THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Efficient GPU-Based Continuous Subgraph Matching on Batch Updates

Guanghua Li[1⋆], Xibo Sun[2⋆], Qiong Luo[2,1], and Lijun Chang[3]

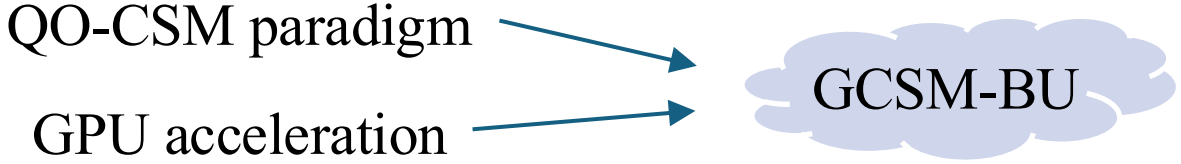
[1] The Hong Kong University of Science and Technology (Guangzhou), China
gli945@connect.hkust-gz.edu.cn
[2] The Hong Kong University of Science and Technology, Hong Kong SAR
xsunax@connect.ust.hk, luo@cse.ust.hk
[3] The University of Sydney, Sydney, Australia
lijun.chang@sydney.edu.au

Presenter: Guanghua Li

Overview

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

- Continuous subgraph matching (CSM) algorithms process edge updates (insertions and deletions) and find new matches of the query graph that either appear or disappear as a result of each update.
- Existing update-oriented CSM (UO-CSM) algorithms effectively handles single updates. However, directly applying it to a batch of updates can lead to redundant computations and duplicate results.
- We propose a novel query-oriented CSM algorithm (QO-CSM), which avoids duplicate matches and ensures consistent results on dynamic graphs with batch updates.
- Building on this paradigm, we develop a GPU-based CSM algorithm, GCSM-BU, which outperforms state-of-the-art CPU-based CSM methods and existing GPU-based UO-CSM methods.

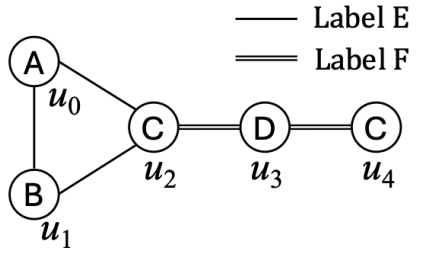


Continuous Subgraph Matching

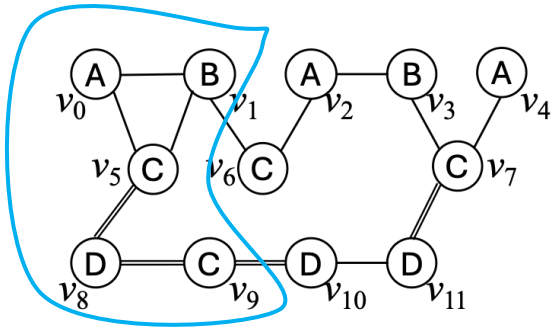
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Subgraph matching (SM) finds all matches of a query graph Q in a data graph G. The set of all matches is denoted by M.

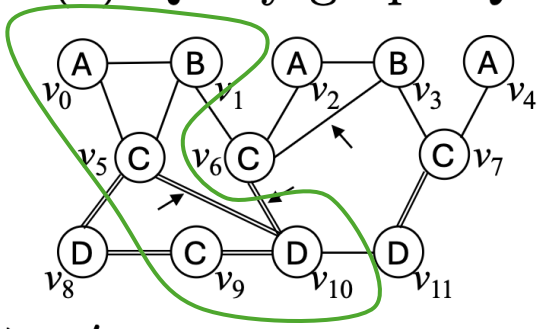
Continuous subgraph matching (CSM) algorithms find new matches of the query graph that either appear or disappear as a result of each edge update (insertion and deletion) .



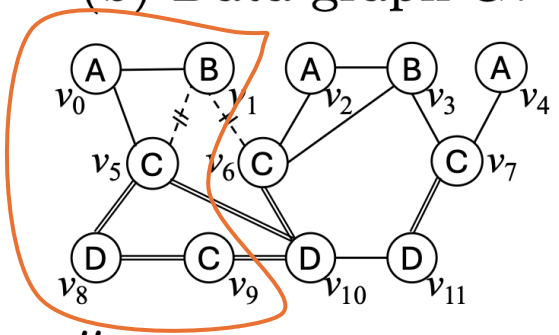
(a) Query graph Q.



(b) Data graph G.



(c) G': Insert edges into G.

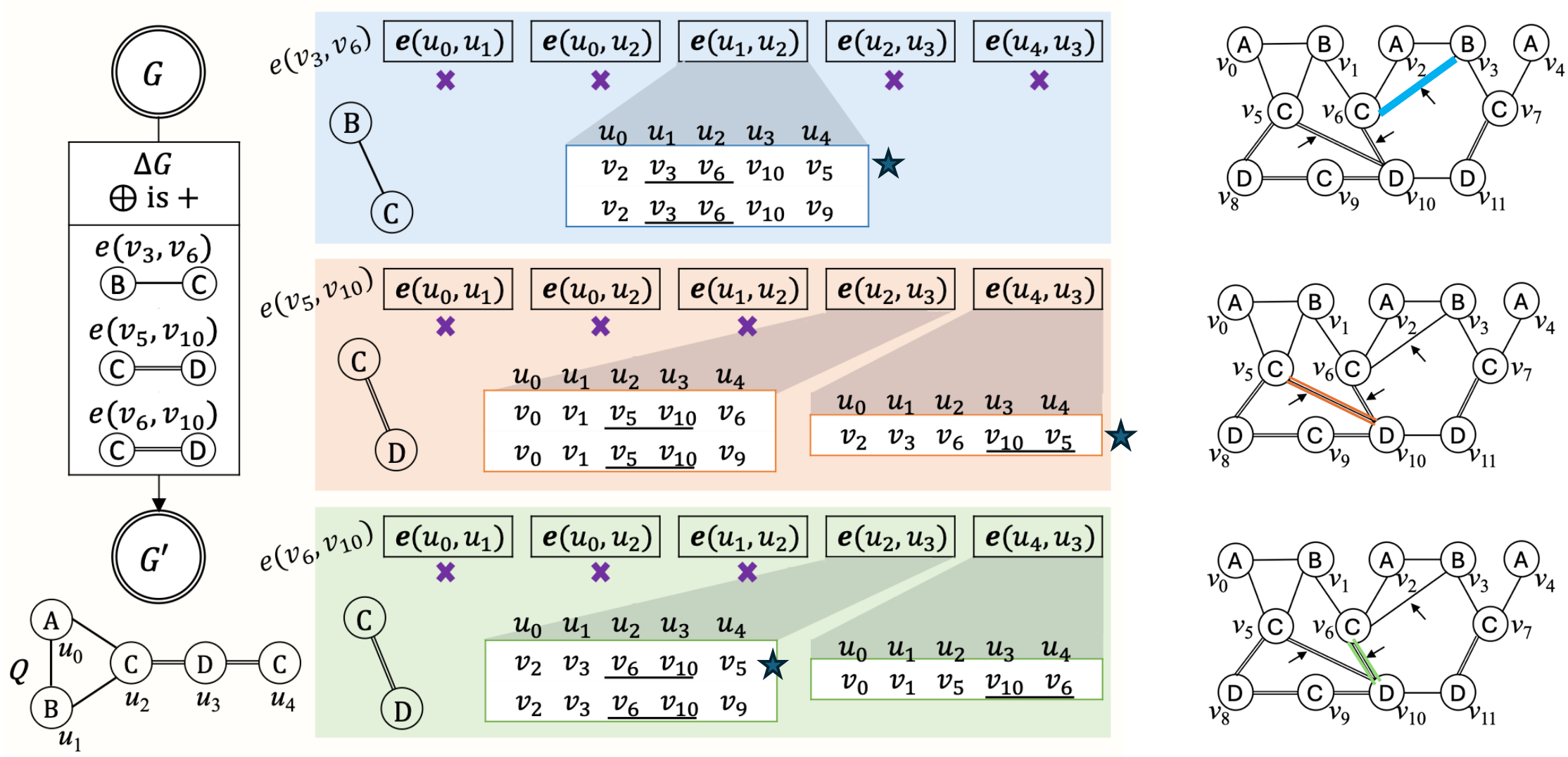


(d) G'': Delete edges from G'.

Update-Oriented v.s. Query-Oriented

Existing Update-Oriented (UO-CSM) Framework

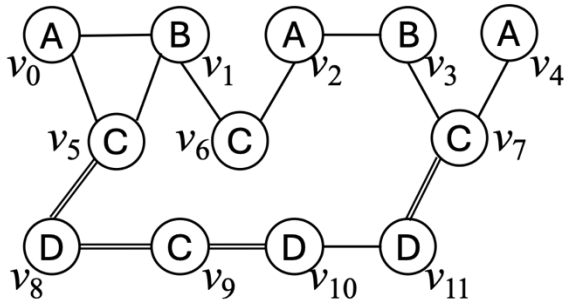
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)



Relation Representation of a Data Graph

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Data Graph



Relation Representation

G

$R_{(A, E, B)}$

A	B
v_0	v_1
v_2	v_3

$R_{(A, E, C)}$

A	C
v_0	v_5
v_2	v_6
v_4	v_7

$R_{(B, E, C)}$

B	C
v_1	v_5
v_1	v_6
v_3	v_7

$R_{(C, F, D)}$

C	D
v_5	v_8
v_7	v_{11}
v_9	v_8
v_9	v_{10}

$R_{(D, E, D)}$

D	D
v_{10}	v_{11}

G^Q

$R(u_0, u_1) \leftarrow R_{(A, E, B)}$

$R(u_0, u_2) \leftarrow R_{(A, E, C)}$

$R(u_1, u_2) \leftarrow R_{(B, E, C)}$

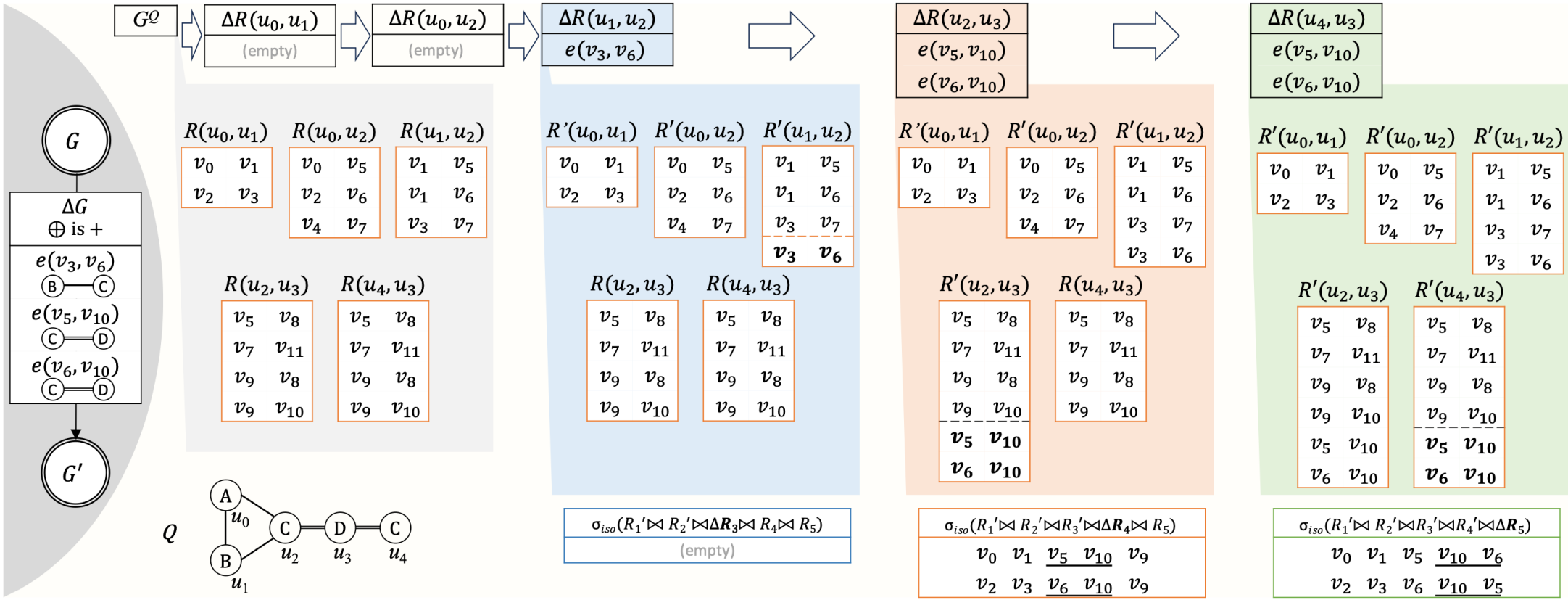
$R(u_2, u_3) \leftarrow R_{(C, F, D)}$

$R(u_4, u_3) \leftarrow R_{(C, F, D)}$

Proposed Query-Oriented QO-CSM Framework

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Main Idea: To put each update (insertion or deletion) edge into its corresponding query edge delta relation ΔR, and then process these ΔRs in order.

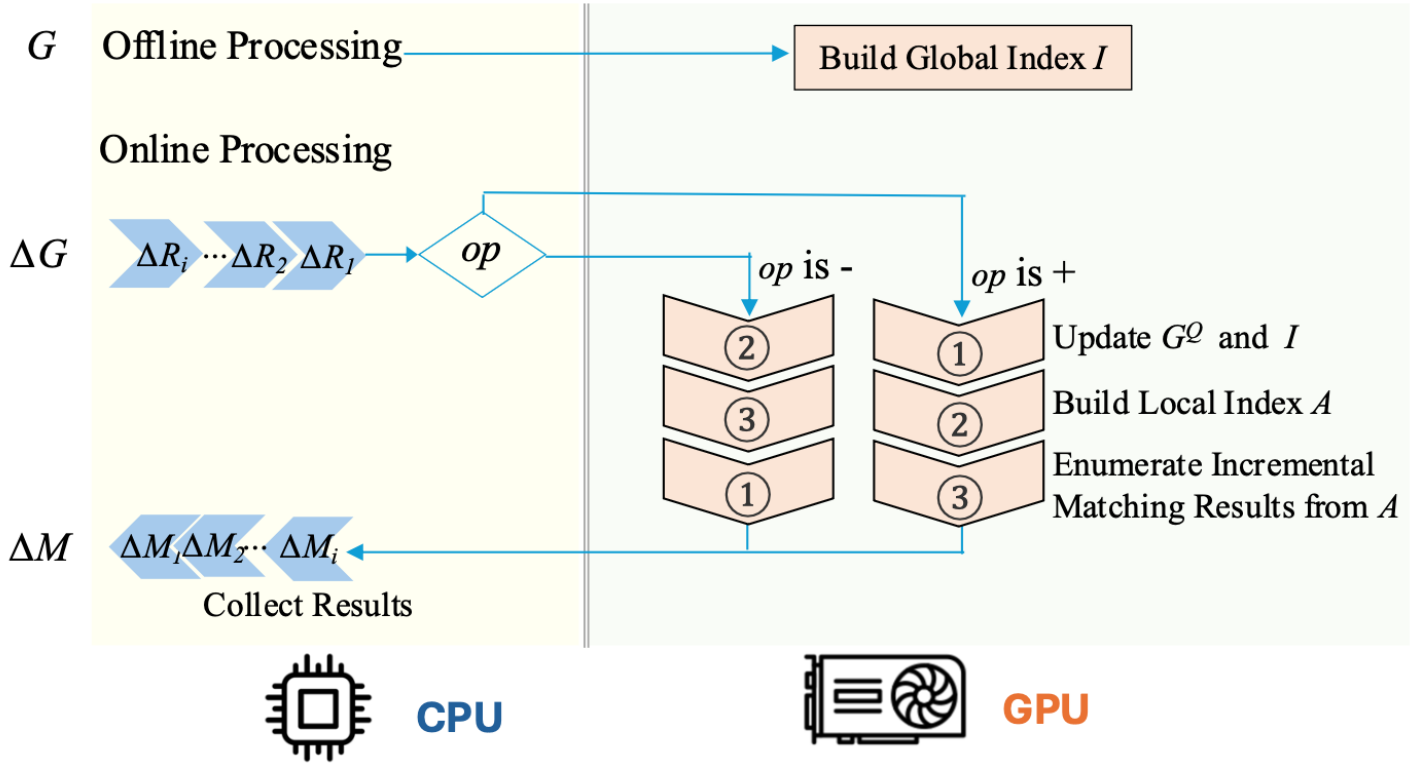


THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Implementation of GCSM-BU

Overview of GCSM-BU

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)



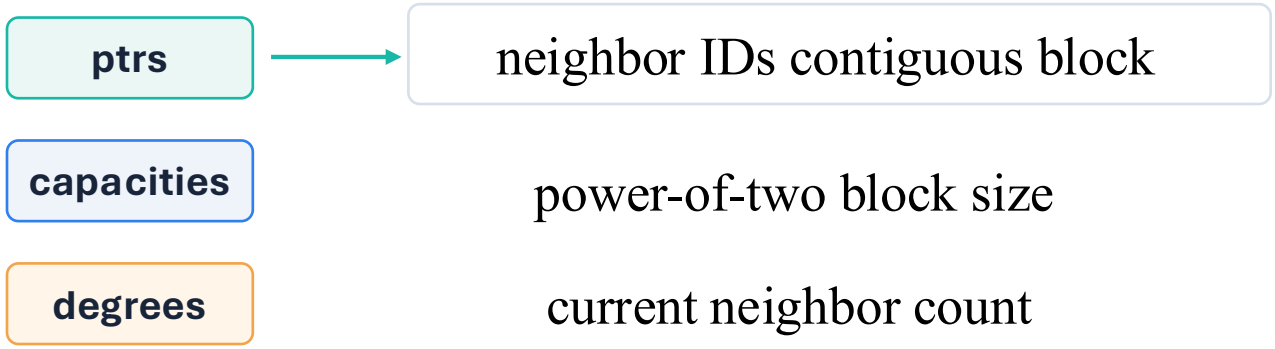
Data Structure

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Consecutive adjacency array

Each raw candidate relation is stored twice (both directions) to support fast neighbor traversal.

Per source vertex ID

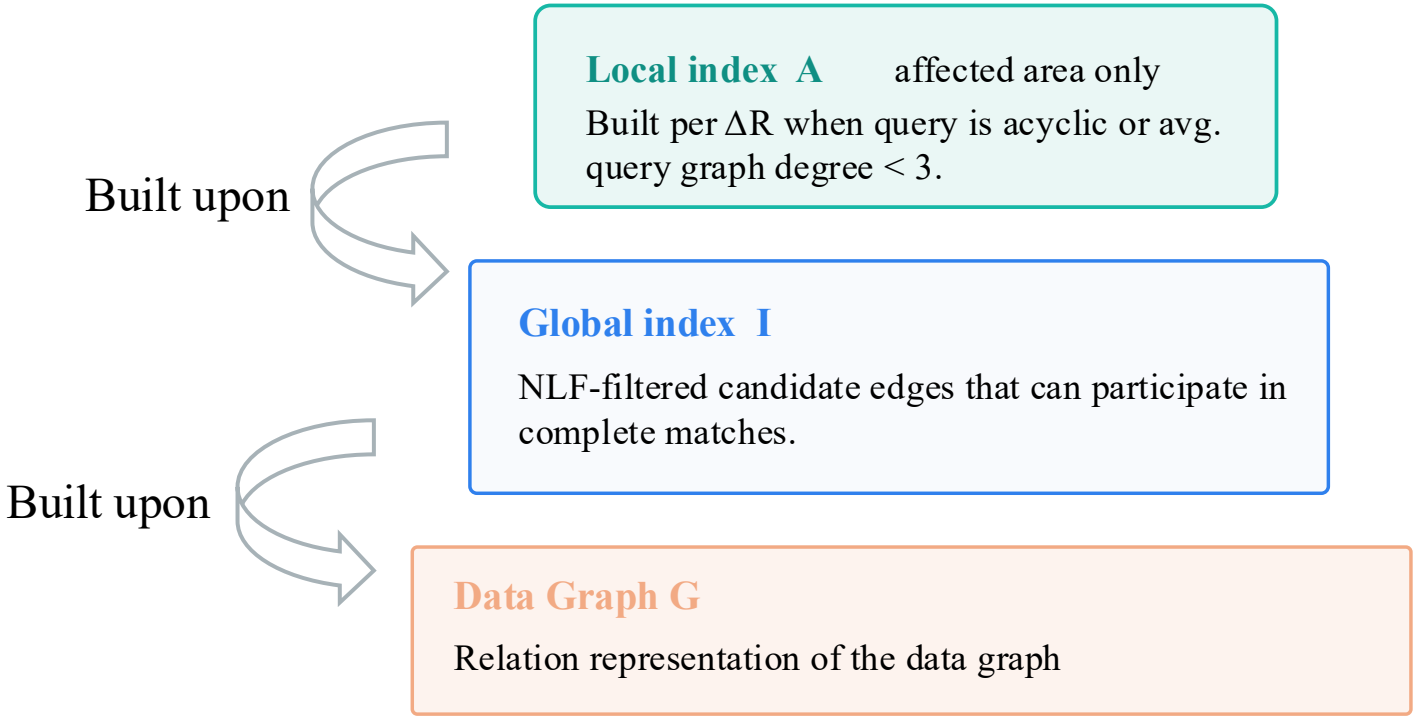


When capacity is insufficient, GCSM-BU allocates a new contiguous segment from a pre-allocated pool and updates the pointer.

Two-Level Indexing

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

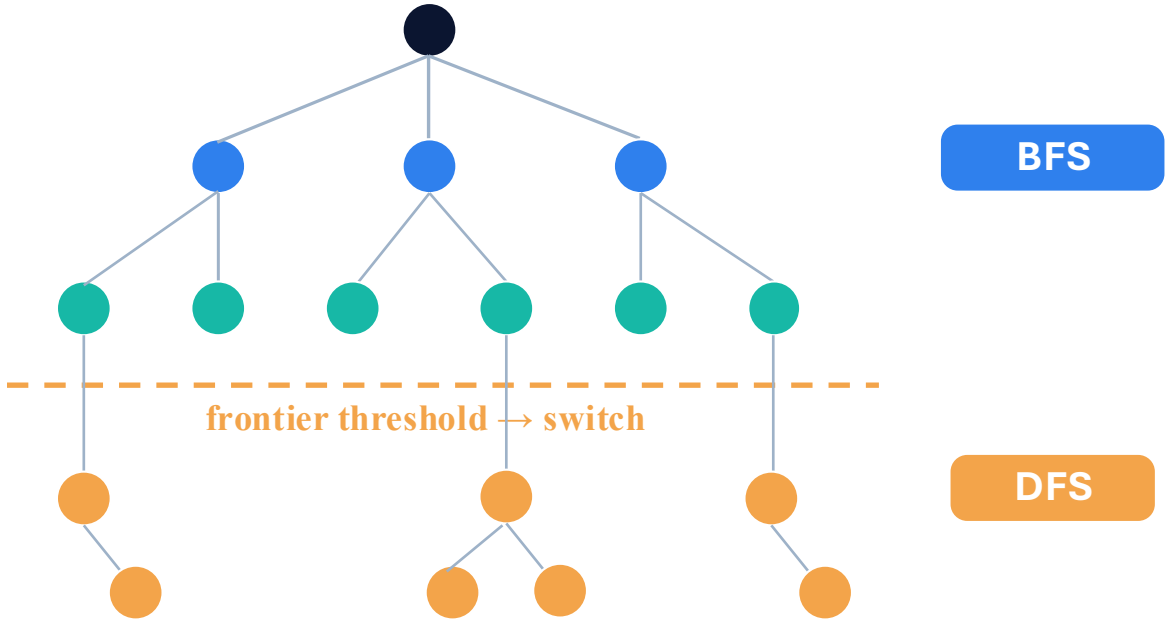
We use a persistent **global index** for pruning, and a **local index** for the subgraph region affected by the current ΔR.



Benefits: Smaller search region + flexible matching order.

Search Strategy - Dynamic Semi-BFS

Start with BFS to expose parallel work; switch to per-warp DFS once the intermediate frontier reaches a predefined threshold.

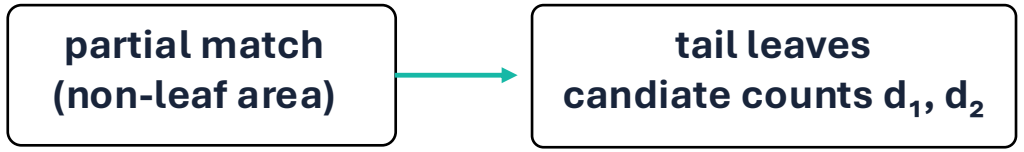


BFS: parallelism ↑ / memory ↑ DFS: memory ↓ / parallelism ↓

Search Strategy - Backtracking Flattening

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

For tail leaf vertices, replace warp-level recursive enumeration with a flat Cartesian-product launch: one GPU thread per potential leaf assignment combination.



each partial match conrresponds to $d_1 \times d_2$ threads

Each thread decodes its combination from thread ID and performs isomorphism checking.

Benefits: To avoid leaf-induced load imbalance.

Experimental Results

Experimental Setup

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Datasets

Dataset	|V|	|E|	davg	|ΣE|
LSBench	5.2M	20.3M	8.2	44
Netflow	3.1M	2.9M	2.0	7
LiveJournal	4.9M	42.9M	18.1	1
Amazon	0.4M	2.4M	12.2	1

Four graph families with different sizes, degrees, and label distributions.

Implementation: C++ / CUDA, compiled with g++ 11.4.0 and CUDA 12.8.

Hardware

Linux server
2× AMD EPYC 7302
256 GiB RAM
NVIDIA RTX 3090 (24 GiB)

Queries & updates

6 query vertices by default
Type: Tree / Sparse / Dense
100 queries / type
10% insertion rate, in one batch

Baselines

CPU: SymBi, RapidFlow, NewSP
GPU: GAMMA*

Framework control: QO-GAMMA*

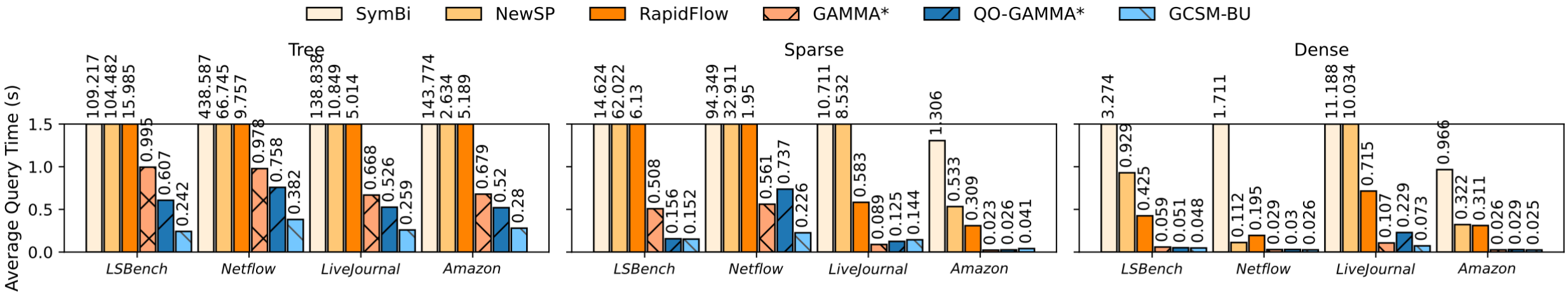
Metric

Online query time
Time limit: 0.5 h / query

Solved query = all incremental matches are found within the limit
Average time on commonly solved queries

Overall Performance

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)



Average query time of solved queries (in seconds).

- Tree queries: 111× / 25× / 20× / 9× faster than the best CPU method.

- Sparse queries: faster or on par with GPU baselines. Dense queries: up to 9× over the best CPU method and roughly on par with GPU methods.

- Comparison with GAMMA*: ≈3–4× faster on tree queries, despite our GAMMA* reimplementation already being faster than originally reported.

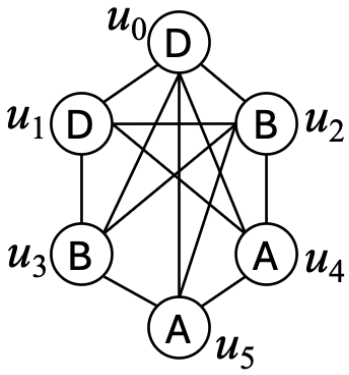
Case study — duplicate avoidance

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Data Graph:

Amazon

Query Graph:



Repetitions	Examples	#matches
7	[272717, 272709, 272713, 272710, 272716, 290853]	1
6	[16994, 390066, 5533, 16997, 48101, 394497] [16994, 390066, 16997, 5533, 48101, 394497] ...	48
5	[11, 135475, 25, 150, 22, 4245] …	332
4	[11, 626, 25, 144, 22, 4245] …	1413
3	[11, 44, 25, 144, 22, 4245] …	5231
2	[11, 44, 25, 144, 22, 34] …	13217
1	[11, 44, 25, 144, 22, 15] …	20796
Note that the query result size of GAMMA*, a DO-CSM algorithm: $70530 = 7 \times 1 + 6 \times 48 + 5 \times 332 + 4 \times 1413 + 3 \times 5231 + 2 \times 13217 + 1 \times 20796$		Sum: 41038

(Amazon, first dense query. Disable duplicate removal in UO-CSM and count the enumerated results.)

70,530 → **41,038**

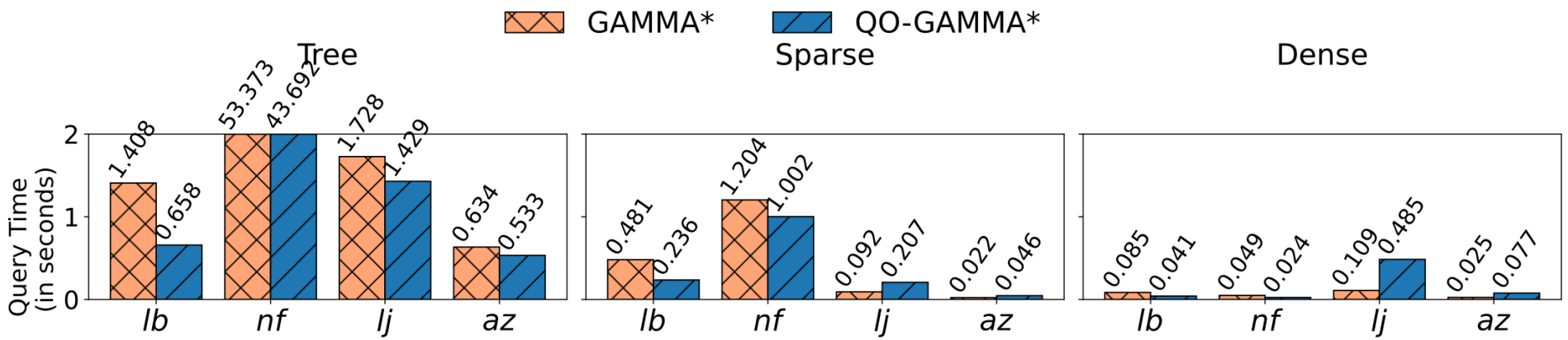
enumerated results by UO-CSM without de-dup

unique matches (QO-CSM)

(1.7× over-enumeration)

Efficiency of QO-CSM Framework

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)



These two implementations share indexing, matching order, and search strategy.
The framework is the controlled variable.

Avoiding explicit de-duplication leads to faster query processing in most cases.

Exception: on some sparse/dense LiveJournal and Amazon queries, the workload is so small that QO-CSM's repeated small ΔR updates outweigh the saved de-duplication work

Conclusion

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

- We studied the problem of continuous subgraph matching for batch updates and proposed a query-oriented CSM framework, QO-CSM, which avoids duplicate query results automatically.
- Based on QO-CSM, we implemented a GPU-based CSM algorithm, GCSM-BU,which integrates various techniques and practical optimizations.
- Experiments show that GCSM-BU significantly outperforms the latest CPU-based CSM algorithms on batch updates and achieves better overall performance than the GAMMA* variants.

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Thanks

Email: gli945@connect.hkust-gz.edu.cn

Presenter: Guanghua Li