CS-E4600 — Algorithmic Methods of Data Mining

Programming project

Mining Massive Graphs

Muhammad Kamal Memon - 600442
Muhammad Rafay Khan - 599838

Introduction

This report describes the approaches and methods used to extract the statistics from massive real world graphs. We have used two mechanisms for computing graph related statistics; exact computation where possible and approximation techniques. In the first half of the project we have computed the exact statistics and secondly we computed the approximate statistics whose results were then compared with the exact computations.

We have used the following four graphs from the Stanford Network Analysis Platform (SNAP) datasets [1]. The table below depicts the number of nodes and edges of each graph for its Largest Strongly Connected Component (LSCC) and Largest Weakly Connected Component (LWCC).

		LSCC		LWCC	
Graph Name	**Total Nodes**	**Nodes**	**Edges**	**Nodes**	**Edges**
wiki-Vote	7115	1300	39456	7066	103663
soc-Epinions1	75879	32223	443506	75877	405739
ego-Gplus	107614	69501	9168660	107614	12238285
soc-Pokec	1632803	1304537	20865476	1632803	22301964

Design and Methodology

We have used the following two Python libraries for our task of mining the above mentioned graphs:

- NetworkX - Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. [2]
- NetworKit - open-source toolkit for large-scale network analysis. Its Python module has performance-aware algorithms written in C++ and exposed to Python via the Cython toolchain. [3]

In this project, NetworkX is used extensively for all graph related computations and approximations except the ANF approximation. For ANF we have used NetworKit. All methods were first tested on the smallest graph; *Wiki-Vote* and were then extended to the other graphs.

Exact Computations

The major steps of the computation methodology for exact statistics are here below:

1. We used the NetworkX I/O methods for the extraction of the largest strongly connected component (LSCC) and the largely weakly connected component (LWCC). For each graph, these components were saved in a separate file.
2. For computing the shortest path distribution, we have used these component files and feed them to the NetworkX methods in order to obtain the shortest path distributions. This step was manually optimized.
3. Once we had the distributions, we then performed analysis using *numpy* to extract the required statistics which were; Mean, Median, Diameter and Effective Diameter.

Approximate Computations

We used three separate approximation schemes for each graph. The randomization part for each scheme were carried out by using the python's *numpy* package random number generator. Methodology for each scheme is describe below:

1. Sampling random pairs

   For sampling random pairs we used the number of random pairs as our approximation parameter. Following were the major steps in our approach:

   a. Random pair of nodes were taken from each of the component.
   b. Shortest paths between these random pairs were computed.
   c. All computed shortest paths were then made a distribution over which we calculated the approximate statistics of Mean, Median, Diameter and Effective Diameter of the Graph.

2. Sampling Random Sources with complete BFS

   For this scheme the accuracy parameter is the number of random sources sampled. Following were the major steps:

   a. We sampled a random node from each component of the graph.
   b. We ran the full Breath First Search for each of the sampled source node on the whole graph.
   c. The results of the BFS for each sources are then accumulated to form a distribution.
   d. The approximate statistics were computed on generated distribution.

3. Using ANF for Graph's hop plot approximation

   Flajolet-Martin have proposed a handy approximation algorithm for mining networks. In our project, we have used Approximate Neighborhood Function which is one of its variant by Palmer et al. [4]. The exact implementation in C++ can be found on [5], whereas mentioned earlier we have used NetworKit python's interface for our implementation.

   The hop-plot is the set of pairs $(d, g(g))$ for each natural number d and where $g(d)$ is the fraction of connected node pairs whose shortest connecting path has length at most d. The approximation parameters for this approach are:

   - k - Number of parallel approximations, larger for more precise result.
   - r - Number of additional bits, important for shorter graphs.

   Following were the major steps in our process for ANF:

   a. Using NetworKit I/O functions, each component of the graph (LSCC/LWCC) is read.
   b. We computed the distance by getting the Hop Plot Approximation for each component.
   c. The hop plot is then processed by computing the cumulative hops over each length and then calculating the Mean by taking the average of all the distances.
   d. For Median, we first take the index of the middle node and find at which hop it lies.
   e. Number of hops with distances less than one define the effective diameter and the length of the hop plot defines the approximate Diameter of the graph.

Optimizations, Experiments and Observations

During our computations for exact statistics we observed a huge bottleneck second step of the process that is for the extraction of path distribution. Time and memory restrictions forced us to make our approach modular as when computing exact statistics the used memory grows exponentially with the size of graph (number of nodes/edges) and makes it impossible to do any computations.

Moreover, the method of computing the shortest path distribution is optimized manually as NetworkX methods do not do parallel computing. We created batches of iterations over edges and each batch was assigned to a predefined number of CPU processes. The path distributions from all these processes in the batch are merged at the end of each batch. When all the nodes in the graph are iterated, the final path distribution is saved into a file. The path distribution is than read from the file and the statistics are extracted from the distribution. This methodology allowed us to extract the path distribution for massive graphs without consuming the available memory of the system.

Our local computers gave up on the second graph mainly due the shortage of memory and hence we had to use the GPUs on force.aalto.fi. However, even after these optimizations we only managed to compute the exact statistics for the first two graphs and it was not feasible to compute them for the last two graphs in a reasonable time.

Results and Findings

TASK 1 - Exact computation results

1.1 - Network Statistics

The tables below shows the calculated values of graph statistics for LSCC and LWCC of each graph:

	Largest Strongly Connected Component			
Graph	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	2.8793	3	9	4
soc-Epinions1	4.4048	4	16	6
ego-Gplus	N/A			
soc-Pokec				

	Largest Weakly Connected Component			
Graph	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	3.2475	3	7	4
soc-Epinions1	4.4536	4	19	6
ego-Gplus	N/A			
soc-Pokec				

As mentioned earlier, we were unable to compute the exact statistics for the last two graphs due to the time and memory restrictions.

1.2 – Running Times (seconds)

For the exact computations we noted the time taken for each step to complete. The following tables shows the running time for each process:

	Largest Strongly Connected Component			
Graph	Reading the edges	Extracting Largest component	Computing Shortest Path Distribution	Total (seconds)
wiki-Vote	0.783	16.5	17.4	34.683
soc-Epinions1	3.81	1261 (21min 01 s)	10575 (2h 56m 15s)	11839.81 (03h 17m 20s)
ego-Gplus	167	2777 (46m 17s)	N/A	
soc-Pokec	246	1665 (27m 45s)		

	Largest Weakly Connected Component			
Graph	Reading the edges	Extracting Largest component	Computing Shortest Path Distribution	Total (seconds)
wiki-Vote	0.783	17.1	157	174.883 (2m 54s)
soc-Epinions1	3.81	1270 (21m 10 s)	4197 (1h 09m 57s)	5470.81 (1h 31m 11s)
ego-Gplus	167	3168 (52m 48s)	N/A	
soc-Pokec	246	1141 (19m 01s)		

TASK 2 – Approximate computation results

2.1 – Approximation Schemes

1. Sampling Random Pairs

Accuracy parameter is the number of random pairs.

	Largest Strongly Connected Component					
Graph	Statistic type	Accuracy Parameter	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	Exact	---	2.8793	3	9	4
	Approx.	5000	2.8902	3	9	4
soc-Epinions1	Exact	---	4.4048	4	16	6
	Approx.	100000	4.4032	4	10	6
ego-Gplus	Exact	---	N/A			
	Approx.	100000	3.3002	3	9	4
soc-Pokec	Exact	---	N/A			
	Approx.	100000	5.11009	5	11	6

	Largest Weakly Connected Component					
Graph	Statistic type	Accuracy Parameter	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	Exact		3.2475	3	7	4
	Approx.	5000	3.3577	3	6	4
soc-Epinions1	Exact		4.4536	4	19	6
	Approx.	100000	4.4147	4	14	6
ego-Gplus	Exact	---	N/A			
	Approx.	100000	3.2692	3	6	4
soc-Pokec	Exact	---	N/A			
	Approx.	100000	4.6829	5	9	6

2. Sampling Random Sources with complete BFS

Accuracy parameter is the number of random sources.

	Largest Strongly Connected Component					
Graph	Statistic type	Accuracy Parameter	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	Exact	---	2.8793	3	9	4
	Approx.	5000	2.8881	3	9	4
soc-Epinions1	Exact	---	4.4048	4	16	6
	Approx.	5000	4.4072	4	15	6
ego-Gplus	Exact	---	N/A			
	Approx.	100	3.4625	3	9	4
soc-Pokec	Exact	---	N/A			
	Approx.	100	5.0996	5	13	6

	Largest Weakly Connected Component					
Graph	Statistic type	Accuracy Parameter	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	Exact		3.2475	3	7	4
	Approx.	5000	3.3293	3	8	4
soc-Epinions1	Exact		4.4536	4	19	6
	Approx.	5000	4.4895	4	19	6
ego-Gplus	Exact	---	N/A			
	Approx.	100	3.1532	3	13	4
soc-Pokec	Exact	---	N/A			
	Approx.	100	4.4735	4	10	5

3. Using ANF for Graph's hop plot approximation

Accuracy parameter is the value of *k*- the number of parallel approximations to get a more robust result. Moreover, *r* – the amount of bits that are added to the length of bitmask to improve the accuracy is 140.

	Largest Strongly Connected Component					
Graph	Statistic type	Accuracy Parameter	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	Exact	---	2.8793	3	9	4
	Approx.	1024	2.2478	2	6	4
soc-Epinions1	Exact	---	4.4048	4	16	6
	Approx.	1024	3.6891	4	11	6
ego-Gplus	Exact	---	N/A			
	Approx.	256	2.6986	3	7	4
soc-Pokec	Exact	---	N/A			
	Approx.	256	4.4854	4	11	6

	Largest Weakly Connected Component					
Graph	Statistic type	Accuracy Parameter	Mean	Median	Diameter	Eff. Diameter
wiki-Vote	Exact		3.2475	3	7	4
	Approx.	1024	3.2372	3	7	4
soc-Epinions1	Exact		4.4536	4	19	6
	Approx.	1024	4.1093	4	11	6
ego-Gplus	Exact	---	N/A			
	Approx.	256	2.7769	3	7	4
soc-Pokec	Exact	---	N/A			
	Approx.	256	4.5100	4	11	5

2.2 – Approximate network statistics versus the accuracy parameter

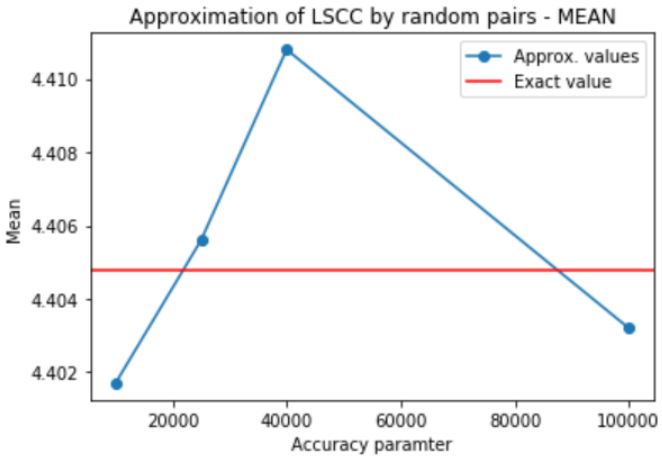
The largest graph for which we have managed to compute the exact statistics is ***soc-Epinions1*** with 75 879 nodes. We are going to perform approximations for the statistics on this graph according to each approximation scheme. In each scheme we will specify multiple accuracy parameters to compare and infer which works converges the best towards the ground truth.

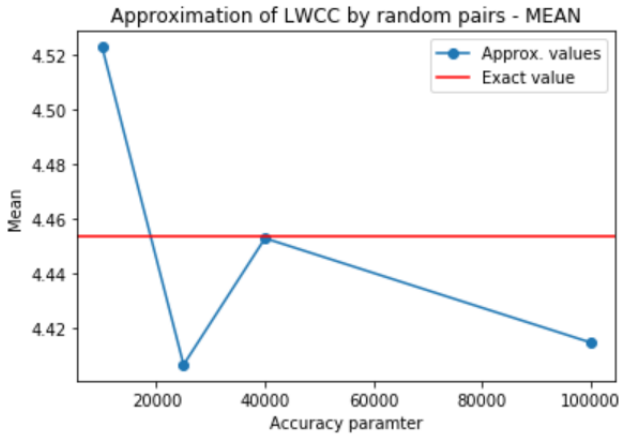
1. Sampling Random Pairs

For this scheme, we have the number of randomly sampled pairs for the shortest distance as our accuracy parameter.

Largest Strongly Connected Component				
Accuracy parameter	Mean	Median	Diameter	Eff. diameter
10000	4.4017	4	11	6
25000	4.4056	4	11	6
40000	4.4108	4	11	6
100000	4.4032	4	10	6
EXACT	4.4048	4	16	6

Largest Weakly Connected Component				
Accuracy parameter	Mean	Median	Diameter	Eff. diameter
10000	4.5229	4	11	6
25000	4.4066	4	11	6
40000	4.4529	4	12	6
100000	4.4147	4	14	6
EXACT	4.4536	4	19	6



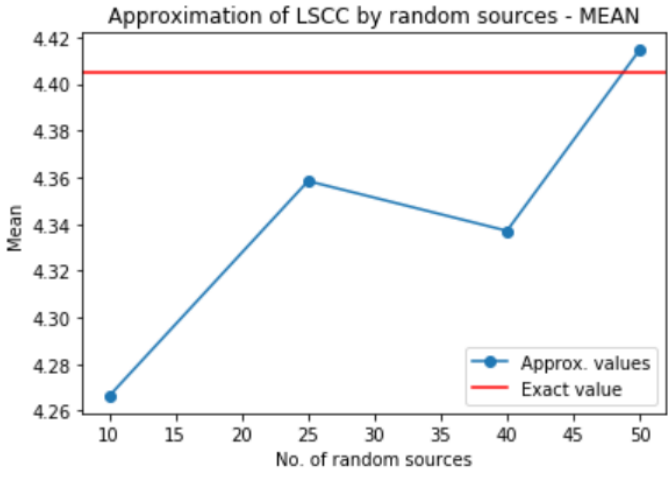


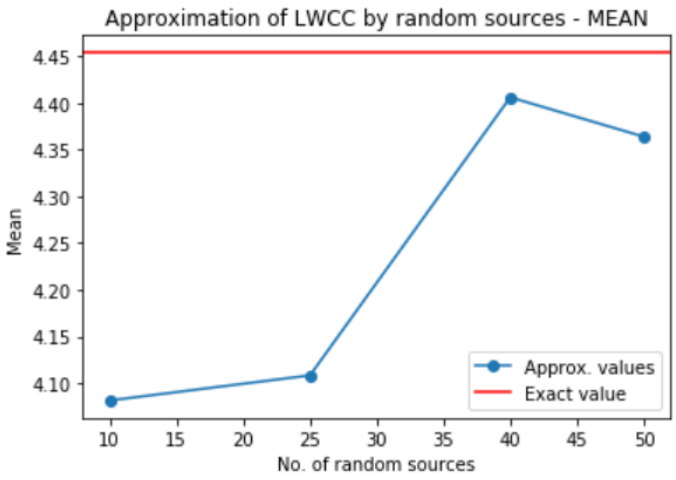
2. Sampling Random Sources with complete BFS

For this scheme, we have the number of randomly sampled sources as our accuracy parameter.

Largest Strongly Connected Component				
Accuracy parameter	Mean	Median	Diameter	Eff. diameter
10	4.2664	4	11	5
25	4.3584	4	13	6
40	4.3371	4	12	6
50	4.4145	4	11	6
EXACT	4.4048	4	16	6

Largest Weakly Connected Component				
Accuracy parameter	Mean	Median	Diameter	Eff. Diameter
10	4.0817	4	10	5
25	4.1085	4	12	6
40	4.4059	4	14	6
50	4.3638	4	16	7
EXACT	4.4536	4	19	6



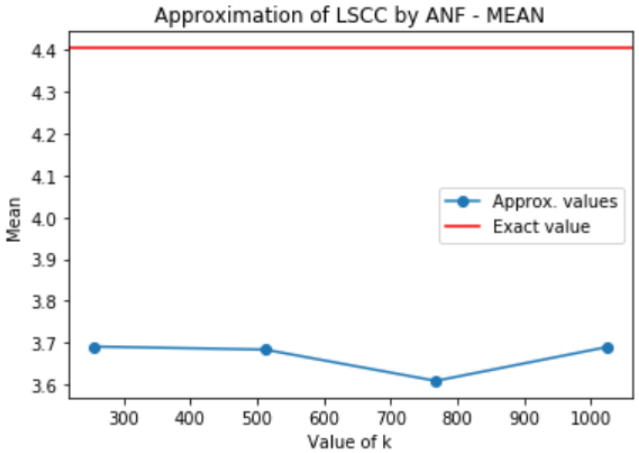


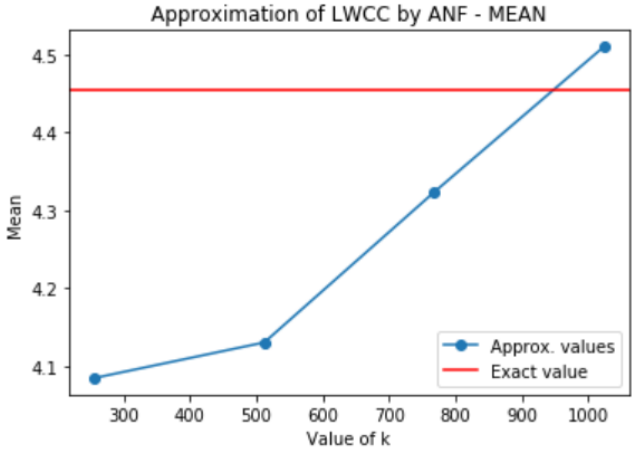
3. Using ANF for Graph's hop plot approximation

For ANF we have the accuracy parameter as the value of *k*- the number of parallel approximations to get a more robust result. Moreover, *r* – the amount of bits that are added to the length of bitmask to improve the accuracy is 140.

Largest Strongly Connected Component				
Accuracy parameter	Mean	Median	Diameter	Eff. diameter
256	3.6905	4	11	6
512	3.6832	4	11	6
768	3.6084	4	11	6
1024	3.6891	4	11	6
EXACT	4.4048	4	16	6

Largest Weakly Connected Component				
Accuracy parameter	Mean	Median	Diameter	Eff. diameter
256	4.0851	4	12	6
512	4.1307	4	14	6
768	4.3234	4	15	7
1024	4.5100	4	16	5
EXACT	4.4536	4	19	6





2.3 – Findings

As observed in the first part of the assignment it is not time efficient to find the exact computations of statistics for large graphs. The cost to find the path distribution of the graph increases exponentially as the size of the graph increases. It is common practice to use approximation schemes for finding graph statistics. In this part we tried using different approximation schemes to find the graph statistics. To improve the accuracy of approximation schemes several number of approximations are performed and average of those approximations is reported as the final statistic.

From our experiments we found that all approximation schemes are pretty good if we consider them on the scale of the time they took as compare to the exact computations time. Results from all three approaches were not very from the exact and hence we can deduce that approximation is valid alternative when the exact computations is either heavy or not possible.

In our findings the Random Sampling Method was the best approximation method as it takes very less time and comes up with very close statistics when compared to the exact computations. But this result is based on the fact that we had exact computations of the second graph for comparison only and since it is not a massive graph, we can't judge ANF approximation performance on it. ANF took the least time for all approximation methods and came close to exact statistics. Hence we can infer that for massive graphs, like the last two; *ego-Gplus* and *soc-Pokec* it will be very feasible.

The recommended accuracy parameter value from our experiments in the Random Pair Sampling approach is 10,000 sample pairs.

References

[1] http://snap.stanford.edu/data/index.html

[2] https://networkx.github.io/

[3] https://networkit.iti.kit.edu/

[4] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. Journal of computer and system sciences, 31(2):182–209, 1985

[5] https://github.com/kit-parco/networkit/blob/Dev/networkit/cpp/distance/HopPlotApproximation.cpp

Furthermore, the solutions to the problems in this project were done with the help of general discussion with classmates, TAs and others. In particular:

Students:

Daniyal Usmani - 662969

Muhammad Rafay Khan - 599838

Haseeb Shehzad - 601483

Abdullah Daniyal - 601250