Applied Intelligence (2022) 52:5034–5046
https://doi.org/10.1007/s10489-021-02684-w

Slither: finding local dense subgraphs measured by average degree

Zixuan Deng[1] · **Yanping Xiang**[1]

Accepted: 12 July 2021 / Published online: 2 August 2021

Abstract
Searching for dense subgraphs is the crux of a variety of graph mining applications. It is also meaningful to investigate problems of finding local dense subgraphs related to particular nodes, especially for large real-world graphs. However, none of the problems focus on maximizing the average degrees of the found subgraphs. We formulate the MDS-N problem, which aims to find such a subgraph with the maximum average degree, near or containing a given node in an undirected graph. We propose the Slither algorithm and the Slither PageRank (PR) algorithm, built on a reduction from the MDS-N problem to the minimum conductance problem and the Lovász-Simonovits Theorem of random walks. A simple hierarchically repetition frame is also proposed to advance the two algorithms. Experiments conducted on both unipartite graphs and bipartite graphs show the effectiveness, stability, and scalability of our algorithms. Additionally, we verify the MDS-N problem and the proposed algorithms on a large social network Twitter, the experimental results of which show our algorithms can successfully detect local fraud-related subgraphs based on particular fraudulent user accounts.

Keywords Dense subgraph mining · Local graph algorithm · Random walks · Targeted fraud detection

1 Introduction

Dense subgraphs reveal important information in graphs, which give birth to many applications. On the one hand, dense subgraphs serve as cornerstones of some graph clustering algorithms [1] and graph partitioning algorithms [2, 3]. On the other hand, they are independently used in real-world scenarios to, e.g., extract research communities in co-authorship networks [4] and detect fraud groups in social or commercial networks [5]. For the latter kind of applications, existing dense subgraph finding methods are almost global-oriented, without prior knowledge, finding dense subgraphs globally and providing them as communities or groups. But sometimes, these methods are expected to be local-oriented or targeted-oriented, i.e., finding dense subgraphs that are related to particular authors, user accounts or product IDs, called *targeted applications*.

Methods to search dense subgraphs can be classified by the density measure, i.e., the way of defining density. Consider an unipartite graph $G_u = (V, E)$, a bipartite graph $G_b = (V_u, V_v, E)$, and a subset of nodes $S \subseteq V$ or $S = S_u \cup S_v$ with $S_u \subseteq V_u$ and $S_v \subseteq V_v$. Let $E(S)$ be the set of edges in the subgraph induced by S. The *average degree* measure of S is defined as $|E(S)|/|S|$ for G_u and $|E(S)|/(|S_u|+|S_v|)$ for G_b. The density measure of S presented in Kannan and Vinay's work [6] is defined as $|E(S)|/\sqrt{|S_u||S_v|}$ only for G_b. The *edge density* measure of S is defined as $|E(S)|/\binom{|S|}{2}$ for G_u and $|E(S)|/(|S_u||S_v|)$ for G_b. For global-oriented methods, Charikar's greedy algorithm [7] produces S for either G_u or G_b with its average degree being at least 1/2 of that of the optimal subset of nodes. Kannan and Vinay proposed a spectral algorithm for G_b, using their customized density measure to identify S with its density within a factor of $\mathcal{O}(\log n)$ [6]. Lots of works in detecting α-quasi-cliques, a kind of dense subgraph, are based on the edge density measure and its variants [4, 8–10].

Take, for example, fraud detection in social networks. The social networks are formed as bipartite user-user networks. Many global-oriented methods are used to detect extremely dense subgraphs in these networks, due to one common type of fraudulent practices is that fraudsters hire workers to add edges towards their customers. The workers and the customers are users in these networks. However, manually reported violations still exist. It is necessary to develop local dense subgraph finding methods

✉ Zixuan Deng
dengzxuan@foxmail.com

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

for assisting people with the discovery of fraud groups related to individually reported user accounts. There are a few of these methods [1, 10] based on density measures of Kannan and Vinay's and edge density, but without average degree. Nevertheless, average degree is a very useful density measure for a fraud detection application, because fraudsters make profits by increasing the average edges between their hired workers and their customers, and for other similar real-world applications is the same.

In this paper, we formulate a new MDS-N problem, aiming at finding a maximum density subgraph (MDS) measured by the average degree, near or containing a given node (N) in an undirected (unipartite or bipartite) graph. We propose the Slither algorithm and the Slither PageRank (PR) algorithm, both consisting of two steps. The first step is a same graph transformation, which transforms the undirected graph into a connected and weighted graph, and reduces the MDS-N problem to the minimum conductance problem. The second step is based on random walks, specifically the lazy random walk for Slither and the personalized PageRank for Slither PR. The theorem behind this step is the Lovász-Simonovits Theorem. Lower bounds on the densities of the subgraphs found by the two algorithms are given. Additionally, we adopt a simple hierarchically repetition frame over the two algorithms for advancing them. The time complexity of these algorithms are analyzed. All these algorithms are easy to implement.

Our algorithms are mainly compared with other existing local dense subgraph finding algorithms on both unipartite graphs and bipartite graphs, in terms of the densities (i.e., the average degrees) of the found subgraphs, the stability of the choice of the given nodes, and the scalability. Experiments show our algorithms are the fastest among those which can get subgraphs with high densities. Slither tends to explore while Slither PR tends to exploit, which can be inferred in the time consumed and other experimental results. Experiments in terms of the hierarchy show that a small hierarchy (no more than two) enables Slither and Slither PR to achieve satisfactory results. Finally, we apply the MDS-N problem in Twitter, a large social network. Our proposed algorithms successfully detect four subgraphs with high fraud rates according to four given fraudulent user accounts.

2 Related work

2.1 Dense subgraph problems

The densest subgraph problem is generally defined as finding a subset of nodes in an undirected graph whose average degree is maximized. This problem is proved to be a polynomial-time problem and can be optimally solved by max-flow methods [11, 12]. Charikar [7] proposed a linear time greedy algorithm guaranteeing a 1/2-approximation result of this problem. Another frequently used density measure is edge density, which is calculated by diving the number of edges in a subgraph by the maximum possible number of edges in it. Directly maximizing this density measure is not meaningful, because a single edge, with two nodes at its ends, obtains the maximum density. Therefore, the α-quasi-clique is introduced, and requires the number of edges to be no less than the maximum possible number of edges times a threshold parameter $\alpha \in (0, 1)$. There are ways to find single or all α-quasi-clique(s) at once [8, 9]. Other dense subgraph finding problems include maximum clique problem [13, 14], maximal clique problem [15, 16], K-core [17], K-plex [18], Kd-clique [19], K-club [20], etc.

2.2 Local dense subgraph finding

A local dense subgraph finding algorithm is to find an approximation of the densest subgraph near or containing a specified node, with a running time depending largely on the number of edges in the subgraph rather than that in the entire input graph. Andersen [1] proposed an algorithm that finds local dense subgraphs in a bipartite graph, based on the density measure presented in Kannan and Vinay's work [6]. This algorithm produces a subgraph with its density proportional to that of the optimal subgraph and with a running time proportional to the square of the node size of the optimal subgraph times the maximum degree of the input graph. Zhang et al. [10] proposed a set of alternative projected gradient based algorithms HiDDen, which can find local dense subgraphs in an undirected graph, measured by edge density. There are some random-walk-based heuristic algorithms [21, 22], relying on the observation that shortened random walks starting from a subgraph with low conductance are inclined to stay within this subgraph, due to the narrow passage between it and the remaining graph. Besides these heuristic algorithms, some local clustering algorithms [2, 3, 23], including PageRank-like algorithms, are further based on the Lovász-Simonovits Theorem [24] and give some local bounds on the conductance of their found subgraphs. However, our work differs from all these algorithms in finding local dense subgraphs measured by average degree.

2.3 Fraud detection methods

Existing fraud detection methods are globally oriented, and do not support the application of detecting a fraud-related subgraph according to a particular node. We simply summarize several representative methods as follows. 1) Feature-based methods [25–27] usually find fraud activities according to the explicit analysis of various features that

are topology-based or not. 2) Propagation-based methods [21, 22, 28] leverage a set of labeled fraudulent nodes and/or labeled normal nodes, and propagate beliefs of label information throughout the graph for predicting labels of other nodes. Our work and this kind of method are both utilizing belief propagation and starting from particular nodes. But, our work solves a local solution only based on one particular node, and provides theoretical bounds on the density of it. 3) Dense subgraph mining methods [5, 29–31] are based on the key insight that fraudsters will not hire enough workers to make them behave like normal users, but will create lots of links (i.e., edges) towards their customers to make money. That our algorithms can be used in fraud detection is also based on this insight.

3 Problem formulation

For a targeted application, it is rational to assume that any given node is in a specific dense subgraph, wanted by the application.

Consider an undirected and unweighted graph $G = (V, E)$ with a set of nodes V and a set of edges E. Set $n = |V|$. Let $E(X)$ be the set of edges in the subgraph induced by $X \subset V$.

Definition 1 We define the density of X

$$den(X) = \frac{|E(X)|}{|X|} \tag{1}$$

Our goal is to find a $X^* \subset V$, which induces a maximum density subgraph near or containing a given node r (MDS-N) in G. Particularly, for targeted applications with the assumption of $r \in X^*$, the MDS-N problem is solved by finding the X^* such that

$$den(X^*) = \max_{r \in X \subset V} den(X) \tag{2}$$

4 Approach

This section introduces the algorithmic details of our approach to solve the MDS-N problem. The core of our approach is in Section 4.1, which is to transform any undirected (unipartite/bipartite) and unweighted graph into a weighted and connected one, for finding the local densest subgraph. This transformation finishes a reduction from the MDS problem to the minimum conductance problem. The equivalence relation of the two problems can be found in the proof of Proposition 1 for two cases. For one case $sv \in S$, due to the equivalence relation hidden in the later defined (8) and (9) is difficult to formulate, we consolidate the hidden equation and the other one for the other case into a single inequity (the later defined (5)), which is in a unified simple form and further simplifies the proof of Theorem 2. Therefore, after we use local graph search algorithms, e.g., random-walk-based algorithms, to solve the transformed minimum conductance problem, the MDS-N problem can then be solved. In Section 4.2, two random-walk-based algorithms, Slither, based on the lazy random walk, and Slither PageRank (PR), based on the personalized PageRank, are proposed. The two subsections complete an algorithmic flow of our approach. Additionally, we provide a simple hierarchical framework in Section 4.3 to optimize the results of the flow.

4.1 Graph transformation

To transform G into a weighted and connected graph $G_c = (V_c, E_c, W)$, we add a source node sv and a sink node tv in the graph G. Then, for each node $i \in V$, we add two edges to connect i with sv and tv respectively. Let $d(i)$ be the degree of node i, and let d_{max} be the maximum degree of G. The weights on the edges connecting i with sv and with tv are given as m_c and $m_c - d(i)$ respectively, where m_c is a constant such that $m_c > d_{max}$. We develop this transformation inspired by the Goldberg's max-flow algorithm [11]. An example of this transformation with G being a bipartite graph is depicted in Fig. 1. Formally, G_c is written as

i. $V_c = V \cup \{sv, tv\}$
ii. $E_c = E \cup \{(sv, i)|i \in V\} \cup \{(tv, i)|i \in V\}$
iii. $w_{ij} = 1, (i, j) \in E$
iv. $w_{sv,i} = m_c, i \in V$
v. $w_{tv,i} = m_c - d(i), i \in V$
vi. $w_{ij} = 0, (i, j) \notin E_c$

Let $S \subset V_c$. After transforming G into G_c, we build a bridge between subsets of nodes $\{X\}$ in G and subsets of nodes $\{S\}$ in G_c. For G_c, there are four possible categories of S derived by all possible cuts shown in Fig. 2: 1) $sv \notin S$ and $tv \notin S$; 2) $sv \in S$ and $tv \notin S$; 3) $sv \notin S$ and $tv \in S$; 4) $sv \in S$ and $tv \in S$. As the constructed node tv is definitely not in X^*, according to the theory of local Cheeger inequality [32], we adopt a specified subset of nodes $V_c - \{tv\}$ in G_c, and it is unnecessary to consider two categories of S that include tv.

Definition 2 Let $(S, V_c - S)$ be an arbitrary cut of graph G_c with $S \subset V_c$ and $S \neq \emptyset$. We define the conductance of $(S, V_c - S)$ as

$$\Phi(S) = \frac{\mu(\delta(S))}{\min(\mu(S), \mu(V_c - S))} \tag{3}$$

where $\delta(S) = \{(i, j) \in E_c|i \in S, j \in V_c - S\}$ is a set of edges called the edge boundary of S; $\mu(S) = \sum_{i \in S} \mu(i)$

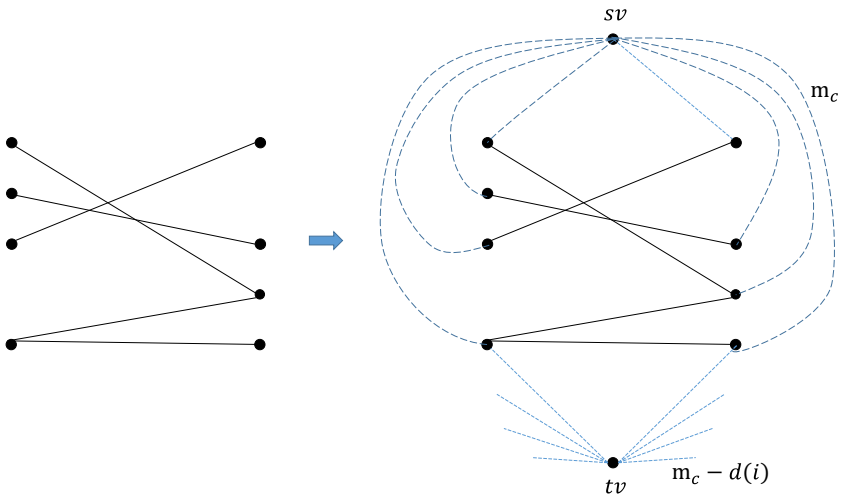


Fig. 1 Instance an unweighted bipartite graph G being transformed into a weighted connected graph G_c. The added edges are dashed lines in G_c

is the volume of S with $\mu(i) = \sum_j w_{ij}$; and $\mu(\delta(S)) = \sum_{(i,j)\in\delta(S)} w_{ij}$ is the volume of $\delta(S)$.

Due to the subgraph derived by the targeted S is relatively small compared to the entire graph G_c, it is rational to assume $\mu(S) \le \mu(V_c - S)$, and (3) is simplified as

$$\Phi(S) = \frac{\mu(\delta(S))}{\mu(S)} \tag{4}$$

Proposition 1 is then proposed to formally give the quantitative relationship between $\{X\}$ in G and $\{S \subset V_c - \{tv\}\}$ in G_c as follows.

Proposition 1 *Let S be an arbitrary subset of nodes in $G_c = (V_c, E_c, W)$, satisfying that $tv \notin S$ and $\mu(S) \le \mu(V_c - S)$. Set $X = S - \{sv\}$, which is a subset of nodes in $G = (V, E)$. A $m_c > d_{max}$ is given. Then*

$$\Phi(S) \ge \frac{1}{1 + 2\frac{|X|}{n}} - \frac{1}{m_c} den(X) \tag{5}$$

Proof For the case $sv \in S$, we can calculate

$$\begin{aligned}
\mu(\delta(S)) &= \sum_{i\in S, j\in V_c - S} w_{ij} \\
&= \sum_{j\in V-X} w_{sv,j} + \sum_{i\in X, j\in V-X} w_{ij} + \sum_{j\in X} w_{tv,j} \\
&= m_c|V - X| + \sum_{i\in X, j\in V-X} w_{ij} + (m_c|X| - \sum_{i\in X} d(i)) \\
&= m_c n + \sum_{i\in X, j\in V-X} w_{ij} - \sum_{i\in X} d(i)
\end{aligned} \tag{6}$$

and

$$\begin{aligned}
\mu(S) &= \sum_{i\in S, j\in V_c - S} w_{ij} + (\sum_{i\in X} d(i) - \sum_{i\in X, j\in V-X} w_{ij}) + 2\sum_{j\in X} w_{sv,j} \\
&= (m_c n + \sum_{i\in X, j\in V-X} w_{ij} - \sum_{i\in X} d(i)) + (\sum_{i\in X} d(i) \\
&\quad - \sum_{i\in X, j\in V-X} w_{ij}) + 2m_c|X| = m_c n + 2m_c|X|
\end{aligned} \tag{7}$$

Substituting (6) and (7) into (4), we have

$$\begin{aligned}
\Phi(S) &= \frac{m_c n + \sum_{i\in X, j\in V-X} w_{ij} - \sum_{i\in X} d(i)}{m_c n + 2m_c|X|} \\
&= \frac{1}{1 + 2\frac{|X|}{n}} - \frac{1}{m_c} \frac{\sum_{i\in X} d(i) - \sum_{i\in X, j\in V-X} w_{ij}}{n + 2|X|}
\end{aligned} \tag{8}$$

Note that

$$den(X) = \frac{|E(X)|}{|X|} = \frac{\sum_{i\in X} d(i) - \sum_{i\in X, j\in V-X} w_{ij}}{2|X|} \tag{9}$$

and constant $n > 0$. Thus (8) implies

$$\Phi(S) \ge \frac{1}{1 + 2\frac{|X|}{n}} - \frac{1}{m_c} den(X) \tag{10}$$

For the case $sv \notin S$, similarly we get

$$\begin{aligned}
\Phi(S) &= \frac{2m_c|X| + \sum_{i\in X, j\in V-X} w_{ij} - \sum_{i\in X} d(i)}{2m_c|X|} \\
&= 1 - \frac{1}{m_c} den(X) \\
&\ge \frac{1}{1 + 2\frac{|X|}{n}} - \frac{1}{m_c} den(X)
\end{aligned} \tag{11}$$

□

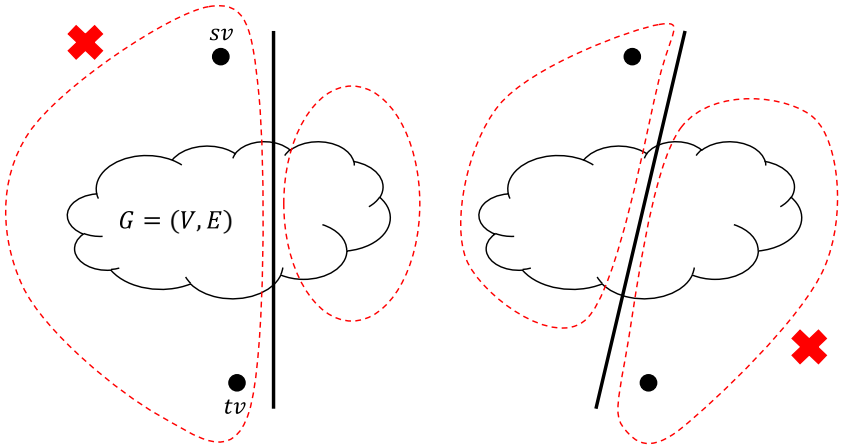


Fig. 2 Four categories of subsets of nodes derived by two kinds of cuts

4.2 The slither and slither PR algorithms

After we have finished the transformation between the two problems in Section 4.1, next is to solve the minimum conductance problem on G_c.

Algorithm 1 Slither or Slither PR algorithm.

Input: $G = (V, E)$ (unweighted graph), r (given node), T (number of iterations)
Output: X^*

1 Transform G into a weighted connected $G_c = (V_c, E_c, W)$.
2 $density \leftarrow 0$
3 $X^* \leftarrow \emptyset$
4 Initialize p_0 by (14).
5 Calculate M by (13).
6 **for** $t = 1$ **to** T **do**
7 $\quad p_t \leftarrow \beta p_0 + (1-\beta) M p_{t-1}$ % Define Slither with $\beta = 0$ and Slither PR with $\beta \in (0, 1)$.
8 $\quad J \leftarrow \arg\max_j den(S_j^t(p_t) - \{sv, tv\})$
9 $\quad$ **if** $den(S_J^t(p_t) - \{sv, tv\}) > density$ **then**
10 $\quad\quad X^* \leftarrow S_J^t(p_t) - \{sv, tv\}$
11 $\quad\quad density \leftarrow den(X^*)$
12 $\quad$ **end**
13 **end**

Before going into the details of the definitions of the variables and the parameters of our forthcoming Slither and Slither PR algorithms (in Alg. 1), we state that our algorithms differ from the algorithms formed by Lovász-Simonovits Theorem only in the addition of a transformation (in Line 1 of Alg. 1) and the $\arg\max_j den$ (in Line 8 of Alg. 1) where the latter are to minimize a conductance. Besides, the two kinds of algorithms share the same rules of random walks and the same calculation of a important term S_j^t, called the sweep-cut.

Here we first present the rules of the lazy random walk. A step of the lazy random walk stays where it is with probability 0.5 and goes from a node $i \in V_c$ to its neighboring node $j \in V_c$ with probability $0.5w_{ij}/\mu(i)$. Set the diagonal matrix $D = diag(\mu(1), \mu(2), \ldots, \mu(n_c))$ with $n_c = |V_c|$, and the n_c-dimensional column vector p_t as the probability distribution over V_c at time t. The lazy random walk, beginning at time 0, can be expressed as

$$p_t = M p_{t-1} = M^t p_0 \tag{12}$$

where the matrix M is calculated as

$$M = (W D^{-1} + I)/2 \tag{13}$$

with I being the $n_c \times n_c$ identity matrix. We set the initial probability distribution p_0 as

$$p_0 = (0, \cdots, 1, \cdots, 0)^{\mathrm{T}} \ st. \ p_0(i) = \begin{cases} 1, i = r \\ 0, otherwise \end{cases} \tag{14}$$

with a given node $r \in V_c - \{sv, tv\}$.

G_c is a connected graph. On a connected graph, it is known that a random walk finally converges to a stationary distribution p_∞, and the convergence is not affected by whatever the value of p_0. Set $2m = \sum_{j \in V_c} \mu(j)$. For a node i, the stationary probability

$$p_\infty(i) = \mu(i)/(2m) \tag{15}$$

is proportional to $\mu(i)$.

At each t, sort $p_t(i)/\mu(i)$ in descending order for $\{i \in V_c\}$. Let $\pi(k)$ be the identifier of the node at position k in the sorted sequence and we obtain

$$\frac{p_t(\pi(k))}{\mu(\pi(k))} \geq \frac{p_t(\pi(k+1))}{\mu(\pi(k+1))} \ (k = 1, 2, \cdots, n_c) \tag{16}$$

Then we set the sweep-cut

$$S_j^t = \{\pi(1), \pi(2), \ldots, \pi(j)\} \ (1 < j \leq n_c) \tag{17}$$

Theorem 1 *For some non-negative integer T, let*

$$\Phi^* = \min_{0 \leq t \leq T} \min_{1 < j \leq n_c} \Phi(S_j^t) \tag{18}$$

Then for each $S \subset V_c$,

$$\sum_{i \in S} p_t(i) - \mu(S)/2m \leq \beta t + \sqrt{\mu(S)}(1 - \frac{1}{8}\Phi^{*2})^t \quad (19)$$

with $\mu(S) \leq \mu(V_c - S) = 2m - \mu(S)$ as we assumed before.

Theorem 1 can be seen as a generalization of the classical Lovász-Simonovits Theorem [24], which additionally introducing a parameter $\beta \in [0, 1)$. The classical one is built based on merely lazy random walks corresponding to $\beta = 0$, and the range of this parameter is extended with the popularity of the personalized PageRank [3, 23, 33] by later researches [3]. For the general $\beta \in [0, 1)$, (12) is written as

$$p_t = \beta p_0 + (1 - \beta) M p_{t-1} \quad (20)$$

The characteristics of the two different kinds of random walks decide that a bigger β makes the corresponding random walk be more inclined to exploit than explore during graph searching. Take these words in mind while tuning β, which is applicable to both the algorithms formed by Lovász-Simonovits Theorem and our algorithms.

Theorem 2 is obtained by combining Theorem 1 with Proposition 1.

Theorem 2 *For some non-negative integer T, let*

$$den^* = \max_{0 \leq t \leq T} \max_{1 < j \leq n_c} den(S_j^t - \{sv, tv\}) \quad (21)$$

Then for each $S \subset V_c$,

$$den^* \geq \beta t + \frac{m_c}{1 + 2\frac{|X|}{n}} - 2\sqrt{2}m_c[1 - \mu(S)^{-\frac{1}{2t}}(\sum_{i \in S} p_t(i) - \mu(S)/2m)^{\frac{1}{t}}]^{\frac{1}{2}} \quad (22)$$

with $X = S - \{sv, tv\}$ and the assumption $\mu(S) \leq 2m - \mu(S)$.

To more clearly identify the lower bounds of den^*, compared to the elegant implicit expression of Φ^* in (19), we place den^* on the left-hand side of (22) and leave the lower bounds of it on the other side. The definitions of the variables and the parameters in the lower bounds have been given before. Like Theorem 1, Theorem 2 also has a strong algorithmic implication, i.e., it can be directly formed into algorithms. The algorithms, Slither corresponding to $\beta = 0$ and Slither PR corresponding to $\beta \in (0, 1)$, are formally presented in Alg. 1 for solving the MDS-N problem.

According to (22), in order to get big lower bounds of den^*, it suggests setting $S = S_j^t$ for each time t, i.e., ordering $p_t(i)/\mu(i)$ for all nodes $i \in V_c$. Then, to choose from these bounds, for each t, the corresponding bound increases with the increase of $\sum_{i \in S_j^t} p_t(i)$ or the decrease of $\mu(S_j^t)$ and $|X| = |S_j^t - \{sv, tv\}|$. Thus, to obtain the maximum $den(X)$, all the values of $(\sum_{i \in S_j^t} p_t(i), \mu(S_j^t), |X|)$ should be evaluated for each t.

4.3 The Hierarchical Algorithms

Algorithm 2 Hierarchical Slither or Hierarchical Slither PR algorithm.

Input: $G = (V, E)$ (unweighted graph), r (given node), T (number of iterations), K (number of levels in the hierarchy)
Output: X^*

```
Transform G into a weighted connected G_c = (V_c, E_c, W).
density ← 0
X ← ∅
X* ← ∅
for k = 1 to K do
    Initialize p_0 by (14).
    Calculate M by (13).
    for t = 1 to T do
        p_t ← βp_0 + (1 − β)Mp_{t−1} % Define Slither with β = 0 and Slither PR with β ∈ (0, 1).
        J ← arg max_j den(S_j^t(p_t) − {sv, tv})
        if den(S_J^t(p_t) − {sv, tv}) > density then
            X ← S_J^t(p_t) − {sv, tv}
            density ← den(X)
        end
    end
    X* ← X if r ∈ X else return X*
    G ← the subgraph derived by X*
    Transform G into a weighted connected G_c = (V_c, E_c, W).
end
```

Inspired by the hierarchical algorithmic structure adopted in [10] for advancing graph mining, we improve the Slither (PR) algorithm by putting it into a hierarchical frame with K levels, called the hierarchical Slither (PR) algorithm, summarized in Alg. 2. It is a heuristic technique, which only repeats the procedure of Slither (PR), but at each round k ($k = 1, \cdots, K$) in Alg. 2, a new MDS-N problem with the same given node r is solved, and a new G is derived by the subset of nodes X obtained from the last round, till $r \notin X$. Slither and Slither PR are the special cases of their hierarchical versions with setting $K = 1$. With the increase of K, the hierarchical algorithms will not get worse results than the non-hierarchical ones, meaning that the former have the same lower bounds as the latter.

4.4 Time complexity

Line 7 in Alg. 1 runs in $\mathcal{O}(|E_c|)$ for both lazy random walk and PageRank, and Line 8 runs in $\mathcal{O}(|V_c|\log(|V_c|)$ for sorting, with $|E_c| = |E|+2n$ and $|V_c| = |V|+2$. Therefore, Slither (PR) runs in a total of $\mathcal{O}(T(|E_c| + |V_c|\log(|V_c|)))$, and hierarchical Slither (PR) runs in a total of $\mathcal{O}(KT(|E_c|+|V_c|\log(|V_c|)))$. Note that, hierarchical Slither (PR) runs more optimistically than what its time complexity looks like, because even the first round of the k-for-loop reduces the size of G largely.

Importantly, if we move Line 8-12 in Alg. 1 (and the same procedure in Alg. 2) out of the t-for-loop, Slither (PR) obtains an $\mathcal{O}(T|E_c|+|V_c|\log(|V_c|))$ and hierarchical Slither (PR) obtains an $\mathcal{O}(K(T|E_c| + |V_c|\log(|V_c|)))$. This move will replace all t in Theorem 2 with T, leaving only one lower bound of den^* with respect to T. This lower bound is no less than the minimum one of the original bound set. Nevertheless, this move saves $T - 1$ times of sorting in calculating S_j^t, which significantly reduces algorithms' runtime. We recommend to adopt this move in tasks with their bottlenecks being time, i.e., tasks taking large graphs as inputs. In one of this kind of tasks, T is often set to a small value for saving time too. This small T makes the lower bound with respect to T close to the minimum one, and makes the adoption of the move in our algorithms succeed in both effectiveness and efficiency for these tasks.

5 Experiments

In this section, we use HS and HSPR to represent the hierarchical Slither algorithm and the hierarchical Slither PR algorithm respectively, for short. $m_c = d_{max} + 1$ is set for both HS and HSPR, and $\beta = 0.74$ is set for HSPR. Although any $\beta \in (0, 1)$ can be set for HSPR, we choose the comparatively big β to clearly show the different search style of HSPR than HS. As there is no existing work that handles the MDS-N problem, we compare our proposed algorithms mainly with two local dense subgraph finding algorithms: FindDense [1] and HiDDen [10], which are using different density measures. All experiments are carried out on a 2.2GHZ Intel Xeon E5-2407 server with 18 GB RAM.

5.1 Experiments on real-world graphs

In this set of experiments, we configure our algorithms with the setting of $K = 10$ and $T = 100$. $K = 10$ is also set for HiDDen. We evaluate algorithms on six publicly-available real-world graphs, of which, three are unipartite graphs including Co-Author [10], Crocodile [34], and Brightkite [35], and another three are bipartite graphs including Epinions [36], Facebook [37], and Amazon [38]. Co-Author is built on a snapshot of AMiner citation dataset [39] collected until the year 2011, covering five research areas: data mining, machine learning, database, information retrieval and bioinformatics. Crocodile is a Wikipedia page-page network on the topic crocodiles. Brightkite is a location-based online social network. Epinions is an Epinions signed who-trust-whom online social network. Facebook consists of social circles from Facebook. Amazon is built on the category Computers of the Amazon product dataset, crawled from May 1996 to July 2014. The main characteristics of the six real-world graphs are shown in Table 1. Note that a bipartite graph has two sets of nodes.

Table 2 shows the densities of subgraphs found by algorithms. For each real-world graph, Charikar's greedy algorithm was used to search the global densest subgraph. Then we randomly sampled ten nodes from it as the given nodes of MDS-N for local subgraph finding algorithms, and averaged the results out. The densities of subgraphs found by the greedy algorithm, labeled as *Greedy*, are presented as baselines. HS or HSPR achieves the maximum densities among all algorithms, even better than the baselines. Between HS and HSPR, HS wins when densest subgraphs are more easy to be found by exploration search, while HSPR wins when exploitation search is better. To further explain the performance of HS and HSPR, we also show the results of hierarchical PageRank (HPR) in Table 2. HPR is a hierarchical version of PageRank and has the same parameter setting as HSPR, which locally finds the subgraphs that are related to cuts with minimum conductance. Most of the time, HPR performs the worst. Except for Co-Author, the only acceptable result of HPR is from Crocodile, and for Crocodile, HSPR performs better than HS, showing the superiority of exploitation search of both HPR and HSPR in Crocodile. For each of the other four graphs, HSPR performs not always the best, but achieves more than 90% density of the subgraph found by HS, compared to the worst result of HPR, owing to the function of graph transformation used in HSPR.

Table 3 presents the results of the top five hierarchies of three hierarchical algorithms: HiDDen, HS, and HSPR. It can be seen that the bigger β of HSPR enables it to

Table 1 Some real-world graphs used in our experiments

	Graph	Nodes	Edges
Unipartite	Co-Author	38622	200332
	Crocodile	11631	341691
	Brightkite	58228	428156
Bipartite	Epinions	(95318,84601)	841372
	Facebook	(3663,4037)	88234
	Amazon	(28158,4266)	28603

Table 2 The densities of subgraphs found by algorithms, averaged among given nodes sampled from the global densest subgraphs detected by Charikar's *Greedy* algorithm

Alg. / Graph	*Greedy*	FindDense	HiDDen	HPR	HS	HSPR
Co-Author	**13**	**13**	**13**	**13**	**13**	**13**
Crocodile	50.79	14.85	42.94	42.93	42.15	**50.99**
Brightkite	30.99	27.90	30.84	13.61	31.11	**34.08**
Epinions	66.05	52.77	54.21	32.24	**68.16**	66.24
Facebook	44.23	40.64	37.44	22.89	**46.15**	42.20
Amazon	**1.38**	1.21	1.00	0.88	**1.38**	**1.38**

Best results are in bold

converge faster than HS (with smaller k to get its optimum). Both HS and HSPR reach or approach their optimum in two hierarchies, indicating that we could set far smaller hierarchies ($k \ll 10$) for them.

5.2 Experiments on Synthetic Graphs

In this set of experiments, we configure our algorithms with the setting of $K = 10$ and $T = 100$. $K = 10$ is also set for HiDDen. From the real-world bipartite graph Epinions, we sampled a random (2000,2000) scale of nodes as the base of two synthetic bipartite graphs, depicted in Fig. 3a with the label "Separate" and in Fig. 3b with the label "Overlapping" separately. Each synthetic graph was constructed by injecting two dense subgraphs. In "Separate", injected a dense subgraph with edge density 0.04 ranged nodes ([0, 150], [0, 150]) and a dense subgraph with edge density 0.01 ranged nodes ([150, 300], [150, 300]). In "Overlapping", injected a dense subgraph with edge density 0.04 ranged nodes ([0, 150], [0, 150]) and a dense subgraph with edge density 0.0025 ranged nodes ([0, 450], [0, 450]). For a synthetic graph, we tested each of the 300 nodes in ([0, 150], [0, 150]) as a given node of MDS-N, and the results were averaged and shown in Table 4. Two metrics, accuracy $\left(AC = \frac{TP+TN}{TP+FN+FP+TN}\right)$ and F-measure $\left(F = \frac{2}{1/precision+1/recall}\right)$, are used.

In Table 4, algorithms FindDense, HiDDen, HS and HSPR all find subgraphs derived by nodes ([0, 150], [0, 150]) with high accuracy (> 0.95). But, for F-measure, only HS and HSPR get values > 0.84 on both two synthetic graphs. HS performs better than HSPR here, due to their algorithmic characteristics. Compared with HSPR, especially which with big β, HS is more inclined to explore than exploit when searching local densest subgraphs, which seems to make HS be beneficial in experiments of this subsection.

Additionally, Fig. 4 depicts a box plot to show the concentration of 300 values of F-measure for each algorithm evaluated on each synthetic graph. It can be seen that with more exploitation, HSPR gets the narrowest range of values of F-measure for each synthetic graph, demonstrating its great stability.

5.3 Scalability

In this set of experiments, we configure our algorithms with the setting of $K = 10$ and $T = 100$. $K = 10$ is also set for HiDDen. We used the category "Apps for Android" of the Amazon product dataset to construct a user-product bipartite graph, from which we sampled a series of subgraphs with an increasing number of edges for evaluating the scalability of algorithms. As depicted in Fig. 5, HiDDen runs the

Table 3 The densities of the found subgraphs vs. the hierarchies

Graph	Algorithm	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Crocodile	HiDDen	34.10	**42.94**	39.59	3.53	3.53
	HS	27.97	**42.15**	42.15	42.15	42.15
	HSPR	**50.99**	50.99	50.99	50.99	50.99
Epinions	HiDDen	28.14	40.00	49.55	53.85	**54.16**
	HS	4.68	66.40	**68.16**	68.16	68.16
	HSPR	**66.24**	66.24	66.24	66.24	66.24

Best results are in bold

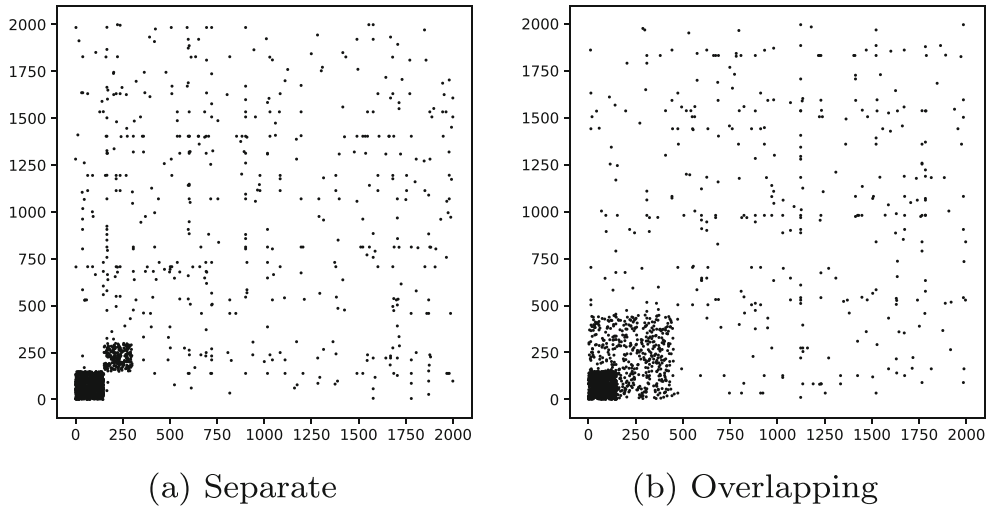


Fig. 3 The scatter plots of two synthetic bipartite graphs, each injected with two dense subgraphs: (a) Separate: a dense subgraph with edge density 0.04 ranged nodes ([0, 150], [0, 150]) and a dense subgraph with edge density 0.01 ranged nodes ([150, 300], [150, 300]); (b) Overlapping: a dense subgraph with edge density 0.04 ranged nodes ([0, 150], [0, 150]) and a dense subgraph with edge density 0.0025 ranged nodes ([0, 450], [0, 450])

Table 4 The accuracy (AC) and F-measure (F) of algorithms to find the known densest subgraphs, derived by nodes ([0, 150], [0, 150]), of the two synthetic bipartite graphs in Fig. 3, averaged among the given nodes in these two subgraphs

	Separate		Overlapping	
	AC	F	AC	F
FindDense	0.9578	0.6129	0.9635	0.6791
HiDDen	0.9695	0.7764	0.9811	0.8566
HS	**0.9819**	**0.8798**	**0.9873**	**0.9121**
HSPR	0.9779	0.8404	0.9825	0.8700

Best results are in bold

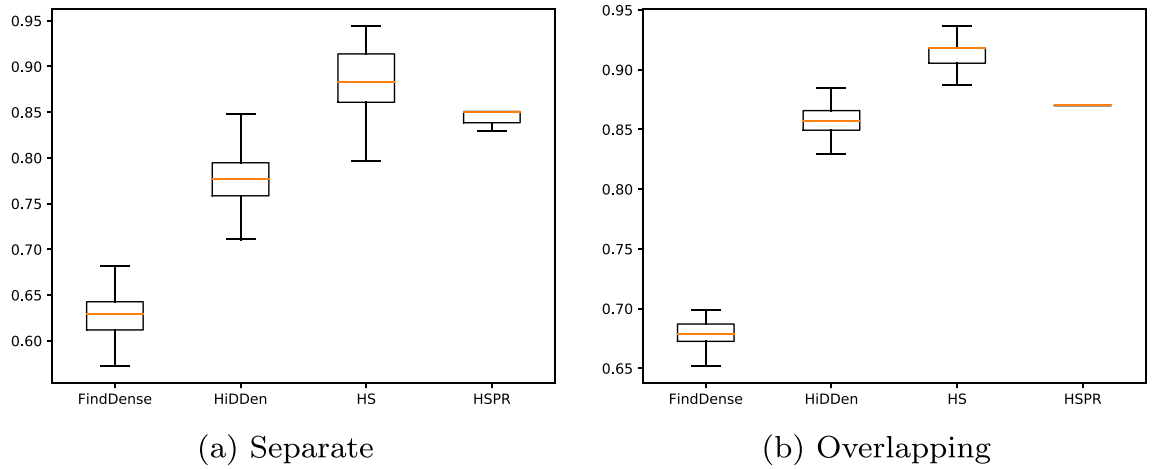


Fig. 4 The stability of algorithms to find the known densest subgraphs, derived by nodes ([0, 150], [0, 150]), of the two synthetic bipartite graphs in Fig 3, with respect to F-measure

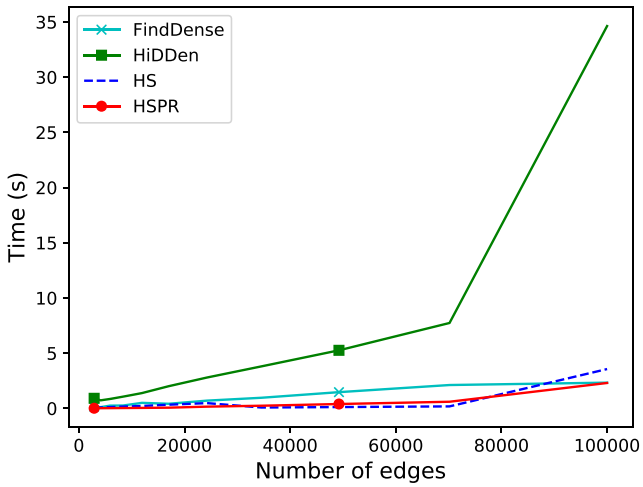


Fig. 5 The scalability of algorithms

slowest, and the gap between its running time and that of the other three algorithms grows rapidly with the number of edges. HSPR shows its faster convergence than HS when the number of edges becomes large. The running time of FindDense increases the slowest with the number of edges.

5.4 Large graph application: targeted fraud detection on twitter

In social/commercial networks, fraudsters often hire workers, being users of these networks, to add links to particular users/products, which leads to unusually dense subgraphs in these networks. These links are added through user behaviors like following, buying, and reviewing. Compared with finding subgraphs sparsely connected to the remaining network, detecting fraud based on dense subgraph finding is naturally camouflage-resistant [5]. This paper provides a way to detect a particular fraud-related subgraph according to a given node, named as targeted fraud detection. Take the large follower-followee social network Twitter as an example. We use the Twitter dataset, which contains 41.7 million users and 1.47 billion social relations crawled in July 2009 [40]. We adopt the criteria in [30] with some modifications for labeling fraudulent user accounts, which is summarized below.

- The account is suspended or deleted.
- The text information of the account is associated with malware or adware, or the account is a follower of such an account.
- The account has a suspicious username, or is followed by users with suspicious usernames, e.g., usernames having identical prefixes/suffixes.
- The account has very few tweets or very few different tweets (< 5), but relatively more followees (> 20).

Algorithms, FindDense, HiDDen, HS and HSPR, were verified on the Twitter dataset with four user accounts as the given nodes: "@tweepme", "@twitbacks", "@tweepi" and "$id = 14868835$". The first three accounts, publishing obvious follower buying advertisements like "helps you get more followers on twitter quick and easy", are still active in gaining followers. $id = 14868835$ is a suspended account found in a forum. Due to the memory limitation of our 18 GB RAM, we cannot load the entire Twitter dataset that contains 24.3 GB of unstructured follower-followee data. For a given node, we cut off a piece from the Twitter dataset as an extracted dataset, with consecutive follower account IDs that contains the given node's ID, and all their followees. An extracted dataset has 100 million edges. For instance, the ID of @tweepme is 23711158, and the follower IDs of its corresponding extracted dataset range from 22563769 to 24907792. The node scales of four extracts corresponding to four accounts are given in Table 5.

For large datasets, we set $k = 1$ for three hierarchical algorithms (i.e., HiDDen, HS and HSPR), and for HS and HSPR, set $T = 10$ and moved Line 10-14 in Alg. 2 out of the t-for-loop. For each algorithm, the densities of the detected fraud-related subgraphs corresponding to four accounts are listed in Table 6. Averaged over the four accounts, the running time of FindDense is 2475s, of HS is 1545s, and of HSPR is 1488s. The results of HiDDen are not shown in Table 6, because HiDDen cannot get any result in 5h. Note that the setting of $k = 1$, etc. makes HS and HSPR run faster even than FindDense, and the results of them are still much better than FindDense, which means HS and HSPR can get acceptable results, that are better than the results of other algorithms used to solve the MDS-N problem, with very small hierarchies (e.g., $k = 1$) and well before convergence (e.g., with $T = 10$). The algorithmic characteristic of more exploitation often causes HSPR to have an advantage in time, but in this set of experiments, HS with more exploration achieves the best results.

Table 5 The node scales of the extracts corresponding to four given nodes in Twitter network

Account	Extract
@tweepme	(1964922,15789443)
@twitbacks	(944378,15710119)
@tweepi	(711255,10975331)
$id = 14868835$	(816078,15995935)

Table 6 The densities of the detected fraud-related subgraphs corresponding to four given nodes in Twitter network

Alg. / Account	FindDense	HiDDen	HS	HSPR
@tweepme	178.62	-	**297.90**	296.22
@twitbacks	22.09	-	**43.91**	33.27
@tweepi	23.47	-	**26.77**	**26.77**
$id = 14868835$	17.19	-	**31.72**	25.00

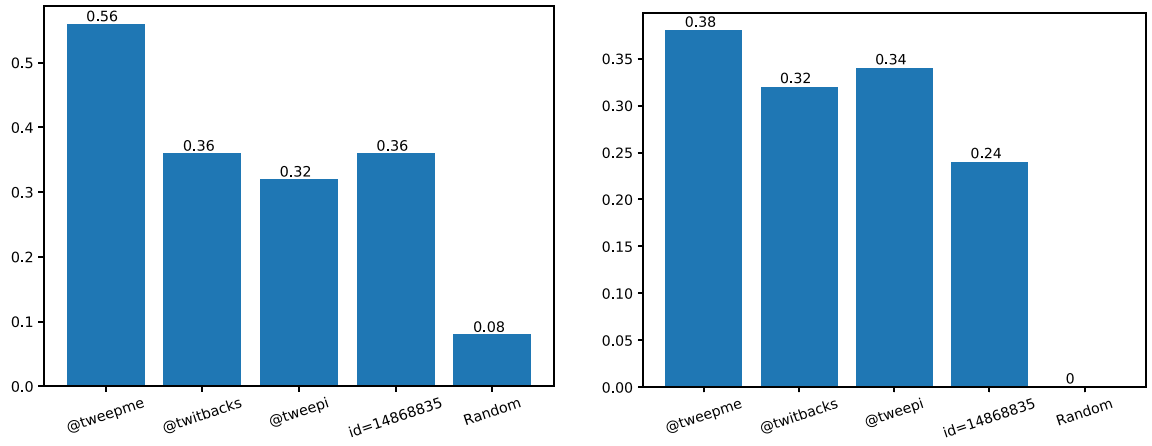
Best results are in bold

Then, we focus on subgraphs detected by HS. The node scales of these subgraphs are: (598,2726) for @tweepme, (812,2200) for @twitbacks, (732,1756) for @tweepi, and (273,2304) for $id = 14868835$. From nodes in each detected subgraph, we randomly selected 50 workers and 50 fraudulent followees, and labeled them according to the previously listed criteria and their profiles and tweets publicly available in https://twitter.com/. The ratios of determined workers and determined fraudulent followees are presented in Fig 6. For comparison, we add another case "Random" by randomly labeling 100 users, consisting of 50 followers and 50 followees, and display its ratios of determined fraud in Fig. 6.

Except for "Random", the ratios of other cases in Fig. 6(a) are over 30% (@tweepme achieves the highest 56%). Three ratios in Fig. 6b are over 30% and $id = 14868835$ is 24%. "Random" obtains the lowest ratios: 8% for determined workers and 0 for determined fraudulent followees. That shows HS can target fraud effectively according to a given node in a real-world network.

6 Conclusion and future work

This paper introduces the MDS-N problem, a local dense subgraph finding problem based on the average degree measure. We present a graph transformation, which transforms an undirected and unweighted graph into a connected and weighted graph, and reduces the MDS-N problem to the minimum conductance problem. After the transformation, the proposed lazy-random-walk-based Slither algorithm and PageRank-based Slither PR algorithm "walk" on the connected and weighted graph to find the densest subgraph according to a particular node. A simple hierarchically repetition frame is used to further advance the two algorithms. Experiments conducted on both unipartite graphs and bipartite graphs show our algorithms are the fastest among algorithms that can find subgraphs with high densities. Slither tends to explore while Slither PR tends to exploit during their "walks". A small hierarchy (no more than two) enables Slither and Slither PR to achieve satisfactory results. We verify the MDS-N problem and



(a) determined workers (b) determined fraudulent followees

Fig. 6 The ratios of determined workers and determined fraudulent followees in subgraphs detected by HS and in 100 randomly selected nodes

the proposed algorithms on a large social network Twitter, the experimental results of which show our algorithms can successfully detect local fraud-related subgraphs based on particular fraudulent user accounts.

In the future, we plan to: (I) combine our two hierarchical algorithms into an algorithm with a dynamically adjusted β for being better applied in different graphs, and (II) utilize graph partition techniques before belief propagation (i.e., walks) to decrease time complexity.

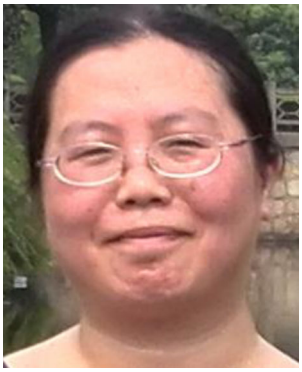
Acknowledgements This work is partially supported by Sub Project of Independent Scientific Research Project(No. ZZKY-ZX-03-02-04).

References

1. Andersen R (2010) A local algorithm for finding dense subgraphs. ACM Trans Algorithm 6(4):1–12
2. Spielman DA, Teng S-H (2013) A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. SIAM J Comput 42(1):1–26
3. Andersen R, Chung F, Lang K (2006) Local Graph Partitioning using PageRank Vectors. In: 2006 47Th annual IEEE symposium on foundations of computer science
4. Tsourakakis CE, Bonchi F, Gionis A, Gullo F, Tsiarli MA (2013) Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 104–112
5. Hooi B, Song HA, Beutel A, Shah N, Shin K, Faloutsos C (2016) FRAUDAR: Bounding Graph fraud in the face of camouflag. In: International conference on knowledge discovery and data mining, pp 895–904
6. Kannan R, Vinay V (1999) Analyzing the structure of large graphs, Technical report
7. Charikar M (2000) Greedy approximation algorithms for finding dense components in a graph. Approximation algorithms for combinatorial optimization, pp 84–95
8. Abello J, Resende MGC, Sudarsky S (2002) Massive quasi-clique detection. In: Latin american symposium on theoretical informatics, pp 598–612
9. Uno T (2010) An efficient algorithm for solving pseudo clique enumeration problem. Algorithmica 56(1):3–16
10. Zhang S, Zhou D, Yildirim MY, Alcorn S, He J, Davulcu H, Tong H (2017) HiDDen: Hierarchical dense subgraph detection with application to financial fraud detection. In: Proceedings of the 17th SIAM International Conference on Data Mining, pp 570–578
11. Goldberg AV (1984) Finding a Maximum Density Subgraph. Technical report, University of California, Berkeley
12. Gallo G, Grigoriadis MD, Tarjan RE (1989) Fast parametric maximum flow algorithm and applications. SIAM J Comput 18(1):30–55
13. Håstad J (1999) Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math 182(1):105–142
14. Feige U (2004) Approximating maximum clique by removing subgraphs. SIAM J Discret Math 18(2):219–225
15. Motzkin TS, Straus EG (1965) Maxima for graphs and a new proof of a theorem of turán. Can J Math 17:533–540
16. Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Commun ACM 16(9):575–577
17. Seidman SB (1983) Network structure and minimum degree. Soc Netw 5(3):269–287
18. Seidman SB, Foster BL (1978) A graph-theoretic generalization of the clique concept. J Math Sociol 6(1):139–154
19. Luce RD (1950) Connectivity and generalized cliques in sociometric group structure. Psychometrika 15(2):169–190
20. Mokken RJ (1979) Cliques, clubs and clans. Qual Quant 13(2):161–173
21. Cao Q, Sirivianos M, Yang X, Pregueiro T (2012) Aiding the detection of fake accounts in large scale social online services. In: Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation, pp 197–210
22. Jia J, Wang B, Gong NZ (2017) Random walk based fake account detection in online social networks. In: 2017 47Th annual IEEE/IFIP international conference on dependable systems and networks (DSN), pp 273–284
23. Fogaras D, Rácz B (2004) Towards scaling fully personalized pagerank. In: International workshop on algorithms and models for the web-graph, pp 105–117
24. Lovasz L, Simonovits M (1990) The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In: 31St annual symposium on foundations of computer science, pp 346–354
25. Jindal N, Liu B (2008) Opinion spam and analysis. In: Proceedings of the 2008 international conference on web search and data mining, pp 219–230
26. Grier C, Thomas K, Paxson V, Zhang M (2010) @Spam The Underground on 140 Characters or Less. In: Proceedings of the 17th ACM conference on Computer and communications security, pp 27–37
27. Jiang M, Cui P, Beutel A, Faloutsos C, Yang S (2014) Catchsync: catching synchronized behavior in large directed graphs. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 941–950
28. Yang C, Harkreader R, Zhang J, Shin S, Gu G (2012) Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In: Proceedings of the 21st international conference on World Wide Web, pp 71–80
29. Prakash BA, Seshadri M, Sridharan A, Machiraju S, Faloutsos C (2010) Eigenspokes: Surprising Patterns and Community Structure in Large Graphs. In: Pacific-asia conference on knowledge discovery and data mining, pp 435–448
30. Shah N, Beutel A, Gallagher B, Faloutsos C (2014) Spotting suspicious link behavior with fbox: an adversarial perspective. In: 2014 IEEE International conference on data mining, pp 959–964
31. Shin K, Hooi B, Faloutsos C (2016) M-zoom: Fast dense-block detection in tensors with quality guarantees. In: Joint european conference on machine learning and knowledge discovery in databases, pp 264–280
32. Chung F (2007) Random walks and local cuts in graphs. Linear Algebra Appl 423(1):22–32
33. Haveliwala TH (2003) Topic-sensitive pagerank:A context-sensitive ranking algorithm for web search. IEEE Trans Knowl Data Eng 15(4):784–796
34. Rozemberczki B, Allen C, Sarkar R (2021) Multi-scale Attributed Node Embedding. J Compl Netw 9:(2)
35. Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: user movement in location-based social networks. In: ACM SIGKDD International conference on knowledge discovery and data mining, pp 1082–1090
36. Leskovec J, Huttenlocher D, Kleinberg J (2010) Signed networks in social media. In: 28Th ACM conference on human factors in computing systems, pp 1361–1370
37. McAuley J, Leskovec J (2012) Learning to discover social circles in ego networks. In: Advances in neural information processing systems, vol 1, pp 539–547

38. McAuley J, Leskovec J (2013) Hidden factors and hidden topics: understanding rating dimensions with review text. In: Proceedings of the 7th ACM conference on Recommender systems, pp 165–172
39. Tang J, Zhang J, Yao L, Li J, Zhang L, Su Z (2008) Arnetminer: Extraction and mining of academic social networks. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 990–998
40. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media?. In: Proceedings of the 19th international conference on World Wide Web, pp 591–600

Zixuan Deng received the B.S. degree in information and computing science from Shanxi Normal University in 2014, finished the M. S. program and is currently pursuing her Ph.D. degree in Computer Science and Technology from the University of Electronic Science and Technology of China, Chengdu, China. Her current research interests include decision making, human computer collaboration, and data mining.



Yanping Xiang received the B.S. and M.S. degrees in computer science from Shanghai Jiao Tong University, Shanghai, China, in 1993 and 1996, respectively, and the Ph.D. degree in system engineering from the National University of Singapore, Singapore, in 2003.

She was a Research Fellow with the Industrial and System Engineering Department, National University of Singapore. In 2008, she joined the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, where she is currently a Professor with the Collaborative Autonomic Computing Laboratory. Her research interests include decision analysis, decision system, autonomic computing, and cloud computing.