科学研究科 〒 565–0871 大阪府吹田市山田丘 1-5

E-mail: †{horiuchi.misato,sasaki,chuanx,onizuka}@ist.osaka-u.ac.jp

あらまし サブグラフ編集距離 (SED) 問合せは，グラフ間の距離として SED を用いた類似サブグラフ問合せであり，分子構造の分析や，ナレッジグラフの検索，パターン認識など様々な応用に用いられる．SED の計算は NP 困難であるため，実践的に効率的な手法が必要である．本研究は，グラフ上の経路に基づいて検索範囲を効果的に枝刈りし，効率的で高精度な SED 問合せ手法を提案する．提案手法では，クエリグラフと対象グラフの経路を比較することで対象グラフの SED の下限値と上限値を計算し，明らかに検索結果に入らない対象グラフを枝刈りする．さらに，下限値と上限値を用いることで，グラフの SED 計算そのものも高速化する．2 種のグラフデータを用いた評価実験により，提案手法は既存手法と比較して高精度かつ高速な SED 問合せが可能であることを示す．

キーワード グラフデータ処理, 問合せ処理, アルゴリズム, グラフ類似度, サブグラフ編集距離

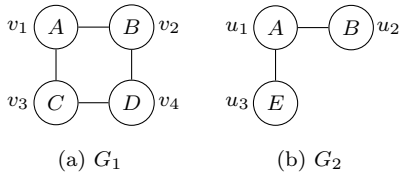
1 背 景

グラフは頂点とそれらを接続する辺により表現されるデータ構造であり，相互に関係する情報をもつデータのモデル化のために利用される．分子グラフや，生物情報科学データ，ウェブグラフ，ソーシャルネットワークグラフ [1] など幅広く利用されている．例えば，分子グラフでは，原子を頂点，原子結合を辺で表すことにより分子構造をモデル化できる．

サブグラフ編集距離（SED）問合せは，検索対象のグラフと入力のクエリグラフのグラフ間距離に SED を利用するグラフ問合せである [2,3]．SED はクエリグラフに編集操作をして対象グラフとサブグラフ同型にするために必要な最少の編集操作回数であり，編集操作は，頂点および辺に対し，挿入/削除/属性変更のそれぞれ 3 操作である [4]．SED 問合せは，検索対象のグラフに対し，入力のクエリグラフとの SED が小さい上位 k 件の対象グラフを出力する Top-k 検索と検索範囲内の SED を持つ全ての対象グラフを出力する範囲検索がある．サブグラフマッチングに比べて，エラーやノイズに寛容な検索を行うことができ，分子構造 [5]，グラフパターンマイニング [6] などの様々な分野に活用できる．

SED の例を図 1 に示す．G_1 をクエリグラフ，G_2 を検索対象グラフとした場合，v_3 のラベルを変更，v_4 を削除，v_2 と v_3 間の枝を削除，v_3 と v_4 間の枝を削除という 4 つの操作で G_1 が G_2 のサブグラフ同型となるため，$SED(G_1, G_2)$ は 4 となる．一方で，G_2 をクエリグラフ，G_1 を検索対象グラフとした場合，v_3 のラベルを変更するのみでサブグラフ同型となるため，$SED(G_2, G_1)$ は 1 となる．

動機．SED は類似サブグラフの検索に有用であるが， NP 困難であり，素朴に値を求める方法は実用的でない．そのため，様々な SED 問合せの高速化技術が開発されている [7]．Closure-tree [2] は複数のグラフの情報を集約して構築する木構造索引と，近似 SED 計算手法による Top-k 検索手法である．しかし，近似 SED 計算手法は正確な SED を計算出来ないため，範囲検索に適さず，Top-k 検索も正確でない．また，SGSimSearch [8] は SED が閾値以下となる対象グラフを抽出する検索手法である．しかし，検索範囲の下界を与える検索はできず，Top-k 検索にも適さない．さらに，機械学習に基づく手法 [3] は，モデルの学習に時間を要することに加え，モデルの学習に近似 SED 手法を利用しており，さらに機械学習の性質から正確な SED を計算できない．著者らの知る限り，Top-k 検索と範囲検索の両方に対して高速かつ高精度な SED 問合せが可能な既存手法は存在しない．



(a) G_1 (b) G_2

図 1: SED の例

貢献．そこで本稿では，Top-k 検索と範囲検索の両方で高速かつ正確な SED 問合せを目的とし，グラフの経路を用いたグラフの枝刈りと効率的な SED 計算アルゴリズムによる SED 問合せ手法を提案する．提案手法は次の 3 つの要素で構成される：(1) 経路の比較による下限値・上限値計算，(2) 問合せの枝刈り技術，(3) 上限値を更新しながら枝刈りする SED 計算アルゴリズム．まず，提案手法はクエリグラフと対象グラフ内に存在する経路（節点ラベルのシーケンス）の比較により，小さい計算コストで SED の下限値と上限値を算出する．次に，SED 問合せアルゴリズムでは，経路より求めた SED の下限値と上限値を利用して検索対象グラフの枝刈りを行う．最後に，SED 計算アルゴリズムでは，下限値と上限値を利用し，効率的な SED 計算を行う．この際，SED 計算中に上限値を更新しながら探索することで検索空間を削減する，

実験では 2 種のグラフデータを用いて Top-k 検索と範囲検

索にて提案手法の検索精度と効率性の評価を行う．検索精度の評価実験では， Top-k 検索では nDCG@10 と RMSE ，範囲検索では F 値を比較することで，提案手法が正確な SED を計算できることを示す．効率性の評価実験では，検索時間と経路抽出時間を比較することで，提案手法が効率的に検索できることを示す．加えて，既存の近似 SED 計算技術を利用することで，高速かつ既存手法以上の高精度で近似 SED を計算できることを示す．

本稿の構成は次の通りである．まず，2 章で関連研究を示し，3 章にて事前知識と問題定義を説明する．4 章に経路に基づく SED の下限値と上限値の計算について述べる．さらに，5 章に下限値と上限値によって効率化した SED 問合せを示す．6 章にて実験結果を示し，7 章にて本稿をまとめる．

2 関 連 研 究

本章では SED 問合せの関連研究として，サブグラフ編集距離 (SED) ，グラフ編集距離 (GED)，サブグラフマッチングについて述べる．

SED は，グラフの局所的な類似性を表す指標であり，クエリグラフに編集操作をして対象グラフのサブグラフ同型にするために必要な最少の編集操作回数である [4]．SED 計算手法は，フィルタ–検証アルゴリズムが一般的であり，これはフィルタで検索範囲を枝刈りし，検証により SED 計算を行う．Closure-tree [2] は複数のグラフを一つのグラフにまとめた (union) インデックスとマッピング手法による高速な SED 計算手法である．また，AppSub [7] はスター構造に基づいて下限値と上限値を求め，枝刈りをする近似 SED 計算手法である．しかし，これらは正確な SED を保証することはできず，また範囲検索にも適していない．SGSimSearch [8] は経路に基づいて下限値を求め，GrafD-Index [9] は特徴間の距離の索引を作成することで高速に検索を行う．しかし，これらは SED が閾値以下となる対象グラフを抽出する検索手法であるため，検索範囲の下界を与える検索や，Top-k 検索に適さない．GREED [3] は機械学習によって SED を学習し高速に SED を計算するという手法である．しかし，モデルの学習に時間を要することに加え，モデルの学習に近似 SED 手法を利用しており，さらに機械学習の性質から正確な SED を計算できない．

GED は，同程度のサイズのグラフの類似性を測るために適した指標で，クエリグラフに編集操作をして対象グラフとグラフ同型にするために必要な最少の編集操作回数である．GED の高速な計算手法は活発に研究されており，κ–AT への分解に基づく手法 [10]，スター型グラフを利用し近似計算する手法 [7]，機械学習に基づく手法 [11] などが存在する．しかし，クエリグラフと検索対象のグラフサイズの差が大きいと GED の値も大きくなってしまうなど，類似グラフ検索のために指標としての利用は課題がある．また，対象グラフのあらゆるサブグラフに対して GED を計算することで SED の計算が可能であるが，GED 計算が高コスト，さらにサブグラフのパターン数は膨大になるため，SED 問合せへの利用は難しい．

サブグラフマッチングは，クエリグラフに対してサブグラフ同型なグラフを列挙する技術であり，SED 問合せにおいて SED がゼロの場合の範囲検索と同等である [12]．FG-index [13] は頻出サブグラフに基づき索引を作成することで高速化する．多数のマッチングが存在するサブグラフはインデックスで高速に処理し，少数のマッチングはそのままサブグラフマッチングを実行することで，効率よくサブグラフ同型を抽出する．また，GADDI [14] は頻出サブグラフに基づく距離を索引にすることで高速化する．しかし，サブグラフマッチング技術を用いて SED 問合せを行う場合，クエリグラフを変更しながら繰り返しサブグラフマッチングを行うことで実現できる．しかし，クエリグラフの変更パターン数が指数関数的に増加するため現実的ではない．

3 事 前 知 識

本章では事前知識として，グラフ，経路，および SED 問合せについて説明する．

グラフ．グラフ G を三つ組 (V, E, L) として表現する．V は頂点の集合，$E \subseteq V \times V$ は $v, v' \in V$ を接続する辺 $e(v, v')$ の集合，L は各頂点からラベル集合への単射である．$V(G)$，$E(G)$ をそれぞれ G の頂点集合，辺集合とする．$v \in V$ に対して $L(v)$ は v のラベルである．$|V(G)|$ を頂点の個数とし，$|E(G)|$ を辺の個数とする．また，グラフの集合を $\mathbb{G}$ とする．

本稿では，辺ラベル，自己ループ，多重辺のない無向グラフを扱う．しかし，提案手法は異なる種類のグラフや，頂点間と辺間に類似度を定義する場合にも簡単に拡張可能である．

経路．次に，経路を以下のように定義する．

定義 1 (経路) あるグラフの経路は，頂点とその間に存在する辺を接続する順序で並べた一連の列である．ただし，経路において頂点と辺はそれぞれ連続して並ばず，端は必ず頂点である．経路の頂点数をその経路の長さ n とする．グラフ G が与えられたとき，G に含まれる全ての経路を重複無し集合 $P(G)$ とし，$P_n(G) \subseteq P(G)$ を長さ n の経路とする．また，経路集合から自然数集合への単射 $C : P(G) \to \mathbb{R}$ とし，$P(G)$ に含まれる経路のラベル列 $p \in P(G)$ の個数を $C(p, P(G))$ とする．

長さ n の経路のラベル列 $p \in P_n(G)$ に対し，$i \leqq n$ 番目のラベルを $p[i-1]$ とする．

SED 問合せ．SED 問合せは，入力として与えるクエリグラフとデータベース内のの対象グラフを比較し，クエリグラフと似た対象グラフを抽出する．グラフの比較のため，頂点の対応づけを表すマッピングを定義する．まず，グラフの拡張を定義する．G が与えられた時，ラベル ϵ の頂点を追加した拡張グラフを G^* とする．これは，$\forall v \in V(G)/V(G^*)$ に対し $L(v) = \epsilon$ が成り立つことを意味する．拡張グラフを利用し，マッピングを以下のように定義する．

定義 2 (マッピング) グラフ G_1, G_2 が与えられたとき，G_2 の拡張グラフを G_2^* として，マッピングは単射 $M : V(G_1) \longrightarrow$

$V(G_2^*)$ とする．また，$V(G_1)$ から $V(G_2^*)$ へ写すマッピングの集合を $\mathbb{M}(G_1, G_2)$ とする．

さらに，サブグラフ編集距離を定義するために，サブグラフ同型を以下のように定義する．

定義 3 (サブグラフ同型) グラフ G_1, G_2 が与えられたとき，$G_2' \subseteq G_2$ として，$\forall v \in V(G_1)$ に対し $L(v) = L(f(v))$ が成り立ち，かつ $\forall e(v,u) \in E(G_1)$ に対し $e(f(v), f(u)) \in E(G_2')$ が成り立つ全単射 $f : V(G_1) \to V(G_2')$ が存在するとき，G_1 は G_2 とサブグラフ同型である．

これらに基づき，サブグラフ編集距離 (SED) を以下のように定義する．

定義 4 (サブグラフ編集距離 (SED)) グラフ G_1, G_2 が与えられたとき，G_1 と G_2 のサブグラフ編集距離 $SED(G_1, G_2)$ は，G_1 に編集操作を行なって G_2 のサブグラフ同型にするために必要な編集操作の最少回数である．

ラベル ϵ にマッピングする頂点の編集操作は削除，それ以外のラベルの頂点のうちマッピング先と異なるラベルの頂点の編集操作はラベル変更となる．また，マッピング前に存在する辺がマッピング先では存在しない場合，編集操作は辺の削除となる．通常，編集距離は頂点の追加および辺の追加とラベル変更を含むが， SED において頂点と辺を追加する操作を行うことはない．また，本稿では辺ラベル無しのグラフを扱うため，辺ラベル変更は無視する．上記の定義より，グラフ G_1, G_2 が与えられたとき，$v \in V(G_1), u \in V(G_2)$ として，

$$\delta(v,u) = \begin{cases} 1 & L(v) \neq L(u) \\ 0 & \text{otherwise} \end{cases}$$

とする，また，辺の削除を $v_1, v_2 \in V(G_1), u_1, u_2 \in V(G_2)$ として，

$$\delta(e(v_1,v_2), e(u_1,u_2)) \quad (1)$$
$$= \begin{cases} 1 & e(v_1,v_2) \in E(G_1) \text{ かつ } e(u_1,u_2) \notin E(G_2) \\ 0 & \text{otherwise} \end{cases}$$

とする．このとき，マッピング $M \in \mathbb{M}(G_1, G_2)$ において G_1 が G_2 のサブグラフ同型になるために必要なグラフの編集操作回数を編集距離 $d(G_1, G_2, M)$ とする． $SED(G_1, G_2)$ は G_1 が G_2 のサブグラフ同型になるために必要な最少の編集操作回数であるから，$SED(G_1, G_2)$ は以下である．

$$d(G_1,G_2,M) = \sum_{v \in V(G_1)} \delta(v, M(v)) + \sum_{e(v,u) \in E(G_1)} \delta(e(v,u), e(M(v), M(u))) \quad (2)$$

$$SED(G_1,G_2) = \min_{M \in \mathbb{M}(G_1,G_2)} d(G_1,G_2,M) \quad (3)$$

なお，一般的に $SED(G_1, G_2) \neq SED(G_2, G_1)$ である．SED 計算は NP 困難であることが証明されている [7].

本稿では SED 問合せを以下のように定義する．

問題定義 (SED 問合せ) 対象グラフの集合 $\mathbb{G}$，クエリグラフ Q とする．*Top-k* 検索では k が与えられたとき，$\forall G_1 \in \mathbb{G}'$, $\forall G_2 \in \mathbb{G} \backslash \mathbb{G}'$ に対し $SED(Q, G_1) \leqq SED(Q, G_2)$ を満たす $\mathbb{G}' \subset \mathbb{G}$ を抽出する．範囲検索では，検索範囲 $[r_1, r_2]$ が与えられたとき，$\forall G_1 \in \mathbb{G}'$ に対し $r_1 \leqq SED(G_1) \leqq r_2$ ，かつ $\forall G_2 \in \mathbb{G} \backslash \mathbb{G}'$ に対し $SED(G_2) < r_1$ または $r_2 < SED(G_2)$ を満たす $\mathbb{G}' \subset \mathbb{G}$ を抽出する．

4 経路に基づく下限値と上限値計算

本章では経路に基づいて SED の下限値と上限値の導出方法について述べる．

4.1 経路を利用した下限値の導出

経路を利用した下限値の導出について説明する．提案手法は，長さ 1, 2 のクエリグラフにあるが対象グラフにない経路に基づいて SED の下限値を求める．

グラフ G_1, G_2 について， SED の下限値を $LB(G_1, G_2)$ と表す．また，長さ n 以下の経路の情報から導出するとき，$LB_n(G_1, G_2)$ とする．これは $\forall 0 \leqq i \leqq n$ に対し，$LB_i(G_1, G_2) \leqq LB(G_1, G_2)$ を満たす．

経路を利用した下限値は下記の定理により求める．

定理 1 クエリグラフ G_1 および検索対象 G_2 が与えられたとき，下記が成り立つ．

$$SED(G_1,G_2) \geqq \sum_{p \in P_1(G_1)} \max(0, C(p, P_1(G_1)) - C(p, P_1(G_2))) + \sum_{p \in P_2(G_1)} \max(0, C_3 - C(p, P_2(G_2)))$$

$C_1 = \min(C(p[0], P_1(G_1)), C(p[1], P_1(G_1)))/2$, $C_2 = \min(C(p[0], P_1(G_2)), C(p[1], P_1(G_2)))/2$, $C_3 = \min(\min(C_1, C_2), C(p, P_2(G_1)))$ である．

証明: まず，頂点数 $|V(G_1)| > |V(G_2)|$ であるとき，明らかに $|V(G_1)| - |V(G_2)|$ 個の頂点はラベル ϵ の頂点にマッピングする．同様に，辺数 $|E(G_1)| > |E(G_2)|$ であるとき，明らかに $|E(G_1)| - |E(G_2)|$ 個の辺はラベル ϵ の頂点にマッピングする．すると，式 (2), (3) より，

$$SED(G_1,G_2) \geqq \max(0, |V(G_1)| - |V(G_2)|) + \max(0, |E(G_1)| - |E(G_2)|)$$

である．この右辺を $LB_1(G_1, G_2)$ とする．次に，長さ 1 の経路に注目する．$p \in P_1(G_1)$ はそれぞれ G_1 に含まれるラベルを意味する．このとき，$p \in P_1(G_1)$ に対し，$C(p, P_1(G_1)) > C(p, P_1(G_2))$ ならば，$\sum_{v \in V(G_1)} \delta(v, M(v)) \geqq C(p, P_1(G_1)) - C(p, P_1(G_2))$ である．したがって，式 (2) より，$SED(G_1, G_2) \geqq C(p, P_1(G_1)) - C(p, P_1(G_2))$ である．これは，編集操作にお

いては，頂点の削除とラベル変更の回数の和に対応する．また，$C(p, P_1(G_1)) \leqq C(p, P_1(G_2))$ ならば $\sum_{v \in V(G_1)} \delta(v, M(v)) \geqq 0$ であるので，式 (2) より，$SED(G_1, G_2) \geqq 0$ である．これらのことから，以下が成り立ち，この右辺を $LB_1(G_1, G_2)$ とする．

$$SED(G_1, G_2) \geqq \sum_{p \in P_1(G_1)} \max(0, C(p, P_1(G_1)) - C(p, P_1(G_2)))$$

さらに，長さ 2 の経路に注目する．経路 $p \in P_2(Q)$ について，頂点を $p[0], p[1]$ とする．長さ 2 の経路は 2 つの経路に複数個頂点を持つため，必ず頂点の個数以下である．また，両方の頂点が同じラベルである経路のみであるとき，この経路の個数は最少である．よって，$\frac{C((p[0], P_1(G_1)))}{2} \leqq C((p, P_2(G_1))) \leqq C((p[0], P_1(G_1)))$, $\frac{C((p[1], P_1(G_1)))}{2} \leqq C((p, P_2(G_1))) \leqq C((p[1], P_1(G_1)))$ である．ここで，G_1, G_2 のグラフに含まれる頂点数に基づいて辺を追加すると形成しうる最大の経路数をそれぞれ C_1, C_2 とすると，それぞれ $C_1 = \frac{\min(C(p[0], P_1(G_1)), C(p[1], P_1(G_1)))}{2}$, $C_2 = \frac{\min(C(p[0], P_1(G_2)), C(p[1], P_1(G_2)))}{2}$ である．片方の頂点を長さ 1 の経路とみなすと，これが G の経路にも含まれる場合，この p はもう一方の頂点のと辺を削除するよりも，辺を削除する方が編集操作回数が小さくなるので，実際の G_1 の経路数 $C(p, P_2(G_1))$ を考慮して，辺の削除回数は C_1, C_2 と実際の経路数の差 $\max(0, \min(\min(C_1, C_2), C(p, P_2(G_1))) - C(p, P_2(G_2)))$ である．これらをまとめると，$LB_2(G_1, G_2)$ は以下となる．

$$C_1 = \frac{\min(C(p[0], P_1(G_1)), C(p[1], P_1(G_1)))}{2}$$
$$C_2 = \frac{\min(C(p[0], P_1(G_2)), C(p[1], P_1(G_2)))}{2}$$
$$C_3 = \min(\min(C_1, C_2), C(p, P_2(G_1)))$$
$$LB_2(G_1, G_2) = LB_1(G_1, G_2) + \sum_{p \in P_2(G_1)} \max(0, C_3 - C(p, P_2(G_2)))$$

□

4.2 経路を利用した上限値の導出

経路を利用した下限値の導出について説明する．提案手法は，長さ 1 以上のクエリグラフと対象グラフに共通する経路に基づいて SED の上限値を求める．

グラフ G_1, G_2 が与えられた時，SED の上限値を $UB(G_1, G_2)$ と表す．また，長さ n 以下の経路の情報から導出できるとき，特に $UB_n(G_1, G_2)$ とする．

経路を利用した上限値は下記の定理により求める．

定理 2 クエリグラフ G_1 および検索対象 G_2 が与えられたとき，以下が成り立つ．

$$UB_n^1(G_1, G_2) = |V(G_1)| + |E(G_1)| - \sum_{p \in P_1(G_1)} \min(C(p, P_1(G_1)), C(p, P_1(G_2)))$$
$$UB_n^2(G_1, G_2) = UB_0(G_1, G_2) - \sum_{0 \leqq i \leqq n} \sum_{p \in P_i'(G_1)} (2i - 1)$$
$$SED(G_1, G_2) \leqq \min(UB_n^1(G_1, G_2), UB_n^2(G_1, G_2)) \quad (4)$$

n は G_1 と G_2 の両方に含まれる経路の最大長となる．

証明: まず，経路を利用しないとき，$\sum_{v \in V(G_1)} \delta(v, M(v)) \leqq |V(G_1)|$ かつ $\sum_{e(v,u) \in E(G_1)} \delta(e(v, u), e(M(v), M(u))) \leqq |E(G_1)|$ である．したがって，式 (2) より以下が成り立つ．

$$UB_0(G_1, G_2) = |V(G_1)| + |E(G_1)| \quad (5)$$

次に，長さ 1 の経路に注目する．このとき，$p \in P_1(G_1)$ に対し，ラベル列の共通数は $\min(C(p, P_1(G_1)) > C(p, P_1(G_2)))$ 個である．すると，$\sum_{v \in V(G_1)} \delta(v, M(v)) \geqq |V(G_1)| - \min(C(p, P_1(G_1)) > C(p, P_1(G_2)))$ である．したがって，式 (2) より，$SED(G_1, G_2) \leqq |V(G_1)| + |E(G_1)| - C(p, P_1(G_2))$ である．これは，編集操作においては，削除やラベル変更を行わない頂点数に対応する．これらのことから，以下が成り立つ．

$$UB_1(G_1, G_2) \leqq UB_0(G_1, G_2) - \sum_{p \in P_1(G_1)} \min(C(p, P_1(G_1)), C(p, P_1(G_2)))$$

次に，長さ $n > 1$ の経路に注目する．このとき，$p \in P_n(G_1)$ に対し，$C(p, P_n(G_1)) > 0, C(p, P_n(G_1)) > 0$ を満たす p が存在するとき，$\sum_{v \in V(G_1)} \delta(v, M(v)) \geqq |V(G_1)| - n$ であり，さらに $\sum_{e(v,u) \in E(G_1)} \delta(e(v, u), e(M(v), M(u))) \leqq |E(G_1)| - (n - 1)$ である．これらのことから，以下が成り立つ．

$$UB_n(G_1, G_2) \leqq UB_0(G_1, G_2) - n - (n - 1) = UB_0(G_1, G_2) - (2n + 1)$$

また，同じラベルを持たない経路集合 $P'(G_1) \subset P(G_1)$ について，これらの経路は互いの編集操作に関わらない．よって以下が成り立つ．

$$UB_n(G_1, G_2) = UB_0(G_1, G_2) - \sum_{0 \leqq i \leqq n} \sum_{p \in P_i'(G_1)} (2i + 1)$$

上限値はこれらをいずれも満たすため，より小さい上限値を $UB_n(G_1, G_2)$ とすることができる． □

ここで，長さ $n > 1$ の経路から上限値を導出する際，提案手法では経路の個数は考慮しない．これは，複数個数の経路がループを構成する場合が存在するためである．

4.3 経路抽出

提案手法は対象グラフの経路を保持することで SED 計算に活用する．各対象グラフの n 以下の長さの経路集合を持ち，幅優先探索，または深さ優先探索手順で抽出する．

（1） 長さ 1 の経路を求め，これを以降の経路の始点とする．これは頂点集合に相当する．

（2） 長さ $m > 0$ の経路について，終点の隣接頂点まで経路を延長し，これを新たな終点とする．これにより長さ $m + 1$ の経路を求める．

この経路抽出の時間計算量は以下の定理のようになる．

定理 3 グラフ集合 $\mathbb{G}$ の平均頂点数を $|V|$, 平均隣接頂点

数を $|V_N|$，経路の最大経路長を n とすると，時間計算量は $O(|\mathbb{G}||V||V_N|^{n-1})$ である．

証明: あるグラフ $G \in \mathbb{G}$ について，まず $n=1$ のとき，経路集合は頂点集合となるため，計算量は $O(|V|)$ である．次に，$n>1$ のとき，長さ n の経路は，長さ $n-1$ の経路にさらに隣接頂点を追加することで求められる．よって経路長 n の計算量を O_n とすると，平均隣接頂点数 $|V_N|$ を用いて $O_n = O_{n-1}|V_N|$ である．以上より $n>0$ に対し 1 つのグラフの計算量は $O(|V||V_N|^{n-1})$ であるので，時間計算量は $O(|\mathbb{G}||V||V_N|^{n-1})$ である． □

実グラフは疎なグラフであることが多く，隣接頂点数 $|V_N|$ は $|V|$ よりも非常に小さいのが一般的である．そのため，経路情報の抽出は高速に実行可能である．

5 SED 問合せ

本章では SED 問合せのアルゴリズムについて述べる．まず，5.1 節で SED 問合せフレームワークを述べ，その後，5.2 節で Top-k 検索，範囲検索に適した枝刈りを行う SED 問合せアルゴリズムを述べる．さらに，5.3 節で上限値を更新しながら枝刈りに活用する SED 計算アルゴリズムを述べる．

5.1 問合せフレームワーク

SED 問合せは SED の下限値と上限値を用いながら検索対象とするグラフを枝刈りし，枝刈りできなかったグラフの SED を計算するフィルタリング–検証アルゴリズムに基づいたフレームワークにて実行する．主な流れは以下の通りである．

（1） グラフサイズ昇順に SED の上限値，下限値を求める

（2） SED の上限値と下限値を利用して対象グラフを枝刈りする

（3） SED の上限値の更新とマッピングごとの下限値と比較による枝刈りを行いながら SED を計算する

提案手法は，Top-k 検索と範囲検索とそれぞれの SED 計算を， SED の下限値と上限値を活用して効率化する．

5.2 検索アルゴリズム

Top-k 検索. Top-k 検索では，暫定の k 番目の SED の値を枝刈りに利用し，クエリグラフとの SED が低い上位 k 個の対象グラフを効率的に求める．k 個以上の対象グラフの SED が計算済みのとき，暫定の k 番目の $SED(Q,G)$ を SED_k とすると，$LB(Q,G) > SED_k$ が成り立つならば，この G は明らかに出力に含まれない．したがって，このような対象グラフ G を枝刈りする．さらに，グラフサイズ昇順で SED 計算することで，高速に k 個のグラフの SED を求めることで，この枝刈りの効果を高める．

範囲検索. 範囲検索では，検索範囲と SED の上限値・下限値を比較して枝刈りに利用し， SED が検索範囲内である対象グラフを効率的に求める．まず，$r_1 \leqq LB(Q,G)$ かつ $UB(Q,G) \leqq r_2$ が成り立つ時，$r_1 \leqq SED(Q,G) \leqq r_2$ が成り立つ．したがって，この G は必ず検索結果に含まれるので， SED 計算無しで検索結果に含める．次に，$UB(Q,G) \leqq r_1$ または $r_2 \leqq LB(Q,G)$ が成り立つ時，この G は明らかに検索結果に含まれない．よって，このような対象グラフ G を枝刈りする．

5.3 SED 計算

SED 計算では，マッピングごとの編集距離とそのマッピングを拡張したマッピングの下限をインクリメンタルに求めながら，SED の上限値を更新し，さらに下限と比較による探索する頂点の組み合わせの枝刈りで，効率的に SED 計算を行う．

マッピングごとの編集距離・下限値と SED の上限値の更新. 提案手法は，上限値を更新しながら SED 計算することで，より枝刈りの効果を高める．以下で，マッピングごとの編集距離と下限値の計算と，編集距離を利用した SED の上限値更新について説明する．

まず，マッピング M に加えて頂点 $v \in V(Q)$，$u \in V(G)$ をマッピングした時の下限値 $LB_{cur}(Q,G,M')$ を計算する．$\forall v \in V'(Q) \subset V(Q)$ に対し $\forall M(v) \notin V'(G) \subset V(G)$ が成り立ち，さらに $\forall e(u,u') \in E'(Q) \subset E(Q)$ に対し $\forall e(M(u),M(u')) \notin E'(G) \subset E(G)$ が成り立つとき，式 (2) より，

$$LB_{cur}(Q,G,M') = LB_{cur}(Q,G,M) + |V'(Q)| + |E'(Q)|$$

が成り立つ．次に，グラフ Q と G のマッピング M について，$\forall v \in V'(Q) \subset V(Q)$ に対し $\forall M(v) \in V'(G) \subset V(G)$ が成り立ち，さらに $\forall e(u,u') \in E'(Q) \subset E(Q)$ に対し $\forall e(M(u),M(u')) \in E'(G) \subset E(G)$ が成り立つとき，式 (2) より，

$$d(Q,G,M) = |V(Q)| + |E(Q)| - |V'(Q)| - |E'(Q)|$$

が成り立つ．

さらに，式 (3) より，$SED(Q,G) \leqq d(Q,G,M)$ である．よって，上限値 $UB(Q,G)$ を $\min(UB(Q,G), d(Q,G,M))$ で更新する．

SED 計算の枝刈り. 提案手法は，SED の上限値と下限値を利用し，明らかに SED 計算に不要なマッピングの枝刈り，または問合せ結果に明らかに入るもしくは入らない対象グラフの SED 計算を終了する．

まず，以下の場合は，$M(v) = M'(v), \forall v \in V(G_1')$ を満たす $M \in \mathbb{M}(G_1,G_2)$ の SED 計算を枝刈りする．

枝刈り条件 1: $d(Q_i,G_i,M_i) > UB(Q,G)$ を満たす場合．このとき，明らかに $SED(Q,G) \neq d(Q,G,M)$ である．

枝刈り条件 2: Top-k 検索にて $d(Q,G,M_i) > SED_k$，範囲検索にて $d(Q,G,M_i) > r_2$ を満たす場合．このとき，G 明らかに検索結果に入らない．

さらに，以下の場合は，対象グラフの SED 計算を終了する．

終了条件 1: $M \in \mathbb{M}(Q,G)$ について $d(Q,G,M) = LB(Q,G)$ を満たす場合．このとき，必ず最小の編集距離であるので，$SED(Q,G,M) = LB(Q,G)$ である．

終了条件 2: $LB(Q,G) \geqq r_1$ のとき，$d(Q,G,M) \leqq r_2$ を満たす M が存在する場合．このとき，$r_1 \leqq SED(Q,G) \leqq r_2$ が成り立ち，G は明らかに検索結果に含まれる．

終了条件 3: $LB(Q,G) < r_1$ のとき，$d(Q,G,M) < r_1$ を満たす M が存在する場合．このとき，$SED(Q,G) < r_1$ が成り立ち，G は明らかに検索結果に含まれない．

これらの枝刈りと終了条件により SED 計算を高速に実行することができる．

SED 計算アルゴリズム．クエリグラフ Q が与えられたとき，対象グラフ G の SED を以下の手順で求める．

（1） 暫定のマッピングを M とする．マッピングしていないある頂点 $v \in V(Q)$ に対し，マッピングしていない頂点 $u \in V(G)$ について，頂点 v および u のラベルが一致するかを調べる．一致するならば $C_v^m(v,u) = 1$, $C_v^u(v,u) = 0$ しないならば $C_v^m(v,u) = 0$, $C_v^u(v,u) = 1$ とする．

（2） 頂点 v に対し，クエリグラフの頂点のうちすでにマッピングしている頂点 $\forall v' \in V'(Q)$ s.t. $e(v,v') \in E'(Q) \subset E(Q)$ について，$L(u) \neq \epsilon$ のとき，$e(u, M(v')) \in E(G)$ を満たす辺の個数 $C_e^m(v,u)$，満たさない辺の個数 $C_e^u(v,u)$ を計算する．$L(u) \neq \epsilon$ のとき， $C_e^m(v,u) = 0$, $C_e^u(v,u) = |E'(Q)|$ とする．

（3） u を v のマッピングとした場合の編集距離 $d(Q,G,M') = d(Q,G,M) - C_v^m(v,u) - C_e^m(v,u)$ が上限値 $UB(Q,G)$ より小さいとき，$UB(Q,G) = d(Q,G,M')$ で更新する．

（4） u を v のマッピングとした場合の編集距離 $d(Q,G,M') = d'(Q,G,M) + C_v^m(v,u) + C_e^m(v,u)$ が上限値 $UB(Q,G)$ より小さいとき，$UB(Q,G) = d(Q,G,M')$ で更新する．

疑似コード: アルゴリズム 1 に SED 計算アルゴリズムを示す．擬似コードにおいては，マッピングの編集距離と下限値をそれぞれ (Q,G,M)，$LB_{cur}(Q,G,M)$ とし，それぞれインクリメンタルに求める．3 行目に示すように，最初に与える uu, vv は Null とする．また， SED の下限値を $LB(Q,G)$，上限値を $UB(Q,G)$ は経路を用いて算出した値を与える． 6 行目で頂点 uu を vv のマッピングとする．9–26 行目で，マッピングの編集距離と下限値を求め，SED の上限値を更新しながら，枝刈りする．$isCandidate(Q,G,M')$ は Top-k 検索と範囲検索それぞれの枝刈り判定を行う．11, 12 行目で，SED の下限と一致する編集距離を見つけた場合にその値を SED の値として返す．13–19 行目で，マッピングの編集距離を計算し，上限値を更新した後，枝刈りを行うか判定する．20 行目でより小さい編集距離で SED の値を更新し，21 行目で，再帰的にこの関数を呼び，次の頂点のマッピングを行う．22–26 行目で，u がラベル ϵ の頂点とマッピングする場合，すなわち対象グラフの頂点とマッピングしない場合に，$LB_{cur}(Q,G,M)$ の更新，マッチ処理と，枝刈り判定を行う．

Algorithm 1: SED 計算アルゴリズム

```
   input : Query graph Q, a target graphs G, lower bound LB(Q,G),
           upper bound UB(Q,G)
   output: SED(Q,G)
1  LB_cur ⟵ 0;
2  d ⟵ |V(Q)| + |E(Q)|;
3  Return CalcSED(Null, Null, LB(Q,G), UB(Q,G), LB_cur, d);
   input : current vertex uu ∈ V(Q), vv ∈ V(G), lower bound
           LB(Q,G), upper bound UB(Q,G), LB_cur, d
   output: SED(Q,G)
4  procedure CalcSED
5  if uu is Null then
6      M(uu) ↩ vv
7  V'(Q) ⟵ {u ∈ V(V)|M(u) ≠ Null};
8  V'(G) ⟵ {v ∈ V(G)|M(u) ≠ v ∀u ∈ V(Q)};
9  foreach u ∈ V'(Q) do
10     foreach v ∈ V(G) s.t. v ∉ M do
11         if d(Q,G,M) = LB(Q,G) then
12             return;
13         Calculating C_v^m(v,u), C_v^u(v,u), C_e^m(v,u), C_e^u(v,u);
14         LB_cur ← LB_cur + C_v^u(v,u) + C_e^u(v,u);
15         d ← d − C_v^m(v,u) − C_e^m(v,u));
16         UB(Q,G) ← min(UB(Q,G), d);
17         if LB_cur > min(SED(Q,G), UB(Q,G)) then
18             continue;
19         if isCandidate(Q,G,M') then
20             SED(Q,G) ← min(SED(Q,G), d);
21             CalcSED(u, v, LB(Q,G), UB(Q,G), LB_cur, d);
22     Calculating C_v^u(v,u), C_e^u(v,u);
23     LB_cur ← LB_cur + C_v^u(v,u) + C_e^u(v,u);
24     Matched(u, ϵ);
25     if LB_cur > min(SED(Q,G), UB(Q,G)) then
26         break;
27 Return SED(Q,G);
```

6 実　　験

本章では，提案手法の有用性の確認のため，Top-k 検索と範囲検索それぞれにおける検索効率，精度を評価する．実験は AMD EPYC 7542 32-Core Processor， 2TB メモリを搭載した Ubuntu 18.04.6 LTS サーバ上で実行し，全てのアルゴリズムは C++ を利用して実装・実行した．

6.1 実 験 設 定

データセット．データセットは， 分子構造データの AIDS [15], BZR [16] を利用する．クエリはデータセットの対象グラフのサブグラフから，頂点数 5 の条件下でランダムに生成した．表 1 にデータセットの詳細を示す．

比較手法．比較手法として，SED 問合せの厳密解および近似解をを計算する手法を用いる．厳密解を計算する手法として，素朴な手法 Naive (exact), Ctree (exact) [2], さらに，4.3 で述べた経路索引を利用し，5 節で述べた提案 SED 計算アルゴリズムを適用する提案手法 Ours (exact) を用いる．Naive (exact) は SED 計算には $C_v^m(v,u)$ のみ考慮する簡単な枝刈りを適用し，全ての対象グラフの SED を計算する手法である．

近似解を計算する手法では，Ctree で使用される近似 SED 計算手法を利用する． 近似 SED 計算は，クエリの頂点に対し，対象グラフのまだマッピングしていない頂点の中からラベルが一致する頂点を選び，マッピングする手法である．それぞれ Naive (approx), Ctree (approx), Ours (approx) と表記する．

Ours では最大経路長 n を 5 とする．また，Ctree は Top-k 検索に特化しているため，範囲検索の実験では，検索範囲の上界よりも索引利用で導出した下限値が小さいときに枝刈りする技術を適用した．

評価方法．それぞれの比較手法で Top-k 検索と範囲検索の検索時間と精度を評価する．Top-k 検索においては，$k = 10$ とし，範囲検索においては，対象グラフ数に対する正解の出力個数の割合が 10% となる検索範囲とする．検索精度では，Top-k 検索

表 1: データセット

名前		グラフ数	平均頂点数	平均辺数	合計ラベル種類数
AIDS	対象グラフ	1811	14.02	14.50	27
	クエリグラフ	50	5.00	4.18	7
BZR	対象グラフ	405	35.75	38.36	10
	クエリグラフ	50	5.00	4.00	6

は nDCG@10 [17] と RMSE, 範囲検索では Recall, Precision, F1 の平均値で評価する．nDCG@10，Recall, Precision, および F1 の値の範囲は $[0, 1]$ であり， 大きい値が高精度であることを意味する．RMSE は小さい値が高精度であることを示す．

6.2 実験結果

提案手法の検索時間と精度および経路抽出の時間とサイズの評価実験結果を示す．

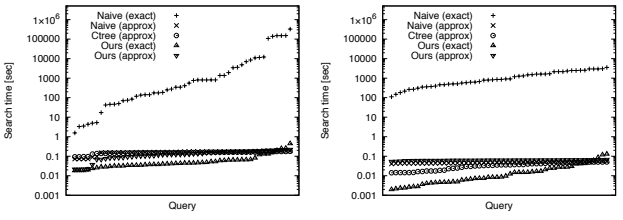
6.2.1 検索時間

表 2 に Top-10 検索と範囲検索の検索時間を示す．結果より，Ours (exact) は Top-10 検索では比較手法の中で最も高速であり，特に近似手法よりも高速である．これは，SED 計算そのものは近似手法のほうが高速であるものの，提案手法は正確な SED 計算により早期に $SED_{10} = 0$ となることによって問合せが終了し，5 節で述べた枝刈りによって SED 計算する対象グラフを大幅に枝刈りしているためである．一方で，Ours (approx) は正確に SED 計算できず，$SED_{10} = 0$ による問合せ終了のために多くの SED 計算が必要となる．これにより，AIDS, BZR では Ours (exact) は Ours (approx) よりも高速である．また，範囲検索においても Ours (exact) は Naive (exact) に対し大幅に検索時間を削減し，近似手法の Naive (approx), Ctree (approx), Ours (approx) と同等に高速な検索時間となった．これは，提案手法は経路の比較による下限値と上限値計算が高速であり，さらに下限値と上限値による SED 計算手法が効果的に枝刈りを行うためである．Ctree (exact) は Naive (exact) よりも時間を要し，実行が終了しない．これは，グラフ索引を近似なしで計算すると，計算コストが大きいためである．したがって，Ctree が利用するグラフ索引は，正確な SED 計算に不適である．一方で，提案手法は経路から高速に上限値と下限値を計算できるため，正確な SED 計算を高速に計算可能である．

図 2, 3 にそれぞれ Top-10 検索，範囲検索の詳細を示す．それぞれのプロットは，クエリの検索時間を昇順に並べた検索時間を示す．左端が最も検索時間が短いクエリの検索時間，右端が最も検索時間が長いクエリの検索時間を表す．AIDS, BZR について，Ours (exact) は Ctree (approx) と比べて，同順位のクエリにおいてそれぞれ最大 5.5 倍，7.2 倍高速な検索時間となった．これは，上述のように，提案手法はグラフサイズ昇順による計算で高速に SED を 10 個以上計算し，$SED_{10} = 0$ による問合せ終了によって多くの対象グラフを枝刈りするためである．また，範囲検索では，Ours (exact) は Naive (exact) に比べて大幅に検索時間を削減し，いずれのクエリにおいても実践的な時間で実行可能である．これは，上限値を更新しながら効果的に枝刈りする SED 計算によって，SED 計算回数が多い場合でも効率的であるためである．加えて，一部のクエリでは近似手法よりも高速である．これは 5 節で述べた検索範囲に特化した枝刈りで多くの対象グラフを枝刈りしたためである．

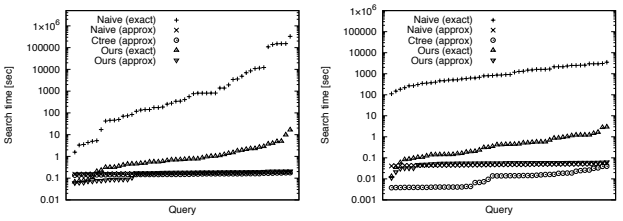
表 2: 平均検索時間．15 日以内に 50 クエリが終了しなかった場合 DNF とする．

手法		Top-10 検索		範囲検索	
		AIDS	BZR	AIDS	BZR
exact	Naive	22027.67	1239.92	22027.67	1239.92
	Ctree	DNF	DNF	DNF	DNF
	Ours	0.07	0.02	1.63	0.58
approx	Naive	0.16	0.05	0.16	0.05
	Ctree	0.15	0.03	0.15	0.01
	Ours	0.13	0.06	0.16	0.05



(a) AIDS (b) BZR

図 2: Top-10 検索時間



(a) AIDS (b) BZR

図 3: 範囲検索時間

6.2.2 検索精度

表 3 は Top-10 検索における近似手法の検索精度を示す．exact の手法は正確な SED を算出するため， nDCG@10 は 1, RMSE は 0 となる．結果より， いずれの近似手法も同程度であり，Naive はわずかに nDCG@10 が小さく，さらに Ctree は Naive, Ours と比べて両方のデータセットでわずかに RMSE が大きい．これは，近似計算では正しい SED を求められず，上位から同立の SED となった対象グラフが偶然上位 10 番目以内に入る/入らないによって，出力結果が異なったためである．さらに，BZR データセットでは， AIDS データセットと比べていずれの手法も nDCG@10 は高く Top-k の精度は高いが，一方で RMSE も大きく正確な SED の値が計算できていない．BZR データセットの方が正解の SED が 0 であるものが多く，SED の値が実際と大きく異なっても BZR では上位 k 個を正解で埋めることができるためである．

表 4 に範囲検索における近似手法の検索精度を示す．結果より，Ours は Naive と同等に正確である．一方で Ctree は BZR にてわずかに精度が低い．これは，近似計算による下限値と検索範囲を比較するために，検索範囲内の SED となる対象グラフを誤って枝刈りするためである．これらの結果は，提案手法は索引利用によって精度を低下させず，正確な SED を計算で

表 3: 近似手法の Top-10 検索精度

手法	AIDS		BZR	
	nDCG@10	RMSE	nDCG@10	RMSE
Naive	0.81	0.25	1.00	1.47
Ctree	0.83	0.31	1.00	1.85
Ours	0.83	0.25	1.00	1.47

表 4: 近似手法の範囲検索精度

手法	AIDS			BZR		
	Recall	Precision	F1	Recall	Precision	F1
Naive	0.33	0.45	0.14	0.04	0.02	0.02
Ctree	0.33	0.45	0.14	0.03	0.01	0.01
Ours	0.33	0.45	0.14	0.04	0.02	0.02

表 5: 構築時間

	構築時間 [秒]		索引サイズ [KB]	
手法	AIDS	BZR	AIDS	BZR
Ctree	27.3	17.1	231.6	178.6
Ours	7.1	4.8	3165.3	1573.5

きることを示している．さらに，6.2 節より，提案手法は高速かつ高精度に実践的な SED 問合せ手法である．

6.2.3 構 築 時 間

最後に，提案手法の経路抽出と Ctree の索引構築を計算時間とサイズの観点から比較する．表 5 に Ours の経路抽出時間と経路情報のサイズと Ctree の索引構築時間と索引のサイズを示す．結果より，構築について Ours は Ctree の 3 倍程度高速である．これは，グラフに基づく索引よりも経路抽出は計算量が小さく，さらに Ctree は木構造を構築するためである．また，サイズは，経路を保持する Ours はグラフを索引する Ctree より大きい．これは，Ours では構造体で経路を表現しており，多くのポインタと構造体を使っている一方で，Ctree は中間の頂点に保持するグラフをシンプルな配列を利用して保持するためである．しかし，最大でも 3MB 程度であり，これは小規模な計算機でも十分に機能する．したがって，提案手法の経路情報は，高速かつ現実的な空間コストで構築可能である．

7 おわりに

本稿では，グラフの経路構築と検索および SED 計算アルゴリズムにおける枝刈りによる，効率的で高精度な SED 問合せを提案した．本手法では，対象グラフの SED の上限値と下限値を計算することで，計算対象を枝刈りし，さらにグラフの SED 計算そのものも高速化した．実験では，提案手法は高速かつ正確に SED 問合せであり，有用であることを示した．

今後の展望としては，経路以外の索引による上限値と下限値の導出と，それらを利用する検索と SED 計算のアルゴリズムのさらなる改良で，より効率的な検索を目指す．また，提案手法では Top-k 検索においてグラフサイズを考慮した計算順であるが，対象グラフ数などの特徴に応じた計算順により，より効率的な検索を目指す．

謝辞 本研究は JST さきがけ JPMJPR18UD および JSPS 科研費 JP20H00583 の支援によって行われた．ここに記して謝意を表す．

文 献

[1] Diane J. Cook and eds Lawrence B. Holder. Mining graph data. 2006.

[2] Huahai He and Ambuj K Singh. Closure-tree: An index structure for graph queries. In *Proceedings of the ICDE*, pp. 38–38, 2006.

[3] Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. Greed: A neural framework for learning graph distance functions. In *Advances in Neural Information Processing Systems*, 2022.

[4] Bruno T Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *Proceedings of the TPAMI*, Vol. 20, No. 5, pp. 493–504, 1998.

[5] Aravind Sankar, Sayan Ranu, and Karthik Raman. Predicting novel metabolic pathways through subgraph mining. *Bioinformatics*, Vol. 33, No. 24, pp. 3955–3963, 2017.

[6] Arlei Silva, Wagner Meira Jr, and Mohammed J Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *arXiv preprint arXiv:1201.6568*, 2012.

[7] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *VLDB Endowment*, Vol. 2, No. 1, pp. 25–36, 2009.

[8] Xiang Zhao, Chuan Xiao, Xuemin Lin, Wei Wang, and Yoshiharu Ishikawa. Efficient processing of graph similarity queries with edit distance constraints. *The VLDB Journal*, Vol. 22, No. 6, pp. 727–752, 2013.

[9] Haichuan Shang, Xuemin Lin, Ying Zhang, Jeffrey Xu Yu, and Wei Wang. Connected substructure similarity search. In *Proceedings of the ACM SIGMOD*, pp. 903–914, 2010.

[10] Guoren Wang, Bin Wang, Xiaochun Yang, and Ge Yu. Efficiently indexing large sparse graphs for similarity search. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 24, No. 3, pp. 440–451, 2010.

[11] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the ACM WSDM*, pp. 384–392, 2019.

[12] Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, Vol. 24, No. 1, p. 90, 1982.

[13] James Cheng, Yiping Ke, Wilfred Ng, and An Lu. Fg-index: towards verification-free query processing on graph databases. In *Proceedings of the ACM SIGMOD*, pp. 857–872, 2007.

[14] Shijie Zhang, Shirong Li, and Jiong Yang. Gaddi: distance index based subgraph matching in biological networks. In *Proceedings of the EDBT*, pp. 192–203, 2009.

[15] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Proceedings of the SPR and the SSPR*, Vol. 5342, pp. 287–297, 2008.

[16] Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *Journal of chemical information and computer sciences*, Vol. 43, No. 6, pp. 1906–1915, 2003.

[17] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the ICML*, pp. 89–96, 2005.