Lab Report 2: **Analysis of Graph Algorithms**

Akash Santhanakrishnan, Paarth Kadakia, Virendra Jethra

February 27 2023

Contents

1 Introduction **1**
- 1.1 Overview . . . 1
- 1.2 Purpose . . . 1

2 Executive Summary **1**

3 Lab Part 1 **1**
- 3.1 Experiment 1 . . . 1
- 3.2 Experiment 2 . . . 2

4 Lab Part 2 **3**
- 4.1 Approximation Experiments: . . . 3
  - 4.1.1 Experiment 1 . . . 3
  - 4.1.2 Experiment 2 . . . 4
  - 4.1.3 Experiment 3 . . . 5
- 4.2 Indepedent Set: . . . 8

5 Appendix **9**

List of Figures

Figure 1 . . . 2
Figure 2 . . . 3
Figure 3 . . . 4
Figure 4 . . . 5
Figure 5 . . . 6
Figure 6 . . . 7
Figure 7 . . . 8
Figure 8 . . . 9

1 Introduction

1.1 Overview

The lab focuses on implementing, analyzing, and optimizing graph algorithms and their properties. The lab will span two weeks and cover:

- BFS/DFS algorithms that return paths instead of boolean values
- Does a graph have a cycle and/or is connected
- Strategy/Implementation to create random graphs
- Graphs having certain properties
- Vertex Cover and Independent Set problems
- Brute force algorithms for these two problems
- Design approximating algorithms for these problems
- Experiments to gauge the accuracy of approximation algorithms

1.2 Purpose

This lab report conducts and discusses a variety of different experiments with different graph algorithms and deals with specific types of problems. There will be code, graphs, and other visuals to supplement the analysis of these algorithms.

2 Executive Summary

The lab report we wrote up consisted of experiments the were devised to find the accuracy of certain graph algorithms. Part 1 discusses BFS and DFS algorithms to return the path of the search instead of a boolean value. Then we showed graphs consisting of the chances of cycles being produced given the number of edges and connected graphs being produced as well. There was found to be a positive correlation between the number of edges and the probability of each of such occurring as well. Part 2 of the lab we were given certain parameters to construct 3 approximation functions. Each of these functions delivers its best possible output of MVC. Approximation 1 is by far the best approximation for the MVC. The lab tests show that the approximation algorithms are efficient in generating reliable results for the Independent Set problem, however, the Vertex Cover problem still exists.

3 Lab Part 1

3.1 Experiment 1

Outline:

- Comparing cycle probability with the number of edges in a graph
- One graph | Number of nodes: **100**, Number of edges: **100**, Number of runs: **100**, Step: **1**

Graphs:

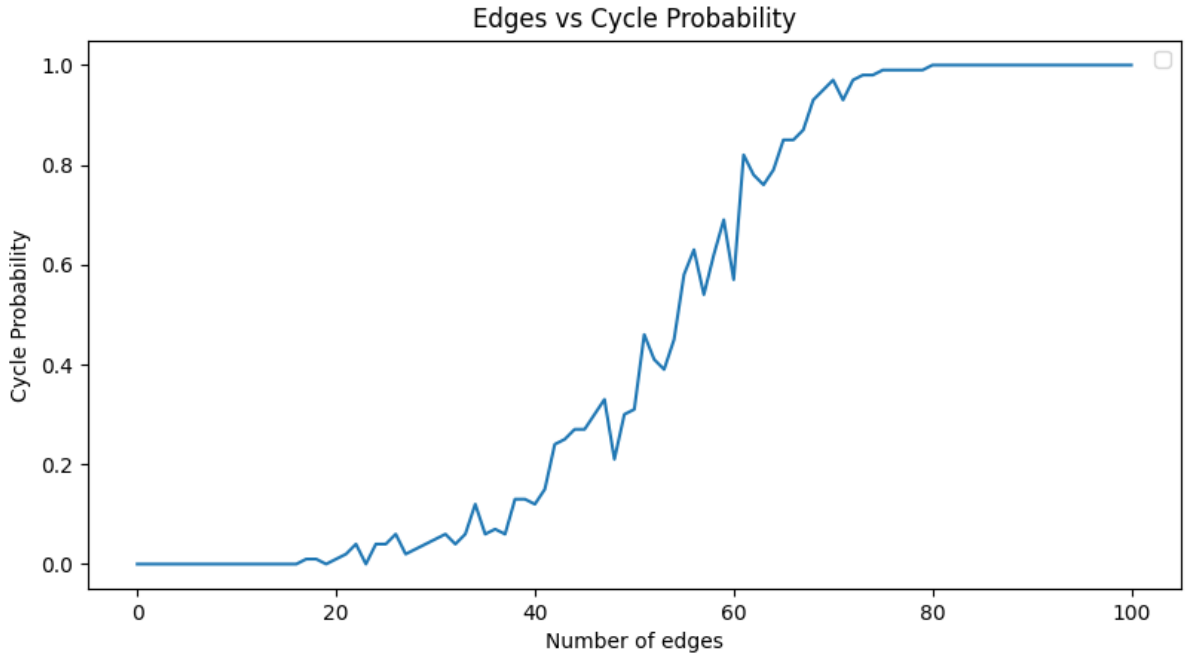

Figure 1: Cycle Probability vs Number of Edges

Conclusion:
This graph shows the cycle probability of an undirected graph with constant nodes increases as the number of edges increases. This makes sense as there are more paths between different nodes to form a cycle.

3.2 Experiment 2

Outline:

- Comparing connected probability with the number of edges in a graph
- One graph | Number of nodes: **100**, Number of edges: **500**, Number of runs: **100**, Step: **10**

Graphs:

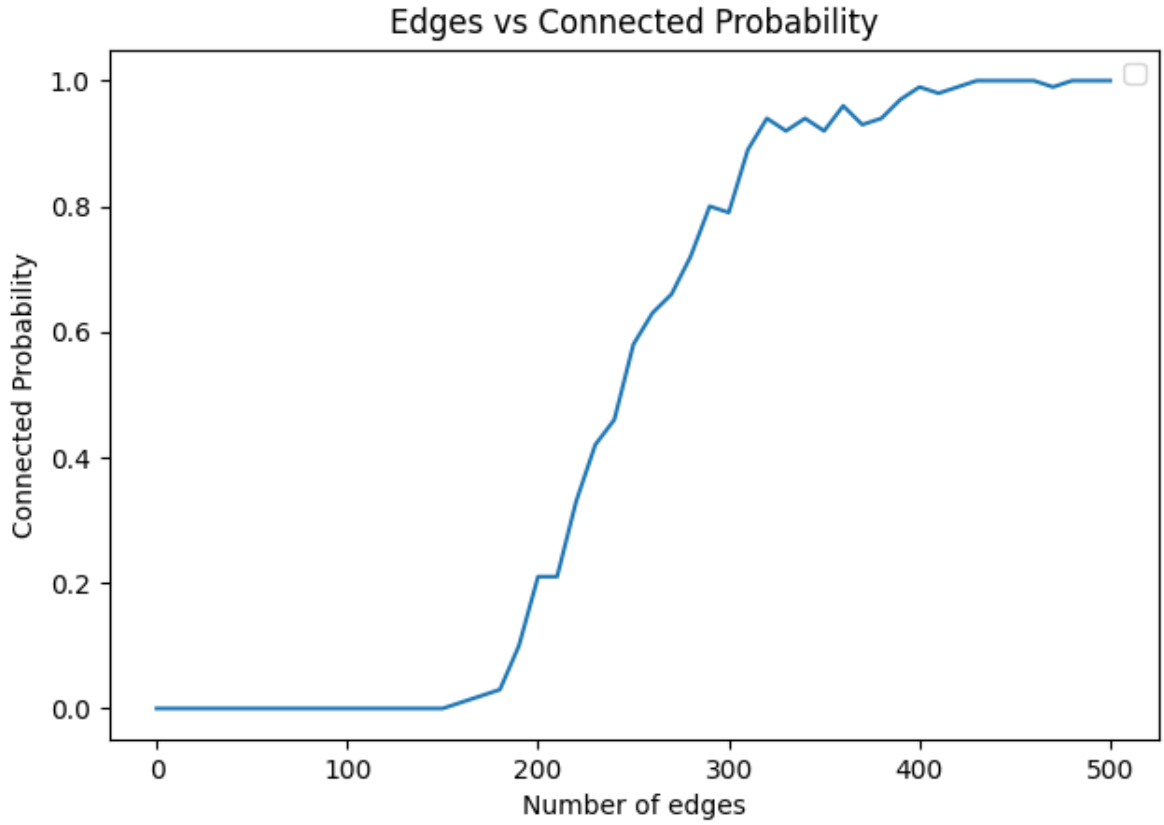

Figure 2: Connected Probability vs Number of Edges

Conclusion:
It seems that the connected probability is also proportional to the number of edges, however, this probability only starts to increase when the number of edges reaches around 180-190. One possible reason for this is that the test graphs have a certain threshold of being disconnected until it does reach a point of being connected. Once it becomes connected, the probability of being connected increases as you add edges because there are more opportunities for the new edges to connect with existing components of the graph.

4 Lab Part 2

4.1 Approximation Experiments:

4.1.1 Experiment 1

Outline:

- Comparing all three approximation algorithms on the ratio of list size vs number of edges
- One graph | Number of nodes: **8**, Max edges: **60**, Number of runs: **1000**

Graph:

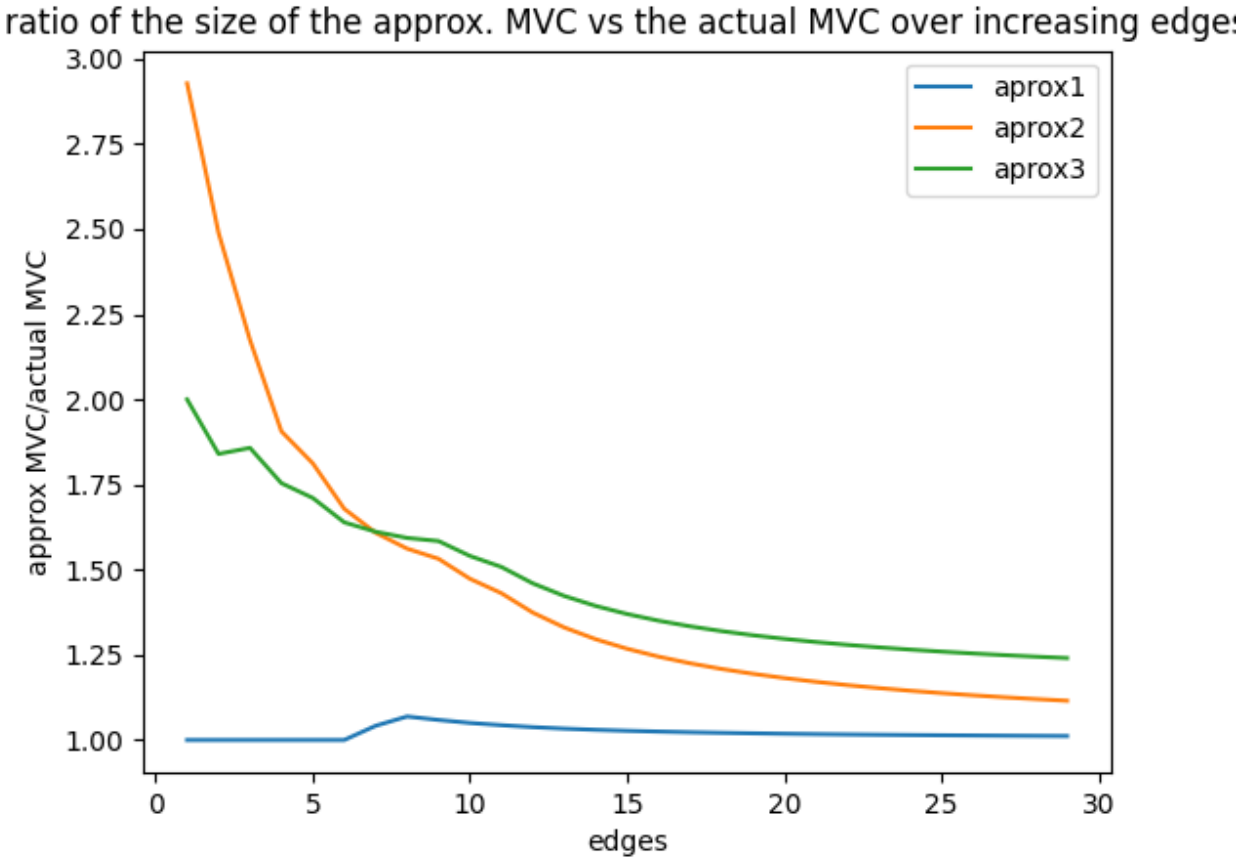

Figure 3: Approximation Experiment 1

Conclusion:
For Approx1, since this algorithm prioritizes high-degree vertices, it is possible that for small graphs with a maximum of 30 edges, the algorithm quickly identifies the most effective vertices to include in the vertex cover, which explains why it consistently produces the smallest sizes. For approx2, the algorithm selects random vertices, so its performance is more varied than the other algorithms. However, since these graphs are small with a low number of edges, the likelihood of it selecting a high-degree vertex, despite the randomness is balanced. One thing to recognize is that the rate of decrease is slow for approx2, especially when the number of edges increases because there are fewer high-degree vertices to select from so its performance worsens. Approx3 selects pair of adjacent vertices instead of just individual vertices, which is why it becomes accurate as the number of edges increases.

4.1.2 Experiment 2

Outline:

- Comparing all three approximation algorithms on the ratio of list size vs number of edges
- One graph | Number of Nodes: **20**, Max edges: **20**, Number of runs: **100**

Graph:

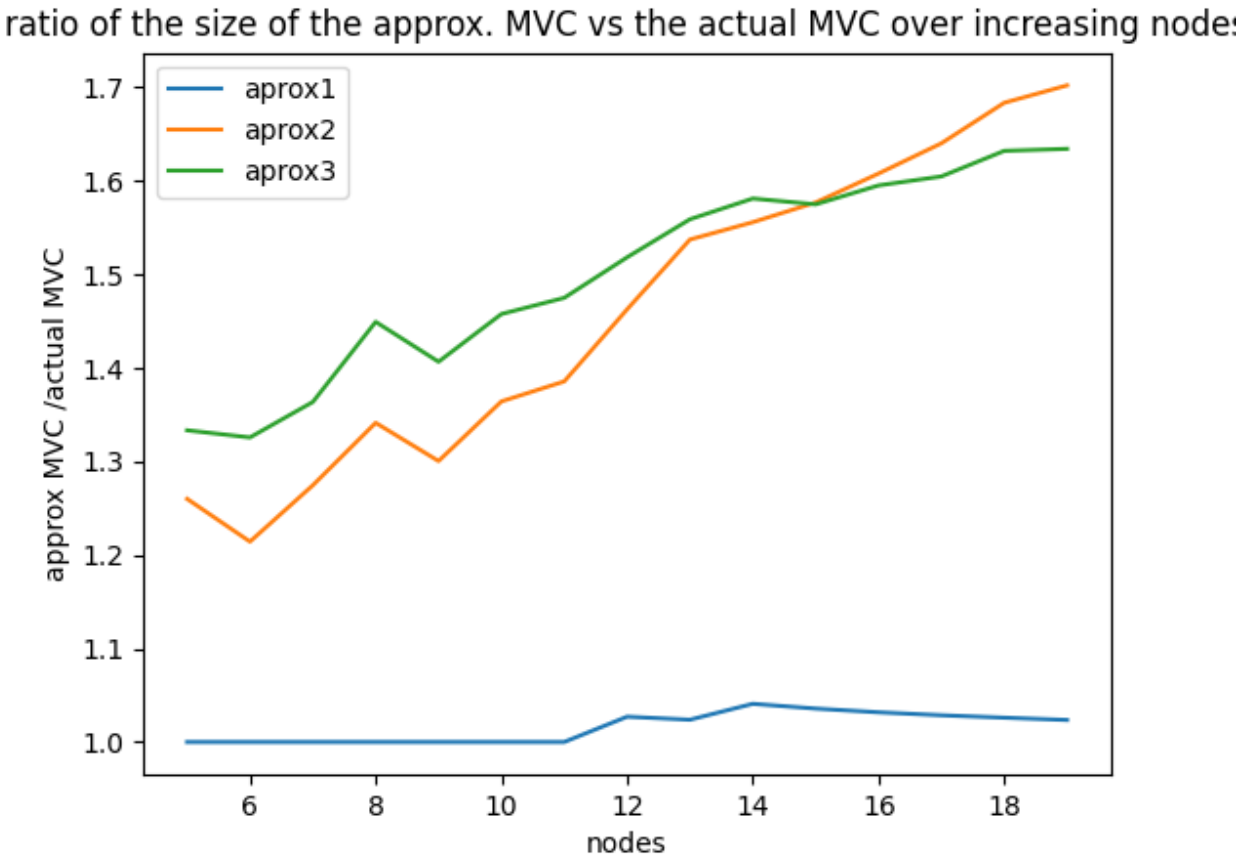

Figure 4: Approximation Experiment 2

Conclusion:
For experiment 1B we are comparing the size of the approximations vs the size of the actual MVC over increasing node values and plotting these values down to find the accuracy of the approximations. As we can see the inaccuracy of the approximations increase over time as the number of nodes increases as well. This makes sense in the fact that increasing the total number of nodes will increase the randomness odds that approx2 and approx3 will be picking out of the graph.

4.1.3 Experiment 3

Outline:

- Comparing connected probability with the number of edges in a graph
- One graph | Number of nodes: **100**, Number of edges: **500**, Number of runs: **100**, Step: **10**

Graphs:

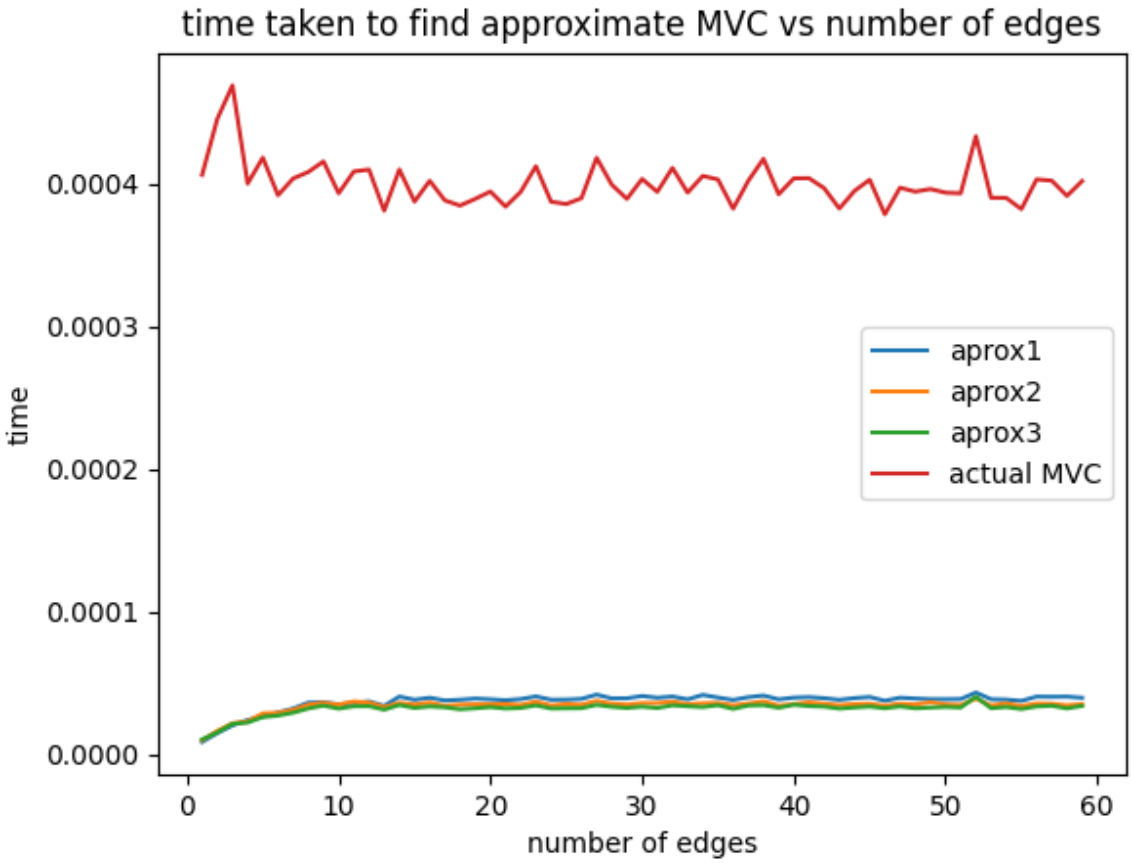

Figure 5: Approximation Experiment 3

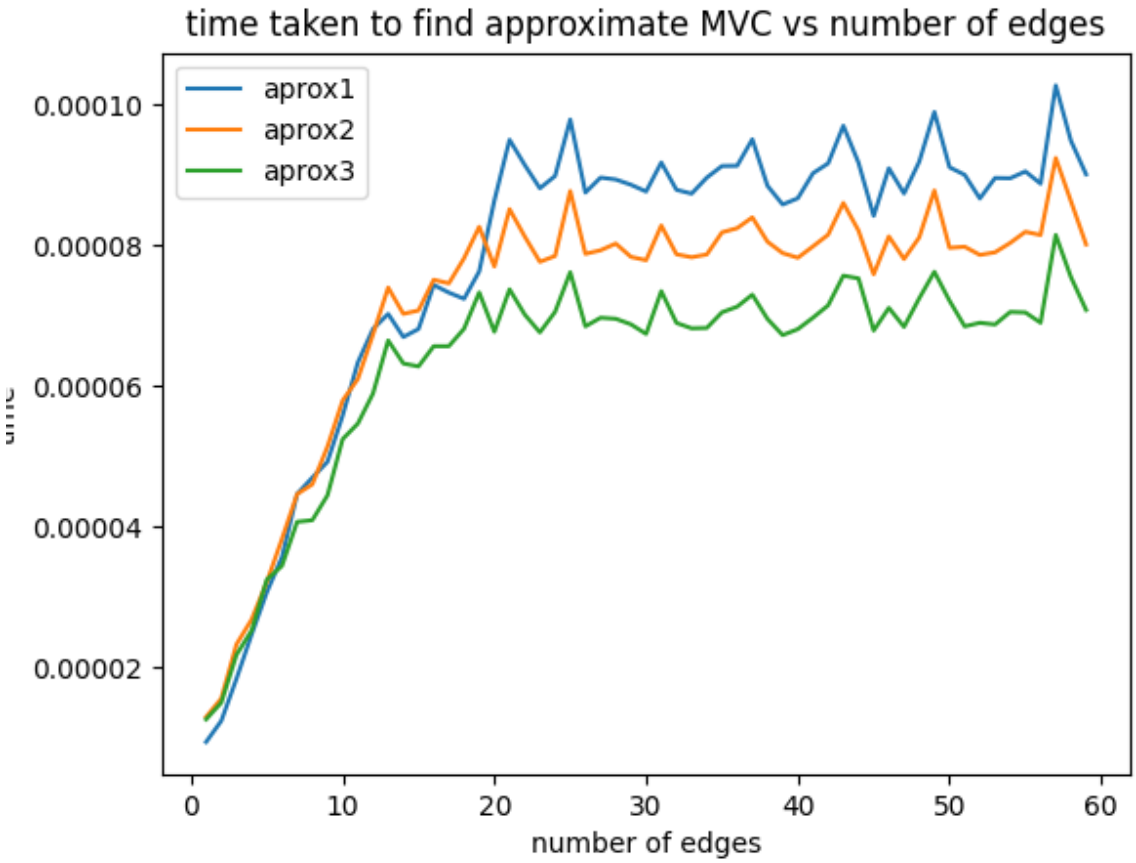

Figure 6: Approximation Experiment 3

Conclusion:
Approx1 involves nested loops and chooses the highest degree vertex in each iteration, making it the slowest among the three. Approx3 is the fastest algorithm among the three, followed by Approx2 and then Approx1. This might be because Approx3 chooses two random vertices and removes them from the graph, which reduces the search space faster than the other two algorithms. Approx2, on the other hand, chooses a random vertex in each iteration and adds it to the vertex cover, which might not be efficient in some cases. Overall, the graph is expected to show an increasing trend in time taken with an increase in the number of edges, but the rate of increase would differ for each algorithm

4.2 Indepedent Set:

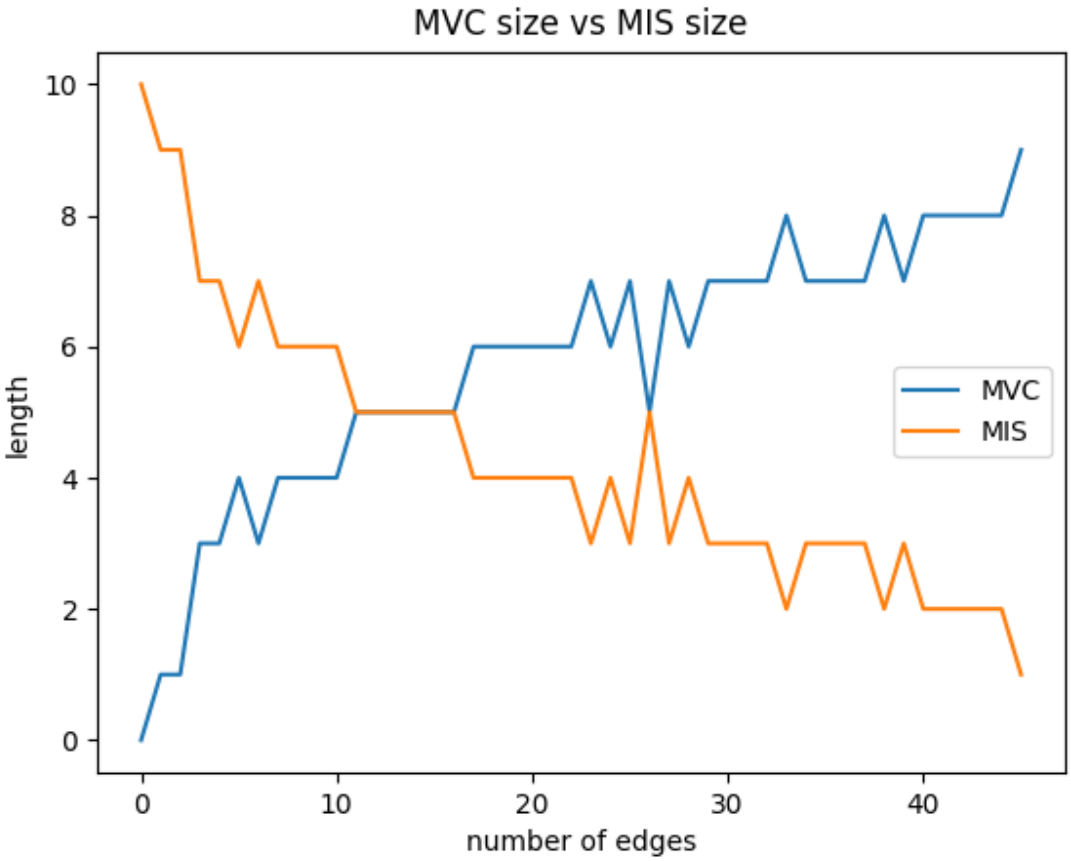

Figure 7: Independent Experiment 1A

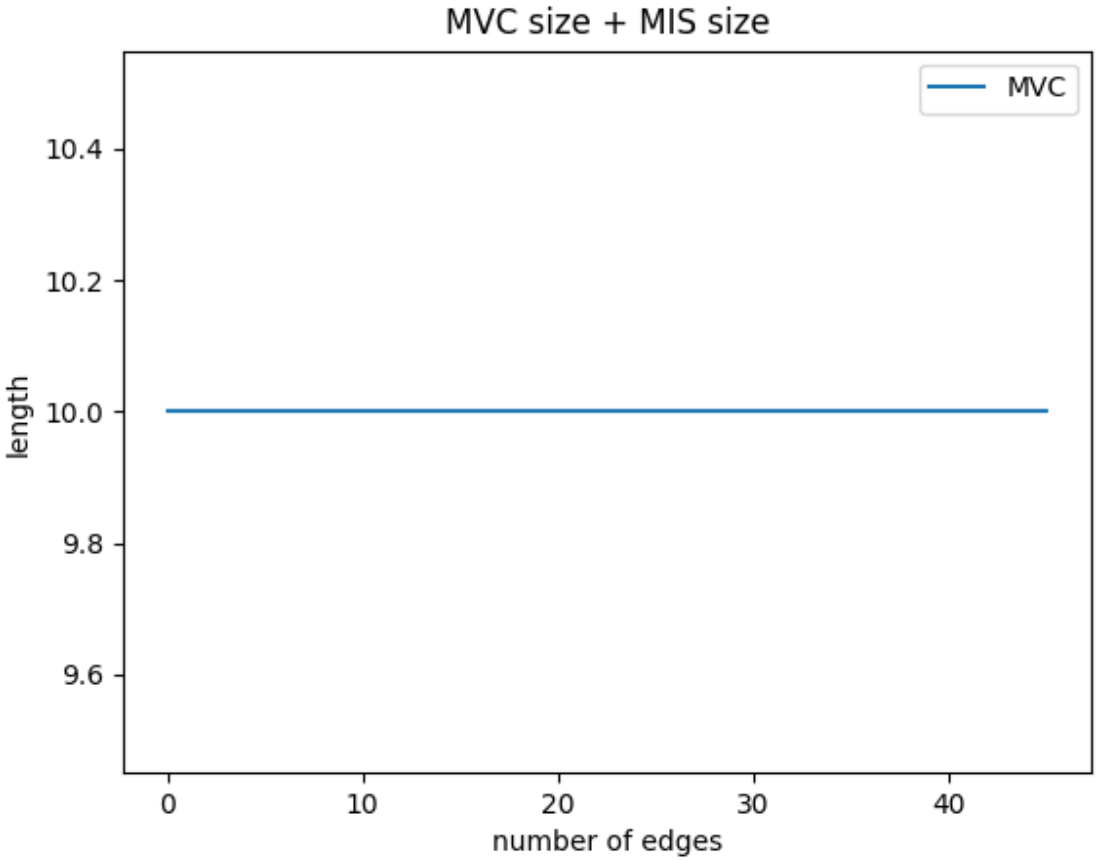

Figure 8: Independent Set Experiment 1B

Conclusion:
The algorithm Approx3 is the fastest algorithm because it chooses two random vertices and removes them from the graph, which reduces the search space faster than the other two approx functions. On the other hand, Approx2 chooses a random vertex in each iteration and adds it to the vertex cover, which might not be as efficient as Approx3. Lastly, Approx1 involves nested loops and chooses the highest degree vertex in each iteration, making it the slowest among all three functions. Overall, the graph is expected to show an increasing trend in time taken with an increase in the number of edges, but the rate of increase would differ for each algorithm

5 Appendix

Code for all experiments can be found on this GitHub repository. There are two files one for Part 1 and one for Part 2. Each experiment corresponds to the method number, e.g. Experiment 1 corresponds to the method "experiment1" in the file. Additional sorting algorithms are added underneath the original algorithms to maintain organization and easy navigation.