GraphProfile Manual

Introduction

GraphProfile is a command-line program meant to compute metrics of a undirected graph, thus in a sense ”profiling” it. In addition to computing metrics, the program can also create ”reduced” graphs, by removing edges (but not vertices) of a given input graph. Lastly, a more specific functionality is that the program will compute metrics of a given input graph, and to several reduced versions of the input graph, in a single command.

Basic Use

The program has 2 main prompts. The first prompt, which will be shown right as you execute the program, tells you to enter an undirected Adjacency List. When this prompt appears, the program expects a full path to an Adjacency List file. This file should only contain pairs of numbers; letters or punctuation will crash the program. The program will read the file and create a graph for the file. Once the graph is read in, the program will proceed to the second prompt.
The second main prompt occurs after a graph has been read in from a file. This prompt asks you to write in one of the list commands. These commands will compute metrics on the graph or create a new reduced graph that will be output into a file. Typing exit will leave the second prompt, and bring you to back the first prompt where you can enter a new graph file or exit the program. Due to high memory usage, the program does not support having more than one graph open at a single time. A list of the commands available during the second prompt are detailed below.

Advanced Use

The other functionality of GraphProfile aggregates the basic commands and combines them into one so that the program can be left on its own computing over long periods of time, such as overnight. When given the prompt to enter an adjacency list filename, you can type *suitemode*. This will prompt to you to enter as many filenames as you want, one line at a time. Then, when finished, the program will compute many graph metrics to the each graph, and its several reduced graphs created from the original.

The graph metrics that will be computed are as follows:

- Number of edges
- Number of vertices
- Approximate Diameter
- Number of Connected Component & Sizes
- Page Rank Rankings in Reduced Graphs vs Original
- Reachability of 1000 random vertex pairs.
- Changes in Distance in the 1000 random vertex pairs in the reduced graph

The reduced graphs for which the metrics will be computed are as follows:

- Keep top k neighbours, for $k = 2, 4, 8, 16$ (suitemodetop)
- Spanning tree using top k degree vertices, for $k = 3, 5$ (suitemodetree)
- Keep k neighbours, prioritizing high degree neighbours, avoiding triangles, for $k = 2, 4, 8, 16$ (suitemodetri)

Graph Commands Computing Graph Metrics

In alphabetical order, these are the commands available for computing graph metrics.

- ”abc” - Computes the approximate betweenness centrality of a certain vertex. Outputs to *Graph Properties.txt.*
- ”adiam” - Computes the approximate diameter of the graph. Outputs to *Graph Properties.txt.*
- ”aprank” - computes the approximate page rank of the vertices of the graph. Outputs to *Approximate Page Rank.txt.*
- ”bbc” - Computes betweenness centrality of every vertex. Uses boost’s betweenness centrality function. Outputs to *Brandes BC.txt.*
- ”bc” - Computes betweenness centrality of every vertex. Very memory intensive. Tends to crash on larger graphs. Outputs to *Betweenness Centrality.txt.*
- ”cc” - Computes the number of connected components, and the size of the largest components. Outputs component sizes until 90% of all vertices are accounted for. Outputs to *Graph Properties.txt.*
- ”dist” - Computes the distance of two vertices. Outputs to Graph Properties.txt.
- ”ediam” - Computes the exact diameter of a graph. This solves the all pairs shortest path problem. Can be expensive, and may crash on large graphs. Outputs to *Graph Properties.txt.*
- ”edges” - Displays the number of edges on the console.
- ”etri” - Computes the number of triangles in a graph. Outputs to *Graph Properties.txt.*
- ”lcc” - Computes the local clustering coeff. of a vertex, or the average between all vertices. Outputs to *Graph Properties.txt.*
- ”prank” - Computes the page rank of all vertices. Outputs to *Page Rank.txt.*
- ”vertices” - Displays the number of vertices on the console.

Graph Commands Creating Reduced Graphs

In alphabetical order, these are the commands available for creating reduced graphs.

- "reduce" - Creates the reduced graph that keeps the edges to the highest k degree neighbours of each vertex. Outputs the adjacency list of the reduced graph to *Reduced Graph.txt*.

- "reducetree" - Prompts the user to select vertices as roots. Uses these roots to create spanning trees. Creates a reduced graph by joining the spanning trees. Outputs the adjacency list of the reduced graph to *Reduced Graph Tree.txt*.

- "reducetree2" - Creates the reduced graph that is made up of spanning trees where the top k degree vertices are the roots. Outputs the adjacency list of the reduced graph to *Reduced Graph Tree2.txt*.

- "reducetri" - Creates the reduced graph that keeps k edges for each vertex, prioritizing high degree neighbours, but avoiding triangles formed by the neighbours. Ouputs the adjacency list of the reduced graph to *Reduced Graph Triangle.txt*.