Final Project: Densest Sub-graph Discovery Survey

Sneha Singh, Shaista Ambreen, Sunanda Saha, Raghu D V

Abstract

The Densest sub-graph discovery (DSD) is one of the foundational problems in graph mining. It is not only the basis of other graph problems but also has immense applications across multiple disciplines. In this report, we explore different algorithms to find the densest sub-graph on various data-sets and compare the results. Through this survey, our main observation is that K-clique-graph based DSD achieves better densities in most of the data-sets. We further find that the performance of Goldberg and K-clique-graph with respect to number of nodes in the sub-graph is better than the other two algorithms. Therefore, it can be seen that K-clique-graph based greedy algorithm has relatively better performance overall.

1 Introduction

Graphs are being extensively used to model a wide variety of concepts: social networks, financial markets, biological networks, and many others. A common problem of interest is to find sub-graphs that contain a large number of connections between their nodes. These sub-graphs may correspond to communities in social networks, correlated assets in a market, link spam in a web graph, or mutually influential proteins in a biological network.

1.1 Preliminary

Given a simple graph $G(V, E)$, we define the densest sub-graph discovery (DSD) problem as finding a sub-graph $S_d(V_d, E_d)$ in G with maximum density score $\gamma(S_d)$.

Definition 1. *The edge density score (γ_e) of any graph G is the ratio of number of edges to the number of vertices in G. Our objective function on maximizing the edge density thus becomes:*

$$\max_{S_d:S_d\subseteq G} \gamma_e(S_d) = \max_{S_d:S_d\subseteq G} \frac{|E_d|}{|V_d|}$$

Definition 2. *The clique density score (γ_c) of any graph G is the ratio of number of cliques (c) to the number of vertices in G. Our objective function on maximizing the clique density becomes:*

$$\max_{S_d:S_d\subseteq G} \gamma_c(S_d) = \max_{S_d:S_d\subseteq G} \frac{c}{|V_d|}$$

We compare Goldberg's DSD algorithm against three greedy-based algorithms. The outputs of the algorithms are later compared with respect to the density(D) of a graph G.

Definition 3. *The density(D) of a graph G is the ratio of the number of edges to the maximum possible number of edges formulated as:*

$$D(S_d) = \frac{2|E_d|}{|V_d|(|V_d|-1)}$$

Table 1 summarizes the notations frequently used in this report.

Table 1: Notations and Meanings

Notation	Meaning
$G(V, E)$	Graph with vertex set V and edge set E
$S_d(V_d, E_d)$	Densest Sub-graph in G with vertex set V_d and edge set E_d
$D(G)$	Density of graph G
$\gamma(G)$	Density score of graph G
$\gamma_e(G)$	Edge density score of graph (G)
$\gamma_c(G)$	Clique density score of graph (G)
c	Number of cliques in a graph
d_i	degree of node i in a graph

2 Finding Densest Sub-graph using Maxflow based on Goldberg's Algorithm

We explore Goldberg's algorithm[1] [2] in this section. At each stage of the algorithm, we make a guess "g" for γ_e. Then an augmented graph is constructed and a min cut procedure is applied to decide the scope of the binary search for γ_e. This eventually finds the maximum density sub-graph.

2.1 Graph Augmentation

We introduce two nodes in the graph G, the source node s and the sink node t. Every (undirected) edge in the graph between any two nodes is replaced by two directed edges of capacity 1 each. Further, every node is connected to the source node by an edge of capacity m and to the sink node by an edge of capacity $m + 2g - d_i$, where m is the number of edges in the graph ($m = |E|$). Thus, all capacities are non-negative (as $0 \leq d_i \leq m$).

When this graph $N(V_N, E_N)$ is partitioned using a maxflow solver, we get two node sets S and T such that $s \cup V_1 = S$ and $t \cup V_2 = T$. The capacity of the cut is $m|V|$ when $V_1 = \emptyset$; otherwise it is:

$$\begin{aligned} cut(S,T) &= m|V_2| + \left(m|V_1| + 2g|V_1| - \sum_{i \in V_1} d_i \right) + \sum_{i \in V_1, j \in V_2} 1 \\ &= m|V| + 2|V_1|(g - \gamma_e(V_1)) \qquad \leq m|V| \end{aligned}$$

From the above inequation, if V_1 is not empty, $g \leq \gamma_e(V_1)$, i.e, there must exist a sub-graph with edge density score greater than g. If V_1 is empty, then cut(S,T) = m|V|, which is not the min s-t cut. Thus, $g \geq \gamma_e(S_d)$.

2.2 Algorithm

The edge density score of the densest sub-graph $S_d(V_d, E_d)$ of G is given as:

$$\gamma_e(S_d) \in \left\{ \frac{|E_d|}{|V_d|} : 0 \leq |E_d| \leq |E|, 0 \leq |V_d| \leq |V| \right\}$$

Thus $\gamma_e(S_d)$ can take finite values between 0 and $|E|$, with the smallest difference between any two $\gamma_e(S_{d1})$ and $\gamma_e(S_{d2})$ as $\frac{1}{|V|(|V|-1)}$. This gives us our basis for the binary search. The complete set of steps are mentioned in Appendix 4.

2.3 Runtime Analysis

The maximum number of searches is given as:

$$\log_2 \frac{|E| + 1}{\frac{1}{|V|(|V|-1)}} = \log_2 \left((|E| + 1)|V|(|V| - 1) \right) = \mathcal{O}(\log |V|)\text{times}$$

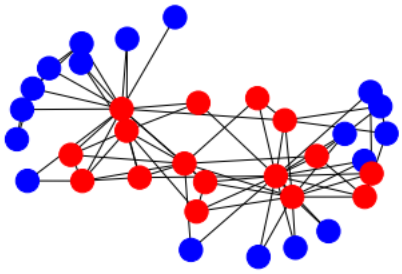

Figure 1: DSD on Karate Club [3] network using Goldberg's algorithm with Maxflow; the nodes in red form the densest sub-graph

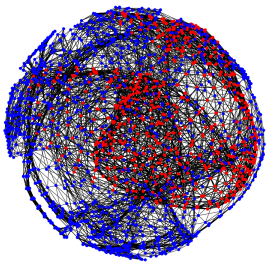

Figure 2: DSD on Data[4] network using Goldberg's algorithm with Maxflow; the nodes in red form the densest sub-graph

If $T(v, e)$ is the time taken to find the maximum flow (or minimum capacity s-t cut) in a graph with v vertices and e edges, then the time complexity of the algorithm for the constructed augmented graph with $|V| + 2$ vertices and $2|V| + 2|E|$ edges is: $\mathcal{O}(T(|V|, |V| + |E|) \log |V|)$

2.4 Experimental Result

We performed a DSD on the Karate club graph [3] that has 34 nodes and 156 edges. The algorithm returned the densest sub-graph with 16 nodes and 42 edges. The edge density scores and density obtained were 2.625 and 0.35 respectively in 0.409951s. The plot of the graph highlighting the densest sub-graph is shown in figure 1.

A similar DSD on the Data graph [4] with 2851 nodes and 30186 edges returns the densest sub-graph with 1152 nodes and 7134 edges. The edge density scores and density obtained were 6.192708 and 0.010760 respectively in 3811s. The plot of the graph highlighting the densest sub-graph is shown in figure 2

The time taken to find the densest sub-graph increases with the size of the graph and for a large graph like Data [4], it is too huge. So we use below greedy-based algorithms to approximate the results.

3 Greedy++ Algorithm

An approximate greedy peeling algorithm [5] is fast but it's approximation factor is far from optimal. The exact minimum cut based algorithm [6] takes more time but gives specific result.

GREEDY++ is an algorithm that combines the best of both worlds - the accuracy of the exact minimum cut based algorithm with the efficiency of the greedy peeling algorithm. It iteratively runs greedy peeling algorithm, while storing some information about the past iterations. This information results in different permutations which in turn yields higher quality outputs.

The algorithm is runs T instances of a weighted version of the greedy peeling algorithm. It takes the graph G and a parameter T as input and maintains a *load* for every node. In each iteration, the *load* of each node is calculated as a function of its induced degree and the *load* from the previous iterations. [7]

While greedy peeling is a $1/2$ approximation algorithm, GREEDY++ gives us $1 + \frac{1}{\sqrt{T}}$ approximation of the densest sub-graph discovery problem. By weighting vertices using their *load* from previous iterations, GREEDY++ implicitly performs a form of load balancing on the graph, thus improving the approximation factor arbitrarily, with increase in iteration count. Each iteration of the above algorithm runs in time $\mathcal{O}((|V| + |E|) \min(\log |V|, T))$.

Algorithm 1 Densest Sub-graph Discovery using Greedy++ Algorithm

Require: Undirected graph G, iteration count T

$G_{densest} \leftarrow G$
initialize the vertex load vector **load** of size same as number of nodes with 0
for $i : 1 \rightarrow T$ **do**
    $H \leftarrow G$
    for $j : 1 \rightarrow deg(H)$ **do**
        find vertex u with minimum $load[j] + deg[j]$
        update the load vector : $load[u] = load[u] + deg[u]$
        remove u from all it's adjacent edges uv from sub-graph H
        if $density(H) > density(G_{densest})$ **then**
            set $G_{densest} \leftarrow H$
return An approximately densest sub-graph of $G : G_{densest}$

```
greedyPlus(2)

Maximum density score objective :  2.611111111111111
Maximum density :  0.30718954248366015
```

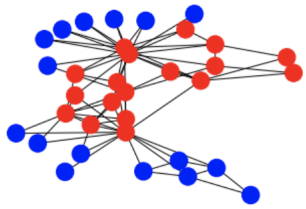

Figure 3: Greedy++ with T=2 on karate.mtx

```
greedyPlus(12)

Maximum density score objective :  2.625
Maximum density :  0.35
```

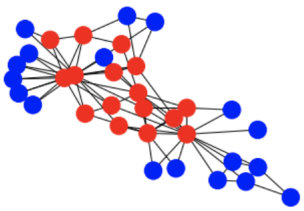

Figure 4: Greedy++ with T=12 on karate.mtx

4 Optimal α-Quasi clique

In this approach, *"clique relaxations"* is used to find a dense sub-graph. According to [8], Quasi-cliques are dense incomplete sub-graphs of a graph that generalize the notion of cliques where not every pair of vertices is connected. So, basically α-Quasi clique (G_S) is a set of vertices S when the number of edges in S is:

$$e_S \geq \alpha\binom{|S|}{2}$$

i.e when the edge density of the induced sub-graph G_S exceeds a threshold parameter $\alpha \in (0, 1)$. So in Quasi clique, each edge in the sub-graph G_S exists with probability α. Therefore, the expected number of edges in G_S is $\alpha \binom{|S|}{2}$. Considering the above condition, α-quasi clique denotes that the sub-graph G_S has more edges than those expected by the binomial model. Using this criterion, we turn the quasi-clique condition into an objective function. In particular, the density function to find the densest sub-graph using Quasi-clique is defined as $f_\alpha(S) = |E(S)| - \alpha \binom{|S|}{2}$, which expresses the edge surplus of a set S over the expected number of edges under the random-graph model. We consider the problem of finding the best α-quasi clique, i.e., a set of vertices S that maximizes the function $f_\alpha(S)$ in this approach. Pseudo-code for the same based on [9] is given in Algorithm 2.

Algorithm 2 Densest Sub-graph Discovery using Optimal α-Quasi clique Algorithm

Require: The undirected graph $G(V, E)$ and α

$H_n \leftarrow G$
for $k : n \rightarrow 1$ **do**
    let v be the smallest degree vertex in H
    $H_{k-1} \leftarrow H_k \backslash \{v\}$
return A quasi-clique $S \in \{H_n, \cdots, H_1\}$ that maximizes $f_\alpha(S)$

The proposed algorithm by Tsourakakis[9] is a greedy approach, a ½-approximation of the densest sub-graph in linear time, $\mathcal{O}(|V| + |E|)$. This can be achieved by keeping track of vertices for each

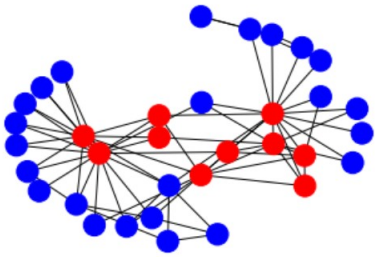

Figure 5: DSD on Karate Club [3] network using α-Quasi-clique with $\alpha = 1/3$; the nodes in red form a dense sub-graph

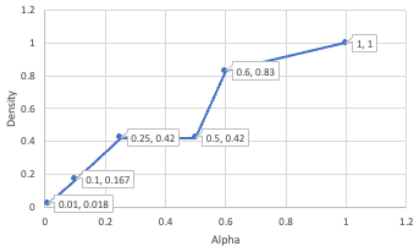

Figure 6: DSD on Delaunay_n10 [10] network on varying α.

possible degree in different lists. During the iterations, the degree of any vertex is modified by moving that vertex to the appropriate degree list.

5 K-clique-graphs

In this section, a new formulation is introduced to calculate dense sub-graphs. We begin by explaining the algorithm for $k = 3$ which uses triangles and then conclude on how the algorithm can be generalized for $k > 3$. The difference lies in the creation of another graph $G'(V', E')$ and the use of a new density function, which is explained in detail below.

As a preprocessing step, we assign a unique label l to all the edges in G. Let $T(G)$ represent all the triangles in G, which is identified using the available triangle listing algorithms [11][12][13]. These algorithms have a worst-case time complexity of $\mathcal{O}(|E|^{\frac{3}{2}})$. We create G' by assigning a vertex v' to each of the triangle, $t \in T(G)$. An edge e' is present in G' if there is a common edge e between the two triangles in G. The label of e' would be the same as that of e.

The next step in the algorithm would be to find a subset of vertices $S' \subset V'$ which forms a dense sub-graph in G. A new density measure called triangle-graph density is introduced. Using this will result in identification of graphs which has more cliques than the other algorithms defined in the above sections.

Definition 4. *Triangle-graph density is given by* $f(S') = \frac{d(S')}{|S'|}$, *where* $d(S') = \sum_{v \in S'} \min_{l \in L(v)}(deg_{S'}(v, l))$. *Here,* $L(v)$ *are the set of 3 labels which forms the triangle represented by* v *in the new graph* G'. $deg_{S'}(v, l)$ *is the number of triangles adjacent to* v *through the edge labeled* l. *The* min *operation ensures that* $f(S')$ *considers all the 3 edges in a triangle.*

Objective 1. *Given a graph* $G = (V, E)$, *create a triangle-graph* $G' = (V', E')$ *and find a subset of vertices* $S^* \subseteq V'$ *such that* $f(S^*) = arg\max_{S' \subseteq V'} f(S')$.

The objective function can be solved by using a greedy algorithm shown in Algorithm 3[14]. It has a time complexity of $\mathcal{O}(|T(G)| + |E'|)$.

Algorithm 3 Densest Sub-graph Discovery using K-clique-graphs - Greedy Algorithm

Require: Graph $G' = (V', E')$
1: $S_{|V'|} \leftarrow V'$
2: **for** $i \leftarrow |V'|$ to 1 **do**
3: let $v_{small} = \min_{v \in S'_i}(\min_{l \in L(v)}(deg_{S'_i}(v, l)))$
4: $S'_{i-1} \leftarrow S'_i \backslash \{v_{small}\}$
5: $S' \leftarrow arg\max_{i=1,\ldots|V'|} f(S')$
6: **return** Subset of vertices $S' \subseteq V'$

The algorithm can be extended for $k > 3$ by constructing the k-clique graph $G' = (V', E')$ similar to the triangle graph above. An edge is drawn in G' if the corresponding k-cliques have a common (k-1)-clique in graph G. Then the triangle-graph density function and the greedy algorithm can be

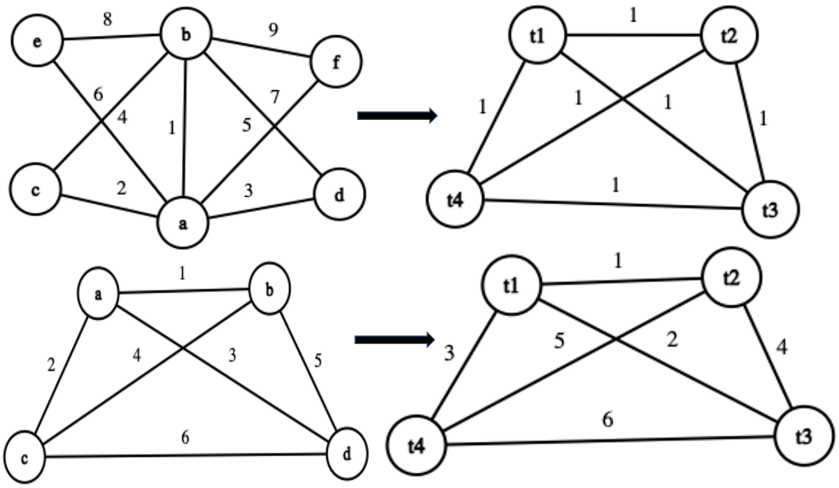

Figure 7: Two different input graphs (left) and their corresponding triangle graphs (right). The algorithm prefers the second graph over the first.

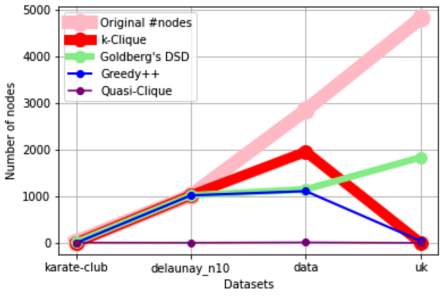

Figure 8: Number of nodes in subgraph returned by the four algorithms across the four datasets along with the number of nodes in the original graph

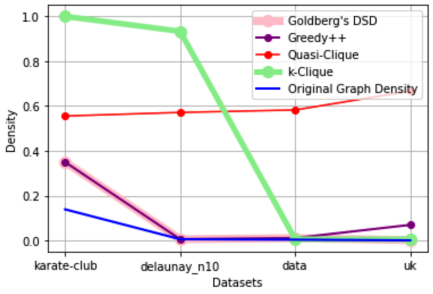

Figure 9: Densities compared on the four datasets using the four algorithms discussed in this paper along with the original graph density

used to find the dense sub-graphs. Please note that the task of finding k-cliques where $k > 3$ is computationally intensive and hence is suggested to use $k = 3$.

6 Experimental Results

We have tested the 4 algorithms described above against 4 datasets. Karate [3] graph has 34 nodes with 78 edges, Delaunay_n10 [10] graph has 1024 nodes with 3056 edges, Data [4] graph has 2851 nodes with 15093 edges, and UK [15] graph has 4824 nodes with 6837 edges. The graph density and the number of the nodes in the sub-graph for each of the dataset against all algorithms is shown in the figures 8 and 9, respectively.

7 Conclusion and Discussion

As seen in the results, K-clique-graph based greedy algorithm outperforms other algorithms in most of the datasets against which we evaluated. Although each algorithm has their pros and cons, K-clique-graph based algorithm has a higher chance of performing better than other algorithms. It can be easily seen that the sub-graphs returned by any of the given algorithms is atleast equal to the original graph density.

References

[1] A.V. Goldberg. *Finding a Maximum Density Subgraph*. URL: https://users.ics.aalto.fi/gionis/goldberg-techreport.pdf.

[2] Charalampos (Babis) E. Tsourakakis2 Aristides (Aris) Gionis1. *Dense Subgraph Discovery (DSD)*. URL: http://people.seas.harvard.edu/~babis/dsd.pdf.

[3] *Newman/karate.mtx*. URL: https://sparse.tamu.edu/Newman/karate.

[4] *data.mtx*. URL: https://sparse.tamu.edu/DIMACS10/data.

[5] Naga V. C. Gudapati, Enrico Malaguti, and Michele Monaci. *In Search of Dense Subgraphs: How Good is Greedy Peeling?* URL: https://arxiv.org/abs/1911.02356.

[6] Michael W. Mahoney. *Lecture Notes on Spectral Graph Methods : Chapter 16 : Diffusions and Random Walks as Robust Eigenvectors*. 2015. URL: https://arxiv.org/pdf/1608.04845.pdf.

[7] Digvijay Boob et al. *Flowless: Extracting Densest Subgraphs Without Flow Computations*. 2020. URL: https://par.nsf.gov/servlets/purl/10175443/.

[8] Srikanta Tirthapura Seyed-Vahid Sanei-Mehri Apurba Das. *Enumerating Top-k Quasi-Cliques*. URL: https://arxiv.org/pdf/1808.09531.pdf.

[9] Charalampos E. Tsourakakis et al. *Denser than the Densest Subgraph: Extracting Optimal Quasi-Cliques with Quality Guarantees*. URL: https://www.francescobonchi.com/denserthanthedensest.pdf.

[10] *delaunay_n10.mtx*. URL: https://sparse.tamu.edu/DIMACS10/delaunay_n10.

[11] G. Nikolentzos et al. *Finding a Minimum Circuit in a Graph*. 1978.

[12] N. Chiba and T. Nishizeki. *Arboricity and Subgraph Listing Algorithms*. 1985.

[13] Wagner Schank T. and D.: Finding. *Counting and Listing all Triangles in Large Graphs, an Experimental Study*. 2005.

[14] G. Nikolentzos et al. *K-clique-graphs for Dense Subgraph Discovery*. 2018. URL: https://arxiv.org/pdf/1610.06008.pdf.

[15] *uk.mtx*. URL: https://sparse.tamu.edu/DIMACS10/uk.

A Appendix

Algorithm 4 Densest Sub-graph Discovery using Maxflow (based on Goldberg's Algorithm)

procedure DSD($G(V, E)$) :
    $l \leftarrow 0$; $u \leftarrow |E|$; $V_1 \leftarrow$ *empty*
    while $u - l \geq \frac{1}{|V|(|V|-1)}$ **do**
        $g \leftarrow \frac{u+l}{2}$
        Construct $N = (V_N, E_N)$
        Find (S, T) = *maxflow*(N)
        if $S = \{s\}$ **then**
            $u \leftarrow g$
        else
            $l \leftarrow g$
            $V_1 \leftarrow S - \{s\}$
    return sub-graph of G induced by V_1