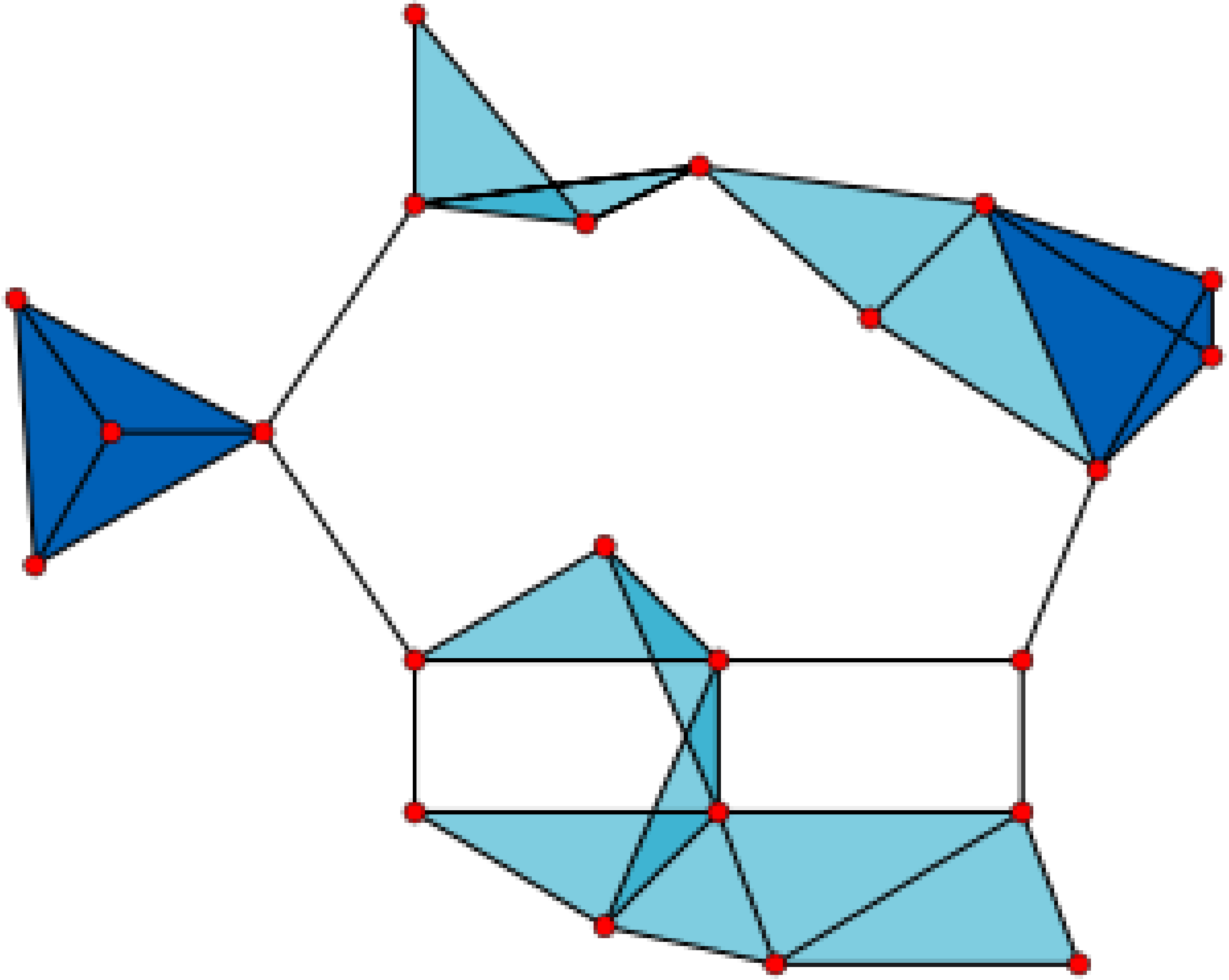


Searching for Clique-k

ALPER GIRAY, FURKAN ÖZHAN, ONAT KUTLU, ONUR ARDA BODUR, ÖMER SEZER KOYUNCU

Outline

PROBLEM DESCRIPTION

ALGORITHM DESCRIPTION

ALGORITHM ANALYSIS

EXPERIMENTAL ANALYSIS

TESTING

CONCLUSION

Problem Description (1)

Clique as a word means a small group of people , with shared interests or other features in common, who spend time together and do not readily allow others to join them.

In the context of computer science, clique is a subset of vertices in an undirected graph that has all the vertices adjacent to each other which means they are all connected by edges to each other.

Clique problem is the computational problem of finding cliques. Based on the need and given parameters, there may be different formulations in the problem.

Common formulations of the clique problem include finding a maximum clique which is a clique with the largest possible number of vertices, finding a maximum weight clique in a weighted graph, or listing all maximal cliques that are cliques that cannot be enlarged.

Problem Description (2)

Clique computation is used in various fields, with bioinformatics being the most common. Other than modeling in bioinformatics, it is also used in electrical engineering and chemistry fields.

In bioinformatics, the problem of clustering gene expression data requires finding the minimum number of changes needed to transform a graph describing the data into a graph formed as the disjoint union of cliques.

In electrical engineering, cliques are used to analyze communications networks, and to design efficient circuits for computing partially specified Boolean functions.

In chemistry, cliques are used to describe chemicals in a chemical database that have a high degree of similarity with a target structure.

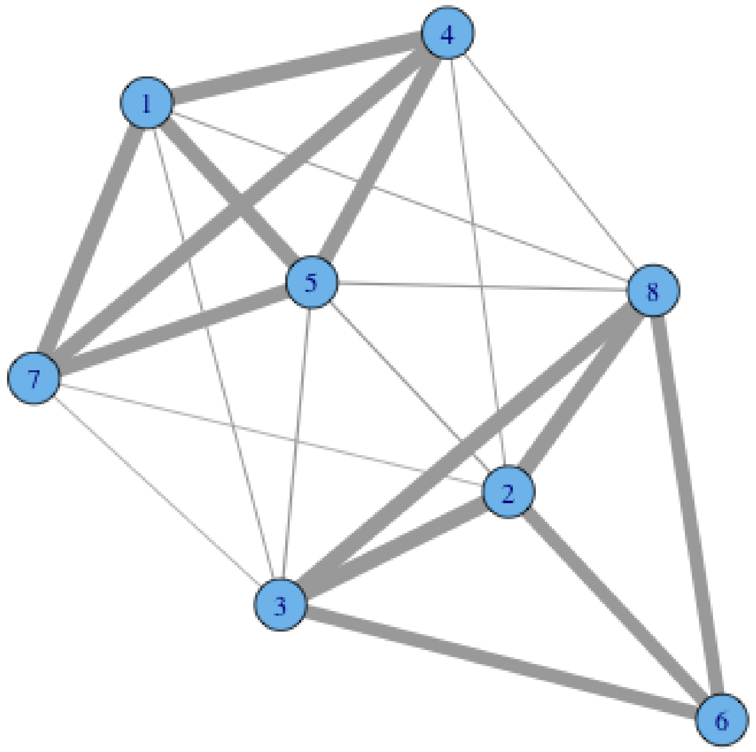
Problem Description (3)

In this project’s context, the problem specified is finding a clique with a requested size. It is described more formally below:

- **Input:** *We have an n-node undirected graph G (V, E) with node set V and edge set E; a positive integer k with k ≤ n.*

- **Problem:** *Does G contain a k-clique, i.e. a subset W of the nodes V such that W*

  has size k and for each distinct pair of nodes u, v in W, {u, v} is an edge of G?



About Problem

- The problem of finding the maximum clique is both fixed-parameter intractable and hard to approximate. And, listing all maximal cliques may require exponential time as there exist graphs with exponentially many maximal cliques.

Algorithm Description

```
def p_clique(graph,k):
  if k == 1:
     return True
  clique = []
  vertices = get_keys_array(graph)
  max_ln=0
  max_dict=[]
  for i in range(0,len(graph)):
    clique=[]
    clique.append(vertices[i])
    for v in vertices:
       if v in clique:
          continue
       isNext = True
       for u in clique:
          if u in graph[v]:
            continue
          else:
            isNext = False
            break
       if isNext:
         clique.append(v)
         if k <= len(clique):
                return True
    if len(clique)>max_ln:
      max_ln=len(clique)
      max_dict=clique
  if k <= len(clique)
     return True
  else:
     return False
```

Algorithm Analysis

```
  clique = []
  vertices =
get_keys_array(graph)
  max_ln=0
  max_dict=[]
  for i in range(0,len(graph)):
    clique=[]
    clique.append(vertices[i])
    for v in vertices:
        if v in clique:
            continue
        isNext = True
        for u in clique:
            if u in graph[v]:
              continue
            else:
              isNext = False
              break
        if isNext:
          clique.append(v)
          if k <= len(clique):
            return True
    if len(clique)>max_ln:
      max_ln=len(clique)
      max_dict=clique
  if k <= len(clique):
            return True
  else :
            return False
```

$O(V)$

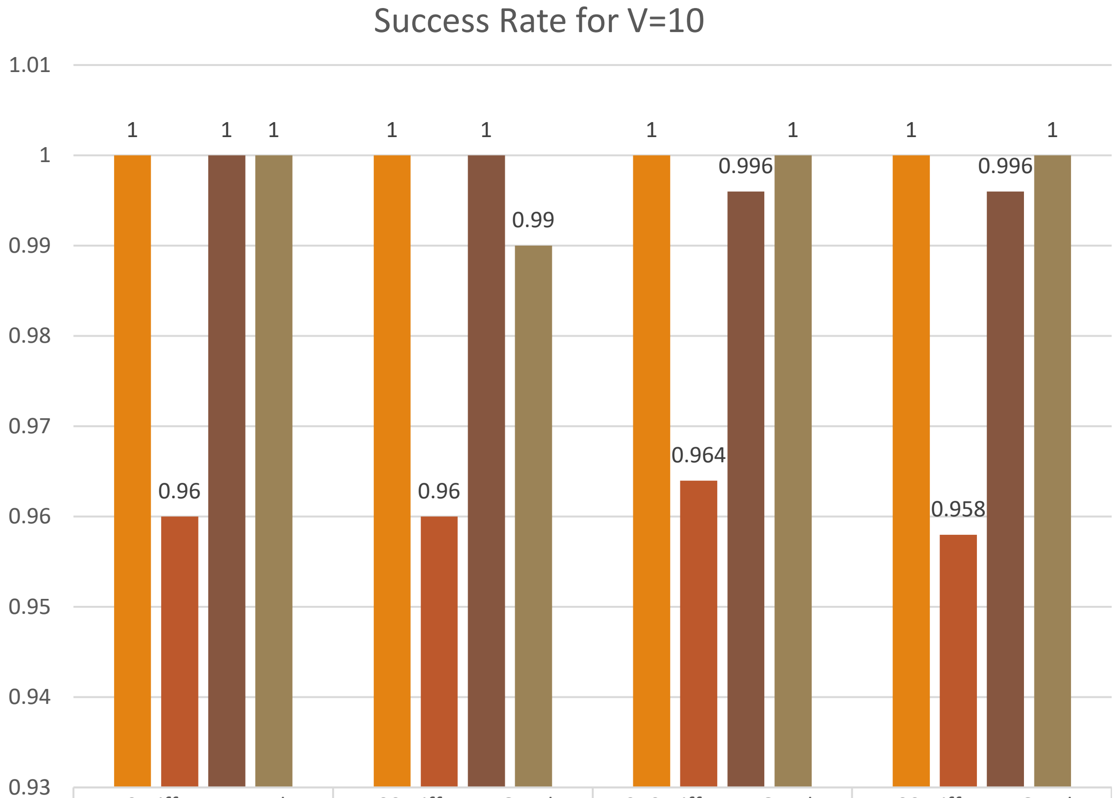
$O(V*V)$

$O(V*V*k)$

Running Time: O(V^2 + k)

Experimental Analysis

We tested the algorithm with randomly generated graphs of sizes 10, 15, 20, 30 with k values of 3, 4, 5, 6 for 50, 100, 250, 500 times in sequence.

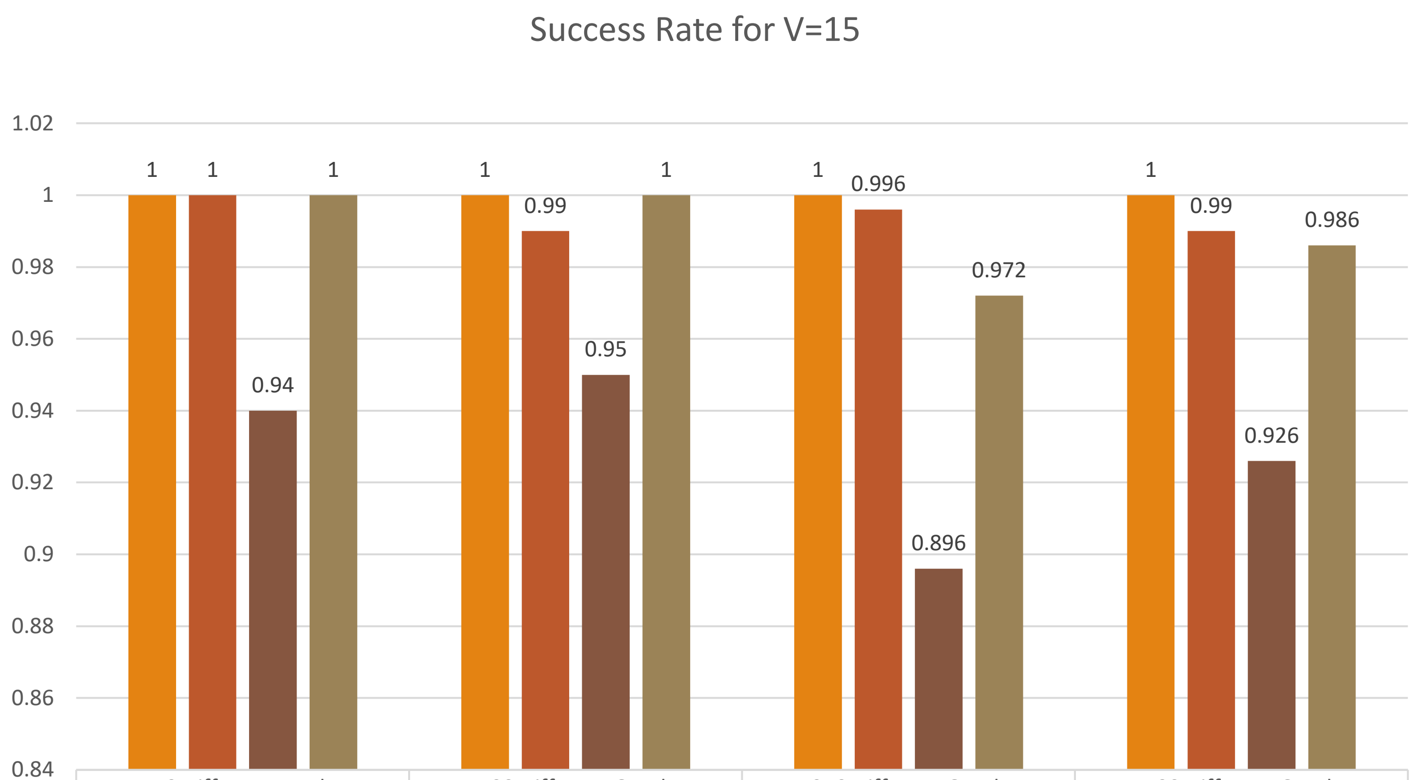


	50 Different graph	100 Different Graph	250 Different Graph	500 Different Graph
k =3	1	1	1	1
k=4	0.96	0.96	0.964	0.958
k=5	1	1	0.996	0.996
k=6	1	0.99	1	1

Success Rates

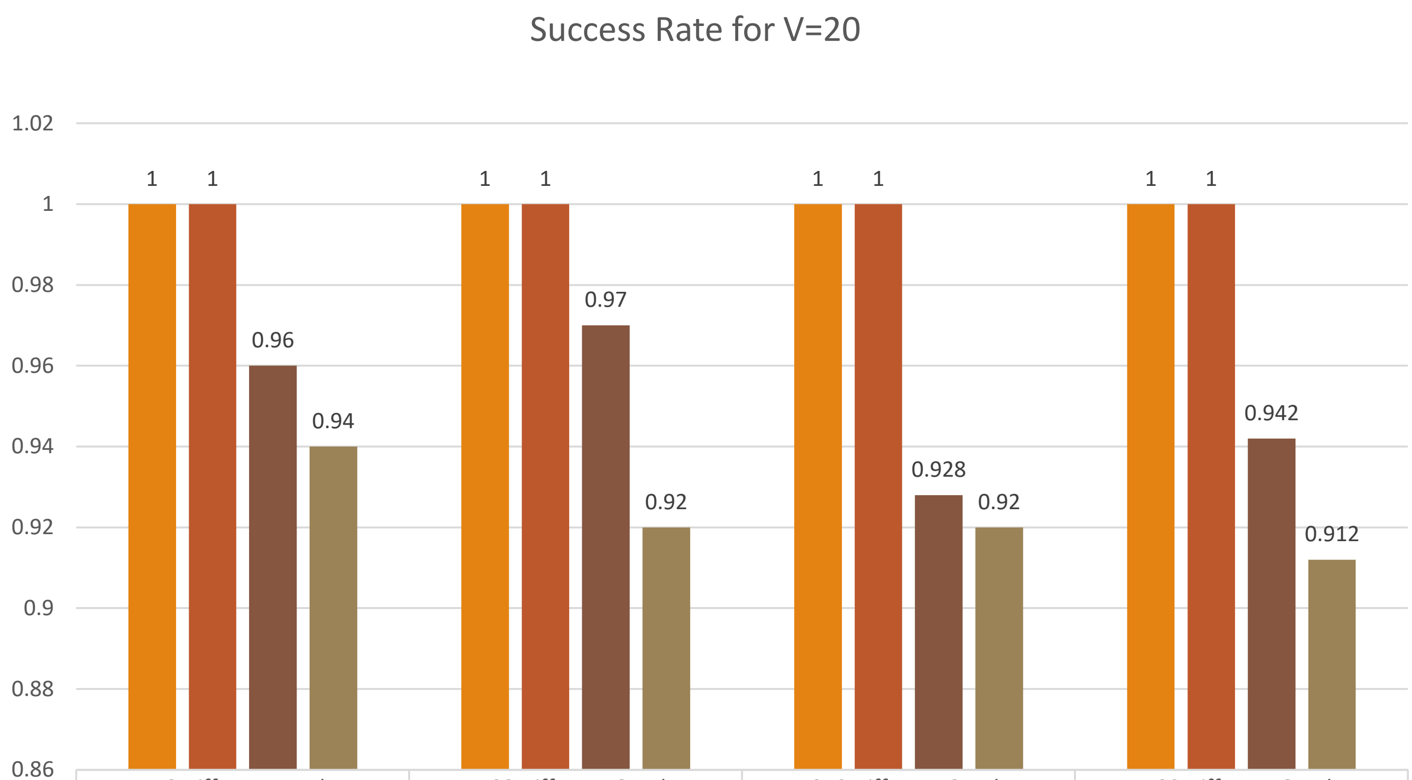
Success Rates are determined as if our algorithm can find k-clique precisely.

Success Rates

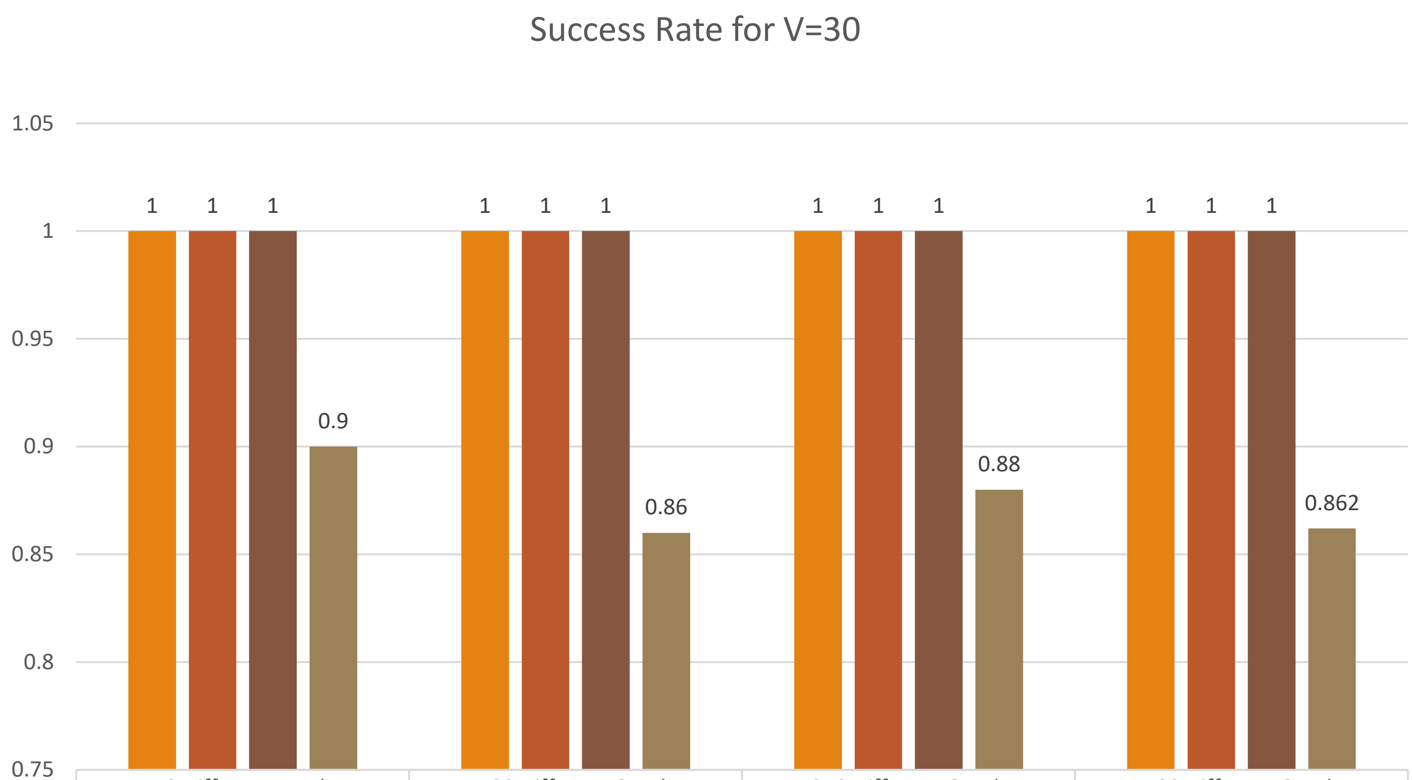


	50 Different graph	100 Different Graph	250 Different Graph	500 Different Graph
k =3	1	1	1	1
k=4	1	0.99	0.996	0.99
k=5	0.94	0.95	0.896	0.926
k=6	1	1	0.972	0.986

Success Rates

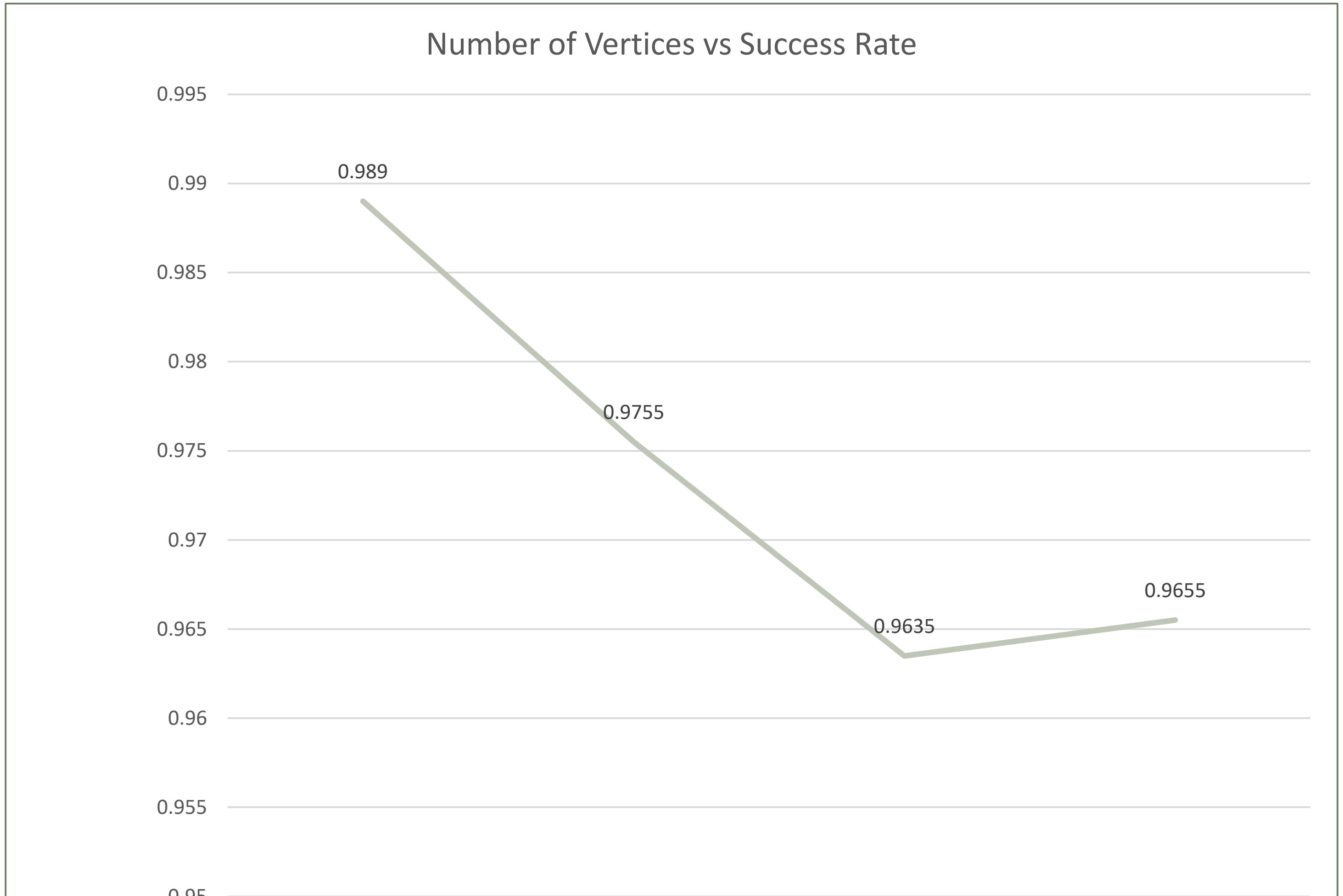


	50 Different graph	100 Different Graph	250 Different Graph	500 Different Graph
k =3	1	1	1	1
k=4	1	1	1	1
k=5	0.96	0.97	0.928	0.942
k=6	0.94	0.92	0.92	0.912



	50 Different graph	100 Different Graph	250 Different Graph	500 Different Graph
k =3	1	1	1	1
k=4	1	1	1	1
k=5	1	1	1	1
k=6	0.9	0.86	0.88	0.862

Success Rates



	V=10	V=15	V=20	V=30
Success Rate	0.989	0.9755	0.9635	0.9655

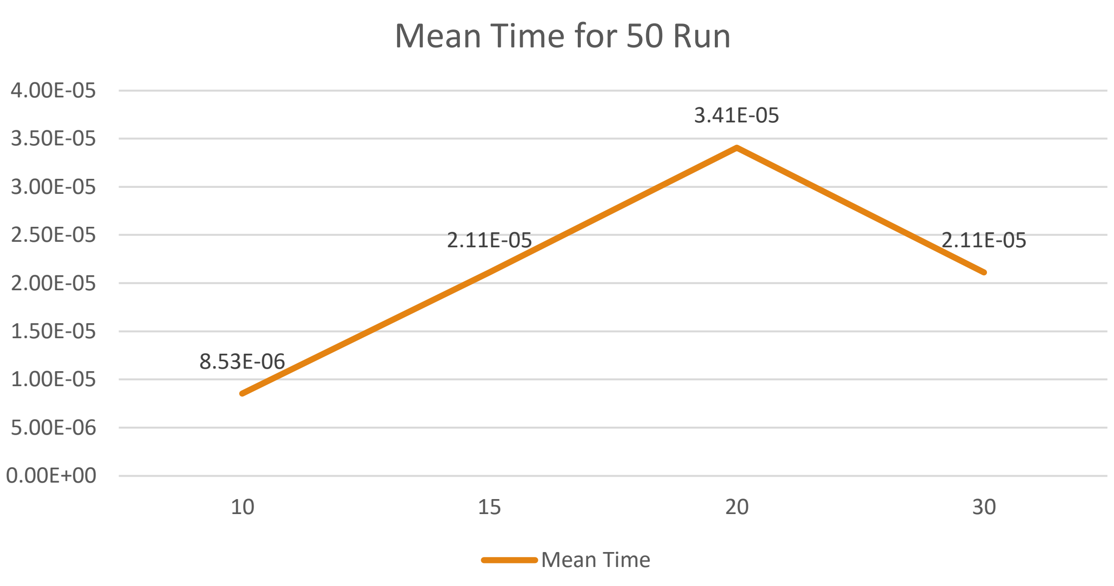
Number of Vertices vs Success Rate

Running Time Experimental Measurement

To experimentally analyze running time of our algorithm we created some functions to help us to interpret some statistic concepts using python libraries. We fixed our K value to 5 because we want to find the effect of only the number of vertices to running time.

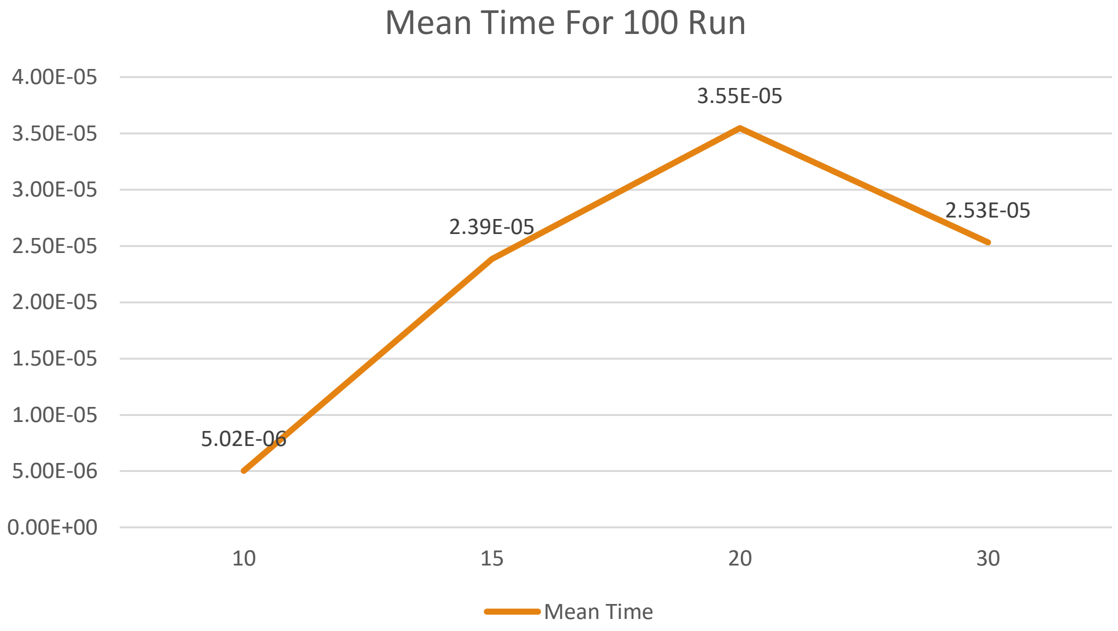
50 Run Per Input Size

Size	Mean Time	Standard Deviation	Standard Error	%90-CL	%95-CL
10	8.534e-06	2.75e-05	1.21e-06	2.95e-05 - 2.55e-05	2.99e-05 - 2.51e-05
15	2.110e-05	3.507e-05	3.015e-06	4.003e-05 - 3.011e-05	4.098e-05 - 2.9166e-05
20	3.407e-05,	4.050e-05	4.867e-06	4.851e-05 - 3.249e-05	5.004e-05 - 3.096e-05
30	2.111e-05	2.256e-05	3.016e-06	2.752e-05 - 1.760e-05	2.847e-05 - 1.665e-05



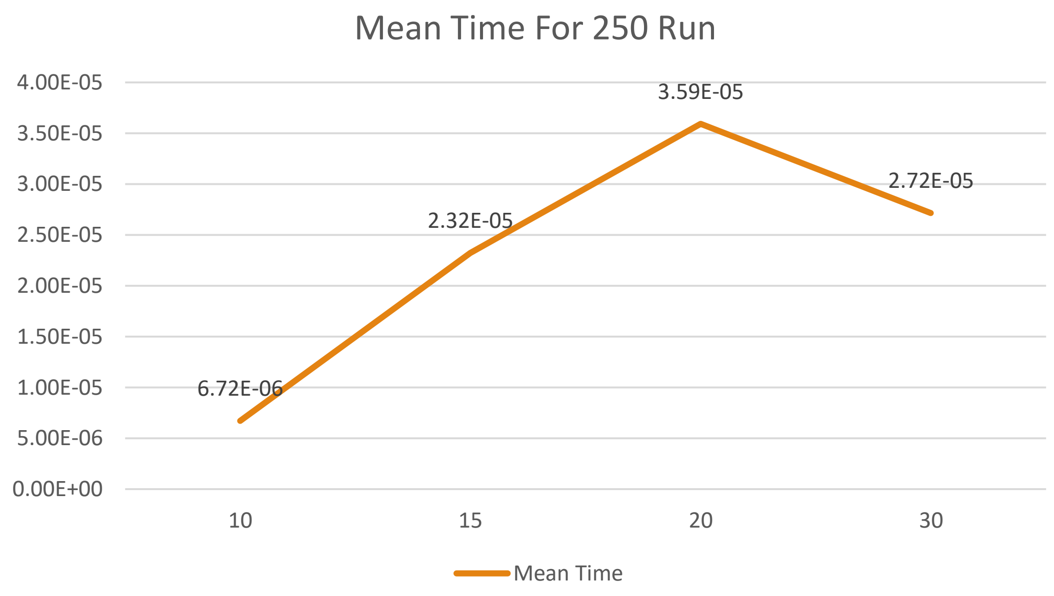
100 Run Per Input Size

Size	Mean Time	Standard Deviation	Standard Error	%90-CL	%95-CL
10	5.016e-06	2.57e-05	5.04e-07	2.65e-05 - 2.49e-05	2.673e-05 - 2.476e-05
15	2.385e-05	3.368e-05	2.397e-06	3.7625e-05 - 2.9737e-05	3.838e-05 - 2.898e-05
20	3.547e-05	4.064e-05	3.564e-06	4.650e-05 - 3.478e-05	4.763e-05 - 3.365e-05
30	2.533e-05	2.420e-05	2.546e-06	2.839e-05 - 2.001e-05	2.919e-05 - 1.921e-05



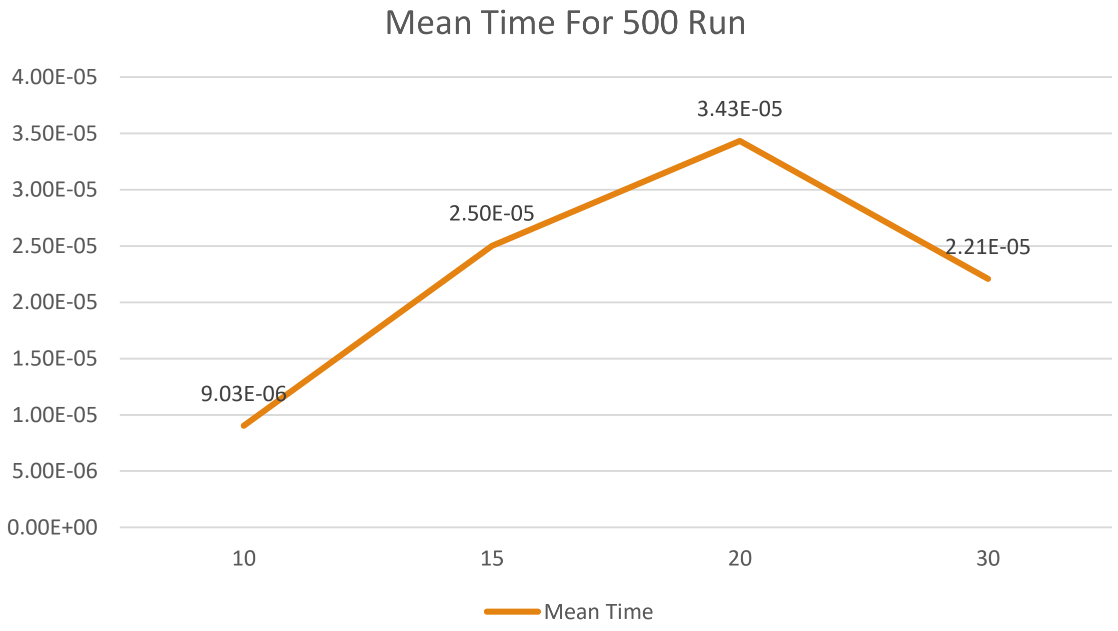
250 Run Per Input Size

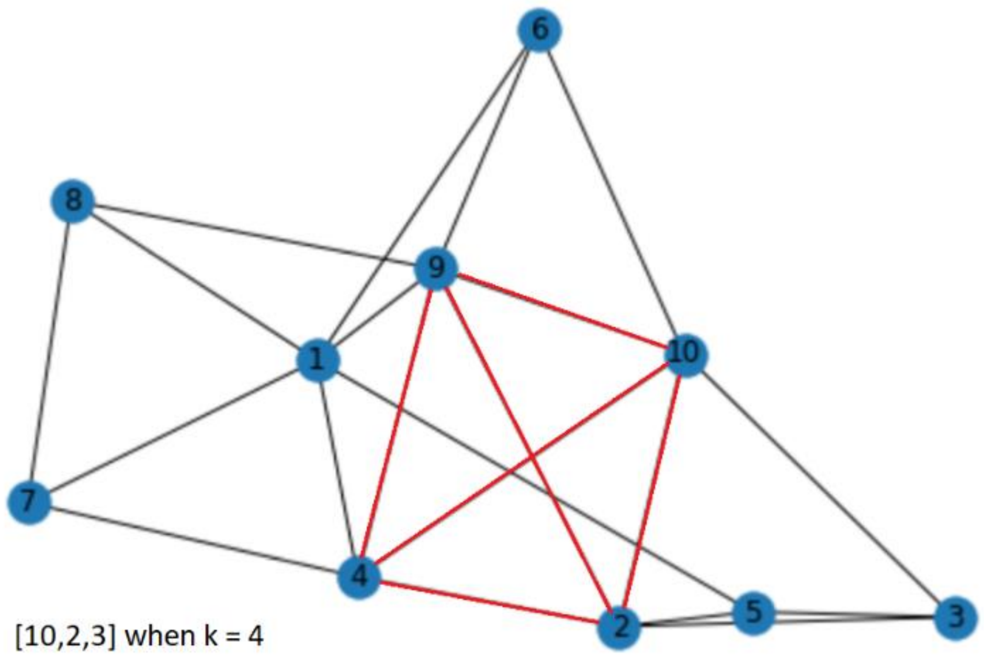
Size	Mean Time	Standard Deviation	Standard Error	%90-CL	%95-CL
10	6.718e-06	2.765e-05	4.257e-07	2.835e-05 - 2.695e-05	2.848e-05 - 2.682e-05
15	2.323e-05	3.928e-05	1.472e-06	4.170e-05 - 3.686e-05	4.216e-05 - 3.639e-05
20	3.593e-05	3.887e-05	2.277e-06	4.262e-05 - 3.513e-05	4.333e-05 - 3.441e-05
30	2.715e-05	2.780e-05	1.720e-06	3.063e-05 - 2.497e-05	3.117e-05 - 2.443e-05



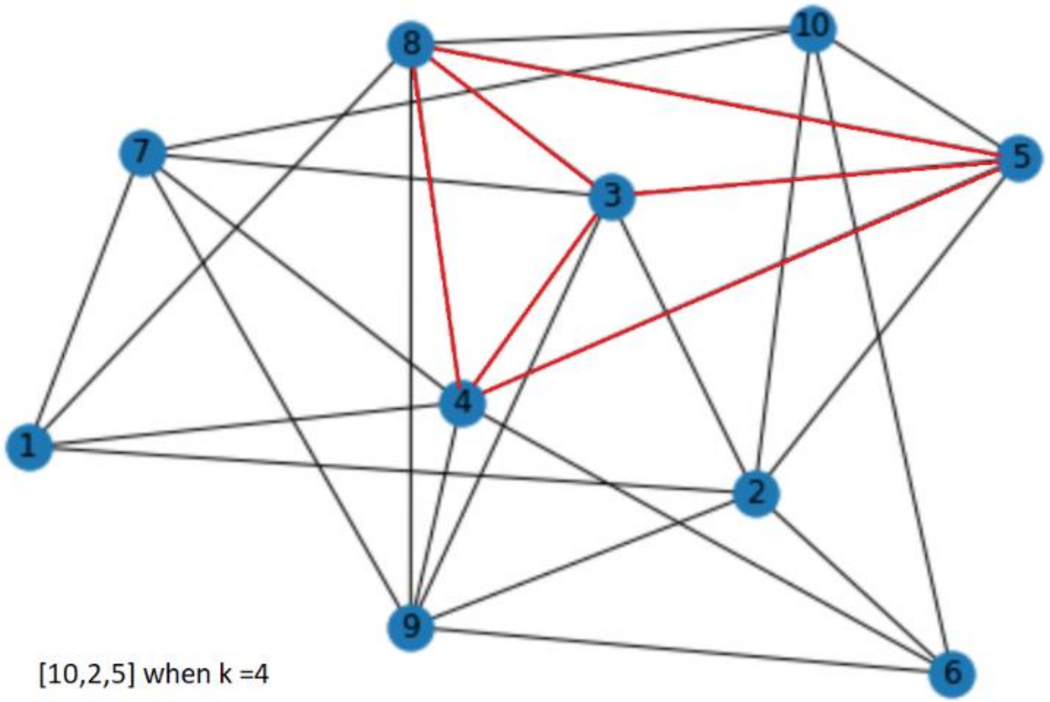
500 Run Per Input Size

Size	Mean Time	Standard Deviation	Standard Error	%90-CL	%95-CL
10	9.031e-06	2.753e-05	4.042e-07	2.820e-05 - 2.687e-05	2.833e-05 - 2.674e-05
15	2.502e-05	4.352e-05	1.120e-06	4.537e-05 - 4.168e-05	4.572e-05 - 4.133e-05
20	3.434e-05	3.734e-05	1.537e-06	3.987e-05 - 3.481e-05	4.035e-05 - 3.432e-05
30	2.208e-05	2.581e-05	9.885e-07	2.744e-05 - 2.419e-05	2.775e-05 - 2.387e-05



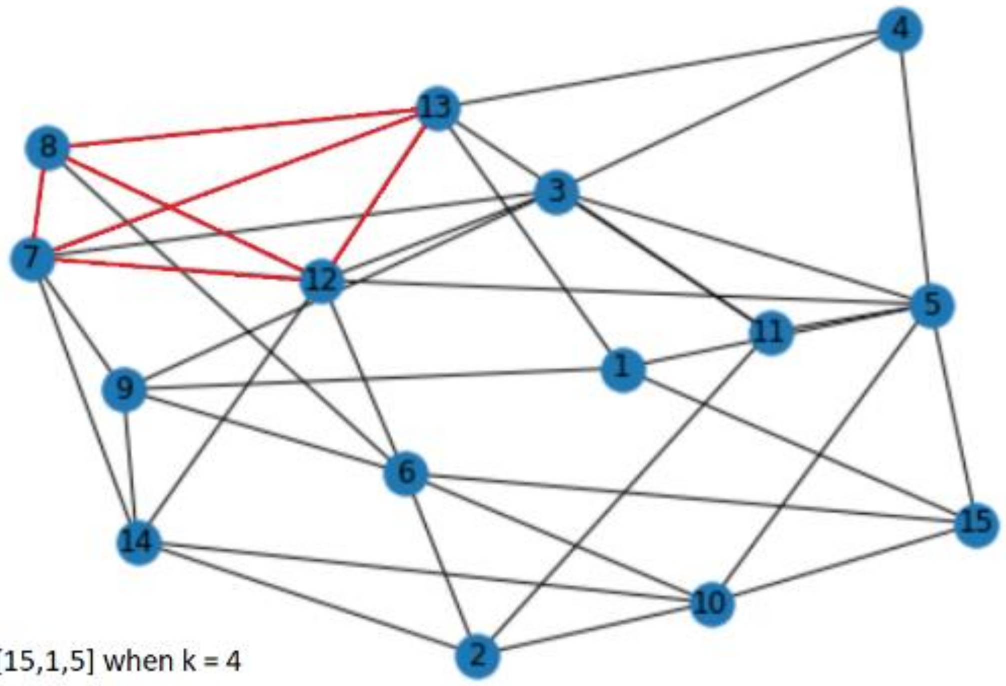


[10,2,3] when k = 4

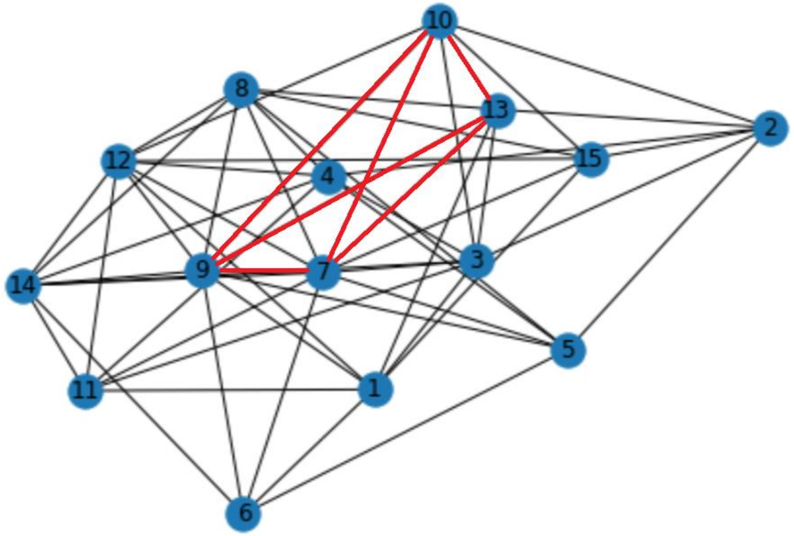


[10,2,5] when k =4

Testing (Failed Conditions)



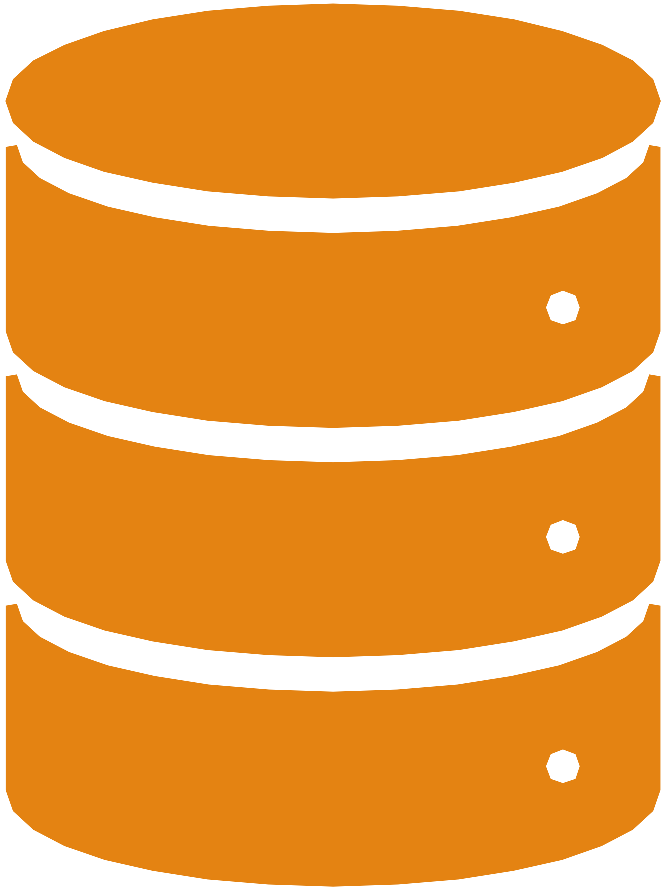
[15,1,5] when k = 4



When k = 4 both algorithm returns true(p algo = [7,3,10,13])
When k = 5 p algorithm returns false but np algo finds =[7,9,8,12,14]

Failed Case

Successful Case



Conclusion (1)

To sum up, we have seen that finding k-clique problem is NP-Complete problem that is reduced from 3-SAT. Also, there is not such an exact efficient algorithm that solves that problem. There are just heuristic algorithms such as greedy max clique algorithms. We changed this algorithm to find k-clique. We examined and analyzed this algorithm in this report and have seen that this algorithm does not always work.

Conclusion (2)

After experimental analysis of this algorithm, we deduced that it has approximately 99% chance to find answer for decision problem.

If we consider the measurement of running time of the algorithm in the experimental analysis part shows a consistent trend with our result. However, when the graph size is 30, the running time decreased because when the graph size increases, number of connections are also increased so finding 5-clique is become easier and fast. Then it is normal to algorithm become faster.