Fraud Detection by Dense Subgraph Detection

Tong Zhao

Dense Subgraph Detection

- Given a graph $G = (V, E)$ with vertices V and edges $E \subseteq V \times V$.
- Find a subgraph S such that $d(S)$ is maximized.
- Edge density (average degree): $d(S) = \frac{|E(S)|}{|S|}$

Charikar's greedy algorithm (2000) [1]

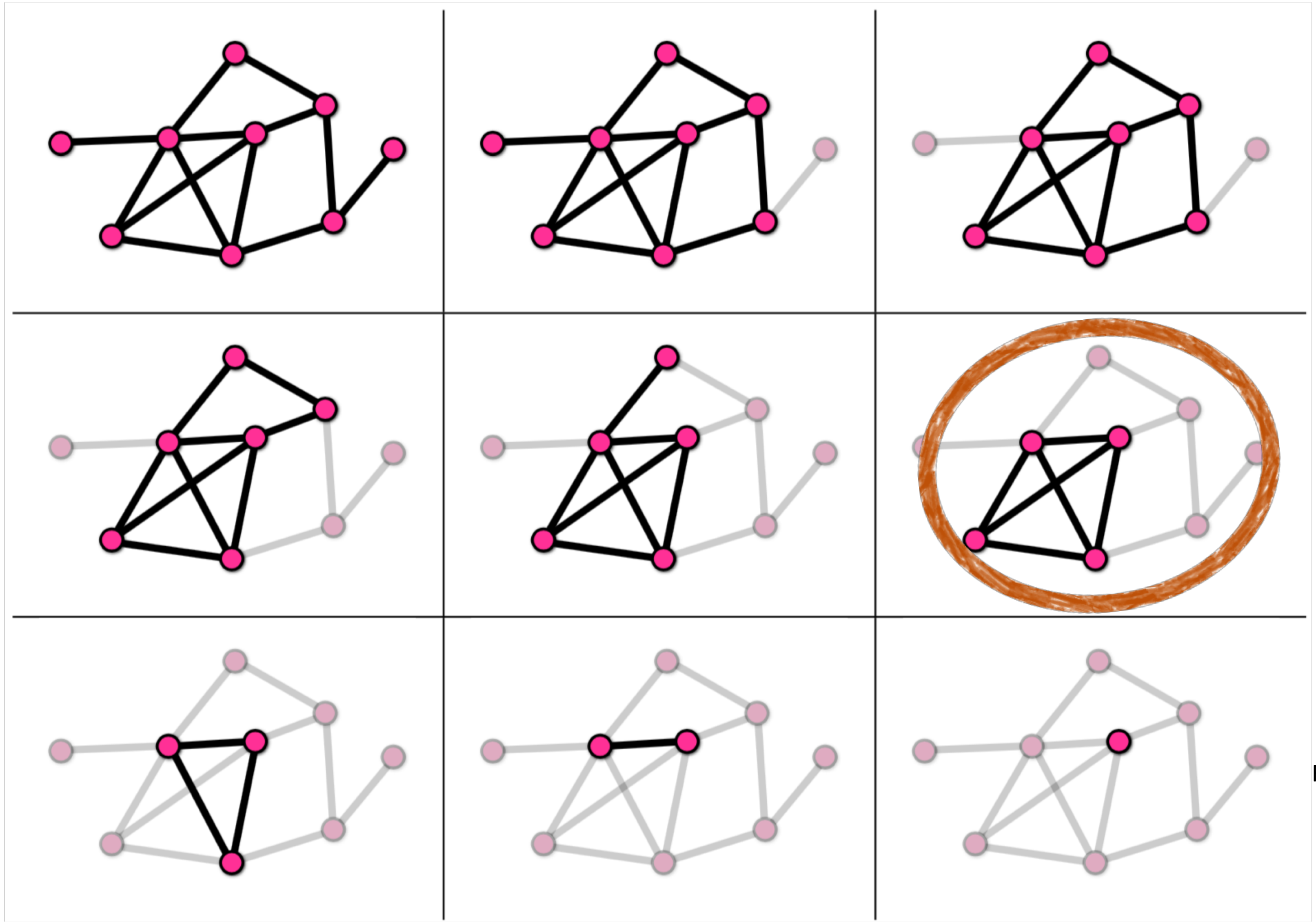

Figure from [3].

2-approximation Guarantee

- Density of the result is theoretically guaranteed.
  - Charikar's algorithm is a provable 2-approximation algorithm.

$$d(S') \geq \frac{1}{2} d(S_{opt})$$

  - S' denotes the result subgraph by Charikar's algorithm.
  - S_{opt} denotes the optimal solution.
- Usually gives close-to-optimal result in real life graphs.

Possible enhancements

- Dense subgraph detection for larger graphs.
- Dense subgraph detection for dynamic graphs.

Scalability

- Observation:
- In social media graphs: |E| >> |V|
- Can we only store the vertices?

Charikar’s greedy algorithm (2000) [1]

procedure DENSEST-SUBGRAPH(G)
    Input: Undirected graph $G = (V, E)$.
    Output: Dense sugraph S of G.
    $n \leftarrow |V|$
    $G_n \leftarrow G$
    for $k \leftarrow n$ down to 1 **do**
        $v \leftarrow$ the vertex with smallest degree in G_k
        Delete all edges incident on v.
        Delete all vertices with 0 degree.
        $G_{k-1} \leftarrow$ the remaining of graph G_k
    return The subgraph with maximum density amoung $G_1, G_2, \ldots, G_n$.

Scalability

- Most intuitive approach:
- Store only the vertices with their degrees in RAM.
- $O(|V|)$ passes.

Enhanced Algorithm [2]

- Remove a set of vertices each time.

Algorithm 1 Densest subgraph for undirected graphs.

Require: $G = (V, E)$ and $\epsilon > 0$

$\tilde{S}, S \leftarrow V$
while $S \neq \emptyset$ **do**
    $A(S) \leftarrow \{i \in S \mid \deg_S(i) \leq 2(1+\epsilon)\rho(S)\}$
    $S \leftarrow S \setminus A(S)$
    if $\rho(S) > \rho(\tilde{S})$ **then**
        $\tilde{S} \leftarrow S$
    end if
end while
return $\tilde{S}$

Enhanced Algorithm

- For any $\epsilon > 0$, this algorithm has
  - $O(\log_{1+\epsilon} |V|)$ passes.
  - $(2 + 2\epsilon)$-approximation guarantee.
- This algorithm can be parallelized or distributed.
  - Originally implemented in MapReduce.

Implementation

- Written in Python 3.
- ~100 lines.
- No paradigm was used.
- multiprocessing for parallelization.

Implementation

```
    def converge(self):
        best_subgraph = []
        best_density = 0.0
        while len(self.subgraph) > 0:
            self.tmp_counter = list(self.subgraph.items())
            q = Queue()
            ranges = self.getST(len(self.tmp_counter))
            processes = []
            for i in range(10):
                p = Process(target=self.getBadVertices, args=(q, ranges[i]))
                p.start()
                processes.append(p)
            for p in processes:
                p.join()
            while not q.empty():
                del self.subgraph[q.get()]
            self.updateDegrees()
            current_density = self.getDensity(self.subgraph)
            if current_density > best_density:
                best_density = current_density
                best_subgraph = list(self.subgraph.elements())
        print('The final result subgraph contains {} vertices with density of {}.'. \
                  format(len(best_subgraph), best_density))
        return best_subgraph, best_density
```

Dataset

- Twitter dataset.
  - 41.7 million users (vertices).
  - 1.47 billion follows (edges).
  - 25Gb.
- Very slow due to the I/O part.
  - $O(\log_{1+\epsilon}|V|)$ passes. (43 with $\epsilon = 0.5$)

Dataset

- Twitter dataset.
  - 41.7 million users (vertices).
  - 1.47 billion follows (edges).
  - 25Gb.
- Very slow due to the I/O part.
  - $O(\log_{1+\epsilon} |V|)$ passes. (43 with $\epsilon = 0.5$)
- It works.
  - Large improvement from MemeroyError.

References

- [1] Charikar, Moses. "Greedy approximation algorithms for finding dense components in a graph." *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, Berlin, Heidelberg, 2000.
- [2] Bahmani, Bahman, Ravi Kumar, and Sergei Vassilvitskii. "Densest subgraph in streaming and mapreduce." *Proceedings of the VLDB Endowment* 5.5 (2012): 454-465.
- [3] Gionis, Aristides, and Charalampos E. Tsourakakis. "Dense subgraph discovery: Kdd 2015 tutorial." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.