# Finding dense sub graph from given undirected graph


## Abstract

Given undirected graph G= (V,E), density of a sub graph on vertex set S is defined as d(S)= $\frac{E(S)}{|S|}$ ,where E(s) is the set of edges in the sub graph induced by nodes in S .Finding cliques in a given graph is NP- hard problem. In this work I focus on finding Maximum dense sub graph which can be found in linear time. Here I used kind of greedy algorithm which uses some heuristics to prune some nodes and edges to find dense part of the graph. This method can be implemented to directed graph as well.


### Introduction

Given undirected graph G=(V,E) , density of a sub graph on vertex set S is defined as d(S)= $\frac{E(S)}{|S|}$, where E(S) is the set of edges in the sub graph induced by nodes in S. The problem of finding a densest subgraph of a given graph G can be solved optimally in polynomial time, despite the fact that there are exponentially there are many sub graph to consider. Charikar [3] showed that we can find approximation to the densest subgraph problem in linear time using a very simple greedy algorithm (the greedy algorithm was previously studied by Asahiro et. al. [1]). This result is interesting because in many applications of analyzing social networks, web graphs etc., the size of the graph involved could be very large and so having a fast algorithm for finding an approximately dense subgraph is extremely useful.
Finding Largest complete subgraph in given graph is well known NP-hard Combinatorial problem that is particularly intractable because its non approximability. However with some heuristics algorithm finding dense part of the graph fascinatingly finds solution in polynomial time.

In real world most of the relationships like social network or genome connections or protein- protein connection commonly represented by graphs are often subject to noise, resulting in erroneously missing or added edge. This motivates generalizations of the densest graph problem in which the objective to find maximum size subgraph that is almost fully connected.

**Related Works**

since last two decade finding cliques and finding densest parts of the graph emerges with social networks intense among people, so in resent years , there have been significant efforts in devising efficient for finding densest subgraphs.Due to large amount of work that has been done in this research area, my work might not be comprehensive. Several formulas have been studied in literature including cliques and quasi cliques ,minimum degree density ,k-core and densest, among these average degree density stands outs as s natural definition of density.given undirected graph its average degree density is defined by the ratio between the number of edges and the number of nodes in such a graph. Charikar [3]devised a linear-programming based approach as well as a linear 2-approximation algorithm for such a problem. In.[10], heuristics for finding quasi-cliques [40] are presented. All previously mentioned results hold for undirected graphs.dense subgraph computation is an important subroutine in graph indexing for efficient reachability and distance queries,as well as graph compression.

## Methodology

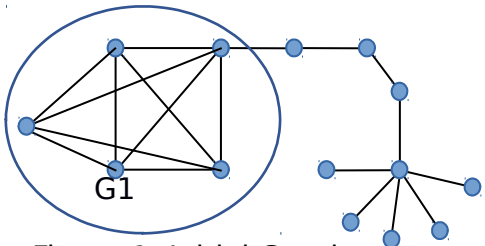



Figure 1: Initial Graph

Given undirected graph G(V,E) has densest graph(G1) and G has average degree $\delta$ ,G has sparse graph as well .

By removing 1 degree nodes in the graph we can successfully increase the graphs density and still densest part of the graph will remain same as it was. Because if the degree of the node < $\delta$ it's most likely not into the densest part of the graph,

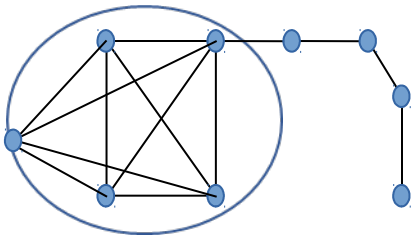


Figure 2: After first iteration

Again check for the 1 degree node if it exists remove that node. If there are no 1 degree node then move on for checking 2 degree node. If its exists remove it and keep on moving until all your nodes degree become > ɤ. By doing this it will lead you to find densest part of the graph.

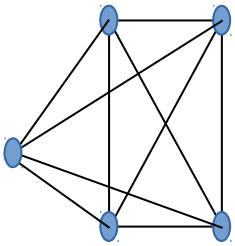


Figure 3: Densest Graph

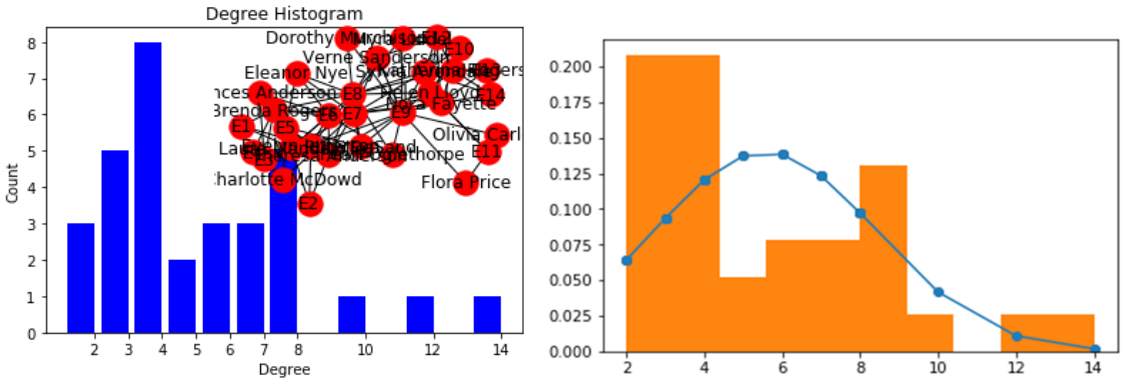
Algorithm

```
Graph=G(V,E):
    for node in nodes:
        if (degree(min(V))<avg.dergree(G):
                remove (V)
                edgeremove(E,min(V))
def edgeremove(E,min(V))
     for edge in E:
         if edge(1)==min(V) or edge(2)==min(V):
             E.remove(edges)
             return(E)
```

## Conclusion

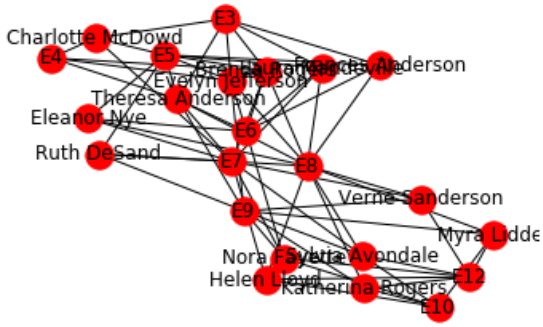
I have tested my algorithm with some well know dataset which is available in Network data repository

- Davis-southern women data set




- Nodes -32
- Edges-89
- Initial density-0.179
- Avg degree-(~5)

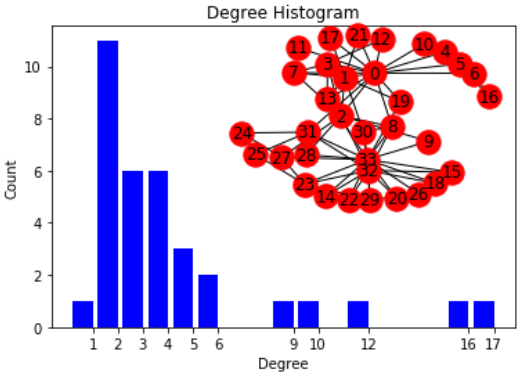
After using my algorithm

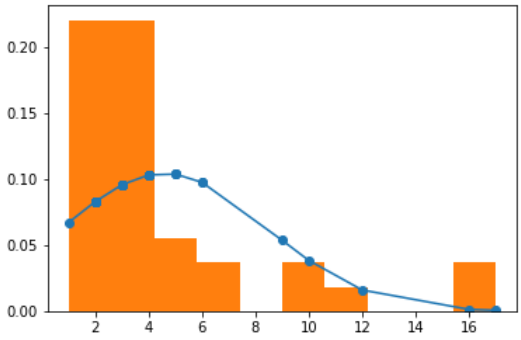



- Optimum densest part of the above graph.
- Nodes-22
- Edges-62
- Density -0.261

According statistic of the graph In this graph less degree node frequency are high it means by using our this algorithm we can find better densest graph.

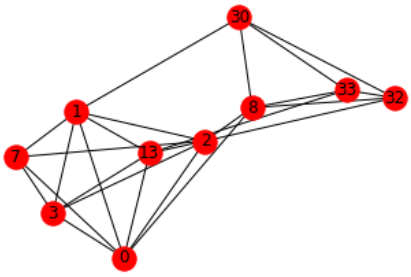
- Karate-club



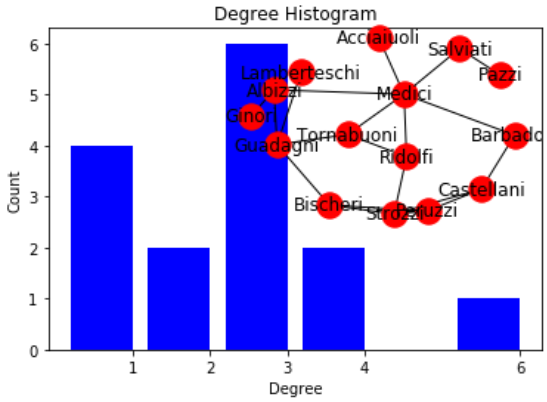
- Nodes -34
- Edges-78
- Initial density-0.139
- Avg degree-(~4)

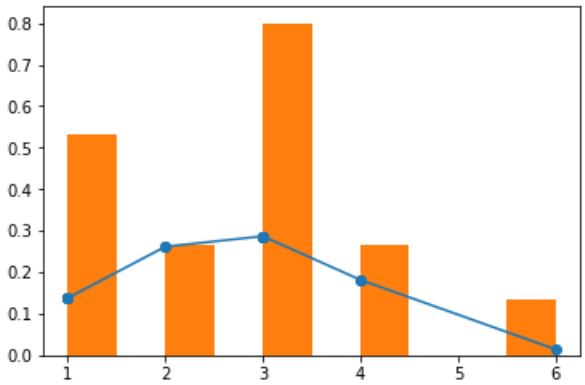
After using the algorithm



- Optimum densest part of the above graph.
- Nodes-10
- Edges-25
- Density -0.491

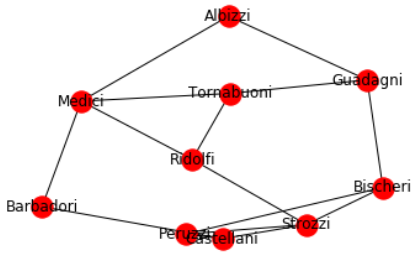
- Florentine families



- Nodes -15
- Edges-20
- Initial density-0.191
- Avg degree-(~3)

After using the algorithm




- Optimum densest part of the above graph.
- Nodes-10
- Edges-15
- Density -0.291

**Summary**

| Name | Avg.degree | No of Edges | No of Nodes | Initial density | Denest part's density |
|---|---|---|---|---|---|
| Davis southern women | 5 | 89 | 32 | 0.179 | 0.261 |
| Karate club | 4 | 78 | 34 | 0.139 | 0.491 |
| Florentine families | 3 | 20 | 15 | 0.191 | 0.291 |

-

## Deficiencies and Future work

some times my algorithm keep on pruning the nodes and edges and it fails to stops with my constraint because in my algorithm I am keep on updating the nodes degree every time I remove the edges from the graph it kind of making problem of converging to the solution. I have to find a optimal heuristic to stop my algorithm while making the graph's density high.

I have to test this algorithm with well know social data , then only I can compare and improve the efficiency by making some subtle changes.

## References


1. Asahiro, Yuichi, and Kazuo Iwama. "Finding dense subgraphs." *Algorithms and Computations* (1995): 102-111.

2.Asahiro, Yuichi, et al. "Greedily finding a dense subgraph." *Journal of Algorithms* 34.2 (2000): 203-221.

3.Charikar, Moses. "Greedy approximation algorithms for finding dense components in a graph." *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer Berlin Heidelberg, 2000.

4.Kannan, Ravi, and V. Vinay. *Analyzing the structure of large graphs*. Forschungsinstitut für Diskrete Mathematik, Rheinische Friedrich-Wilhelms-Universität, 1999.

5.Feige, Uriel, David Peleg, and Guy Kortsarz. "The dense k-subgraph problem." *Algorithmica* 29.3 (2001): 410-421.

6.Khuller, S., & Saha, B. (2009). On finding dense subgraphs. *Automata, languages and programming*, 597-608.

7.Goldberg, Andrew V. *Finding a maximum density subgraph*. Berkeley, CA: University of California, 1984.

8.Mishra, N., Schreiber, R., Stanton, I., & Tarjan, R. E. (2008). Finding strongly knit clusters in social networks. *Internet Mathematics*, *5*(1-2), 155-174.

9.Seidman, Stephen B. "Network structure and minimum degree." *Social networks* 5.3 (1983): 269-287.

10.Tsourakakis, Charalampos, et al. "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees." *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013.