Session 9
Graph Data 2

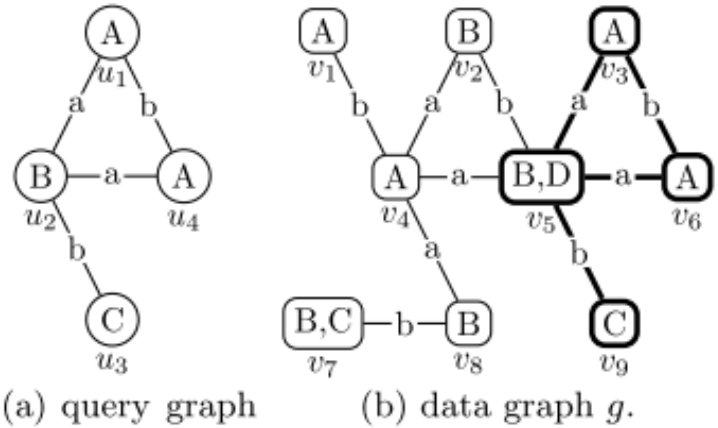
Jinsoo Lee, Wook-Shin Han, Romans Kasperovics, Jeong-Hoon Lee: An In-depth Comparison of Subgraph Isomorphism Algorithms in Graph Databases

An In-depth Comparison of Subgraph Isomorphism Algorithms in Graph Databases

- データベース中のグラフdからクエリグラフgと同型のサブグラフを検索するアルゴリズムの性能を比較
- **比較対象**
  - Ullmannの手法 (1976)

小さなグラフ用
  - VF2 [Cordella, PAMI04]
  - QuickSI [Shang, PVLDB08]

大規模グラフ用
  - GraphQL[He, SIGMOD08]
  - GADDI[Zhang, EDBT09]
  - SPath[Zhao, PVLDB10]



(a) query graph (b) data graph g.

- **得られた結果**
  1. QuickSIがGraphQL, GADDI, Spathより性能がよい
  2. QuickSI, VF2, GADDIは木データに対するサブグラフを妥当な時間内に発見できなかった
  3. GraphQLは唯一すべてのクエリセットを処理できるが，多くのデータセットでQuickSIより遅い
  4. SPathは一貫してGraphQLより遅い（使ってる言語違うじゃんというツッコミ）
  5. 全てのアルゴリズムはjoin order selectionと同じ問題を持つ

アルゴリズムに共通するフレームワーク

Algorithm 1 GENERICQUERYPROC

Input: query graph q
Input: data graph g
Output: all subgraph isomorphisms of q in g

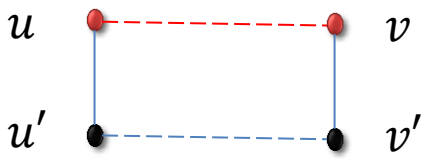
```
1: M := ∅;
2: for each u ∈ V(q) do
3:     C(u) := FILTERCANDIDATES (q, g, u, ...);
       [[ ∀v ∈ C(u)((v ∈ V(g)) ∧ (L(u) ⊆ L(v))) ]]
4:     if C(u) = ∅ then
5:         return;
6:     end if
7: end for
8: SUBGRAPHSEARCH (q, g, M, ...);
```

Subroutine SUBGRAPHSEARCH $(q, g, M, \ldots)$

```
1: if |M| = |V(q)| then
2:     report M;
3: else
4:     u := NEXTQUERYVERTEX (...);
       [[ u ∈ V(q) ∧ ∀(u', v) ∈ M(u' ≠ u) ]]
5:     C_R := REFINECANDIDATES (M, u, C(u), ...);
       [[ C_R ⊆ C(u) ]]
6:     for each v ∈ C_R such that v is not yet matched do
7:         if ISJOINABLE (q, g, M, u, v, ...) then
8:             [[ ∀(u', v') ∈ M((u, u') ∈ E(q) ⟹
                   (v, v') ∈ E(g) ∧ L(u, u') = L(v, v')) ]]
9:             UPDATESTATE (M, u, v, ...);
               [[ (u, v) ∈ M ]]
10:            SUBGRAPHSEARCH (q, g, M, ...)
11:            RESTORESTATE (M, u, v, ...);
               [[ (u, v) ∉ M ]]
12:        end if
13:    end for
14: end if
```

再帰関数
- 再帰の回数をいかに減らすかがキモだと思っていたのだが…

- FILTERCANDIDATES
  - クエリ内のノードuに対する候補ノード$C(u)$を検索する
- NEXTQUERYVERTEX
  - u'の次にマッチングするノードuを選択する
- REFINECANDIDATES
  - 候補ノード$C(u)$を$C_R \subset C(u)$に絞り込む
- ISJOINABLE
  - クエリ内の(u, u')とDB内の(v, v')がマッチするか



サブグラフ同型アルゴリズム その1
（Ullmannの手法をベースにしたもの）

	Ullmann	VF2	QuickSI
FILTER CANDIDATES	ラベルが一致するノードを候補にする	（Ullmannと同様）	（Ullmannと同様）
NEXT QUERYVERTEX (u'の次ノードu)	u'の次に入力されたノード	既にマッチしたクエリサブグラフM_qに隣接するノード	ノードラベルと辺ラベルが低頻度な隣接ノード
REFINE CANDIDATE	uより次数の低い$v \in C(u)$を消す	uがM_qに隣接することを利用した枝刈(※1)	最小全点木を利用した枝刈（※2)
ISJOINABLE	uの隣接ノードをすべてチェックする	(Ullmannと同様)	必要ない（実は結構重要）

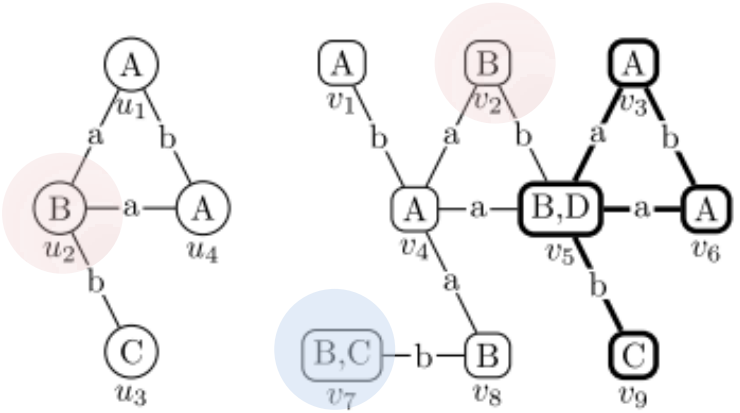
※1：後で書く
※2：クエリグラフで最小全点木上のuの親ノードu'に対応するDBグラフ上のノードv'とvがつながっていなければ消す

近接シグネチャ（GraphQL, SPath）

- GraphQL neighborhood signature
  - 隣接ノードのラベルの集合

    $sig_{GraphQL}(u_2) = \{A, A, C\}$

    $sig_{GraphQL}(v_2) = \{A, B, D\}$



(a) query graph (b) data graph g.

- SPath neighborhood signature
  - 近接ノードに関するtripleの集合
    - (d, l, c) : uからの最短距離がdでラベルがlである点がc個ある

      $sig_{GraphQL}(v_7) = \{(1, B, 1)\}$ （$k_0 = 1$の場合）

      $sig_{GraphQL}(v_7) = \{(1, B, 1), (2, A, 1)\}$ （$k_0 = 2$の場合）

サブグラフ同型アルゴリズム その2（シグネチャを使う方法）

- シグネチャを使って早めに候補を少なくし，再帰の回数をできるだけ削減する

	GADDI	GraphQL	SPath
FILTER CANDIDATES	（省略）	Ullmannの手法＋枝刈 i) $sig_{GraphQL}(u) \not\subseteq sig_{GraphQL}(v)$な$v$を排除 ii) 擬似同型を使った枝刈(※1)	Ullmannの手法＋枝刈 • 距離kまでののノード数の合計がuより少ないラベルがあったら排除
NEXT QUERYVERTEX (u'の次ノードu)	（省略）	中間結果のサイズを最小にする隣接ノードを選択	
REFINE CANDIDATE	（省略）		
ISJOINABLE	（省略）		

```
Subroutine SubgraphSearch (q, g, M, k0)
 1: if |M| = |V(q)| then
 2:    report M;
 3: else
 4:    pq := NextQueryPath (q, g, k0);
 5:    P := GetCandidatePaths (pq, M, C);
 6:    for each pg ∈ P do
 7:       if IsJoinable (pq, pg, M) then
 8:          UpdateState (pq, pg, M);
 9:          SubgraphSearch (q, g, M, k0);
10:          RestoreState (pq, pg, M);
11:       end if
12:    end for
13: end if
```

ノード単位でなく
Path単位でマッチングする

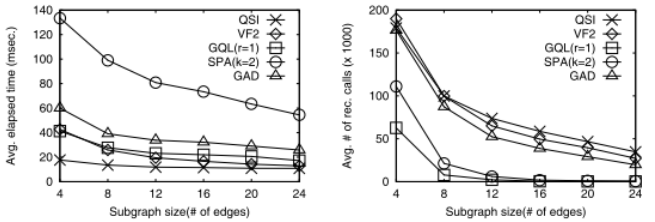
評価

Table 1: Summary of the real-world datasets.

Dataset	AIDS	NASA	Yeast	Human
# of graphs	10000	36790	1	1
# of vertices	2~214	2~889	3112	4675
# of edges	1~217	1~888	12519	86282
Avg. degree	1.95	1	8.05	36.82
Max. degree	11	245	168	771
# of distinct v. labels	51	117302	184	90
# of distinct e. labels	4	0	0	0
Avg. # of labels per vertex	1	1	7.55	4.63

- 問合せ処理

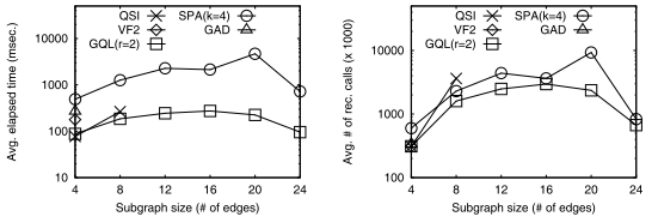
AIDS: 小さく疎なグラフ



(a) average elapsed time. (b) avg. # of recursive calls.

Figure 6: AIDS dataset.

NASA:ラベルの種類の多い木構造データ



(a) average elapsed time. (b) avg. # of recursive calls.

Figure 7: NASA dataset.

- SPathは再帰回数は減っているが一番遅い
- QuickSIは再帰回数多いが速い
  - ISJOINABLEをしなくていいのが効いている
- GraphQLはFILTERCANDIDATESの処理が全体の63.16%
  - SPathは61.23%
- SPathはPathベースのマッチングでオーバーヘッドがかかっている

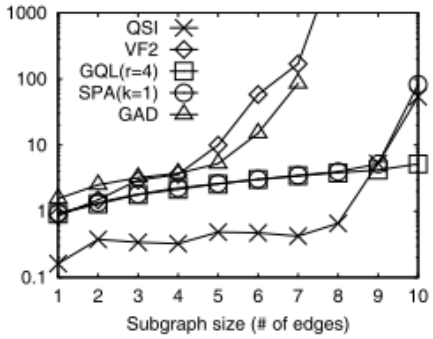
- VF2, QuickSI, GADDIは指数オーダで増加して結果が得られなくなった
- GraphQL, SPathは最後まで処理できた

シグネチャによる早めの枝刈が効いている

- GraphQLはSPathより平均10.38倍速い
  - シグネチャの生成時間(5.5倍)
  - FILTERCANDIDATESの時間(2.45倍)

評価

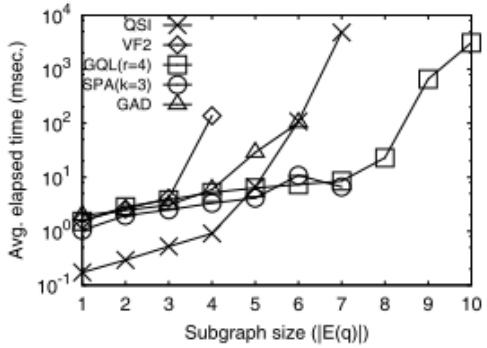
Yeast:1個の大きく密なグラフ



(a) subgraph queries.

- QuickSI,GraphQL, SPath以外は小さなサブグラフで計算不能に
- QuickSIが最も性能がよい
- シグネチャによる枝刈があまり効果がない
  - 半数くらいにしか減らない

Human:Yeastより
大きく密な1つのグラフ



(a) subgraph queries.

- GraphQL以外は小さなサブグラフで計算不能に
- クエリサイズが大きいと探索空間サイズが大きくなるため
- join order selection problem