MATH OVERFLOW

TEMPORAL NETWORK

under the guidance of
Dr.Animesh Chaturvedi

TEAM MEMBERS

- 20bcs041 - chaitanya dharmireddi
- 20bcs062 - janadhana reddy s
- 20bcs068 - kavala nirmal swaroop
- 20bcs084 - vishnu teja malem

TABLE OF CONTENT

INTRODUCTION

- Networks are a fundamental tool for modeling complex systems in a variety of domains including social and communication networks as well as biology and neuroscience.

- The counts of small subgraph patterns in networks, called network motifs, are crucial to understanding the structure and function of these systems. However, the role of network motifs for temporal networks, which contain many timestamped links between nodes, is not well understood.

INTRODUCTION

- Here we develop a notion of a temporal network motif as an elementary unit of temporal networks and provide a general methodology for counting such motifs. We define temporal network motifs as induced subgraphs on sequences of edges, design several fast algorithms for counting temporal network motifs, and prove their runtime complexity.

- We use our algorithms to count temporal network motifs in a variety of real-world datasets. Results show that networks from different domains have significantly different motif frequencies, whereas networks from the same domain tend to have similar motif frequencies.

DATASET

This is a temporal network of interactions on the stack exchange website Math Overflow. There are three different types of interactions represented by a directed edge (u, v, t):

- user u answered user v's question at time t (in the graph sx-mathoverflow-a2q)
- user u commented on user v's question at time t (in the graph sx-mathoverflow-c2q)
- user u commented on user v's answer at time t (in the graph sx-mathoverflow-c2a)

DATASET

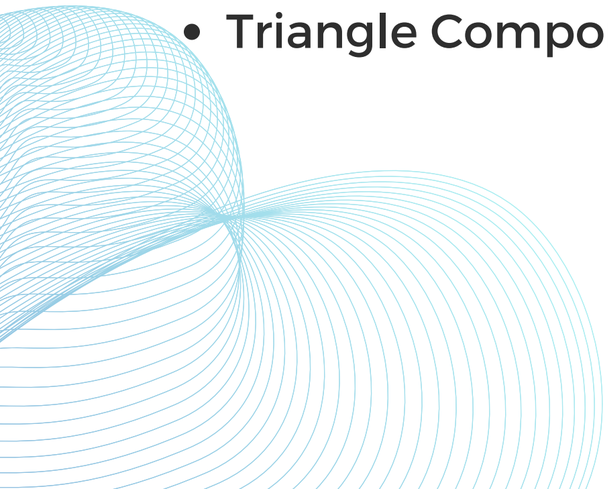
Dataset statistics (sx-mathoverflow-a2q)	
Nodes	21688
Temporal Edges	107581
Edges in static graph	90489
Time span	2350 days

The graph sx-mathoverflow contains the union of these graphs. These graphs were constructed from the Stack Exchange Data Dump. Node ID numbers correspond to the 'OwnerUserId' tag in that data dump.

GRAPH ALGORITHMS

GraphX includes a set of graph algorithms to simplify analytics tasks. The algorithms are contained in the org.apache.spark.graphx.lib package and can be accessed directly as methods on Graph via GraphOps.

- PageRank
- Connected Component
- Triangle Component



PAGE RANK

- PageRank measures the importance of each vertex in a graph, assuming an edge from u to v represents an endorsement of v’s importance by u. For example, if a Twitter user is followed by many others, the user will be ranked highly.

- PageRank is an algorithm used to measure the importance of nodes in a graph by assigning a numerical weight to each node. It evaluates the link structure of the graph and ranks nodes based on their importance.

PAGE RANK

Code Explanation:

- The code first loads a graph from an edge list file using GraphLoader.edgeListFile.
- PageRank algorithm is applied to the graph using graph.pageRank(0.0001), where 0.0001 is the tolerance value.
- The top 20 nodes are then printed based on their PageRank scores.

CONNECTED COMPONENTS

- The connected components algorithm labels each connected component of the graph with the ID of its lowest-numbered vertex. For example, in a social network, connected components can approximate clusters.

- Connected Components algorithm identifies connected subgraphs in the graph where nodes within each subgraph are connected to each other, but not to nodes in other subgraphs.

CONNECTED COMPONENTS

Code Explanation:

- Connected Components algorithm is applied to the graph using graph.connectedComponents().
- The sizes of connected components are calculated, and the top 20 largest components are printed.

TRIANGLE COUNTING

- A vertex is part of a triangle when it has two adjacent vertices with an edge between them. GraphX implements a triangle counting algorithm in the TriangleCount object that determines the number of triangles passing through each vertex, providing a measure of clustering

- Triangle Counting identifies the number of triangles in the graph. A triangle is a set of three nodes where each node is connected to the other two nodes.

TRIANGLE COUNTING

Code Explanation:

- The code calculates the triangle counts for each vertex using the AggregateMessages API in GraphX.
- Triangles are counted for each vertex and divided by 2 as each triangle is counted 3 times in the graph representation.
- The top 20 nodes with the highest triangle counts are then printed.

CONCLUSION

Overall Code :

1. Loading the Graph:
   - The graph is loaded from an edge list file.
2. PageRank Calculation:
   - PageRank algorithm is applied to the graph to rank nodes.
   - Top 20 nodes based on PageRank scores are printed.
3. Connected Components Calculation:
   - Connected Components algorithm is applied to identify connected subgraphs.
   - Sizes of connected components are calculated and the top 20 largest components are printed.
4. Triangle Counting:
   - Triangle counting algorithm is applied to calculate the number of triangles for each node.
   - Top 20 nodes with the highest triangle counts are printed.
5. Spark Context Shutdown:
   - Finally, the Spark context is stopped to release resources.

THANK YOU