Parallel Graph Mining with GPUs

Robert Kessl[1], Nilothpal Talukder[2], Pranay Anchuri[2], Mohammed J. Zaki[2,3]
[1]Czech Technical University, Prague, Czech Republic [2] Rensselaer Polytechnic Institute, Troy, NY, USA
[3]Qatar Computing Research Institute, Doha, Qatar

BigMine14 #BigDataMining August 24, New York City

Frequent Subgraph Mining

- Discovering frequent subgraph patterns from a set of graphs (multi-graph) or a single graph.
- Practical applications in biology (PPI network), chemistry (similar compounds), or social networks (similar communities), etc
- The problem is computationally hard as it requires:
  - Enumeration of an exponential number of candidate subgraph patterns
  - Checking their presence in a set of graphs (subgraph isomorphism)
- We address multi-graph problem with Graphics Processing Units

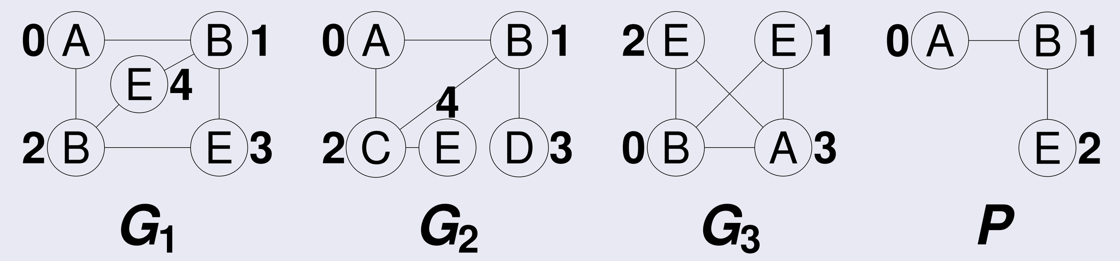


Figure : Graph database, $\mathcal{D} = \{G_1, G_2, G_3\}$ and pattern, P

Support: The number of graphs in $\mathcal{D} = \{G_i, \cdots G_n\}$ that contain P, i.e., $\sigma(P, \mathcal{D}) = |\{G : G \in \mathcal{D}$ and P subgraph isomorphic to $G\}|$.
Frequent Subgraph Mining: Given $\mathcal{D}$ and $minsup \in \mathbb{Z}$, find all frequent subgraph patterns, P, s.t. $\sigma(P, \mathcal{D}) \geq minsup$.

Graphics Processing Units (GPUs)

- GPUs have SIMT (single instruction multi-thread) architecture
- Energy efficient/less expensive choice for general purpose HPC
- Available programming platforms, such as NVIDIA CUDA[1]

Main challenges for graph mining:

- Serialized data for GPU kernels (no linked list or hashmap)
- Global memory coalescing as memory read in thread-groups
- Expensive GPU-CPU I/O transfer and dynamic allocation
- Algorithm steps are transformed into a set of parallel primitives (Prefix sum and Reduction etc.)

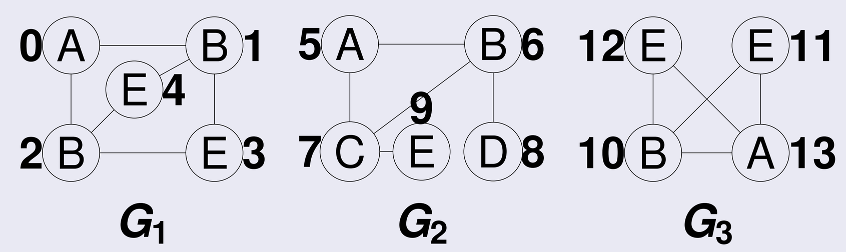
DFS code and Right-most Extension

- Graph mining algorithms [2,3] systematically generate all candidate subgraph patterns starting from a single edge
- gSpan algorithm [2] uses *minimal DFS code* [2] to uniquely represent patterns and avoid duplicates in candidate generation
- Perfoms right-most path extension from a minimal and frequent DFS code (subgraph pattern) by extending an edge
- Our variant keeps all occurences/embeddings [3] of the patterns (original gSpan does not) that grow with patterns

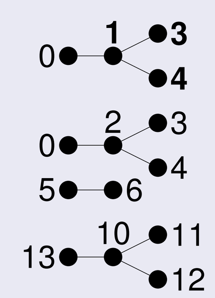
Graph Mining with GPUs

Algorithm: GraphMining($\mathcal{D}$, $minsup$, P, Embeddings $\Sigma_{\mathcal{D}}(P)$)
//Initial Call: $P = \emptyset, \Sigma_{\mathcal{D}}(P) = \emptyset$
(GPU step) Get all possible edges extensions (exts) $\mathcal{E}_{\mathcal{D}}(P)$ of P
(GPU step) Compute support for exts in $\mathcal{E}_{\mathcal{D}}(P)$ and remove infrequent exts
for each $e = (v_i, v_j, l_i, l_{ij}, l_j) \in \mathcal{E}_{\mathcal{D}}(P)$ **do**
  $P' \leftarrow P$ extended by e
  if $P' = minDFS(P')$ **then**
    output P'
    (GPU step) Create $\Sigma_{\mathcal{D}}(P')$
    GraphMining($\mathcal{D}$, $minsup$, P', $\Sigma_{\mathcal{D}}(P')$)
  end if
end for

Data Structures



(a) Graph database $\mathcal{D}$ with *global vertex ids*



Prefix tree rep.

Q_1 idx	Q_1 vid
-1	0
-1	0
-1	5
-1	13

Q_2 idx	Q_2 vid
0	1
1	2
2	6
3	10

Q_3 idx	Q_3 vid
0	3
0	4
1	3
1	4
3	11
3	12

In GPU memory

(b) Embeddings of $P = (0, 1, A, -, B)(1, 2, B, -, E)$

global id	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N	1 2	0 3 4	0 3 4	1 2	1 2	6 7	5 7 8	5 6 9	6	7	11 12 13	10 13	10 11 12	10 11 12

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
O	0	2	5	8	10	12	14	17	20	21	22	25	27	30	-1

(c) GPU representation (partial) of graph database

Figure : Graph database with *global vertex ids* and its storage in GPU memory.

Major GPU Steps

- **Pattern emumeration via embeddings:** Get all valid extensions on the right-most path of P in parallel
  - Look-up the neighborhood of each vertex, and set the entry in a large binary array for a valid forward/backward extension
  - Produce the ordered indexes by prefix sum and store extensions in EXT
  - Threads are allocated in column-major fashion for memory locality
  - EXT are segmented, EXT_k contains all exts from a pattern vertex
  - EXT_k contains blocks of elements corresponding to the same graph

Extensions

k	EXT_0							EXT_1				EXT_2			
v_i^g	0	0	0	0	5	13	13	1	1	6	10	3	4	11	12
v_j^g	2	2	1	1	7	11	12	3	4	8	12	2	2	13	13
l_j	B	B	B	B	C	E	E	E	E	D	E	B	B	A	A
B	0	0	0	1	1	0	0	0	1	1	0	0	1	1	0
scan **B**	0	0	0	0	1	2	2	0	0	1	2	0	0	1	1

Support Computation

	$(0,B)$			$(0,C)$			$(0,E)$			$(1,E)$			$(1,D)$		
F	1	0	0	0	1	0	0	0	1	1	0	1	0	1	0
scan of **F**	1			1			1			2			1		

Figure : The extensions of the pattern $P = (0, 1, A, -, B)(1, 2, B, -, E)$, and support computation of the extensions from vertices **0** and **1**.

Major GPU Steps

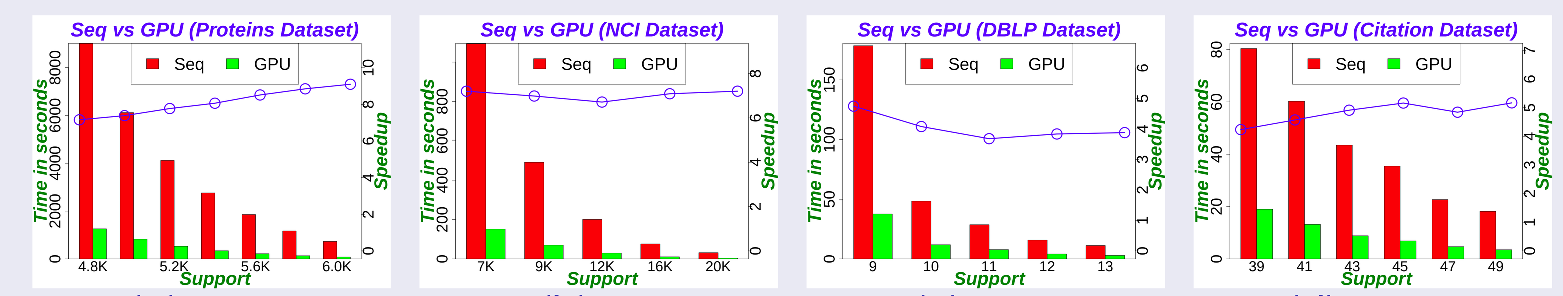
- **Support Computation:** Compute the support of P from the extensions
  - First, extract unique DFS code extensions from EXT
  - Support computation variants are based on the number of extentions processed in parallel : (a) single-ext, (b) single-seg, and (c) multi-seg
  - Number of threads is a bottleneck; single-seg which processes EXT_k demonstrates the best performance
- **Growing the Embeddings:**
  - Forward extensions from EXT_k introduces a new embeddings column
  - For the backward extensions the last embeddings column is filtered

Results

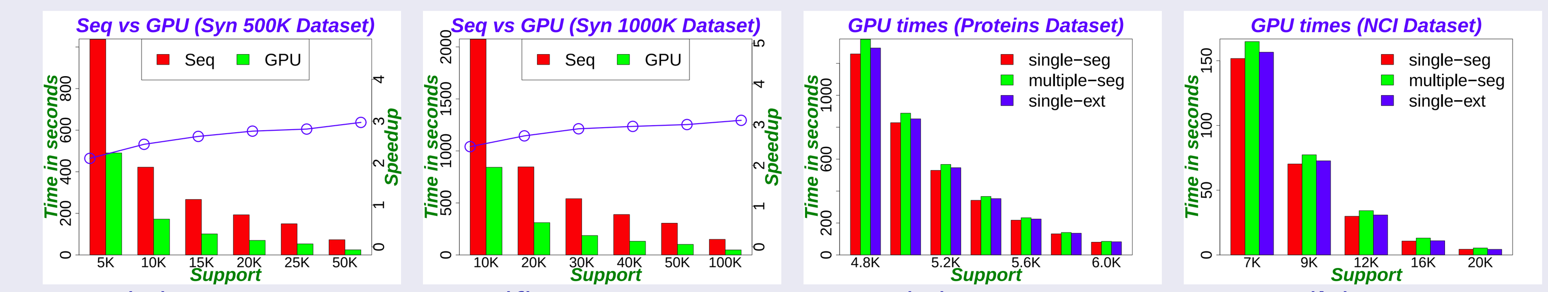
Setup: Sequential - Opteron 2.1GHz (256GB memory SMP 16 cores)
GPU - Tesla C2075 (448 cores and 6GB memory)

dataset	num of graphs	avg. $\lvert V\rvert$	avg. $\lvert E\rvert$	V labels	E labels	avg. deg	avg. clus. coeff.
Citation	81	469	541	15	1	2.042	0.008
Protein	6875	1113	4392	22	1	7.891	0.554
DBLP	21	205	300	26	26	2.796	0.594
NCI	24595	39	78	54	1	3.954	0.0014
Syn500K	500000	25	31	5	5	2.517	0.383
Syn1000K	1000000	25	31	5	5	2.517	0.383

Table : Properties of the datasets



(a) Protein (b) NCI (c) DBLP (d) Citation



(e) Syn 500K (f) Syn 1000K (g) Protein (h) NCI

Conclusions and Future Directions

- GPU memory is constrained for storing embeddings
- Investigate large graph mining in disributed setting
- Explore other accelerators, e.g. Intel Xeon Phi coprocessor

References

1. NVIDIA CUDA: http://www.nvidia.com/object/cuda_home_new.html
2. X. Yan, J. Han. gSpan: Graph-Based Substructure Pattern Mining. International Conference on Data Mining, 2002
3. S. Nijssen, J. Kok. A quickstart in frequent structure mining can make a difference. ACM International conference on Knowledge discovery and data mining, 2004
4. G. Buehrer, S, Parthasarathy, Y. Chen. Adaptive Parallel Graph Mining for CMP Architectures. International Conference on Data Mining, 2006

Acknowledgements: This work was supported in part by NSF Awards CCF-1240646 and IIS-1302231.