Subgraph Isomorphism in GPU

George Joseph (CS15M021)

Guided by : Rupesh Nasre

Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Outline

Motivation

Applications

- Social Network
- Bio-Informatics
- Chemical Compound matching
- Pattern Recognition

- Subgraph Isomorphism Query is one of the most important graph queries.

Problem Statement

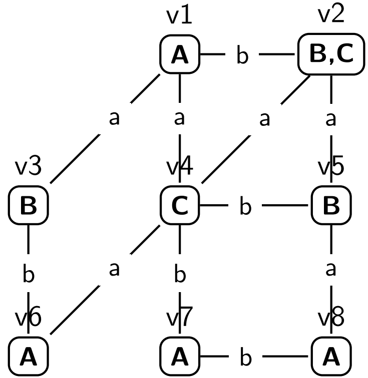
Input

A data Graph D, and Query Graph Q. The graphs D and Q are undirected with nodes and edges having label.
Graphs given as adjacency list.

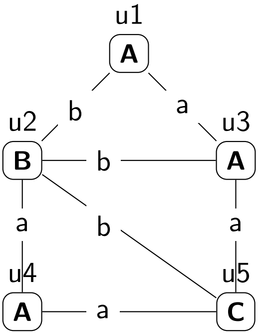
Output

Give all the matching mapping of each node in Q to node in D

Example



Data Graph



Query Graph

General Algorithm

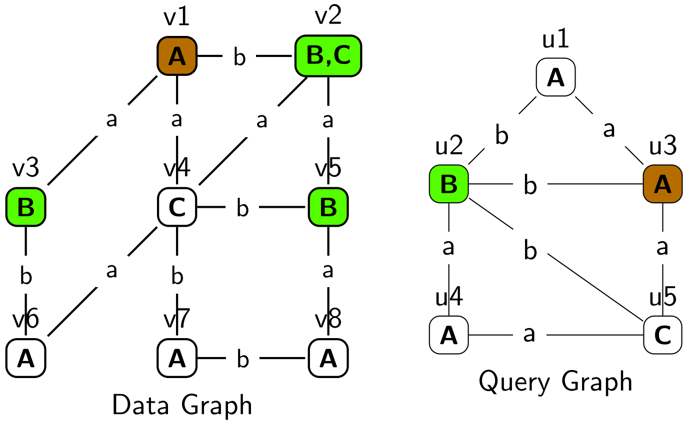
Algorithm 1 SubGraph Isomorphism

Input: Query Graph (Q) and Data Graph(D)

1. Find Candidates(CS) for each vertex in Q
2. Procedure SubgraphMatching
    1. if all vertices processed output the map
    2. u= NextVertex(Q)
    3. C_r= RefinedSet(CS,Q,D,v)
    4. for each v in C_r
        1. if IsJoinable(Q,D,M,v)
        2. UpdateState(M,v)
        3. Call SubgraphMatching
        4. RestoreState(M,v)

Output: Mappings M

Candidate Vertex Set



- Based on Node Value
- Based on Degree

Sugbraph Matching

NextVertex

- Order of processing query nodes
- Taking u2 first gives a more refined search than taking any other node.

RefinedSet

- Remove already mapped nodes
- Once u3 is mapped v1 can be removed from CVS of u1.

IsJoinable

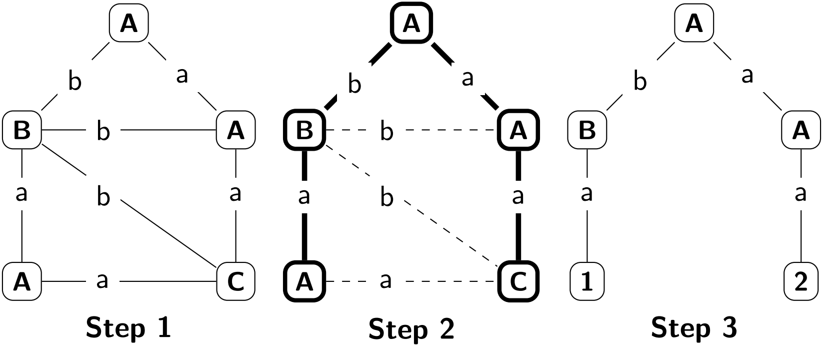
- Check existence of edges

UpdateState and Restore State

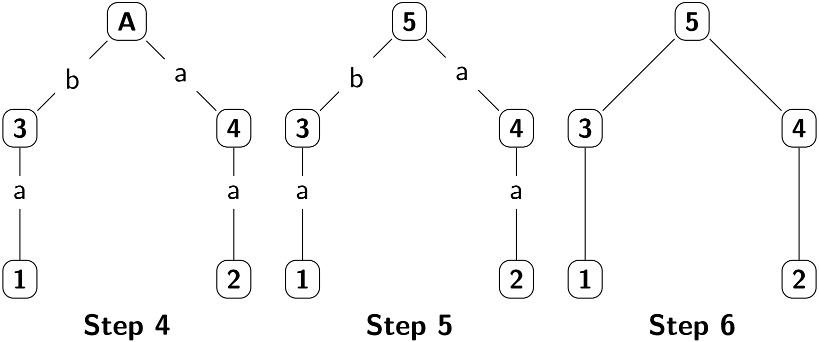
- Push and restore current mapped vertex

TurboIso

Neighbourhood Equivalence Class

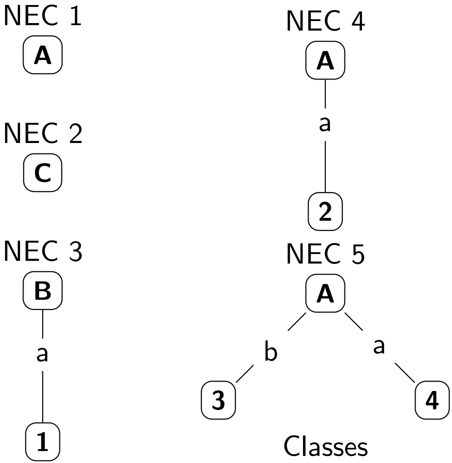


TurboIso
Neighbourhood Equivalence Class



TurboIso

Neighbourhood Equivalence Class

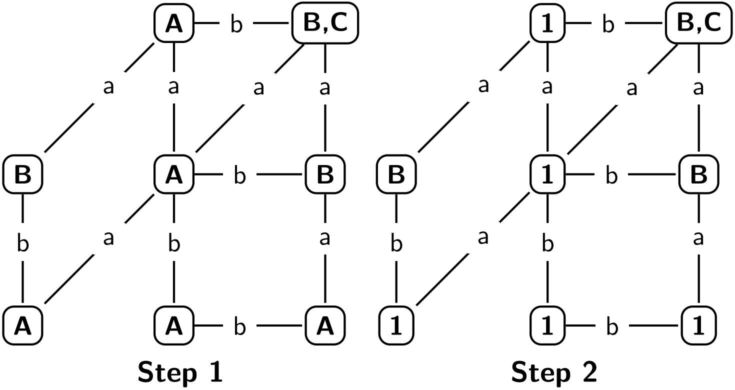


Classes

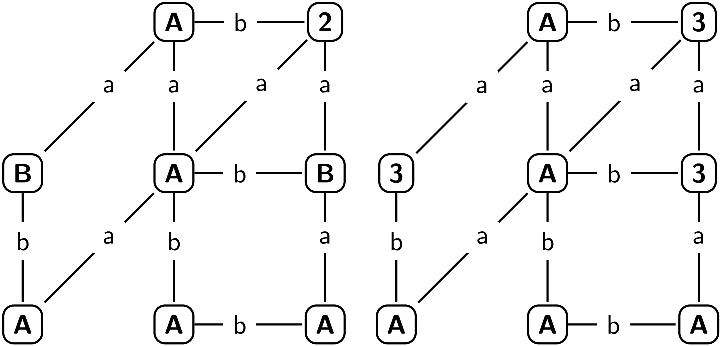
Parallel NEC

- Identify neighbourhood
  1. (Parallel) If all neighbours have numbered
  2. Find the hash of neighbourhood
- Use thrust-Scan to assign unique numbers
- Find unique number for the neighbourhood
  1. (Parallel) find the hash of neighbourhood
  2. Assign the unique number at the hash location

CVS Generation

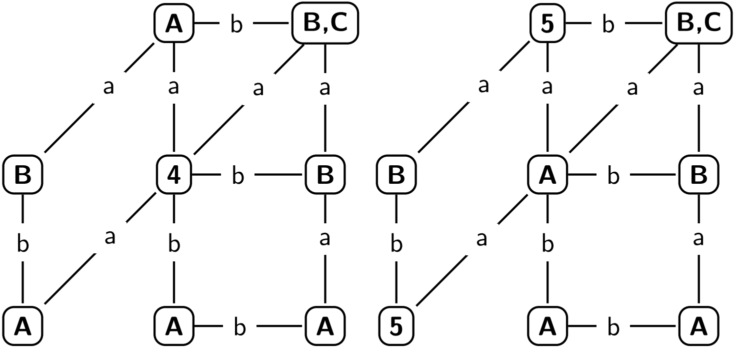


CVS Generation



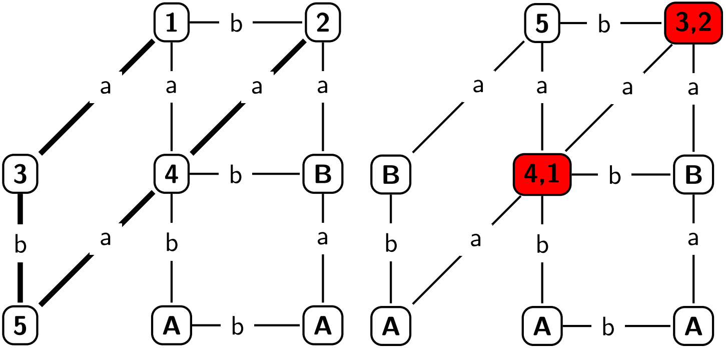
Step 3 **Step 4**

CVS Generation



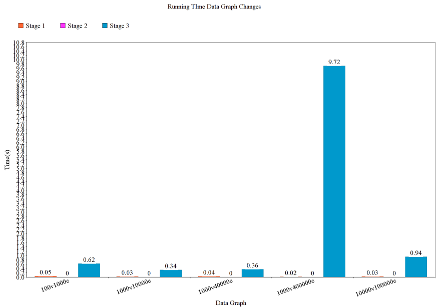
Step 5 **Step 6**

CVS Generation



Step 7 **Step 8**

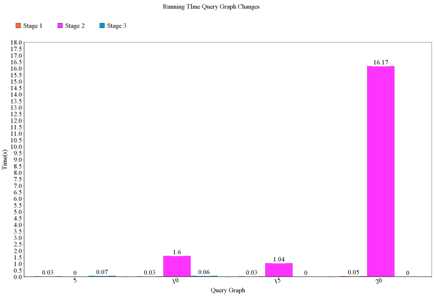
Experiments - Data Graph Size varies



Inferences

- Density of data graph is crucial

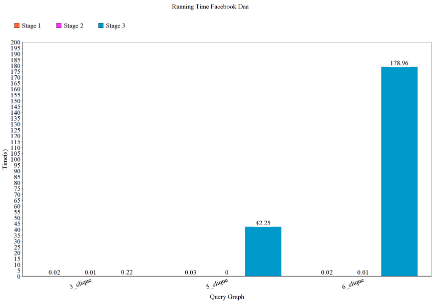
Experiments - Query Graph Size varies



Inferences

- Query graph size affects stage 2

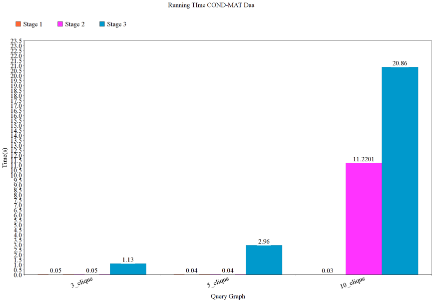
Experiments - Facebook Data



Inferences

- 4039 nodes and 88234 edges.Highly Dense.

Experiments - Condense Matter collaboration network Dataset

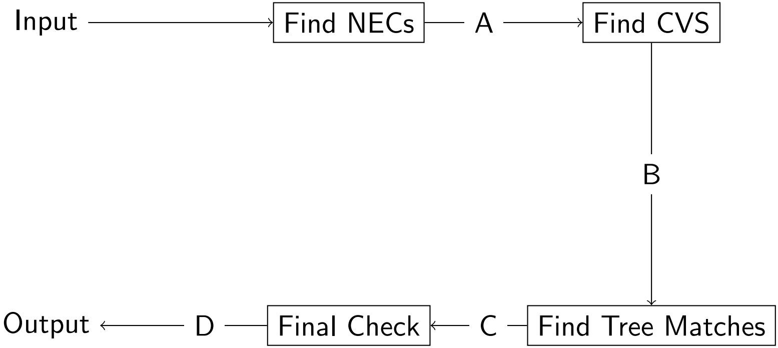


Inferences

- 23133 nodes and 93497 edges.

Dynamic Queries

Intermediate Results



Queries

	Query Add Edge	Query Remove edge	Query changed
Data Add Edge	Dificult	Difficult	Easy
Data Remove Edge	Easy	Difficult	Easy
Data Unchanged	Easy	Difficult	Static

Trivial Cases

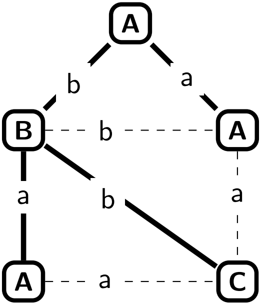
Adding Edge to Query Graph

1. need to check only the previous answers

Deleting Edge from Data Graph

1. need to check only the previous answers

Delete Edge in Query Graph



Tree in Query Grpah

1. Delete tree edge
2. Delete non-tree edge

Deleting Edge from Query Graph

Deleting a non-tree edge

1. need to go through all the tree matching(C)

Deleting a tree edge

1. All the nodes in the path from u to root(parent,grand-parent,.. of u) should recalculate the CVS

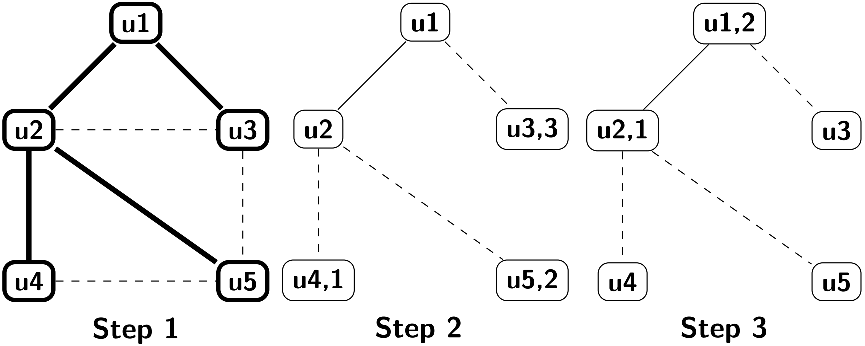
Algorithm-Deleting a tree edge

Algorithm 2 Dynamic tree edge deletion of thread t

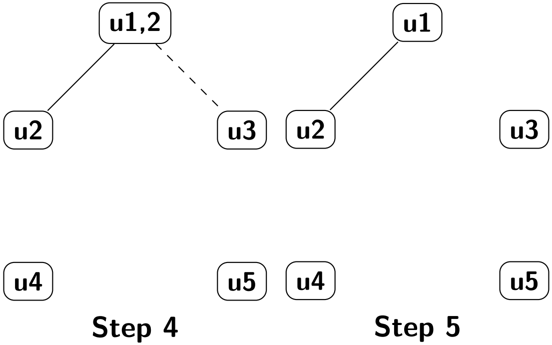
Input: Data Graph D,Query Graph Q,Delete u-v(u is parent of v).
Output: CVS updates.

1. w=v
2. for each parent of w(till root)
    1. mark w for t
3. for each parent of w(till root)
    1. if mark at w is t, acquire lock for w
4. for each parent of w(till root)
    1. if mark at w is t and able to acquire locks for all child of w
    2. Recompute NEC of w
    3. if not a previously computed NEC then
    4. C(w)=*FindCandidates*(w,D) update
    5. Release all locks

Deleting Edge in Query Graph



Deleting Edge in Query Graph



Adding Edge in Data Graph

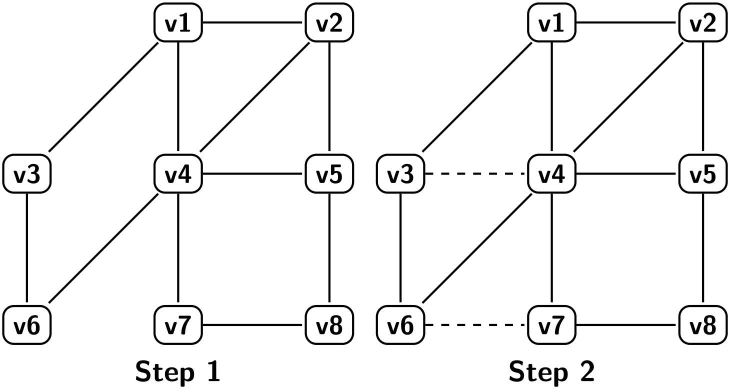
Algorithm 3 Dynamic data edge addition of thread t

Input: Data Graph D,Query Graph Q,Delete u-v(u is parent of v).

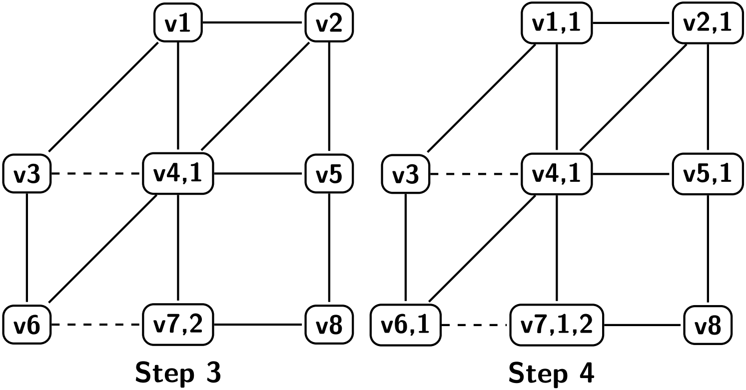
Output: CVS updates.

1. w=v (for u also)
2. for each child of w(till $|Q|$ length)
    1. if acquire locks for all child of w
    2. for each NEC's x
    3. C(x)=*FindCandidates*(x,D) update
    4. Release all locks

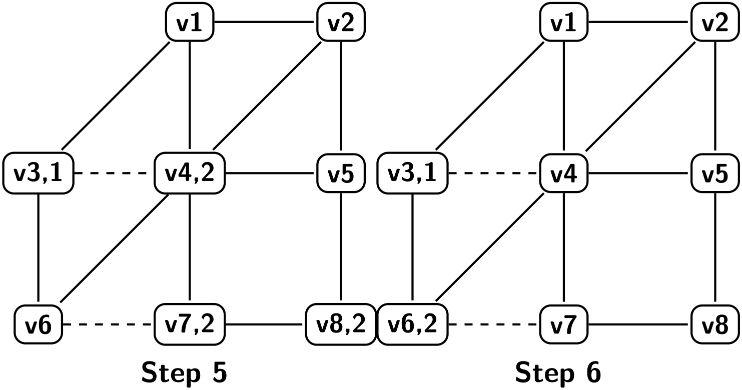
Adding Edge in Data Graph



Adding Edge in Data Graph



Adding Edge in Data Graph



Parallel Execution

How??

1. Trivial cases are working on D
2. Other cases are working on A and B

Conclusion

Conclusion

1. We studied the Sub-graph Isomorphism problem and the state-of-art algorithms. We came to know they are using different pruning techniques to avoid the false candidates as early as possible.
2. FindTreeMatch should be run at each output stage which is the costlier of all stages. Running a subgraph isomorphism solution after applying all edge updates becomes equally fast as the dynamic version

Thank You