Improving Subgraph Isomorphism with Pruning by Bipartite Maximum Matching

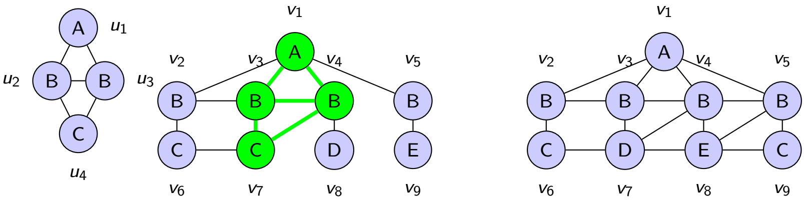
Yunyoung Choi

Theory and Application Lab

November 25, 2023

Outline

Problem Definition



(a) query graph q

(b) data graph G_1

(c) data graph G_2

Definition

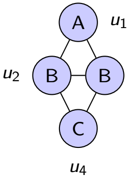
Given a query graph q = (V(q), E(q), L_q) and a data graph G = (V(G), E(G), L_G), an embedding is a function M : $V(q) \rightarrow V(G)$ such that.

1. M is injective (i.e., $M(u) \neq M(u')$ for $u \neq u'$ in $V(q)$).
2. $L_q(u) = L_G(M(u))$ for every $u \in V(q)$.
3. $(M(u), M(u')) \in E(G)$ for every $(u, u') \in E(q)$.

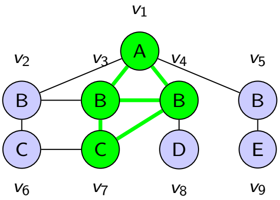
An embedding of an induced graph is a partial embedding.

- There is an embedding to G_1, $\{(u_1, v_1), (u_2, v_3), (u_3, v_4), (u_4, v_7)\}$
- There is no embedding to G_2.

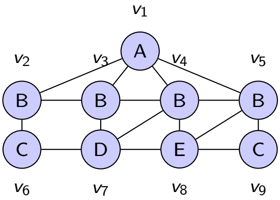
Problem Definition



(a) query graph q



(b) data graph G_1



(c) data graph G_2

Definition

Subgraph isomorphism is the problem of determining whether there is embedding or not.

- general framework (filtering-verification framework)
  1. Filtering : finding a candidates set C(u) for each $u \in V(q)$.
     - ★ (e.g. $C(u_1) = \{v_1\}$, $C(u_2) = \{v_2, v_3, v_4\}$)
  2. Verification : Choosing matching order and applying backtracking.
     - ★ (e.g. $(u_1, v_1) \rightarrow (u_2, v_3) \rightarrow (u_3, v_4) \rightarrow (u_4, v_7)$)
- state-of-the-art algorithms
  - *Turbo*$_{iso}$ [Han, Lee Lee. In Proceedings of SIGMOD, 2013]
  - *CFL* – *Match* [Bi, Chang, Lin, Qin Zhang, In Proceedings of SIGMOD, 2016]
  - *DAF* [Han, Kim, Gu, Park Han. In Proceedings of SIGMOD, 2019]
- DAF is a baseline in this paper.

Overview of DAF

Algorithm 0: DAF

Input: a query graph q, a data graph G

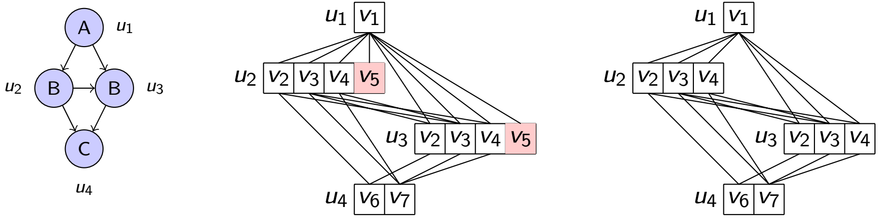
Output: true/false(is there any isomorphic subgraph)

$q_D \leftarrow BuildDAG(q, G)$;
$C \leftarrow BuildCS(q, q_D, G)$;
$M \leftarrow \emptyset$;
if $\textsc{Backtrack}(q, G, C, M)$ **then**
    return true
else
    return false

1. BuildDAG
   - Make DAG q_D from q by directing all edges.
2. BuildCS
   - Make candidate sets called CS structure
3. Backtrack
   - Find at least one embedding by backtracking.

Filtering

- BuildDAG
  - Make rooted DAG q_D by directing all edges in q.
- BuildCS (with DAG DP)
  - CS is a complete search space.
    - ★ $C(u)$ is a set of candidates can be mapped to u.
    - ★ $v_1 \in C(u_1)$ and $v_2 \in C(u_2)$ are connected if and only if $(u_1, u_2) \in E(q)$ and $(v_1, v_2) \in E(G)$.
  - DAG DP in DAF makes CS more compact.

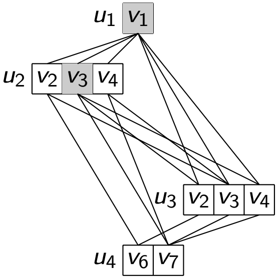


(a) q_D (b) C_{ini} (c) C from DAG DP

Verification

Algorithm 0: $\textsc{Backtrack}(q, q_D, \mathrm{CS}, M')$

if $|M| = |V(q_D)|$ **then**
    return true;
else
    Select a next extendable vertex u;
    foreach $v \in C_M(u)$ **do**
        if *v is unvisited* **then**
            $M' \leftarrow M \cup \{(u, v)\}$;
            Mark v as visited;
            if $\textsc{Backtrack}(q, q_D, \mathrm{CS}, M') ==$ TRUE **then**
                return true;
            Mark v as unvisited;
    return false;



- Given a partial embedding $M = \{(u_1, v_1), (u_2, v_3)\}$
- u_3 is an extendable vertex.
- u_4 is not an extendable vertex.

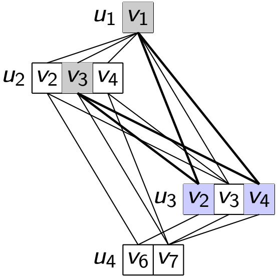
Definition

An unvisited query vertex is extendable regarding partial embedding M if all its parents are matched in M.

Verification

Algorithm 1: BACKTRACK(q, q_D, CS, M')

if $|M| = |V(q_D)|$ **then**
    return true;
else
    Select a next extendable vertex u;
    foreach $v \in C_M(u)$ **do**
        if *v is unvisited* **then**
            $M' \leftarrow M \cup \{(u, v)\}$;
            Mark v as visited;
            if BACKTRACK(q, q_D, CS, M') == TRUE **then**
                return true;
            Mark v as unvisited;
    return false;



- A given partial embedding $M = \{(u_1, v_1), (u_2, v_3)\}$
- Extendable candidates $C_M(u_3) = \{v_2, v_4\}$
- M' $\leftarrow$ M $\cup \{(u_4, v_7)\}$

Definition

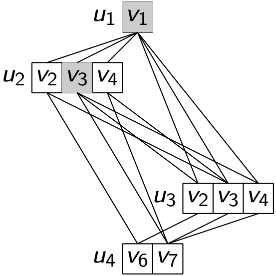
Extendable candidates given partial embedding M.
($N_u^{u_p}(v_p)$ is the list of vertices v adjacent to v_p such that v $\in C(u)$.)

$$C_M(u) \begin{cases} \text{C(u)} & \textit{no mapped parent vertices of } u \\ (\bigcap_{i=1}^{k} N_u^{p_i}(M(p_i))) & \textit{mapped parent vertices of } u = \{\, p_1, p_2, ..., p_k \} \end{cases}$$

Verification

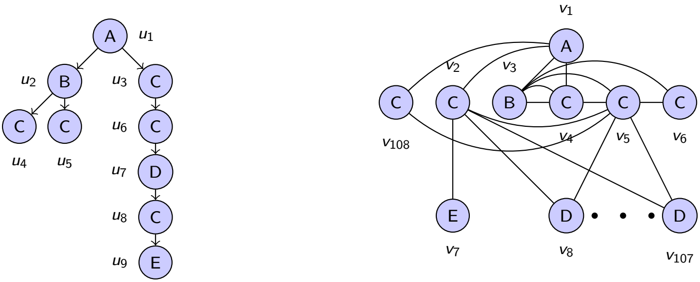
Algorithm 2: $\textsc{Backtrack}(q, q_D, \mathrm{CS}, M')$

```
if |M| = |V(q_D)| then
    return true;
else
    Select a next extendable vertex u;
    foreach v ∈ C_M(u) do
        if v is unvisited then
            M′ ← M ∪ {(u, v)};
            Mark v as visited;
            if BACKTRACK(q, q_D, CS, M′) == TRUE then
                return true;
            Mark v as unvisited;
    return false;
```



- DAF suggests pruning by failing sets.

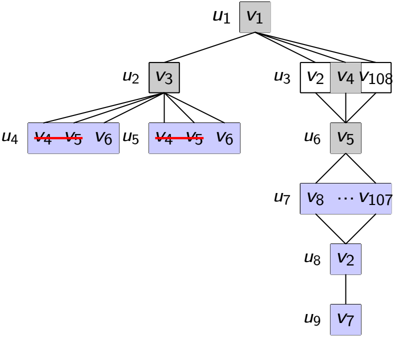
Method1 : Pruning By the Maximum Bipartite Matching

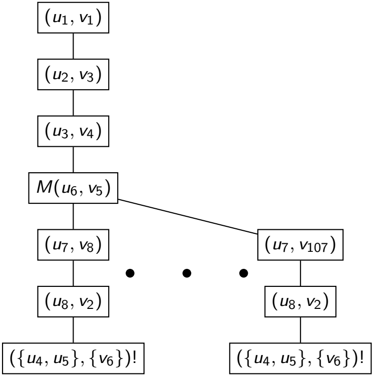


(a) q_D from q (b) G

Figure: query graph q and data graph G

- Partial embedding $M = \{(u_1, v_1), (u_2, v_3), (u_3, v_4), (u_6, v_5)\}$





(a) Bipartite for M

- $M = \{(u_2, v_3), (u_2, v_3), (u_3, v_4), (u_6, v_5)\}$
- There is only one vertex that can be mapped for two vertices u_4 and u_5.
  - $u_4 \to \{v_6\}$
  - $u_5 \to \{v_6\}$
- The injectiveness cannot be satisfied.
- Exploration subtree rooted at M is unnecessary.

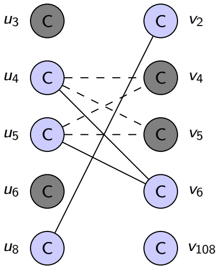
Definition

$V(q)_M$ (a set of mapped query vertices). $V(G)_M$ (a set of mapped data vertices.)
$V(q) \setminus V(q)_M$ (a set of unmapped query vertices).
$V(G) \setminus V(G)_M$ (a set of mapped data vertices.)
Defining bipartite graph $B_M = (V(B_M), E(B_M), L_{B_M})$ as follows

- $V(B_M) = (V(q) \setminus V(q)_M) \cup (V(G) \setminus V(G)_M)$
- $(u, v) \in E(B_M)$ if and only if $v \in C_M(u)$

Lemma

If the size of maximum bipartite matching is less than $|V(q) \setminus V(q)_M|$, M cannot be extended to any full embedding.



- Induced bipartite graph from nodes with label C.
  - $V(q) \setminus V(q)_M = \{u_4, u_5, u_8\}$.
  - $V(G) \setminus V(G)_M = \{v_2, v_6, v_{108}\}$.
  - $E(B_M) = \{(u_4, v_6), (u_5, v_6), (u_8, v_2)\}$

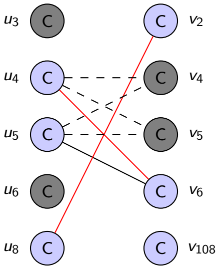
Definition

$V(q)_M$ (a set of mapped query vertices). $V(G)_M$ (a set of mapped data vertices.)
$V(q) \setminus V(q)_M$ (a set of unmapped query vertices).
$V(G) \setminus V(G)_M$ (a set of mapped data vertices.)
Defining bipartite graph $B_M = (V(B_M), E(B_M), L_{B_M})$ as follows

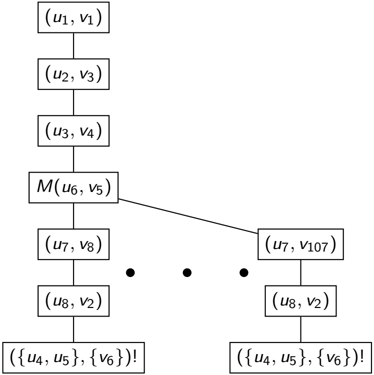
- $V(B_M) = (V(q) \setminus V(q)_M) \cup (V(G) \setminus V(G)_M)$
- $(u, v) \in E(B_M)$ if and only if $v \in C_M(u)$

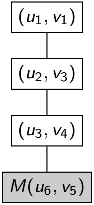
Lemma

If the size of maximum bipartite matching is less than $|V(q) \setminus V(q)_M|$, M cannot be extended to any full embedding.



- Induced bipartite graph from nodes with label C.
  - $V(q) \setminus V(q)_M = \{u_4, u_5, u_8\}$.
  - $V(G) \setminus V(G)_M = \{v_2, v_6, v_{108}\}$.
  - $E(B_M) = \{(u_4, v_6), (u_5, v_6), (u_8, v_2)\}$
- maximum bipartite matching is $\{(u_4, v_6), (u_8, v_2)\}$.
- no injective mappings for $\{u_4, u_5, u_8\}$





- Prune the search tree rooted M.

Method2 : Failing Set for the Maximum Bipartite Matching

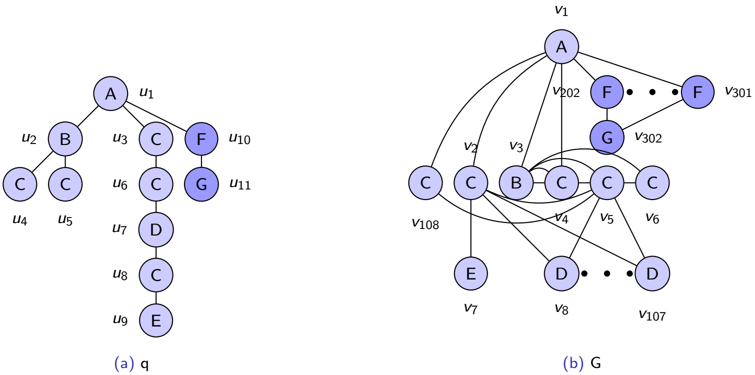
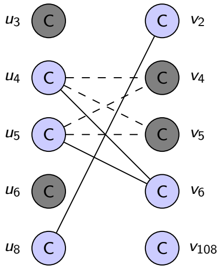
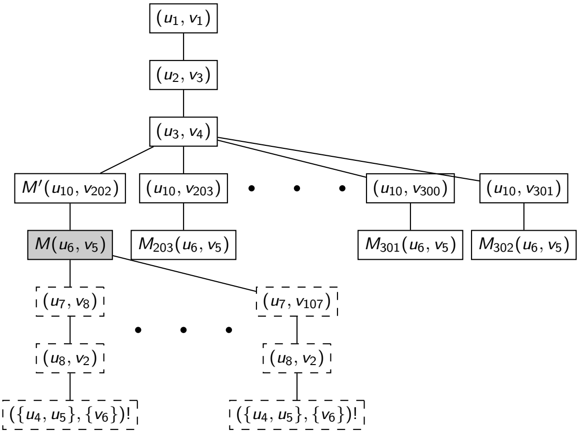


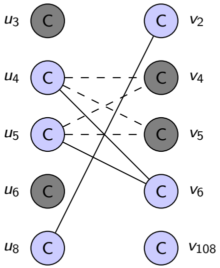
Figure: query graph q and data graph G

- Extended examples
  - adding u_{10}, u_{11} in q.
  - adding $v_{202}, ..., v_{301}, v_{302}$ in G.
- Partial embedding $M = \{(u_1, v_3), (u_2, v_3), (u_3, v_4), (u_{10}, v_{202}), (u_6, v_5)\}$.

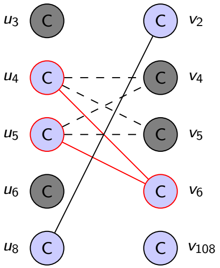




- B_M is same as the previous bipartite graph.
- Bipartite graph for label C are same in M_{202}, M_{203} , ... , M_{301}.
- Siblings of M' should be pruned.



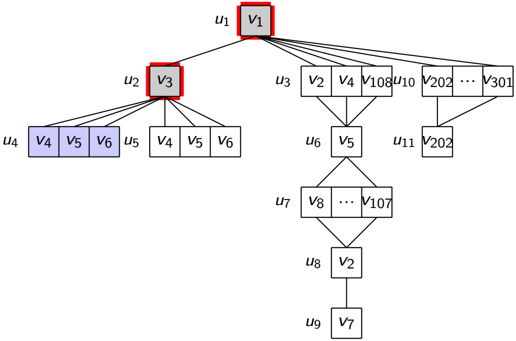
(a) Bipartite for M

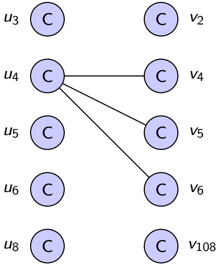


(b) Bipartite for M

Figure: Bipartite graph

- Cause of the conflict is the induced graph of $\{u_4, u_5, v_6\}$.
- Four ingredients.
  - $C_M(u_4) = \{v_4, v_5, v_6\}$
  - $C_M(u_5) = \{v_4, v_5, v_6\}$
  - (u_3, v_4)
  - (u_6, v_5)

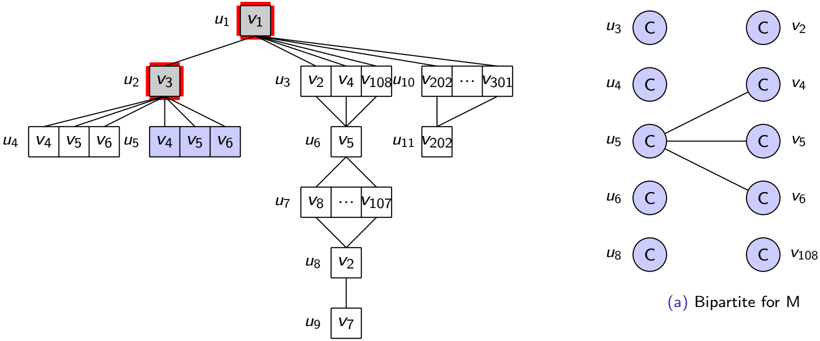




(a) Bipartite for M

Figure: Bipartite graph

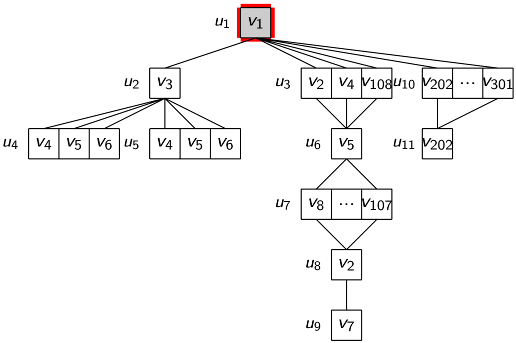
- $C_M(u_4) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$

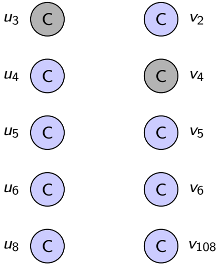


(a) Bipartite for M

Figure: Bipartite graph

- $C_M(u_4) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- $C_M(u_5) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$

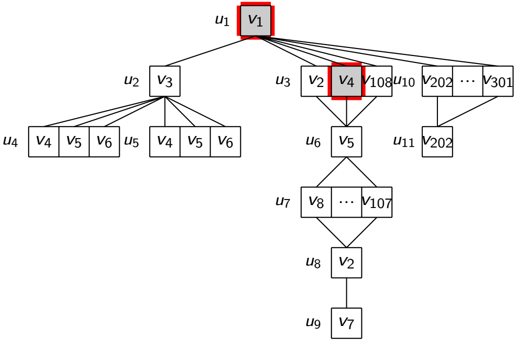


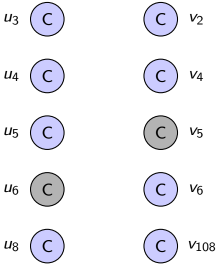


(a) Bipartite for M

Figure: Bipartite graph

- $C_M(u_4) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- $C_M(u_5) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- (u_3, v_4) : determined by $\{(u_1, v_1)\}$

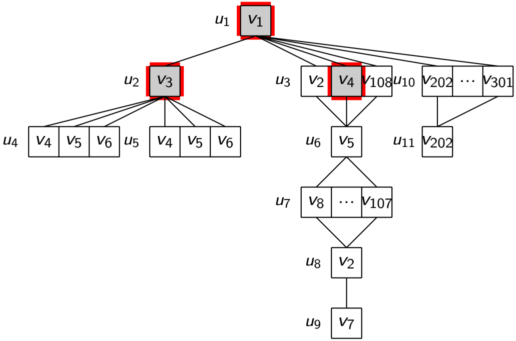


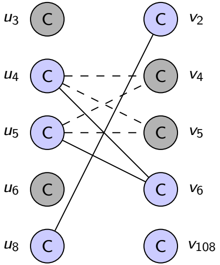


(a) Bipartite for M

Figure: Bipartite graph

- $C_M(u_4) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- $C_M(u_5) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- (u_3, v_4) : determined by $\{(u_1, v_1)\}$
- (u_6, v_5) : determined by $\{(u_1, v_1), (u_3, v_4)\}$

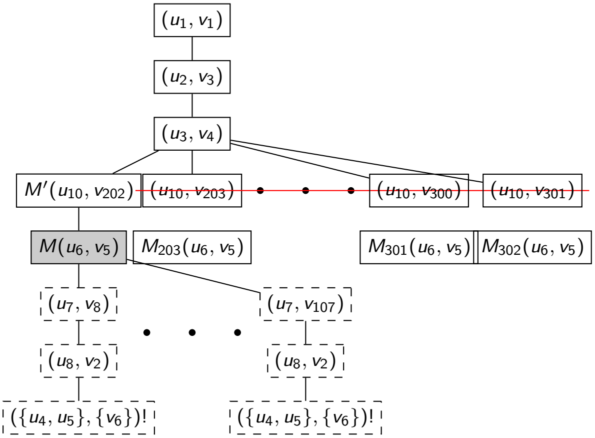




(a) Bipartite for M

Figure: Bipartite graph

- $C_M(u_4) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- $C_M(u_5) = \{v_4, v_5, v_6\}$: determined by $\{(u_1, v_1), (u_2, v_3)\}$
- (u_3, v_4) : determined by $\{(u_1, v_1)\}$
- (u_6, v_5) : determined by $\{(u_1, v_1), (u_3, v_4)\}$
- Induced graph of $\{u_4, u_5, v_6\}$ determined by $\{(u_1, v_1), (u_2, v_3), (u_3, v_4)\}$



- $(u_{10}, v_{202}) \notin \{(u_1, v_1), (u_2, v_3), (u_3, v_4)\}$
- Mappings for u_{10} lead same pruning.
- Prune siblings of (u_{10}, v_{202}).

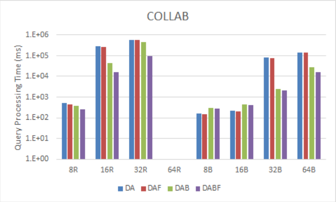
Real world graph data

- Using four real data graph (COLLAB, IMDB, pcms, PPI)
  - Eight query sets are used
    - Random-walk with i edges and BFS with i edges ($i \in \{8, 16, 32, 64\}$).
  - Each query set consists of 100 query graphs.
- Four variants of DAF (DA, DAF, DAB, DABF)
  - Use DAG DP filtering
  - candidate size matching order (choose an extendable vertex with the minimum extendable size)
  - PMBM (pruning by maximum bipartite matching)
  - PFS (pruning by failing sets)
- We set the time limit to 10 minutes.

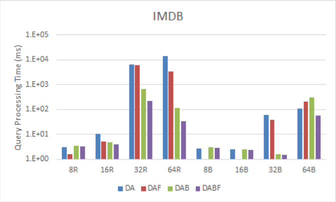
variants	PMBM	PFS
DA	X	X
DAF	X	O
DAB	O	X
DABF	O	O

	Dataset		Average per graph			
	$\lvert D\rvert$	$\lvert\Sigma\rvert$	$\lvert V(G)\rvert$	$\lvert E(G)\rvert$	degree	$\lvert\Sigma\rvert$
PCMS	200	21	377	4,340	23.01	18.9
PPI	20	46	4,942	26,667	10.87	28.5
IMDB	1,500	10	13	66	10.14	6.9
COLLAB	5,000	10	74	2,457	65.97	9.9

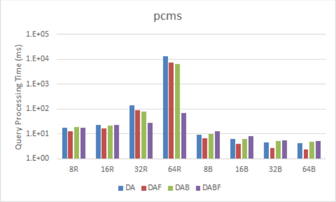
Processing time



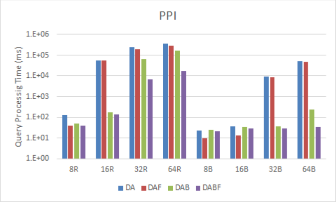
(a) COLLAB



(b) IMDB



(c) pcms

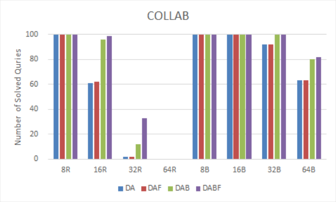


(d) PPI

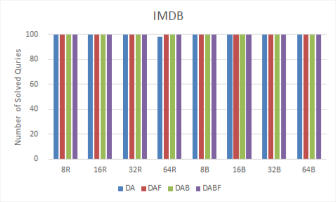
Figure: **Query processing time**

- Time gap between DA and DAB shows effectiveness of pruning by bipartite maximum match.
- Time gap between DAB and DABF show method for calculating failing set work well.
- Especially, our method shows significant in IMDB 64R, pcms 64R, PPI 16R, 64B.
- Poor performance on small query sets like pcms.

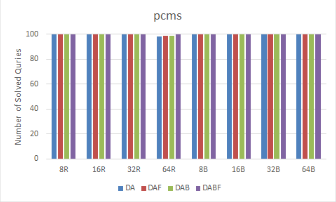
Solved query numbers



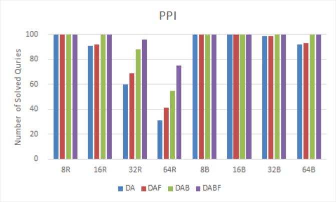
(a) COLLAB



(b) IMDB



(c) pcms



(d) PPI

Figure: **Number of solved queries**

- Number of solved queries among 100 queries.
- Pruning by maximum bipartite matching shows effectiveness in COLLAB.

Conclusion

- Pruning by maximum bipartite matching
  - New pruning technique by using maximum bipartite matching
  - Method for calculating failing set to use pruning by failing sets proposed by DAF.
- Effectiveness
  - Probing effectiveness by conducting on real word graph data.
- Future work
  - We should to other subgraph isomorphism algorithms

- [1] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. Efficient subgraph matching by postponing cartesian products. In Proceedings of the 2016 International Conference on Management of Data, pages 1199–1214, 2016.
- [2] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together.In Proceedings of the 2019 International Conference on Management of Data,pages 1429–1446, 2019.
- [3] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. Turboiso: towards ultra fast and robust subgraph isomorphism search in large graph databases. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pages 337–348, 2013.