Chapter 1

Fraud Detection - Dense Subgraph Detection

Contributed by Tong Zhao

1.1 Introduction

Fraud behaviors can be spotted everywhere on online applications such as social networks where the behavior data can be represented as large bipartite graphs. These graphs consist of links between followers and followees. Fraudulent actions such as fake followers usually result with creating large and dense subgraphs. For example, as a large number of follower buyers buy followers from one major follower seller, these follower buyers together with the bot followers controlled by the seller will form a subgraph with high density. Hence many existing detection methods [15, 27, 25] estimate the suspiciousness of users by identifying whether they are within a dense subgraph.

1.2 The Problem as a Graph

Here we define the definitions of density for graphs according to [4, 12, 18]. Let $G = (V, E)$ be a undirected graph with vertices $V$ and edges $E \subseteq V \times V$. $E(V)$ stands for the set of edges induced by $V$, that is

$$E(V) = \{(i, j) \in E : i \in V, j \in V\}$$

Then the density of subgraph induced by $S \subseteq V$ can be defined as

$$d(S) = \frac{|E(S)|}{|S|}$$

Note that $2d(S)$ is actually the average degree of the subgraph induced by $S$. The Densest Subgraph problem can be defined as

$$DS(G) = \max_{S \subseteq V} \{d(S)\}$$

For directed graphs. Let $G = (V, E)$ be a directed graph with vertices $V$ and edges $E \subseteq V \times V$. $E(S, T)$ stands for the set of edges from vertices in $S \subseteq V$ to vertices in $T \subseteq V$, that is

$$E(S, T) = \{(i, j) \in E : i \in S, j \in T\}$$
Dense Subgraph Detection

Then the density of subgraph induced by \( S, T \subseteq V \) can be defined as

\[
d(S, T) = \frac{|E(S, T)|}{\sqrt{|S||T|}}
\]

The Densest Subgraph problem can be defined as

\[
DS(G) = \max_{S,T \subseteq V} \{d(S, T)\}
\]

### 1.3 Some Realistic Data Sets

The data sets that this application encounter can come from social networks (botnet followers) or e-commercial platforms (bully buyers). Thus the graph size can be huge. For example, the Twitter follower-followee data set used in Fraudar [15] contains 41.7 million nodes with 1.47 billion edges. Similar data sets for social networks such as Twitter and e-commercial platforms such as Amazon can be found on SNAP [21] or other platforms. It is intuitive that the size of graphs for such problem in real industry will be growing continuously. Hence it is important for the algorithms to have a linear or near linear run-time or be able to parallelize.

### 1.4 Dense Subgraph Detection-A Key Graph Kernel

Multiple algorithms exists for detecting the dense subgraphs. One majorly used algorithm is proposed by Charikar in 2000 [4], which is an approximation algorithm by greedy approach. Although Charikar’s algorithm sacrificed quality of the result subgraph for much better time complexity, this algorithm still has a provable 2-approximation guarantee. That is, if the densest existing subgraph \( S' \) has edge density of \( d(S') = \lambda \), the result subgraph \( S \) of Charikar’s algorithm will have edge density of \( d(S) \geq \lambda/2 \).

The greedy idea of Charikar’s algorithm is to remove the vertex that is least likely in the densest subgraph at each step according to certain rule. In the case of undirected graph, the rule can obviously be to remove the vertex with lowest degree. Then Charikar’s algorithm can be described as following. [4]

```plaintext
1: procedure Densest-Subgraph(G)
2: Input: Undirected graph \( G = (V, E) \).
3: Output: Dense sugraph \( S \) of \( G \).
4: \( n \leftarrow |V| \)
5: \( G_n \leftarrow G \)
6: for \( k \leftarrow n \) down to 1 do
7: \( v \leftarrow \) the vertex with smallest degree in \( G_k \)
8: \( G_{k-1} \leftarrow G_k \setminus \{v\} \)
9: return The subgraph with maximum density amoung \( G_1, G_2, \ldots, G_n \).
```

A detailed proof for this algorithm to achieve a 2-approximation can be found in [19].

For directed graphs, Khuller and Saha proposed a approximation algorithm based on Charikar’s algorithm in 2009 [19] that ultilized the same greedy idea. The key point of this algorithm for directed graphs is to first duplicate all the vertices and construct a bipartite graph such that one copy of the vertices have only outgoing edges and the other copy of the vertices have only incoming edges. Then the algorithm can be described as following. [19]

```plaintext
1: procedure Densest-Subgraph-Directed(G)
```
Dense Subgraph Detection

2: **Input:** Directed graph \( G = (V, E) \).
3: **Output:** Dense subgraph \( S \) of \( G \).
4: \( n \leftarrow |V| \)
5: \( G_{2n} \leftarrow G \)
6: for \( k \leftarrow 2n \) down to 1 do
7: \( v \leftarrow \) the vertex with smallest degree in \( G_k \)
8: if \( v \) has outgoing edges then
9: Delete all the outgoing edges incident on \( v \).
10: else
11: Delete all the incoming edges incident on \( v \).
12: Deleted all vertices with 0 degree.
13: \( G_{k-1} \leftarrow \) the remaining of graph \( G_k \)
14: return The subgraph with maximum density among \( G_1, G_2, \ldots, G_{2n} \).

A detailed proof for this algorithm to achieve a 2-approximation can also be found in [19].

Both algorithms have time complexity of \( O(|V| \log |V|) \) and space complexity of \( O(|V|^2|E|) \). To evaluate the performance of these two algorithms, both density on the result subgraph and runtime can be used.

1.5 Prior and Related Work

1.5.1 Dense Subgraph Problem

The history for Dense Subgraph problem for static graphs has a rather short history, as the best exact solution was proposed by Goldberg in 1984 [12] and the best approximation algorithm so far were proposed by Charikar in 2000 [4] and Khuller and Saha in 2009 [19] for undirected and directed graphs.

Goldberg’s solution works only for undirected graph. His idea was to interestingly transfer this dense subgraph problem into a well-know min-cut problem by adding two vertices \( s \) and \( t \). Both \( s \) and \( t \) are connected with all the vertices in graph \( G = (V, E) \). For each vertex \( v_i \in V \), edge \((s, v_i)\) has edge weight that is the same as the degree of \( v_i \) and edge \((v_i, t)\) has edge weight of a positive constant \( c \). All the edges in the original graph has an edge weight of 1. Then buy performing a min-cut call that splits \( s \) and \( t \) into two subgraphs, one of the subgraphs would be the densest subgraph of \( G \) after removing \( s \) or \( t \).

Since min-cut problem can be solved using the parametrix max-flow algorithm, this algorithm has a \( O(|V||E|) \) time complexity. Thus Goldberg’s algorithm is not scalable for large graphs; faster approximation algorithms are more preferred in industry situations.

1.5.2 Fraud Detection

Data-driven approaches have received great success in the field of fraud detection [20, 15]: most methods indentify unexpected dense regions of the bipartite graph, as creating fake reviews/ratings unavoidably generates edges in the graph [16, 6, 23, 36, 29, 2].

**Unexpected spectral patterns.** Global graph mining methods model the entire graph to find fraud based on singular value decomposition (SVD), latent factor models, and belief propagation (BP). **SPOKEN** [27] considered the “spokes” pattern produced by pairs of eigenvectors of graphs, and was later generalized for fraud detection. **fBox** [28] focuses on mini-scale attacks missed by
Dense Subgraph Detection

spectral techniques. BP has been used for fraud classification on eBay [25], link farming on Twitter [9], and fake software review detection [1].

**Unexpected high density in subgraphs.** Finding dense subgraphs has been studied from a wide array of perspectives such as mining frequent subgraph patterns [22, 37], detecting communities [10, 5, 26], and finding quasi-cliques [11, 33, 8, 30]. [4] shows that average degree of subgraph can be maximized with approximation guarantees. [34] optimizes the density of adjacency matrix of subgraph with quality guarantees. [15] adopts both node degree and edge density to model suspiciousness of subgraph and further increases accuracy in binary adjacency matrix of bipartite graph.

**Unexpected high density in time-series.** Typically there are two kinds of representation on density in time-series. One is dense subgraphs in evolving graphs [7]. COPYCATCH [3] uses local search heuristics to find \( \Delta t \)-bipartite cores in which users consistently likes the same Facebook pages at the same short time interval. The other is dense subtensors in high-order tensors of a time dimension [24, 17, 31] or tensor streams [32]. [35, 13] consider fraud detection methods that are robust to camouflage attacks. [14] adopts a Bayesian model to find early spikes of outlier ratings in time series. All these methods focus on the time-series domain, observing changes in the behavior from system access logs rather than graph data.

1.6 A Sequential Algorithm

Discuss here the outlines of a sequential algorithm. What programming paradigms might make the most sense? What are the key data structures? Does the computational complexity differ from that in the Section 1.4?

1.7 A Reference Sequential Implementation

Discuss here your implementation of the basic sequential code. Include what language/paradigm you used for the code.

1.8 Sequential Scaling Results

Discuss here results from your sequential implementation. Include software and hardware configuration, where the input graph data sets came from, and how input data set characteristics were varied. Did the performance as a function of size vary as you predicted?

1.9 An Enhanced Algorithm

Discuss here the outlines of an enhanced algorithm. This could be a parallel code, a code with some significant heuristics, or a code written in a non-traditional programming paradigm. Pseudocode is fine. Discuss what you think is the computational complexity.

1.10 A Reference Enhanced Implementation

Discuss here an implementation of the enhanced algorithm. Include what language/paradigm you used for the code.
1.11 Enhanced Scaling Results

Discuss here results from the enhanced algorithm. Include software and hardware configuration, where the input graph data sets came from, and how input data set characteristics were varied. Ideally plots of performance vs BOTH problem size changes AND hardware resources are desired. Did the performance as a function of size vary as you predicted?

1.12 Conclusion

Summarize your paper. Discuss possible future work and/or other options that may make sense.

1.13 Response to Reviews

This will be included only in the second and third iterations, and will be a summary of what you learned from the reviews you received from the prior pass, and how you modified the paper accordingly.
Bibliography


Dense Subgraph Detection


[22] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S Yu. Mining behavior graphs for “backtrace” of noncrashing bugs. In SDM, pages 286–297, 2005.


Dense Subgraph Detection


