Abstract of the Paper: Efficient Algorithms for Densest Subgraph Discovery on Large Directed Graphs

Background

In many applications, such as fraud detection, community mining, and graph compression, it is necessary to find the densest subgraph in a directed graph, known as the Directed Densest Subgraph (DDS) problem. DDS problems have wide applications in fields such as social networks, web graphs, and knowledge graphs. For example, in social networks, it can be used to detect fake followers, and in web graphs, it can help discover network communities. However, existing DDS solutions face efficiency and scalability issues. For instance, some of the best deterministic algorithms (those determining the densest subgraph) require three days to compute for a graph with only three thousand edges.

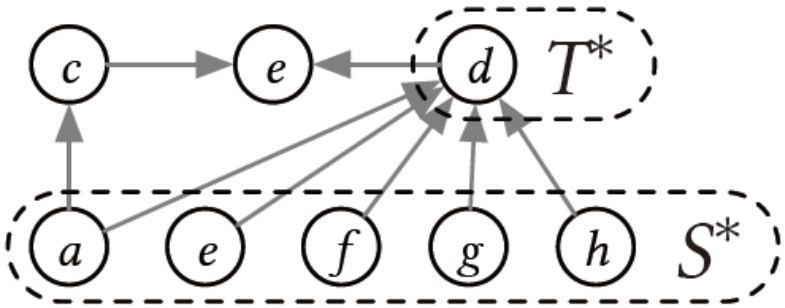


Figure 1: An example of fake follower detection.

The above figure is an illustration of fake follower detection. It shows a microblogging network where edges represent "following" relationships. By issuing a DDS query, two sets of users S* and T* are returned. Compared to other users, d (in T*) has an abnormally high number of followers (a, e, f, g, h) in S*. This could indicate that d has bribed users in S* to follow them.

Challenges

The goal of the DDS problem is to find a subgraph whose density is the highest among all subgraphs. Given a directed graph G=(V,E), the DDS problem aims to find two sets of vertices S* and T* such that the ratio of the number of edges from S* to T* to the square root of the number of vertices is maximized. Existing solutions face two main challenges: high computational complexity, making it infeasible to handle large-scale graph data, and suboptimal precision and efficiency of approximate algorithms.

Innovative Algorithms

To address the above problems, this paper proposes two innovative algorithms: a deterministic algorithm (DC-Exact) and an approximate algorithm (Core-Approx).

Deterministic Algorithm (DC-Exact)

This algorithm is based on the concept of [x,y]-core and employs a divide-and-conquer strategy to improve efficiency.

[x,y]-core Concept

- **Definition**: Given two sets of vertices S and T, if each vertex in S has at least x outgoing edges to T, and each vertex in T has at least y incoming edges from S, then the subgraph is called an [x,y]-core.
- **Significance**: The [x,y]-core concept can significantly reduce the problem size to a dense subregion of the graph, making the discovery of the densest subgraph more efficient.

Implementation Principles

1. **Internal Loop Optimization**: Introduces the concept of [x,y]-core to reduce the size of the graph to be processed.
2. **External Loop Optimization**: Proposes a divide-and-conquer strategy that significantly reduces the number of possible values for $|S| / |T|$, thus optimizing the external loop.
3. **Algorithm Process**:
    - Enumerate all possible values of $|S| / |T|$.
    - Use binary search to guess the maximum density value.
    - For each $|S| / |T|$ value, construct a flow network and run the maximum flow algorithm to compute the minimum st-cut (using the maximum flow algorithm to find the maximum flow from the source (s) to the sink (t), thereby determining how to split the graph into two parts with the minimum cut edges).
    - Update the densest subgraph.
    - The total time complexity is O(n^3 m log n), but the actual computation is significantly reduced through the optimizations mentioned above.

Approximate Algorithm (Core-Approx)

This algorithm is based on the concept of [x, *y*]-core and improves efficiency by computing a 2-approximate solution.

[x, *y*]-core Concept (the core subgraph with the largest x*y value among all possible [x, y]-cores)

- **Definition**: Among all [x,y]-cores, find the one that maximizes x*y. Theoretically, this [x*, y*]-core is a 2-approximate solution to the DDS problem.
- **Significance**: This core is not only simple to compute but also provides a near-optimal solution in practical applications.

Implementation Principles

1. **Compute [x, *y*]-core**: Compute the [x, *y*]-core using an efficient algorithm with a time complexity of

$$O(\sqrt{m}(n+m))$$

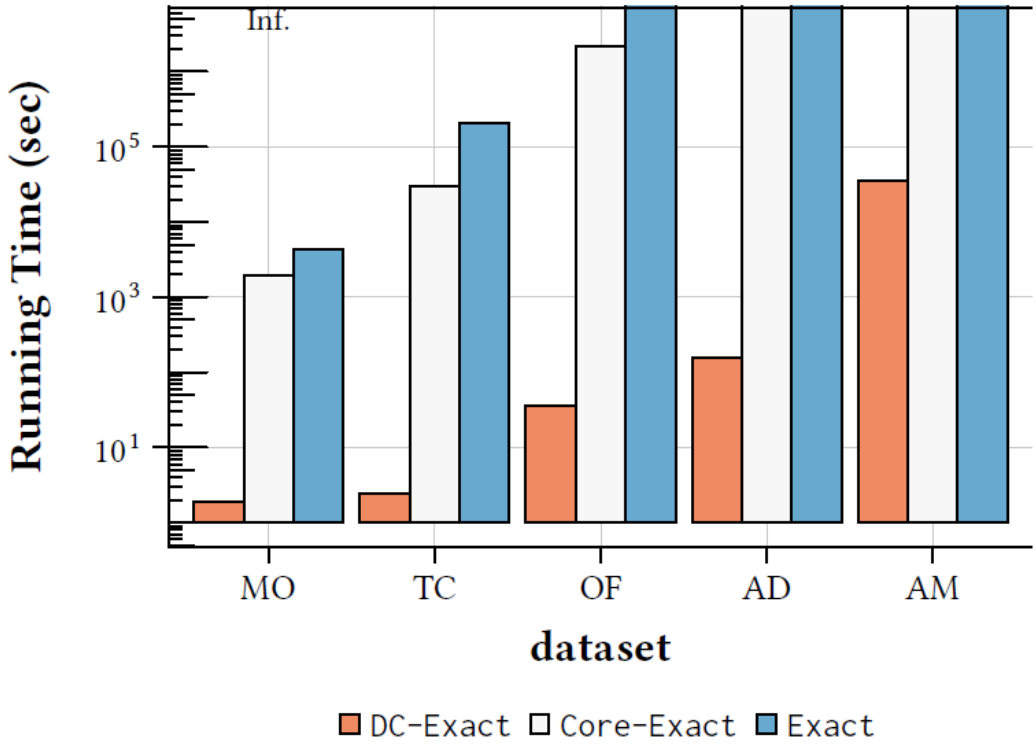
2. **Algorithm Process**:

- Compute the [x, *y*]-core.
- Return this core as the 2-approximate solution.

Results and Conclusion

Results Report

Experimental results show that the deterministic algorithm DC-Exact achieves a speedup of six orders of magnitude when processing graphs with approximately 6500 vertices and 51000 edges. The approximate algorithm Core-Approx can scale to graphs with billions of edges, being six orders of magnitude faster than existing 2-approximate algorithms.



The above figure shows the efficiency results of the exact algorithms on the first five datasets (MO, TC, OF, AD, and AM). As the algorithms could not complete the computation within a reasonable time on larger datasets, those results are not fully recorded in the figure. It can be seen that Core-Exact is at least twice as fast as the state-of-the-art exact algorithm Exact, and DC-Exact is six and five orders of magnitude faster than Exact and Core-Exact, respectively. The primary reason is that DC-Exact adopts a divide-and-conquer strategy, significantly reducing the number of $|S|/|T|$ values to be considered.

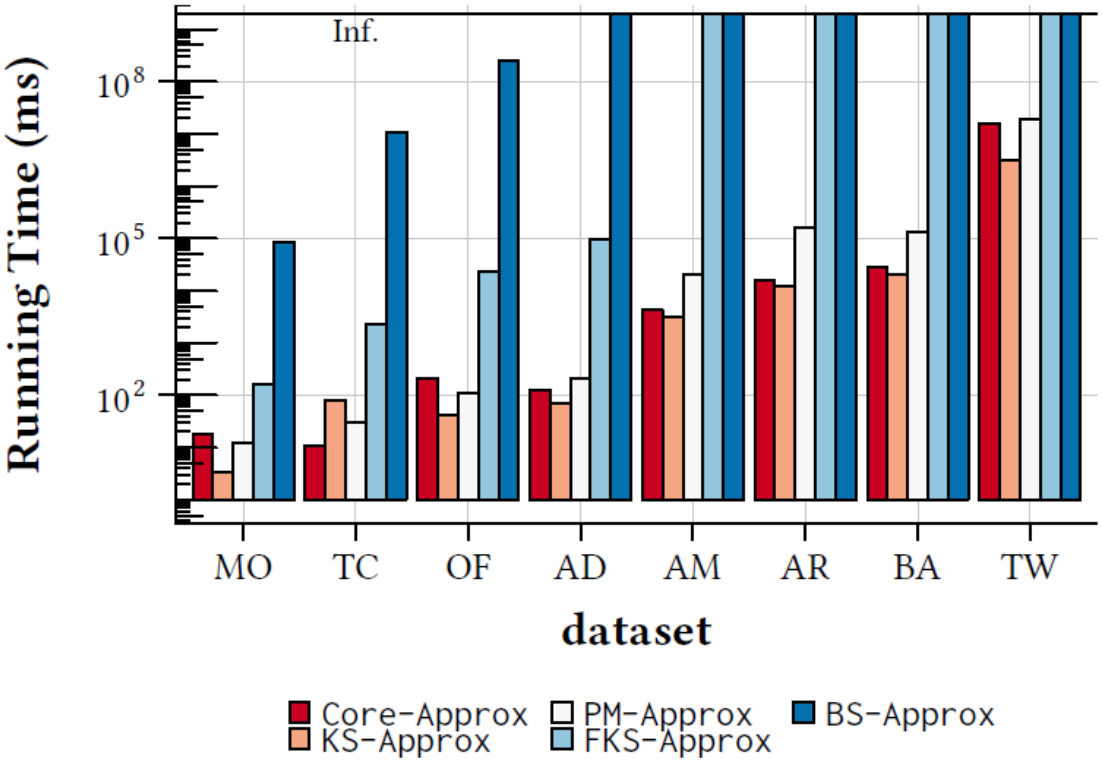


Figure 11: Efficiency of approximation algorithms.

The above figure shows the running time of the approximate algorithms on eight datasets, with the bar touching the top boundary indicating that the algorithm could not complete within 200 hours. It can be observed that BS-Approx and FKS-Approx are the least efficient algorithms, while KS-Approx is the most efficient on almost all datasets. Core-Approx is the second most efficient, followed by PM-Approx, with Core-Approx being six and three orders of magnitude faster than BS-Approx and FKS-Approx, respectively. Additionally, Core-Approx can handle graph data with billions of edges, achieving high efficiency while maintaining high-quality results.

Conclusion

The innovative algorithms proposed in this paper effectively address the efficiency and scalability issues of the DDS problem, especially when dealing with large-scale datasets. Future research can extend in the following areas:

1. **Algorithm Optimization**: Further optimize the time complexity of the algorithms to handle even larger graph data.
2. **Application Extension**: Apply these algorithms to more practical scenarios, such as biological network analysis and recommendation systems.
3. **Theoretical Enhancement**: Conduct in-depth research on the theoretical properties of [x,y]-core and explore more characteristics of dense subgraphs.

These algorithms have extensive potential in practical applications, particularly in handling large-scale social networks, web graphs, and knowledge graphs, providing efficient and accurate data analysis tools.