**Motivation**

<https://www.frontiersin.org/articles/10.3389/fnins.2019.00585/full>

* Introduction of graph mining
* Applications of graph mining
* Concept of streaming graphs
* Concept of active node (0 or 1)
* Concept of active path or active component over a period of time
* Explain the process how it all goes together
* Explain how to create a synthetic generator
  + Make it more efficient:
    - <http://complexnetworks.fr/wp-content/uploads/2011/01/random.pdf>
* What are potential solutions?
* Description of my naïve solution
* Description of a better solution

The need for Big Data analytics to drive meaningful insights has been on the rise in recent years. The rapid development of data mining techniques has enabled an unprecedented speed and accuracy of making sense of massive amounts of data. Among state-of-the art techniques, Graph Mining is a way to solve Big Data challenges through graph representations. The graphs are especially useful in modelling existing relationships between entities. Platforms that have a readily available network structure, such as the human brain, can largely benefit from graph analytics to drive meaningful insights. For example,

**Positioning**

* Include related work and how we are different
  + K evolving community paper, easy to talk about it

**Problem Statement**

* Explain maintaining active paths over a period of time

**Different Approaches**

* Explain Baseline method and Naïve implementation
* Dikstra’s algorithm for pairwise shortest paths of all nodes
* Explain a better algorithm

**Evaluations**

**Introduction**

The need for Big Data analytics to drive meaningful insights has been on the rise in recent years. The rapid development of data mining techniques has enabled an unprecedented speed and accuracy of making sense of massive amounts of data. Among state-of-the art techniques, Graph Mining is a way to solve Big Data challenges through graph representations. The graphs are especially useful in modelling existing relationships between entities. Platforms that have a readily available network structure, such as social media, can largely benefit from graph analytics to drive meaningful insights.

The objective of this project is to design, develop, and evaluate methods for detecting active paths in streaming graphs. An active path is a rich data source that can be used to derive meaningful information from the data. Insights about active paths can be drawn via frequency estimation of data at the node-level of a graph. Two connected nodes with high frequencies create an active path along the edges connecting them.

Unlike conventional graph analysis which rely on static data, graph mining techniques require data streaming input, giving rise to streaming graphs. Streaming graphs are dynamic in nature where the data input is a continuous and a dynamic stream of data. In order to perform frequency estimation at the node-level, we need new algorithms. This project will investigate learning-based frequency estimation algorithms that use machine learning techniques to provide accurate data streaming results. We will apply such techniques to graphs on the node-level taking dynamic data streams as input.

**Base Line**

Our approach is to maintain the set of connected components in a static graph over time series. So far, I have the following:

* Used Watts-Strogatz model to construct a connected small-world graph with the following properties:
  + Rewiring probability ‘p’: 0.001, 0.01, 0.1
  + Number of neighbors per node ‘k’ = 5
  + Number of nodes ‘n’ = 100 | 1,000 | 10,000
* Created a graph random-walk algorithm to generate synthetic data of active paths across time steps:
  + Every time step duration [t, t’] is calculated based on a parameter k, where t’ is defined as: t + k
  + For every iteration, there is an ‘alpha’ chance to pick a neighbor and a ‘1 – alpha’ chance to stop.
  + Once the synthetic data is ready, it is transformed into a matrix M:
    - Every row ‘i’ represents a node
    - Every column ‘j’ represents a time step
    - Entry at M[i, j] = 1 is an active node at time t
    - Entry at M[i, j] = 0 is an Inactive node at time t
  + Matrix M is then converted into an input file .txt
* For every time step in the input file, I create a subgraph which only consists of active nodes
  + Subgraphs are generated by removing in active nodes which is really O(n) pretty fast
  + Subgraphs are stored as adjacency lists
  + The graph contains many active connected components
* Finally, for each subgraph I generate the list of connected components, which are all active

**Related Work**