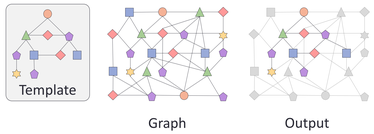
Pattern Matching on Massive Metadata Graphs at Scale

Tahsin Reza
Advised by Professor Matei Ripeanu
Electrical and Computer Engineering
University of British Columbia
netsyslab.ece.ubc.ca

1. Problem Overview

Pattern matching in graphs, that is, finding subgraphs that *match* a small *template graph* within a large *background graph*, is fundamental to graph analysis and has applications in multiple areas such as social network analysis, bioinformatics and information mining.



$G = (V,E)$ – background graph
V – set of vertices, E – set of edges
$d(v)$ – degree of a vertex $v \in V$
$l(v)$ – *label of a vertex v*
$G_0 = (V_0,E_0)$ – template subgraph
$G^* = (V^*,E^*)$ – pruned / solution graph

Hypothesis

Systematic *graph pruning* is an effective building block toward enabling scalable pattern matching on massive metadata graphs using distributed memory systems

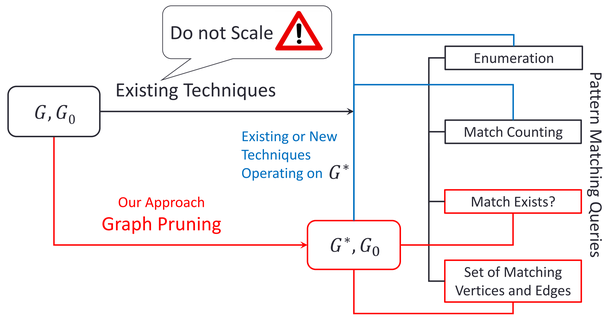
Challenges

- ❑ NP in the general case [1].
- ❑ Susceptible to combinatorial explosion of the algorithm state.
- ❑ Existing techniques are difficult to parallelize and scale in a distributed setting.
- ❑ Join operations are expensive and subgraph indexing is infeasible for billion-edge graphs.

Opportunities / Intuitions

- ❑ Decoupling pruning from enumeration.
- ❑ Low complexity heuristics are possible for pruning.
- ❑ Substructures of the template can be verified without relying on expensive join operations.
- ❑ Vertex-centric formulation for pruning algorithms are possible to harness distributed frameworks like HavoqGT, GraphLab and Giraph.

2. A Graph Pruning Path for Scalable Pattern Matching

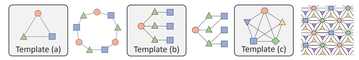


High-level Pseudocode of the Main Pruning Loop

```
Input: background graph G(V,E), template G0(V0,E0)
Output: the solution subgraph G*(V*,E*)
Generate non-local constraint set K0 from G0(V0,E0)
G* ← LOCAL_CONSTRAINT_CHECKING(G,G0)
while K0 is not empty do
  pick and remove next constraint C0 from K0
  G* ← NON_LOCAL_CONSTRAINT_CHECKING(G*,G0,C0)
  if any vertex has been eliminated then
    G* ← LOCAL_CONSTRAINT_CHECKING(G*,G0)
```

Local Constraint Checking (LCC)

- ❑ Iteratively verifies if one-hop neighbors meet template constraints.
- ❑ Performs (i) Vertex and (ii) Edge elimination.
- ❑ Polynomial-time algorithm [4, 5].

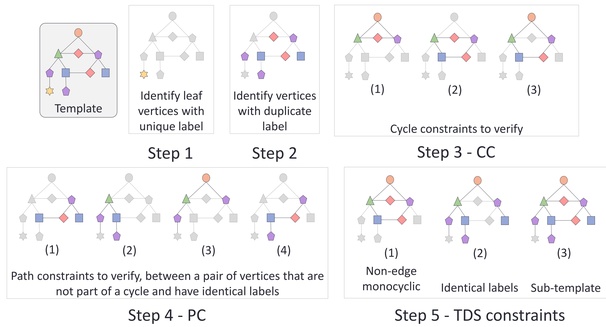


Examples of templates and substructures in the background graph that justify the need for NLCC. For these examples, LCC is not sufficient to guarantee 100% precision. Template (a) requires cycle checking where as Template (b) and (c) require NLCC-TDS.

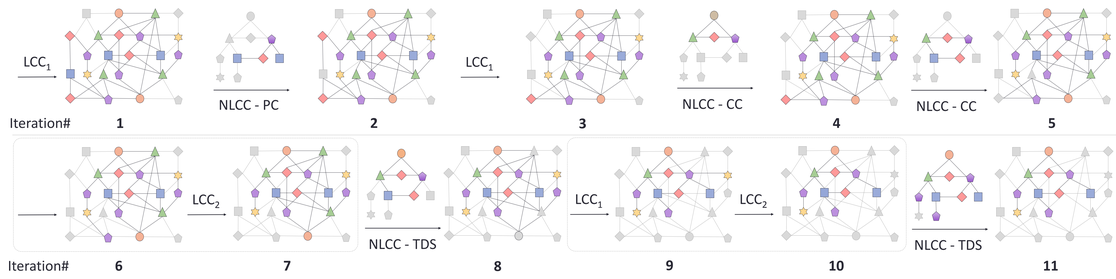
Non-local Constraint Checking (NLCC)

- ❑ Verifies topological requirements beyond one-hop neighborhood of a vertex.
- ❑ Required for templates for that LCC cannot guarantee 100% precision [4, 5].
- ❑ Leverages asynchronous Token Passing: the token issuing vertex is eliminated if constraint is not met.
- ❑ Three types of non-local constraints to verify:
  (i) Cycle Constraint (CC), (ii) Path Constraints (PC), and (iii) that require Template-Drive Search (TDS).

Non-local Constraint Generation



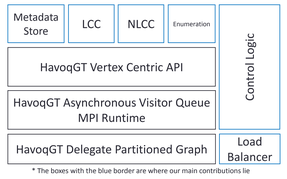
Algorithm Walk Through: Depicting Vertices and Edges in $G^*(V^*,E^*)$ Eliminated in each Iteration



3. Performance

Testbed – Quartz System Details	
CPU Arch.	Intel Xeon E5-2695 (2.1 GHz)
Cores/Node	36
Memory/Node	128GB
Total Nodes	2,634
Peak Perf.	2.6PFlop
Interconnect	Intel Omni-Path

Prototype implementation on HavoqGT showing Key System Components



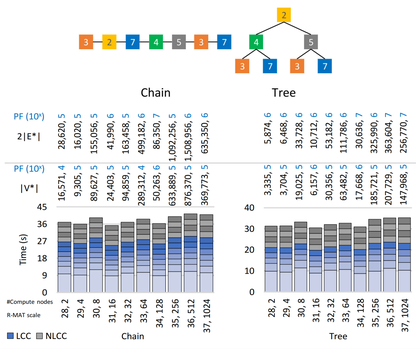
* The boxes with the blue border are where our main contributions lie

- ❑ Weak and Strong scaling experiments on up to 1024 nodes (36,864 cores).
- ❑ We search naturally occurring patterns. For real world graphs, the vertex labels that are among the most frequent (up to 220M) in the background graph.
- ❑ Vertex labels for the R-MAT graphs were created using the formula $l(v) = \lceil \log_2(d(v)+1) \rceil$.
- ❑ Scales to large graphs, e.g., $10^9 – 10^{12}$ edges, and tens of thousands of cores.
- ❑ Prune the graph by orders of magnitude, enabling full enumeration on trillion-edge graphs. (PF – pruning factor.)
- ❑ Optimizations: (i) aggressive edge elimination, (ii) asynchronous communication, (iii) tracking and preventing duplicate tokens from being forwarded, (iv) load balancing and (v) intuitive non-local constraint ordering offer up to an order of magnitude performance improvements (details in [4, 5]).

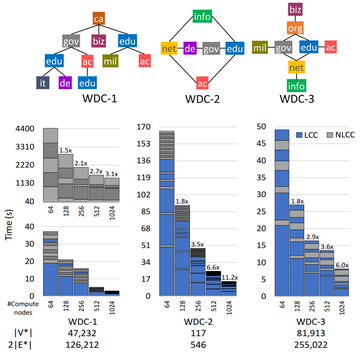
Datasets

Graphs	Type	$\|V\|$	$2\|E\|$	d_{max}	d_{avg}
Web Data Commons (WDC)	Real	3.5B	257B	95M	72.25
Reddit (RDT)	Real	3.9B	14B	19M	3.74
IMDB	Real	5M	29M	552K	5.83
R-MAT up to Scale 37	Synthetic	137B	4.4T	612M	32

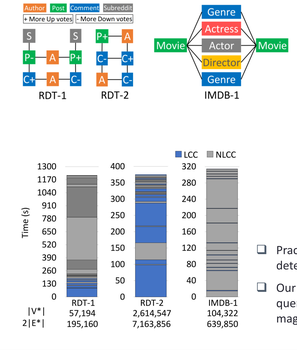
Weak Scaling



Strong Scaling



Use case: Social Network Analysis

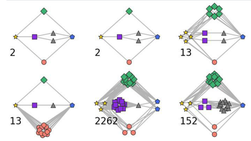


Pattern Enumeration

Pattern	Count	Time
WDC-1	668M	4min
WDC-2	2,444	1.84s
WDC-2	1.49B	40h
RDT-1	24K	78.98s
RDT-2	518K	2.29s
IMDB-1	1.68M	10h
R-MAT 37 Chain	55K	10.10s
R-MAT 37 Tree	32K	5.53s

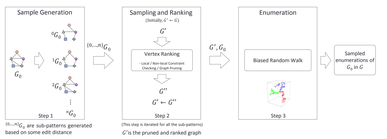
- ❑ Practical use case of Social Network Analysis: anomaly detection and pattern mining (details in [5]).
- ❑ Our system can handle complex patterns, beyond RDF-like queries, on real-world datasets, at least an order of magnitude larger than used in the past for similar problems.

4. Discussions and Work in Progress



The WDC-2 pattern matches in the background graph. The number of matches in each of the six connected components are also shown.

High-level Illustration of the Proposed Approximate Matching Pipeline Depicting Operations at each Step



Discussions

- ❑ Our solution targets Exact matching, supports arbitrary templates, and guarantees 100%, precision and 100% recall.
- ❑ The design is informed by requirements of distributed systems, i.e., Vertex-centric processing and Asynchronous communication.
- ❑ Optimizations to achieve scalability and slowdown the growth of the algorithm state.
- ❑ Opportunities for trade-offs between performance and precision.
- ❑ We show 6-10x speedup over a recent work, QFrag (see [5]).
- ❑ We argue that presenting the results as the union of all matches (figure above), which our pruning routines produce implicitly, is often easier to perceive by a human analyst , rather than explicit match enumeration (in a result set of millions of matches).
- ❑ Future improvements: (i) Identifying a near-optimal set of non-local constraints and their execution order, (ii) Flow control, and (iii) Message prioritization to accelerate token passing.

Work in Progress – Approximate Matching

- ❑ Approximate matching – the template and the match are just similar by some defined similarity metric [2].
- ❑ Well-known models: Edit distance, Random walk, Conductance, Metapath, …
- ❑ Adapt the distributed graph pruning infrastructure for ranking vertices based on their match probability.
- ❑ We propose a pattern approximation pipeline consists of three steps, namely, (i) Sample generation for Ranking, (ii) Sampling, Ranking and Pruning, and (iii) Enumeration.
- ❑ Samples are generated base on some edit distance [3].
- ❑ Two sets of useful output: (i) Vertex ranks, indicating their match probability and (ii) Samples of match enumerations.

References

[1] Ullmann, J. R. An algorithm for subgraph isomorphism, J. ACM, vol. 23, no. 1, pp. 31–42, January, 1976.

[2] Berry, J. W. Practical heuristics for inexact subgraph isomorphism, in Technical Report SAND2011-6558W. SandiaNationalLaboratories, 2011.

[3] Bunke, H. and G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recogn. Lett., vol. 1, no. 4, pp. 245–253, May, 1983.

[4] Reza, T., Klymko, C., Ripeanu, M., Sanders, G., and Pearce, R. Towards Practical and Robust Labeled Pattern Matching in Trillion-Edge Graphs. In Proceedings of The 19th IEEE International Conference on Cluster Computing (Cluster'17), Honolulu, Hawaii, 05 - 08 September, 2017.

[5] Reza, T., Ripeanu, M., Tripoul, N., Sanders, G., and Pearce, R. PruneJuice: Pruning Trillion-edge Graphs to a Precise Pattern-Matching Solution. In Proceedings of The IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'18), Dallas, Texas, 11 - 16 November, 2018.