HiDDen: Hierarchical Dense Subgraph Detection with Application to Financial Fraud Detection

Si Zhang* Dawei Zhou* Mehmet Yigit Yildirim* Scott Alcorn†
Jingrui He* Hasan Davulcu* Hanghang Tong*

Abstract

Dense subgraphs are fundamental patterns in graphs, and dense subgraph detection is often the key step of numerous graph mining applications. Most of the existing methods aim to find a single subgraph with a high density. However, dense subgraphs at different granularities could reveal more intriguing patterns in the underlying graph. In this paper, we propose to *hierarchically* detect dense subgraphs. The key idea of our method (HiDDen) is to envision the density of subgraphs as a *relative* measure to its background (i.e., the subgraph at the coarse granularity). Given that the hierarchical dense subgraph detection problem is essentially a nonconvex quadratic programming problem, we propose effective and efficient alternative projected gradient based algorithms to solve it. The experimental evaluations on real graphs demonstrate that (1) our proposed algorithms find subgraphs with an up to 40% higher density in almost every hierarchy; (2) the densities of different hierarchies exhibit a desirable variety across different granularities; (3) our projected gradient descent based algorithm scales *linearly* w.r.t the number of edges of the input graph; and (4) our methods are able to reveal interesting patterns in the underlying graphs (e.g., synthetic ID in financial fraud detection).

Keywords— Hierarchical dense subgraph detection, financial fraud detection, graph mining

1 Introduction

Dense subgraphs offer many meaningful insights in graphs. More often than not, extracting dense subgraphs (i.e., to find cohesively connected subgraphs with a large density) is of key importance in numerous application domains. For example, in the co-authorship networks, a dense subgraph could represent a group of researchers with similar research interests [10]. In the product-review bipartite network, extracting dense subgraphs could help find suspicious fake reviews and detect fraudsters [13]. Moreover, by finding dense subgraphs from a protein-protein interaction (PPI) network, scientists could discover new protein complexes [25].

A major difference behind different dense subgraph detection techniques lies in the specific definition of the density. For example, [4] proposed a greedy algorithm to maximize the *average degree* ($\frac{2m_s}{n_S}$) and extract the *densest subgraphs*. Moreover, the *edge surplus* (i.e., the surplus between the number of edges in the extracted subgraph and its corresponding *α-quasi-clique*) is used to find a denser subgraph than the *densest subgraph* [24].

However, the implicit assumption behind these existing works is to find dense subgraphs at a *single* granularity, that is, to extract one or more exclusive partitions from the given graph. On the other hand, we might gain more insights by finding dense subgraphs at different granularities. For example, given a co-authorship network, the existing methods may successfully extract a dense subgraph containing a group of well established researchers who closely collaborate with each other. By finding dense subgraphs at coarser granularities, we could find a larger subgraph containing those senior researchers as well as other mid-career researchers in the related areas. We can further extract an even larger subgraph containing all those researchers and their Ph.D students. In this example, detecting dense subgraphs at different granularities helps reveal a more comprehensive view of the intrinsic collaboration relationships in the underlying graph.

Despite its key importance, it remains a challenging task to hierarchically detect dense subgraphs. Specifically, the following questions have largely remained open. First (*Q1. Formulation*), it is not clear how to formulate the hierarchical dense subgraph detection from the optimization perspective. Some existing methods model the flat dense subgraph detection (i.e., with only one hierarchy) as a quadratic programming optimization problem constrained on a simplex. Yet, limited by the simplex constraint, it is unknown how to generalize those methods to different hierarchies in a single optimization problem. Second (*Q2. Algorithm*), despite the various density definitions, most of them lead to a problem with an unknown complexity [24] or a NP-hard

*Arizona State University. Email:{szhan172, dzhou23, yigity-ildirim, jingrui.he, HasanDavulcu, hanghang.tong}@asu.edu
†Early Warnings LLC. Email: scott.alcorn@earlywarning.com

problem [3], even though there exist some approximate algorithms to solve those problems in polynomial time. How can we develop an effective and scalable solver for hierarchical dense subgraph detection problem? Third (*Q3. Generalizations*), most of the existing methods are designed for unipartite graphs, independent of the specific queries. How can we detect dense subgraphs on rich graphs (e.g., bipartite graphs)? How can we find (hierarchical) dense subgraphs w.r.t certain query nodes (i.e., query-specific hierarchical dense subgraph detection problem)?

This paper addresses the hierarchical dense subgraph detection problem, aiming to answer all these questions. The main contributions of this paper are:

1. **Formulation.** We formulate the hierarchical dense subgraph detection problem from optimization perspective. The key idea is to maximize the number of edges and minimize the total penalties on the missing edges in the subgraphs in all hierarchies.
2. **Algorithms and Analysis.** We propose an effective algorithm (HiDDen) based on the alternative projected gradient descent method. Our analysis shows that the proposed method converges to a stationary point with a *linear* time complexity w.r.t the number of edges in the graph.
3. **Variants.** We further generalize the proposed HiDDen algorithm to (1) extract hierarchical dense subgraphs on bipartite graphs, and (2) detect query-specific hierarchical dense subgraphs.
4. **Evaluations and Case Studies.** We conduct extensive experiments to validate the effectiveness and the efficiency of our algorithms. Our experimental results show that (1) our algorithms can find unipartite/bipartite subgraphs with an up to 40% higher density than existing methods do in almost each hierarchy; (2) the densities of different hierarchies exhibit a desirable variety across different resolutions; (3) our algorithms achieve a better tradeoff between the running time and density quality and scale *linearly.* Case studies on academic group finding and synthetic identity fraud detection show that our methods are able to reveal some interesting patterns in the underlying graphs.

The rest of paper is organized as follows. Section 2 defines the hierarchical dense subgraph detection problem. Section 3 presents an optimization-based solution, followed by two variants in Section 4. Section 5 presents experimental results. Related work and conclusion are given in Section 6 and Section 7, respectively.

2 Problem Definitions

In this section, we give the formal definitions of hierarchical dense subgraph detection problems. Table 1 summarizes the main symbols and notations used throughout the paper. We use bold uppercase letters for matrices (e.g., $\mathbf{A}$), bold lower cases for vectors (e.g., $\mathbf{x}$), and lower cases for scalars (e.g., d). We denote $\mathbf{A}'$ as the transpose of the matrix $\mathbf{A}$. We use the superscript k to index the kth hierarchy (e.g., $\mathbf{x}^k$ as the indicator vector of the kth hierarchy), and use the subscript i to index the ith entry of a vector.

Table 1: Symbols and Notation

Symbols	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	a graph
$\mathbf{A}$	the adjacency matrix of the given graph $\mathcal{G}$
$\mathbf{I}$	an identity matrix
$\mathbf{0}, \mathbf{1}$	a vector of all 0s and all 1s, respectively
$\mathbf{x}^k, \mathbf{y}^k$	the indicator vectors of nodes in the kth hierarchy subgraph
m, n	the number of edges and nodes of the graph $\mathcal{G}$
m_s, n_s	the number of edges and nodes of the subgraph
K	the number of hierarchies
d^k	the density of the kth hierarchy subgraph
p	the penalty parameter for the missing edges, $p > 0$
η	the parameter that controls the density ratio, $\eta > 1$
β	the coefficient of the regularization term
j, k	hierarchy indices of subgraphs

Given a graph $\mathcal{G}$, we want to extract dense subgraphs in different hierarchies. Throughout our paper, we represent the dense subgraphs by the corresponding subgraph indicator vectors $\mathbf{x}^k$. The subgraph indicator vector $\mathbf{x}^k$ is a binary vector, and the ith node belongs to the sugraph in the kth hierarchy iff $\mathbf{x}_i^k = 1$. In order to find meaningful subgraphs in different hierarchies (e.g., to avoid finding identical dense subgraphs across different hierarchies), we require that for two dense subgraphs in adjacent hierarchies ($k-1$ and k, $\forall\ 1 < k \le K$), (1) the corresponding densities should differ significantly from each other (i.e., $d^k \ge \eta d^{k-1}$); and (2) the corresponding node sets are nested, i.e., $\mathcal{V}^k \subseteq \mathcal{V}^{k-1}$.

Figure 1 presents an illustrative example. In the figure, the original graph (at the bottom) has three obvious hierarchies, including (1) the first hierarchy with three cliques (dense blocks with all except diagonal entries nonzero) and some inter-clique edges, (2) the second hierarchy with two of the three cliques and some inter-clique edges, and (3) the third hierarchy with the largest clique. A straightforward heuristic to find these dense subgraphs is to run an existing dense subgraph detection method *recursively*: it first runs on the input graph and retrieves the first hierarchy; then it runs the same algorithm on the resultant subgraph and extracts the second hierarchy, so on and so forth. However, such a recursive heuristic might return trivial solutions (e.g., identical dense subgraphs across different hierarchies). Moreover, it is not clear what the overall objective function the recursive heuristic aims to optimize. This is exactly what this paper aims to address. Formally, the hierarchical dense subgraph detection problem is defined as follows.

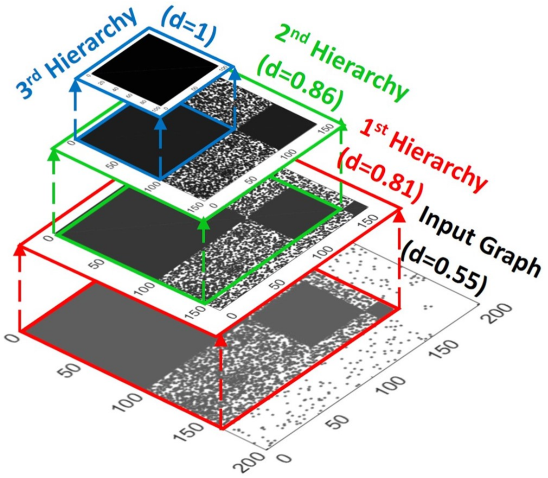


Figure 1: An illustration of hierarchical dense subgraph detection. Given an input graph, we want to detect a set of nested subgraphs, with an increasingly higher density across different hierarchies (from bottom to top).

Problem 1. Hierarchical Dense Subgraph Detection (HiDDen).

Given: *(1) the adjacency matrix* $\mathbf{A}$ *of the input graph* $\mathcal{G}$*, (2) the penalty parameter* p *for each missing edge in the graph, (3) the number of hierarchies* K*, and (4) the density ratio between two adjacent hierarchies* η*.*

Output: *the subgraph indicator vectors* $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^K$ *representing the* K *hierarchical dense subgraphs.*

In some applications, there might be one or more query nodes. In this setting, we are also interested in finding the hierarchical dense subgraphs containing all the query nodes, which is formally defined as below.

Problem 2. Query-Specific Hierarchical Dense Subgraph Detection (HiDDen-Query).

Given: *(1) the adjacency matrix* $\mathbf{A}$ *of the input graph* $\mathcal{G}$*, (2) the penalty parameter* p *for each missing edge in the graph, (3) the number of hierarchies* K*, (4) the density ratio between two adjacent hierarchies* η*, and (5) the query node set* $\mathcal{V}_s$*.*

Output: *a set of subgraph indicator vectors* $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^K$*, representing the* K *hierarchical dense subgraphs that contain all the nodes in the query set* $\mathcal{V}_s$*.*

In the both definitions, we require that $d^K \geq \eta d^{K-1} \geq \cdots \geq \eta^{K-1} d^1$ and $\mathcal{V}^K \subseteq \mathcal{V}^{K-1} \subseteq \cdots \subseteq \mathcal{V}^1$.

3 Hierarchical Dense Subgraph Detection

In this section, we present our solutions to hierarchical dense subgraph detection problem. We first formulate Problem 1 from optimization perspective, then propose the algorithms to solve it, followed by some analyses.

3.1 Optimization Formulation.

A - Density Measure. Given an input graph $\mathcal{G}$, a straightforward way to measure the density is by $\frac{2m_s}{n_s(n_s-1)}$, where m_s, n_s is the number of the edges and nodes in the subgraph. However, directly maximizing this density leads to some trivial solutions, e.g., a clique with only two nodes. Another possible way is to maximize $\mathbf{x}'\mathbf{A}\mathbf{x}$, i.e., to maximize the number of edges in the subgraph. Nevertheless, one trivial result is the graph $\mathcal{G}$ itself. In order to avoid such trivial solutions, we need an alternative formulation.

The key idea behind our proposed formulation is to simultaneously (1) maximize the number of edges, and (2) minimize the number of the missing edges in the subgraph. To be specific, we have the following formulation to detect the dense subgraph in one hierarchy

$$(3.1)\qquad \begin{aligned} \max_{\mathbf{x}} \quad & J(\mathbf{x}) = (1+p)\mathbf{x}'\mathbf{A}\mathbf{x} - p\mathbf{x}'(\mathbf{1}_{n\times n} - \mathbf{I})\mathbf{x} \\ \text{s.t} \quad & \mathbf{x} \in \{0,1\}^n. \end{aligned}$$

where $\mathbf{1}_{n\times n}$ and $\mathbf{I}$ are the matrix with all entries equal to 1 and the identity matrix with the same size of $\mathbf{A}$ respectively, and $\mathbf{x}$ is the subgraph indicator vector. Each existing edge in the resultant subgraph contributes a value of 1 to the objective function, while each missing edge is penalized by a value of p, where $p > 0$ is the penalty parameter. In this way, the above objective function maximizes the total number of the edges in the resultant dense subgraph while minimizing the total penalty of the missing edges. We remark that the objective function in Eq. (3.1) has a close relationship with the classic α-quasi-clique detection problem [24], as summarized in the following lemma.

Lemma 1. *Maximizing Eq.* (3.1) *is equivalent to maximizing the edge surplus density.*

Proof. By dividing Eq. (3.1) by $1+p$, the first term $\mathbf{x}'\mathbf{A}\mathbf{x}$ is the number of edges of the inferred dense subgraph $\mathcal{S}$. Since $\mathbf{x}'(\mathbf{1}_{n\times n} - \mathbf{I})\mathbf{x}$ is essentially the number of the edges in the clique of size n_s, the second term represents the number of edges in the $(\frac{p}{1+p})$-quasi-clique. The *edge surplus* density defined in [24] means the difference in the number of edges between the inferred dense subgraph and the corresponding quasi-clique. Therefore, maximizing Eq. (3.1) is equivalent to maximizing the *edge surplus* density.

B - Constraints and Relaxation. The binary constraint in Eq. (3.1) makes the corresponding optimization problem an integer programming problem, which is very hard to solve due to its combinatorial nature. To address this issue, we relax the constraint to $\mathbf{0} \leq \mathbf{x} \leq \mathbf{1}$. The relaxed constraint represents the possibility of each node belonging to the dense subgraph.

In order to detect dense subgraphs hierarchically, we basically want to maximize the objective function in Eq.(3.1) for each hierarchy, i.e., to maximize $\sum_{k=1}^{K} J(\mathbf{x}^k)$. As mentioned before, we have two additional constraints. First (*density variety*), for the k^{th}

hierarchy, we require that $d^k \geq \eta d^{k-1}$ where $\eta > 1$ is the density increase ratio to make sure that the densities in two adjacent hierarchies exhibit a significant difference. Here, we define the density of the subgraph in the k^{th} hierarchy as

$$d^k = \frac{(\mathbf{x}^k)'\mathbf{A}\mathbf{x}^k}{(\mathbf{x}^k)'(\mathbf{1}_{n\times n} - \mathbf{I})\mathbf{x}^k}$$

Second (*nested node sets*), we require that $\mathcal{V}^{k+1} \subseteq \mathcal{V}^k \subseteq \mathcal{V}^{k-1}$. This could be done by introducing an additional constraint $\mathbf{x}^{k+1} \leq \mathbf{x}^k \leq \mathbf{x}^{k-1}$.

C - Overall Formulation. Putting everything together, we obtain the following formulation:

$$\begin{aligned}
&\max_{\mathbf{x}^1,\cdots,\mathbf{x}^K} && \sum_{k=1}^{K}(\mathbf{x}^k)'\mathbf{M}\mathbf{x}^k \\
(3.2)\quad &\text{s.t} && d^{j+1} \geq \eta d^j \text{ (density variety)} \\
& && \mathbf{x}^{j+1} \leq \mathbf{x}^j \leq \mathbf{x}^{j-1} \text{ (nested node sets)} \\
& && \forall j = 1, 2, \cdots, K
\end{aligned}$$

where $\mathbf{x}^0 = \mathbf{1}$, $\mathbf{x}^{K+1} = \mathbf{0}$, $\mathbf{M} = (1+p)\mathbf{A} - p(\mathbf{1}_{n\times n} - \mathbf{I})$.

Notice that the first constraint (i.e., density variety) is a quadratic constraint, which makes the above optimization problem to be a non-convex quadratic constrained quadratic programming problem (QCQP). Since QCQP itself is typically very time-consuming to solve, we further relax the quadratic constraint to a regularization term in the objective function. Denote $C^{j-1} = \eta d^{j-1}$ and note that it is a constant w.r.t $\mathbf{x}^j$, so the regularization term can be written as $(\mathbf{x}^j)'\mathbf{A}\mathbf{x}^j - C^{j-1}(\mathbf{x}^j)'(\mathbf{1}_{n\times n} - \mathbf{I})\mathbf{x}^j$. After the relaxation, the hierarchical dense subgraph detection problem can be formulated as the following optimization problem:

$$\begin{aligned}
(3.3)\quad &\max_{\mathbf{x}^1,\cdots,\mathbf{x}^K} && \sum_{k=1}^{K}(\mathbf{x}^k)'\mathbf{M}\mathbf{x}^k + \beta\sum_{k=2}^{K}(\mathbf{x}^k)'\mathbf{Q}^k\mathbf{x}^k \\
&\text{s.t} && \mathbf{x}^{j+1} \leq \mathbf{x}^j \leq \mathbf{x}^{j-1}, \forall j = 1, 2, \cdots, K
\end{aligned}$$

where $\mathbf{x}^0 = \mathbf{1}$, $\mathbf{x}^{K+1} = \mathbf{0}$, and $\mathbf{Q}^k = \mathbf{A} - C^{k-1}(\mathbf{1}_{n\times n} - \mathbf{I})$ and the parameter β controls the importance of the second term. If we define $\mathbf{P}^k = (1+p+\beta)\mathbf{A} - (p + \beta C^{k-1})(\mathbf{1}_{n\times n} - \mathbf{I})$, Eq. 3.3 can be further re-written as $(\mathbf{x}^1)'\mathbf{M}\mathbf{x}^1 + \sum_{k=2}^{K}(\mathbf{x}^k)'\mathbf{P}^k\mathbf{x}^k$. The intuition is that, for the k^{th} hierarchy ($k \geq 2$), this equivalent formulation aims to maximize the number of edges while minimizing the total penalty of the missing edges with a *larger* penalty parameter compared with that for the $(k-1)^{\text{th}}$ hierarchy, in order to encourage density variety.

3.2 Optimization Algorithms.

Let us start with a baseline algorithm for solving the optimization problem in Eq. (3.3) (referred to as HiDDen-Basic) as follows. First, the optimization problem in Eq. (3.3) is essentially a (non-convex) quadratic programming problem, which can be solved by the conventional quadratic programming methods. Second, the objective function in Eq. (3.3) is essentially a nonconvex bounded quadratic function w.r.t each indicator vector $\mathbf{x}^k$. This property allows us to use the alternative optimization method, that is, in each iteration, we fix all other variables as constants except one variable. Third, the subgraph indicator vectors are nested, that is, the feasible solution of $\mathbf{x}^k$ should be in the high-dimensional space bounded by $\mathbf{x}^{k-1}$ and $\mathbf{x}^{k+1}$. In order to use the quadratic programming method, we relax the nested constraints to $\mathbf{0} \leq \mathbf{x}^k \leq \mathbf{x}^{k-1}$ and we can then solve $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^K$ in order. That is, we solve the $\mathbf{x}^1$ with the constraint $\mathbf{0} \leq \mathbf{x}^1 \leq \mathbf{1}$ by ignoring the constraints of other variables. After we obtain the indicator vector $\mathbf{x}^1$ of the 1st hierarchy, we solve $\mathbf{x}^2$ under the constraint $\mathbf{0} \leq \mathbf{x}^2 \leq \mathbf{x}^1$ by ignoring the constraints of other variables. In this way, we can solve the K dense subgraph detection subproblem hierarchy-by-hierarchy. To solve each nonconvex bounded quadratic programming subproblem, we use the *trust-region* method [8]. We omit the full details of this baseline algorithm due to the space limit.

A major limitation of HiDDen-Basic lies in computational efficiency. This is because the input matrices of the conventional quadratic programming methods (i.e., $\mathbf{M}$ and $\mathbf{P}^k$) are full matrices with all nonzero entries. Thus, these methods are very time and memory consuming, and impractical for large-scale graphs.

In order to develop a more efficient solution, we notice that because $\mathbf{x}^k \geq \mathbf{0}$, the term $(\mathbf{x}^k)'(\mathbf{1}_{n\times n} - \mathbf{I})\mathbf{x}^k$ can be viewed as an equivalent regularization term $\|\mathbf{x}^k\|_1^2 - \|\mathbf{x}^k\|_2^2$. In this way, the optimization problem in Eq. (3.3) can be re-written as follows.

$$\begin{aligned}
&\min_{\mathbf{x}^1,\cdots,\mathbf{x}^K} && f(\mathbf{x}^1, \cdots, \mathbf{x}^K) = \\
& && -(1+p)(\mathbf{x}^1)'\mathbf{A}\mathbf{x}^1 + p(\|\mathbf{x}^1\|_1^2 - \|\mathbf{x}^1\|_2^2) \\
(3.4)\quad & && -(1+p+\beta)\sum_{k=2}^{K}(\mathbf{x}^k)'\mathbf{A}\mathbf{x}^k \\
& && +\sum_{k=2}^{K}(p + \beta C^{k-1})(\|\mathbf{x}^k\|_1^2 - \|\mathbf{x}^k\|_2^2) \\
&\text{s.t} && \mathbf{x}^{j+1} \leq \mathbf{x}^j \leq \mathbf{x}^{j-1}, \forall j = 1, 2, \cdots, K
\end{aligned}$$

To solve the optimization method in Eq. (3.4), we resort to the *alternative projected gradient descent* method [16]. The gradient of $f(\mathbf{x}^1, \cdots, \mathbf{x}^K)$ w.r.t the variable $\mathbf{x}^1$ and $\mathbf{x}^k$ ($k \geq 2$) can be computed by

$$\bigtriangledown_{\mathbf{x}^1} f = -2(1+p)\mathbf{A}\mathbf{x}^1 + 2p\|\mathbf{x}^1\|_1\mathbf{1} - 2p\mathbf{x}^1$$

$$\bigtriangledown_{\mathbf{x}^k} f = -2(1+p+\beta)\mathbf{A}\mathbf{x}^k + 2(p + \beta C^{k-1})(\|\mathbf{x}^k\|_1\mathbf{1} - \mathbf{x}^k)$$

To update $\mathbf{x}^k$, we use $\mathbf{x}^k_{\text{new}} = P[\mathbf{x}^k - a \bigtriangledown_{\mathbf{x}^k} f]$ as the update rule, where the operator $P[\cdot]$ projects the vector back to its bounded feasible region. Recap that the well-known Armijo's rule line search is to compute a good step size a until the following condition is satisfied.

$$(3.5)\qquad f(\mathbf{x}^k_{\text{new}}) - f(\mathbf{x}^k) \leq \sigma(\bigtriangledown_{\mathbf{x}^k} f)'(\mathbf{x}^k_{\text{new}} - \mathbf{x}^k)$$

However, it is often very time consuming to search for a good step size a. Thus, we adopt the adjusted Armijo's rule line search [16] as follows. We assume the step size for updating $\mathbf{x}^k$ and $\mathbf{x}^k_{\text{new}}$ to be similar. Therefore, we use the step size for $\mathbf{x}^k$ as the initial guess of the step size for $\mathbf{x}^k_{\text{new}}$ and then either increase or decrease it to satisfy Eq. (3.5). The algorithm of updating one variable $\mathbf{x}^k$ is summarized in Algorithm 1. The key of Algorithm 1 is to search a good step size efficiently (Lines 6-10). That is, if the condition Eq. (3.5) is not satisfied, we reduce the step size by the factor b; otherwise, we increase the step size by $\frac{1}{\sqrt{b}}$.

Algorithm 1 Updating one variable (Update-One).

Input: (1) the current indicator vector $\mathbf{x}^k$ to be updated, (2) the initial step size a, (3) the ratio of decreasing the step size $0 < b < 1$, and (4) the parameter $0 < \sigma < 1$.
Output: the new subgraph indicator vector $\mathbf{x}^k_{\text{new}}$.

```
 1: Compute the gradient $\bigtriangledown_{\mathbf{x}^k} f$ of $\mathbf{x}^k$ by Eq. 3.5;
 2: Compute the objective function value of $\mathbf{x}^k$ by Eq. 3.4;
 3: while Amijo's condition is not satisfied do
 4:     $\mathbf{x}^k_{\text{new}} = P[\mathbf{x}^k - a \bigtriangledown_{\mathbf{x}^k} f]$;
 5:     Compute the new value $F_{\text{n}}$ of $\mathbf{x}^k_{\text{new}}$ by Eq. 3.4;
 6:     if $F_{\text{n}} - F > \sigma(\bigtriangledown_{\mathbf{x}^k} f)'(\mathbf{x}^k_{\text{new}} - \mathbf{x}^k)$ then
 7:         $a \leftarrow ba$;
 8:     else
 9:         $a \leftarrow \frac{a}{\sqrt{b}}$;
10:     end if
11: end while
12: Return the updated subgraph indicator vector $\mathbf{x}^k_{\text{new}}$.
```

To solve all variables $\mathbf{x}^1, \cdots, \mathbf{x}^K$, we run the Algorithm 1 for each variable in an alternative way in each iteration. At the beginning, we initialize each node to have a possibility of 0.5 to be in the first hierarchy, i.e., $\mathbf{x}^1 = 0.5 \times \mathbf{1}$. In order to avoid trivial solutions, we also set $\mathbf{x}^k = 0.01 \times \mathbf{1}$, for $2 \le k \le K$. We adopt the following stopping criterion [16] for the alternative projected gradient descent method

$$\|[\bigtriangledown^P_{\mathbf{x}^1} f, \cdots, \bigtriangledown^P_{\mathbf{x}^K} f]\|_2 \le \epsilon \|[\bigtriangledown^P_{\mathbf{x}^1_{\text{init}}} f, \cdots, \bigtriangledown^P_{\mathbf{x}^K_{\text{init}}} f]\|_2 \tag{3.6}$$

where the operator $\bigtriangledown^P_{\mathbf{x}^k} f$ is defined as

$$(\bigtriangledown^P_{\mathbf{x}^k} f)_i = \begin{cases} (\bigtriangledown_{\mathbf{x}^k} f)_i & \text{if } \mathbf{x}^{k+1}_i < \mathbf{x}^k_i < \mathbf{x}^{k-1}_i \\ \min(0, (\bigtriangledown_{\mathbf{x}^k} f)_i) & \text{if } \mathbf{x}^k_i = \mathbf{x}^{k+1}_i \\ \max(0, (\bigtriangledown_{\mathbf{x}^k} f)_i) & \text{if } \mathbf{x}^k_i = \mathbf{x}^{k-1}_i \end{cases}$$

The complete algorithm to solve the optimization problem Eq. (3.4) is summarized in Algorithm 2.

Algorithm 2 Hierarchical Dense Subgraph Detection (HiDDen-OPT).

Input: (1) the adjacency matrix $\mathbf{A}$ of the given graph, (2) the penalty value of each missing edge p, (3) the number of hierarchies K, (4) the density ratio η between two adjacent hierarchies, and (5) the maximum number of iterations $t_{\max}$.
Output: the subgraph indicator vectors $\mathbf{x}^1, \cdots, \mathbf{x}^K$ for each hierarchy.

```
1: Initialize $\mathbf{x}^1, \cdots, \mathbf{x}^K$, and set $t = 1$;
2: Compute the initial norm $\|[\bigtriangledown^P_{\mathbf{x}^1_{\text{init}}} f, \cdots, \bigtriangledown^P_{\mathbf{x}^K_{\text{init}}} f]\|_2$;
3: while Eq. 3.6 is not satisfied and $t \le t_{\max}$ do
4:     for $k = 1 \rightarrow K$ do
5:         $\mathbf{x}^k \leftarrow$ Update-One($\mathbf{x}^k$);
6:     end for
7:     $t \leftarrow t + 1$;
8: end while
9: Return the subgraph indicator vectors $\mathbf{x}^1, \cdots, \mathbf{x}^K$.
```

3.3 Proof and Analysis.

In this subsection, we analyze the *convexity*, the *convergence* and the *complexity* of HiDDen-OPT. We start with Theorem 1, which says our problem of Eq. (3.4) is nonconvex.

Theorem 1. *The optimization problem in Eq.* (3.4) *is a nonconvex optimization.*

Proof. We prove this by showing the Hessian matrix of each subproblem of $\mathbf{x}^k$ is not a positive semi-definite matrix. We have that the Hessian matrix of the variable $\mathbf{x}^k$ is

$$\frac{\partial f^2}{\partial^2 \mathbf{x}^1} = -2(1+p)\mathbf{A} + 2p(\mathbf{1}_{n\times n} - \mathbf{I})$$
$$\frac{\partial f^2}{\partial^2 \mathbf{x}^k} = -2(1+p+\beta)\mathbf{A} + 2(p + \beta C^{k-1})(\mathbf{1}_{n\times n} - \mathbf{I})$$

By the Weyl's inequality theorem [6], since the matrix $-2(1+p)\mathbf{A}$ has negative eigenvalues and so does the matrix $2p(\mathbf{1}_{n\times n} - \mathbf{I})$, the Hessian matrix w.r.t $\mathbf{x}^1$ has the negative eigenvalues. Similarly, the Hessian matrix w.r.t $\mathbf{x}^k$, for $2 \le k \le K$, also has the negative eigenvalues. In this way, the subproblem of each $\mathbf{x}^k$ is a nonconvex optimization problem, so the whole problem of Eq. (3.4) is a nonconvex optimization problem.

Lemma 2. HiDDen-OPT *algorithm converges to a stationary point.*

Proof. Omitted for space.

Lemma 3. *The time complexity of* HiDDen-OPT *algorithm is $O(mKt_{line}t_{max})$ where t_{line} is the number of iterations for line search.*

Proof. Omitted for space.

4 Generalization

In this section, we generalize the proposed HiDDen algorithm in two scenarios: (1) hierarchical dense subgraph detection on bipartite graphs, and (2) query-specific hierarchical dense subgraph detection. Due to the limited space, we highlight the key points of these two variants and omit the full description of the corresponding algorithms.

4.1 Hierarchical Dense Subgraph Detection on Bipartite Graphs. Given a bipartite graph $\mathcal{G} = (\mathbf{B}_{n_1 \times n_2}, \mathcal{V}, \mathcal{E})$, where $\mathbf{B}$ has n_1 rows and n_2 columns, we want to extract K hierarchical dense subgraphs. Here, the nuance is that the number of edges in a bipartite clique is equal to $|\mathcal{V}_r| \times |\mathcal{V}_c|$ (e.g., $n_1 \times n_2$ for the whole bipartite graph). In addition, we need two subgraph indicator vectors to define a bipartite subgraph, i.e., the vector $\mathbf{x}$ to select row nodes and the vector $\mathbf{y}$ to select column nodes. In this way, the corresponding optimization problem becomes as follows.

$$\begin{aligned}
\min \quad & f(\mathbf{x}^1, \cdots, \mathbf{x}^K, \mathbf{y}^1, \cdots, \mathbf{y}^K) = \\
& -(1+p)(\mathbf{x}^1)'\mathbf{A}\mathbf{y}^1 - (1+p+\beta)\sum_{k=2}^{K}(\mathbf{x}^k)'\mathbf{A}\mathbf{y}^k \\
& + p\|\mathbf{x}^1\|_1\|\mathbf{y}^1\|_1 + \sum_{k=2}^{K}(p+\beta C^{k-1})\|\mathbf{x}^k\|_1\|\mathbf{y}^k\|_1 \\
\text{s.t} \quad & \mathbf{x}^{j+1} \le \mathbf{x}^j \le \mathbf{x}^{j-1},\ \mathbf{y}^{j+1} \le \mathbf{y}^j \le \mathbf{y}^{j-1} \\
& \forall j = 1, 2, \cdots, K
\end{aligned}$$

where $\mathbf{x}^0 = \mathbf{1}_{n_1}$, $\mathbf{x}^{K+1} = \mathbf{0}_{n_1}$, $\mathbf{y}^0 = \mathbf{1}_{n_2}$, $\mathbf{y}^{K+1} = \mathbf{0}_{n_2}$.

To solve the above optimization problem, we use the similar alternative projected gradient descent method as in Algorithm 1 and Algorithm 2. We refer to this algorithm as HiDDen-BP. The main difference is that in each iteration we need to alternate between $\mathbf{x}^k$ and $\mathbf{y}^k$ to update the indicator vectors in the k^{th} hierarchy. The stopping criterion Eq. (3.6) is changed to

$$\begin{aligned}
&\|[\nabla^P_{\mathbf{x}^1} f, \cdots, \nabla^P_{\mathbf{x}^K} f, \nabla^P_{\mathbf{y}^1} f, \cdots, \nabla^P_{\mathbf{y}^K} f]\|_2 \le \\
&\epsilon \|[\nabla^P_{\mathbf{x}^1_{\text{init}}} f, \cdots, \nabla^P_{\mathbf{x}^K_{\text{init}}} f, \nabla^P_{\mathbf{y}^1_{\text{init}}} f, \cdots, \nabla^P_{\mathbf{y}^K_{\text{init}}} f]\|_2
\end{aligned}$$

4.2 Query-Specific Hierarchical Dense Subgraphs. In Problem 2, we want to find the hierarchical dense subgraphs containing a set of pre-specified query nodes $\mathcal{V}_s$. Mathematically, we can set all the entries corresponding to the query nodes of the indicator vectors $\mathbf{x}^1, \cdots, \mathbf{x}^K$ to be 1s. However, with this treatment, the optimization problem becomes to a *mixed integer programming* problem which is typically solved by computationally exhaustive search. To resolve this issue, we relax these entries to be a number that is smaller than but close to 1 (say 0.9 in this paper). Thus, the corresponding optimization problem becomes the same as Eq. (3.4), except the lower bound

$$\mathbf{x}_i^{K+1} = \begin{cases} 0.9 & \text{if } i \in \mathcal{V}_s \\ 0 & \text{otherwise} \end{cases}$$

Thus, we can use the same algorithm as Algorithm 1 and Algorithm 2 to solve Problem 2 with new constraints. We refer to this algorithm as HiDDen-Query.

5 Experimental Results

In this section, we present the experimental results, to evaluate:

1. *Effectiveness*: How effective are our proposed algorithms?
2. *Efficiency*: How fast and scalable are our proposed algorithms?

5.1 Experimental Setup

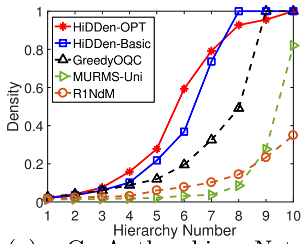
Datasets. We evaluate our proposed algorithms on three real-world datasets.

- *Co-Authorship Network (CO).* We construct our co-author network based on a snapshot of AMiner citation dataset, which collects all papers up to year 2011, and consists of 1,632,442 papers and 1,036,999 authors [21]. We build our graph upon all papers in 5 research areas, including data mining (DM), machine learning (ML), database (DB), information retrieval (IR), and bioinformatics (BIO) [18]. The constructed graph consists of 38,624 authors and 200,332 co-authorship relationships.
- *Financial Network (FN).* The dataset contains information about 25,813,372 bank accounts. We build a bipartite graph with two types of nodes. One type is account nodes, each of which represents a unique account number. The other type is the personal identity information (PII), e.g., mailing address, email address. Each edge between an account node and a PII node represents the account has the corresponding personal identity information. From the constructed graph, we use a subgraph with 29,851 account nodes, 61,159 PII nodes and 98,577 edges for evaluation.
- *Autonomous System Network (AS).* This graph has 6,474 nodes and 25,142 edges. Each node represent an autonomous system and each edge means there is a traffic flow exchanging (or communication) between two autonomous systems [24].

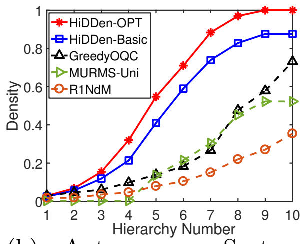
Comparison Methods. For the proposed HiDDen algorithms, we test our HiDDen-Basic and HiDDen-OPT on the unipartite graphs and compare the results with the following existing algorithms, including (1) *GreedyOQC* [24], (2) *MURMS-Uni* [9], and (3) *R1NdM* [5]. To evaluate our HiDDen-BP algorithm on bipartite graphs, we compare it with (1) *GreedyOQC*, (2) *MURMS-Bi* [9], and (3) *R1NBi* [11].

5.2 Effectiveness Analysis

Quantitative Results on Unipartite Graphs. We first evaluate how density changes across different hierarchies on unipartite graphs. Note that all baseline methods are solely for *flat* dense subgraph detection (i.e., one hierarchy). To extract multiple hierarchies of dense subgraphs, we (1) run the corresponding algorithm on background graph and treat the extracted dense subgraph as new background graph for next hierarchy, (2) adjust input parameters to avoid trivial solutions (e.g., to return an identical subgraph as the input



(a) Co-Authorship Network.



(b) Autonomous System Network.

Figure 2: Subgraph densities vs. the hierarchies.

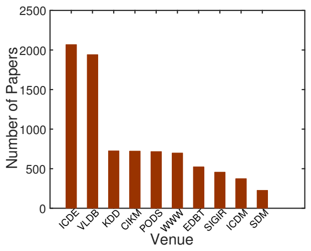


Figure 3: Difference of the number of papers in different venues between the 1st and the 5th hierarchy.

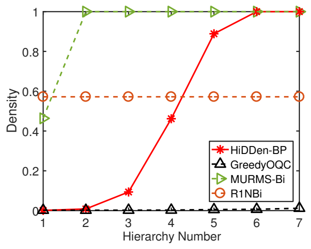


Figure 4: Densities vs. hierarchies number on financial network.

subgraph). We measure the density defined by $\frac{2m_s}{n_s(n_s-1)}$. The results are summarized in Figure 2. As we can see, our algorithms (two solid lines) obtain denser subgraphs than three existing methods in almost every hierarchy in both graphs. Specifically, our method HiDDen-OPT can find subgraphs with an up to 40% higher density than existing methods do in almost each hierarchy. The densities of the subgraphs detected by our methods exhibit a desirable variety across different hierarchies, i.e., starting with a relative low density (yet still dense enough) in the first hierarchy, the densities of the detected subgraphs consistently increase up to 1 (i.e., corresponding to a clique) w.r.t the increasing hierarchies.

Case Studies on Unipartite Graphs. We conduct a case study on the co-authorship network to find dense subgraphs in 10 hierarchies, which reveal some interesting patterns. For example, by comparing the difference between the 1st and the 5th hierarchy, those unique authors who appear in the 1st hierarchy but not in the 5th hierarchy mainly consist of researchers who (1) publish their papers in database conferences (e.g., ICDE and VLDB) as shown in Figure 3, or (2) obtain their Ph.D degree around 2011. And many of the researchers in the 5th hierarchy are in their mid-careers (either as assistant/associate professors in academy or in the leading industrial research labs, e.g., IBM, Microsoft). On the other hand, the authors in the 10th hierarchy (the highest hierarchy) are all flagship researchers in both databases and data mining fields with close collaborations with each other, e.g., *Jiawei Han*, *Philip S. Yu*, *Jian Pei*, etc.

Quantitative Results on Bipartite Graphs. Here, we evaluate our HiDDen-BP algorithm on bipartite graphs. Notice that *GreedyOQC* algorithm is designed for unipartite graphs. Therefore, we first transform the original bipartite graph into unipartite graph in order to run *GreedyOQC*. The result is shown in Figure 4. As we can see, our method can extract hierarchical dense bipartite subgraphs with a desirable density variety, i.e., starting from a relative dense subgraph up to a biclique. Meanwhile, neither *GreedyOQC* or *R1NBi* is able to find hierarchies with noticeable density increases (i.e., the corresponding curves are almost flat across different hierarchies). Although *MURMS-Bi* can also extract a biclique, it only differentiates dense subgraphs in two hierarchies (i.e., the first hierarchy vs. the remaining).

Case Studies on Bipartite Graphs. We conduct a case study on the Financial Network (FN) dataset. The dense subgraphs in different hierarchies by the proposed HiDDen-BP algorithm *collectively* reveal some interesting patterns in relation to potential *synthetic identity fraud* [26]. A typical approach that the synthetic identity fraudsters utilize is to open bank accounts using the stolen PII of other people. This means that some suspicious accounts may share the same PII with other accounts. Consequently, in the bipartite financial network, suspicious account nodes and PII nodes are likely to form dense subgraphs with many legitimate nodes. Figure 5 presents a miniature of our results on the financial network. In order to protect user privacy, we have anonymized the actual PII information. As we can see, the 7th hierarchy contains a biclique with only *Address1, Phone1 and HolderName1* as PII nodes and all the account nodes associated with them. However, it is very risky to flag it as synthetic identity solely based on this biclique, because a legitimate user could open multiple accounts using exactly the same Address/Phone/HolderName. By further examining the detected dense subgraph in the 1st hierarchy, we find that these accounts also use various email addresses and other phone numbers; yet there is a suspicious similarity between different email address names (e.g., one address name seems to be a robotic editing of another). In this case, flat dense subgraph detection would either miss the potential fraud, or lead to a false alarm (e.g., using the 7th hierarchy only).

Case Studies on Query-Specific Hierarchical Dense Subgraph Detection. Finally, to demonstrate the effectiveness of our HiDDen-Query algorithm, we query the hierarchical dense subgraphs with *Geoffrey E. Hinton* as the query in the co-authorship network. We visualize the results in three hierarchies in Figure 6. The subgraph in the first hierarchy, consists of the researchers who share the same research interest as the

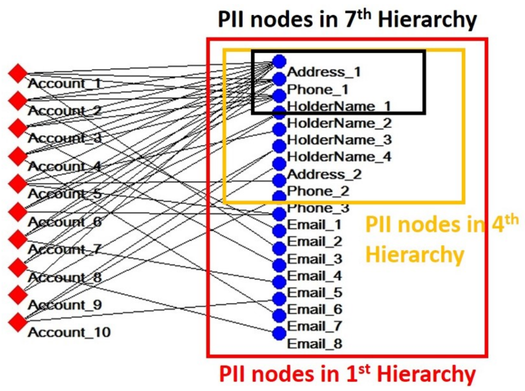


Figure 5: An example of suspicious synthetic ID fraud revealed by our algorithms.

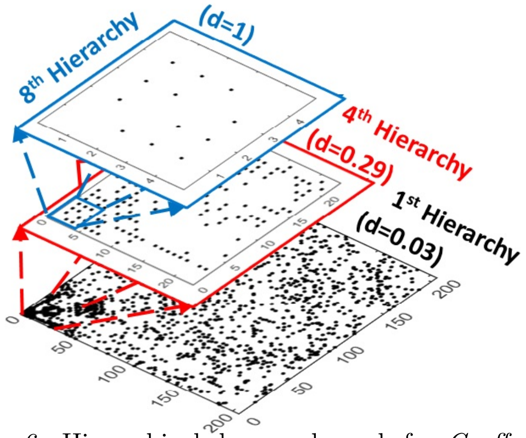


Figure 6: Hierarchical dense subgraph for *Geoffrey E. Hinton*.

query *Geoffrey E. Hinton* (i.e., machine learning and deep learning), including some of his former Ph.D students. As we zoom into a denser subgraph, in the 4th hierarchy, there are more active researchers, mostly his current and prior colleagues. In the last hierarchy, we extract a full clique, with all prominent researchers in the field of deep learning and machine learning, all with very close collaboration with *Geoffrey E. Hinton* in machine learning and deep learning, including *Zoubin Ghahramani*, *Lawrence K. Saul*, etc.

5.3 Efficiency Analysis

Quality-Speed Trade-off. We first evaluate how different methods balance between the running time and the quality of all hierarchies of subgraphs on the AS network. Ideally, we would like the densities of the detected subgraphs across different hierarchy to exhibit a high variety (i.e., high variance of the densities), in the meanwhile the density of each detected subgraph should be reasonably high (i.e., high average density). As we can see from Figure 7, the running time of our HiDDen-OPT algorithm is close to the existing methods, but our method obtains a better average density with a higher variance. Between the two proposed algorithms

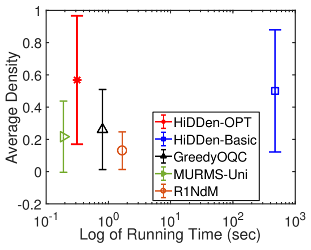


Figure 7: Balance between quality and speed.

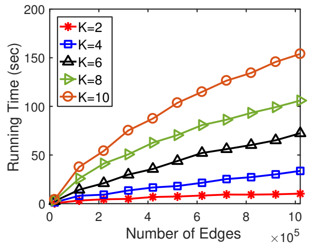


Figure 8: Scalability of HiDDen-OPT.

(HiDDen-Basic vs. HiDDen-OPT), even though the quality of HiDDen-Basic is also good, it takes much more time. Overall, HiDDen-OPT achieves the best tradeoff between the running time and the quality.

Scalability. The scalability of HiDDen-OPT is summarized in Figure 8. As we can see, the running time is *linear* w.r.t the number of edges, which is consistent with the complexity analysis.

6 Related Work

Dense subgraph detection is a fundamental problem in both algorithmic graph theory and graph mining. It lies in the heart of numerous domains, ranging from text mining [12, 22], bioinformatics [25, 1], fraud detection [13, 2], to community detection [14, 20].

Finding dense subgraphs has been studied for many years. Different types of dense subgraphs can be extracted with different density measures. For example, the most straightforward density measure is *edge density* defined by $\frac{2m_s}{n_s(n_s-1)}$. However, maximizing this type of density could be trivial because a small subgraph with only two nodes and an edge between them has the highest *edge density*. *densest subgraph* problem is to find subgraphs with a large *average degree* defined by $\frac{m_s}{n_s}$. Asashiro et al. propose a greedy algorithm to extract the *densest subgraphs* [4] and Charikar shows that this algorithm can run in linear time with approximation guarantees [7]. Besides, Tsourakakis et al. define a novel density measurement as the *edge surplus* between the subgraph and the corresponding α*-quasi-clique*. By maximizing the *edge surplus* density, they show that the extracted dense subgraphs can be denser than the *densest subgraphs* in terms of *edge density* [24].

In addition to these traditional dense subgraph detection methods, there are some other related problems. One of them is the *maximum/maximal clique* problem. Motzkin and Straus proved that the *maximal clique* problem is equivalent to maximizing the function $\mathbf{x}'\mathbf{A}\mathbf{x}$ with a simplex constraint [17]. Besides, [11, 5] propose a rank-one nonnegative matrix factorization method to extract maximum biclique/clique. Other related problems with size restrictions have also been studied, such as *k-clique* [23], *k-core* [19] and *k-plex* problem [15].

7 Conclusions

In this paper, we study the hierarchical dense subgraph detection problem. First, we formulate our problem from optimization perspective, i.e., to maximize the number of edges and minimize the penalties on the missing edges, with carefully designed constraints. Second, we propose an alternative projected gradient descent based algorithm to solve the hierarchical dense subgraph detection problem, which converges to the stationary point with a linear time complexity. In addition, we also present two variants to handle bipartite graphs and query-specific hierarchical dense subgraph detection. Finally, we conduct extensive evaluations on three real graphs, which demonstrate the effectiveness and the efficiency of the proposed algorithm (HiDDen). Future work includes generalizing the proposed HiDDen algorithm to tensors as well as time-varying graphs.

8 Acknowledgements

This work is supported by NSF grants IIP-1430144, IIS-1552654 and IIS-1651203, ONR grant N00014-15-1-2821, DTRA grant HDTRA1-16-0017, ARO grant W911NF-16-1-0168, NIH grant R01LM011986, an IBM Faculty Award and a Baidu gift. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

References

[1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek, *Cfinder: locating cliques and overlapping modules in biological networks*, Bioinformatics, 22 (2006), pp. 1021–1023.

[2] L. Akoglu, H. Tong, and D. Koutra, *Graph based anomaly detection and description: a survey*, Data Mining and Knowledge Discovery, 29 (2015).

[3] Y. Asahiro, R. Hassin, and K. Iwama, *Complexity of finding dense subgraphs*, Discrete Applied Mathematics, 121 (2002), pp. 15–26.

[4] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama, *Greedily finding a dense subgraph*, Journal of Algorithms, 34 (2000), pp. 203–221.

[5] M. T. Belachew and N. Gillis, *Solving the maximum clique problem with symmetric rank-one nonnegative matrix approximation*, arXiv preprint arXiv:1505.07077, (2015).

[6] R. Bhatia, *Linear algebra to quantum cohomology: the story of alfred horn's inequalities*, The American Mathematical Monthly, 108 (2001), pp. 289–318.

[7] M. Charikar, *Greedy approximation algorithms for finding dense components in a graph*, in International Workshop on Approximation Algorithms for Combinatorial Optimization, Springer, 2000, pp. 84–95.

[8] T. F. Coleman and Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM Journal on optimization, 6 (1996), pp. 418–445.

[9] C. Ding, T. Li, and M. I. Jordan, *Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding*, in ICDM, 2008, pp. 183–192.

[10] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis, *Evaluating cooperation in communities with the k-core structure*, in Advances in Social Networks Analysis and Mining (ASONAM), 2011, pp. 87–93.

[11] N. Gillis and F. Glineur, *A continuous characterization of the maximum-edge biclique problem*, Journal of Global Optimization, 58 (2014), pp. 439–464.

[12] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, *Robust disambiguation of named entities in text*, in EMNLP, Association for Computational Linguistics, 2011.

[13] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, *Fraudar: bounding graph fraud in the face of camouflage*, in KDD, ACM, 2016, pp. 895–904.

[14] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, *Trawling the web for emerging cyber-communities*, Computer networks, 31 (1999).

[15] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal, *A survey of algorithms for dense subgraph discovery*, in Managing and Mining Graph Data, pp. 303–336.

[16] C.-J. Lin, *Projected gradient methods for nonnegative matrix factorization*, Neural computation, (2007).

[17] T. S. Motzkin and E. G. Straus, *Maxima for graphs and a new proof of a theorem of turán*, Canad. J. Math, 17 (1965), pp. 533–540.

[18] J. Ni, H. Tong, W. Fan, and X. Zhang, *Inside the atoms: ranking on a network of networks*, in KDD, ACM, 2014, pp. 1356–1365.

[19] K. Shin, T. Eliassi-Rad, and C. Faloutsos, *Corescope: Graph mining using k-core analysis - patterns, anomalies and algorithms*, in ICDM, 2016.

[20] M. Sozio and A. Gionis, *The community-search problem and how to plan a successful cocktail party*, in KDD, ACM, 2010, pp. 939–948.

[21] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, *Arnetminer: extraction and mining of academic social networks*, in KDD, ACM, 2008, pp. 990–998.

[22] A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, *Gowvis: a web application for graph-of-words-based text visualization and summarization*, ACL 2016, p. 151.

[23] C. Tsourakakis, *The k-clique densest subgraph problem*, in WWW, ACM, 2015, pp. 1122–1132.

[24] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli, *Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees*, in KDD, ACM, 2013, pp. 104–112.

[25] M. Wu, X. Li, C.-K. Kwoh, and S.-K. Ng, *A core-attachment based method to detect protein complexes in ppi networks*, BMC bioinformatics, 10 (2009), p. 1.

[26] D. Zhou, K. Wang, N. Cao, and J. He, *Rare category detection on time-evolving graphs*, in ICDM, 2015, pp. 1135–1140.