Parallel Depth-First Search for Directed Acyclic Graphs

Divyanshu Talwar (2015028), Viraj Parimi (2015068)
Advisor: Dr. Ojaswa Sharma

Introduction

- Let a graph G = (V, E), be defined by its vertex V = {1, 2, ..,n} and edge E = {(i1, j1), (i2, j2), ..,(im, jm)} sets, with |V | = n and |E| = m. The sequential lexicographic Depth-First Search (DFS) algorithm was proposed in [4].
- The DFS traversal problem requires us to compute : parent information , pre-order (start time) and post-order (end time) for every node in G.
- The sequential DFS algorithm[4], in itself, is not at all parallelizable.
- In this work, we aim to implement parallel-DFS algorithm proposed in [2] [3] and report the it's speed-up. (using [1] dataset).

Parallel DFS Algorithm

The algorithm is subdivided into 3 components : DAG to DT conversion, subgraph size calculation, and pre and post order calculation; where in each execution of component aids in solving the DFS problem. The GPU kernels implemented in the CUDA program are as follows :

1. **dag_to_dt** - It converts a given DAG to a DT. Since, for computing the parent information one needs to have only one parent per node. This algorithm computes the parent part of the DFS problem.
2. **subgraph_size** - The algorithms traverses the graph in a top-down fashion computing the prefix sum values of the zeta values of the nodes, which are used in calculating the pre and post order of the nodes..
3. **pre_post_order** - This takes the zeta prefix sum values as input to compute the pre (discovery time) and post (finish time) order of the nodes.

The final algorithm, solves the DFS problem by calculating it's three subproblems. This parallel DFS algorithm is proven to be work-efficient.

Optimizations

1. **Path pruning.**
2. **CUDA streams** for asynchronous memory transfer.
3. **Inline functions f**or faster execution.
4. Use of **pinned memory.**
5. Use of **arithmetic shift operators.**
6. **Structure of Arrays vs Array of Structures** for storing the dataset in CSC format.
7. Extensively used **__constant_memory** (where-ever possible)**.**

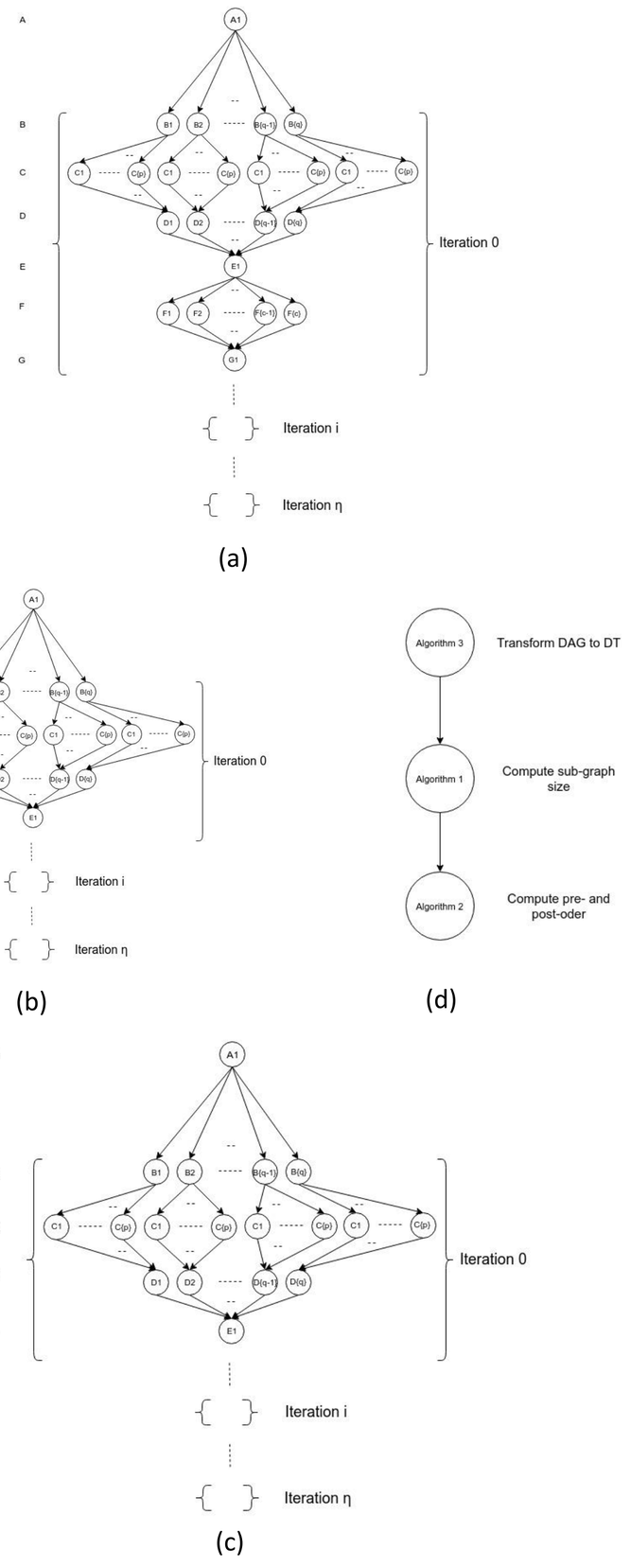
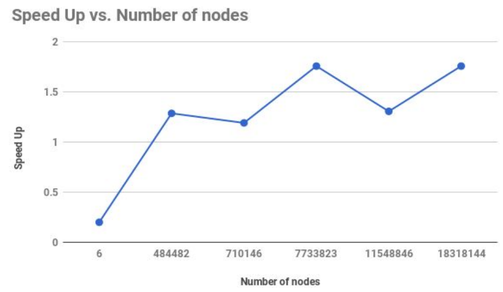


Figure 1: Task Dependency Graphs for (a) Subgraph size calculation,, (b) pre-post order, (c) DAG to DT conversion, and (d) final algorithms.

Results



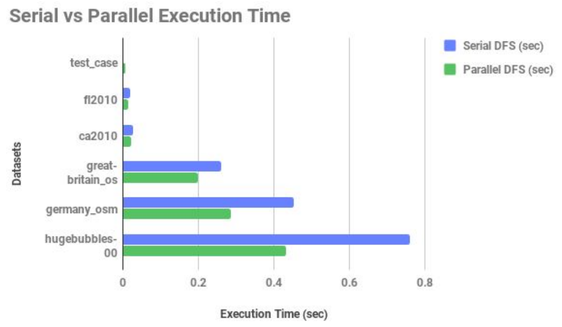


Table 1. Sample DIMACS graphs/adjacency matrices

Datasets	n	m	d	η
test_case	6	6	2	4
hugebubbles-00	18318144	30144175	3	6724
fl2010	484482	1270757	121	105
ca2010	710146	1880571	119	125
great-britain_os	7733823	8523976	7	7076
germany_osm	11548846	12793527	12	8890

Table 2. Serial vs Parallel Execution Times

Dataset	Number of nodes	DFS (sec)	Parallel DFS (sec)	Speed Up
test_case	6	0.001	0.005	0.2
fl2010	484482	0.018	0.014	1.285
ca2010	710146	0.025	0.021	1.190
great-britain_os	7733823	0.259	0.1984	1.756
germany_osm	11548846	0.451	0.286	1.305
hugebubbles-00	18318144	0.759	0.432	1.756

References

[1] UF Sparse Matrix Collection. 2015. https://www.cise.ufl.edu/research/sparse/matrices/list_by_dimension.html
[2] Maxim Naumov, Alysson Vrielink, and Michael Garland. 2017. Parallel Depth-First Search for Directed Acyclic Graphs. NVIDIA Technical
[3] Maxim Naumov, Alysson Vrielink, and Michael Garland. 2017. Parallel Depth-First Search for Directed Acyclic Graphs. In proceedings of the Seventh Workshop on Irregular Applications: Architectures and Algorithms (IA3 '17). NVIDIA, Santa Clara, CA, Article 4.
[4] R. E. Tarjan. 1972. Depth-first Search and Linear Graph Algorithms. (SIAM J. Comput. 1). Article 7, 146-160 pages