DualSim: Parallel Subgraph Enumeration in a Massive Graph on a Single Machine

Hyeonji Kim, Juneyoung Lee, Sourav S. Bhowmick, Wook-Shin Han, JeongHoon Lee, Seongyun Ko, Moath H.A. Jarrah

Presented by: Bishesh Khadka
MIT 6.886 - Graph Analytics

Motivation

- Subgraph enumeration
  - Network motif discovery
  - Graphlet kernel computation
  - Subgraph frequency

Existing Work

- MapReduce & Distributed Graph Engines
  - Redundant work-- duplicated edges
  - Multiway join memory intensive
- Shao et al.
  - PSGL
    - No multiway join
- All approaches have serious performance issues

Problem Statement

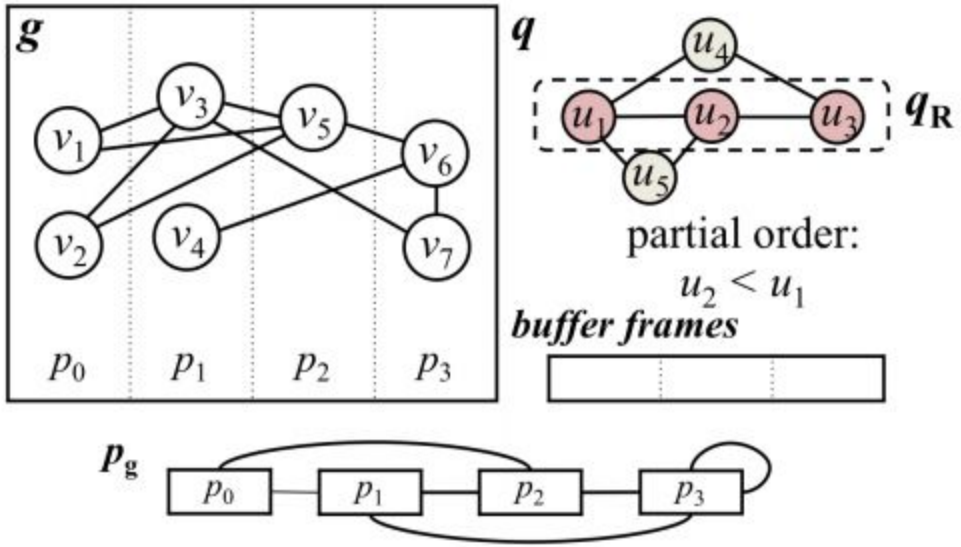
- Can subgraph enumeration be done disk-based, on a single machine in a way that is scalable and efficient?
- Motivating Principles
  - Existing methods fail due to exponential partial solutions
  - Disk access one of costliest bottlenecks
  - CPU stall also notable bottleneck

Key Contributions

- DualSim does not maintain explosive partials
- Dual Approach
  - V-group sequence
  - V-group forest
  - Further optimizations
- Red Black Ivory query graph transformation
- Evaluation vs. state-of-the-art subgraph enumeration techniques

Dual Approach

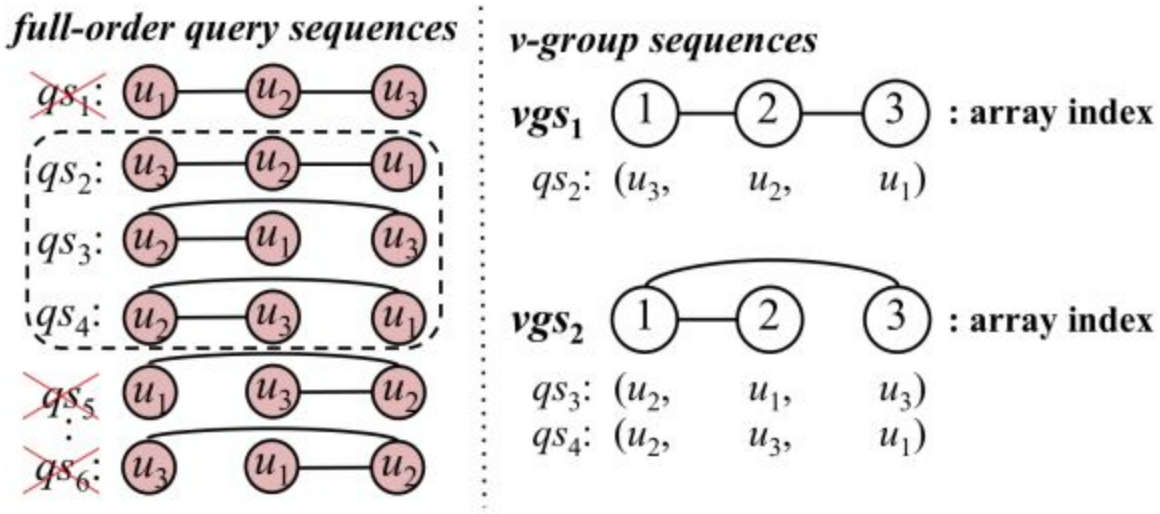
- Given query graph q, data graph g, page graph p



(a) Data graph g, query graph q, and page graph p_g.

Dual Approach

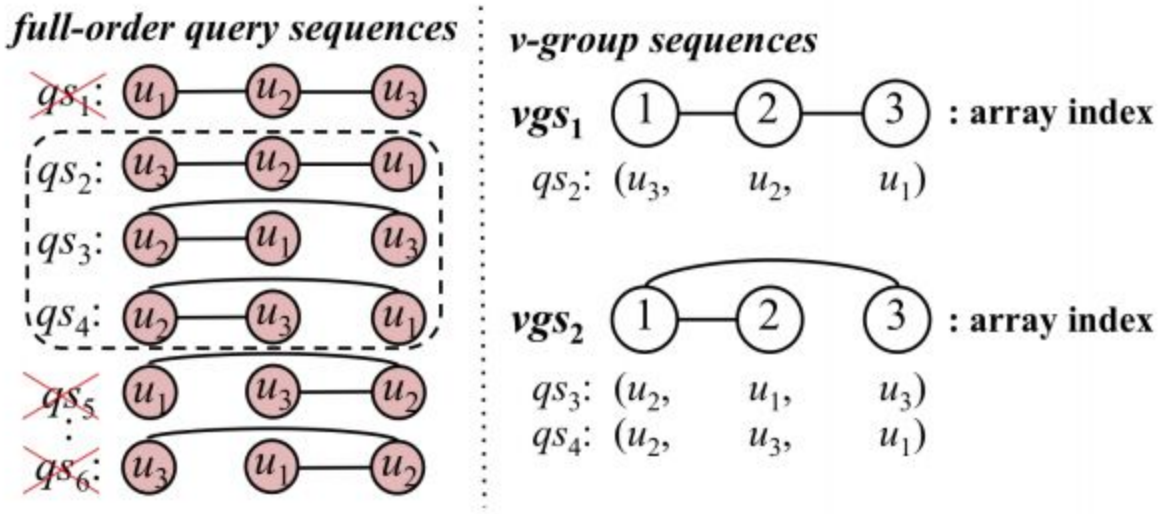
- Given query graph q, data graph g, page graph p
- Enumerate all possible query sequences
- Full-order query sequences
  - Each matches an ordered data seq. ⇒ fixes data seq.



(b) Full-order query sequences and *v*-group sequences.

Dual Approach

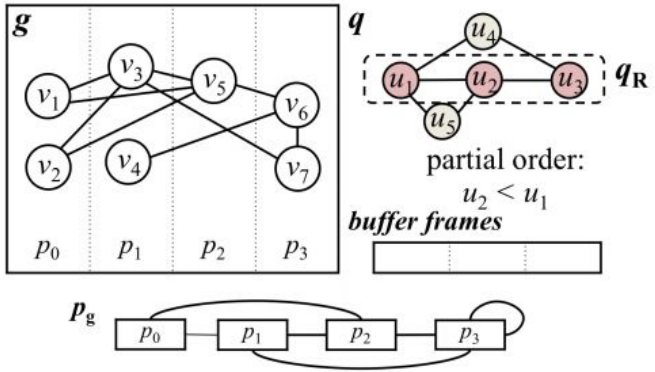
- Given query graph q, data graph g, page graph p
- Enumerate all possible query sequences
- Full-order query sequences
  - Each matches an ordered data seq. ⇒ fixes data seq.
- Prune FOQS with given partial orders
- Group FOQS into v-group sequences based on topology



(b) Full-order query sequences and *v*-group sequences.

Dual Approach

- Given query graph q, data graph g, page graph p
- Enumerate all possible query sequences
- Full-order query sequences
  - Each matches an ordered data seq. ⇒ fixes data seq.
- Prune FOQS with given partial orders
- Group FOQS into v-group sequences based on topology
- Traverse graph stored in pages based on v-group sequences

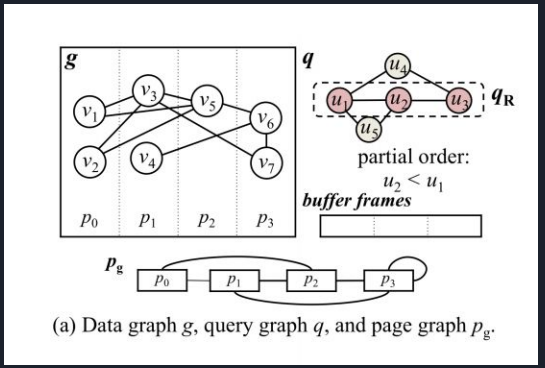


(a) Data graph g, query graph q, and page graph p_g.

page sequences	*matching v-group sequences*
①. (p_0, p_1, p_1)	vgs_2
②. (p_0, p_1, p_2)	vgs_1, vgs_2
③. (p_0, p_1, p_3)	vgs_1
④. (p_0, p_2, p_2)	vgs_2
⑤. (p_0, p_2, p_3)	vgs_1
⑥. (p_1, p_2, p_2)	vgs_2
⑦. (p_1, p_2, p_3)	vgs_1, vgs_2
⑧. (p_1, p_3, p_3)	vgs_2
⑨. (p_2, p_3, p_3)	vgs_1, vgs_2
⑩. (p_3, p_3, p_3)	vgs_2

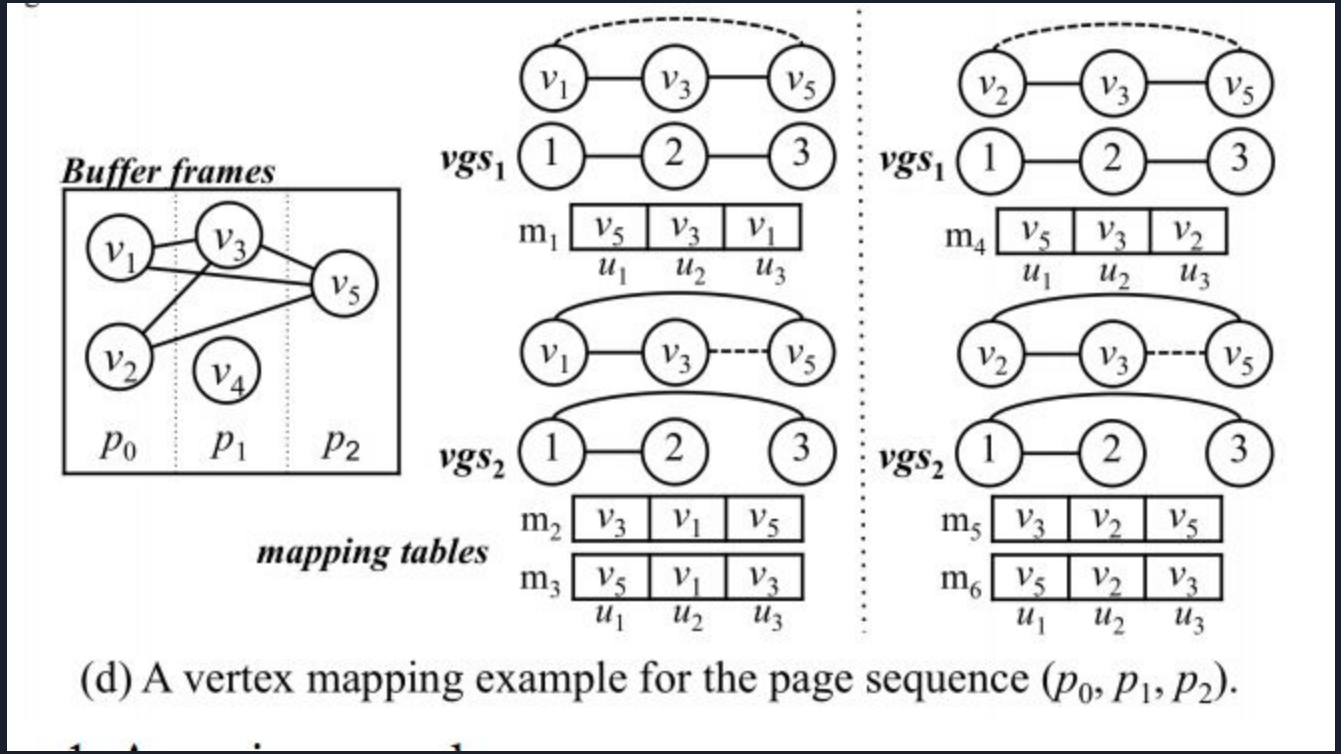
(c) A page mapping example.

Dual Approach cont'd



(a) Data graph g, query graph q, and page graph p_g.

- For each valid mapping
  - enumerate all FOQS in v-group sequence
  - Generate data mappings
  - ②. (p_0, p_1, p_2) ⋮ vgs_1, vgs_2
    - Vertex level match: (v1, v3, v5)
    - For Vgs1: only 1 FOQS: (u3, u2, u1)
      - Solution: {(u3, v1), (u2, v3), (u1, v5)}
      - (v2, v3, v5) also valid match



(d) A vertex mapping example for the page sequence (p_0, p_1, p_2).

RBI Graph

- Idea: Disk reads minimized if we use minimum number of query vertices during graph traversal
- Colored vertices:
  - Red: Data must retrieve adj. List
  - Ivory:
    - Is adj to m > 1 reds
    - m-way intersection
  - Black:
    - Is adj to m = 1 red
    - Scan red's adj list
- MCVC is colored red
  - Reachability
  - NP hard but |Vq| small enough for approx.

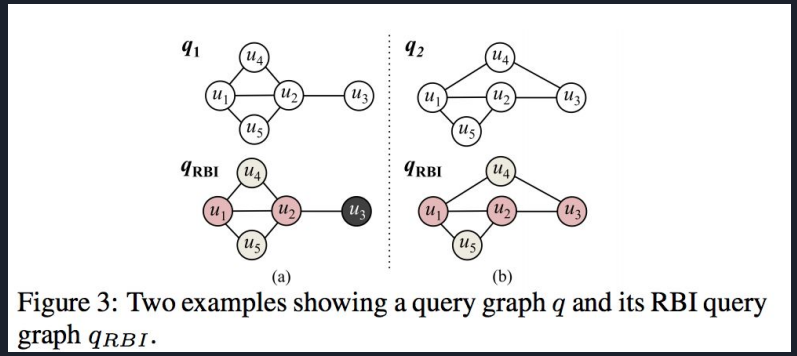


Figure 3: Two examples showing a query graph q and its RBI query graph q_{RBI}.

DualSim Algorithm

Algorithm 1. DUALSIM

Input: A data graph g, A query graph q

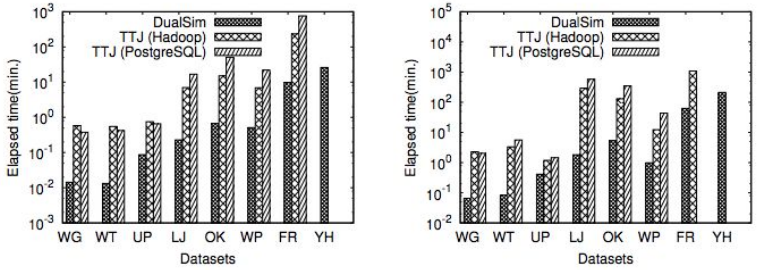
/* 1. Preparation step */

1: $PO \leftarrow$ FINDPARTIALORDERS(q);
2: $\langle q_{RBI}, q_R \rangle \leftarrow$ GENERATERBIQUERYGRAPH(q, PO);
3: $\{vgs_i\} \leftarrow$ FINDVGROUPSEQUENCES(q_R, PO);
4: $mo_g \leftarrow$ FINDGLOBALMATCHINGORDER($\{vgs_i\}$));
5: $\{vgf_i\} \leftarrow$ BUILDVGROUPFORESTS($\{vgs_i\}$, mo_g);

/* 2. Execution step */

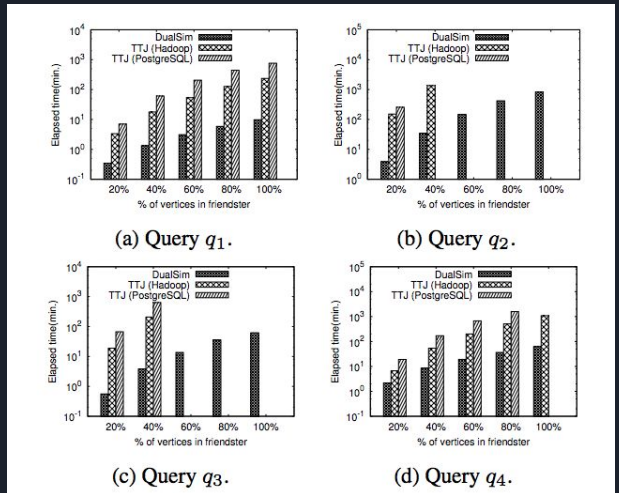
6: INITIALIZECANDIDATESEQUENCES(root nodes in $\{vgf_i\}$);
7: **foreach** (merged vertex/page window (mvw_1, mpw_1) from $\{vgf_i[1]\}$) **do**
8: | **foreach** (page id $pid \in mpw_1$) **do**
9: | | AsyncRead(pid, COMPUTECANDIDATESEQUENCES(pid, $\{cvw_{i,1}\}$, all child nodes of $\{vgf_i[1]\}$);
10: | **end**
11: | wait until COMPUTECANDIDATESEQUENCES executions are finished
12: | $level \leftarrow 2$;
13: | DELEGATEEXTERNALSUBGRAPHENUMERATION($level$, q_{RBI}, $\{vgs_i\}$, $\{vgf_i\}$, $\{mvw_j\}$, $\{mpw_j\}$);
14: | INTERNALSUBGRAPHENUMERATION(mvw_1, mpw_1);
15: | UNPINPAGES(mpw_1);
16: | CLEARCANDIDATESEQUENCES(the children of $\{vgf_i[1]\}$);
17: **end**

Evaluation - Single Machine



(a) Query q_1. (b) Query q_4.

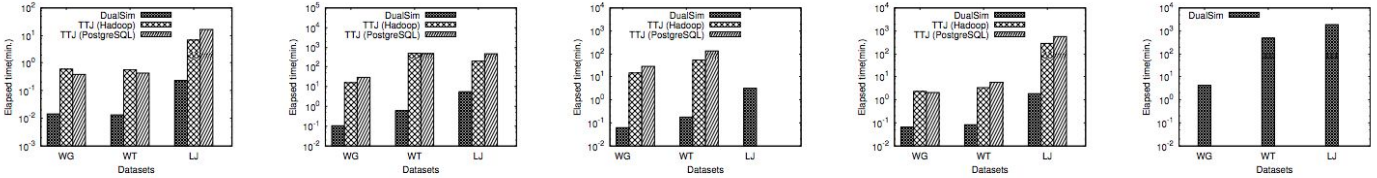
Figure 10: Varying datasets in a single machine.



(a) Query q_1. (b) Query q_2.

(c) Query q_3. (d) Query q_4.

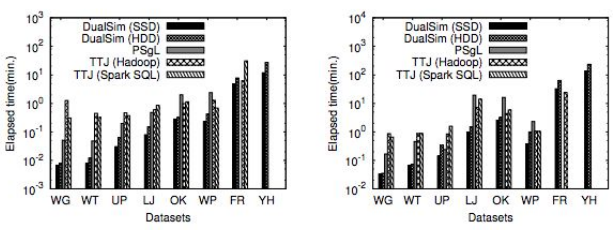
Figure 12: Varying graph size in a single machine.



(a) Query q_1. (b) Query q_2. (c) Query q_3. (d) Query q_4. (e) Query q_5.

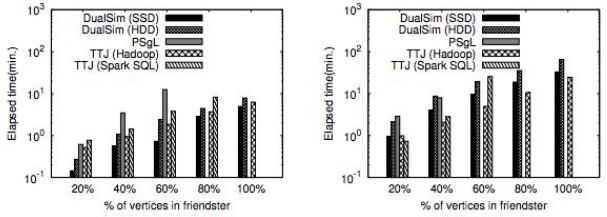
Figure 11: Varying queries in a single machine.

Evaluation - Cluster



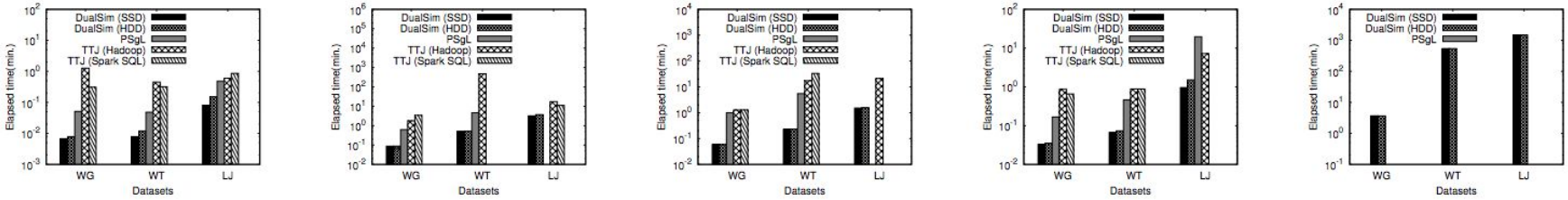
(a) Query q_1. (b) Query q_4.

Figure 13: Varying datasets in a cluster.



(a) Query q_1. (b) Query q_4.

Figure 15: Varying graph size in a cluster.



(a) Query q_1. (b) Query q_2. (c) Query q_3. (d) Query q_4. (e) Query q_5.

Figure 14: Varying queries in a cluster.

Conclusion

- Significant CPU processing reduction due to dual approach's traversal
- Disk I/O reduction
- DualSim out performs existing solutions in both single machine and distributed environment for subgraph enumeration

References

- Hyeonji Kim, Juneyoung Lee, Sourav S. Bhowmick, Wook-Shin Han, JeongHoon Lee, Seongyun Ko, Moath H.A. Jarrah
  DualSim: Parallel Subgraph Enumeration in a Massive Graph on a Single Machine