Fast Parallel Subgraph Matching on the GPU

Leyuan Wang, Yangzihao Wang, John D. Owens
University of California, Davis, CA, USA

Objectives

Graphs have been used to model interactions between entities in a broad spectrum of applications. Subgraph patterns help to understand the underlying structure of these graphs. We address the problem of Subgraph Matching (SM), finding all embeddings of a query graph q in a large data graph g. SM can be easily modified to handle graph homomorphism which is the RDF pattern matching semantics, by just removing the injectivity constraint. RDF is primarily designed for building the Semantic web and has been widely adopted in database and data mining communities. High-performance SM can be tamed for efficient RDF processing (e.g. SPARQL queries) and could potentially outperform the representative RDF engines.

Challenges

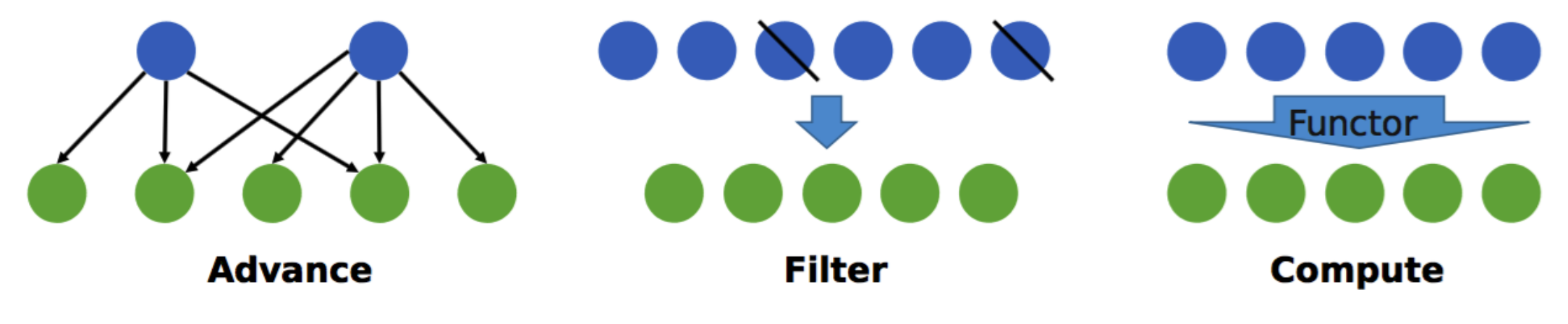
- GPU operations are based on warps (which are groups of threads to be executed in single-instruction-multiple-data fashion), so different execution paths generated by backtracking algorithms may cause a warp divergence problem.
- Irregular memory access patterns cause memory accesses to become uncoalesced, which degrades GPU performance.

Our Solution

Our method follows a filtering-and-joining procedure and is implemented using the Gunrock programming model [3].
Filtering Phase. Prune away candidate vertices that cannot contribute to the final solutions using the massively parallel *advance* and *filter* operations in Gunrock.
Joining Phase. We do the actual joining only when the candidate edges satisfy the intersection rules in order to reduce the number of intermediate results and consequently, the amount of required memory.

Gunrock Programming Model

- **Advance:** generates a new frontier (in green) via visiting the neighbor vertices/edges in the current frontier (in blue).
- **Filter:** removes elements from a frontier via validation test.
- **Compute:** user-defined vertex/edge-centric computations that run in parallel, can be combined with advance or filter.



Parallel Subgraph Matching

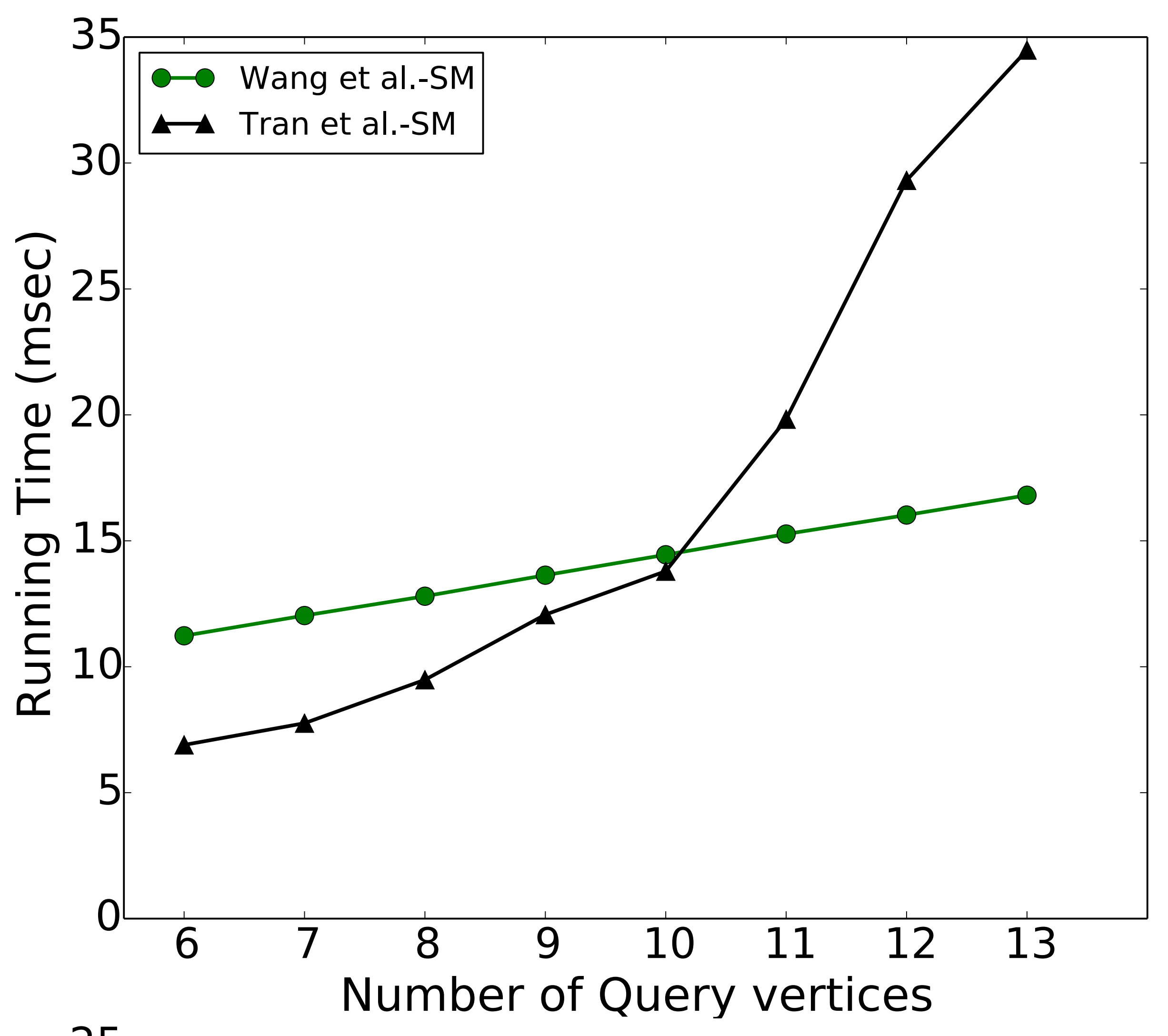
Input: Query Graph Q, Data Graph G.
Output: Number of matched query graphs *count* and listings of all occurrences *outputlist*.

```
procedure INITIALIZE_CANDIDATE_SET($Q, G$)
    ADVANCE($G$)  ▷ Fill c_set based on node label and degree
end procedure

procedure COLLECT_CANDIDATE_EDGES($G, Q, c\_set$)
    ADVANCE($G$) ▷ Label candidate edges with query_edge_id
    FILTER($G$)                    ▷ Collect candidate edges
    return $ELIST$
end procedure

procedure JOIN($ELIST, intersection\_rule$)
    parallel for each candidate edge combination $\{e_i \ldots e_j\}$
    if $\{e_i \ldots e_j\}$ satisfy $intersection\_rules$ then
        Write $\{e_i \ldots e_j\}$ to $outputlist$
        Add 1 to $count$
    end if
    return $count$, $outputlist$
end procedure
```

Experiments & Results



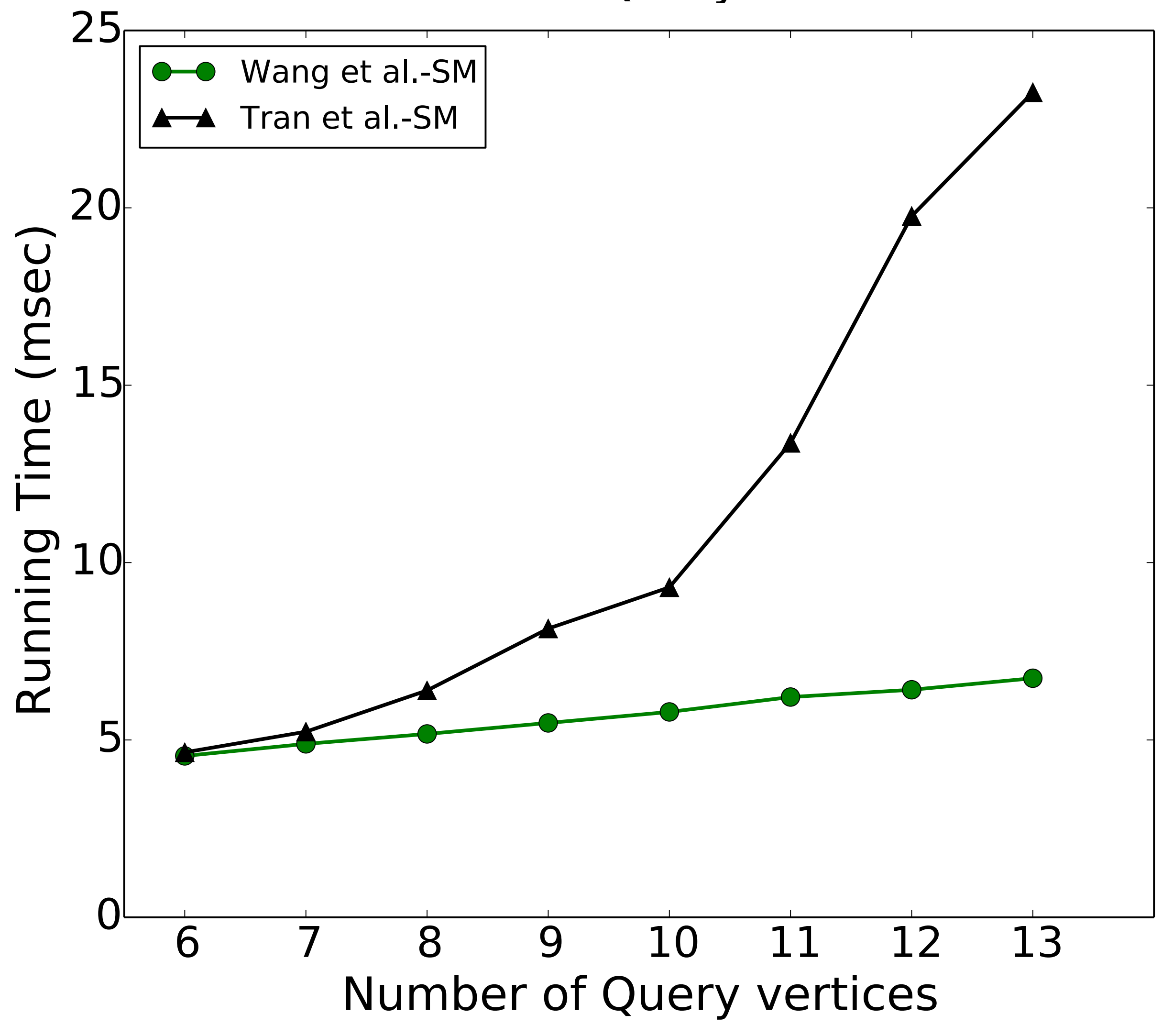


Figure: Speedups on Gowalla (top) and Enron (bottom) datasets of both our PSM and a state-of-the-art GPU implementation by Tran et al. [1].

Dataset	PSM (ms)	TC (ms)	Speedup
coAuthorsCiteseer	29.47	275.49	9.35
coPapersDBLP	387.15	100654.05	259.98
road_central	404.43	3254	8.05
kron_n17	160.06	578558.69	3614.62
delaunay_n20	113.15	544.36	4.81

Table: Performance comparisons of our algorithm (PSM) with CPU triangle counting (TC) [2] in both elapsed time (milli seconds) and speedup.

Conclusion

In this work, we introduce an efficient method which takes advantage of GPU parallelism to deal with large-scale subgraph matching problem. We achieve up to 3.5× speedup against previous state-of-the-art method on the GPU. We also achieve significant speedups compared with specific triangle counting algorithm implementations. Our method, despite its memory bound, shows good performance on mesh like graphs that contain a large part of leaf nodes.
We note that joining order selection and reducing intermediate results are considerably more challenging. We hope that the most challenging ones will serve as interesting future work developing frameworks for those particular methodologies.

Dataset Description

Dataset	Vertices	Edges	Max Degree	Type
coAuthorsCiteseer	227,320	1,628,268	1372	rs
coPapersDBLP	540,486	30,491,458	3299	rs
road_central	14,081,816	33,866,826	8	rm
kron_n17	131,072	10,227,970	28804	rs
delaunay_n20	1,048,576	6,291,372	17	rm
enron	69244	549216	1391	rs
gowalla	196578	1900654	394	rs

Table: Dataset Description Table. The edge number shown is the number of directed edges when the graphs are treated as undirected graphs and de-duplicate the redundant edges. Graph types are: r: real-world, s: scale-free, and m: mesh-like.

References

[1] H.-N. Tran, J.-j. Kim, and B. He.
Fast subgraph matching on large graphs using graphics processors.
In M. Renz, C. Shahabi, X. Zhou, and A. M. Cheema, editors, *Proceedings of the 20th International Conference on Database Systems for Advanced Applications*, DASFAA 2015, pages 299–315. Springer International Publishing, Cham, Apr. 2015.

[2] L. Wang, Y. Wang, C. Yang, and J. D. Owens.
A comparative study on exact triangle counting algorithms on the GPU.
In *Proceedings of the 1st High Performance Graph Processing Workshop*, HPGP '16, May 2016.

[3] Y. Wang, A. Davidson, Y. Pan, Y. Wu, A. Riffel, and J. D. Owens.
Gunrock: A high-performance graph processing library on the GPU.
In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP 2016, pages 11:1–11:12, Mar. 2016.

Contact Information

Email: leywang@ucdavis.edu