NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Parallel and Distributed Computing
Semester Project: ParButterfly

Submitted By:
I22-xxxx Arshiq Rehman
I22-1071 Rehal Saeed
I22-0757 Samra Mashaam

ParButterfly: Parallel butterfly counting in bipartite graphs

Initial research and planning:

The ParButterfly framework as outlined in the paper authored by Jessica Shi and Julian Shun was as such: The first step is ranking the vertices. Then wedges are retrieved and counted before the butterfly counting begins. The butterflies are counted through wing and tip peeling, which is done by repeatedly removing the vertex or edge with the least butterflies. The ParButterfly algorithm also requires preprocessing, which is where the program ranks the nodes, and sorts the neighbours and degrees of nodes.

The initial plan is outlined below:

1. Graph partitioning with METIS: Divide the bipartite graph into balanced subgraphs. METIS ensures edges spanning across partitions are minimized, with the rest being handled through duplicates or ghost nodes.
2. MPI (Inter Node parallelization): Each partition is assigned to a different MPI process. The goal is to distribute local subgraphs and local vertex/edge data to each node. Broadcast ranking schemes, and use all to all or point to point exchange for wedge edges that span partitions. The processes will synchronize after wedge counting and during peeling, when butterfly counts are updated. MPI can also be used to handle exchange of ghost node info.
3. OpenMP or OpenCL (Intra Node parallelization): Use OpenMP for wedge retrieval, wedge aggregation, butterfly counting, and/or peeling rounds. On the other hand, OpenCL can launch parallel kernels for wedge extraction from edge lists and update counts for edges/vertices.

Serial Implementation:

The program handles .txt and .csv files as input. It also generates graphs if no input is available. The graph nodes are read and stored as vectors in a BipartiteGraph structure. The nodes are preprocessed, where they are sorted according to their ranks using either “degree” or “side”, and the neighbours are sorted. The graph is run through the exact_counts function, which does the counting for both U and V nodes. A ButterflyCounts structure is made for this function, where per_vertex and per_edge counts are kept in maps/hash tables. There is also an approximate_counts function, which approximates the counts based on a sample probability. Then the tip decomposition is done before the wing decomposition, both using priority queues.

Parallel Implementation:

MPI Parallelization:
The parallel version of the program is built on top of the serial version with changes. The root, that is the process with rank 0, handles the reading of the files. A specialized function, broadcastBipartiteGraph, is used to broadcast the graph to other processes. First, the sizes of U and V node lists are broadcasted. Then the adjacency lists for U and V are broadcasted. Another function, divideGraph, splits and assigns sections of the graph to the processes. This is done based on the process number and size of the division for the U vertices. The program preprocesses the graph, performs exact counting, and wing and tip decomposition. At each step, MPI barrier is called to ensure smooth and evenly paced work. A function called combineButterflyCounts combines the processes. The counts from each process and each step are accumulated here, before each process displays its results. This did provide a noticeable speedup, leading to successful implementation of MPI.

METIS:
During the implementation of this program, the plan regarding METIS was changed. This was because the whole graph had to first be converted to CSR format (which has still been implemented in the program), before METIS returned arrays with the required data. Then, the nodes had to be divided according to the results. During testing, this whole procedure ended up taking too much preprocessing time, creating an even slower version of the program than its serial counterpart. As a result, METIS was not considered a viable solution for speeding up the program and was removed.

Tools used: Gprof and Time

Serial Profiling:

devs.csv:
Preprocessing: 383.39ms
Exact Counting: 93678.0390000 ms
Approximate butterfly count (p=0.100000): 1167494599
Approximate Counting: 8768.3630000 ms
Exact counting time: 93024.3060000 ms
Approximate counting time: 8768.2830000 ms
Total Execution: 584506.8370000 ms
Real time: 9m54.226s
User: 9m40.352s
System time: 0m2.176s

moderators.csv:
Preprocessing: 461.3380000 ms
Exact Counting: 259541.7950000 ms
Approximate butterfly count (p=0.100000): 931599
Approximate Counting: 81.7200000 ms
Exact counting time: 259045.3960000 ms
Approximate counting time: 81.6490000 ms
Total Execution: 833506.2530000 ms

txt_1.txt:
Exact counting time: 0.3240000 ms
Approximate counting time: 0.0750000 ms
Total Execution: 1952.9300000 ms

txt_2.txt:
Exact counting time: 0.6200000 ms
Approximate counting time: 0.8690000 ms
Total Execution: 1819.0090000 ms

Parallel profiling (MPI Only):

devs.csv:
Preprocessing: 365.8950000 ms
Exact Counting: 83042.0140000 ms
Exact Counting: 125321.0780000 ms
Exact counting time: 46479.2010000 ms
===== Program Timing =====
Total program execution time: 791478 ms
real	13m12.452s
user	52m23.731s
sys	0m7.906s

txt_1.txt:
Preprocessing: 0.0570000 ms
Exact Counting: 0.2240000 ms
Exact Counting: 2.4800000 ms
Tip Decomposition: 7.3280000 ms
Exact Counting: 0.2760000 ms
Wing Decomposition: 1.1490000 ms

```
===== Counting Statistics =====
Maximum process counting time: 2.4280000 ms
Average process counting time: 0.7910000 ms
Exact counting time: 0.2180000 ms

===== Program Timing =====
Total program execution time: 9483 ms
```

txt_2.txt:

```
Preprocessing: 0.0380000 ms
Tip Decomposition: 1.0880000 ms
Exact Counting: 0.2720000 ms
Wing Decomposition: 0.9660000 ms
Exact counting time: 0.6200000 ms
Approximate counting time: 0.8690000 ms
Total Execution: 1819.0090000 ms
```

Parallel Profiling (MPI and OpenMP):

txt_1.txt:

```
Preprocessing: 3.7020000 ms
Exact Counting: 0.5300000 ms
Exact Counting: 25.1250000 ms
Tip Decomposition: 30.5020000 ms
Exact Counting: 26.5780000 ms
Wing Decomposition: 27.1670000 ms

===== Counting Statistics =====
Maximum process counting time: 22.6190000 ms
Average process counting time: 9.2137500 ms
Exact counting time: 0.5230000 ms

===== Program Timing =====
Total program execution time: 8841 ms
```

txt_2.txt:

```
Preprocessing: 16.9610000 ms
Exact Counting: 16.4810000 ms
Exact Counting: 28.9660000 ms
Tip Decomposition: 31.6680000 ms
Exact Counting: 0.7450000 ms
Wing Decomposition: 4.2810000 ms

===== Counting Statistics =====
Maximum process counting time: 22.6450000 ms
Average process counting time: 18.4460000 ms

Exact counting time: 16.4730000 ms

===== Program Timing =====
Total program execution time: 7058 ms
```