Chapter 1

Simulations

1.1 Introduction

In this chapter, several simulations to determine the effectiveness of certain metrics on various types of graphs will be discussed. Python scripts were written to run these simulations and visualize them. These simulations will hopefully give insights on properties of graphs that allow the DSD metric to be more effective concerning the classification problem.

1.2 Bipartite Graph

This section will discuss simulations run starting with a complete bipartite graph to determine the difference in effectiveness between the DSD metric and other metrics (e.g. the shortest path metric) on this graph.

1.2.1 Graph Construction

The graph used in the simulations in this section was constructed starting with a $K_{n,m}$ complete bipartite graph. This was done in order to initially label the nodes of the graph in a meaningful manner. All nodes in one set (size n) of the bipartite graph were given the same label, and all nodes not in this set (in the other set of size m) were given a label different from that of the first set.

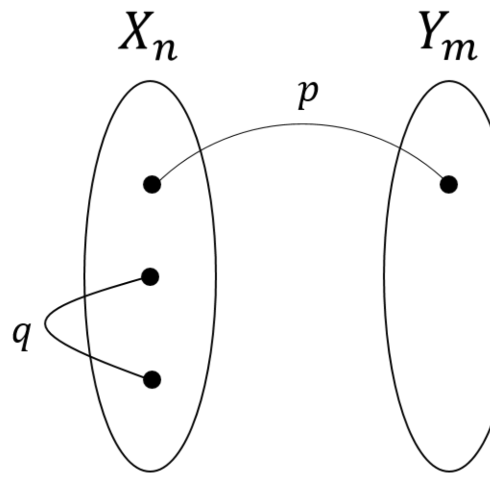

Figure 1.1: An illustration of the initial complete bipartite graph used in the simulation with labeled parameters.

The two disjoint and independent sets of the initial $K_{n,m}$ complete bipartite graph are represented by X_n and Y_m, as shown in Figure 1.1. The number of nodes in each cluster are n and m, respectively, and were set as parameters.

In order to introduce randomness to this graph, edges were added and removed from the initial graph, $G = (V, E)$. Figure 1.1 illustrates this procedure. For each edge in the initial graph $e = (u, v) \epsilon E$, $u \epsilon X_n$, $v \epsilon Y_m$, the edge e was removed with a probability p. For each node in the disjoint set of the initial graph, X_n, edges were added between pairs of nodes $u \epsilon X_n$ and $v \epsilon X_n$ with a probability q. A similar procedure was done with nodes from the disjoint set of the initial graph, Y_m. The two probabilities, p and q, were set as parameters as well.

1.2.2 Data Collection

In order to collect data, a Python script was run to construct the graph described previously given the parameters n, m, p, and q. Figure 1.2 shows an example of a constructed graph with parameters $n = 10$, $m = 10$, $p = 0.4$, and $q = 0.1$. This graph was stored and some percentage of the node labels were censored. The edges of this graph were then written to a .csv file for processing by another script that calculated the top k closest nodes for each node using the DSD metric and the shortest path distance metric. These distances were used to classify censored nodes and the number of correctly classified nodes was recorded. The percent of the censored nodes that were classified correctly was compared between metrics to see whether or not there was a significant difference in correctness of classification.

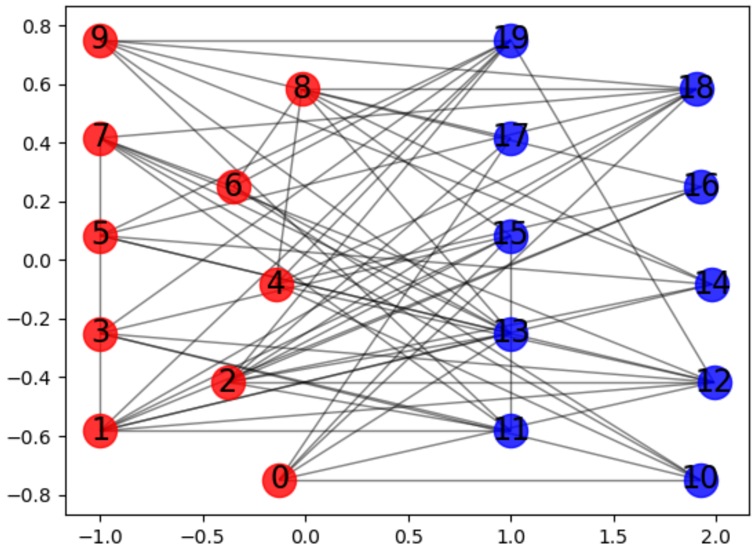

Figure 1.2: A visualization of the graph constructed from the previous section with 63 edges.

1.2.3 Analysis

Compare shortest paths distances data with DSD data. Analyze results.