Unveiling Densest Multilayer Subgraphs via Greedy Peeling

Dandan Liu
Harbin Institute of Technology
Harbin, Heilongjiang, China
ddliu@hit.edu.cn

Zhaonian Zou
Harbin Institute of Technology
Harbin, Heilongjiang, China
znzou@hit.edu.cn

Run-An Wang
Harbin Institute of Technology
Harbin, Heilongjiang, China
wangrunan@stu.hit.edu.cn

ABSTRACT

Densest subgraphs in multilayer (ML) graphs unveil intricate relationships that are hidden from the simple graph representations, offering profound insights and applications across diverse domains. In this paper, we present a layer-oriented view of the existing density measures for ML graphs and highlight the problems in identifying the densest subgraphs under the layer-oriented design, including inefficiency, poor approximation guarantees, and the lack of a unified algorithmic framework. In light of this, we introduce a new family of vertex-oriented density measures called generalized density. Controlled by two parameters p and q, the generalized density provides the flexibility to shift focus on higher or lower vertex degrees and the degrees across layers. We investigate the problem of finding the densest ML subgraphs based on this density framework and show that half of the problem space can be addressed using a unified greedy peeling paradigm. We delve into four settings of q and p, carefully design the vertex selection strategies for the peeling process, and provide non-trivial approximation guarantees and complexity results. We also present a near-linear-time fast implementation to expedite the greedy peeling algorithm for certain cases. Experiments on 10 real-world ML graphs show that our generalized density and greedy peeling algorithms effectively uncover various types of dense subgraphs in large-scale ML graphs.

PVLDB Reference Format:
Dandan Liu, Zhaonian Zou, and Run-An Wang. Unveiling Densest Multilayer Subgraphs via Greedy Peeling. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/SSSuperDan/MGDS.

1 INTRODUCTION

Multilayer (ML) graph models are effective tools for modeling intricate real-world relationships established through complex channels, such as interactions between individuals across different social media platforms [17], transportation connections between hubs using different modes [2], and relationships among biomolecular entities derived from biological experiments [26, 30]. An ML graph is represented as a set of layered graphs, with each graph (also called layer) naturally describing a specific kind of relationship. An example of an ML graph is shown in Figure 1.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

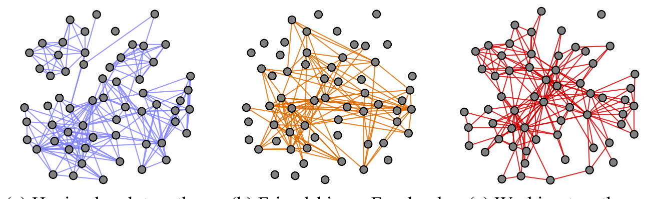


Figure 1: An example of an ML graph extracted from the AUCS dataset [17] with three layers representing different relationships between employees in a university.

Identifying dense structures within large graphs is a key graph primitive that has been extensively studied for decades. It finds applications in diverse domains [3, 14, 18, 21, 23, 24, 29, 30, 32, 36, 37, 45, 47]. While dense subgraph discovery in simple graphs has made significant progress and is close to being fully resolved [36], its adaptation to ML graph settings is still in its infancy.

Finding dense subgraphs in ML graphs holds the potential to reveal more reliable and meaningful patterns unseen from single-layer representations [30, 39, 41, 43]. For example, in biological research, researchers organize gene co-expression networks derived from multiple microarray datasets into ML graphs and identify functionally homogeneous clusters by extracting vertices that are densely connected on multiple layers. This approach has been demonstrated to effectively reduce spurious edges compared to deriving clusters from a single microarray dataset [30].

The task of identifying dense subgraphs is commonly approached as a combinatorial optimization problem of finding an induced subgraph that maximizes a density measure. In the context of ML graphs, density measures are usually formulated in a layer-oriented manner: the density of an ML graph is defined as the aggregation of the single-layer density of its individual layer. Different density measures [31, 50] employ unique single-layer density measures such as the average degree density or the minimum degree, coupled with specific aggregation operations like taking the average or minimum. Recent efforts have recognized the impact of noise or insignificant layers and proposed measures of multilayer density [22] and p-mean multilayer density [6], which allow for selecting a subset of layers that maximizes the density of the resulting subgraph.

However, designing algorithms to optimize these layer-oriented density measures is challenging. Existing solutions often face scalability issues [11, 27, 31] or offer poor quality guarantees [11]. More importantly, these layer-oriented densities are typically studied in isolation, lacking a unified, scalable, and effective algorithmic framework that can provide insight into solutions for maximizing layer-oriented densities adopting other single-layer density measures or aggregation operations for various application scenarios.

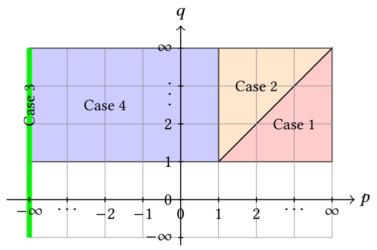


Figure 2: An partitioning of the problem space in terms of the setting of the parameters p and q of the proposed vertex-oriented density measure.

Table 1: The approximation ratios and the time complexity of the algorithms proposed for handling different cases, where V, E and L are the sets of vertices, edges and layers in the input ML graph, respectively, and $\epsilon \in [0, 1]$.

Case	Condition	Approximation Ratio	Time Complexity
1	$p \geq q \geq 1$	$(1+p)^{1/p}$	$O(\sum_{v \in V} d_v^2 \lvert L\rvert \log \lvert V\rvert)$
		$\left((1+2\epsilon)(1+p\lvert L\rvert^{1-\frac{1}{q}})\right)^{1/p}$	$O\left(\frac{\lvert E\rvert \log \lvert V\rvert \log \lvert L\rvert \lvert V\rvert}{\log(1+\frac{\epsilon}{p-1})}\right)$
2	$q \geq p \geq 1$	$\left(1+p\lvert L\rvert^{1-\frac{1}{q}}\right)^{1/p}$	$O(\sum_{v \in V} d_v^2 \lvert L\rvert \log \lvert V\rvert)$
		$\left((1+2\epsilon)(1+p\lvert L\rvert^{1-\frac{1}{q}})\right)^{1/p}$	$O\left(\frac{\lvert E\rvert \log \lvert V\rvert \log \lvert L\rvert \lvert V\rvert}{\log(1+\frac{\epsilon}{p-1})}\right)$
3	$p = -\infty$	1	$O(\lvert L\rvert \lvert E\rvert + \lvert E\rvert \log \lvert V\rvert)$
4	$-\infty < p \leq 1 \leq q$	$\left(1+\lvert L\rvert^{1-\frac{1}{q}}\right)$	$O(\lvert L\rvert \lvert E\rvert + \lvert E\rvert \log \lvert V\rvert)$

In light of these challenges, we introduce a new family of density measures for ML graphs, which adopts an inherently different vertex-oriented approach. The key idea is to compute a representative aggregated degree for each vertex that captures the overall degree information and reflects the distribution of degrees across layers. These aggregated degrees are then incorporated into single-layer density measures.

Considering the power of generalized means which allow for flexible focus on larger or smaller elements, which has been successfully adopted in the p-density framework in simple graphs [53], we present the following vertex-oriented generalized density for ML graphs. Specifically, we define a notion of q-mean degree, calculated as the q-mean of the degrees of a vertex across all layers. The generalized density, also called (q, p)-density, is then defined as the ratio of the p-mean of the q-mean degrees of vertices to the total number of vertices. The (q, p)-density framework allows for adjusting the emphasis on higher or lower cross-layer degrees of vertices and the q-mean degrees of vertices through parameters q and p. This flexibility paves the way for characterizing a wide range of dense patterns. When $q = p$, the (q, p)-density also shows equivalence to certain layer-oriented density measures [50].

This paper studies the *generalized densest ML subgraph problem*, which finds the densest subgraphs from ML graphs that maximize the (q, p)-density. We investigate the problems under four settings of q and p, as outlined in Figure 1, and address them with a unified greedy peeling framework [4, 8, 10, 12, 32, 51]. For each setting, we customize the peeling algorithm with distinct vertex selection strategies. An additional fast implementation is also proposed for Case 1 and Case 2 to accelerate the peeling process. Table 1 summarizes the approximation guarantees and time complexities of the peeling algorithms proposed for handling different cases.

We apply our (q, p)-density framework and algorithms on 10 real-world ML graphs and obtain the following key results: (1) The (q, p)-density and the proposed algorithms can effectively uncover dense subgraphs with diverse characteristics in ML graphs. The q-mean degree can mitigate the influence of noisy layers, achieving a similar effect to selecting significant layers in layer-oriented density measures. (2) Our greedy peeling algorithms are fast and can gracefully scale to large graphs. They show greater efficiency improvements compared to layer-oriented densest subgraph detection algorithms as the number of layers increases.

The main contributions of this work are summarized as follows: **(1)** We formalize existing density measures for ML graphs into a layer-oriented density framework and highlight limitations in the densest ML subgraph detection algorithms based on this framework. **(2)** We introduce a novel vertex-oriented density measure for ML graphs and thoroughly discuss its features and relation to existing density measures. **(3)** We customize different versions of the greedy peeling framework to address the generalized densest ML subgraph problems for four settings of q and p. The approximation guarantees and time complexity results are summarized in Table 1. **(4)** We evaluate our density measure and algorithms on 10 real-world ML graphs. They showcase the ability to uncover various types of dense subgraphs in large-scale ML graphs.

2 MOTIVATION

In this section, we discuss the motivation of this paper.

2.1 Preliminaries

Multilayer Graphs. A *multilayer graph* (abbreviated as *ML graph*) is a triple $\mathcal{G} = (V, E, L)$, where V is the set of vertices, $L = \{1, 2, \ldots, |L|\}$ is the set of layer numbers, and $E \subseteq V \times V \times L$ is the set of edges. An edge $(u, v, i) \in E$ indicates that vertices u and v are adjacent on layer i. All vertices and edges on layers i constitute the graph on layer i, denoted as $G_i = (V, E_i)$, where $E_i = \{(u, v, i) | u, v \in V\}$. For any vertex $v \in V$, $N_v(i)$ represents the set of neighbors of v in G_i, and $d_v(i) = |N_v(i)|$ represents the degree of v in G_i. When the layer is not explicitly specified, we use N_v to denote the set of all neighbors of v, i.e., $N_v = \bigcup_{i \in L} N_v(i)$, and let $d_v = |N_v|$.

Given a vertex subset $S \subseteq V$, the subgraph of G_i induced by S is denoted as $G_i[S] = (S, E_i[S])$, where $E_i[S]$ represents the set of edges in E_i with both endpoints in S. Similarly, the subgraph of the ML graph $\mathcal{G}$ induced by S is denoted as $\mathcal{G}[S] = (S, E[S], L)$, where $E[S]$ represents the set of edges in E with both endpoints in S. The set of neighbors and the degree of any vertex $v \in S$ in $G_i[S]$ are denoted as $N_v(S, i)$ and $d_v(S, i)$, respectively. Moreover, we let $N_v(S) = \bigcup_{i \in L} N_v(S, i)$ and $d_v(S) = |N_v(S)|$.

Generalized Means. The *generalized mean* of a vector of positive real numbers $\mathbf{x} = [x_1, x_2, \cdots, x_n] \in \mathbb{R}_+^n$ with exponent p is

$$M_p(\mathbf{x}) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^p\right)^{1/p},$$

where $p \in \mathbb{R} - \{0\}$. When $p \in \{-\infty, 0, \infty\}$, the generalized mean can be defined by taking limits, and we have $M_{-\infty}(\mathbf{x}) = \min_{i=1}^{n} x_i$, $M_0(\mathbf{x}) = \left(\prod_{i=1}^{n} x_i\right)^{1/n}$ and $M_{\infty}(\mathbf{x}) = \max_{i=1}^{n} x_i$. So, p can be any affinely extended real number. A generalized mean is also known as a *power mean*, *Hölder mean* or p-mean. Particularly, the 1-mean is the arithmetic mean, while the 0-mean is the geometric mean.

In this paper, we extend the domain of x_i to include 0, i.e., $\mathbf{x} \in \mathbb{R}^n_{0+}$. Without loss of generality, suppose $x_1 = x_2 = \cdots = x_m = 0$, where $m \le n$. The generalized mean of $\mathbf{x}$ is defined as

$$M_p(\mathbf{x}) = \lim_{(x_1,x_2,\ldots,x_m)\to(0^+,0^+,\ldots,0^+)} \left(\frac{1}{n}\sum_{i=1}^{n} x_i^p\right)^{1/p}. \tag{1}$$

For brevity, we omit the lim symbol in Eq. (1).

Proposition 2.1 (Monotonicity). *For all $\mathbf{x} \in \mathbb{R}^n_{0+}$, $M_{p_1}(\mathbf{x}) \le M_{p_2}(\mathbf{x})$ if $p_1 \le p_2$.*

Generalized Vector Norms. Vector norms measure the "length" of vectors in a vector space. The *p-norm* for $p \ge 1$, also known as the *L^p-norm*, of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}. \tag{2}$$

Proposition 2.2 (Triangle Inequality). *For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\|\mathbf{x} + \mathbf{y}\|_p \le \|\mathbf{x}\|_p + \|\mathbf{y}\|_p$.*

The p-norms with different $p \ge 1$ satisfy the following relationship which is essential to derive the results in this paper.

Proposition 2.3. *For all $\mathbf{x} \in \mathbb{R}^n$ and $p \ge 1$, $\|\mathbf{x}\|_1 \le n^{1-1/p}\|\mathbf{x}\|_p$.*

From Eq. (1) and Eq. (2), for all $\mathbf{x} \in \mathbb{R}^n_{0+}$ and $p \ge 1$, we can write $M_p(\mathbf{x}) = n^{-1/p}\|x\|_p$.

2.2 Layer-Oriented Density of ML Graphs

In the literature, the density of an ML graph $\mathcal{G} = (V, E, L)$ is typically designed following the *layer-oriented* manner, that is, the density of $\mathcal{G}$ is defined as the aggregation of the density of the layers in $\mathcal{G}$ which can be formulated in the following unified way:

$$\delta_{q,p}(\mathcal{G}) = M_q([M_p([d_v(l)]_{v\in V})]_{l\in L}).$$

The unified layer-oriented density formulation is either a generalization or a component of many existing density measures for ML graphs. Figure 3 depicts the relationships between these layer-oriented density measures.

(1) *Minimum Average Degree Density.* Jethava and Beerenwinkel [31] studied the first densest subgraph problem on ML graphs, which extracts the densest common subgraph (DCS) maximizing the minimum average degree density. Specifically, the minimum average degree density of an ML graph $\mathcal{G} = (V, E, L)$ is defined as

$$\delta^{MAD}(\mathcal{G}) = MIN([AVG([d_v(l)]_{v\in V})]_{l\in L}),$$

where MIN returns the minimum, and AVG returns the average. Because $M_{-\infty}(\cdot) = MIN(\cdot)$ and $M_1(\cdot) = AVG(\cdot)$, we have $\delta^{MAD}(\mathcal{G}) = \delta_{1,-\infty}(\mathcal{G})$.

(2) *Aggregated Density.* Semertzidis et al. [50] studied the best friends forever (BBF) problem. Given an ML graph and two aggregation operations $AGG_1, AGG_2 \in \{MIN, AVG\}$, the BBF problem finds the induced subgraph with the maximum aggregated density. The aggregated density of an ML graph $\mathcal{G} = (V, E, L)$ is defined as

$$\delta^{AGG}(\mathcal{G}) = AGG_1([AGG_2([d_v(l)]_{v\in V})]_{l\in L}).$$

The minimum average degree density $\delta^{MAD}(\mathcal{G})$ is apparently a special case of $\delta^{AGG}(\mathcal{G})$ for $AGG_1 = MIN$ and $AGG_2 = AVG$. $\delta^{AGG}(\mathcal{G})$ is a special case of $\delta_{q,p}(\mathcal{G})$ for $p, q \in \{-\infty, 1\}$.

(3) *Multilayer Density.* Considering the trade-off between the high single-layer density of a subgraph and the number of layers where the high density holds, Galimberti et al. [22] define the multilayer density of an ML graph $\mathcal{G} = (V, E, L)$ as

$$\delta^{MLD}_{\beta}(\mathcal{G}) = \max_{L'\subseteq L} |L'|^{\beta} \cdot \delta_{-\infty,1}(\mathcal{G}|_{L'}),$$

where $\mathcal{G}|_{L'}$ denotes the ML graph comprising the layers in L', and $\beta \ge 0$ controls the tradeoff. $\delta^{MLD}_{\beta}(\cdot)$ is a composition of $\delta_{q,p}(\cdot)$.

(4) *p-mean Multilayer Density.* Behrouz et al. [6] extends the multilayer density to the p-mean multilayer density:

$$\delta^{MLD}_{\beta,p}(\mathcal{G}) = \max_{L'\subseteq L} |L'|^{\beta} \cdot \delta_{-\infty,p}(\mathcal{G}|_{L'}).$$

Certainly, $\delta^{MLD}_{\beta,p}(\cdot)$ is a composition of $\delta_{q,p}(\cdot)$.

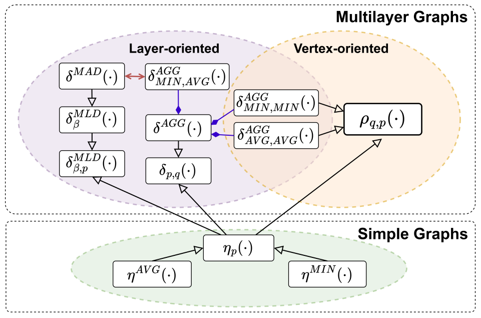


Figure 3: The relationships between the density measures for simple graphs and ML graphs.

2.3 Limitations of Layer-Oriented Density

The densest ML subgraph problems based on the layer-oriented density are often difficult to solve. There is currently no unified approach to solving these problems. Only special cases have been studied in the literature, and the existing solutions typically face scalability issues or return suboptimal results. Given an ML graph $\mathcal{G} = (V, E, L)$ and two affinely extended real numbers p and q, the problem of finding the subgraph $\mathcal{G}[S]$ where $S \subseteq V$ that maximizes $\delta_{q,p}(\mathcal{G}[S])$ is referred to as the layer-oriented densest ML subgraph (L-MGDS) problem in this paper.

(1) The DCS problem [31] based on the minimum average degree density $\delta^{MAD}(\cdot)$ is a special case of the L-MGDS problem for $p = 1$ and $q = -\infty$. It has been shown that this problem cannot be approximated within a factor of $2^{\log^{(1-\epsilon)}|V|}$, where $\epsilon \in (0, 1)$ [11]. The algorithm based on linear programming [31] is slow and cannot scale to large ML graphs. The greedy-selection-based algorithm [11] produces $\sqrt{|V|\log|L|}$-approximate solutions, and the greedy peeling algorithm [31] yields $2r$-approximations, where r depends on the features of the input ML graph with an uncommon best-case

value of 1. These approximation algorithms are typically unstable and may produce bad results.

(2) The BFF problem [50] based on the aggregated density $\delta^{AGG}(\cdot)$ with $AGG_1 = AVG$ and $AGG_2 = MIN$ is a special case of the L-MGDS problem for $p = -\infty$ and $q = 1$. Charikar et al. [11] have proven that this problem is NP-hard and cannot be approximated within a factor of $|V|^{1-\epsilon}$, where $\epsilon \in [0, 1]$. The exact algorithm [11] based on ML core decomposition [22] requires $O(|V|^{|L|}(|V||L| + |E|))$ time and is inapplicable to a large ML graph, especially when the graph contains a lot of layers. The approximation algorithm [50] based on greedy peeling attains a bad approximation ratio of $|V|$.

(3) The densest ML subgraph problem based on the multilayer density $\delta^{MLD}_{\beta}(\cdot)$ is also NP-hard [22]. Galimberti et al. [22] prove that the densest ML core in the input ML graph is a $2|L|^{\beta}$ approximation to the optimal solution. However, computing all ML cores requires exponential time in $|L|$. In addition, the best approximation ratio of 2 is achieved when $\beta \to 0$. When β is larger, denser subgraphs across more layers are preferred. However, the approximation ratio significantly degrades in this situation.

(4) The densest ML subgraph problem based on the p-mean multilayer density $\delta^{MLD}_{\beta,p}(\cdot)$ is certainly NP-hard [6]. Behrouz et al. [6] propose a generalized-FirmCore-based algorithm to solve this problem, which requires conducting $|L|$ peeling processes on the input ML graph. The algorithm runs in $O(|E||L|^2+|V||L|\log|L|)$ time and attains an approximation ratio of $\frac{(p+1)^{1/p}|L|^{\beta+1/p}}{\lambda^{\max\{1,\beta\}}}$, where $\lambda \in [1, |L|]$ depends on the features of the input ML graph. Notably, when $p = 1$, this algorithm achieves a better approximation for the problem based on the ML density $\delta^{MLD}_{\beta}(\cdot)$ for $\beta > 1$. However, this algorithm inherits the drawbacks of the ML-core-based approach to addressing the special case for $p = 1$. Although significantly alleviated, this algorithm remains time-consuming when dealing with a large number of layers, and the approximation ratio tends to degrade as β increases.

The limitations of these densest ML subgraph algorithms derive from the layer-oriented nature of the existing ML graph density measures. In the rest of the paper, we introduce a new design of ML graph density called the *vertex-oriented density*, which attains elegant properties that enable half of the problems in the problem space (as shown in Figure 2) to be efficiently solved with sound approximation guarantees using a unified greedy peeling framework.

3 PROBLEM STATEMENT

In this section, we propose a new family of density measures for ML graphs and formalize the problem studied in this paper.

Vertex-Oriented Density of ML Graphs. The layers in an ML graph usually have different structures, so the degree of a vertex varies across layers.

Definition 3.1 (Degree Vector). For an ML graph $\mathcal{G} = (V, E, L)$, the degrees of a vertex $v \in V$ on all layers constitute the degree vector $[d_v(1), d_v(2), \cdots, d_v(|L|)]$ of v, denoted as $\mathbf{d}_v$.

To extend the p-density [53] to ML graphs, we introduce a representative aggregated vertex degree that captures the overall degree information of a vertex and reflects its distribution across layers:

Definition 3.2 (q-mean degree). For an ML graph $\mathcal{G} = (V, E, L)$, the q-mean degree of a vertex $v \in V$, denoted as d^q_v, is the q-mean of the degree vector $\mathbf{d}_v$, where q is an affinely extended real number.

The q-mean degree offers a flexible and comprehensive approach to characterizing the distribution of vertex degrees across layers. For $q = 1$, the q-mean degree is the average degree across layers. As q grows larger, more emphasis is placed on larger degrees. For $q = \infty$, the q-mean degree is the maximum degree across layers. For $q = -\infty$, the q-mean degree is the minimum degree across layers. Adopting the q-mean degree allows for capturing various aspects of degree distributions, tailored to the specific characteristics of the ML graphs and application scenarios. In the special cases where $\mathcal{G}$ consists of only one layer, i.e., $|L| = 1$, the q-mean degree of a vertex is the degree of the vertex on the only layer.

Then, we give a formal definition of the *vertex-oriented generalized density* of an ML graph:

Definition 3.3 (Vertex-Oriented Generalized Density). Given an ML graph $\mathcal{G} = (V, E, L)$ and affinely extended real values q and p, the generalized density (also called the (q, p)-density) of $\mathcal{G}$ is the p-mean of the q-mean degrees of all vertices $v \in V$, denoted as

$$\rho_{q,p}(\mathcal{G}) = M_p([d^q_v]_{v\in V}).$$

The vertex-oriented generalized density $\rho_{q,p}(\cdot)$ is parameterized by two parameters q and p, with q adjusting emphasis on each vertex's cross-layer degrees and p shifting focus towards higher or lower q-mean degrees of vertices in the graph. The (q, p)-density exhibits the following monotonic property which directly follows the monotonicity of generalized means given in Proposition 2.1.

Lemma 3.4 (Monotonicity). *$\rho_{q_1,p_1}(\mathcal{G}) \le \rho_{q_2,p_2}(\mathcal{G})$ if $q_1 \le q_2$ and $p_1 \le p_2$.*

Relationships with Existing Density Measures. As illustrated in Figure 3, the vertex-oriented generalized density $\rho_{q,p}(\cdot)$ generalizes a number of density measures studied in the literature.

(1) *Aggregated Density.* Recall the aggregated density $\delta^{AGG}(\cdot)$ [50] described in Section 2.2. For $AGG_1 = AGG_2 = MIN$, we have

$$\begin{aligned}\delta^{AGG}(\mathcal{G}) &= \min([\min([d_v(l)]_{v\in V})]_{l\in L})\\ &= \min([\min([d_v(l)]_{l\in L})]_{v\in V}) = \rho_{-\infty,-\infty}(\mathcal{G}).\end{aligned}$$

Similarly, for $AGG_1 = AGG_2 = AVG$, we have

$$\begin{aligned}\delta^{AGG}(\mathcal{G}) &= \frac{1}{|L|}\sum_{l\in L}\frac{1}{|V|}\sum_{v\in V} d_v(l)\\ &= \frac{1}{|V|}\sum_{v\in V}\frac{1}{|L|}\sum_{l\in L} d_v(l) = \rho_{1,1}(\mathcal{G}).\end{aligned}$$

(2) *p-Density.* The *p-density* [53] of a simple graph $G = (V, E)$ is the p-mean of the degrees of all vertices in G, denoted as $\eta_p(G)$. For an ML graph $\mathcal{G} = (V, E \times \{1\}, \{1\})$ consisting of only one layer $G_1 = (V, E)$, we have $\rho_{q,p}(\mathcal{G}) = \eta_p(G_1)$.

(3) *Average Degree Density.* The *average degree density* [25] of a simple graph $G = (V, E)$ is $\eta^{AVG}(G) = |E|/|V|$. For $p = 1$ and an ML graph $\mathcal{G} = (V, E \times \{1\}, \{1\})$ containing only one layer $G_1 = (V, E)$, we have $\rho_{q,p}(\mathcal{G}) = 2\eta^{AVG}(G_1)$.

(4) *Minimum Degree Density.* The *minimum degree density* [53] of a simple graph $G = (V, E)$ is the minimum degree of the vertices in

Algorithm 1 MGDS Framework

```
Input: An ML graph $\mathcal{G} = (V, E, L)$ and two parameters $p, q \in \mathbb{R}$
Output: The subgraph $\mathcal{G}[S]$ where $S \subseteq V$ that maximizes the $(q,p)$-density $\rho_{q,p}(S)$
1: $S_1 \leftarrow V; k \leftarrow 1$
2: for $i \leftarrow 1$ to $|V|$ do
3:     $u \leftarrow \arg\min_{v \in S_i} \mathtt{score}_{p,q}(v, S_i)$
4:     $S_{i+1} \leftarrow S_i - \{u\}$
5:     if $\rho_{q,p}(S_i) > \rho_{q,p}(S_k)$ then
6:         $k \leftarrow i$
7: return $\mathcal{G}[S_k]$
```

Table 2: The score functions adopted in different cases.

Case	Condition	Score Function $\mathtt{score}_{q,p}(v,S)$	Description
1	$1 \le q \le p$	$\Delta_{q,p}(v,S)$ (Eq. (4))	Section 4.2
2	$1 \le p \le q$	$\hat{\Delta}_{q,p}(v,S)$ (Eq. (5))	Section 4.3
3	$p = -\infty$	$\|\mathbf{d}_v(S)\|_q$	Section 4.4
4	$-\infty < p \le 1 \le q$	$\|\mathbf{d}_v(S)\|_q$	Section 4.5

G, denoted as $\eta^{MIN}(G)$. For $p = -\infty$ and an ML graph $\mathcal{G} = (V, E \times \{1\}, \{1\})$ with one layer $G_1 = (V, E)$, we have $\rho_{q,p}(\mathcal{G}) = \eta^{MIN}(G_1)$.

Problem Formulation. The *vertex-oriented generalized densest ML subgraph (MGDS) problem* studied in this paper is stated as follows: Given an ML graph $\mathcal{G} = (V, E, L)$ and two affinely extended real numbers q and p, find the induced subgraph $\mathcal{G}[S]$ where $S \subseteq V$ that has the maximum generalized density $\rho_{q,p}(\mathcal{G}[S])$.

For ease of notation, we write the degree vector of a vertex v in a subgraph $\mathcal{G}[S]$ as $\mathbf{d}_v(S)$, the q-mean degree of v in $\mathcal{G}[S]$ as $d_v^q(S)$ and the (q,p)-density of $\mathcal{G}[S]$ as $\rho_{q,p}(S)$ in the rest of this paper because $\mathcal{G}[S]$ is uniquely determined by S given $\mathcal{G}$.

4 ALGORITHMS

In this section, we propose a number of algorithms for the MGDS problem. These algorithms are designed based on the well-known greedy peeling framework [4, 8, 10, 12, 32, 51], wherein the condition for choosing a vertex to be peeled (removed from the graph) depends on the setting of q and p. We provide the theoretical bounds on the accuracy and the time complexity of these algorithms for the cases when $q \ge 1$ or $p = -\infty$ as illustrated in Table 1.

4.1 Greedy Peeling Framework

The greedy peeling framework has been widely adopted in finding the densest subgraphs [4, 8, 32, 51] or cohesive graphs [5, 22, 28, 39] from various types of graphs. Algorithm 1 outlines the adaptation of this framework to extract the generalized densest subgraph from an ML graph $\mathcal{G} = (V, E, L)$. The for loop in lines 2–6 repeats the greedy peeling step. For $i = 1, 2, \ldots, |V|$, let $S_i \subseteq V$ be the set of vertices remaining before the ith peeling step. Initially, S_1 is set to V. In the ith peeling step, a vertex $v \in S_i$ with the minimum score $\mathtt{score}_{q,p}(v, S_i)$ is removed from S_i, thus obtaining S_{i+1} (lines 3–4). The formulation of $\mathtt{score}_{q,p} : V \times 2^V \to \mathbb{R}$ depends on q and p, which will be introduced in the rest of this section. The greedy peeling step repeats $|V|$ times until all vertices have been removed. The algorithm finally returns the induced subgraph $\mathcal{G}[S_i]$ that has the largest (q,p)-density $\rho_{q,p}(S_i)$.

The formulation of the score function $\mathtt{score}_{q,p}(\cdot,\cdot)$ is pivotal in ensuring a sufficiently good approximation to the densest ML subgraph. We have studied the design of the score function and found that the suitable score function depends on the values of q and p. In the rest of this section, we propose specific score function formulations for four settings of q and p as summarized in Table 2.

4.2 Case 1 ($1 \le q \le p$)

This subsection considers the situation $1 \le q \le p$. For $S \subseteq V$, we devise another objective function

$$f_{q,p}(S) = \frac{1}{|S|} \sum_{v \in S} \left(\sum_{l \in L} d_v(S,l)^q \right)^{p/q}. \quad (3)$$

Apparently, $\rho_{q,p}(S) = \left(\frac{1}{|L|}\right)^{1/q} f_{q,p}(S)^{1/p}$. Thus, the subset $S \subseteq V$ maximizing $f_{q,p}(S)$ must maximize $\rho_{q,p}(S)$. A c-approximation to the problem of maximizing $f_{q,p}(\cdot)$ directly implies a $c^{1/p}$-approximation to the problem of maximizing $\rho_{q,p}(\cdot)$.

Let $g_{q,p}(S)$ be the numerator of $f_{q,p}(S)$. $g_{q,p}(S)$ has a nice property that leads to a polynomial-time exact solution to the MGDS problem as well as an efficient approximation algorithm based on the greedy peeling framework described in Algorithm 1.

Exact Solution. Before presenting the exact solution, we introduce the concept of supermodular functions.

Definition 4.1. Given a discrete set X, let $h : 2^X \to \mathbb{R}$ be a function. The function h is *supermodular* if $h(A) + h(B) \le h(A \cap B) + h(A \cup B)$ for all $A, B \subseteq X$.

Lemma 4.2. *For $p \ge q \ge 1$, $g_{q,p}(\cdot)$ is supermodular.*

Proof. It is easy to extend the proof of Lemma 4.1 in [53] to prove this lemma. See Appendix A in the technical report. □

We outline a polynomial-time solution to the MGDS problem. Consider the decision version of the MGDS problem: is there a subset $S \subseteq V$ such that $f_{q,p}(S) \ge \theta$, where $\theta \ge 0$? The decision problem can be converted to the problem of finding the subset $S \subseteq V$ that maximizes

$$\Psi_{q,p}(S) = g_{q,p}(S) - \theta|S|.$$

Let $S^* = \arg\max_{S \subseteq V} \Psi_{q,p}(S)$. We have $f_{q,p}(S^*) \ge \theta$ if and only if $\Psi_{q,p}(S^*) \ge 0$. The supermodularity of $\Psi_{q,p}(\cdot)$ for $p \ge q \ge 1$ naturally follows from Lemma 4.3.

Lemma 4.3. *For $p \ge q \ge 1$, $\Psi_{q,p}(\cdot)$ is supermodular.*

Lemma 4.3 indicates that any polynomial-time algorithm for maximizing a supermodular objective function such as [38] can be used to find the subset $S \subseteq V$ maximizing $\Psi_{q,p}(S)$. Subsequently, the MGDS problem can be solved by performing a binary search over the range $[0, g_{q,p}(V)]$ of all possible values of θ to find the maximum θ^* of θ such that there is a subset $S^* \subseteq V$ satisfying $f_{q,p}(S^*) \ge \theta^*$. Since there are no more than $2^{|V|}$ distinct values of $f_{q,p}(S)$ for all $S \subseteq V$, a binary search returns θ^* within $|V|$ iterations. The subset S^* induces the densest subgraph.

It should be noted that the exact solution is practically inefficient for large real-world ML graphs. Therefore, we proceed to explore a more practical approximate solution using the greedy peeling framework described in Algorithm 1.

An Approximate Solution. The key idea behind this approximate solution is to greedily improve the density of the remaining subgraph by iteratively removing vertices that cause the minimum decrease of $g_{q,p}(\cdot)$, as defined below:

$$\Delta_{q,p}(v,S) = g_{q,p}(S) - g_{q,p}(S - \{v\}), \tag{4}$$

where $S \subseteq V$ and $v \in S$. We have the following result:

Theorem 4.4. *For $p \geq q \geq 1$, if $\Delta_{q,p}(v,S)$ is used as the score function in Algorithm 1, the subgraph returned by Algorithm 1 is a $(p+1)^{1/p}$-approximation to the optimal solution.*

Since $g_{q,p}(S)$ is supermodular, the problem of maximizing $f_{q,p}(S)$, that is, $g_{q,p}(S)/|S|$ is a special case of the densest supermodular subset (DSS) problem studied by Chekuri et al. [12] which is rephrased as follows: Given a ground set X and a non-negative supermodular function $h : 2^X \to \mathbb{R}_{\geq 0}$ where $h(\emptyset) = 0$, the DSS problem finds the subset $A \subseteq X$ to maximize $h(A)/|A|$.

Theorem 4.5 ([12]). *By repeatedly removing elements from X that minimizes the decrease of $h(X)$, the subset X' obtained during the process with the largest $h(X')/|X'|$ is a c_h-approximation to the optimal solution of the DSS problem, where*

$$c_h = \max_{A \subseteq X} \frac{\sum_{v \in A} h(A) - h(A - \{v\})}{h(A)}.$$

Based on Theorem 4.5, we can establish the result in Theorem 4.4 by letting $h = g_{q,p}$ and showing the following lemma:

Lemma 4.6. *For $p \geq q \geq 1$, we have $c_h \leq p + 1$.*

Proof. See Appendix B in the technical report. □

Time Complexity. Given a vertex subset $S \subseteq V$, computing the score of a vertex $v \in S$ costs $O(|L| d_v(S))$ time. Consider using a set or min-heap to dynamically maintain the scores of vertices. Computing the scores of all vertices and initializing the data structure costs $O(\sum_{v \in V} |L| d_v(V) + |V| \log |V|) = O(|L||E| + |V| \log |V|)$ time. Each peeling step identifies and removes a vertex with the minimum score, taking $O(\log |V|)$ time. When a vertex v is removed, the scores of vertices within a 2-hop distance from v are affected and need update. For any neighbor u of v, the scores of up to $O(d_u)$ neighbors of u are recomputed and reflected in the data structure, costing $O(d_u |L| \log |V|)$ time. Since u can be a neighbor for at most d_u vertices, the total score updates across all peeling steps require $O(\sum_{v \in V} d_v^2 |L| \log |V|)$ time. We then have the time complexity of Algorithm 1 in handling this case is $O(\sum_{v \in V} d_v^2 |L| \log |V|)$.

4.3 Case 2 ($1 \leq p \leq q$)

This subsection focuses on the situation $q \geq p \geq 1$. As $p \geq 1$, the subset $S \subseteq V$ maximizing $\rho_{q,p}(S)$ also maximizes $f_{q,p}(S)$ defined in Eq. (3). Unfortunately, in this case, the function $g_{q,p}(\cdot)$ is no longer supermodular, and applying the score function $\Delta_{q,p}(v,S)$ defined in Eq. (4) in Algorithm 1 cannot achieve the same approximation guarantee as stated in Theorem 4.4.

Despite this, we will demonstrate that by deriving an upper bound of $\Delta_{q,p}(v,S)$ as the score function, Algorithm 1 can still yield a dense ML subgraph with a guaranteed approximation. Let $\boldsymbol{\delta}_{u,v}$ denote an $|L|$-dimensional 0-1 vector where the ith component is 1 if and only if the vertices u and v are adjacent on layer i. We can express $\Delta_{q,p}(v,S)$ using norms and have

$$\begin{aligned}
&\Delta_{q,p}(v,S) \\
=&\|\mathbf{d}_v(S)\|_q^p + \sum_{u \in N_v(S)} \|\mathbf{d}_u(S)\|_q^p - \|\mathbf{d}_u(S) - \boldsymbol{\delta}_{u,v}\|_q^p \\
\leq&\|\mathbf{d}_v(S)\|_q^p + \sum_{u \in N_v(S)} p\|\mathbf{d}_u(S)\|_q^{p-1} \left(\|\mathbf{d}_u(S)\|_q - \|\mathbf{d}_u(S) - \boldsymbol{\delta}_{u,v}\|_q\right) \\
\leq&\|\mathbf{d}_v(S)\|_q^p + \sum_{u \in N_v(S)} p\|\mathbf{d}_u(S)\|_q^{p-1} \|\boldsymbol{\delta}_{u,v}\|_q.
\end{aligned}$$

The two inequalities are derived from the following Proposition 4.7 and the triangle inequality of q-norms (Proposition 2.2), respectively.

Proposition 4.7. *For $p \geq 1$ and $y \geq x \geq 0$, we have*

$$p(y-x)x^{p-1} \leq y^p - x^p \leq p(y-x)y^{p-1}.$$

Let

$$\hat{\Delta}_{q,p}(v,S) = \|\mathbf{d}_v(S)\|_q^p + \sum_{u \in N_v(S)} p\|\mathbf{d}_u(S)\|_q^{p-1} \|\boldsymbol{\delta}_{u,v}\|_q. \tag{5}$$

We have $\Delta_{q,p}(v,S) \leq \hat{\Delta}_{q,p}(v,S)$ for $q, p \geq 1$, and $\Delta_{q,p}(v,S) = \hat{\Delta}_{q,p}(v,S)$ for $q = p = 1$. Choosing $\hat{\Delta}_{q,p}(v,S)$ as the score function is due to its monotonicity described in the following lemma.

Lemma 4.8. *When $p \geq 1$, for all $R \subseteq T \subseteq V$, $\hat{\Delta}_{q,p}(v,R) \leq \hat{\Delta}_{q,p}(v,T)$ for all $v \in R$.*

Proof. Since $R \subseteq T$, it is easy to show that $\|\mathbf{d}_v(R)\|_q \leq \|\mathbf{d}_v(T)\|_q$ and $\|\mathbf{d}_u(R)\|_q \leq \|\mathbf{d}_u(T)\|_q$ for $u \in N_v(R)$. The lemma holds. □

Theorem 4.9. *When $q \geq p \geq 1$, if $\hat{\Delta}_{q,p}(v,S)$ is used as the score function in Algorithm 1, the subgraph returned by Algorithm 1 is a $\left(p|L|^{1-1/q} + 1\right)^{1/p}$-approximation to the optimal solution.*

Proof. As $p \geq 1$, a c-approximation to the problem of maximizing $f_{q,p}(\cdot)$ is immediately a $c^{1/p}$-approximation to the problem of maximizing $\rho_{q,p}(\cdot)$. We rewrite $f_{q,p}(S)$ with generalized norms:

$$f_{q,p}(S) = \frac{1}{|S|} \sum_{v \in S} \|\mathbf{d}_v(S)\|_q^p$$

Let S^* be the optimal solution that maximizes $f_{q,p}(S)$ for $S \subseteq V$. Certainly, S^* maximizes $\rho_{q,p}(S)$. The optimality of S^* ensures that removing any vertex v from S^* will result in a reduction of $f_{q,p}(\cdot)$:

$$\frac{1}{|S^*|} \sum_{u \in S^*} \|\mathbf{d}_u(S^*)\|_q^p \geq \frac{1}{|S^*| - 1} \left(\left(\sum_{u \in S^*} \|\mathbf{d}_u(S^*)\|_q^p \right) - \Delta_{q,p}(v,S^*) \right).$$

By simple mathematics, we have $\Delta_{q,p}(v,S^*) \geq f_{q,p}(S^*)$.

Suppose v is the first vertex in S^* removed by Algorithm 1 and $T \subseteq V$ is the vertex subset obtained during the peeling process just before v is removed. It is obvious that $S^* \subseteq T$. By Lemma 4.8, we have $\hat{\Delta}_{q,p}(v,S^*) \leq \hat{\Delta}_{q,p}(v,T)$. Moreover, since v is greedily selected in T, v must minimize $\hat{\Delta}_{q,p}(u,T)$ among all vertices $u \in T$. Therefore,

$$f_{q,p}(S^*) \leq \Delta_{q,p}(v,S^*) \leq \hat{\Delta}_{q,p}(v,S^*) \leq \hat{\Delta}_{q,p}(v,T)$$

$$\leq \frac{1}{|T|}\sum_{u\in T}\hat{\Delta}_{q,p}(u,T). \tag{6}$$

The right-hand side of the inequality can be expanded to

$$\begin{aligned}
&\frac{1}{|T|}\sum_{u\in T}\hat{\Delta}_{q,p}(u,T)\\
=&\frac{1}{|T|}\sum_{u\in T}\left(\|\mathbf{d}_u(T)\|_q^p+\sum_{w\in N_u(T)}p\|\mathbf{d}_w(T)\|_q^{p-1}\left(\|\boldsymbol{\delta}_{w,u}\|_q\right)\right)\\
=&f_{q,p}(T)+\frac{1}{|T|}\sum_{u\in T}\sum_{w\in N_u(T)}p\|\mathbf{d}_w(T)\|_q^{p-1}\|\boldsymbol{\delta}_{w,u}\|_q)\\
=&f_{q,p}(T)+\frac{1}{|T|}\sum_{w\in T}p\|\mathbf{d}_w(T)\|_q^{p-1}\sum_{u\in N_w(T)}\|\boldsymbol{\delta}_{w,u}\|_q.
\end{aligned}$$

As $\|\boldsymbol{\delta}_{w,u}\|_q$ is strictly decreasing as q increases, we have

$$\begin{aligned}
\sum_{u\in N_w(T)}\|\boldsymbol{\delta}_{w,u}\|_q \leq \sum_{u\in N_w(T)}\|\boldsymbol{\delta}_{w,u}\|_1 &= \sum_{u\in N_w(T)}\sum_{l\in L}\boldsymbol{\delta}_{w,u}[l]\\
&=\sum_{l\in L}\sum_{u\in N_w(T,l)}1=\sum_{l\in L}d_w(T,l)=\|\mathbf{d}_w(T)\|_1\\
&\leq |L|^{1-\frac{1}{q}}\|\mathbf{d}_w(T)\|_q,
\end{aligned} \tag{7}$$

where the last inequality follows from Proposition 2.3. Then,

$$\begin{aligned}
&\frac{1}{|T|}\sum_{u\in T}\hat{\Delta}_{q,p}(u,T)\\
\leq&f_{q,p}(T)+\frac{1}{|T|}\sum_{w\in T}p\|\mathbf{d}_w(T)\|_q^{p-1}\left(|L|^{1-\frac{1}{q}}\|\mathbf{d}_w(T)\|_q\right)\\
=&f_{q,p}(T)+\frac{1}{|T|}\sum_{w\in T}p|L|^{1-\frac{1}{q}}\|\mathbf{d}_w(T)\|_q^p\\
=&f_{q,p}(T)+p|L|^{1-\frac{1}{q}}f_{q,p}(T)=\left(1+p|L|^{1-\frac{1}{q}}\right)f_{q,p}(T).
\end{aligned} \tag{8}$$

Consequently, $f_{q,p}(S^*) \leq \left(1+p|L|^{1-\frac{1}{q}}\right)f_{q,p}(T)$, and therefore, $\rho_{q,p}(S^*) \leq \left(1+p|L|^{1-\frac{1}{q}}\right)^{1/p}\rho_{q,p}(T)$. As T is in the sequence of vertex subsets greedily generated by Algorithm 1, the algorithm must return a $\left(1+p|L|^{1-\frac{1}{q}}\right)^{1/p}$-approximation to S^*. □

Notably, the proof of Theorem 4.9 does not depend on $q \geq p$. Therefore, using $\hat{\Delta}_{q,p}(v,S)$ as the score function in Algorithm 1 also leads to a $\left(1+p|L|^{1-\frac{1}{q}}\right)^{1/p}$-approximation algorithm for the situation $p \geq q \geq 1$ (Case 1) studied in Section 4.2. However, this algorithm is suboptimal compared with the $(p+1)^{1/p}$-approximation algorithm designed for Case 1 ($p \geq q \geq 1$) in Section 4.2.

Time Complexity. Similar to Case 1 where $p \geq q \geq 1$, removing a vertex will result in changes to the scores of vertices within its 2-hop neighborhood. Thus, Algorithm 1 incurs the same time cost $O(\sum_{v\in V} d_v(V)^2|L|\log|V|)$ to handle Case 2 as it deals with Case 1.

4.4 Case 3 ($p = -\infty$)

This subsection studies the situation $p = -\infty$. In this situation, $\rho_{q,p}(S)$ is exactly the minimum q-mean degree of the vertices in $\mathcal{G}[S]$, that is, $\rho_{q,p}(S) = \min_{v\in S} d_v^q(S)$. Inspired by the algorithm for identifying the densest subgraph maximizing the minimum degree in simple graphs that iteratively removes the vertex with the minimum degree, also known as computing the k-core with the maximum integer k [5], we use the q-mean degree of a vertex v in $\mathcal{G}[S]$ as the score function in Algorithm 1. As $d_v^q(S) = |L|^{-1/q}\|\mathbf{d}_v(S)\|_q$ for all $v \in S$, we can simply use $\|\mathbf{d}_v(S)\|_q$ as the score function.

Theorem 4.10. *When $p = -\infty$, if $\|\mathbf{d}_v(S)\|_q$ is used as the score function in Algorithm 1, Algorithm 1 returns the optimal solution.*

Proof. Let $S^* \subseteq V$ induce the optimal solution $\mathcal{G}[S^*]$. Suppose v is the first vertex in S^* removed by Algorithm 1 during the peeling process. Let $T \subseteq V$ be the vertex subset obtained in Algorithm 1 just before v is removed. It is clear that $S^* \subseteq T$ and $\mathbf{d}_v(S^*)$ is element-wisely no larger than $\mathbf{d}_v(T)$. Therefore, $\|\mathbf{d}_v(S^*)\|_q \leq \|\mathbf{d}_v(T)\|_q$.

Since v is greedily selected from T, v must have the minimum score $\|\mathbf{d}_v(T)\|_q$ among all vertices $u \in T$. Thus, we have

$$\min_{u\in T}\|\mathbf{d}_u(T)\|_q = \|\mathbf{d}_v(T)\|_q \geq \|\mathbf{d}_v(S^*)\|_q = \min_{u\in S^*}\|\mathbf{d}_u(S^*)\|_q.$$

It follows that

$$\rho_{q,p}(T) = \min_{u\in T}|L|^{-\frac{1}{q}}\|\mathbf{d}_u(T)\|_q \geq \min_{u\in S^*}|L|^{-\frac{1}{q}}\|\mathbf{d}_u(S^*)\|_q = \rho_{q,p}(S^*).$$

Since S^* induces the optimal solution, we have $\rho_{q,p}(S^*) \geq \rho_{q,p}(T)$. Therefore, $\rho_{q,p}(S^*) = \rho_{q,p}(T)$. Of course, $\mathcal{G}[T]$ is returned by Algorithm 1. Thus, the theorem holds. □

Time Complexity. In contrast to the previous cases, when adopting $\|\mathbf{d}_v(S)\|_q$ as the score function, removing a vertex results in changes solely to the scores of its immediate neighbors. For a removed v, recomputing the scores of v's neighbors and reflecting this change in the set or heap structure takes $O(d_v(|L| + \log|S|))$ time. Therefore, the entire peeling process runs in $O(\sum_{v\in V} d_v(|L| + \log|S|) = O(|L||E| + |E|\log|V|)$ time. For the extreme case when $q = -\infty$, $\|\mathbf{d}_v(S)\|_q$ is the minimum degree of v in $\mathcal{G}[S]$ across all layers and thereby is an integer value. This allows for a linear-heap-implementation [9] of the peeling process, which incurs lightweight insertion and update operations in $O(1)$ time. Thus, the peeling process can be accomplished within $O(|L|(|V| + |E|))$ time.

4.5 Case 4 ($-\infty < p \leq 1 \leq q$)

This subsection investigates the situation $-\infty < p \leq 1 \leq q$. In simple graphs, the densest subgraph maximizing the p-density for $p = -\infty$ serves as a $1/2$-approximation to the densest subgraph that maximizes the p-density for $p = 1$ [19, 53]. Inspired by this idea, we will show that the algorithm proposed for Case 3 ($p = -\infty$) in Section 4.4 achieves a satisfactory approximation ratio in the situation $q \geq p = 1$. Moreover, this theoretical result can be generalized to the situation $q \geq 1 \geq p > -\infty$.

Theorem 4.11. *When $q \geq p = 1$, if $\|\mathbf{d}_v(S)\|_q$ is used as the score function in Algorithm 1, the subgraph returned by Algorithm 1 is a $\left(|L|^{1-\frac{1}{q}}+1\right)$-approximation to the optimal solution.*

Proof. The proof overlaps with the proofs of Theorem 4.9 and Theorem 4.10, and we outline the key differences and streamline the similar deduction process. As $p = 1$, $f_{q,p}(S)$ can be simplified to

$$f_{q,p}(S) = \frac{1}{|S|}\sum_{v\in S}\|\mathbf{d}_v(S)\|_q.$$

Let $S \subseteq V$ induce the optimal solution $\mathcal{G}[S]$ that maximizes $f_{q,1}(S)$. By the same deduction process as in the proof of Theorem 4.9, we have $\Delta_{q,p}(v,S) \geq f_{q,1}(S)$. As $p = 1$,

$$\begin{aligned}\Delta_{q,p}(v,S) &= \|\mathbf{d}_v(S)\|_q + \sum_{u \in N_v(S)} \|\mathbf{d}_u(S)\|_q - \|\mathbf{d}_u(S) - \boldsymbol{\delta}_{u,v}\|_q \\ &\leq \|\mathbf{d}_v(S)\|_q + \sum_{u \in N_v(S)} \|\boldsymbol{\delta}_{u,v}\|_q \\ &\leq \|\mathbf{d}_v(S)\|_q + |L|^{1-\frac{1}{q}} \|\mathbf{d}_v(S)\|_q \\ &= \left(1 + |L|^{1-\frac{1}{q}}\right) \|\mathbf{d}_v(S)\|_q,\end{aligned}$$

where the two inequalities follow from the triangle inequality of q-norms (Proposition 2.2) and Eq. (7) established in the proof of Theorem 4.9.

Suppose v is the first vertex in S removed by Algorithm 1 and $T \subseteq V$ is the vertex subset obtained during the peeling process before v is removed. We have $S \subseteq T$ and $\|\mathbf{d}_v(S)\|_q \leq \|\mathbf{d}_v(T)\|_q$. The relationship between $f_{q,1}(T)$ and $f_{q,1}(S)$ is derived as follows:

$$\begin{aligned}f_{q,1}(T) &= \frac{\sum_{u \in T} \|\mathbf{d}_u(T)\|_q}{|T|} \geq \|\mathbf{d}_v(T)\|_q \geq \|\mathbf{d}_v(S)\|_q \\ &\geq \frac{\Delta_{q,p}(v,S)}{1 + |L|^{1-\frac{1}{q}}} \geq \frac{f_{q,1}(S)}{1 + |L|^{1-\frac{1}{q}}}.\end{aligned}$$

Consequently, we have $\rho_{q,1}(T) \geq \left(1 + |L|^{1-\frac{1}{q}}\right)^{-1} \rho_{q,1}(S)$. Since T is in the sequence of vertex subsets obtained by Algorithm 1, the result returned by Algorithm 1 is a $\left(1 + |L|^{1-\frac{1}{q}}\right)$-approximation to the optimal solution $\mathcal{G}[S]$. □

The proof of Theorem 4.11 yields the following lemma as a byproduct, which lays the foundation for deriving guaranteed approximations for $-\infty < p < 1 \leq q$.

Lemma 4.12. *Let $S^*_{q,p} \subseteq V$ be the subset that maximizes $\rho_{q,p}(S)$ for $S \subseteq V$. It holds that $\rho_{q,1}(S^*_{q,1}) \leq \left(1 + |L|^{1-1/q}\right) \rho_{q,-\infty}(S^*_{q,-\infty})$ when $q \geq 1$.*

Proof. See Appendix D in the technical report. □

Based on Lemma 4.12, we establish an approximation algorithm for the situation $-\infty < p < 1 \leq q$.

Theorem 4.13. *When $-\infty < p < 1 \leq q$, if $\|\mathbf{d}_v(S)\|_q$ is used as the score function in Algorithm 1, the subgraph returned by Algorithm 1 is a $\left(|L|^{1-\frac{1}{q}} + 1\right)$-approximation to the optimal solution.*

Proof. Let $S^*_{q,p} \subseteq V$ be the subset that maximizes $\rho_{q,p}(S)$ for $S \subseteq V$, and let $S_{q,p}$ be the set of vertices in the subgraph returned by Algorithm 1. In this case, when q is fixed, Algorithm 1 carries out the same peeling process regardless of p because the score function $\|\mathbf{d}_v(S)\|_q$ is independent of p. For $p < 1$ and $q \geq 1$, since $S_{q,p}$ is returned as the result, we have $\rho_{q,p}(S_{q,p}) \geq \rho_{q,p}(S_{q,-\infty})$. Then,

$$\rho_{q,p}(S_{q,p}) \geq \rho_{q,p}(S_{q,-\infty}) \geq \rho_{q,-\infty}(S_{q,-\infty}) = \rho_{q,-\infty}(S^*_{q,-\infty}),$$

where the second inequality follows from the monotonicity of $\rho_{q,p}(\cdot)$ described in Lemma 3.4, and the last equality is due to Theorem 4.10 (the algorithm returns the optimal solution when $p = -\infty$). Moreover,

$$\rho_{q,p}(S^*_{q,p}) \leq \rho_{q,1}(S^*_{q,p}) \leq \rho_{q,1}(S^*_{q,1}) \leq \left(1 + |L|^{1-\frac{1}{q}}\right) \rho_{q,-\infty}(S^*_{q,-\infty}),$$

where the first inequality follows from Lemma 3.4, the second inequality is due to the optimality of $S^*_{q,1}$, and the last inequality is due to Lemma 4.12. Finally, we have

$$\rho_{q,p}(S^*_{q,p}) \leq \left(1 + |L|^{1-\frac{1}{q}}\right) \rho_{q,p}(S_{q,p}).$$

Thus, the theorem holds. □

Time Complexity. Algorithm 1 employs the same score function $\|\mathbf{d}_v(S)\|_q$ for this case as in Case 3 ($p = -\infty$). Consequently, Algorithm 1 yields the same peeling process and exhibits the same time complexity of $O(|L||E| + |E| \log |V|)$ in both cases.

4.6 Common Cases and Unexplored Case

Common Cases. As illustrated in Figure 2, the situations studied in the previous subsections have overlappings. In this subsection, we will demonstrate the approximation performance of the proposed algorithms in these common situations.

(1) Case 1, Case 2 and Case 4 intersect at $q = p = 1$. In this situation,

$$\begin{aligned}\Delta_{q,p}(v,S) = \hat{\Delta}_{q,p}(v,S) &= \|\mathbf{d}_v(S)\|_1 + \sum_{u \in N_v(S)} \|\boldsymbol{\delta}_{u,v}\|_1 \\ &= \|\mathbf{d}_v(S)\|_1 + \sum_{l \in L} d_v(S,l) = 2\|\mathbf{d}_v(S)\|_1.\end{aligned}$$

for all $S \subseteq V$ and all $v \in S$. It implies that the score functions $\Delta_{q,p}(v,S)$, $\hat{\Delta}_{q,p}(v,S)$ and $\|\mathbf{d}_v(S)\|_q$ used in these cases are consistent in this situation, and they will lead to the same peeling process and therefore the same result. Theorems 4.4, 4.9 and 4.11 show that the algorithms designed for Case 1, Case 2 and Case 4 all have the approximation ratio of 2 in this situation.

(2) Case 1 and Case 2 also intersect at $q = p > 1$. As discussed at the end of Section 4.3, using $\hat{\Delta}_{q,p}(v,S)$ as the score function leads to an approximation ratio of $\left(1 + p|L|^{1-\frac{1}{q}}\right)^{1/p}$, while using $\Delta_{q,p}(v,S)$ as the score function results in a better approximation ratio of $(p+1)^{1/p}$ because $|L|^{1-1/q} > 1$ for $|L| > 1$ and $q > 1$.

(3) Case 2 and Case 4 also intersect at $q > 1$ and $p = 1$. Theorems 4.9 and 4.11 show that using the score functions $\hat{\Delta}_{q,p}(v,S)$ and $\|\mathbf{d}_v(S)\|_q$ both lead to the approximation ratio of $(|L|^{1-1/q} + 1)$ in this situation.

Unexplored Case. Designing efficient algorithms with provable approximation guarantees for $p > -\infty$ and $q < 1$ is still an open problem. This is mainly because the triangle inequality of q-norms (Proposition 2.2) which serves as the foundation for proving the approximation guarantees for Case 1 and Case 2 does not hold in this situation. Moreover, when $q < 1$ is fixed, the key to establishing the approximation guarantee for Case 3, i.e., the relationship between the densities $\rho_{q,1}(S^*_{q,1})$ and $\rho_{q,-\infty}(S^*_{q,-\infty})$ described in Lemma 4.12, where $S^*_{q,p}$ is the subset that maximizes $\rho_{q,p}(S)$ for $S \subseteq V$, does not hold in this situation.

Algorithm 2 Fast implementation for MGDS

```
Input: An ML graph $\mathcal{G} = (V, E, L)$ and two parameters $p, q \geq 1$.
Output: The subgraph $\mathcal{G}[S]$ where $S \subseteq V$ that maximizes the $(q, p)$-density $\rho_{q,p}(S)$.
 1: $S_1 \leftarrow V; k \leftarrow 1$
 2: for $v \in V$ do
 3:     $score_v \leftarrow \hat{\Delta}_{q,p}(v, V)$        ▷ $score_v$ is the approximate score value of $v$
 4:     $\hat{d}_v \leftarrow \|\mathbf{d}_v(V)\|_q$        ▷ $\hat{d}_v$ maintains the approximate $q$-norm of $v$'s degree vector
 5: for $i \leftarrow 1$ to $|V|$ do
 6:     $v \leftarrow \arg\min_{v \in S_i} score_v$
 7:     $S_{i+1} \leftarrow S_i - \{v\}$
 8:     if $\rho_{q,p}(S_i) > \rho_{q,p}(S_k)$ then
 9:         $k \leftarrow i$
        /* Update approximate scores of vertices */
10:     for $u \in N_v(S_i)$ do
11:         $score_u \leftarrow score_u - \left(\|\mathbf{d}_u(S_i)\|_q^p - \|\mathbf{d}_u(S_{i+1})\|_q^p\right)$        ▷ Update the first term
12:         $score_u \leftarrow score_u - p\|\hat{d}_v\|_q^{p-1}\|\boldsymbol{\delta}_{u,v}\|_q$  ▷ Update the second term for removing $v$
13:         if $\hat{d}_u > \left(1 + \frac{\epsilon}{p-1}\right)\|\mathbf{d}_u(S_{i+1})\|_q$ then
14:             for $w \in N_u(S_{i+1})$ do
15:                 $score_w \leftarrow score_w - p\left(\|\mathbf{d}_u(S_i)\|_q^{p-1} - \|\mathbf{d}_u(S_{i+1})\|_q^{p-1}\right)(\|\boldsymbol{\delta}_{w,u}\|_q)$
16:             $\hat{d}_u \leftarrow \|\mathbf{d}_u(S_{i+1})\|_q$
17: return $\mathcal{G}[S_k]$
```

5 A FASTER APPROXIMATION ALGORITHM

As discussed in Section 4.2 and Section 4.3, in the peeling process of Algorithm 1 for $q, p \geq 1$ (Case 1 and Case 2), the removal of a vertex $v \in V$ affects the scores of vertices within a 2-hop distance from v. Updating these scores incurs substantial computational overhead, especially for dense ML graphs with vertices of large degrees. In this section, we extend the fast greedy peeling process for the densest subgraph problem maximizing the p-density [53] proposed by Chekuri and Torres [13] and introduce a more efficient implementation for $p, q \geq 1$ at the expense of an insignificant loss in the result accuracy.

Recalling Section 4.3, we demonstrated that when $p, q \geq 1$, utilizing $\hat{\Delta}q, p(v, S)$ as the score function leads to a $\left(1 + p|L|^{1-\frac{1}{q}}\right)^{1/p}$-approximation to the optimal solution. Building upon this, our efficient implementation employs a "lazy" approach to update the scores of vertices while ensuring a $(1 + 2\epsilon)$-approximation to the score for each vertex, denoted as $\widetilde{\Delta}_{q,p}(v, S)$, where $\epsilon \in [0, 1]$.

Theorem 5.1. *When $p, q \geq 1$, if $\widetilde{\Delta}_{q,p}(v, S)$ is used as the score function in Algorithm 1, the subgraph returned by Algorithm 1 is a $\left((1 + 2\epsilon)\left(1 + p|L|^{1-1/q}\right)\right)^{1/p}$-approximation to the optimal solution.*

Proof. The theorem can be proved by slightly adapting the proof of Theorem 4.9. See Appendix E in the technical report. □

Now, the challenge lies in maintaining an approximate score for each vertex with minimal update times. Recall Eq. (5), where the score of a vertex v with respect to a set S is defined as

$$\hat{\Delta}_{q,p}(v, S) = \|\mathbf{d}_v(S)\|_q^p + \sum_{u \in N_v(S)} p\|\mathbf{d}_u(S)\|_q^{p-1}\|\boldsymbol{\delta}_{u,v}\|_q.$$

Here, the first term reflects v's contribution to the numerator of $f_{q,p}(S)$, i.e., $g_{q,p}(S)$, and the second term bounds the decreases of the contributions of v's neighbors to $g_{q,p}(S)$ when v is removed. In fact, direct changes to v's degrees affect the first term, whereas changes to the neighbors' degrees impact the second term. We adopt the following "lazy" update strategies: changes to v's degrees are immediately reflected in the first term, while the second term is updated only when the degrees of a neighbor u undergo substantial change. We here use the q-norm of the u's degree vector to characterize the overall degree of u across all layers. By maintaining a $\left(1 + \frac{\epsilon}{p-1}\right)$-approximation of $\|\mathbf{d}_u(S)\|_q$, we update the second term if this approximate value changes. We present the following lemma to ensure that $\widetilde{\Delta}_{q,p}(v, S) \leq (1 + 2\epsilon)\hat{\Delta}_{q,p}(v, S)$.

Lemma 5.2. *When $p \geq 1$, it holds that $\left(\gamma\|\mathbf{d}_u(S)\|_q\right)^{p-1} \leq (1 + 2\epsilon)\|\mathbf{d}_u(S)\|_q^{p-1}$, where $\epsilon \in [0, 1]$ and $\gamma = 1 + \frac{\epsilon}{p-1}$.*

Proof. See Appendix F in the technical report. □

Algorithm 2 outlines this efficient implementation. Theorem 5.1 and Lemma 5.2 ensure the correctness of Algorithm 2.

Time Complexity. The time overhead of Algorithm 2 is primarily caused by the **for** loop in lines 14–16, which updates scores of the 2-hop neighbors of the removed vertex. For any vertex u, lines 14–16 are triggered at most $O(\log_{1+\frac{\epsilon}{p-1}}\|\mathbf{d}_u(V)\|_q)$ times, with each run using $O(d_u|L|\log|V|)$ time. Consequently, the total time complexity for lines 14–16, and thus the time complexity of Algorithm 2, is $O(\sum_{u \in V} d_u|L|\log|V| \cdot \log_{1+\frac{\epsilon}{p-1}}|L||V|) = O\left(\frac{|E|\log|V|\log|L||V|}{\log(1+\frac{\epsilon}{p-1})}\right)$.

Table 3: The statistics of the ML graphs.

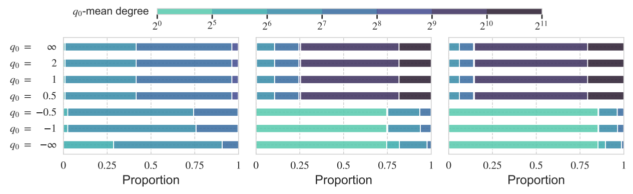
Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert L\rvert$	$\max_{v \in V} d_v$
FAO	214	268,339	364	189
Sacchcere	6,570	247,152	7	3,254
Homo	18,190	153,922	7	9,724
Amazon	410,236	8,132,506	4	3,260
Higgs	456,631	13,706,153	4	70,357
Friendfeed	505,104	18,673,521	3	75,071
DBLP-small	513,627	1,015,808	10	360
DBLP-large	1,824,674	10,169,845	22	2,119
ObamaInIsrael	2,279,535	3,827,964	3	37,241
StackOverflow	2,584,164	38,125,417	24	44,065

6 EXPERIMENTS

6.1 Experimental Setup

Algorithms. We compare our vertex-oriented density $\rho_{q,p}(\cdot)$ with two representative layer-oriented density measures: the multilayer density $\delta_{\beta}^{MLD}(\cdot)$ and the p-mean multilayer density $\delta_{\beta,p}^{MLD}(\cdot)$. The reasons are two-fold: (1) These two density definitions generalize the minimum average degree density $\delta^{MAD}(\cdot)$ and allow tuning focus to layers that contribute to the density, making them robust against noisy layers. (2) The densest subgraphs maximizing these two density measures can both be approximated in polynomial time, making them applicable for large ML graphs.

The following algorithms designed for these density measures are evaluated: **(1)** MGDS: the algorithm (Algorithm 1) for finding the generalized densest ML subgraph under our density $\rho_{q,p}(\cdot)$. MGDS-Cx represents the version of MGDS proposed to solve Case x of the problem, where $x \in \{1, 2, 3, 4\}$. **(2)** MGDS-Fast: the fast implementation of MGDS (Algorithm 2) for Case 1 and Case 2. MGDS-Fast-Cx represents the version of MGDS-Fast for Case x, where $x \in \{1, 2\}$. **(3)** FC: the FirmCore-based algorithm [27] for finding the densest ML subgraph under the multilayer density $\delta_{\beta}^{MLD}(\cdot)$. **(4)** GFC: the generalized-FirmCore-based algorithm [6] for finding the densest ML subgraph under the density $\delta_{\beta,p}^{MLD}(\cdot)$.



(a) $p = -\infty$, $q = -\infty$. (b) $p = -\infty$, $q = 1$. (c) $p = -\infty$, $q = 2$.

Figure 4: The distribution of the q_0-mean degrees of the vertices in the subgraph output by MGDS on Friendfeed.

Datasets. The experiments were conducted on 10 real-world ML graphs, with the statistics listed in Table 3. The FAO dataset is sourced from [16]. The datasets DBLP-large and StackOverflow are obtained from the KONECT Project [35] and are transformed into ML graphs by organizing edges into different layers according to their timestamps. The other ML graphs are obtained from [22].

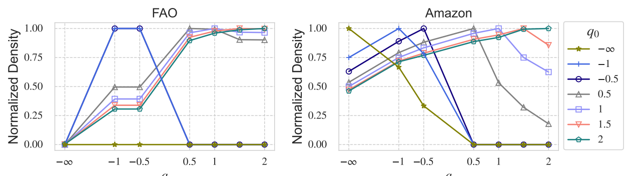
Environments. The experiments were conducted on a server with an Intel Xeon Gold 5218R processor and 754GB of RAM, running 64-bit Ubuntu 22.04. All the algorithms were implemented in C++ and compiled with GCC 9.4.0, utilizing -O3 optimization.

6.2 Effectiveness Evaluation

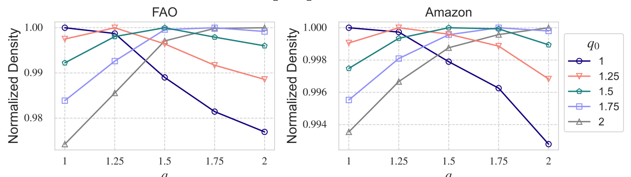
Characteristics of the Generalized Densest ML Subgraphs. When q is fixed and p increases, we observe a consistent increase in size and decrease in edge density of the ML subgraphs output by MGDS. This aligns with the findings reported in [53] on the densest subgraph maximizing the p-density in a single-layer graph.

We next examine the dense ML subgraphs produced by MGDS when fixing p and varying q. Figure 4 depicts the distribution of the q_0-mean degrees of vertices in the subgraph obtained for $p = -\infty$ and $q \in \{-\infty, 1, 2\}$ on Friendfeed. We observe a correlation between q and q_0: as q increases (or decreases resp.), the resulting subgraph encompasses more vertices with higher q_0-mean degrees for large (or smaller resp.) q_0. This fully illustrates how q adjusts the preference on the vertex degrees across layers. When adopting a larger q, the q-mean degree captures higher degrees in the degree vector, and the resulting subgraph tends to contain vertices with sufficiently large degrees on certain layers, which exhibit high q_0-mean degrees for $q \geq 0.5$. However, due to the skewness in vertex degrees across layers, these vertices may exhibit very low degrees on other layers, which we consider noisy or insignificant for those vertices. Consequently, the q_0-mean degrees drop significantly when q_0 decreases from 0.5 to -0.5. When $q = -\infty$, the resulting subgraph covers vertices with sufficient minimum degrees across layers, which shows high q_0-mean degrees for small q_0.

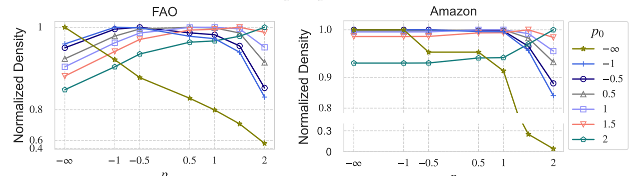
Performance of the MGDS Framework. To demonstrate the effectiveness of our MGDS framework, we conduct a comparative analysis of the results obtained under different settings of q and p and evaluate their quality with various (q_0, p_0)-densities. As illustrated in Figure 5, each line representing a generalized density measure typically displays a distinct pattern. It suggests that our MGDS framework is able to uncover meaningful dense ML subgraphs with diverse characteristics. Furthermore, we see consistent trends from the figure: when varying q (or p), the peak density for



(a) $p = p_0 = -\infty$.



(b) $p = p_0 = 1$.



(c) $q = q_0 = 1$.

Figure 5: Comparison between the results output by MGDS under different settings of q and p on FAO and Amazon in terms of the (q_0, p_0)-densities for various q_0 and p_0.

Table 4: The densities of the subgraphs output by MGDS, FC, and GFC on Sacchcere.

Method	Per-layer Density	$\rho_{1,1}(\cdot)$	$\delta_2^{MLD}(\cdot)$	$\delta_{2,1}^{MLD}(\cdot)$
MGDS	[**19.3**, **26.1**, 7.4, **38.9**, 0.9, 0.7, **76.1**]	**24.2**	308.2	517.4
FC	[**23.9**, **23.8**, 11.2, **27.4**, 1.2, 0.8, **59.8**]	21.1	**380.8**	528.9
GFC	[**20.1**, **14.5**, 10.6, **14.7**, 0.7, 0.5, **39.6**]	14.4	264.3	**568.3**

each (q, p_0)-density (or (q_0, p)-density) measure occurs at $q = q_0$ (or $p = p_0$), and the density shows monotonicity on both sides of this point. It verifies that applying the MGDS framework for different q and p optimizes different regimes of the generalized densest ML subgraph problem.

Comparison with the Layer-oriented Densest ML Subgraphs. We begin by reporting the densities of the subgraphs output by MGDS, FC, and GFC on Sacchcere for $p = q = 1$ and $\beta = 2$ in Table 4. Unsurprisingly, each dense subgraph competes with the others in terms of the density measure it optimizes. From a layer-oriented perspective, the multilayer density $\delta_2^{MLD}(\cdot)$ of the subgraph identified by FC is determined by the four layers with the maximum average degree densities (shown in bold), while the remaining layers are considered noisy and ignored. The subgraph output by MGDS exhibits comparable density in each layer selected by FC. It suggests the effectiveness of our vertex-oriented density in handling noisy layers. The subgraph output by GFC has relatively lower per-layer densities. It is due to its emphasis on maximizing the 2-mean of the degrees of vertices, which may lead to the inclusion of vertices with low degrees. We offer another vertex-oriented-perspective comparison between these subgraphs. Due to space limitations, we include it in Appendix G.2 in the technical report.

We then assess the average degree, defined as the total number of edges in the subgraph divided by the product of the number

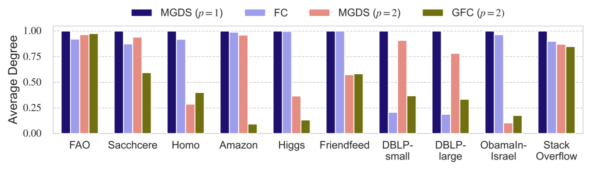


Figure 6: The average degrees of the subgraphs obtained by MGDS, FC, and GFC.

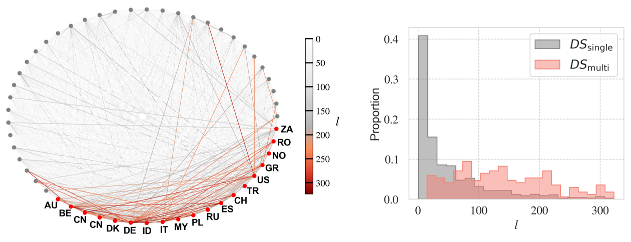


Figure 7: Comparison between the densest subgraphs identified from FAO and its flattened single-layer version.

of vertices and the number of layers. Our results are reported in Figure 6. We can see that the average degree of the subgraph output by MGDC is slightly higher than that identified by FC. We attribute this to the adoption of the q-mean degree in MGDS, which aggregates the vertex degree information from every layer, while FC disregards information it deems insignificant. The advantage of MGDC is notable on graphs DBLP-small and DBLP-large. In these cases, MGDC extracts subgraphs with exceptionally high single-layer densities of 37.2 and 143.0, respectively. In contrast, FC identifies subgraphs with average densities of 0.79 and 2.5 on selected layers. This finding suggests that the vertex-oriented density is preferable when considering overall vertex degrees across layers, while layer-oriented densities may be preferred when the vertices are required to be closely connected in specific layers. When $p = 2$, both the subgraphs found by MGDS and GFC show a decrease in the average degree across most graphs. This is because they prioritize vertices with large q-mean degrees or degrees in selected layers, neglecting the inclusion of low-degree neighbors.

Case Study on FAO. The ML graph FAO describes the worldwide food import/export relationships among countries or their subdivisions in 2010, with each layer representing transactions of a specific food product [16]. We apply our MGDS framework to the graph with $q = 1$ and $p = -\infty$ and obtain a dense ML subgraph, denoted as DS_{multi}, containing 37 vertices. Among these, 31 (83.8%) match the top-37 regions with the largest trade value in 2010 [1]. Furthermore, we extract the densest subgraph DS_{single} maximizing the p-density of the flattened single-layer version of graph FAO where vertices are connected if they are neighbors in at least one layer. DS_{single} contains a substantial number of 105 vertices covering all vertices in DS_{multi}. We depict it in Figure 7, with the vertices included in DS_{multi} highlighted in red and labeled with the standard region code. Edge colors represent the number l of layers where the edge appears. We see that edges occur on a large number of layers, representing transactions of a diverse range of products, often connecting vertices in DS_{multi}. This diversity reflects prosperous trade transactions and higher trade values. On the right side of Figure 7, we present the detailed distribution of l. S_{single} contains a large proportion of edges present in only a few layers. This suggests that the dense subgraph extracted from the flattened single-layer graph, which combines information from different layers, may not accurately reflect meaningful results.

6.3 Efficiency Evaluation

In the efficiency tests, we report the results obtained under representative settings of q and p for each solved case. Without otherwise stated, we use $\epsilon = 0.2$ for the fast implementation MGDS-Fast.

Efficiency of the Densest ML Subgraph Detection algorithms. We compare the running time of our MGDS framework with layer-oriented densest ML subgraph detection algorithms. Figure 8 depicts the results, where MGDS-C3-min corresponds to the extreme case of Case 3 for $q = p = -\infty$. We make the following observations:

(1) The greedy peeling framework exhibits distinct running time in dealing with the four studied cases. MGDS-C1 and MGDS-C2 takes notably longer running time than MGDS-C3 and MGDS-C4, especially on denser graphs like Higgs and Friendfeed. This is primarily due to Case 1 and Case 2 requiring the update of scores for a 2-hop neighborhood of the removed vertex during the peeling process. Benefiting from the linear-heap-based implementation, MGDS-C3-min consistently achieves the shortest running time.

(2) MGDS-Fast significantly outperforms MGDS, achieving an average speedup of 134.6× for Case 1 and 86.3× for Case 2. The only exception is observed on graph FAO, where MGDS-C2 runs faster than MGDS-Fast-C2. We attribute this to the large number of layers in graph FAO, which increases the computational cost for maintaining the approximate q-norm of vertices' degrees.

(3) MGDS-C1 and GFC shows comparable running time. Akin to MGDS-C1, GFC adopt a peeling process to compute the generalized FirmCore, which also involves updating the information of the 2-hop neighbors of the removed vertex. However, GFC becomes notably slower than MGDS-C1 on graphs with a larger number of layers, such as FAO and StackOverflow. This happens because GFC executes the peeling process as many times as the number of layers, while MGDS-C1 performs just a single peeling process. A similar situation can be observed for MGDS-C3 (MGDS-C4) and FC.

Effectiveness of the Fast Implementation. Figure 9 reports the running time and generalized density of the output of MGDS and MGDS-Fast. MGDS-Fast-Cx with $\epsilon = 0.1$ achieves a substantial speedup over MGDS-Cx, up to 217.4× and 301.1× on the tested graphs for cases $x = 1$ and 2, respectively. As ϵ increases from 0.1 to 0.5, the efficiency improvements become insignificant or remain consistent. Regarding the quality of the results, except for graph DBLP-Large, where the generalized density outputted by MGDS-Fast-C1 drops to under 0.85 of that obtained by MGDS-C1, the density degradation on other graphs is no more than 0.05. Moreover, it's surprising that MGDS-Fast-C2 consistently produces the same results as MGDS-C2 when increasing ϵ from 0.1 to 0.5. These observations indicate the effectiveness of MGDS-Fast and its potential to replace MGDS in practical scenarios.

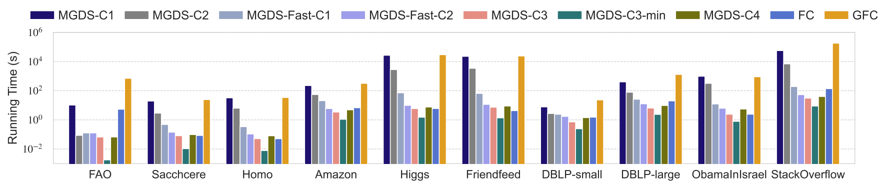


Figure 8: Running time of different densest ML subgraph detection algorithms.

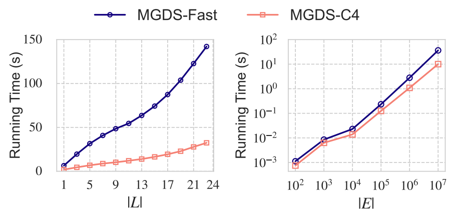
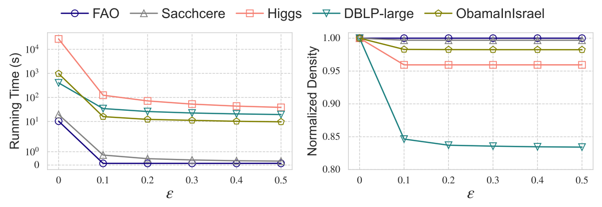
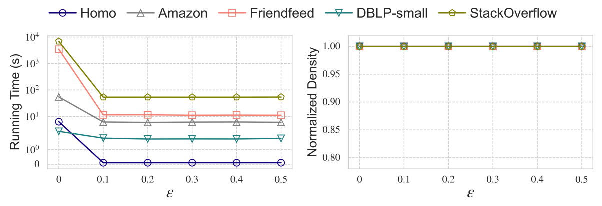


Figure 10: The running time of MGDS-Fast and MGDS-C4 on StackOverflow.



(a) MGDS-C1 V.S. MGDS-Fast-C1



(b) MGDS-C2 V.S. MGDS-Fast-C2

Figure 9: Comparison between MGDS and MGDS-Fast in terms of running time and results quality (density) for varying ϵ, where $\epsilon = 0$ corresponds to MGDS.

Scalability. We test the running time of MGDS-Fast and MGDS-C4 using different versions of graph StackOverflow obtained by selecting a variable number of layers from 1 to 24 and subsets of edges with sizes ranging from 10^2 to 10^7. For MGDS-Fast, we set $q = p = 2$, which caters to both Case 1 and Case 2. Additionally, since MGDS-C3 and MGDS-C4 share the same peeling process for a fixed q, we opt for the slightly slower MGDS-C4 as a representative. Therefore, the performance of these two tested algorithms can effectively reflect the performance of solving the studied cases of the MGDS problem in practice. We report the results in Figure 10. There is a clear trend that both algorithms scale gracefully with the increasing number of layers or edges, confirming their abilities to handle large-scale ML graphs. Notably, both FC and GFC for finding layer-oriented densest ML subgraphs scale at least quadratically with the number of layers.

7 RELATED WORK

Densest Subgraphs in Simple Graphs. Over decades of study, researchers have accumulated a wealth of density measures, algorithms for finding the densest subgraphs, and related theoretical results [20, 24, 24, 25, 29, 36, 44, 51–54]. In recent years, there has been a notable trend towards introducing general density frameworks that incorporate many existing density measures and provide solutions for finding the densest subgraphs under a family of density measures [12, 52–54].

Most of the densest subgraph problems are polynomially solvable via reductions to the maximum flow problem [25, 33, 34, 51, 54] or linear programming approaches [10, 33, 34]. However, these methods often become impractical when dealing with large-scale graphs. As a practical and efficient alternative, the greedy peeling framework is widely adopted to produce approximate solutions for maximizing a broad range of density measures [4, 8, 10, 12, 32, 51].

Densest Subgraphs in ML Graphs. We have made a detailed introduction to existing density measures [6, 11, 22, 27, 31, 50] for ML graphs and present their limitations in Section 2. There are also studies exploring finding the densest subgraphs in temporal networks [46] and uncertain graphs [56], which can be viewed as ML graphs with edges labeled by timestamps and probabilities, respectively. For further details, please see [36].

Connections with Cohesive Subgraphs. Cohesive subgraph models like k-core [49], k-truss [15], k-nucleus [48], and cliques [40] impose constraints on relationships between basic graph components of the subgraph like vertices, edges, and triangles. Most cohesive subgraphs demonstrate a hierarchical structure [55] and are typically dense, yet they do not guarantee the maximum density. The k-core model shows close connections with the densest subgraphs. In simple graphs, the k-core with the maximum k maximizes the minimum degree of vertices, equivalent to the subgraph maximizing the p-density for $p = -\infty$ [19, 53]. Furthermore, there is a significant body of research focused on developing core-based approximations for the densest subgraphs under various density measures [6, 7, 22, 27] or preprocessing approaches to expedite the exact densest subgraph search [19, 42, 54].

8 CONCLUSIONS

Existing density measures for ML graphs can be formalized into a layer-oriented density framework, which poses challenges in devising unified, scalable, and effective solutions for identifying the densest ML subgraphs. The (q, p)-density generalizes single-layer density measures in a vertex-oriented approach. It provides flexibility in adjusting the aggregation of vertex degrees across layers and the q-mean degrees of vertices, offering a general framework for capturing dense patterns of various characteristics. The vertex-oriented generalized densest ML subgraph problem can be largely solved by the greedy peeling framework with approximation guarantees and time complexity results outlined in Table 1. Extensive experiments demonstrate the effectiveness of our (q, p)-density and the proposed algorithms in uncovering diverse types of dense ML subgraphs within large-scale ML graphs.

REFERENCES

[1] [n.d.]. Food Products Exports, Imports, Tariffs by country 2010 | WITS Data — wits.worldbank.org. https://wits.worldbank.org/CountryProfile/en/Country/WLD/Year/2010/TradeFlow/EXPIMP/Partner/by-country/Product/16-24_FoodProd#. [Accessed 31-03-2024].

[2] Alberto Aleta, Sandro Meloni, and Yamir Moreno. 2017. A multilayer perspective for the analysis of urban transportation systems. *Scientific reports* 7, 1 (2017), 44359.

[3] Albert Angel, Nikos Sarkas, Nick Koudas, and Divesh Srivastava. 2012. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *Proceedings of the VLDB Endowment* 5, 6 (2012), 574–585.

[4] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. 2000. Greedily finding a dense subgraph. *Journal of algorithms* 34, 2 (2000), 203–221.

[5] Vladimir Batagelj and Matjaz Zaversnik. 2003. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[6] Ali Behrouz and Farnoosh Hashemi. 2023. Generalized Densest Subgraph in Multiplex Networks. In *International Conference on Complex Networks and Their Applications*. Springer, 49–61.

[7] Francesco Bonchi, Arijit Khan, and Lorenzo Severini. 2019. Distance-generalized core decomposition. In *proceedings of the 2019 international conference on management of data*. 1006–1023.

[8] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: Extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020*. 573–583.

[9] Lijun Chang and Lu Qin. 2018. *Cohesive subgraph computation over large sparse graphs: algorithms, data structures, and programming techniques*. Springer.

[10] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *International workshop on approximation algorithms for combinatorial optimization*. Springer, 84–95.

[11] Moses Charikar, Yonatan Naamad, and Jimmy Wu. 2018. On finding dense common subgraphs. *arXiv preprint arXiv:1802.06361* (2018).

[12] Chandra Chekuri, Kent Quanrud, and Manuel R Torres. 2022. Densest subgraph: Supermodularity, iterative peeling, and flow. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1531–1555.

[13] Chandra Chekuri and Manuel R Torres. 2023. On the generalized mean densest subgraph problem: complexity and algorithms. *arXiv preprint arXiv:2306.02172* (2023).

[14] Jie Chen and Yousef Saad. 2010. Dense subgraph extraction with application to community detection. *IEEE Transactions on knowledge and data engineering* 24, 7 (2010), 1216–1230.

[15] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008).

[16] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. 2015. Structural reducibility of multilayer networks. *Nature communications* 6, 1 (2015), 6864.

[17] Mark E Dickison, Matteo Magnani, and Luca Rossi. 2016. *Multilayer social networks*. Cambridge University Press.

[18] Yixiang Fang, Wensheng Luo, and Chenhao Ma. 2022. Densest subgraph discovery on large graphs: Applications, challenges, and techniques. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3766–3769.

[19] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks VS Lakshmanan, and Xuemin Lin. 2019. Efficient algorithms for densest subgraph discovery. *arXiv preprint arXiv:1906.00341* (2019).

[20] Uriel Feige, David Peleg, and Guy Kortsarz. 2001. The dense k-subgraph problem. *Algorithmica* 29 (2001), 410–421.

[21] Eugene Fratkin, Brian T Naughton, Douglas L Brutlag, and Serafim Batzoglou. 2006. MotifCut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics* 22, 14 (2006), e150–e157.

[22] Edoardo Galimberti, Francesco Bonchi, Francesco Gullo, and Tommaso Lanciano. 2020. Core decomposition in multilayer networks: Theory, algorithms, and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 1 (2020), 1–40.

[23] Aristides Gionis, Flavio PP Junqueira, Vincent Leroy, Marco Serafini, and Ingmar Weber. 2013. Piggybacking on social networks. In *VLDB 2013-39th International Conference on Very Large Databases*, Vol. 6. 409–420.

[24] Aristides Gionis and Charalampos E Tsourakakis. 2015. Dense subgraph discovery: Kdd 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2313–2314.

[25] Andrew V Goldberg. 1984. Finding a maximum density subgraph. (1984).

[26] Zaynab Hammoud and Frank Kramer. 2020. Multilayer networks: aspects, implementations, and application in biomedicine. *Big Data Analytics* 5, 1 (2020), 2.

[27] Farnoosh Hashemi, Ali Behrouz, and Laks VS Lakshmanan. 2022. Firmcore decomposition of multilayer networks. In *Proceedings of the ACM Web Conference 2022*. 1589–1600.

[28] Farnoosh Hashemi, Ali Behrouz, and Laks VS Lakshmanan. 2022. FirmCore Decomposition of Multilayer Networks. In *Proceedings of the ACM Web Conference 2022*. 1589–1600.

[29] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 895–904.

[30] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. 2005. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* 21, suppl_1 (2005), i213–i221.

[31] Vinay Jethava and Niko Beerenwinkel. 2015. Finding dense subgraphs in relational graphs. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II 15*. Springer, 641–654.

[32] Jiaxin Jiang, Yuan Li, Bingsheng He, Bryan Hooi, Jia Chen, and Johan Kok Zhi Kang. 2022. Spade: a real-time fraud detection framework on evolving graphs. *Proceedings of the VLDB Endowment* 16, 3 (2022), 461–469.

[33] Yasushi Kawase and Atsushi Miyauchi. 2018. The densest subgraph problem with a convex/concave size function. *Algorithmica* 80 (2018), 3461–3480.

[34] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *International colloquium on automata, languages, and programming*. Springer, 597–608.

[35] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*. 1343–1350.

[36] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. 2023. A survey on the densest subgraph problem and its variants. *Comput. Surveys* (2023).

[37] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu C Aggarwal. 2010. A Survey of Algorithms for Dense Subgraph Discovery. *Managing and mining graph data* 40 (2010), 303–336.

[38] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. 2015. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 1049–1065.

[39] Dandan Liu and Zhaonian Zou. 2023. gCore: Exploring Cross-Layer Cohesiveness in Multi-Layer Graphs. *Proceedings of the VLDB Endowment* 16, 11 (2023), 3201–3213.

[40] RD Luce and Albert D Perry. 1949. A method of matrix analysis of group structure. (1949).

[41] Dongsheng Luo, Yuchen Bian, Yaowei Yan, Xiao Liu, Jun Huan, and Xiang Zhang. 2020. Local community detection in multiple networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 266–274.

[42] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2020. Efficient algorithms for densest subgraph discovery on large directed graphs. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1051–1066.

[43] Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 228–238.

[44] Lu Qin, Rong-Hua Li, Lijun Chang, and Chengqi Zhang. 2015. Locally densest subgraph discovery. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 965–974.

[45] Polina Rozenshtein, Aris Anagnostopoulos, Aristides Gionis, and Nikolaj Tatti. 2014. Event detection in activity networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1176–1185.

[46] Polina Rozenshtein, Francesco Bonchi, Aristides Gionis, Mauro Sozio, and Nikolaj Tatti. 2020. Finding events in temporal networks: segmentation meets densest subgraph discovery. *Knowledge and Information Systems* 62, 4 (2020), 1611–1639.

[47] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. 2010. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology: 14th Annual International Conference, RECOMB 2010, Lisbon, Portugal, April 25-28, 2010. Proceedings 14*. Springer, 456–472.

[48] Ahmet Erdem Sariyuce, C Seshadhri, Ali Pinar, and Umit V Catalyurek. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the 24th International Conference on World Wide Web*. 927–937.

[49] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[50] Konstantinos Semertzidis, Evaggelia Pitoura, Evimaria Terzi, and Panayiotis Tsaparas. 2019. Finding lasting dense subgraphs. *Data Mining and Knowledge Discovery* 33 (2019), 1417–1445.

[51] Charalampos Tsourakakis. 2015. The k-clique densest subgraph problem. In *Proceedings of the 24th international conference on world wide web*. 1122–1132.

[52] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 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*. 104–112.

[53] Nate Veldt, Austin R Benson, and Jon Kleinberg. 2021. The generalized mean densest subgraph problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1604–1614.
[54] Yichen Xu, Chenhao Ma, Yixiang Fang, and Zhifeng Bao. 2023. Efficient and effective algorithms for generalized densest subgraph discovery. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.
[55] Ying Zhang, Lu Qin, Fan Zhang, and Wenjie Zhang. 2019. Hierarchical decomposition of big graphs. In *2019 IEEE 35Th International Conference on Data Engineering (ICDE)*. IEEE, 2064–2067.
[56] Zhaonian Zou. 2013. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *Proceedings of MLG Workshop*.

A PROOF OF LEMMA 4.2

Before proving Lemma 4.2, we first present the following proposition.

Proposition A.1. *Let $p \geq 1$ and $a, b, c, d \in \mathbb{R}_{0+}$. If $a \leq b, c \leq d$, and $b + c \leq a + d$, we have $b^p + c^p \leq a^p + d^p$.*

Proof. Without loss of generality, let $b \leq c$. If either $a = b$ or $c = d$ holds, the proposition is obvious. When $a < b$ and $c < d$, let $h(x) = x^p$. The mean value theorem guarantees that there exist $x \in (a, b)$ and $y \in (c, d)$ satisfying that $h'(x) = \frac{h(b)-h(a)}{b-a}$ and $h'(y) = \frac{h(d)-h(c)}{d-c}$. As $h'(x)$ is increasing when $p \geq 1$, we have $h'(x) \leq h'(y)$, that is, $\frac{b^p-a^p}{b-a} \leq \frac{d^p-c^p}{d-c}$. Because $d - c \geq b - a$, we must have $b^p + c^p \leq a^p + d^p$. □

The proof of Lemma 4.2 is given as below.

Proof. By the definition of supermodular functions (Definition 4.1), it is sufficient to prove that for any $S, T \subseteq V$

$$g(S) + g(T) \leq g(S \cap T) + g(S \cup T). \tag{9}$$

We extend the notion of vertex degrees: Given a set $R \subseteq V$, we define $d_v(R, l) = 0$ if $v \notin R$. Then, $g(S)$ can be rewritten as

$$g(S) = \sum_{v \in V} \left(\sum_{l \in L} d_v(S, l)^q \right)^{p/q}.$$

Let E_l be the set of edges adjacent to v on the lth layer. We define

$$\begin{aligned} A_l &= \{(i, j) \in E_l | i \in S, j \in S\}, \\ B_l &= \{(i, j) \in E_l | i \in T, j \in T\}, \\ C_l &= \{(i, j) \in E_l | i \in S \cap T, j \in S \cap T\}, \\ D_l &= \{(i, j) \in E_l | i \in S \cup T, j \in S \cup T\}. \end{aligned}$$

It follows that

$$\begin{aligned} d_v(S, l) &= |A_l|, \\ d_v(T, l) &= |B_l|, \\ d_v(S \cap T, l) &= |C_l| = |A_l \cap B_l|, \\ d_v(S \cup T, l) &= |D_l| \geq |A_l \cup B_l|. \end{aligned}$$

Let $\Omega(R) = |R|^q$. When $q \geq 1$, $\Omega(\cdot)$ is supermodular. Therefore,

$$\begin{aligned} d_v(S, l)^q + d_v(T, l)^q &= |A_l|^q + |B_l|^q = \Omega(A_l) + \Omega(B_l) \\ &\leq \Omega(A_l \cap B_l) + \Omega(A_l \cup B_l) \leq |C_l|^q + |D_l|^q \\ &= d_v(S \cap T, l)^q + d_v(S \cup T, l)^q. \end{aligned}$$

By summing the above inequality for all $l \in L$, we have

$$\sum_{l \in L} d_v(S, l)^q + \sum_{l \in L} d_v(T, l)^q \leq \sum_{l \in L} d_v(S \cap T, l)^q + \sum_{l \in L} d_v(S \cup T, l)^q.$$

In addition, due to the definition of $d_v(\cdot)$ and the monotonically increasing nature of the power function $h(x) = x^q$ for $x \in \mathbb{R}_{0+}$ and $q \geq 1$, we easily have

$$\sum_{l \in L} d_v(S \cap T, l)^q \leq \sum_{l \in L} d_v(S, l)^q, \quad \sum_{l \in L} d_v(T, l)^q \quad \leq \sum_{l \in L} d_v(S \cup T, l)^q.$$

As $p/q \geq 1$, we can establish the following inequality by applying Proposition A.1.

$$\left(\sum_{l \in L} d_v(S, l)^q \right)^{p/q} + \left(\sum_{l \in L} d_v(T, l)^q \right)^{p/q} \leq \left(\sum_{l \in L} d_v(S \cap T, l)^q \right)^{p/q} + \left(\sum_{l \in L} d_v(S \cup T, l)^q \right)^{p/q}.$$

By summing the inequality for all $v \in V$, we have the desired result in Eq. (9). Thus, the lemma holds. □

B PROOF OF LEMMA 4.6

Proof. When $h = g_{q,p}$, c_h can be rewritten as

$$c_h = \max_{S \subseteq V} \frac{\sum_{v \in S} \Delta_{q,p}(v, S)}{g_{q,p}(S)}. \tag{10}$$

We will prove $c_h \leq p + 1$ by showing that $\sum_{v \in S} \Delta_{q,p}(v, S) \leq (p + 1) g_{q,p}(S)$ for all $S \subseteq V$.

For simplicity, for all $S \subseteq V$, let

$$E_v(S) = \sum_{l \in L} d_v(S, l)^q, \qquad E_{v,u}(S) = \sum_{l \in L} \left(d_u(S, l) - \delta_{u,v,l} \right)^q,$$

where $\delta_{u,v,l} = 1$ if v and u are adjacent on layer l, and $\delta_{u,v,l} = 0$ otherwise. From the definition of $g_{q,p}(S)$ and $\Delta_{q,p}(v, S)$, we have

$$\begin{aligned} g_{q,p}(S) &= \sum_{v \in S} E_v(S)^{p/q}, \\ \Delta_{q,p}(v, S) &= E_v(S)^{p/q} + \sum_{u \in N_v(S)} E_u(S)^{p/q} - E_{v,u}(S)^{p/q}. \end{aligned}$$

Therefore,

$$\begin{aligned} \sum_{v \in S} \Delta_{q,p}(v, S) &= \sum_{v \in S} \left(E_v(S)^{p/q} + \sum_{u \in N_v(S)} E_u(S)^{p/q} - E_{v,u}(S)^{p/q} \right) \\ &= g_{q,p}(S) + \sum_{v \in S} \sum_{u \in N_v(S)} E_u(S)^{p/q} - E_{v,u}(S)^{p/q}. \end{aligned} \tag{11}$$

Next, we will show that the second term in Eq. (11) does not exceed $p \cdot g_{q,p}(S)$. As $q \geq 1$, the following inequality follows from Proposition 4.7 for any $l \in L$:

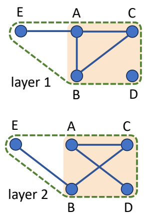
$$d_u(S, l)^q - \left(d_u(S, l) - \delta_{u,v,l} \right)^q \leq q \cdot \delta_{u,v,l} \cdot d_u(S, l)^{q-1}. \tag{12}$$

Summing Eq. (12) over all $l \in L$ establishes that

$$E_u(S) - E_{v,u}(S) \leq q \sum_{l \in L} \delta_{u,v,l} \cdot d_u(S, l)^{q-1}.$$

Because $p/q \geq 1$, it follows from Proposition 4.7 that

$$\begin{aligned} E_u(S)^{p/q} - E_{v,u}(S)^{p/q} &\leq \frac{p}{q} \left(E_u(S) - E_{v,u}(S) \right) E_u(S)^{p/q-1} \\ &\leq p \left(\sum_{l \in L} \delta_{u,v,l} \cdot d_u(S, l)^{q-1} \right) E_u(S)^{p/q-1} \\ &= p \cdot E_u(S)^{p/q-1} \left(\sum_{l \in L} \delta_{u,v,l} \cdot d_u(S, l)^{q-1} \right). \end{aligned}$$



Vertex	$S = \{A, B, C, D\}$			$S' = S - \{D\}$			$T = \{A, B, C, D, E\}$			$T' = T - \{D\}$		
	$d_v(S,1)$	$d_v(S,2)$	$\|\mathbf{d}_v(S)\|_3^2$	$d_v(S',1)$	$d_v(S',2)$	$\|\mathbf{d}_v(S')\|_3^2$	$d_v(T,1)$	$d_v(T,2)$	$\|\mathbf{d}_v(T)\|_3^2$	$d_v(T',1)$	$d_v(T',2)$	$\|\mathbf{d}_v(T')\|_3^2$
A	2	2	6.350	2	1	4.327	3	2	10.700	3	1	9.221
B	2	1	4.327	2	1	4.327	2	2	6.350	2	2	6.350
C	2	2	6.350	2	2	6.350	2	2	6.350	2	2	6.350
D	0	1	1.000	–	–		0	1	1.000	–	–	
E	–	–	–	–	–		1	1	1.587	1	1	1.587

Figure 10: A counter-example demonstrating that the function $g_{q,p}(\cdot)$ is not supermodular in the case when $q \geq p \geq 1$.

Finally, we have

$$\begin{aligned}
&\sum_{v\in S}\sum_{u\in N_v(S)} E_u(S)^{p/q} - E_{v,u}(S)^{p/q} \\
\leq &\sum_{v\in S}\sum_{u\in N_v(S)} p\cdot E_u(S)^{p/q-1}\left(\sum_{l\in L}\delta_{u,v,l}\cdot d_u(S,l)^{q-1}\right) \\
=&p\sum_{u\in S}\sum_{v\in N_u(S)} E_u(S)^{p/q-1}\left(\sum_{l\in L}\delta_{u,v,l}\cdot d_u(S,l)^{q-1}\right) \\
=&p\sum_{u\in S} E_u(S)^{p/q-1}\sum_{l\in L}\sum_{v\in N_u(S,l)} d_u(S,l)^{q-1} \\
=&p\sum_{u\in S} E_u(S)^{p/q-1}\sum_{l\in L} d_u(S,l)^{q} \\
=&p\sum_{u\in S} E_u(S)^{p/q} \\
=&p\cdot g_{q,p}(S).
\end{aligned} \tag{13}$$

By Eq. (10), (11) and (13), we have $c_h \leq p+1$. □

C THE FUNCTION $g_{q,p}(\cdot)$ IS NOT SUPERMODULAR WHEN $q \geq p \geq 1$

Figure 10 illustrates a counter-example demonstrating that $g_{q,p}(\cdot)$ is not supermodular for $q \geq p \geq 1$. Given the ML graph on the left, let $S = \{A, B, C, D\}$, $T = \{A, B, C, D, E\}$, $q = 3$, and $p = 2$. The key components for computing $g_{q,p}(S)$, $g_{q,p}(T)$, $g_{q,p}(S - \{D\})$, and $g_{q,p}(T - \{D\})$ are listed in the table on the right. Here, we represent $g_{q,p}(S)$ in the form of generalized norms, that is, $g_{q,p}(S) = \sum_{v\in S}\|\mathbf{d}_v(S)\|_q^p$. We have

$$\begin{aligned}
g_{q,p}(S) &= \sum_{v\in S}\|\mathbf{d}_v(S)\|_q^p = 18.026, \\
g_{q,p}(T) &= \sum_{v\in T}\|\mathbf{d}_v(T)\|_q^p = 25.986, \\
g_{q,p}(S-\{D\}) &= \sum_{v\in S-\{D\}}\|\mathbf{d}_v(S-\{D\})\|_q^p = 15.003, \\
g_{q,p}(T-\{D\}) &= \sum_{v\in T-\{D\}}\|\mathbf{d}_v(T-\{D\})\|_q^p = 23.507.
\end{aligned}$$

Therefore, $g_{q,p}(S) - g_{q,p}(S-\{D\}) > g_{q,p}(T) - g_{q,p}(T-\{D\})$, which violates the definition of supermodular functions.

D PROOF OF LEMMA 4.12

Proof. Apparently, $\rho_{q,-\infty}(S) = |L|^{-1/q}\min\|\mathbf{d}_v(S)\|_q$ for $S \subseteq V$. Since $S^*_{q,-\infty}$ maximizes $\rho_{q,-\infty}(S)$ among all subsets $S \subseteq V$, we have

$$\begin{aligned}
\rho_{q,-\infty}(S^*_{q,-\infty}) &\geq \max_{S\subseteq V}\rho_{q,-\infty}(S) \geq \rho_{q,-\infty}(S^*_{q,1}) \\
&= |L|^{-1/q}\min_{v\in S^*_{q,1}}\|\mathbf{d}_v(S^*_{q,1})\|_q.
\end{aligned}$$

From the proof for Theorem 4.11, we can easily have

$$f_{q,1}(S^*_{q,1}) \leq \Delta_{q,p}(v, S^*_{q,1}) \leq \left(1+|L|^{1-\frac{1}{q}}\right)\|\mathbf{d}_v(S^*_{q,1})\|_q$$

for any $v \in S^*_{q,1}$. Then,

$$f_{q,1}(S^*_{q,1}) \leq \left(1+|L|^{1-\frac{1}{q}}\right)\min_{v\in S^*_{q,1}}\|\mathbf{d}_v(S^*_{q,1})\|_q.$$

Finally,

$$\begin{aligned}
\rho_{q,1}(S^*_{q,1}) &= |L|^{-1/q} f_{q,1}(S^*_{q,1}) \\
&\leq |L|^{-1/q}\left(1+|L|^{1-\frac{1}{q}}\right)\min_{v\in S^*_{q,1}}\|\mathbf{d}_v(S^*_{q,1})\|_q \\
&\leq \left(1+|L|^{1-\frac{1}{q}}\right)\rho_{q,-\infty}(S^*_{q,-\infty}).
\end{aligned}$$

Consequently, the lemma holds. □

E PROOF OF THEOREM 5.1

Proof. The theorem can be proved by slightly modifying the proof for Theorem 4.9. Specifically, we maintain the inequality

$$\hat{\Delta}_{q,p}(v,S) \leq \widetilde{\Delta}_{q,p}(v,S) \leq (1+2\epsilon)\hat{\Delta}_{q,p}(v,S). \tag{14}$$

Then, Eq. 6 can be refined as follows:

$$f_{q,p}(S^*) \leq \hat{\Delta}_{q,p}(v,S) \leq \hat{\Delta}_{q,p}(v,T) \leq (1+2\epsilon)\frac{1}{|T|}\sum_{u\in T}\hat{\Delta}_{q,p}(u,T),$$

where v is greedily selected in T that minimizes $\widetilde{\Delta}q,p(v,T)$. The last inequality holds because

$$\begin{aligned}
\hat{\Delta}_{q,p}(v,T) \leq \widetilde{\Delta}_{q,p}(v,T) &\leq \frac{1}{|T|}\sum_{u\in T}\widetilde{\Delta}_{q,p}(u,T) \\
&\leq (1+2\epsilon)\frac{1}{|T|}\sum_{u\in T}\hat{\Delta}_{q,p}(u,T).
\end{aligned}$$

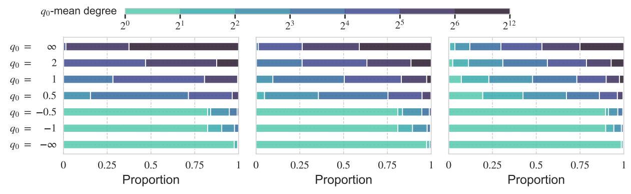
Further, according to Eq. 8, we have

$$f_{q,p}(S^*) \leq (1+2\epsilon)\left(1+p|L|^{1-\frac{1}{q}}\right)f_{q,p}(T),$$

and therefore $\rho_{q,p}(S^*) \leq \left((1+2\epsilon)\left(1+p|L|^{1-\frac{1}{q}}\right)\right)^{1/p}\rho_{q,p}(T)$. Thus, the theorem holds. □

Table 5: Settings for the Efficiency Tests.

Case	Tested Settings
Case 1 ($1 \le q \le p$)	$q = 1, p = 2$
Case 2 ($1 \le p \le q$)	$q = 2, p = 1$
Case 3 ($p = -\infty$)	$q = 1, p = -\infty$
Case 4 ($p \le 1 \le q$)	$q = 1, p = -1$



(a) MGDS. **(b) FC.** **(c) GFC.**

Figure 11: Distribution of the q_0-mean degrees of vertices in the subgraphs output by MGDS, FC, and GFC on Sacchcere.

F PROOF OF LEMMA 5.2

Proof. For simplicity, let $d = \|\mathbf{d}_u(S)\|_q$. Consider the function $h(x) = \frac{(\xi(x)\cdot d)^{p-1}}{e^x}$, where $x \in [0,1]$ and $\xi(x) = 1 + \frac{x}{p-1}$. We have

$$h'(x) = \frac{\left(\frac{1}{\xi(x)} - 1\right)(\xi(x)\cdot d)^{p-1}}{e^x} \le 0$$

for all $x \in [0,1]$. Thus, $h(x)$ is decreasing, so $h(\epsilon) \le h(0) = d^{p-1}$. It follows that $(\gamma \cdot d)^{p-1} \le e^{\epsilon} d^{p-1} \le (1+2\epsilon)d^{p-1}$, where the last inequality is due to the fact that $e^{\epsilon} \le 1 + 2\epsilon$. Thus, the lemma holds. □

G SUPPLEMENTS TO THE EXPERIMENTS

G.1 Settings for the Efficiency Tests.

Table 5 outlines the tested settings used in the efficiency tests.

G.2 A Supplementary Comparison of Vertex-oriented and Layer-oriented Densities

Following the settings in Section 6.2, we present the distributions of the q_0-mean degrees of vertices in the dense subgraphs output by MGDS, FC, and GFC on graph Sacchcere in Figure 11. The subgraphs obtained by MGDS and FC exhibit similarities: they prioritize vertices with high q_0-mean degrees for large q_0, while as q_0 approaches $-\infty$, the q_0-mean degrees of almost all vertices decrease to less than 5. This observation aligns with the findings in Tables 4 that these two subgraphs exhibit large densities on four layers. For larger q_0 values, the q_0-mean degree captures the higher vertex degrees from these layers, while for smaller q_0 values, the lower vertex degrees from the noisy layers dominate. Additionally, the q_0-mean degrees of vertices in the subgraph obtained by GFC tend to be smaller compared to the others.