Vol.2020-DBS-172 No.11
2020/12/22

グラフ編集距離を用いた類似包摂グラフ検索

山田 真賢[1,a)]　猪口 明博[1]

概要：検索技術の 1 つであるグラフ検索のうち本稿では包摂グラフ検索について議論する．近年，創薬化学などの分野で構造が類似した化合物を検索する需要が高まっている．しかし，既存の包摂グラフ検索は類似包摂グラフを検索することができない．本論文では，クエリグラフの部分グラフからグラフ編集距離が閾値以内にあるグラフを検索する類似包摂グラフ検索問題とそのための手法を提案する．また，評価実験において，検索されたグラフ数や計算時間の特徴を述べる．

Similar Supergraph Search based on Graph Edit Distance

1. はじめに

コロナウイルスの感染が広がっている．その感染症の有望な抗ウイルス剤として，ファビピラビル（医薬品名：アビガン）が検討された．ファビピラビルは元々抗インフルエンザウイルス剤として開発されたものである．ファビピラビルがリボース化されると，その構造はアカデシンに似る．アカデシンは RNA の材料となるグアノシンやアデノシンの前段階のイノシンに至る前駆物質である [10]．そのため，ウイルスの RNA 依存性 RNA 合成酵素は，伸長中のウイルス RNA に，ファビピラビルをグアノシンなどと間違えて取り込んでしまい，そこで，RNA 合成が停止する．これにより，体内でのウイルスの増殖を抑制することができる．図 1 はグアノシン，イノシン，アカデシン，リボース化されたファビピラビルの化学構造である．赤で描かれた部分構造 a は一致しているので，新薬のシード化合物探索の際には，多数の化合物を含むデータベースに対して，部分構造 a をクエリとして検索する方法（部分構造検索）が考えられる．一方，新薬開発に重要な部分構造が事前に多数既知であれば，このような部分構造をデータベースに蓄積しておき，ある新規化合物が開発された際には，それをクエリとして後者のデータベースを検索すること（包摂構造検索）が考えられる．本稿では，後者に関連する手法について論ずる．図 1 の青で描かれた部分構造は，部分構造 a に類似しているが，完全には一致していない．抗インフルエンザ剤に寄与する部分構造 a がデータベースに登録されている状況下で，リボース化されたファビピラビルをクエリとして与え，部分構造 a が検索結果として見つかれば有用であると考えられる．

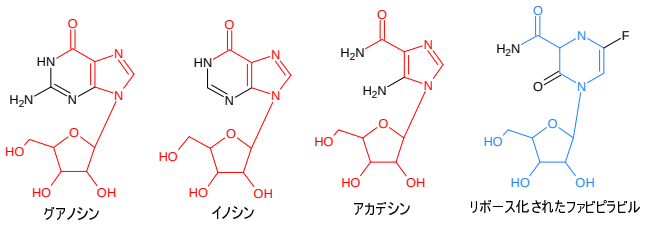

図 1　類似した化合物

化合物は，原子，化学結合をそれぞれグラフの頂点と辺に対応させると，グラフで表現することができる．上記の部分構造検索や包摂構造検索問題は，それぞれ部分グラフ検索 [11] [4] [13] [12] や包摂グラフ検索問題 [8] [5] [2] [3] [14] に対応し，近年，盛んに研究が行われている．グラフは，人間関係ネットワーク，タンパク質相互作用ネットワーク [1]，RDF (Resource Descrip-tion Framework)，コンピュータ支援設計（CAD：computer-aided design），Computer Vision など実社会の対象を表現する汎用性の高いデータ構造であり，部分/包摂グラフ検索は，化合物への応用に関わらず，広い適用範囲をもつ．しかし，類似するグラフを検索する手法は，文献のみに限られている．しかも，文献の方法では，図 1 で示した部分構造の類似性を扱うことはできない．本研究では，類似包摂グラフ検索問題を新たに提案し，それを解くための手法を提案する．

[1] 関西学院大学大学院　理工学研究科　情報科学専攻
Department of Informatics, Graduate School and Technology, Kwansei Gakuin University, 2–1, Gakuen, Sanda, 669–1337, Japan
[a)] eyd13231@kwansei.ac.jp

2. 問題定義

ラベル付きグラフ $g = (V, E, \Sigma, \ell)$ は，頂点の集合 V，頂点対を結ぶ辺の集合 $E \subseteq V \times V$，頂点と辺のラベルの集合 Σ，頂点や辺にラベルを割り付ける関数 $\ell : V \cup E \to \Sigma$ で表される．グラフ g の辺 $(v_1, v_2) \in E$ が向きを持つ順序対であるとき g を有向グラフと呼ぶ．一方，グラフ g の辺 $(v_1, v_2) \in E$ が向きを持たないとき g を無向グラフと呼ぶ．また，頂点 v_1 から v_2 に至る辺の系列をパスと呼び，辺の向きを除いたグラフ中の任意の頂点間にパスが存在するグラフを連結グラフという．

グラフ $g = (V, E, \Sigma, \ell)$ と $g' = (V', E', \Sigma', \ell')$ が与えられ，$\forall v, v_1, v_2 \in V'$ に対して，単射 $\phi : V' \to V$ が以下を満たすとき，g' を g の部分グラフと呼び，$g' \subseteq g$ と表す．

- $(\phi(v_1), \phi(v_2)) \in E$ if $(v_1, v_2) \in E'$
- $\ell'(v) = \ell(\phi(v))$
- $\ell'((v_1, v_2)) = \ell((\phi(v_1), \phi(v_2)))$

さらに，$(\phi(v_1), \phi(v_2)) \in E$ iff $(v_1, v_2) \in E'$ であるとき，g' を g の誘導部分グラフと呼び，$g' \subseteq_i g$ と表す．本稿では，連結な無向グラフを対象とするが，本稿が扱う手法は，非連結グラフ，あるいは有向グラフにも拡張可能である．

グラフ編集距離は，グラフ g を g' に変換するときの最小の編集操作の回数と定義される．ここでの編集操作とは，頂点や辺の追加，削除，ラベルの変更を指す．本稿では，上記の編集コストをすべて 1 とする．グラフ $g = (V, E, \Sigma, \ell)$ と $g' = (V', E', \Sigma, \ell')$ の間の編集操作数 [9] は

$$dist(g, g', \varphi) = \sum_{i=1}^{|V'|} c_{i\varphi(i)} + \sum_{i=1}^{|V'|-1} \sum_{j=i+1}^{|V'|} c\left((v_i, v_j) \to \left(v_{\varphi(i)}, v_{\varphi(j)}\right)\right) \quad (1)$$

と定義できる．ここで，$|V| \leq |V'|$ とし，V は $V^+ = V \cup \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{|V'|-|V|}\}$ と拡張される．また，g が g' に編集されるとき，2 つの頂点集合の頂点の対応は全単射 $\varphi : V^+ \to V'$ で表される．さらに，c_{ij} は $v_i \in V$ を $v_j \in V'$ に置き換える（挿入，削除，ラベル変更のいずれか）コストを表しており，g の辺 (v_i, v_j) を g' の辺 $(v_{i'}, v_{j'})$ に編集するコストを $c((v_i, v_j) \to (v_{i'}, v_{j'}))$ と表す．以上よりグラフ編集距離 $ed(g, g')$ は

$$d(g, g') = \min_{\varphi \in \Phi(|V'|, |V'|)} dist(g, g', \varphi) \quad (2)$$

と定義される．ここで，$\Phi(\alpha, \beta)$ は整数 $1, 2, \ldots, \alpha$ から β 個を選び，可能なすべての順列の集合である．式 (2) は二次割り当て問題の一つであり，NP 完全である．

本研究で扱う問題はグラフの集合 $G = \{g_1, g_2, \ldots, g_n\}$，クエリグラフ q，閾値 θ を入力とし，

$$\{g_i \in G \mid \exists q' \subseteq q \; s.t. \; ed(g_i, q') \leq \theta\} \quad (3)$$

を出力する問題である．この問題において $\theta = 0$ のとき，包摂グラフ検索問題と同じであるので，式 (3) で表される問題は包摂グラフ検索問題の拡張であるといえる．

3. 解を制限した場合の類似グラフ検索法

前節で本稿が扱う問題を定義した．本節では，説明の簡単化のため，まず以下を出力する手法を提案する．

$$\{g_i \in G \mid \exists q' \subseteq_i q \; s.t. \; ed(g_i, q') \leq \theta, q' \text{は連結}\} \quad (4)$$

そして，次節において，その手法が元の問題 (3) を解けるように容易に拡張できることを説明する．

定義 3.1 グラフ $g = (V, E, \Sigma, \ell)$ が与えられたとき，各頂点に 1 から $|V|$ の ID を割り当て，各頂点を $v_1, v_2, \ldots, v_{|V|}$ とする．ただし，$v_1, v_2, \ldots, v_i$ $(1 \leq i \leq |V|)$ によって誘導される g の誘導部分グラフは連結であるとする．g は $|V| \times |V|$ の隣接行列で表現できる．その (i, j)-要素 $x_{i,j}$ は $(v_i, v_j) \in E$ ならば $\ell((v_i, v_j))$ であり，それ以外で 0 となる．グラフ g の AcGM コード [6] は，グラフ g の隣接行列の上三角部分の要素を並べて，

$$code(g, \langle v_1, v_2, \ldots, v_{|V|} \rangle) = s_1 s_2 \cdots s_{|V|} \quad (5)$$

と定義される．ただし，コードフラグメント s_i は以下とする．

$$s_i = \ell(v_i) x_{1,i} x_{2,i} \cdots x_{i-2,i} x_{i-1,i} \blacksquare$$

$s_1 s_2 \cdots s_i$ を式 (5) の接頭辞と呼ぶ．また，AcGM コード c_1 の接頭辞と AcGM コード c_2 が一致するとき，$c_2 \subseteq c_1$ と書き，ある AcGM コード c が表すグラフを $g(c)$ と書く．

補題 3.2 2 つの AcGM コードが一致するならば，それらのコードが表すグラフは同型である．■

ただし，2 つのグラフが同型だからといって，それらのコードが常に一致するとは限らない．つまり，グラフ g に対して，定義 5 を満たす AcGM コードは複数存在する．そこで，g の AcGM コードの集合を $\Omega(g)$ と書く．

補題 3.3 AcGM コード c_1 が c' の接頭辞であるならば，$g(c_1)$ は連結であり，$g(c')$ の誘導部分グラフである．■

以上より式 (4) で定式化した問題を解くためには，$c \in \Omega(q)$ の接頭辞 $c' \subseteq c$ が表すグラフ $g(c')$ と $g_i \in G$ とのグラフ編集距離が計算できればよい．しかし，$g(c')$ と g_i のグラフ編集距離を式 (2) で解くことは非効率であるので，以下ではコードの間の編集操作数を求めることを考える．

補題 3.4 以下に示す AcGM コードが与えられたとする．

$$c = \ell(v_1)\ell(v_2)x_{1,2} \cdots \ell(v_a)x_{1,a} \cdots x_{a-1,a} \quad (6)$$

$$c' = \ell(v'_1)\ell(v'_2)x'_{1,2} \cdots \ell(v'_b)x'_{1,b} \cdots x'_{b-1,b} \quad (7)$$

$g(c)$ の i 番目の頂点が $g(c')$ の i 番目の頂点に対応するとき，$g(c)$ を $g(c')$ を一致させるための編集操作数 $ed(c, c')$ は

Algorithm 1: Naive Search Algorithm for Problem (4)

Data: $G = \{g_1, g_2, \ldots, g_n\}$, q, θ
Result: $\{g_i \in G \mid \exists q' \subseteq_i q \; s.t. \; ed(g_i, q') \leq \theta, q'$は連結 $\}$

```
1  S ← ∅;
2  for g_i ∈ G do
3      c_i ← code(g_i)//任意のコード;
4      for c ∈ Ω(q) do
5          for j ∈ [1, |V(q)|] do
6              c' ← pre(c, j);
7              if ed(c_i, c') ≤ θ then
8                  S ← S ∪ {g_i};
9                  break;
10         if g_i ∈ S then
11             break;
12 return S;
```

$$
\begin{aligned}
&\sum_{1\leq i\leq \min\{a,b\}} \delta'(\ell(v_i), \ell(v'_i)) \\
&+ \sum_{1<j\leq \min\{a,b\}} \sum_{1\leq i<j} \delta'(x_{i,j}, x'_{i,j}) + |a-b| \\
&+ \begin{cases} \sum_{b<j\leq a} \sum_{1\leq i<j} \delta'(x_{i,j}, 0) & (b < a) \\ 0 & (a = b) \\ \sum_{a<j\leq b} \sum_{1\leq i<j} \delta'(0, x'_{i,j}) & (a < b) \end{cases} \qquad (8)
\end{aligned}
$$

となる．ここで，δ はクロネッカーのデルタ，$\delta'(\alpha, \beta) = 1 - \delta(\alpha, \beta)$ である．式 (8) の計算量は $O(\max\{a, b\}^2)$ となる．$ed(c, c')$ は対称関数 $ed(c, c') = ed(c', c)$ である．■

式 (8) の第 1 項目は $g(c)$ の i 番目（ただし，$1 \leq i \leq \min\{a, b\}$）の頂点を $g(c')$ の i 番目の頂点に対応させたときの頂点ラベルの不一致による編集操作数の総和であり，第 2 項目は $g(c)$ の辺 (i, j)（ただし，$1 \leq i < j \leq \min\{a, b\}$）を $g(c')$ の辺 (i, j) に対応させたときの編集操作数の総和である．また，第 3 項目は $g(c)$ への頂点追加操作数の総和であり，第 4 項目は $g(c)$ への辺追加操作数の総和である．式 (8) は，2 つのグラフの頂点の対応関係が既知の下で，2 つのグラフの間の編集操作数を導出する計算式であるので，式 (1) に相当する．以上より，式 (2) は，式 (8) を用いて以下のように表すことができる．

補題 3.5 グラフ g の AcGM コードの集合を $\Omega(g)$ とし，g' のある AcGM コードを c' とするとき，以下が成り立つ．

$$ed(g, g') = \min_{c\in\Omega(g)} ed(c, c') \blacksquare \qquad (9)$$

この補題より，以下の系が成り立つ．

系 3.6 AcGM コード c と c' に対して $ed(c, c') \leq \theta$ が成り立つならば $ed(g(c), g(c')) \leq \theta$ が成り立つ．■

Algorithm 1 に問題 (4) を解くための単純なアルゴリズムを示す．3 行目でデータベースの各グラフ g_i に対して，任意の 1 つの AcGM コード c_i を得る．q から得られるすべての AcGM コードのうち 1 つを 4 行目で得て，c_i の j 番目までのコードフラグメントからなる接頭辞 c' を 6 行目で得る．c' は c の接頭辞であるので，$g(c')$ は q の連結な誘導部分グラフに限られる．続いて，系 3.6 に基づいて $ed(c_i, c') \leq \theta$ ならば，g_i は問題 (4) の解であるので，g_i を S に追加する．Algorithm 1 の計算量は $O(|G||\Omega(q)||V(q)|^3)$ となる． しかし，この Algorithm 1 には以下で述べる課題がある．

(1) $\Omega(q)$ の各 AcGM コード c に対して，6 行目で $|V(q)|$ 個の接頭辞を生成する．これらの接頭辞に対して，7 行目において式 (8) を $|V(q)|$ 回計算するが，c から生成される 2 つの接頭辞 c' と c'' は，$c' \subset c''$ となるので，式 (8) の繰り返し計算が無駄である．$ed(c_i, c')$ の結果を $ed(c_i, c'')$ の計算に用いることができれば，計算を高速化できる．

(2) 3 行目において $G_c = \{code(g_i) \mid g_i \in G\}$ を求めるが，G_c の中には，接頭辞が等しい AcGM コード $code(g_i)$ と $code(g_j)$ が含まれている．それらに対する式 (8) の計算に無駄がある．その接頭辞を c_s とすると，$ed(c_s, c)$ の結果を $ed(code(g_i), c)$ と $ed(code(g_j), c)$ の計算に用いることができれば，計算を高速化できる．

4. 解を制限した場合の類似グラフ検索法 2

上記の課題 (1) を解決するために，次の補題を導出する．

補題 4.1 任意のコード c と c' が与えられたとき，$i < j$ に対して

$$ed(pre(c, i), pre(c', i)) \leq ed(pre(c, j), pre(c', j)) \qquad (10)$$

が成り立つ．ただし，c がもつフラグメント数よりも i が大きいとき，$pre(c, i)$ は c そのものを返すとする．■

系 4.2 補題 3.1 より $ed(pre(c, i), pre(c', i)) > \theta$ が成り立てば，$ed(c, c') > \theta$ が成り立つ．■

したがって，Algorithm 1 の 7 行目における式 (8) の計算を打ち切ることができる．

補題 4.3 以下に示す長さが等しいコードフラグメントが与えられたとする．

$$s = \ell(v_i)x_{1,i} \cdots x_{i-1,i} \qquad (11)$$

$$s' = \ell(v'_i)x'_{1,i} \cdots x'_{i-1,i} \qquad (12)$$

これらのフラグメントを一致させるための編集操作数 $ed(s, s')$ は

$$\delta'(\ell(v_i), \ell(v'_i)) + \sum_{1\leq j<i} \delta'(x_{j,i}, x'_{j,i})$$

となる．s と s' のいずれかが空のとき，

$$ed(s, s') = \begin{cases} 1 + \sum_{1\leq j<i} \delta'(0, x'_{j,i}) & (s = null) \\ 1 + \sum_{1\leq j<i} \delta'(x_{j,i}, 0) & (s' = null). \end{cases} \qquad (13)$$

Algorithm 2: Naive Search Algorithm 2 for Problem (4)

Data: $G = \{g_1, g_2, \ldots, g_n\}$, q, θ
Result: $\{g_i \in G \mid \exists q' \subseteq_i q \; s.t. \; ed(g_i, q') \leq \theta, q'$は連結 $\}$

```
1  S ← ∅;
2  for g_i ∈ G do
3      c_i ← code(g_i);
4      for c ∈ Ω(q) do
5          ae ← 0; //ae は累積編集操作数
6          for j ∈ [1, |V(q)|] do
7              ae ← ae + ed(frag(c_i, j), frag(c, j));
8              if ae > θ then
9                  break;
10             if j = |V(q)| then
11                 S ← S ∪ {g_i};
12         if g_i ∈ S then
13             break;
14 return S;
```

$ed(s, s')$ は対称関数 $ed(s, s') = ed(s', s)$ である． ■

Algorithm 2 に課題 (1) を解決したアルゴリズムを示す．7 行目では, AcGM コード c の j 番目のフラグメント $frag(c, j)$ を取り出し，2 つのフラグメント間の編集操作数を補題 4.3 により得て，それを累積変数操作数 ac に加算する．8 行目では系 3.2 に基づいて c と c_i の関係から g_i が解となるとは言えないので，q の異なるコードに対して，同じ計算を繰り返す．Algorithm 2 の計算量は $O(|G||\Omega(q)||V(q)|^2)$ となる．

続いて，前述の 2 つ目の課題を解決するために，G の AcGM コードの接頭辞木を用いる．その接頭辞木を本稿ではコード木と呼び，文献 [5] で提案されているものを用いる．

定義 4.4 コード木 T は 3 つ組 $(\top, N, B)$ で定義される．ここで，$\top \in N$ は木の根，N は木の節点集合，B は木の枝の集合である．加えて，木の各節点はコードフラグメントとグラフ ID の集合をもつ．根からある節点 n に至るまでのパス上の節点がもつコードフラグメントを $s(n)$ とし，$g_i \in G$ のあるコードが $s(n)$ と等しいならば，節点 n のグラフ ID 集合は i を含むものとする．■

コード木の節点 n がもつコードフラグメントとグラフ ID の集合をそれぞれ $fr(n)$ と $ID(n)$ で表す．木の根に対しては $fr(\top) = null$，$ID(\top) = \emptyset$ となる．コード木の例を以下で示す．

例 4.5 グラフデータベース $G = \{g_1, g_2, g_3, g_4\}$ が図 2 で示される 4 つのグラフで構成されるとする．各グラフから様々なコードを生成できるが，各グラフから 1 つの AcGM コードを生成したとすると，4 つのグラフのコードは以下のようになる．

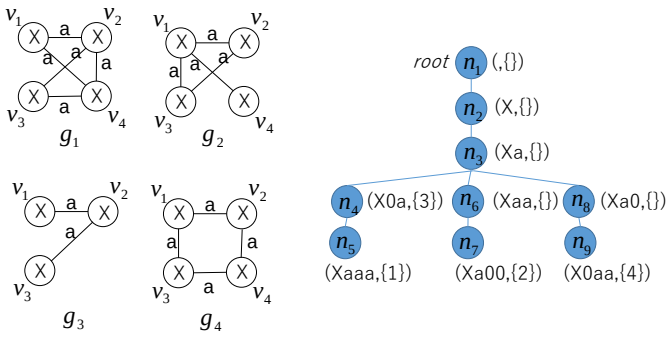

図 2 コード木

Algorithm 3: Code Tree Search for Problem (4)

Data: グラフ ID の集合 S，クエリグラフ q，閾値 θ，現時点での節点 $n, \langle w_1, w_2, \ldots, w_h \rangle$， 累積編集操作数 ae
Result: $\{g_i \in G \mid q' \subseteq_i q \; s.t. \; ed(q', g_i) \leq \theta, q'$は連結 $\}$

```
1  S ← S ∪ ⋃_{i∈ID(n)} {g_i};
2  N ← children(n);
3  if h < |V(q)| then
4      C ← {(w, s) | s_1 s_2 ··· s_h s =
         code(q, ⟨w_1, w_2, ..., w_h, w⟩) ⊆ c, c ∈ Ω(q)};
5      for (m, (w, s)) ∈ N × C do
6          e ← ed(fr(m), s);
7          if ae + e ≤ θ then
8              ω ← ⟨w_1, w_2, ..., w_h, w⟩;
9              S ← search(S, q, θ, m, ω, ae + e);
   else
10     for m ∈ N do
11         e ← ed(fr(m), null);
12         if ae + e ≤ θ then
13             S ← search(S, q, θ, m, null, ae + e);
14 return S;
```

$$code(g_1, \langle v_1, v_2, v_3, v_4 \rangle) = XXaX0aXaaa,$$
$$code(g_2, \langle v_1, v_2, v_3, v_4 \rangle) = XXaXaaXa00,$$
$$code(g_3, \langle v_1, v_2, v_3 \rangle) = XXaX0a, and$$
$$code(g_4, \langle v_1, v_2, v_3, v_4 \rangle) = XXaXa0X0aa.$$

図 2 の右図はこれらのグラフに対するコード木である．節点 n_5 はコードフラグメント $Xaaa$ とグラフ ID の集合 $ID(n) = \{1\}$ をもつ．根から節点 n_5 までのパス上の節点がもつコードフラグメントを連結したとき，$XXaX0aXaaa$ を得るが，それは g_1 の AcGM コードである．g_1 のコードだけでなく，上記の 4 つのコードが根からある節点までのパスとして表されている．

Algorithm 3 はコード木を巡回しながら問題 (4) の解を探索するアルゴリズムの擬似コードである．Algorithm 3 は節点 n において，$\langle w_1, w_2, \ldots, w_h \rangle$ から生成される q の AcGM コードに 1 つの頂点 w を追加する．その後，それによって追加されるフラグメントに類似するフラグメント

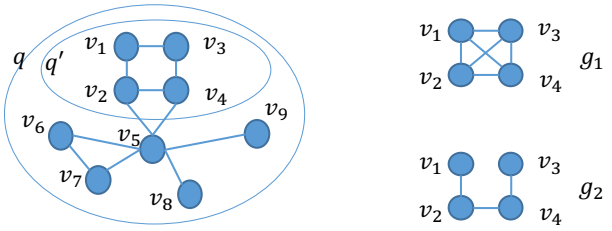

図 3 グラフの包含関係と類似

をもつ n の子節点を再帰的に探索する．Algorithm 2 では，2 行目においてデータベースから 1 つグラフ g_i を得て，g_i が解となるかを探索していたが，コード木ではデータベースの複数のグラフの AcGM コードの接頭辞が 1 つの節点に集約されているため，Algorithm 3 は複数のグラフと q の対応関係を調べることができるので，高速に動作する．

5. 元の問題への拡張

Algorithm 3 は深さ h にある節点 n にいるとき，長さが h 以下の 2 つのフラグメント系列 c と c' の累積操作数 ae を保持している．

- c はコード木の根から n に至るパス上の節点がもつフラグメントの連結あり，データベース中の幾つかのグラフが共通してもつ連結な誘導部分グラフのコード．
- c' は $\Omega(q)$ に含まれるコードの接頭辞である．AcGM コードの定義より，$g(c')$ は連結であり，かつ q の誘導部分グラフに制限される．

元の問題 (3) を解くためにこれらの制限を緩和する．$g(c')$ が連結であるという制約は，$code(q, \langle w_1, w_2, \ldots, w_h \rangle)$ が q の AcGM コード $\Omega(q)$ の接頭辞であるという制約から由来する．このため，Algorithm 3 の 4 行目で $\langle w_1, w_2, \ldots, w_h \rangle$ が任意の順列を取るように緩和すればよい．

つぎに，$g(c')$ が q の誘導部分グラフに限られるという制約を緩和する．Algorithm 1 では，q から連結誘導部分グラフを取り出すため，q のあるコードの接頭辞を切り出した．q の連結誘導部分グラフではなく，q の部分グラフを取り出すためには，q のコードの接頭辞だけではなく，コード内の任意の $x_{i,j}$ を 0 に置き換えたコードも取り出す必要がある．しかし，そうすると q の頂点の順列問題だけでなく，辺の組み合わせ問題も解く必要があるが，以下では後者が必要ないことを述べる．図 3 に示すグラフ q とその誘導部分グラフ q'，2 つのグラフ g_1 と g_2 を考える．

- $ed(q', g_1) = 2$ である．q' から 1 つの辺を除いた q'' を考えたとき，$ed(q', g_1) < ed(q'', g_1) = 3$ となる．問題 (3) では，$ed(q', g_i) \leq \theta$ を満たす q の部分グラフ（誘導部分グラフとは限らない）を見つければよい．したがって，“q' のある 2 頂点間に辺があり，かつそれに対応する辺が g_i にも辺がある場合”，q' の辺を取り除いた q'' を考える必要はない．つまり，q' のコードに含まれるある $x_{i,j}$ を 0 に置き換える必要はない．

Algorithm 4: Code Tree Search for Problem (3)

Data: グラフ ID の集合 S，クエリグラフ q，閾値 θ，現時点での節点 $n, \langle w_1, w_2, \ldots, w_h \rangle$，累積編集操作数 ae
Result: $\{g_i \in G \mid q' \subseteq q \; s.t. \; ed(q', g_i) \leq \theta\}$

```
1  S ← S ∪ ⋃_{i∈ID(n)} {g_i};
2  N ← children(n);
3  if h < |V(q)| then
4      C ← {(w, s) | s_1 s_2 ··· s_h s =
         code(q, ⟨w_1, w_2, ..., w_h, w⟩), ⟨w_1, w_2, ..., w_h, w⟩ ∈
         Φ(|V(q)|, h)};
5      for (m, (w, s)) ∈ N × C do
6          e ← ed_asym(fr(m), s);
7          if ae + e ≤ θ then
8              ω ← ⟨w_1, w_2, ..., w_h, w⟩;
9              S ← search(S, q, θ, m, ω, ae + e);
   else
10     for m ∈ N do
11         e ← ed_asym(fr(m), null);
12         if ae + e ≤ θ then
13             S ← search(S, q, θ, m, null, ae + e);
14 return S;
```

- $ed(q', g_2) = 1$ であるが，前述の q'' を考えたとき，$ed(q', g_2) > ed(q'', g_2) = 0$ となる．よって，“q' のある 2 頂点間に辺があり，かつそれに対応する辺が g_i にも辺が無い場合”，q' の対応する辺がない場合を考慮する必要はない．
- 上記以外は “q' のある 2 頂点間に辺なし場合” であるが，$x_{i,j} = 0$ であり，$x_{i,j}$ を 0 に置き換える必要はない．

上記の議論の結果として補題 4.3 の修正が必要となる．

補題 5.1 $ed(s, s') =$

$$\begin{cases} \delta'(\ell(v_i), \ell(v'_i)) + \sum\limits_{\substack{1 \leq j < i \\ x_{i,j} \neq 0}} \delta'(x_{j,i}, x'_{j,i}) & (s \neq null \wedge s' \neq null) \\ 1 + \sum\limits_{1 \leq j < i} \delta'(x_{j,i}, 0) & (s' = null) \\ 1 & (s = null) \end{cases} \tag{14}$$

この関数は非対称になるので，これ以降 $ed_{asym}(c, c')$ と書く．この関数により q の任意の部分グラフを考慮できるので，上記の下線を引いた接頭辞だけでよい．■

Algorithm 4 は問題 (3) を解くための擬似コードである．Algorithm 3 と Algorithm 4 の違いは，後者の 4，6，11 行目である．4 行目では，クエリ q の頂点の任意の順列 $\Phi(|V(q)|, h)$ によりコードを生成する．ここで生成されるコードは AcGM コードに限られない．また，6，11 行目では，式 (14) を用いる．これにより，4 行目において生成されるコードは q のコードの接頭辞に限られるため，提案手法である Algorithm 4 は効率的に動作する．

表 1　評価用データの特性

グラフ数 $\|G\|$	39,338
頂点ラベルの種類	59
辺ラベルの種類	3
平均頂点数	25.15
平均次数	2.16
最大頂点数	222
最大辺数	234
クエリの数 $\|Q\|$	234

表 2　1 クエリあたりの出力グラフ数

編集操作数の閾値 θ	1	2	3
出力グラフ数/1 クエリ	4.8	54.8	347.9

6. 評価実験

類似包摂グラフを検索する問題は，本稿で新たに提案した問題であるため，提案手法と比較できる手法はまだない．そこで．本稿の提案手法と従来の CodeTree [5] を比較する．提案手法において $\theta = 0$ のときは，提案手法の CodeTree が求める解と同じであり，θ を増やすことで，出力されるグラフ数や計算時間がどう変化するのかを調査する．

本節で示す評価実験は Xeon E2174 3.8GHz の CPU と 8GB の主記憶が搭載されたワークステーションで行った．また，評価実験に用いられたデータは，文献 [5] と同様に，39,338 個の化合物からなるデータ [7] であり，化合物の原子，結合，原子の種類，結合の種類をそれぞれ，グラフの頂点，辺，頂点ラベル，辺ラベルとして扱った．ただし，各化合物の水素，及びそれに繋がる結合は取り除かれている．この化合物をグラフに変換し，連結グラフのみをデータベース G とし，これから辺の数が 34 から 36 のグラフを取り出しクエリグラフの集合 Q とした．表 6 はこの評価実験に用いたデータの特性をまとめたものである．

表 6 は，θ を変化させたときの出力されるグラフ数である．θ を増やしたとき，出力されるグラフ数は指数的に増加する．G には部分グラフが互いに類似したグラフが多数存在するが，θ を増やすことにより，このようなグラフが検索されたからである．図 4 は，θ を変化させたときの各クエリに対する計算時間である．図の各点 (t, t_θ) は各クエリに対する $\theta = 0$ のときの計算時間 t と $\theta > 0$ のときの計算時間 t_θ を表している．θ を増やしたとき，計算時間もまた指数的に増加する．

7. まとめ

本稿では，類似包摂グラフ検索問題を新たに提案し，それに対する手法を提案した．提案手法は，CodeTree [5] に基づいており，あるクエリが与えられたとき，データベースの複数のグラフとの頂点の対応関係を探索するため効率的である．さらに，クエリ q の部分グラフ $q_s \subseteq q$ と類似するグラフをデータベースから検索する場合は，q の辺を q_s に含めるかで膨大な種類の q_s が考えられるが，それが必要がないことを 5 節で議論し，その議論に基づく編集操作数を補題 5.1 によって示した．これにより提案手法は，効率よく q_s の類似グラフを検索できる．今後，提案手法の効率性を様々な実験を通して解明していく予定である．

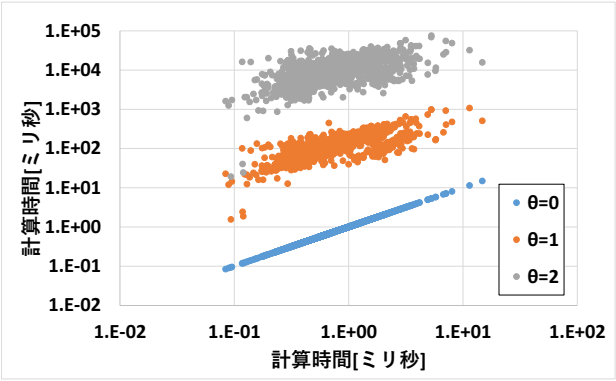

図 4　様々なクエリに対する計算時間

参考文献

[1] M. Cannataro, et al. Protein-to-Protein Interactions: Technologies, Databases, and Algorithms. *ACM Computing Surveys*, Vol. 43, No. 1, pp. 1:1–1:36, 2010.
[2] C. Chen, et al. Towards Graph Containment Search and Indexing. In *Proc. of 33rd Int'l Conf. on VLDB*, pp. 926–937, 2007.
[3] J. Cheng, et al. Fast Graph Query Processing with A Low-Cost Index. *VLDB J.*, Vol. 20, No. 4, pp. 521–539, 2011.
[4] R. Giugno, et al. Grapes: A Software for Parallel Searching on Biological Graphs Targeting Multi-Core Architectures. *PLOS ONE*, Vol. 8, No. 10, pp. e76911, 2013.
[5] S. Imai and A. Inokuchi. Efficient Supergraph Search Using Graph Coding. *IEICE Transactions on Information and Systems*, Vol. 103, No. 1, pp. 130–141, 2020.
[6] A. Inokuchi, et al. A Fast Algorithm for Mining Frequent Connected Subgraphs. Technical Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.
[7] S. Kramer, et al. Molecular Feature Mining in AIDS Data. In *Proc. of Int'l Conf. on Data Mining*, pp. 136–143, 2001.
[8] B. Lyu, et al. Scalable Supergraph Search in Large Graph Databases. In *Proc. of IEEE 32nd Int'l Conf. on Data Engineering*, pp. 157–168, 2016.
[9] K. Riesen. *Structural Pattern Recognition with Graph Edit Distance -Approximation Algorithms and Applications-*, Advances in Computer Vision and Pattern Recognition. Springer, 2015.
[10] 白木 公康. 緊急寄稿（2）新型コロナウイルス感染症（COVID-19）治療候補薬アビガンの特徴. **週刊日本医事新報**, 5005 号, pp. 25, 2020.
[11] S. Sun and Q. Luo. Scaling Up Subgraph Query Processing with Efficient Subgraph Matching. In *Proc. of 2019 IEEE 35th Int'l Conf. on Data Engineering*, pp. 220–231, 2019.
[12] Y. Xie and P. S. Yu. CP-Index: on The Efficient Indexing of Large Graphs. In *Proc. of The 20th ACM Int'l Conf. on Information and Knowledge Management*, pp. 1795–1804, 2011.
[13] D. Yuan and P. Mitra. Lindex: A Lattice-Based Index for Graph Databases. *The VLDB J.*, Vol. 22, No. 2, pp. 229–252, 2013.
[14] D. Yuan, et al. Mining and Indexing Graphs for Supergraph Search. In *Proc. of The VLDB Endowment*, Vol. 6, No. 10, pp. 829–840, 2013.