Network Motif Discovery: A GPU Approach

Wenqing Lin[†,‡], Xiaokui Xiao[‡], Xing Xie[§], and Xiao-Li Li[†]

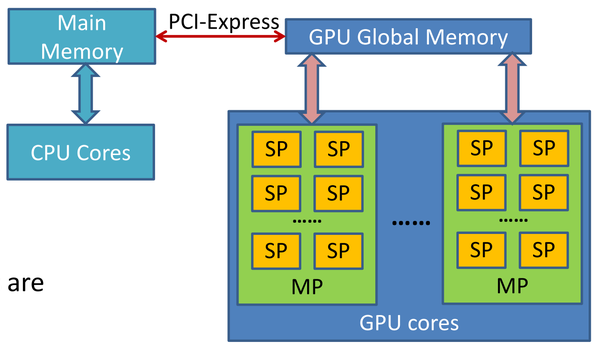
[†] Agency for Science, Technology and Research SINGAPORE
[‡] NANYANG TECHNOLOGICAL UNIVERSITY
[§] Microsoft Research

1. Introduction

- A *network motif* in a graph G is a *subgraph* g of G, such that g appears much more frequently in G than in *random graphs* whose degree distributions are similar to that of G.
- Applications such as:
  - Predict protein interactions in biological networks.
  - Understand the characteristics of circuits in electronic engineering.
  - Study the functionalities of brain networks.
- Challenges:
  - A large number of random graphs, subgraph isomorphism tests, and subgraphs enumerated from G.
- Our contributions:
  - A novel and cost-effective solution that exploits the strengths of GPUs.
  - Three optimization techniques that improve the scalability of our solution, avoid under-utilization of GPU, and eliminate redundant computation.
  - Demonstrate that our GPU-based solution is orders of magnitude faster and more cost-effective than the state-of-the-art CPU-based methods.

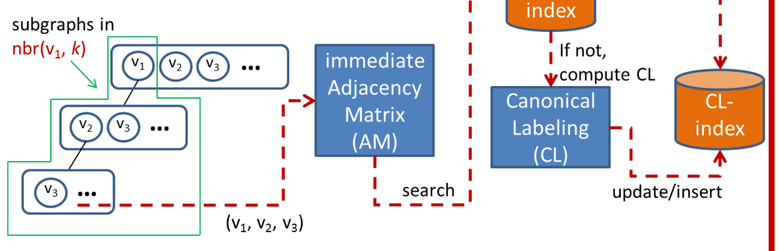
2. Preliminaries

- Network Motifs
  - Given two graphs G and g, the *frequency* of g in G is the number of subgraphs in G each of which is isomorphic to g, denoted by $f(g, G)$.
  - Consider a set $\mathcal{G}_r$ of r random graphs, each of which is *degree-equivalent* to G. The expected frequency of g in a random graph can be estimated as
    $$\tilde{f}(g) = \frac{1}{r}\sum_{G' \in \mathcal{G}_r} f(g, G'),$$
    and the corresponding sample standard deviation is
    $$\tilde{\sigma}(g) = \sqrt{\frac{1}{r-1}\sum_{G' \in \mathcal{G}_r} \left(f(g, G') - \tilde{f}(g)\right)^2}.$$
  - Given a user-defined threshold $\theta > 0$, we say g is a ***motif*** of G if and only if $\tilde{\sigma}(g) > 0$ and $f(g, G) - \tilde{f}(g) \geq \theta \cdot \tilde{\sigma}(g)$.
  - **Problem definition**. Given G, $r > 0$, $k > 2$, and $\theta > 0$, the problem of network motif discovery asks for all size-k motifs of G with respect to $\mathcal{G}_r$.
- Graphics Processing Units (GPUs)
  - Stream processors (SPs) in each Multiprocessor (MP) work in SIMD manner.
  - Nvidia CUDA Programming Model
    - *Kernels*: Run on GPU, invoked by CPU. Transfer data between CPU and GPU.
    - *Thread Hierarchy*: Each thread executes a kernel on a GPU core. Organized by blocks, each of which run on an MP.
    - *Branch Divergence*: Different execution paths are executed sequentially.
    - *Memory Coalescing*: Requests to Consecutive memory space will be one access.



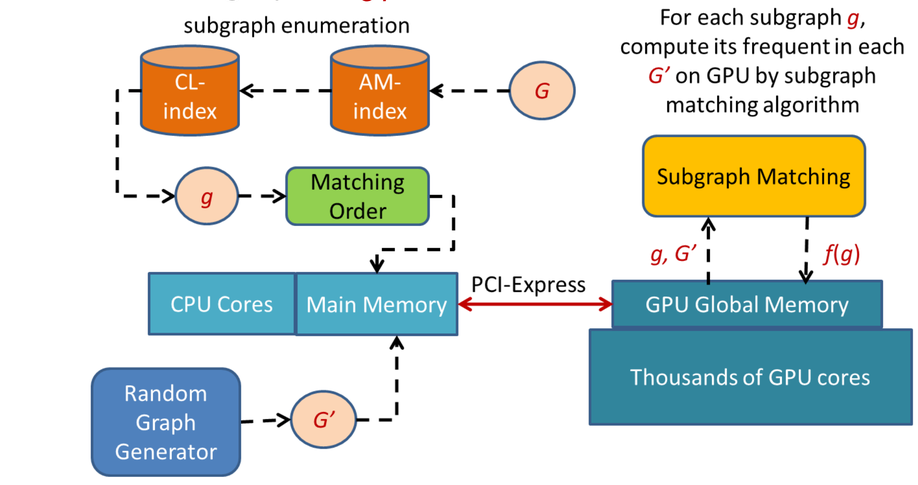
3. CPU-based Methods Revisited

- Two phases of computation
  - *Phase 1*. Subgraph enumeration in G.
  - *Phase 2*. Frequency estimation in random graphs $\mathcal{G}_r$.
- Difficulties in GPU Translations
  - Computing canonical labeling contains complicated execution paths.
  - Random access to AM-index and CL-index.
  - Index size is too large to fit in GPU global memory.
  - Inefficient by examining subgraphs that have different structures.



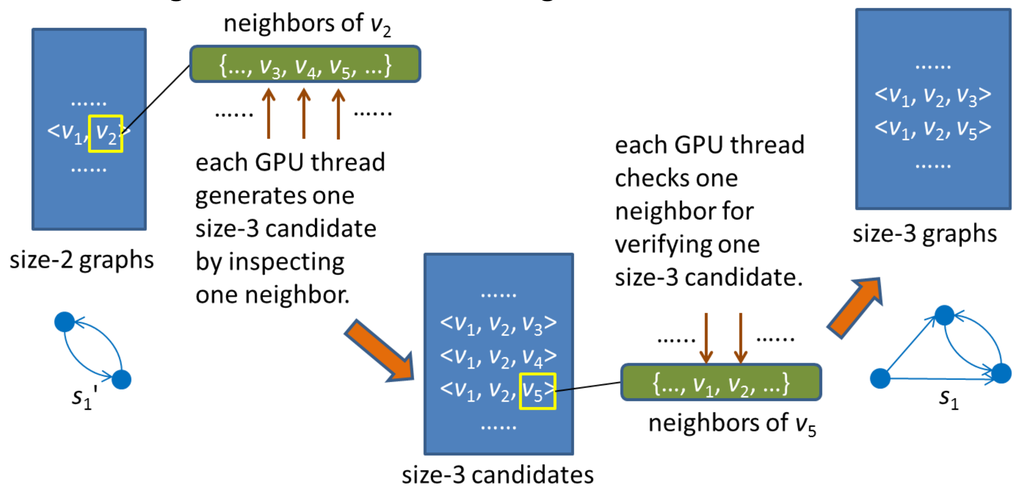
4. Solution overview

- Enumerate subgraphs in G using existing methods.
- Parallelize subgraph matching on GPU for each subgraph g and each random graph in $\mathcal{G}_r$.



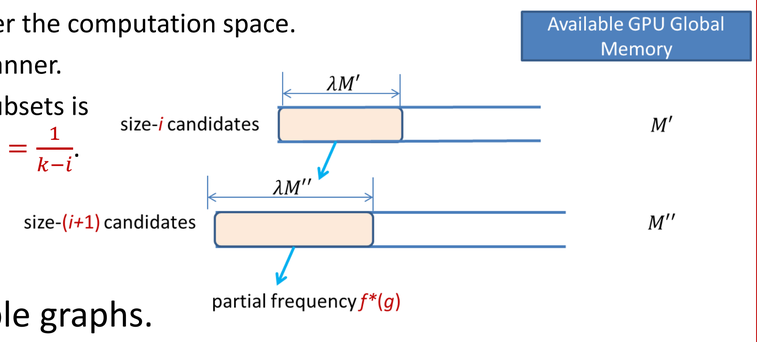
5. GPU-based Subgraph Matching

- Compute a matching order of vertices in g.
- Iteratively compute the frequency of g:
  - Matching one vertex in the matching order in one iteration.

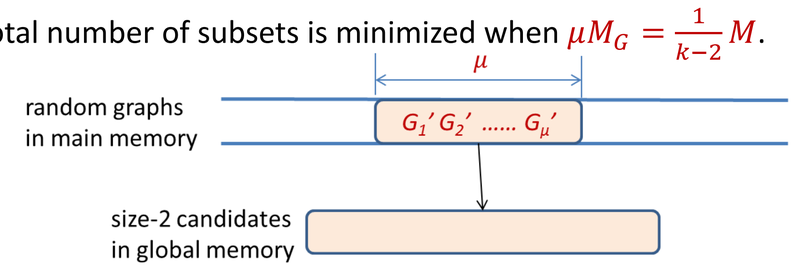


6. Optimizations

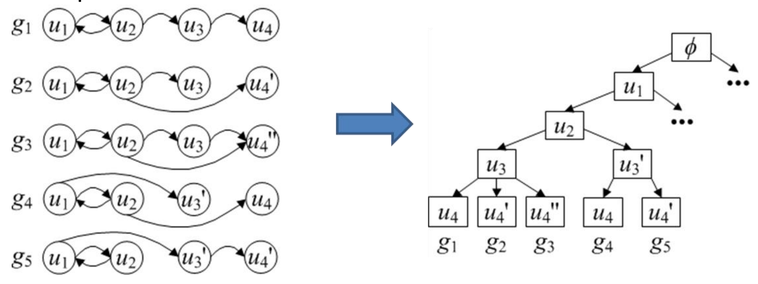
- Handling large intermediate results.
  - Divide and Conquer the computation space.
  - Traverse in DFS manner.
  - Total number of subsets is minimized when $\lambda = \frac{1}{k-i}$.



- Handling multiple graphs.
  - Total number of subsets is minimized when $\mu M_G = \frac{1}{k-2} M$.



- Matching tree.
  - Common prefix of matching order for two graphs.
  - DFS traversal on the matching tree.
  - Re-use computation in ancestor nodes.



7. Selected Experiments

- Datasets.

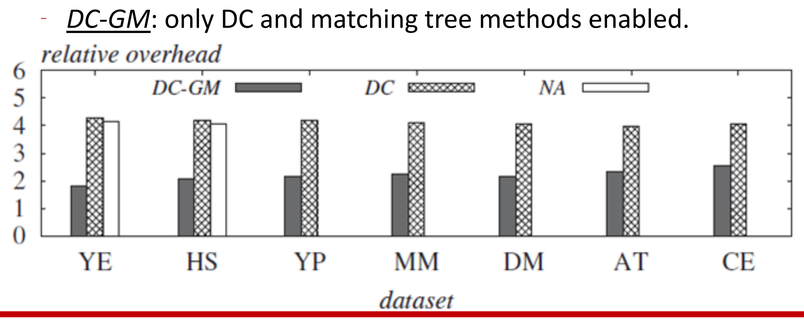
datasets	YE	HS	YP	MM	DM	AT	CE
|V|	688	1,509	2,361	4,293	6,303	9,216	17,179
|E|	1,079	5,598	6,646	7,987	18,224	50,669	124,599
Avg deg	3.14	7.42	5.63	3.72	5.78	11.00	14.51

- Devices.

name	# of cores	GPU Memory	Price (USD)
Intel Xeon E5645 CPU	6	N/A	513.39
Quadro 2000 GPU (Q2000)	192	1GB	277.77
Telsa K20 GPU (K20)	2496	5GB	2695.00

- Default: K20, k =6, r =1000.

- Effects of optimizations.
  - *NA*: all optimization methods are disabled.
  - *DC*: only the divide-and-conquer method enabled.
  - *DC-GM*: only DC and matching tree methods enabled.



- Comparisons with CPU-based techniques.