Density Decomposition of Multilayer Graphs

Jiaqi Jiang, Rong-Hua Li, Yalong Zhang

Beijing Institute of Technology, Beijing, China;
jiaqijiang@bit.edu.cn; lironghuabit@126.com; yalong-zhang@qq.com

***Abstract*—Multilayer graphs have emerged as a powerful model for representing complex systems with diverse types of interactions. Identifying cohesive subgraphs in such graphs is a fundamental task with broad applications in community detection, fraud analysis, and e-commerce recommendation. However, existing models either lack explicit density guarantees or fail to preserve across-layer structural cohesiveness. To address these limitations, in this paper, we propose a novel k-dense subgraph model, denoted as $D_{\mathbf{k}}$, which enforces layer-wise density constraints. Given a density vector $\mathbf{k} = [k_\ell]_{\ell \in L} \in \mathbb{Z}^{|L|}$, our model guarantees that the induced subgraph on each layer satisfies a minimum density threshold k_ℓ.**

We further study the density decomposition problem: computing all non-empty $D_{\mathbf{k}}$s for feasible density vectors k. To this end, we design a suite of efficient algorithms. For computing a single $D_{\mathbf{k}}$, we first propose SLFBE, which uses network flow techniques to extract layer-wise subgraphs satisfying density thresholds. We then develop LCVSP, which combines k-core pruning and layer-constrained vertex set propagation to eliminate irrelevant vertices and accelerate convergence. For full density decomposition, we leverage the density lattice structure and propose HPDD, which incrementally constructs subgraphs by extending partial vectors layer by layer, enabling localized exploration and early pruning. To further improve scalability, we develop the HPDD+ algorithm, which adopts a recursive depth-first hierarchical strategy to lower memory cost. Additionally, HPDD+ employs an adaptive upper-bounded divide-and-conquer approach that eliminates redundant computations. Extensive experiments on 9 real-world multilayer graphs demonstrate that our model discovers significantly higher-quality subgraphs than existing methods, while our algorithms exhibit superior efficiency and scalability.

I. Introduction

Multilayer graphs have emerged as a powerful abstraction for modeling complex interconnected systems where entities interact through multiple types of relationships. Examples span diverse domains, including social networks [1], [2], biological systems [3], [4], and financial transaction graphs [5], where each layer captures a different type of interaction or context. Therefore, mining cohesive subgraphs in these structures is a fundamental problem in acdemic research. In these settings, discovering cohesive subgraphs, defined as groups of entities that are densely connected across multiple layers, is a fundamental problem with broad applications in community detection [6], fraud analysis [7], [8], and urban transportation analysis [9].

To tackle this important task, several multilayer cohesive subgraph models have been proposed. The multilayer **k**-core with $\mathbf{k} = [k_1, k_2, \cdots, k_{|L|}]$ [10] enforces layer-wise degree thresholds, requiring that each node has at least k_ℓ neighbors in each layer ℓ. The d-CC model [11] generalizes this by enforcing a uniform degree threshold d over a specified subset of layers. FirmCore [12] introduces flexibility by requiring each node to satisfy the k-degree constraint in at least λ arbitrary layers. FocusCore [13] extends this further, combining strict layer constraints on a focus set $\mathcal{F}$ with loose support across λ layers. FirmTruss [14] imposes truss-based constraints, requiring adjacent vertices to participate in at least $k-2$ common triangles across multiple layers.

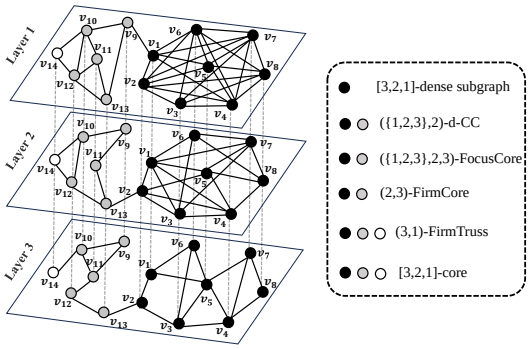


Fig. 1. A toy multilayer graph with 3 layers and illustration of existing models

However, these models often exhibit two critical limitations: (1) **Lack of layer-wise density guarantees.** Existing models rely primarily on degree-based (triangle-based) constraints, which may not guarantee layer-wise minimum density in a strict sense. As a result, even if the output subgraph appears structurally cohesive, it may still lack sufficient edge density. (2) **Lack of cross-layer structural cohesiveness.** Most models either apply uniform degree thresholds (*i.e.*, k) across layers or relax constraints to only a subset of layers. As a result, the extracted subgraphs tend to be cohesive in a few layers but loose or disconnected in others, lacking balanced structural consistency across the multilayer graph.

To address these limitations, we propose a novel **k**-dense subgraph model for multilayer graphs, denoted as $D_{\mathbf{k}}$, which enforces layer-wise density lower bound $\rho(G_\ell[S]) = \frac{E_\ell[S]}{V_\ell[S]} > k_\ell$ for each layer ℓ. Compared to prior models, $D_{\mathbf{k}}$ more accurately captures dense communities across all layers. For example, as illustrated in Fig. 1, the community extracted by $D_{\mathbf{k}}$ that is densely connected in every layer, which achieves density of 3.12 (Layer 1), 2.37 (Layer 2), and 1.62 (Layer 3). In comparison, existing models produce communities that include sparse regions, leading to significantly lower density. For example, the communities discovered by d-CC, FocusCore, and FirmCore yield density of 2.53, 1.92, 1.46 across three layers, respectively, while **k**-core and FirmTruss produce even sparser results with density of 2.50, 1.92, 1.42.

The ability of $D_{\mathbf{k}}$ to identify dense communities across multiple layers enables impactful real-world applications. In social networks, such cross-platform cohesive groups can improve friend recommendation accuracy by capturing users with strong connections and interests. They also enhance audience targeting for advertising and content promotion, as users in multilayer-dense communities tend to exhibit higher engagement. In financial networks, $D_{\mathbf{k}}$ can uncover groups—such

as shell companies or coordinated fraud rings—that exhibit consistently dense transactional relationships across different banks, subsidiaries, or time windows, which is a strong indicator of covert or organized activity.

While the $D_{\mathbf{k}}$ provides a rigorous way to capture multilayer cohesion, each $D_{\mathbf{k}}$ represents only a single density configuration. In real multilayer systems, cohesive patterns often vary across layers and density scales. To obtain a holistic view of how these patterns evolve and interact, we aim to decompose the multilayer graph into all non-empty $D_{\mathbf{k}}$s, leading to the **density decomposition of multilayer graphs** problem.

Given a graph G with L layers, our goal is to compute all non-empty $D_{\mathbf{k}}$s for all feasible density vectors $\mathbf{k}$. This problem poses two key challenges. First, density constraints are intrinsically non-local. Degree-based models can be solved by simple local peeling: repeatedly delete vertices whose degree falls below the threshold. By contrast, density constraints are global. Vertices removals in one layer may cause density violations in others, triggering cascading updates. Thus, computing $D_{\mathbf{k}}$ demands global coordination rather than local peeling. Second, the number of feasible density vectors grows exponentially with the number of layers, leading to a combinatorial search space of size $O(\prod_{\ell=1}^{L}\lceil\rho^*(G_\ell)\rceil)$, where $\rho^*(G_\ell)$ is the maximum achievable density in layer ℓ. Exhaustive enumeration is infeasible, calling for principled strategies to prune infeasible paths and avoid the redundant computation. To tackle these challenges, we propose a suite of efficient algorithms for $D_{\mathbf{k}}$ computation and density decomposition.

For the first challenge, we first propose the SLFBE algorithm, which utilizes network flow techniques to identify subgraphs with density greater than k_ℓ in each layer ℓ. Building upon this, we introduce a basic search algorithm, IMLI, that computes $D_{\mathbf{k}}$ for a given density vector $\mathbf{k}$ using an iterative intersection process. However, IMLI suffers from inefficiency. To overcome this, we develop the LCVSP framework, which incorporates a layer-constrained vertex set propagation mechanism and a $\mathbf{k}$-core pruning strategy. The $\mathbf{k}$-core pruning efficiently eliminates irrelevant vertices, and the propagation mechanism further reduces the number of iterations, significantly improving overall performance.

For the second challenge, we leverage the structural properties of the density lattice to guide the search and reduce unnecessary computations. We first present the VSHT algorithm, which explores the density lattice in a breadth-first manner and processes all density vectors at each level. While effective, this approach incurs substantial redundancy. To address this, we propose the HPDD algorithm, which incrementally constructs multilayer dense subgraphs by extending partial density vectors layer by layer. This hierarchical strategy enables localized exploration and early pruning, greatly reducing the search space and computational cost. Furthermore, to improve the efficiency and reduce the memory cost of HPDD, we introduce HPDD+, which integrates a recursive depth-first search and a divide-and-conquer approach underpinned by a novel adaptive upper bound. This mechanism adaptively limits the search space, eliminating redundant computations. Besides, this approach reduces the space complexity from exponential to linear in the number of layers (as detailed in Lemma 11).

Our main contributions are summarized as follows.

- We propose a novel density-based model, $\mathbf{k}$-dense subgraph ($D_{\mathbf{k}}$), which guarantees minimum density in each layer.
- We develop LCVSP, an efficient algorithm for computing $D_{\mathbf{k}}$, which combines $\mathbf{k}$-core-based pruning with a layer-constrained vertex set propagation strategy to accelerate convergence and reduce unnecessary computation.
- We advance multilayer density decomposition by proposing a hierarchical progressive approach and developing a efficient HPDD algorithm, along with HPDD+ algorithm, which achieves scalable and memory-optimal computation with linear space complexity.
- We conduct extensive experiments on 9 real-world multilayer graphs, showing that our model consistently discovers higher-quality subgraphs compared to other models, and our algorithms achieve superior efficiency and scalability.

Due to the space limits, all the missing proofs can be found in the full version of this paper [15].

II. Preliminaries

We consider an undirected multilayer graph $G = (V, E, L)$, where V is a set of vertices, L is a set of layers, and $E \subseteq V \times V \times L$ is a set of edges. For a given layer $\ell \in L$, let $E_\ell = \{(u, v, \ell) \in E\}$ be the set of edges on layer ℓ, and let the corresponding layer-specific graph be $G_\ell = (V, E_\ell)$. The degree of a vertex $u \in V$ in layer ℓ is denoted by $d_\ell(u) = |\{(u, v, \ell) \in E_\ell\}|$. Let $\mu(\ell)$ and $\mu(\hat{L})$ denote the minimum degree of a vertex in layer ℓ and over a subset of layers $\hat{L} \subseteq L$, respectively. For a vertex set S, let $\mu(S, \ell)$ and $\mu(S, \hat{L})$ denote the corresponding values in the subgraph induced by S.

Given a node set $S \subseteq V$, the subgraph of G induced by S over all layers is denoted by $G[S] = (S, E[S], L)$, where $E[S] = \{(u, v, \ell) \in E : u \in S, v \in S\}$. For a specified subset of layers $\hat{L} = \{1, 2, \ldots, \ell\} \subseteq L$, the induced multilayer subgraph on S restricted to $\hat{L}$ is denoted by $G_{\hat{L}}[S] = (S, E_{\hat{L}}[S], \hat{L})$. For each layer $\ell \in L$, the layer-specific induced subgraph by S is $G_\ell[S] = (S, E_\ell[S])$, and the degree of $u \in S$ in layer ℓ is $d_\ell^S(u)$. For two disjoint vertex sets X and Y satisfying $X \cap Y = \emptyset$, we define the *cross edge* in layer ℓ between X and Y as $E_\times(X, Y, \ell) = \{(x, y) \in E_\ell \mid x \in X,\, y \in Y\}$, capturing edges that connect the two sets.

An orientation of the multilayer graph G is represented as $\vec{G} = (V, \vec{E}, L)$, where each edge in E is assigned a direction. For each layer $\ell \in L$, the corresponding directed graph is $\vec{G}_\ell = (V, \vec{E}_\ell)$, where each directed edge $\langle u, v\rangle \in \vec{E}_\ell$ represents an edge from u to v. The outdegree of a vertex $u \in V$ in layer ℓ is $\vec{d}_\ell(u) = |\{v \mid \langle u, v\rangle \in \vec{E}_\ell\}|$. Similarly, the induced subgraph on layer ℓ with vertex set S is $\vec{G}_\ell[S] = (S, \vec{E}_\ell[S])$, and the outdegree of $u \in S$ within this subgraph is $\vec{d}_\ell^S(u)$. A directed path from s to t in layer ℓ, denoted $s \rightsquigarrow t$, is a sequence of vertices $s = x_0, x_1, \ldots, x_k = t$, such that $\langle x_{i-1}, x_i\rangle \in \vec{E}_\ell$ for all $i = 1, \ldots, k$.

Following the classical definition of density [16], the density of a graph G is defined as $\rho(G) = \frac{|E[G]|}{|V[G]|}$. Let $\rho^*(G)$ denote the density of the densest subgraph within the graph G.

While classical graph density captures cohesiveness in single-layer graphs, real-world networks often involve multiple types of interactions, naturally represented as multilayer graphs. We thus generalize density to this setting via a layer-wise density vector $\mathbf{k} = [k_\ell]_{\ell \in L} \in \mathbb{Z}^{|L|}$, where k_ℓ denotes

the minimum density in layer ℓ, leading to the definition of a multilayer dense subgraph (MDS).

Definition 1: (Multilayer Dense Subgraph (MDS) and Density Vector) Given a multilayer graph $G = (V, E, L)$ and a density vector $\mathbf{k} = [k_\ell]_{\ell \in L} \in \mathbb{Z}$, let $\vec{G}$ be an orientation of G, where for each layer $\ell \in L$, the oriented graph is denoted by $\vec{G}_\ell = (V, \vec{E}_\ell)$. For each layer $\ell \in L$, define the vertex sets:

$$S_\ell = \left\{ u \in V \mid \vec{d}_\ell(u) > k_\ell \right\}, \quad T_\ell = \left\{ u \in V \mid \vec{d}_\ell(u) < k_\ell \right\}.$$

If, for all $\ell \in L$, there exists no path $s \rightsquigarrow t$ in $\vec{G}_\ell$ with $s \in S_\ell$ and $t \in T_\ell$, define the corresponding dense region as $D_\ell = S_\ell \cup \{v \in V \mid \exists u \in S_\ell,\ u \rightsquigarrow v \text{ in } \vec{G}_\ell\}$.

The multilayer $\mathbf{k}$-dense subgraph is defined as the MDS $G[D] = (D \subseteq V, E[D], L)$ such that $D = D_\ell$ for all $\ell \in L$; that is, D induces the same vertex set across all layers.

For brevity, given a density vector $\mathbf{k}$, $D_\mathbf{k}$ refers to either the $\mathbf{k}$-dense subgraph $G[D]$ or its vertex set D. This definition leads to the following observations:

Observation 1: For every $\ell \in L$, the $D_\mathbf{k}$ satisfies:

(1) **Boundary orientation:** All edges between $V \setminus D_\mathbf{k}$ and $D_\mathbf{k}$ in $\vec{G}_\ell$ are oriented towards $D_\mathbf{k}$, *e.g.*, $E_\times(D_\mathbf{k}, V \setminus D_\mathbf{k}, \ell) \subseteq \vec{E}_\ell$.

(2) **Minimum outdegree guarantee:** $\forall u \in D_\mathbf{k}$, $\vec{d}^{G}_\ell(u) = \vec{d}^{D_\mathbf{k}}_\ell(u) \geq k_\ell$.

(3) **External boundedness:** $\forall u \in V \setminus D_\mathbf{k}$, $\vec{d}^{G}_\ell(u) \leq k_\ell$.

Motivation and design rationale. As summarized in Observation 1, the multilayer $\mathbf{k}$-dense subgraph exhibits a clear separation between vertices inside and outside the dense region. To preserve this equilibrium, our model adopts an *orientation constraint*, which directs all boundary edges inward to ensure local stability, and a *no $S \to T$ path condition*, which prevents weakly connected nodes from invading dense regions. These constraints jointly guarantee that the induced subgraph remains internally dense and externally sparse across all layers. This mechanism guarantees that every vertex in $D_\mathbf{k}$ maintains sufficient connections within the subgraph, which directly leads to a provable density lower bound on each layer (Theorem 1), while preventing weakly connected vertices from being absorbed into the dense region. Consequently, $D_\mathbf{k}$ achieves layer-wise density guarantees and ensures a stable and uniquely defined multilayer structure (Theorem 2).

Theorem 1: Let $G = (V, E, L)$ be a multilayer graph, and let $\mathbf{k} = [k_\ell]_{\ell \in L} \in \mathbb{Z}^{|L|}$ be a density vector. Suppose $D_\mathbf{k}$ is induced by vertex set D according to Definition 1, satisfying $D = D_\ell$ for all $\ell \in L$. Then, for each $\ell \in L$, the following density inequality holds that $\rho(G_\ell[D]) = \frac{|E_\ell[D]|}{|D|} > k_\ell$.

Proof: Fix a layer $\ell \in L$, and consider the orientation $\vec{G}_\ell = (V, \vec{E}_\ell)$. Let $G_\ell[D] = (D, E_\ell[D])$ be the subgraph induced by D. Partition D into two subsets: $S_{k_\ell+1} = \{u \in D : \vec{d}^{S}_\ell(u) \geq k_\ell + 1\}$, $S_{k_\ell} = \{u \in D : \vec{d}^{S}_\ell(u) = k_\ell\}$. $|S_{k_\ell+1}| > 0$ as there exists at least one vertex u with $\vec{d}^{D}_\ell(u) > k_\ell$.

Given that $|S_{k_\ell+1}| > 0$, we can establish:

$$\rho(G_\ell[D]) = \frac{|E_\ell[D]|}{|D|} \geq \frac{(k_\ell+1)|S_{k_\ell+1}| + (k_\ell)|S_{k_\ell}|}{|S_{k_\ell+1}| + |S_{k_\ell}|} > \frac{(k_\ell)(|S_{k_\ell+1}| + |S_{k_\ell}|)}{|S_{k_\ell+1}| + |S_{k_\ell}|} = k_\ell$$

We can observe that $\rho(G_\ell[D]) > k_\ell$. □

While the density guarantee confirms the tightness of our model, theoretical soundness further requires the uniqueness of the MDS. We now show that each $D_\mathbf{k}$ is uniquely determined by the density vector $\mathbf{k}$.

Theorem 2: Given a multilayer graph G and a density vector $\mathbf{k}$, the $D_\mathbf{k}$ is unique.

Proof: We prove by contradiction. Suppose there exist two distinct $D_\mathbf{k}$s of G, denoted by D_1 and D_2, such that $D_1 \neq D_2$. Define the non-empty set $D = D_1 \setminus D_2$, so that $D \subseteq D_1$ and $D \cap D_2 = \emptyset$. Consequently, we have $|E_\times(D, D_2, \ell)| > |E_\times(D, D_1 \setminus D, \ell)|$.

On the one hand, by the Observation 1 (2), the total outdegree from D in D_1 satisfies $\sum_{u \in D} \vec{d}^{D_1}_\ell(u) \geq k_\ell \cdot |D|$. To compute the minimum of total consists of the internal edges of D and the cross edges from D to $D_1 \setminus D$, we assume that each vertex $u \in D$ has outdegree exactly k_ℓ in D_1, then by Definition 1, there must exist at least one directed edge from a vertex in $D_1 \setminus D$ to a vertex in D, which implies $|E_\ell[D]| + |E_\times(D, D_1 \setminus D, \ell)| > \sum_{u \in D} \vec{d}^{G}_\ell(u) = \sum_{u \in D} \vec{d}^{D_1}_\ell(u) = k_\ell \cdot |D|$. On the other hand, according to the Observation 1 (1) and (3), it implies that the cross edges from D to D_2 satisfy $|E_\ell[D]| + |E_\times(D, D_2, \ell)| \leq \sum_{u \in D} \vec{d}^{G}_\ell(u) = k_\ell \cdot |D|$. Combining these results, we derive a contradiction: $k_\ell \cdot |D| < |E_\ell[D]| + |E_\times(D, D_1 \setminus D, \ell)| < |E_\ell[D]| + |E_\times(D, D_2, \ell)| \leq k_\ell \cdot |D|$. Therefore, it follows that $D_1 = D_2$, which implies the uniqueness of $D_\mathbf{k}$. □

Although each density vector $\mathbf{k}$ uniquely determines a $D_\mathbf{k}$, the reverse is not necessarily true. The same vertex set $D \subseteq V$ may satisfy the density conditions for multiple different density vectors. For example, the subgraph induced by vertices $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ in Fig. 1 forms an MDS under each of the following density vectors, *i.e.*, $[3, 0, 0]$, $[3, 1, 0]$, $[3, 0, 1]$, $[3, 1, 1]$, $[3, 2, 0]$ and $[3, 2, 1]$. Among these, the density vector $[3, 2, 1]$ is not dominated by any other, denoted as the maximal density vector.

Definition 2: (Maximal Density Vector) Let $G = (V, E, L)$ be a multilayer graph, and let $D \subseteq V$ be a vertex set such that $D = D_\mathbf{k}$ for some density vector $\mathbf{k} = [k_\ell]_{\ell \in L}$. We say that $\mathbf{k}$ is a *maximal density vector* of D if there does not exist another density vector $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ such that $D = D_{\mathbf{k}'}$, $k'_\ell \geq k_\ell$ for all $\ell \in L$ and $k'_{\hat{\ell}} > k_{\hat{\ell}}$ for some $\hat{\ell} \in L$.

Lemma 1: Each MDS has a unique maximal density vector.

Proof: We prove the theorem by contradiction. Suppose that there exist two distinct maximal density vectors $\mathbf{k} = [k_\ell]_{\ell \in L}$ and $\mathbf{k}' = [k'_\ell]_{\ell \in L}$ for the same dense subgraph D of G. Since $\mathbf{k} \neq \mathbf{k}'$ and both are assumed maximal, there exist at least two layers $\hat{\ell}$ and $\bar{\ell}$ such that $k_{\hat{\ell}} > k'_{\hat{\ell}}$ and $k'_{\bar{\ell}} > k_{\bar{\ell}}$.

Let us define a new density vector $\mathbf{k}^* = [k^*_\ell]_{\ell \in L}$ where $k^*_\ell = \max\{k_\ell, k'_\ell\}$ for all $\ell \in L$. Since both $\mathbf{k}$ and $\mathbf{k}'$ are valid density vectors of D, and for each $\ell \in L$ we have $\rho(G_\ell[D]) > k_\ell$ and $\rho(G_\ell[D]) > k'_\ell$ (by Definition 1), it follows that $\rho(G_\ell[D]) > k^*_\ell$ for all $\ell \in L$. Therefore, $\mathbf{k}^*$ is also a valid density vector of D.

Moreover, for layers $\hat{\ell}$ and $\bar{\ell}$, we have $k^*_{\hat{\ell}} = k_{\hat{\ell}} > k'_{\hat{\ell}}$ and $k^*_{\bar{\ell}} = k'_{\bar{\ell}} > k_{\bar{\ell}}$, implying that $\mathbf{k}^*$ strictly dominates both $\mathbf{k}$ and $\mathbf{k}'$. This contradicts the assumption. Hence, the maximal density vector of any MDS is unique. □

The definition of MDS naturally aligns with the concept of density decomposition in single-layer graph [17]. Inspired by this connection, we aim to extend the notion of density decomposition to multilayer graphs. Formally, we define our problem as follows.

Problem statement: Given a multilayer graph $G = (V, E, L)$, compute the set of all non-empty and distinct $D_{\mathbf{k}}$s of G, each associated with its maximal density vector $\mathbf{k}$. This collection forms the *density decomposition* of the multilayer graph.

III. $\mathbf{K}$-Dense Subgraph Search

Given a multilayer graph $G = (V, E, L)$ and a density vector $\mathbf{k}$, we now address the problem of computing the corresponding $D_{\mathbf{k}}$. However, computing the $D_{\mathbf{k}}$ poses two fundamental algorithmic challenges as follows:

(1) Layer-wise subgraph computation. For each layer $\ell \in L$, identifying the subgraph D_ℓ requires satisfying a non-trivial reachability condition: there must be no directed path from any vertex in $S_\ell = \{u \mid \vec{d}_\ell(u) > k_\ell\}$ to any vertex in $T_\ell = \{u \mid \vec{d}_\ell(u) < k_\ell\}$. Naïvely eliminating all $s \rightsquigarrow t$ paths from S_ℓ to T_ℓ is computationally prohibitive, especially in large dense graphs. Thus, the key difficulty is to design a method that can enforce the global path elimination constraint efficiently.

(2) Cross-layer consistency. Even if all layer-specific subgraphs D_ℓ are computed, enforcing that a common vertex set D satisfies $D = D_\ell$ for all $\ell \in L$ is non-trivial. Due to structural differences across layers and varying density thresholds, the sets D_ℓ may not align. Ensuring consistency requires iterative refinement across layers: removing a vertex from one layer may invalidate the density constraint in another, triggering further updates. Designing a convergence-guaranteed refinement procedure with minimal recomputation is critical for scalability.

To address the above challenges, we first propose the SLFBE algorithm, which transforms the $s \rightsquigarrow t$ path elimination requirement into a maximum flow problem, enabling efficient extraction of dense subgraphs on each layer. To enforce cross-layer consistency, we first introduce the IMLI framework, which iteratively applies SLFBE on all layers and refines the candidate vertex set via intersection. To further enhance efficiency, we develop the LCVSP algorithm, which accelerates the computation of $D_{\mathbf{k}}$ by combining multilayer $\mathbf{k}$-core pruning (Theorem 3) with a layer-constrained vertex set propagation strategy.

A. Single-Layer Flow-Based Extraction

To efficiently compute $D_{\mathbf{k}}$, we first extract the dense subgraph D_ℓ from each individual layer $\ell \in L$. This single-layer extraction serves as the computational backbone of both the IMLI and LCVSP frameworks, since $D_{\mathbf{k}} = D_\ell$ is derived from these layer-wise results.

Here, we propose the SLFBE algorithm, which efficiently identifies all edges that need to be reversed in order to eliminate any reachability from S_ℓ to T_ℓ. Instead of iteratively removing individual $s \rightsquigarrow t$ paths, SLFBE reformulates the reachability constraint as a single maximum flow problem over a flow network. The key insight is that the set of saturated edges in this network directly corresponds to the set of edges in the original graph whose reversal will eliminate all $S_\ell \to T_\ell$ reachability. That is, instead of tracking and reversing specific paths, the algorithm computes all necessary edge reversals in one shot by solving a flow problem. This approach is based on the re-orientation network framework [18]. Below, we present how to construct the re-orientation network for a single layer of the multilayer graph.

Algorithm 1: SLFBE

Input: A single layer graph $\vec{G}_\ell = (V, \vec{E}_\ell)$ and an integer k_ℓ.
Output: k_ℓ-dense subgraph.

```
V' ← V ∪ {s, t} ;                                   // Add source and sink
for each vertex v ∈ V do
    if d⃗_ℓ(v) > k_ℓ then
        Add arc ⟨s, v⟩ to the E⃗_ℓ' with c(s, v) = d⃗_ℓ(v) − k_ℓ;
    if d⃗_ℓ(v) < k_ℓ then
        Add arc ⟨v, t⟩ to the E⃗_ℓ' with c(v, t) = k_ℓ − d⃗_ℓ(v);
for each ⟨u, v⟩ ∈ E⃗_ℓ do
    Add arc ⟨u, v⟩ to the E⃗_ℓ' with c(u, v) = 1;
Compute the maximum flow from s to t of G⃗_ℓ' = (V', E⃗_ℓ', c);
return D_ℓ ← All vertices reachable from vertex s in the residual network;
```

Definition 3: (Re-orientation Flow Network) [18] For an orientation graph $\vec{G}_\ell = (V, \vec{E}_\ell)$ and a positive integer k_ℓ, we construct a flow network $\vec{G}_\ell{}' = (V', \vec{E}_\ell{}', c)$ on the ℓ-th layer graph. Here, $V' = V \cup \{s, t\}$ where s is the source and t is the sink. The edge set $\vec{E}_\ell{}'$ includes three types of edges with non-negative capacities $c(e)$: (1) for each $\langle u, v\rangle$ in $\vec{G}_\ell$ with $c(u, v) = 1$; (2) for each u in V where $\vec{d}_\ell(u) > k_\ell$, an edge $\langle s, u\rangle$ with $c(s, u) = \vec{d}_\ell(u) - k_\ell$; and (3) for each u in V where $\vec{d}_\ell(u) < k_\ell$, an edge $\langle u, t\rangle$ with $c(u, t) = k_\ell - \vec{d}_\ell(u)$.

The complete procedure is shown in Algorithm 1. It first constructs the re-orientation network, then computes the maximum flow. Once the maximum flow is computed, the resulting minimum s-t cut guarantees that no path remains from S_ℓ to T_ℓ, effectively satisfying the reachability condition defined for D_ℓ. The set of nodes reachable from s in the residual graph forms the desired subgraph. The following theorem shows the correctness and complexity of Algorithm 1.

Lemma 2: The SLFBE algorithm can output D_ℓ correctly with a time complexity of $O(|E_\ell|^{1.5})$ and a space complexity of $O(|E_\ell|)$.

Proof: We compute the maximum flow in $\vec{G}_\ell{}'$, by the max-flow min-cut theorem, there is no augmenting path remains. In particular, all directed paths from any node in S_ℓ to any node in T_ℓ must be "cut" by saturated arcs. Suppose for each *saturated* edge $\langle x, y\rangle \in \vec{E}_\ell{}'$ in $\vec{G}_\ell{}'$, it is reversed in $\vec{G}_\ell{}'$. Thus the saturated-arc reversals eliminate all such $s \rightsquigarrow t$ paths. Then in $\vec{G}_\ell{}'$, the set S_ℓ will contain only the nodes in the residual network connected to s by unsaturated edges, and the set T_ℓ will contain only the nodes in the residual network connected to t by unsaturated edges. Because there is no path $s \rightsquigarrow t$ in the residual network, there is also no path from the set S_ℓ to the set T_ℓ in $\vec{G}_\ell{}'$. By Definition 1, the reachability closure of S_ℓ then exactly yields the unique k_ℓ-dense subgraph D_ℓ.

The time and space complexity of the maximum flow computation on G_ℓ is $O(|E_\ell|^{1.5})$ and $O(|E_\ell|)$ respectively [19]. □

Algorithm 2: IMLI

Input: A multilayer graph $\vec{G} = (V, \vec{E}, L)$ and an integer vector $\mathbf{k} = [k_\ell]_{\ell \in L} \in \mathbb{Z}$.
Output: $D_{\mathbf{k}}$.

1 $C \leftarrow V$; $changed \leftarrow$ True;
2 **while** $changed$ **do**
3 $\quad changed \leftarrow$ False; $C_{old} \leftarrow C$;
4 $\quad$ **for** *each layer* $\ell \in L$ **do**
5 $\quad\quad D_\ell \leftarrow \mathsf{SLFBE}(\vec{G}_\ell[C], k_\ell)$;
6 $\quad C \leftarrow \bigcap_{\ell \in L} D_\ell$;
7 $\quad$ **if** $C = \emptyset$ **then return** $\emptyset$;
8 $\quad$ **if** $C \neq C_{old}$ **then**
9 $\quad\quad C \leftarrow C_{old}$; $changed \leftarrow$ True;
10 **return** $G[C] = (C, E[C], L)$;

Algorithm 3: LCVSP

Input: A multilayer graph $\vec{G} = (V, \vec{E}, L)$ and an integer vector $\mathbf{k} = [k_\ell]_{\ell \in L} \in \mathbb{Z}$.
Output: $D_{\mathbf{k}}$

1 $C \leftarrow V$; $changed \leftarrow$ True;
2 **while** $\exists u \in C, \exists \ell \in L : d_\ell^C(u) < k_\ell + 1$ **do**
3 $\quad C \leftarrow C \setminus \{u\}$;
4 **while** $changed$ **do**
5 $\quad changed \leftarrow$ False;
6 $\quad$ **for** *each layer* $\ell \in L$ **do**
7 $\quad\quad$ **if** $G_\ell[C]$ *is empty* **then**
8 $\quad\quad\quad$ **return** $\emptyset$;
9 $\quad\quad D \leftarrow \mathsf{SLFBE}(\vec{G}_\ell[C], k_\ell)$; // Algorithm 1
10 $\quad\quad$ **if** $D \neq C$ **then**
11 $\quad\quad\quad C \leftarrow D$; $changed \leftarrow$ True;
12 **return** $G[C] = (C, E[C], L)$;

B. Iterative Multi-Layer Intersection

After solving the subproblem of extracting each D_ℓ from individual layers, the main challenge is to consolidate these layer-specific subgraphs into a single unified $D_{\mathbf{k}}$. To this end, we introduce the IMLI algorithm, as shown in Algorithm 2. The algorithm initializes the candidate set with all vertices (line 1), and iteratively refines it until convergence. At each iteration, it invokes SLFBE on the subgraph induced by the current candidate set in each layer (lines 4–5), and computes the intersection of the resulting subgraphs (line 6). It checks whether the candidate set has changed compared to the previous iteration and continues only if further updates are required (lines 8–9). The process terminates when the candidate set C stabilizes.

Lemma 3: The IMLI framework achieves a time complexity of $O(t \cdot |L| \cdot \sum_{\ell \in L} |E_\ell|^{1.5})$ and a space complexity of $O(\sum_{\ell \in L} |E_\ell|)$, where t is the number of iterations until convergence, $|L|$ is the number of layers, and $|E_\ell|$ denotes the number of edges in layer ℓ.

Proof: The total cost contains two aspects. First, in each iteration, the algorithm applies SLFBE to each layer, which takes $O(|E_\ell|^{1.5})$ per layer. Summing over all layers and iterations gives the time complexity $O(t \cdot |L| \cdot \sum_{\ell \in L} |E_\ell|^{1.5})$. Second, the intersection and candidate update operations are bounded by $O(|V|)$ per iteration, which is dominated by the flow computation. □

C. Layer-Constrained Vertex Set Propagation

As mentioned, although the IMLI algorithm ensures cross-layer consistency through iterative intersection, it suffers from inefficiencies. Each layer is processed independently, leading to redundant computations. Specifically, vertices that are already excluded by one layer may still be unnecessarily processed in other layers, which not only increases computational overhead but also delays overall convergence.

To overcome this limitation, we propose a more efficient framework, LCVSP, which introduces a layer-constrained vertex set propagation mechanism. Instead of recomputing on the entire candidate set in each layer, LCVSP processes layers sequentially: the result of SLFBE on one layer is propagated as the input subgraph to the next layer. This progressive refinement avoids considering vertices already filtered out by earlier layers, effectively reducing redundant computation and improving convergence efficiency.

Moreover, to further accelerate computation, LCVSP incorporates a multilayer $\mathbf{k}$-core pruning strategy. Based on the Theorem 3, we preemptively eliminate vertices that cannot belong to the final $D_{\mathbf{k}}$, thereby significantly narrowing the search space. The combined design of cross-layer vertex set propagation and $\mathbf{k}$-core-based pruning makes LCVSP both theoretically sound and practically scalable for large multilayer graphs. Next, we introduce the concept of multilayer $\mathbf{k}$-core.

Definition 4: (Multilayer $\mathbf{k}$-core) [10], [20] Given a multilayer graph $G = (V, E, L)$ and an $|L|$-dimensional integer vector $\mathbf{k} = [k_\ell]_{\ell \in L}$, the multilayer $\mathbf{k}$-core of G is a maximal subgraph $G[C] = (C \subseteq V, E[C], L)$ such that $\forall \ell \in L : \mu(C, \ell) \geq k_\ell$.

We observe that the multilayer $\mathbf{k}$-core is closely related to the target $D_{\mathbf{k}}$. This relationship is formalized below:

Theorem 3: Let $G = (V, E, L)$ be a multilayer graph and $\mathbf{k} = [k_\ell]_{\ell \in L}$ be a density vector. Let $C_{\mathbf{k}'}$ denote the multilayer core where $\mathbf{k}' = \mathbf{k} + \mathbf{1}$ (i.e., $k'_\ell = k_\ell + 1, \forall \ell \in L$). Then, $D_{\mathbf{k}} \subseteq C_{\mathbf{k}'} = C_{\mathbf{k+1}}$.

Proof: Fix a layer $\ell \in L$, and consider the $\vec{G}_\ell = (V, \vec{E}_\ell)$ of the ℓ-th layer. This trivially holds when $D_{\mathbf{k}} = \emptyset$, so we focus on case where $D_{\mathbf{k}} \neq \emptyset$. By Observation 1, every node $u \in D_{\mathbf{k}}$ satisfies $\vec{d}_\ell^{D_{\mathbf{k}}}(u) \geq k_\ell$. In particular, if $\vec{d}_\ell^{D_{\mathbf{k}}}(u) = k_\ell$, then u must have at least one incoming edge in $\vec{G}(D_{\mathbf{k}})$. Otherwise, by Definition 1, node u would not qualify for inclusion in $D_{\mathbf{k}}$, contradicting our assumption. Since the total degree of a node in $\vec{G}_\ell$ equals the sum of its indegree and outdegree in the orientation $\vec{G}_\ell$, we conclude that $d_\ell^{D_{\mathbf{k}}}(u) \geq k_\ell + 1$. Therefore, $D_{\mathbf{k}} \subseteq C_{\mathbf{k+1}}$ □

This inclusion result provides a powerful pruning rule: any vertex not in $C_{\mathbf{k}+1}$ is guaranteed not to appear in $D_{\mathbf{k}}$. By applying this result, we can significantly narrow the initial search space before invoking costly flow-based computations.

The LCVSP algorithm equipped with such core reduction technique is depicted in Algorithm 3. LCVSP begins by extracting $C_{\mathbf{k+1}}$ to initialize the candidate set C (lines 1-3). Then, the key idea of LCVSP lies in its layer-constrained vertex set propagation strategy. Specifically, in each iteration (lines 4–11), the algorithm sequentially processes each layer $\ell \in L$, applying the SLFBE algorithm on the subgraph $G_\ell[C]$ induced by C (line 7). This step enforces the layer-specific density constraint k_ℓ, ensuring that only vertices meeting the requirement in layer ℓ are retained. If pruning occurs in any

layer (*i.e.*, $D \neq C$), the updated set $C \leftarrow D$ is immediately propagated to subsequent layers in the same iteration (lines 9–11). The process terminates once C stabilizes and satisfies the density requirements across all layers.

Although the worst-case time and space complexity of LCVSP matches that of IMLI, namely $O(t \cdot |L| \cdot \sum_{\ell \in L} |E_\ell|^{1.5})$ and $O(\max_{\ell \in L} |E_\ell|)$ respectively, LCVSP achieves notable empirical speedups. This improvement stems from two sources: the $(\mathbf{k}+\mathbf{1})$-core pruning eliminates non-qualifying vertices early, and the layer-constrained propagation strategy avoids repeated evaluation of irrelevant candidates. Together, these enhancements make LCVSP substantially more efficient in practice while maintaining correctness and scalability.

IV. Multilayer Density Decomposition

Density decomposition of multilayer graphs is a fundamental yet computationally challenging task due to the combinatorial explosion of candidate MDSs. Specifically, given a multilayer graph with L layers and a maximum attainable density $\rho^*(G_\ell)$ at each layer ℓ, the naive enumeration of all possible density vectors $\mathbf{k}$ would entail exploring a space of size $O(\prod_{\ell=1}^{L} \lceil \rho^*(G_\ell) \rceil)$. This combinatorial explosion renders exhaustive search approaches impractical.

To focus only on density vectors $\mathbf{k}$ whose corresponding $D_\mathbf{k}$ are non-empty, we introduce the concept of a *density lattice*, illustrated in Fig. 2. This structure takes the form of a directed acyclic graph (DAG), where each node represents a feasible density vector $\mathbf{k}$ with non-empty $D_\mathbf{k}$, and directed edges encode unit-wise increments: there is an edge from $\mathbf{k}$ to $\mathbf{k}'$ if $\mathbf{k}' = \mathbf{k} + \mathbf{e}_\ell$ for some $\ell \in L$. Each node $D_\mathbf{k}$ is treated as the parent of its immediate successors $D_{\mathbf{k}'}$. The density lattice offers a structured way to explore the density decomposition. To efficiently traverse it, we propose three algorithms that systematically enumerate these subgraphs. First, we present the VSHT algorithm, which employs a breadth-first search strategy to explore the density lattice level by level. This approach enumerates and processes all complete density vectors $\mathbf{k} \in \mathbb{Z}^L_{\geq 0}$, but it suffers from redundant computations. To overcome this, we propose a novel HPDD algorithm. It adopts a layer-wise decomposition strategy, progressively constructing the decomposition hierarchy in a layer-by-layer manner; Finally, to further enhance scalability by minimizing memory consumption, we propose an advanced algorithm, denoted as HPDD+, that combines recursive enumeration with divide-and-conquer techniques to achieve both space efficiency and reduced computational redundancy.

A. Vector Space Hierarchical Traversal

The VSHT algorithm performs a level by level traversal of the lattice, building on the following theorem.

Theorem 4: Given a multilayer graph $G = (V, E, L)$, let $D_\mathbf{k}$ and $D_{\mathbf{k}'}$ denote the MDSs of G with density vectors $\mathbf{k} = [k_\ell]_{\ell \in L}$ and $\mathbf{k}' = [k'_\ell]_{\ell \in L}$, respectively. If $\mathbf{k}' \leq \mathbf{k}$ (i.e., $\forall \ell \in L : k'_\ell \leq k_\ell$), then the $D_\mathbf{k}$ is a subset of the $D_{\mathbf{k}'}$, formally: $D_\mathbf{k} \subseteq D_{\mathbf{k}'}$.

Proof: We proceed by contradiction. Assume, for the sake of contradiction, that $D_\mathbf{k} \not\subseteq D_{\mathbf{k}'}$, implying that the set $D = D_\mathbf{k} \setminus D_{\mathbf{k}'}$ is non-empty (i.e., $D \neq \emptyset$). According to Theorem 2, the induced subgraph on D in layer ℓ must satisfy the density condition: $|E_\ell(D)| + |E_\times(D, D_\mathbf{k} \setminus D, \ell)| > k_\ell \cdot |D|$, $|E_\ell(D)| + |E_\times(D, D_{\mathbf{k}'}, \ell)| \leq k'_\ell \cdot |D|$. Because $D = D_\mathbf{k} \setminus D_{\mathbf{k}'}$, all vertices in D are not in $D_{\mathbf{k}'}$. Therefore, $D_\mathbf{k} \setminus D \subseteq D_{\mathbf{k}'}$. As a result, $|E_\times(D, D_{\mathbf{k}'}, \ell)| \geq |E_\times(D, D_\mathbf{k} \setminus D, \ell)|$. Combining these inequalities and the condition $k'_\ell \leq k_\ell$, we derive the contradiction $k_\ell \cdot |D| < |E(D)| + |E_\times(D, D_\mathbf{k} \setminus D, \ell)| \leq |E(D)| + |E_\times(D, D_{\mathbf{k}'}, \ell)| \leq k'_\ell \cdot |D|$. Thus, $D_\mathbf{k} \subseteq D_{\mathbf{k}'}$. □

Algorithm 4: VSHT

Input: A multilayer graph $G = (V, E, L)$.
Output: The set $\mathcal{D}$ of all non-empty MDSs of G.

1 $\mathcal{D} \leftarrow \emptyset$, $Q \leftarrow \{[0]_{|L|}\}$, $\mathcal{P}([0]_{|L|}) \leftarrow \emptyset$; // Track parent nodes
2 **while** $Q \neq \emptyset$ **do**
3 dequeue density vector $\mathbf{k}$ from Q;
4 $P_n \leftarrow \bigcap_{F \in \mathcal{P}(\mathbf{k})} F$; // Lemma 4
5 $\mathcal{D}_k \leftarrow \mathsf{LCVSP}(G[P_n], \mathbf{k})$; // Algorithm 3
6 **if** $\mathcal{D}_k \neq \emptyset$ **then**
7 $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{D}_k\}$;
8 **for** *each* $\ell \in L$ **do**
9 $\mathbf{k}' \leftarrow [k_1, \ldots, k_\ell + 1, \ldots, k_{|L|}]$;
10 enqueue $\mathbf{k}'$ into Q;
11 $\mathcal{P}(\mathbf{k}') \leftarrow \mathcal{P}(\mathbf{k}') \cup \{\mathcal{C}_k\}$;
12 **return** $\mathcal{D}$;

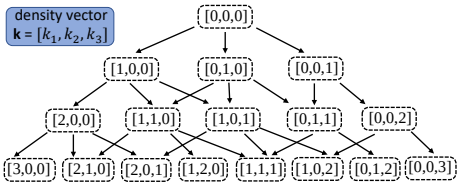


Fig. 2. A three-layer graph density decomposition organizes all $D_\mathbf{k}$s into a density lattice, revealing the evolution of cohesive patterns across layers.

This containment property enables pruning in the density lattice. In particular, any $D_\mathbf{k}$ must be fully contained within the intersection of its parent subgraphs in the density lattice. This yields the following lemma:

Lemma 4: Given a multilayer graph G and a density vector $\mathbf{k}$, let $\mathcal{P}(D_\mathbf{k})$ be the set of parents of $D_\mathbf{k}$ in the density lattice of G. It holds that $D_\mathbf{k} \subseteq \bigcap_{\hat{D} \in \mathcal{P}(D_\mathbf{k})} \hat{D}$.

Proof: By definition of density lattice, the density vector of all parents $\mathcal{P}(D_\mathbf{k})$ of $D_\mathbf{k}$ is dominated by the $\mathbf{k}$. Thus, according to Theorem 4, it holds that $D_\mathbf{k} \subseteq D_{\mathbf{k}'}$, $\forall D_{\mathbf{k}'} \in \mathcal{P}(D_\mathbf{k})$. Assume a vertex $u \notin \bigcap_{\hat{D} \in \mathcal{P}(D_\mathbf{k})} \hat{D}$, $u \in D_\mathbf{k}$ exists. This implies that there exists a parent $D' \in \mathcal{P}(D_\mathbf{k})$ such that $D_\mathbf{k} \not\subseteq D'$, thus leading to a contradiction. □

The VSHT algorithm, as showned in Algorithm 4, implements these theoretical insights through a breadth-first search strategy that systematically traverses the density vector space. Starting with the root density vector $[0]_{|L|}$ (line 1), it maintains a queue of density vectors, dequeuing and processing each density vector $\mathbf{k}$ by computing its $D_\mathbf{k}$ through Algorithm 3 on the intersection of parent nodes (line 5). Non-empty $D_\mathbf{k}$ are added to the result set, and child density vectors are generated by incrementing individual layer density (lines 9-11).

Lemma 5: Let B be the number of density vectors visited in the density lattice. The total time complexity is $O(B \cdot t \cdot |L| \cdot \sum_{\ell \in L} |E_\ell|^{1.5})$. In practice, B is much smaller than the theoretical upper bound $O\left(\prod_{\ell=1}^{|L|} \rho^*(G_\ell)\right)$, as the algorithm uses parent intersection to restrict subgraph size and prunes branches leading to empty subgraphs.

For space complexity, VSHT only maintains subgraphs with active children in the queue, requiring at most two levels of

the density lattice. Thus, the space usage is $O(|V| \cdot W)$, where W is the maximum width of two consecutive lattice levels.

B. Hierarchical Progressive Density Decomposition

In this subsection, we now introduce the HPDD algorithm. Unlike VSHT, which traverses the density vector space by evaluating each density vector $\mathbf{k} \in \mathbb{Z}^L_{\geq 0}$ defined over all layers, HPDD incrementally constructs density vector $\mathbf{k}$ in a layer-wise manner, by progressively extending partial density vectors one layer at a time. This hierarchical progressive strategy enables localized exploration and early pruning, thereby reducing both the search space and the number of subgraph computations required.

Below, we first introduce the concept of partial density vector, which serve as building blocks for constructing density vector $\mathbf{k}$ layer by layer.

Definition 5: (Partial Density Vector) Let $G = (V, E, L)$ be a multilayer graph, and let $\mathbf{k} = [k_1, \ldots, k_{|L|}]$ be a density vector defined over L layers. For a subset of the first j layers $\hat{L} = \{1, 2, \ldots, j\} \subset L$ with $1 \leq j < |L|$, let $\mathbf{k}^j = [k_1, k_2, ..., k_j]$ denote the corresponding partial density vector over $\hat{L}$.

We next establish a containment property between MDSs defined over density vectors $\mathbf{k}$ and partial density vectors $\mathbf{k}^j$.

Lemma 6: Let $D_{\mathbf{k}}$ and $D_{\mathbf{k}^j}$ be the MDSs of multilayer graph $G = (V, E, L)$ and the induced subgraph $G_{\hat{L}} = (V, E_{\hat{L}}, \hat{L})$, respectively. We have $D_{\mathbf{k}} \subseteq D_{\mathbf{k}^j}$.

Proof: We proceed by contradiction. Assume that $D_{\mathbf{k}} \nsubseteq D_{\mathbf{k}^j}$. This implies the existence of a vertex $u \in D_{\mathbf{k}}$ such that $u \notin D_{\mathbf{k}^j}$. Thus, there must exist at least one layer $j \in [1, \ldots, j]$ where u fails to satisfy the condition for $\mathbf{k}^j$. However, by the given condition $\mathbf{k}^j = [k_1, k_2, ..., k_j]$, this contradicts the initial assumptions. Therefore, we conclude that $D_{\mathbf{k}} \subseteq D_{\mathbf{k}^j}$. □

This containment property lays the theoretical foundation for HPDD. It ensures that any $D_{\mathbf{k}}$ satisfying the density vector $\mathbf{k}$ is always a subset of the corresponding MDS computed over any prefix of layers. Based on this, we define how partial density vectors are progressively extended in HPDD.

Definition 6: (Partial Density Vector Extension) Given a partial density vector $\mathbf{k}^j = [k_1, \ldots, k_j]$, we define its extension by a value i as the new partial density vector $\mathbf{k}^{(j+1)} = \mathbf{k}^j \circ i = [k_1, \ldots, k_j, i]$.

We now formalize a key pruning property of partial density vector extensions, which guarantees the correctness and efficiency of the hierarchical progressive search process.

Lemma 7: Let $G = (V, E, L)$ be a multilayer graph, and let $\mathbf{k}^j = [k_1, \ldots, k_j]$ be a partial density vector defined over the first j ($j < |L|$) layers. If $D_{\mathbf{k}^j} = \emptyset$, then for any value $i \geq 0$, $D_{\mathbf{k}^j \circ i} = \emptyset$.

Intuitively, this lemma formalizes the monotonic pruning effect: once a partial density vector yields an empty MDS, then any of its extended density vectors will also yield empty results. As a result, these branches can be safely eliminated from further exploration. These theoretical results lead us to a comprehensive strategy for enumerating all non-empty MDSs:

Theorem 5: For a multilayer graph $G = (V, E, L)$ with $\ell = |L|$ layers, all non-empty $D_{\mathbf{k}}$ can be enumerated through the recursive procedure:

Algorithm 5: HPDD

Input: A multilayer graph $G = (V, E, L)$.
Output: The set $\mathcal{D}$ of all non-empty MDSs of G.

$\mathcal{D} \leftarrow \emptyset$, $Q \leftarrow \emptyset$, enqueue $(V, \emptyset)$ into Q;
for $\ell \leftarrow 1$ **to** $|L|$ **do**
  $Q' \leftarrow \emptyset$;
  while $Q \neq \emptyset$ **do**
    dequeue (S, K) from Q;
    $\hat{L} \leftarrow \{1, 2, \ldots, \ell\}$;
    $G'[S] \leftarrow (S, E[S], \hat{L})$;
    Arbitrarily orient the subgraph $G'[S]$ to obtain $\vec{G}'[S]$;
    for *each* $(u, \ell) \in S \times \hat{L}$ **do**
      $\vec{d}_\ell(u) \leftarrow$ outdegree of u in $\vec{G}'_\ell[S]$;
    enqueue $(S, K \circ 0)$ into Q';
    for $i \leftarrow 1, 2, 3 \ldots$ **do**
      $\mathbf{k} \leftarrow K \circ i$;
      $D_{\mathbf{k}} \leftarrow \mathsf{LCVSP}(\vec{G}'[S], \mathbf{k})$;
      if $D_{\mathbf{k}} \neq \emptyset$ **then**
        enqueue $(D_{\mathbf{k}}, \mathbf{k})$ into Q';
        if $\ell = |L|$ **then** $\mathcal{D} \leftarrow \mathcal{D} \cup D_{\mathbf{k}}$;
      else break;
  $Q \leftarrow Q'$;

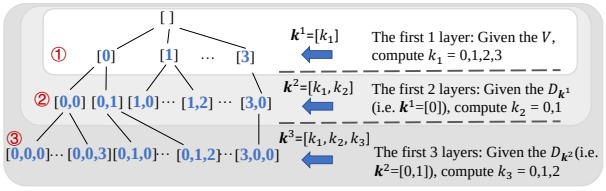


Fig. 3. An example of HPDD for 3 layer graph

1) Compute all non-empty $D_{\mathbf{k}^1}$ on the first layer with $\mathbf{k}^1 = [k_1]$.
2) For each non-empty $D_{\mathbf{k}^j}$, compute all non-empty $D_{\mathbf{k}^{(j+1)}}$ induced by $D_{\mathbf{k}^j}$ on the first $j+1$ layers, where $\mathbf{k}^{(j+1)} = [k_1, k_2, \ldots, k_j, k_{j+1}]$.
3) Continue this process until all ℓ layers are considered.

Proof: We prove this by induction on the number of layers.

Base case: For a single-layer graph, computing all non-empty $D_{\mathbf{k}^1}$, where $\mathbf{k}^1 = [k_1]$ for $k_1 \geq 0$, enumerates all possible MDSs.

Inductive hypothesis: Assume the theorem holds for any multilayer graph with j layers, where $1 \leq j < \ell$.

Inductive step: Consider a multilayer graph with the first $j+1$ layers. By the inductive hypothesis, we can enumerate all non-empty $D_{\mathbf{k}^j}$ for the first j layers, where $\mathbf{k}^j = [k_1, k_2, \ldots, k_j]$. For any non-empty $D_{\mathbf{k}^{(j+1)}}$ where $\mathbf{k}^{(j+1)} = [k_1, k_2, \ldots, k_j, k_{j+1}]$, by Lemma 6, we know that $D_{\mathbf{k}^{(j+1)}} \subseteq D_{\mathbf{k}^j}$. Conversely, by Lemma 7, if $D_{\mathbf{k}^j} = \emptyset$, then $D_{\mathbf{k}^{(j+1)}} = \emptyset$ for any $k_{j+1} \geq 0$. Therefore, we only need to consider extensions of non-empty $D_{\mathbf{k}^j}$. For each non-empty $D_{\mathbf{k}^j}$, we compute all $D_{\mathbf{k}^{(j+1)}}$ for $k_{j+1} \geq 0$ by computing the dense subgraphs induced by $D_{\mathbf{k}^j}$ on the $(j+1)$-th layer. This procedure ensures that we enumerate all possible non-empty dense subgraphs for the $(j+1)$-layer graph, as any such subgraph must be contained within some non-empty dense subgraph of the first j layers. By the principle of induction, the theorem holds for multilayer graphs.

□

The pseudocode of HPDD is presented in Algorithm 5. HPDD begins with the entire vertex set and an empty density

vector (line 1), and iteratively processes layers from $\ell = 1$ to $|L|$ (lines 2–19). At each layer ℓ, HPDD considers all candidate pairs $(S, \mathbf{k}^{(\ell-1)})$ in Q, where S is the vertex set corresponding to a valid partial density vector up to layer $\ell-1$. For each such pair (lines 4–18), the algorithm constructs a multilayer subgraph $G'[S]$ induced by S over the first ℓ layers (lines 6–7), and computes an orientation (lines 8–10).

It then extends $\mathbf{k}^{(\ell-1)}$ by incrementing the ℓ-th component (lines 11–18). For each candidate $\mathbf{k}^\ell = \mathbf{k}^{(\ell-1)} \circ i$, the LCVSP algorithm is applied to $\vec{G}'[S]$ to compute the corresponding $D_{\mathbf{k}^\ell}$. If the result is non-empty, the pair $(D_{\mathbf{k}^\ell}, \mathbf{k}^\ell)$ is enqueued into Q' for further exploration. If $\ell = |L|$, the result is added to the $\mathcal{D}$. Otherwise, if an empty subgraph is encountered, further increments are pruned.

To concretely demonstrate how HPDD operates, we consider the example shown in Fig. 3.

Example 1: ①: Start with layer 1 by constructing $G_{\hat{L}} = (V, E_{\hat{L}}, \hat{L})$ and computing all $\mathbf{k}^1 = [k_1]$ with $k_1 \in \{0, 1, 2, 3\}$ and corresponding MDSs $D_{\mathbf{k}^1}$. ②: For each $D_{\mathbf{k}^1}$, the algorithm constructs $G_{\hat{L}} = (D_{\mathbf{k}^1}, E_{\hat{L}}, \hat{L})$ with $\hat{L} = \{1, 2\}$. Then compute all $\mathbf{k}^2 = [k_1, k_2]$ with valid k_2 values (e.g., $k_2 \in \{0, 1\}$ when $k_1 = 0$). ③: For each $D_{\mathbf{k}^2}$, further extend to layer $\{3\}$ to obtain $\mathbf{k}^3 = [k_1, k_2, k_3]$; e.g., for $\mathbf{k}^2 = [0, 1]$, compute $k_3 \in \{0, 1, 2\}$.

Lemma 8: The worst-case time complexity of the Algorithm 5 is identical to that of the Algorithm 4, i.e., $O(B \cdot t \cdot |L| \cdot \sum_{\ell \in L} |E_\ell|^{1.5})$. This bound arises because, in the worst case, HPDD may enumerate all possible non-empty density vectors. However, due to the hierarchical progressive strategy, the number of explored subgraph candidates at each stage is significantly reduced. As a result, the practical value of t is much smaller for HPDD than for VSHT.

The space complexity of the Algorithm 5 is bounded by $O(|V| \cdot \prod_{\ell=1}^{L} \lceil \rho^*(G_\ell) \rceil)$. However, in real-world multilayer graphs, the space complexity is often significantly smaller than its theoretical bound. This is because it is generally impossible for a vertex subset to simultaneously achieve the maximum density on all layers. Consequently, the actual number of non-empty candidates is much lower than the worst-case bound.

Proof: Let us consider the first ℓ layers, the set of all partial density vectors as K^ℓ, with $|K^\ell| \leq \prod_{j=1}^{\ell} \lceil \rho^*(G_\ell) \rceil$. For each density vector $\mathbf{k}^\ell \in K^\ell$, the corresponding candidate subgraph $D_{\mathbf{k}^\ell}$ is stored explicitly.

We prove by induction on ℓ:

Base case ($\ell = 1$): At the first layer, the number of partial density vectors is at most $\rho^*(G_1)$, each with a candidate subgraph of size at most $|V|$. So, total space: $O(|V| \cdot \rho^*(G_1))$.

(Inductive step): Assume that at layer ℓ, the number of non-empty candidates is at most $\prod_{j=1}^{\ell} \lceil \rho^*(G_j) \rceil$. At layer $\ell + 1$, for each such candidate, the number of possible extensions is at most $\lceil \rho^*(G_{\ell+1}) \rceil$. Thus, at layer $\ell + 1$, the number of non-empty candidates is at most: $|K^{(\ell+1)}| \leq |K^\ell| \cdot \lceil \rho^*(G_{\ell+1}) \rceil \leq \prod_{j=1}^{\ell+1} \lceil \rho^*(G_j) \rceil$. Each associated subgraph requires at most $|V|$ space. By induction, after all $|L|$ layers, the total space required is: $O\left(|V| \cdot \prod_{\ell=1}^{|L|} \lceil \rho^*(G_\ell) \rceil\right)$. □

C. The Space Efficient Algorithm: HPDD+

While HPDD improves the efficiency of multilayer density decomposition, it is still hindered by two critical limitations on large-scale graphs: (1) excessive memory consumption due to the need to maintain all intermediate MDSs corresponding to partial density vectors, and (2) redundant computation when exploring different density thresholds within each layer. To address these bottlenecks, we propose HPDD+, an advanced algorithm that combines a recursive depth-first search with a divide-and-conquer techniques, achieving both space efficiency and computational efficiency.

Lemma 9: Let $G = (V, E, L)$ be a multilayer graph. For any $D_{\mathbf{k}}$ with $\mathbf{k} = [k_1, k_2, \ldots, k_{|L|}]$, there exists a sequence of partial density vectors $\{\mathbf{k}^1, \mathbf{k}^2, \ldots, \mathbf{k}^{(|L|-1)}\}$. A recursive exploration that, for each $D_{\mathbf{k}^j}$, immediately computes its extensions to layer $j + 1$ through $|L|$ before exploring other $D_{\mathbf{k}^j}$ at layer j, is sufficient to discover all MDSs.

Proof: By Theorem 5, all $D_{\mathbf{k}}$s can be enumerated through a recursive procedure that considers layer-by-layer extensions of partial density vectors. Consider a recursive depth-first strategy: for each valid $D_{\mathbf{k}^j}$ at layer j, immediately compute all possible extensions to $D_{\mathbf{k}^{(j+1)}}$ at layer $j + 1$, and for each such $D_{\mathbf{k}^{(j+1)}}$, recursively extend to layer $j+2$, and so on until layer ℓ, before backtracking to explore other $D_{\mathbf{k}^j}$ at layer j. □

According to Lemma 9, HPDD+ follows a single recursive paths through the layers sequentially. Specifically, for each partial density vector, extensions to subsequent layers are processed immediately and recursively. This significantly reduces memory requirements as it eliminates the need to store all the MDSs corresponding to all partial density vectors, reducing the space complexity from $O\left(|V| \cdot \prod_{\ell=1}^{|L|} \lceil \rho^*(G_\ell) \rceil\right)$ to $O\left(|V| \cdot \sum_{\ell=1}^{L} \lceil \rho^*(G_\ell) \rceil\right)$, as proved in Lemma 11.

In addition to memory efficiency, HPDD+ leverages a divide-and-conquer technique to efficiently compute all MDSs at each layer. This approach relies on hierarchical containment property among dense subgraphs with varying density values for the same prefix of partial density vector.

Lemma 10: Let $G = (V, E, L)$ be a multilayer graph, and let $\hat{L} \subseteq L$ be a subset of layers. For any two partial density vectors $\mathbf{x}, \mathbf{y}$ defined on $\hat{L}$, if they differ only in one component $j = |\hat{L}|$ such that $x_j > y_j$ and $x_i = y_i$ for all $i \neq j$, then $D_{\mathbf{x}} \subseteq D_{\mathbf{y}}$ in the induced subgraph $G_{\hat{L}} = (V, E_{\hat{L}}, \hat{L})$.

Based on the hierarchical containment property, we can derive the locally resolvable property which provides a stronger theoretical basis for our divide-and-conquer algorithm.

Theorem 6: Let $G = (V, E, L)$ be a multilayer graph and $\hat{L} = \{1, 2, \ldots, \ell\} \subseteq L$ be the first ℓ layers. For three partial density vectors $\mathbf{x}^\ell, \mathbf{y}^\ell, \mathbf{p}^\ell \in \mathbb{Z}^\ell$, if $\mathbf{x}^\ell[i] = \mathbf{y}^\ell[i] = \mathbf{p}^\ell[i]$, $\forall i < \ell$, and $x_\ell < p_\ell < y_\ell$, and the MDSs $D_{\mathbf{x}^\ell}$ and $D_{\mathbf{y}^\ell}$ are known, then the MDS corresponding to $\mathbf{p}^\ell$ can be computed as: $D_{\mathbf{p}^\ell} = \text{LCVSP}\left(\vec{G}[D_{\mathbf{x}^\ell} \setminus D_{\mathbf{y}^\ell}],\ p_\ell\right) \cup D_{\mathbf{y}^\ell}$.

Proof: By Lemma 10, we have $D_{\mathbf{y}^\ell} \subseteq D_{\mathbf{p}^\ell}$ and $D_{\mathbf{p}^\ell} \subseteq D_{\mathbf{x}^\ell}$. Therefore, $D_{\mathbf{p}^\ell}$ can be partitioned as $D_{\mathbf{p}^\ell} = (D_{\mathbf{p}^\ell} \setminus D_{\mathbf{y}^\ell}) \cup D_{\mathbf{y}^\ell}$.

Now, we need to compute the set $D_{\mathbf{p}^\ell} \setminus D_{\mathbf{y}^\ell}$, which consists of vertices that are in $D_{\mathbf{p}^\ell}$ but not in $D_{\mathbf{y}^\ell}$. Since $D_{\mathbf{p}^\ell} \subseteq D_{\mathbf{x}^\ell}$, we know that $D_{\mathbf{p}^\ell} \setminus D_{\mathbf{y}^\ell} \subseteq D_{\mathbf{x}^\ell} \setminus D_{\mathbf{y}^\ell}$.

According to Observation 1 regarding boundary orientation, all edges between $D_{\mathbf{y}^\ell}$ and $D_{\mathbf{x}^\ell} \setminus D_{\mathbf{y}^\ell}$ are oriented towards

Algorithm 6: HPDD+

Input: A multilayer graph $G = (V, E, L)$.
Output: The set $\mathcal{D}$ of all non-empty MDSs of G.

$\mathcal{D} \leftarrow \emptyset$; $\mathbf{k}^0 \leftarrow []$;
Recursive$(G, V, [0], 0, 1)$;
Function Recursive$(G, C, \mathbf{k}^\ell, ub, \ell)$:
  $Q \leftarrow \emptyset$;
  if $k_\ell = 0$ **then**
    $ub \leftarrow \delta(G_\ell[C])$; // Core decomposition [21]
  $l \leftarrow 0, R^l_\ell \leftarrow C$; $u \leftarrow ub, R^u_\ell \leftarrow \emptyset$;
  Divide-and-Conquer $(\vec{G}, \mathbf{k}^\ell, \ell, R^l_\ell, R^u_\ell)$;
  for $i \leftarrow 0$ **to** u **do**
    if $R^i_\ell \neq \emptyset$ **then**
      if $\ell = |L|$ **then**
        $\mathbf{k} \leftarrow \mathbf{k}^\ell$; $D_{\mathbf{k}} \leftarrow R^i_\ell$; $\mathcal{D} \leftarrow \mathcal{D} \cup D_{\mathbf{k}}$;
      else
        $ub \leftarrow i$; enqueue $(R^i_\ell, \mathbf{k}^{(\ell-1)} \circ i)$ into Q;
  $ub' \leftarrow ub$; // Preserve upper bound before recursion
  if $\ell + 1 > |L|$ **then**
    for *each pair* $(D_{\mathbf{k}^\ell}, \mathbf{k}^\ell)$ *in* $(\mathcal{C}, \mathcal{K})$ **do**
      $\hat{L} \leftarrow \{1, 2, \ldots, \ell, \ell + 1\}$;
      $G'[D_{\mathbf{k}^\ell}] \leftarrow (D_{\mathbf{k}^\ell}, E[D_{\mathbf{k}^\ell}], \hat{L})$;
      Arbitrarily orient the subgraph $G'[S]$ to obtain $\vec{G}'[S]$;
      for *each* $(u, \ell') \in S \times \hat{L}$ **do**
        $\vec{d}_{\ell'}(u) \leftarrow$ outdegree of u in $\vec{G}'_{\ell'}[S]$;
      $\mathbf{k}^{\ell+1} \leftarrow \mathbf{k}^\ell \circ 0$;
      Recursive$(G'[D_{\mathbf{k}^\ell}], D_{\mathbf{k}^\ell}, \mathbf{k}^{\ell+1}, ub, \ell + 1)$;
  $ub \leftarrow ub'$;

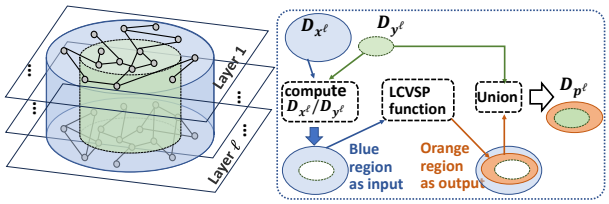


Fig. 4. Illustration of the locally resolvable property

$D_{\mathbf{y}^\ell}$. This means that vertices in $D_{\mathbf{y}^\ell}$ cannot have outgoing edges to vertices in $D_{\mathbf{x}^\ell} \setminus D_{\mathbf{y}^\ell}$.

Therefore, the computation of vertices in $D_{\mathbf{x}^\ell} \setminus D_{\mathbf{y}^\ell}$ is independent of the vertices in $D_{\mathbf{y}^\ell}$. Specifically, the LCVSP algorithm computes a $D_{\mathbf{k}}$ by iteratively applying the SLFBE algorithm to each layer until convergence. When SLFBE is applied to compute a p_ℓ-dense subgraph in layer ℓ, any vertex u with outdegree $\vec{d}_\ell(u) \geq y_\ell > p_\ell$ in $D_{\mathbf{y}^\ell}$ will be connected to the source s in the flow network and will definitely be included in the result.

Hence, $D_{\mathbf{p}^\ell} = \text{LCVSP}(\vec{G}[D_{\mathbf{x}^\ell} \setminus D_{\mathbf{y}^\ell}], p_\ell) \cup D_{\mathbf{y}^\ell}$. □

This resolution process, illustrated in Fig. 4, efficiently computes any intermediate subgraph $D_{\mathbf{p}^\ell}$ once the MDSs corresponding to $\mathbf{x}^\ell$ and $\mathbf{y}^\ell$ are known. The recursion then proceeds independently on the two disjoint regions—$D_{\mathbf{x}^\ell} \setminus D_{\mathbf{p}^\ell}$ and $D_{\mathbf{p}^\ell} \setminus D_{\mathbf{y}^\ell}$, enabling reuse of previous results and progressive search space reduction.

To apply the Theorem 6 in practice, consider a partial density vector $\mathbf{k}^{(\ell-1)}$, whose corresponding subgraph $D_{\mathbf{k}^{(\ell-1)}}$ has been computed. Our goal is to explore all valid extensions $\mathbf{k}^\ell = \mathbf{k}^{(\ell-1)} \circ k_\ell$, where k_ℓ varies across a feasible interval. The lower bound of interval is naturally $lb = 0$, since $k_\ell = 0$ retains the previous subgraph $D_{\mathbf{k}^\ell} = D_{\mathbf{k}^{\ell-1}}$. However, determining a tight upper bound ub is crucial: a loose or overly conservative

Algorithm 7: Divide-and-Conquer

Input: A multilayer graph $\vec{G} = (V, \vec{E}, L)$, $\mathbf{k}^\ell$, ℓ, R^u_ℓ, R^l_ℓ.
Output: k-dense subgraph.

if $u - l \leq 1$ ***or*** $R^u_\ell = R^l_\ell$ **then**
  if $R^u_\ell = R^l_\ell$ **then**
    for $i \leftarrow l + 1$ **to** $u - 1$ **do** $R^i_\ell \leftarrow R^l_\ell$;
  return;
$mid \leftarrow \lceil (u + l + 1)/2 \rceil$;
$R \leftarrow R^l_\ell \setminus R^u_\ell$;
for *each* $\ell \in L$ **do**
  for *each* $(u, v) \in E_\times(R, R^u_\ell, \ell)$ **do**
    if $\langle v, u \rangle \wedge (u \in R) \wedge (v \in R^u_\ell))$ **then**
      reorient edge as $\langle u, v \rangle$;
      $\vec{d}_\ell(u) \leftarrow \vec{d}_\ell(u) + 1$;
$\mathbf{k}^\ell \leftarrow \mathbf{k}^{\ell-1} \circ mid$;
$R^{mid}_\ell \leftarrow \text{LCVSP}(\vec{G}[R], \mathbf{k}^\ell) \cup R^u_\ell$; // Algorithm 3
if $R^{mid}_\ell \neq \emptyset$ **then**
  Divide-and-Conquer $(\vec{G}, \mathbf{k}^\ell, \ell, R^{mid}_\ell, R^l_\ell)$;
  Divide-and-Conquer $(\vec{G}, \mathbf{k}^\ell, \ell, R^u_\ell, R^{mid+1}_\ell)$;

bound can trigger redundant recursive calls and unnecessary computation. We thus introduce an adaptive upper bound to restrict the search interval and maximize the efficiency of the divide-and-conquer procedure.

Theorem 7: Let $G = (V, E, L)$ be a multilayer graph. For any partial density vector $\mathbf{k}^j$ and its corresponding $D_{\mathbf{k}^j}$, the maximum possible density value for layer $j + 1$ in any extension $\mathbf{k}^{(j+1)} = [k_1, k_2, \ldots, k_{j+1}]$ is bounded by: $\max_{k_{j+1}}\{k_{j+1} : D_{\mathbf{k}^{(j+1)}} \neq \emptyset\} \leq \min\{\max_{\mathbf{p}^j \prec \mathbf{k}^j} \triangle_{j+1}(\mathbf{p}^j), \delta(G_{j+1}[D_{\mathbf{k}^j}])\}$,

where: (1) $\delta(G_{j+1}[D_{\mathbf{k}^j}])$ is the degeneracy of the induced subgraph on layer $j+1$; (2) $\mathbf{p}^j \prec \mathbf{k}^j$ means $\mathbf{p}^j$ precedes $\mathbf{k}^j$ in the recursive exploration order; (3) $\triangle_{j+1}(\mathbf{p}^j)$ is the maximum density value observed at layer $j+1$ when extending from $\mathbf{p}^j$;

Proof: First, by the lemma from Chang et al. [22], for any unipartite graph G, the density of its densest subgraph is bounded by $\rho^*(S) \leq \delta(G)$, where $\delta(G)$ is the degeneracy of G. When considering the induced subgraph $G_{j+1}[D_{\mathbf{k}^j}]$ on layer $j+1$, this implies that any dense subgraph within it must have a density at most $\delta(G_{j+1}[D_{\mathbf{k}^j}])$.

This gives us our first upper bound: $\max_{k_{j+1}}\{k_{j+1} : D_{\mathbf{k}^{(j+1)}} \neq \emptyset\} \leq \delta(G_{j+1}[D_{\mathbf{k}^j}])$.

Now, let's consider the recursive exploration pattern of HPDD+. For a two-layer graph example with $L = \{1, 2\}$, we compute multiple $D_{\mathbf{k}^1}$ where $\mathbf{k}^1 = [k_1]$ and $k_1 \in \{0, 1, \ldots, \triangle_1\}$, with $\triangle_1 \leq \delta(G_1[V])$.

When we extend from $\mathbf{k}^1 = [0]$ to layer 2, we initially set an upper bound $ub = \delta(G_2[D_{[0]}])$ and compute all $D_{\mathbf{k}^2}$ where $\mathbf{k}^2 = [0, k_2]$ and $k_2 \in \{0, 1, \ldots, \triangle_2\}$, with $\triangle_2 \leq \delta(G_2[D_{[0]}])$. When we backtrack and extend from $\mathbf{k}^1 = [1]$ to layer 2, we know that $D_{[1]} \subset D_{[0]}$ by Lemma 10. This means that the induced subgraph $G_2[D_{[1]}]$ is a subgraph of $G_2[D_{[0]}]$. This also means that the maximum possible density value at layer 2 when extending from $\mathbf{k}^1 = [1]$ cannot exceed the maximum density value observed when extending from $\mathbf{k}^1 = [0]$, which is $\triangle_2$. Similarly, when we extend from $\mathbf{k}^1 = [2]$ to layer 2, the maximum possible density value cannot exceed the maximum observed when extending from $\mathbf{k}^1 = [1]$.

Generalizing this pattern, for any partial density vector $\mathbf{k}^j$, the maximum possible density value at

layer $j+1$ is bounded by the maximum density value observed at layer $j+1$ when extending from any preceding partial density vector $\mathbf{p}^j$ in the recursive exploration order. This gives us our second upper bound: $\max_{k_{j+1}}\{k_{j+1} : D_{\mathbf{k}^{(j+1)}} \neq \emptyset\} \leq \max_{\mathbf{p}^j \prec \mathbf{k}^j} \triangle_{j+1}(\mathbf{p}^j)$. Combining these two constraints, we get: $\max_{k_{j+1}}\{k_{j+1} : D_{\mathbf{k}^{(j+1)}} \neq \emptyset\} \leq \min\{\max_{\mathbf{p}^j \prec \mathbf{k}^j} \triangle_{j+1}(\mathbf{p}^j), \delta(G_{j+1}[D_{\mathbf{k}^j}])\}$. □

Based on the above results, we develop the HPDD+ algorithm, presented in Algorithm 6, and the recursive divide-and-conquer subroutine is detailed in Algorithm 7.

The HPDD+ algorithm initiates the recursive enumeration by invoking the `Recursive` function (line 2), starting from the initial vector $[0]$ on layer $\ell = 1$. Each recursive call computes all valid extensions of a given partial density vector $\mathbf{k}^\ell = [k_1, k_2, \ldots, k_\ell]$ (lines 3–25). At each layer ℓ, it maintains the feasible interval $[0, ub]$ of k_ℓ values, where the upper bound ub is either inherited or computed via core decomposition if $k_\ell = 0$ (line 5). Given this interval, the algorithm invokes the Divide-and-Conquer subroutine (line 7) to compute all R^i_ℓ $(D_{\mathbf{k}^\ell})$ for $i \in [0, ub]$.

If a valid R^i_ℓ is found, the algorithm either outputs it (when $\ell = |L|$), or stores it for further expansion to the next layer. For each such subgraph, the algorithm constructs the induced multilayer subgraph and orients edges to support local density constraint checking in the next recursive call (lines 15-23). The process continues until all layers have been processed.

The divide-and-conquer subroutine (Algorithm 7) computes all feasible values of k_ℓ. It recursively splits the interval $[l, u]$, applies LCVSP on the residual region $R = R^l_\ell \setminus R^u_\ell$, and merges results with R^u_ℓ to obtain R^{mid}_ℓ (line 12). Once R^{mid}_ℓ is computed, the algorithm proceeds to explore both halves of the current interval.

Lemma 11: The worst-case time complexity of the Algorithm 6 is $O(B \cdot t \cdot |L| \cdot \sum_{\ell \in L} |E_\ell|^{1.5})$. The space complexity of the Algorithm 5 is bounded by $O(|V| \cdot \sum_{\ell=1}^{L} \lceil \rho^*(G_\ell) \rceil)$.

V. Experiments

A. Experimental Setup

Algorithms. To evaluate our proposed multilayer $\mathbf{k}$-dense subgraph model, we implement algorithms for two key tasks: (1) computing $D_{\mathbf{k}}$ for a given density vector $\mathbf{k}$, and performing density decomposition over all feasible $\mathbf{k}$. Specifically, we implement IMLI (Algorithm 2) and LCVSP (Algorithm 3) for computing a $D_{\mathbf{k}}$, and three decomposition algorithms: VSHT (Algorithm 4), HPDD (Algorithm 5), and HPDD+ (Algorithm 6). Since our formulation of $\mathbf{k}$-dense subgraph is novel and tailored to multilayer graphs, no existing methods directly support $D_{\mathbf{k}}$ computation or complete decomposition. As baselines, we therefore use simplified versions of our own algorithms: a basic IMLI algorithm for $D_{\mathbf{k}}$ computation, and the VSHT algorithm for density decomposition. During decomposition, $D_{\mathbf{k}}$ computation is repeatedly invoked as a subroutine, where we employ LCVSP for optimal performance. All algorithms are implemented in C++ and compiled with the g++ compiler. Experiments are conducted on a Linux system equipped with an Intel Xeon Gold 5218R CPU and 96GB DDR4 memory.

TABLE I
Datasets used in the experiments

Graphs	Abbr.	Category	$\|V\|$	$\|E\|$	$\|L\|$	$\min_{\ell \in L} \frac{\|E_\ell\|}{\|V\|}$	$\max_{\ell \in L} \frac{\|E_\ell\|}{\|V\|}$
Homo	HO	Genetic	18k	153k	7	0.014	4.583
SacchCere	SC	Genetic	6.5k	247k	7	0.205	13.997
Cannes2013	CA	Social	438k	962k	3	0.179	1.121
DBLP	DB	Co-authorship	513k	1.0M	10	0.187	0.220
ObamaInIsrael	OI	Social	2.2M	3.8M	3	0.244	0.792
Amazon	AM	Co-purchasing	410k	8.1M	4	2.193	5.956
Higgs	HI	Social	456k	13M	4	0.061	27.392
FriendFeed	FF	Social	505k	18M	3	0.527	35.669
FlickrGrowth	FG	Social	2.3M	23M	2	2.139	6.686

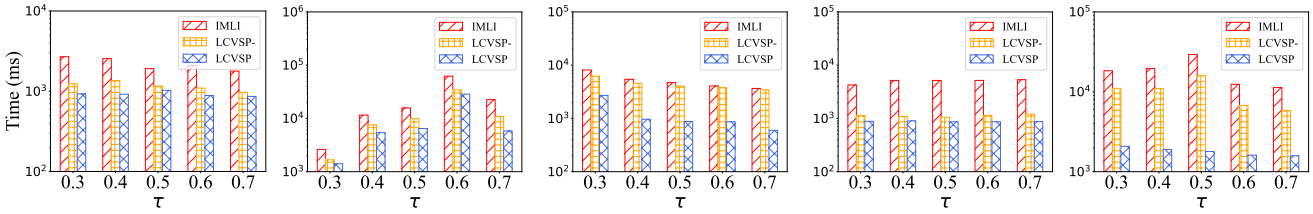
Datasets. We evaluate our methods on 9 real-world multilayer graphs, summarized in Table I. The Homo and SacchCere datasets are genetic interaction networks [23]; Cannes2013 [24], Higgs [23], ObamaInIsrael [23], FriendFeed [25], and FlickrGrowth [26] are social networks, where FlickrGrowth is transformed into a multilayer graph by grouping edges by timestamp. Finally, DBLP[1] and Amazon[2] represent collaboration networks from bibliographic and product co-purchasing domains, respectively.

B. Efficiency Evaluation

Exp-1: Runtime of k-dense sugbraph search algorithms. To evaluate the efficiency of our proposed algorithms, we compare LCVSP (Algorithm 3), LCVSP- (*i.e.*,LCVSP without $\mathbf{k}$-core pruning) and IMLI (Algorithm 2) on five large multilayer graphs, by varying the threshold parameter $\tau \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. For each value of τ, the corresponding density vector $\mathbf{k}$ is set as $k_\ell = \lfloor \tau \cdot \rho^*(G_\ell) \rfloor$. As shown in Fig. 5, IMLI generally incurs higher runtime. In contrast, LCVSP- significantly outperforms IMLI on most datasets, benefiting from the vertex set propagation technique that reduces redundant computation across layers. Building upon this, LCVSP further improves efficiency by incorporating an initial $\mathbf{k}$-core pruning, which eliminates globally unpromising vertices. For example, on the FG dataset, LCVSP completes in 1,789 ms, while IMLI takes 29,330 ms—resulting in a 16.39× speed-up. Overall, LCVSP achieves the best runtime performance among all methods.

Exp-2: Runtime of various denstiy decomposition algorithms. We compare the runtime of three density decomposition algorithms. As illustrated in Fig. 6, VSHT exhibits the highest runtime across all datasets. HPDD and HPDD+ consistently outperform the baseline VSHT algorithm across all datasets, with more pronounced speedup on larger and denser graphs such as FF and FG datasets. The HPDD algorithm achieves a 1.5× to 2× speedup over VSHT on most datasets, validating the effectiveness of its hierarchical progressive pruning strategy in reducing unnecessary computation. HPDD+ further accelerates the process using a divide-and-conquer approach, achieving up to a 4× speedup on the FF dataset. These empirical findings are consistent with theoretical expectations, as the enumeration of $D_{\mathbf{k}}$ remains the core computational bottleneck. These results underscore that advanced pruning and progressive decomposition strategies, even when the worst-case complexity remains unchanged, offer tangible benefits for multilayer dense subgraph discovery.

[1]https://dblp.uni-trier.de/xml/
[2]http://snap.stanford.edu/data/



(a) OI (b) AM (c) HI (d) FF (e) FG

Fig. 5. Runtime of different $\mathbf{k}$-dense sugbraph search algorithms (vary τ)

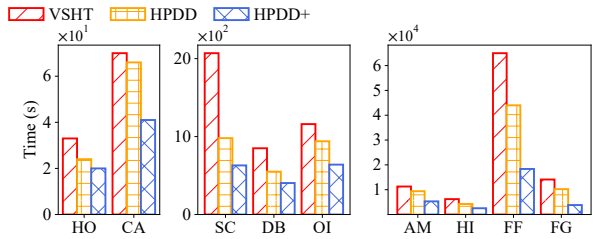


Fig. 6. Runtime of density decomposition algorithms

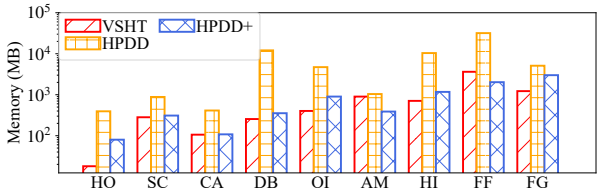
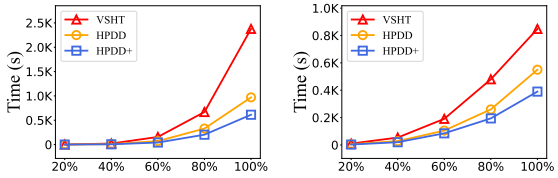
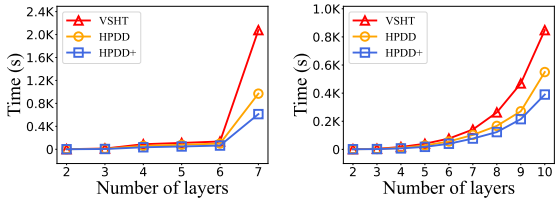


Fig. 7. Memory costs of density decomposition algorithms



(a) SC (b) DB

Fig. 8. Scalability testings (vary $|V|$)



(a) SC (b) DB

Fig. 9. Runtime of density decomposition algorithms w.r.t. the number of layers in the input multilayer graph.

Exp-3: Memory costs of density decomposition algorithms. In terms of memory usage of our proposed algorithms, the results are depicted in Fig. 7. VSHT consistently consumes the least memory among most datasets. In contrast, HPDD incurs the highest memory overhead across all datasets. The reason is that it requires storing a large number of intermediate subgraphs and density vector states during computation, which are consistent with the theoretical analysis of the space complexity of the algorithm. For instance, on the FF dataset, HPDD's memory cost exceeds 30GB, while VSHT and HPDD+ remain below 3GB. Despite the memory efficiency of VSHT, it suffers from long runtimes as previously shown. In this regard, HPDD+ strikes a balance: it achieves a runtime that is significantly better than both VSHT and HPDD, while maintaining memory cost comparable to VSHT. This demonstrates the advantage of the divide-and-conquer technique, HPDD+ also provides a memory-efficient solution for density decomposition.

Exp-4: Scalability testings. First, we study how the algorithms scale with the number of vertices. To this end, we randomly sample 20% to 80% of vertices from the original multilayer graphs (SC and DB), and measure runtime on the induced subgraphs. Actually, similar findings are observed on the remaining datasets as well. As shown in Fig. 8, all algorithms require more running time as the vertex set $|V|$ increase. However, both HPDD and HPDD+ consistently outperform VSHT, and the gap widens on larger graphs. Notably, the runtime incurred by HPDD+ shows a steady increase, whereas VSHT demonstrates a substantially steeper increase in runtime. This trend is consistent across datasets with different vertex scales, highlighting the superior scalability and efficiency of HPDD+.

Second, we evaluate scalability with respect to the number of layers. Specifically, we incrementally construct multilayer graphs by selecting the first ℓ layers ($2 \leq \ell \leq |L|$) from the SC and DB datasets. Fig. 9 illustrates that the runtime of all algorithms grows rapidly with the increase in the number of layers, reflecting the exponential complexity associated with multilayer dense subgraph enumeration. Interestingly, a sharp runtime increase is observed for all methods when transitioning from 6 to 7 layers on the SC dataset, due to increased cohesiveness in $G_7[V]$. Despite this, HPDD and especially HPDD+ exhibit significantly better scalability. The runtime gap between VSHT and the two advanced algorithms becomes pronounced as the number of layers increases.

C. Density Decomposition Characterization

Exp-5: Comparison between computed and actual $D_{\mathbf{k}}$s. To evaluate the redundancy in the density decomposition, we compare the number of $D_{\mathbf{k}}$ subgraphs computed by three algorithms with the actual number of distinct and non-empty subgraphs. As shown in Fig. 10, for all datasets, the number of computed $D_{\mathbf{k}}$s significantly exceeds the actual count. This gap reflects two intrinsic issues, (1) many density vectors $\mathbf{k}$ lead to empty subgraphs, and (2) multiple non-maximal density vectors may induce the same vertex set, as captured by the Definition 2. Among all methods, VSHT consistently produces

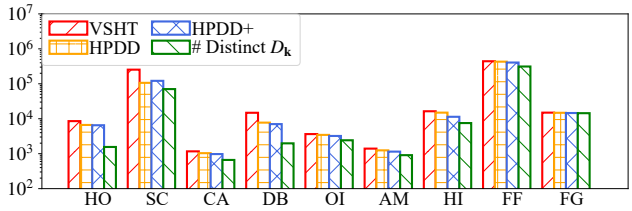
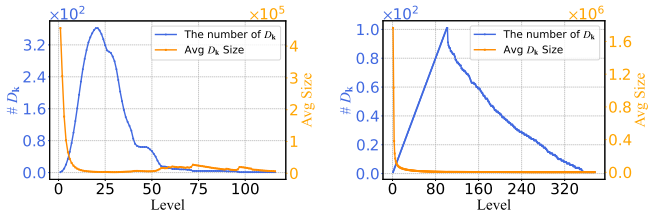


Fig. 10. Comparison between computed and actual distinct $D_{\mathbf{k}}$ across algorithms.



(a) HI (b) FG

Fig. 11. Level-wise distribution of the number and average size of $D_{\mathbf{k}}$s in the density lattice

the largest number of redundant results, while HPDD shows moderate improvement. Notably, HPDD+ yields computed subgraph counts that closely match the ground truth across datasets. This observation aligns with earlier experiments where HPDD+ demonstrated the highest efficiency.

Exp-6: Distribution of $D_{\mathbf{k}}$ along the density lattice. We analyze the structural characteristics of the density lattice by examining the distribution of $D_{\mathbf{k}}$s across lattice levels. As seen in Fig. 11, despite the different datasets, both HI and FG exhibit consistent trends. The number of $D_{\mathbf{k}}$s per level grows rapidly and reaches its peak within the first 25–30% of the lattice depth, where the density vectors contain mostly small entries (*i.e.*, k_ℓ). After reaching the peak, the number of $D_{\mathbf{k}}$s gradually declines as the density constraints become more restrictive. In contrast, the average size of $D_{\mathbf{k}}$s follows an inverse trend. $D_{\mathbf{k}}$s at the lower levels contain a large portion of the original nodes, while those at the higher levels shrink to less than 1% of the initial size, often comprising a few nodes.

D. Effectiveness Evaluation

We evaluate the effectiveness of the proposed $D_{\mathbf{k}}$ model by comparing it with 5 representative multilayer cohesive subgraph models, namely **k**-core [10], [20], d-CC [11], [27], FirmCore [12], FocusCore [13], and FirmTruss [14].

Exp-7: Quality of subgraphs discovered by different models. To further assess the effectiveness of the $D_{\mathbf{k}}$ model beyond density, we evaluate the quality of the densest subgraphs (with $\beta = 1$) discovered by each method using two widely-used metrics: the *global clustering coefficient* (GCC) [28] and *conductance* (CDT) [29]. GCC measures the internal connectivity within a subgraph. For each layer $\ell \in L$, $\text{GCC}_\ell = \frac{3\times\#\text{ triangles in } G_\ell[D]}{\#\text{ connected triples in } G_\ell[D]}$. A higher GCC indicates a more cohesive structure. CDT measures separation from the rest of the graph: $\text{CDT}_\ell = \frac{|E_\ell(D, V\setminus D)|}{\sum_{v\in D} d_\ell(v)}$. Lower CDT values imply stronger structural separation. We report the minimum, maximum, and average values of each metric across layers (e.g., $\text{GCC}_{\min} = \min_{\ell\in L}\text{GCC}_\ell, \text{GCC}_{\max} = \max_{\ell\in L}\text{GCC}_\ell$, $\text{GCC}_{\text{avg}} = \frac{1}{|L|}\sum_{\ell\in L}\text{GCC}_\ell$, and similarly for CDT)

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT METRICS ACROSS VARIOUS DATASETS. (METRICS: I: $\text{GCC}_{\min}$, II: $\text{GCC}_{\max}$, III: GCC_{avg}, IV: $\text{CDT}_{\min}$, V: $\text{CDT}_{\max}$, VI: CDT_{avg})

Dataset	SC						OI						HI					
Metric	i	ii	iii	iv	v	vi	i	ii	iii	iv	v	vi	i	ii	iii	iv	v	vi
k-core	0.11	0.38	0.21	0.54	0.87	0.75	0.03	0.12	0.08	0.90	0.91	0.91	0.01	0.14	0.05	0.83	0.95	0.90
FirmCore	0.09	0.23	0.14	**0.35**	**0.60**	**0.46**	0.03	0.11	0.07	0.87	0.91	0.89	0.01	0.14	0.05	0.83	0.95	0.90
d-CC	0.11	0.38	0.21	0.54	0.87	0.75	0.02	0.11	0.07	0.89	0.89	0.89	0.01	0.14	0.05	0.83	0.95	0.90
FocusCore	0.11	0.38	0.21	0.54	0.87	0.75	0.02	0.11	0.07	0.89	0.89	0.89	0.01	0.14	0.05	0.83	0.95	0.90
FirmTruss	0.03	**0.77**	0.23	0.84	0.98	0.94	0.04	0.12	0.10	0.91	0.92	0.92	0.02	**0.35**	0.13	0.87	0.96	0.93
$D_{\mathbf{k}}$	**0.18**	0.52	**0.31**	0.37	0.77	0.61	**0.05**	**0.14**	**0.11**	**0.81**	**0.83**	**0.82**	**0.02**	0.22	**0.15**	**0.70**	**0.90**	**0.81**

TABLE III
DENSITY OF SUBGRAPHS DISCOVERED BY DIFFERENT MODELS IN APPROXIMATING THE MULTILAYER DENSEST SUBGRAPH WITH VARYING β VALUES

Dataset	Model	$\beta=1$	$\beta=2$	$\beta=3$	$\beta=4$	$\beta=5$	$\beta=6$
SC	**k**-core	52.60	186.26	931.37	4556.73	26144.22	183009.55
	FirmCore	47.60	**190.41**	786.19	3930.99	19688.21	137817.40
	d-CC	**52.60**	172.93	864.68	4323.44	21617.22	143189.42
	FocusCore	**52.60**	**190.41**	884.83	4424.15	22120.78	144169.39
	FirmTruss	32.65	80.02	400.11	2000.53	10002.69	58312.98
	$D_{\mathbf{k}}$	**52.60**	186.26	**933.34**	**4656.73**	**29505.62**	**206539.35**
OI	**k**-core	**43.79**	**87.58**	207.54	628.42	1885.27	5655.83
	FirmCore	42.17	84.35	168.70	412.10	1236.32	3708.96
	d-CC	43.77	87.55	207.46	622.38	1867.15	5601.46
	FocusCore	43.77	87.55	207.46	622.38	1867.15	5601.46
	FirmTruss	14.22	28.44	56.88	162.00	486.00	1458.00
	$D_{\mathbf{k}}$	**43.79**	**87.58**	**209.32**	**629.36**	**1887.73**	**5664.25**
HI	**k**-core	113.51	113.51	316.96	950.50	2852.71	9436.86
	FirmCore	113.49	113.49	291.02	873.07	2619.22	7857.66
	d-CC	113.51	113.51	315.58	946.75	2840.27	9420.80
	FocusCore	113.51	113.51	315.58	946.75	2840.27	9436.86
	FirmTruss	74.35	76.36	229.09	687.27	2061.81	6185.45
	$D_{\mathbf{k}}$	**114.02**	**114.02**	**317.04**	**951.12**	**2853.37**	**9528.58**

The results are summarized in Table II. Across all datasets, $D_{\mathbf{k}}$ consistently achieves the highest average GCC and lowest average CDT across datasets, indicating that its subgraphs are both cohesive and well-separated. While some baselines (e.g., FirmTruss) may excel in individual layers, they often show imbalance across layers. In contrast, $D_{\mathbf{k}}$ maintains strong performance on both metrics, demonstrating its ability to discover well-rounded and structurally robust subgraphs.

Exp-8: Approximation of the multilayer densest subgraph. To assess how well each model approximates the densest subgraph, we adopt the widely used multilayer densest subgraph definition [10], [20], which aims to find a vertex subset $S^* \subseteq V$ maximizing the density objective: $\rho(S) = \max_{\hat{L}\subseteq L} \min_{\ell\in\hat{L}} \frac{|E_\ell[S]|}{|S|} \cdot |\hat{L}|^\beta$, where $\beta > 0$ balances the trade-off between density and layer coverage. A smaller β emphasizes subgraph cohesiveness, while a larger β favors support across more layers.

For each model, we compute the densest subgraph it discovers, and report its $\rho(S)$ value for β ranging from 1 to 6. As shown in Table III, the proposed $D_{\mathbf{k}}$ model consistently achieves the highest or near-highest density across all settings, with its advantage becoming more pronounced as β increases. While baseline models perform similarly under loose constraints, their quality degrades as the requirement for multilayer support intensifies. In contrast, $D_{\mathbf{k}}$ maintains high-quality results even under stringent conditions—for instance,

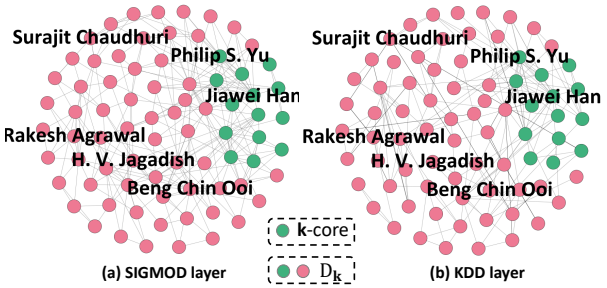


Fig. 12. Community comparison between $\mathbf{k}$-core and $D_{\mathbf{k}}$ across layers

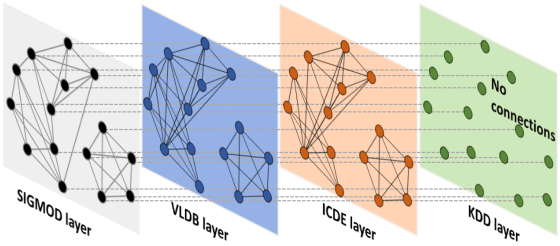


Fig. 13. Cross-layer community distribution derived from FocusCore

on the SC dataset with $\beta = 6$, it outperforms the next-best model by over 40%. These results highlight the robustness and adaptability of $D_{\mathbf{k}}$ in capturing cohesive structures, demonstrating its practical effectiveness over existing alternatives.

E. Case Studies.

Exp-9: Cross-domain community detection on DBLP**.** We perform a case study on DBLP dataset to uncover cross-domain research communities. We construct a four-layer graph where each node represents a researcher and edges denote co-authorships. The layers correspond to SIGMOD, VLDB, ICDE (database systems), and KDD (data mining), enabling the analysis of collaboration across domains. To ensure fair comparison, we apply $D_{\mathbf{k}}$ with density vector $\mathbf{k} = [3, 2, 3, 2]$, and run $\mathbf{k}$-core under the same parameters. FocusCore and d-core (s = 3 and $k = 4$) focus only on the first three layers (SIGMOD, VLDB and ICDE). FirmCore either produces excessively large communities (383 researchers) or returned no results with stricter parameters, and is omitted for clarity.

Our $D_{\mathbf{k}}$ discovers a well-balanced community of 77 researchers, as shown in Fig. 12 (VLDB and ICDE layers exhibit similar structural patterns to SIGMOD, thus omitted for clarity). This community includes prominent bridge researchers between database and data mining domains, such as "Jiawei Han", "Philip S. Yu", "Beng Chin Ooi", "Surajit Chaudhuri", "H. V. Jagadish", and "Rakesh Agrawal". In contrast, the $\mathbf{k}$-core identifies only 14 researchers, which is a strict subset of our community—with lower achieved density with $[2, 1, 2, 2]$.

FocusCore model produces structurally flawed results. As shown in Fig. 13, it generates disconnected components across all database layers (SIGMOD, VLDB, ICDE) due to its stricter constraints. More critically, in the KDD layer, researchers appear as completely isolated nodes without any connections, failing to capture cross-domain collaboration patterns.

Exp-10: Multilayer community detection in Twitter during the NBA Finals. We conduct a case study on a Twitter dataset collected during the 2015 NBA Finals [30], which includes three types of user interactions—retweets, mentions, and replies—forming a natural three-layer graph. To ensure fair comparison, all models are configured to emphasize cross-layer cohesiveness. For d-CC, FirmCore, and FocusCore, we set $\lambda = 3$ to require support on all layers and select the maximum k yielding non-empty outputs ($k = 4$). For $\mathbf{k}$-core and $D_{\mathbf{k}}$, we set $\mathbf{k} = [5, 5, 3]$, representing the highest consistent thresholds supported across all layers.

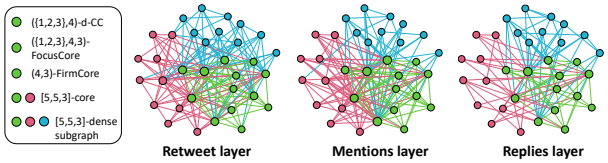


Fig. 14. Communities discovered by different models on the Twitter NBA dataset

The resulting communities are visualized in Fig. 14, which highlights a clear contrast between the models. The subgraph extracted by our $D_{\mathbf{k}}$ forms a densely connected community across all layers. In comparison, the communities produced by d-CC, FirmCore, and FocusCore are identical and strictly contained within the $D_{\mathbf{k}}$ community. Similarly, the $\mathbf{k}$-core community is also a proper subset of $D_{\mathbf{k}}$. In terms of in-layer edge density, the $D_{\mathbf{k}}$ community achieves [5.10, 5.03, 3.03] on the three layers respectively, compared to [4.65, 4.95, 2.73] for $\mathbf{k}$-core and [3.92, 4.21, 2.64] for d-CC/FirmCore/FocusCore. This demonstrates that $D_{\mathbf{k}}$ consistently produces the densest community on all layers.

Unlike degree-based models, which often retain loosely connected nodes due to local thresholds, $D_{\mathbf{k}}$ captures a well-connected group that meets prescribed density constraints in every layer. This ability to enforce global structural balance allows our model to uncover more meaningful and cohesive communities in multilayer networks.

VI. Related Work

Cohesive Subgraph Models in Multilayer Graphs. Prior studies on multilayer graphs have primarily focused on extracting structurally cohesive subgraphs. Jethava *et al.* [31] formulated the densest common subgraph problem and proposed a linear programming-based solution. Azimi *et al.* [32] introduced the notion of multilayer $\mathbf{k}$-core, which was later extended by Galimberti *et al.* [10], [20] through algorithms for enumerating all valid $\mathbf{k}$-cores. Liu *et al.* [11] studied the CoreCube problem, which explores subgraphs where each node satisfies uniform degree constraints on selected layers—a special case of the general $\mathbf{k}$-core model. Zhu *et al.* [27] addressed the diversification of such subgraphs, while Hashemi *et al.* [12] and Wang *et al.* [13] introduced the FirmCore and FocusCore models to improve robustness and adaptability. Except core-based subgraph decomposition on multilayer graphs, truss-based models offer another direction. Huang *et al.* [33] proposed TrussCube, identifying subgraphs where each edge maintains sufficient support across layers. FirmTruss [14] further generalized this concept by aggregating edge support across multiple layers. While these approaches capture certain forms of cohesiveness, they are not designed to model subgraph density in a strict sense. In contrast, our

model enforces explicit density lower bounds for each layer, thereby proving the cohesiveness consistency across all layers.

Dense Subgraph Models. In unipartite graphs, Tatti *et al.* [34] introduced a locally-dense decomposition framework based on densest subgraphs, enabling the identification of all locally dense regions. Danish *et al.* [35] further accelerated this process by leveraging convex optimization techniques. Qin *et al.* [36] proposed the notion of locally densest subgraphs to capture the densest subgraph structures across different graph regions. Borradaile *et al.* [17] formalized density decomposition as a comprehensive framework for dissecting graph density structures. Zhang *et al.* [37] further accelerated this process by using reorientation network. Additionally, Zhang *et al.* [38] also proposed density decomposition for bipartite graphs. While these methods provide valuable insights and tools for dense subgraph analysis, they primarily focus on single-layer graphs and bipartite graphs. To the best of our knowledge, no prior work has extended density decomposition models to multilayer graphs. Our research addresses this gap by proposing a novel density decomposition framework tailored for multilayer graphs.

VII. Conclusion

In this paper, we propose a novel $\mathbf{k}$-dense subgraph model for multilayer graphs, which enforces explicit layer-wise density constraints. Building upon this model, we study the problem of multilayer density decomposition. For a single $D_{\mathbf{k}}$ computation, we design the LCVSP framework, which combines network flow techniques, core-based pruning, and layer-constrained propagation for fast convergence. For full decomposition, we present the HPDD and HPDD+ algorithms, leveraging the density lattice structure to enable scalable and memory-efficient exploration of the solution space. Extensive experiments demonstrate that our model consistently discovers higher-quality subgraphs compared to other models, while our algorithms achieve superior efficiency and scalability.

References

[1] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer social networks*. Cambridge University Press, 2016.

[2] S. Chen, H. Qian, Y. Wu, C. Chen, and X. Wang, "Efficient adoption maximization in multi-layer social networks," in *2019 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2019, pp. 56–60.

[3] B. Zhao, S. Hu, X. Li, F. Zhang, Q. Tian, and W. Ni, "An efficient method for protein function annotation based on multilayer protein networks," *Human genomics*, vol. 10, pp. 1–15, 2016.

[4] X. Liu, E. Maiorino, A. Halu, K. Glass, R. B. Prasad, J. Loscalzo, J. Gao, and A. Sharma, "Robustness and lethality in multilayer biological molecular networks," *Nature communications*, vol. 11, no. 1, p. 6043, 2020.

[5] B. Han, Y. Wei, Q. Wang, F. M. D. Collibus, and C. J. Tessone, "Mt 2 ad: multi-layer temporal transaction anomaly detection in ethereum networks with gnn," *Complex & Intelligent Systems*, vol. 10, no. 1, pp. 613–626, 2024.

[6] X. Huang, D. Chen, T. Ren, and D. Wang, "A survey of community detection methods in multilayer networks," *Data Mining and Knowledge Discovery*, vol. 35, pp. 1–45, 2021.

[7] X. Du, R. Jin, L. Ding, V. E. Lee, and J. H. Thornton Jr, "Migration motif: a spatial-temporal pattern mining approach for financial markets," in *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, 2009, pp. 1135–1144.

[8] A. F. Colladon and E. Remondi, "Using social network analysis to prevent money laundering," *Expert Systems with Applications*, vol. 67, pp. 49–58, 2017.

[9] H. Zhang, C. Zhuge, and X. Yu, "Identifying hub stations and important lines of bus networks: A case study in xiamen, china," *Physica A: Statistical Mechanics and Its Applications*, vol. 502, pp. 394–402, 2018.

[10] E. Galimberti, F. Bonchi, and F. Gullo, "Core decomposition and densest subgraph in multilayer networks," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1807–1816.

[11] B. Liu, F. Zhang, C. Zhang, W. Zhang, and X. Lin, "Corecube: Core decomposition in multilayer graphs," in *Web Information Systems Engineering–WISE 2019: 20th International Conference, Hong Kong, China, January 19–22, 2020, Proceedings 20*. Springer, 2019, pp. 694–710.

[12] F. Hashemi, A. Behrouz, and L. V. Lakshmanan, "Firmcore decomposition of multilayer networks," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1589–1600.

[13] R.-A. Wang, D. Liu, and Z. Zou, "Focuscore decomposition of multilayer graphs," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 2792–2804.

[14] A. Behrouz, F. Hashemi, and L. V. Lakshmanan, "Firmtruss community search in multilayer networks," *arXiv preprint arXiv:2205.00742*, 2022.

[15] J. Jiang, R.-H. Li, Y. Zhang, and G. Wang, "Density decomposition of multilayer graphs (full version)," 2025. [Online]. Available: https://github.com/jiangjiaqi6/density-decomposition-multilayer-graph/blob/master/Density_decomposition_of_multilayer_graphs_fullversion.pdf

[16] A. V. Goldberg, "Finding a maximum density subgraph," 1984.

[17] G. Borradaile, T. Migler, and G. Wilfong, "Density decompositions of networks," in *Complex Networks IX: Proceedings of the 9th Conference on Complex Networks CompleNet 2018 9*. Springer, 2018, pp. 15–26.

[18] I. Bezáková, "Compact representations of graphs and adjacency testing," 2000.

[19] M. Blumenstock, "Fast algorithms for pseudoarboricity," in *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, M. T. Goodrich and M. Mitzenmacher, Eds. SIAM, 2016, pp. 113–126.

[20] E. Galimberti, F. Bonchi, F. Gullo, and T. Lanciano, "Core decomposition in multilayer networks: Theory, algorithms, and applications," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 14, no. 1, pp. 1–40, 2020.

[21] V. Batagelj and M. Zaversnik, "An o (m) algorithm for cores decomposition of networks," *arXiv preprint cs/0310049*, 2003.

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

[23] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer Social Networks*. Cambridge University Press, 2016.

[24] E. Omodei, M. De Domenico, and A. Arenas, "Characterizing interactions in online social networks during exceptional events," *Frontiers in Physics*, vol. 3, p. 59, 2015.

[25] J. A. Baggio, S. B. BurnSilver, A. Arenas, J. S. Magdanz, G. P. Kofinas, and M. De Domenico, "Multiplex social ecological network analysis reveals how social changes affect community robustness more than resource depletion," *Proceedings of the National Academy of Sciences*, vol. 113, no. 48, pp. 13 708–13 713, 2016.

[26] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Growth of the flickr social network," in *Proceedings of the first workshop on Online social networks*, 2008, pp. 25–30.

[27] R. Zhu, Z. Zou, and J. Li, "Diversified coherent core search on multilayer graphs," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 701–712.

[28] R. D. Luce and A. D. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949.

[29] M. A. Reed, C. Zhou, C. Muller, T. Burgin, and J. Tour, "Conductance of a molecular junction," *Science*, vol. 278, no. 5336, pp. 252–254, 1997.

[30] M. De Domenico and E. G. Altmann, "Unraveling the origin of social bursts in collective attention," *Scientific reports*, vol. 10, no. 1, p. 4629, 2020.

[31] V. Jethava and N. Beerenwinkel, "Finding dense subgraphs in relational graphs," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 641–654.

[32] N. Azimi-Tafreshi, J. Gómez-Gardenes, and S. Dorogovtsev, "k-core percolation on multiplex networks," *Physical Review E*, vol. 90, no. 3, p. 032816, 2014.

[33] H. Huang, Q. Linghu, F. Zhang, D. Ouyang, and S. Yang, "Truss decomposition on multilayer graphs," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5912–5915.

[34] N. Tatti and A. Gionis, "Density-friendly graph decomposition," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1089–1099.

[35] M. Danisch, T.-H. H. Chan, and M. Sozio, "Large scale density-friendly graph decomposition via convex programming," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 233–242.

[36] L. Qin, R.-H. Li, L. Chang, and C. Zhang, "Locally densest subgraph discovery," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 965–974.

[37] Y. Zhang, R.-H. Li, Q. Zhang, H. Qin, and G. Wang, "Efficient algorithms for density decomposition on large static and dynamic graphs," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 2933–2945, 2024.

[38] Y. Zhang, R.-H. Li, Q. Zhang, H. Qin, L. Qin, and G. Wang, "Density decomposition of bipartite graphs," *Proceedings of the ACM on Management of Data*, vol. 3, no. 1, pp. 1–25, 2025.