Efficient Cross-layer Community Search in Large Multilayer Graphs

Longxu Sun
Hong Kong Baptist University
Hong Kong, China
cslxsun@comp.hkbu.edu.hk

Xin Huang
Hong Kong Baptist University
Hong Kong, China
xinhuang@comp.hkbu.edu.hk

Zheng Wu
Hong Kong Baptist University
Hong Kong, China
cszhengwu@comp.hkbu.edu.hk

Jianliang Xu
Hong Kong Baptist University
Hong Kong, China
xujl@comp.hkbu.edu.hk

***Abstract*—Community search is a query-dependent graph task to find communities containing a given set of query vertices, which is useful for personalized search and recommendation. Recently, community search over multilayer networks has gained attention thanks to its strong ability to capture cross-layer relationships among diverse entities from multiple domains. This brings significant advantages against the classical studies of community search over only single-layer graphs. However, most existing multilayer community models suffer from two major limitations: 1) failure to identify informative communities with the most layers when a multilayer graph is associated with a large number of layers; 2) missing to distinguish the degree of connections in internal layers and cross-layers.**

To tackle the above limitations, this paper proposes a novel multilayer subgraph model called (k, d)-core. A (k, d)-core based community requires that every two layers have enough k internal layer connections and d cross-layer connections for each vertex in this community. We formulate the problem of multilayer community search (MCS-problem), which finds a (k, d)-core connected subgraph H containing query vertices to achieve the largest number of cross-layers. For cross-layer connectivity, we consider two-fold definitions of *full-layer* and *path-layer* connectivities. First, we consider a strong definition of full-layer connectivity, which constrains that every two layers are connected in H. We show that the MCS-problem under full-layer connectivity is NP-hard. We propose two methods of exact exploration and heuristic search for finding MCS answers. Second, to improve the efficiency of community search, we further study a relaxation of *path-layer* connectivity, allowing two layers to be connected via a path of immediate layers. Then, we develop a fast search algorithm to identify path-layer-based communities and then refine them to full-layer answers. Furthermore, we develop a novel (k, d)-core index that effectively captures essential (k, d)-core structure, including the neighborhood information, the layer connectivities, and the internal/cross-layer corenesses. Extensive experiments on nine real-world multilayer graphs demonstrate the effectiveness and efficiency of our MCS model and algorithms.

I. Introduction

Graph is a mathematical model widely used to represent entities and their relationships in real-world scenarios, such as social networks, biological networks, financial networks, brain networks, transportation systems, and so on [1]–[3]. However, a simple single-layer graph model is often hard to depict various kinds of entities and the complex characteristics of relationships, which brings significant challenges in comprehensive graph data analytics [4], [5]. Multilayer graph (MG) is an advanced graph model that consists of multiple layers, where each layer represents a different type of relationship or interaction between nodes. Multilayer graphs have many real-world applications in society, finance, and biology. For instance, a multilayer social network can represent different types of relationships between individuals, such as friendships, professional connections, family ties, and online interactions. Each layer captures a different aspect of the social network. Thus, the integrated information of all layers offers a comprehensive social overview. In biology, multilayer graphs are used to represent complex interactions within biological systems. A multilayer graph can model protein-protein interactions, gene regulatory networks, and metabolic pathways. Each layer captures a specific aspect of the biological system, providing insights into its layered structure and diverse functions. A multilayer financial graph can model complex interactions between financial entities, such as banks and companies associated with stock exchanges and financial transactions. Fig. 1(a) shows a collaboration network of multilayer graph MG, associated with four different layers of published paper layer in G_1, research expertise layer in G_2, interest group layer in G_3, and scholar collaboration layer in G_4.

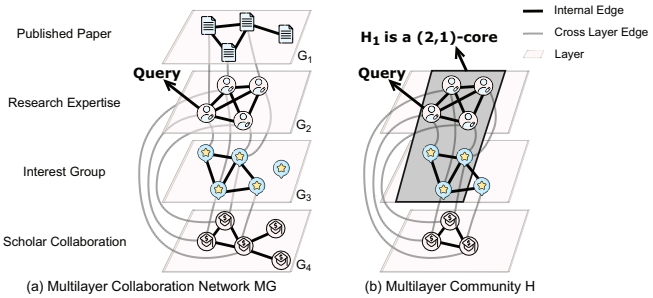


Fig. 1. An example of community search in a multilayer collaboration network with four layers of published paper layer, research expertise layer, interest group layer, and scholar collaboration layer.

Several studies work on multilayer community detection, which identifies all communities with cohesive multilayer connectivity and attribute similarity [6]–[10]. Different from community detection, community search is a query-oriented task to uncover highly personalized and densely interconnected communities containing the given query vertices [11]–[14]. The applications of community search are demonstrated, e.g., influential communities in social networks, determining efficient travel routes in transportation networks, and uncovering relationships between genes and proteins in biological net-

2375-026X/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDE60146.2024.00230

works [15]–[18]. However, extending the existing techniques of multilayer community detection to community search is difficult, which desires new models and efficient solutions.

On the other hand, existing community search over multilayer networks [12]–[14] also suffers from several limitations. First, the models [12]–[14] are developed upon *multiplex graphs*, which is a special instance of multilayer graphs requiring that vertices in various layers are identical. Extending the models and algorithms to general multilayer graphs is challenging, where vertices at each layer represent different entities of distinct types. Another variant of MLG called *heterogeneous information network (HIN)* [19]–[22] focuses on meta-path-based models that mainly consider cross-layer relationships and ignore internal layer connections. Second, existing studies [23], [24] extend from single-layer graphs to handle multiple layers, which may lead to a significantly increased complexity with the increased number of layers. Scalable algorithms and techniques are needed to handle a multilayer graph with a large number of layers. Third, there is an inflexible density balance between the internal layer and cross-layer. Existing community models [12], [14] give a hard constraint for layer density failing to balance between internal and cross-layer densities, which may result in an inflexible model to fit more real communities. Thus, defining a reasonable density measure to capture the strength of internal layer and cross-layer connections is a challenging but important task.

To address the above limitations, we propose a novel multilayer community search model. Specifically, we design a flexible community model called (k, d)-*core* in general MGs by considering the connections of internal layers and cross-layers, respectively. k-core is a fundamental dense subgraph model in a single-layer homogeneous graph, which requires each vertex in k-core to have at least k neighbors. We extend k-core to (k, d)-core in multilayer graphs. In the (k, d)-core, every pair of layers requires the vertices to have at least k internal edges in its layer and at least d cross-layer edges to vertices in another layer. Based on the fundamental element of (k, d)-core, we formally formulate the community search problem in multilayer graphs (MCS-problem). Given a multilayer graph MG, a set of query vertices Q, parameters k and d, MCS-problem is to find a connected multilayer subgraph H containing all query vertices Q such that every pair of layers exists a (k, d)-core of H, meanwhile H achieves the largest number of layers indicating the most meaningful cross-layers. Consider an example of collaboration network MG in Fig. 1 with four layers G_1, G_2, G_3, and G_4, the subgraph H_1 is a $(2, 1)$-core located in layers G_2 and G_3, which is depicted in grey region. Each vertex in G_2 has at least 2 neighbors in G_2 and 1 cross-layer neighbor with G_3. In this example, the user queries a collaboration community related to the query author in layer G_2; our problem will return the multilayer community H shown in Fig. 1(b), there exists a $(2, 1)$-core between every two layers, which also achieves the largest number of 3 layers, i.e., G_2, G_3, and G_4.

We theoretically analyze the structural properties of (k, d)-core and also the problem hardness. We show that the MCS-problem is NP-hard, which can be reduced from a classical NP-complete Maximum Clique decision problem. This brings significant challenges to the development of fast multilayer community search algorithms. To tackle it efficiently, we propose a MCS framework. It consists of three steps: (1) extracting k-core components in each layer; (2) checking the existence of (k, d)-core for every two layers; and (3) finding a layer-clique community with the maximum number of layers. Among these three steps, the most time-consuming operations are Steps 2 and 3. We further propose fast techniques to accelerate them respectively. As Step 2 needs to check a total of $\frac{l\times(l-1)}{2}$ for l-layered graph, we design an index that effectively captures essential graph structure features to speed up the search process, including neighborhood information, layer connectivity, and the vertex coreness in internal layers/cross-layers. Leveraging the (k, d)-core index, we can quickly determine if two layers have no (k, d)-core connectivity and safely prune such layer candidates. Due to the problem NP-hardness for checking maximum clique in Step 3, we first propose a greedy algorithm to find heuristic answers directly. Then, we obtain an upper bound of layers and design a bounded pruning algorithm to reduce search space.

We relax the *full-layer* connectivity to the *path-layer* one to help improve the efficiency and obtain the upper bound for Step 3. Specifically, a path-layer-based community does not need that every pair of layers is (k, d)-core connected, but only requires that there exists a path of (k, d)-core connected layers connecting these two layers. Leveraging this, the search for the path-layer community can be done in polynomial time. Moreover, the discovery of path-layer community can be used to accelerate the discovery of full-layer answers further. In summary, we make the following contributions to this paper.

- We propose a novel dense subgraph of (k, d)-core in multilayer graphs, strengthening the connections in internal layers and cross-layers. Based on (k, d)-core, we formulate our new problem of multilayer community search to maximize the number of cross-layers. (Section III)
- We prove the MCS-problem to be NP-hard. We propose an exact enumeration algorithm to find multilayer communities. Furthermore, we develop heuristic techniques, determine an upper bound of community layers, and then design bound-and-search methods to accelerate efficiency. (Section III, IV)
- We study a relaxation problem of multilayer community search, which reduces the full-layer connectivity to path-layer connectivity. We develop a polynomial-time online search algorithm to find path-layer communities and further improve the search for full-layer communities. (Section V)
- In addition, we design a new (k, d)-core index to store all (k, d)-core information and propose an index-based algorithm to speed up community search. (Section VI)
- We conduct extensive experiments to evaluate the effectiveness and efficiency of our full-layer and path-layer community models, the corresponding search al-

gorithms, and the designed index. Experiments show the superiority of our multilayer model against state-of-the-art FirmTruss [14] and RWM [13] on nine real-world datasets. (Section VII)

We discuss related work in Section II and conclude the paper in Section VIII.

II. Related Work

In this section, we review the related studies including *community search*, *multilayer graph analytics*, and *heterogeneous information network analytics*.

Community search. Community search is a query processing task to identify query-oriented densely connected communities in a graph [11], [15]. In the literature, there are several dense subgraph based community search models, including k-core [19], [20], [25], [26], k-truss [14], [27], [28], k-clique [29], k-plex [30], and so on. A cohesive subgraph of k-core represents a group where all vertices have at least k neighbors [25], [26], which are widely used for community models and accelerating clique discovery. A comprehensive survey of community search can be found in [11], [15]. Besides the structural community search over simple graphs, community search has been studied in various graph data, including dynamic graphs [16], [17], labeled graph [31], and heterogeneous information networks [19], [20]. *However, most existing community search studies work on one single-layer graph, but ignore the real-life multilayer interactions.*

Multilayer graph analytics. Several graph analytics studies have focused on multilayer graphs, addressing various tasks such as clustering [32]–[34], graph summarization [2], community detection [6], [7], [10], community search [13], [14] and advanced analytics tasks [1], [3], [35], [36]. SpectralMix has been proposed by using a joint dimensionality reduction technique for multi-relational graphs clustering with categorical node attributes [32]. MultiGBS is a domain-specific summarizer for biomedical data summarization that utilizes the multilayer graph model to help incorporate multiple features of the text simultaneously [2]. The k-core-based model over two-layer networks has been proposed to determine a user community and a location cluster in geo-social networks [37]. A multilayer gCore is developed to find cohesive communities that are densely connected to specified layers [10]. Luo et al. study the community search problem by proposing random walk-based algorithms on multiplex networks where the vertices are the same at different layers [13]. Most recently, FirmTruss is proposed to extend k-truss model to multiplex networks [14]. A multilayer graph analytics survey for community detection can be found in [6]. *Different from most existing community search on multiplex networks, our problem focuses on a more general multilayer graph model meanwhile optimizing the communities' cross-layer interactions.*

Heterogeneous graph analytics. Heterogeneous information networks (HINs) consist of multiple typed nodes and multiple typed links representing various semantic relations. HIN analytics has been studied in dense structure analysis [38]–[40], recommendation systems [41], item detection [42], [43], and community search [19]–[22], [44]. A comprehensive survey of HIN analytics can be found in [45]. Sun et al. propose a graph mining approach to analyze semi-structured and multi-typed HINs [38]. Another line of HIN analytics relevant to ours is community search over HINs. Fang et al. propose a meta-path-based $(k, \mathcal{P})$-core model and design algorithms for community search in HINs [21]. The meta-path-based structure is designed to capture closer relationships among vertices on HINs [19] [20]. Random walk-based community search algorithms are also proposed in HINs [13], [44]. *Although HINs also capture different typed entities and complex relationships as multilayer graphs, most graph analytics tasks need an input of meta-path patterns. A useful meta-path is key for community search to produce high-quality results. However, meta-paths are hard for users to formulate, and our proposed multilayer community search is meta-path-free.*

TABLE I
Frequently used notations

Notation	Description
$\mathsf{MG}\ (V_M, E_M, \mathcal{L})$	A multilayer graph MG
l	The number of layers in MG
$\mathcal{L}(H)$	The set of layer types in H
$\lvert\mathcal{L}(H)\rvert$	The total number of layers in H
$deg_H(v)$	The degree of v in subgraph H
$G_i(V_i, E_i)$	The i-th induced layer
$G_{ij}(G_i, G_j, E_{ij})$	The induced subgraph cross i-th and j-th layers
$H_i(V_i, E_i)$	The induced subgraph of H in i-th layer
$H_{ij}(H_i, H_j, E_{ij}^H)$	The induced subgraph cross i-th and j-th layers
$H_i \overset{H}{\longleftrightarrow} H_j$	Strong cross-layer connectivity SLC
$H_i \overset{H}{\rightsquigarrow} H_j$	Path layer connectivity PLC
$\Phi((u, v))$	(k, d)-coreness of cross-layer edge (u, v)

III. Preliminaries

In this section, we first introduce multilayer graphs (MGs) and a novel definition of multilayer dense subgraph as (k, d)-core. Next, we formulate the problem of multilayer community search, and provide the problem analysis and extensions.

A. Multilayer Graph and (k, d)-Core

A *simple graph* $G(V, E)$ is formed by the vertices V and the edges $E \subset V \times V$ where all vertices and edges belong to the same type, regarded as a single-layer graph [15]. For a subgraph $H \subseteq G$, we represent the neighbors of vertex $v \in V(H)$ as $N_H(v) = \{u \in V : (v, u) \in E\}$ and the degree as $\deg_H(v) = |N_H(v)|$ in subgraph H.

Multilayer graphs. A *multilayer graph* (MG) consists of multiple graphs in different layers [1], [32], [46]. Assume that MG has a total of $l \in \mathbb{Z}^+$ layers and a layer function $\mathcal{L}$ to identify the specifically located layer of vertex v by $\mathcal{L}(v) \in \mathcal{L}(\mathsf{MG}) = \{i \in \mathbb{Z}^+ : 1 \leq i \leq l\}$. A multilayer graph is denoted as $\mathsf{MG} = \{G_i(V_i, E_i) : 1 \leq i \leq l\} \cup \{E_{ij} \subseteq V_i \times V_j : 1 \leq i < j \leq l\}$ [1]. At each layer, the graph G_i has the same typed vertices V_i, where each vertex may be associated with two kinds of edges (*internal edges* and *cross-layer edges*). For an internal edge $e = (v, u) \in E_i$, two

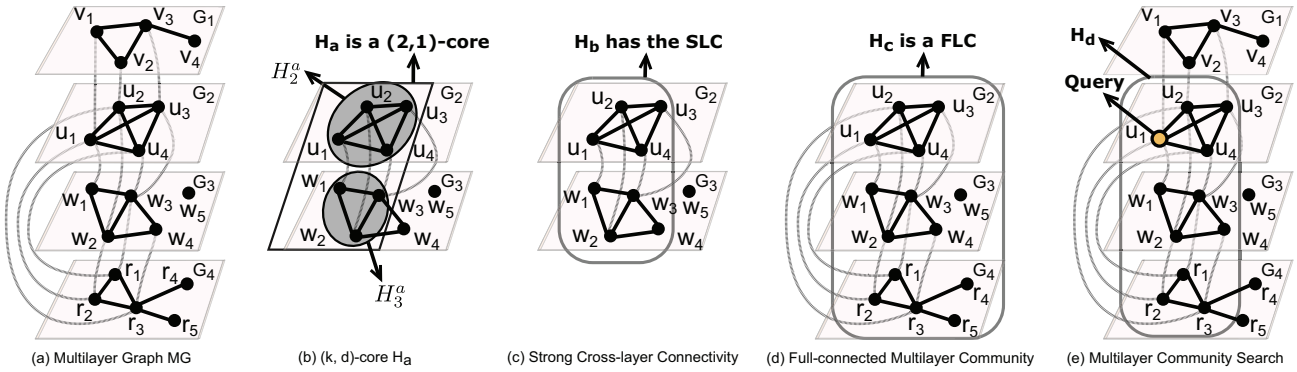


Fig. 2. Examples of multilayer (k, d)-core and multilayer community search. Here $Q = \{u_1\}$, $k = 2$, and $d = 1$.

endpoints v, u are the same typed vertices at one layer graph, i.e., $\mathcal{L}(v) = \mathcal{L}(u)$. For a cross-layer edge $e_{ij} = (v, u) \in E_{ij}$, the vertices $v \in V_i$ and $u \in V_j$ belong to two different i-th and j-th layers, respectively, indicating $\mathcal{L}(v) \neq \mathcal{L}(u)$. The vertices are disjoint, i.e., $V_i \cap V_j = \emptyset$. Finally, we can model a mutilayer graph as $\mathsf{MG}(V_M, E_M, \mathcal{L})$, where $V_M = \bigcup_{i=1}^{l} V_i$, $E_M = \bigcup_{i=1}^{l}\{E_i \cup \bigcup_{i<j\leq l} E_{ij}\}$, and a function of layer identification $\mathcal{L}(.) \in \mathcal{Z}^+$. Moreover, the total number of layers in MG is denoted as $|\mathcal{L}(\mathsf{MG})| = l$. In addition, for a multilayer subgraph $H \subseteq \mathsf{MG}$, the subgraph of H located at i-th layer is "projecting" H to G_i, denoted as $H_i = H \downarrow_{G_i} = (V(H) \cap V_i, E(H) \cap E_i)$. Corresponding, the cross-layer subgraph of H at two layers G_i and G_j is "projecting" H to E_{ij}, denoted as $E_{ij}^H = H \downarrow_{E_{ij}} = E(H) \cap E_{ij}$.

Example 1. *Fig. 2(a) shows an example of multilayer graph* MG *with* $l = 4$ *layers. The vertex* u_1 *located at layer* G_2*, has three internal edges* (u_1, u_2)*,* (u_1, u_3)*,* (u_1, u_4)*, and three cross-layer edges* (u_1, v_1)*,* (u_1, w_1)*,* (u_1, r_2) *with layer* G_1*,* G_3*, and* G_4 *respectively.*

(k, d)-Core. We consider a typical two-layer subgraph of MG as $G_{ij} = (G_i, G_j, E_{ij})$, which is a two-layer induced subgraph of MG formed by two typed vertices V_i and V_j. In addition, a well-known concept of k-core H is a useful dense subgraph in single-layer graphs, requiring that each vertex has at least k neighbors in H, i.e., $\forall v \in V(H)$, $\deg_H(v) \geq k$ [25]. The minimum degree of k-core H is used to quantify structural cohesiveness. Extending from k-core to multilayer graphs, we give a new definition of multilayer subgraph (k, d)-core $H \subseteq \mathsf{MG}$ with $|\mathcal{L}(H)| = 2$ in the following.

Definition 1 ((k, d)-core)**.** Given a multilayer graph MG and two parameters $k, d \in \mathbb{Z}^{\geq 0}$, a connected two-layer subgraph $H(H_i, H_j, E_{ij}^H) \subseteq \mathsf{MG}$ located at layers G_i and G_j is (k, d)-core *if and only if* H admits the following conditions:

1) $\forall v \in V_i(H)$, the intra-degree at layer G_i: $\deg_{H_i}(v) \geq k$;
2) $\forall v \in V_j(H)$, the intra-degree at layer G_j: $\deg_{H_j}(v) \geq k$;
3) $\forall v \in V_i(H) \cup V_j(H)$, the inter-degree: $\deg_{E_{ij}^H}(v) \geq d$.

Example 2. *Fig. 2(b) shows a* $(2, 1)$*-core subgraph* $H_a(H_2^a, H_3^a, E_{23}^{H_a})$ *cross two layers* G_2 *and* G_3*. Each vertex* $u \in H_2^a$ *has at least* 2 *internal neighbors in* H_2^a *and* 1 *cross-layer neighbor in* H_3^a*; each vertex* $w \in H_3^a$ *has at least* 2 *internal neighbors in* H_3^a *and* 1 *cross-layer neighbor in* H_2^a*, such that* $deg_{H_2^a}(u) \geq 2$*,* $deg_{H_3^a}(w) \geq 2$*, and* $deg_{E_{23}^{H_a}}(u, w) \geq 1$.

B. Multilayer Community and Problem Formulation

Based on the two-layer (k, d)-core, we can give a definition of *strong cross-layer connectivity* (SLC) as follows.

Definition 2 (Strong Cross-layer Connectivity)**.** Given a multilayer graph H, we say that H has the *strong cross-layer connectivity* between two layers G_i and G_j *if and only if* there exists a non-empty two-layer subgraph of (k, d)-core $H' \subseteq H$ at layers G_i, G_j, denoted as $H_i \overset{H}{\longleftrightarrow} H_j$.

Here $H' = (H_i', H_j', E_{ij}^{H'})$ is a (k, d)-core cross layers G_i and G_j. Note that the (k, d)-core constraint in the SLC definition is a decision version of the problem of finding (k, d)-core, such that given two parameters k, d, to check whether there exists a (k, d)-core H' cross layers G_i and G_j in the target multilayer connected subgraph H, such that $H' \subseteq H$.

Example 3. *Consider the subgraph* H_a *in Fig. 2(b) and* H_b *in Fig. 2(c).* H_a *is a* $(2, 1)$*-core,* H_a *is a subgraph of* H_b*. We say that* H_b *has* SLC *cross layers* G_2 *and* G_3*, denoted as* $H_2^b \overset{H_b}{\longleftrightarrow} H_3^b$*, although* w_4 *is not a vertex of* $(2, 1)$*-core.*

Fully-connected multilayer community. Based on two layers' SLC, we can define a novel multilayer community model H, which has much stronger cross-layer connectivity among multiple layers. We require that every pair of layers has SLC in H, denoted as the full-layer connectivity (FLC). The detailed definition of our fully-connected multilayer community is described as follows.

Definition 3 (Fully-connected Multilayer Community)**.** Given a multilayer subgraph $H \subseteq \mathsf{MG}$ and two numbers k, d, we say that H is a full-layer connected multilayer community *if and only if* for every pair of layers $i, j \in \mathcal{L}(H)$, there exists a strong cross-layer connectivity between G_i and G_j, such that, $\forall i, j \in \mathcal{L}(H)$, $H_i \overset{H}{\longleftrightarrow} H_j$.

Example 4. *Given a multilayer graph* MG*,* $k = 2$*, and* $d = 1$*,* H_c *in Fig. 2(d) is a fully-connected multilayer community. For* $\mathcal{L}(H_c) = \{2, 3, 4\}$*, there exists* SLC *on* H_c*, such that* $H_2^c \overset{H_c}{\longleftrightarrow} H_3^c$*,* $H_2^c \overset{H_c}{\longleftrightarrow} H_4^c$*,* $H_3^c \overset{H_c}{\longleftrightarrow} H_4^c$*.*

Built upon the proposed model of fully-connected multilayer community by Def. 3, we can formulate the problem of cross-layer community search over multilayer graphs MCS-problem as follows.

Problem 1 (MCS-problem)**.** *Given a multilayer graph* $\mathsf{MG}(V_M, E_M, \mathcal{L})$*, a set of query vertices* $Q \subseteq V_M$*, two parameters* $k, d \in \mathbb{Z}^{\geq 0}$*, the problem of cross-layer community search in* MG *(*MCS-problem*) is to find a connected community* $H \subseteq$ MG *satisfying the following four constraints:*

1) ***Query-dependent personalization:*** $Q \subseteq V(H)$*;*
2) ***Core-dense internal layers:*** $\forall i \in \mathcal{L}(H)$*,* H_i *is a connected* k*-core;*
3) ***Fully-connected cross-layers:*** $\forall i, j \in \mathcal{L}(H)$*, two layers* H_i *and* H_j *are connected via a* (k, d)*-core in* H*;*
4) ***Cross-layer maximization:*** $|\mathcal{L}(H)|$ *is maximized.*

The constraint (1) of query-dependent personalization ensures that the result H contains all query vertices. Moreover, the constraint (2) of k-core requires that all members in each layer are densely connected with at least k neighbors internally. The constraint (3) of fully-connected cross-layers ensures a cohesive cross-layer relationship for every two layers. In addition, the constraint (4) of cross-layer maximization ensures that the community H has the most layers of various typed entities and connections, indicating the most informative community w.r.t. query Q.

Example 5. *For a given multilayer graph* MG*, a query vertex* $Q = \{u_1\}$*,* $k = 2$*, and* $d = 1$*,* H_d *in Fig. 2(e) is the answer for* MCS-problem*.* H_d *satisfies all* 4 *constraints: (i)* $u_1 \in V(H_d)$*; (ii)* H_2^d*,* H_3^d*, and* H_4^d *are* 2*-cores;* r_4*,* r_5 *are not belong to* 2*-core in* G_4*; (iii)* H_d *is* FLC*; (iv)* H_d *has the largest layer number of* 3 *among all candidates.*

C. Problem Analysis and Extensions

We analyze the hardness of MCS-problem and also the extension of MCS-problem over other complex graphs.

Problem hardness. We prove that the MCS-problem is NP-hard reduced from a well-known NP-complete problem Maximum Clique (MC) problem. Formally, given an undirected graph $G(V, E)$, a positive integer k, a k-clique is a complete subgraph of G with size k. The Maximum Clique decision problem is to check whether a k-clique H exists in graph $G(V, E)$ for a given k. We first present the decision version of multilayer community search (dMCS-problem).

Problem 2 (dMCS-problem)**.** *Given a multilayer graph* MG*, query vertices* Q*, parameters* k*,* d*, and* $\alpha \in \mathbb{Z}^+$*, the problem is to check whether there exists a fully-connected multilayer community* H *containing* Q *in* MG *satisfies the constraint of an exact layer number* $|\mathcal{L}(H)| = \alpha$*.*

Theorem 1. *The* dMCS-problem *is NP-hard.*

Proof. We reduce the Maximum Clique decision problem to the dMCS-problem. Let the graph $G(V, E)$ with n vertices be an instance of the Maximum Clique decision problem. We construct a corresponding instance of dMCS-problem as follows. We reconstruct the homogeneous single-layer graph $G(V, E)$ as a multilayer graph $\mathsf{MG}(V_M, E_M, \mathcal{L})$. First, we put each vertex v_i in V to a separate layer G_i, i.e., $V_{G_i} = \{v_i\}$ for $1 \leq i \leq |V|$. In addition, we add a set of dummy vertex Q where $|Q| \geq 1$. Let each dummy vertex be located in one new layer. Thus, $V_M = V \cup Q$ and the total number of layers is $|\mathcal{L}(\mathsf{MG})| = |V| + |Q| = |V_M|$. Second, we connect each vertex $q \in Q$ with each other vertex in V_M, such that $E_M = E \cup \{(u, v) | u \in Q, v \in V_M \setminus \{u\}\}$. Based on MG, we set the query as Q and parameters $k = 0$, $d = 1$ for the dMCS-problem. Given the graph $G(V, E)$ and an integer α, we show that the instance of Maximum Clique decision problem is a YES-instance, *if and only if* the corresponding instance of dMCS-problem is a YES-instance in the following.

($\Leftarrow$): Suppose that there is a α-clique H in graph $G(V, E)$, that is, for each pair of vertices $u, v \in V(H)$, there exists an edge $(u, v) \in E(H)$ and $|V(H)| = \alpha$. The corresponding answer H' induced by $V(H) \cup Q$ in MG containing $|\mathcal{L}(H')| = \alpha + |Q|$ layers is a connected $(\alpha + |Q|)$-clique, which satisfies the rule of strong cross-layer connectivity, i.e., $(0, 1)$-core exists in each pair of layers in H'. Moreover, for $i \in \mathcal{L}(H')$, H'_i is a 0-core since each layer only contains one vertex. Thus, H' satisfies the conditions of multilayer community, which is a YES-instance of dMCS-problem.

($\Rightarrow$): Suppose that a connected H' is a YES-instance of dMCS-problem in multilayer graph MG. Let H be the subgraph induced by $V(H') \setminus Q$ of MG. We infer that $|\mathcal{L}(H')| = \alpha + |Q|$. Since H' has only one vertex in each layer, the number of vertices in H' is $|V(H')| = \alpha + |Q|$. Due to the strong cross-layer connectivity of multilayer community H' by Def. 2, H' must be a connected $(\alpha + |Q|)$-clique, such that each vertex in H' must have $\alpha + |Q| - 1$ neighbors. After deleting the vertices Q and their induced edges from H', the rest of subgraph H' is the α-clique H in graph G. Thus, H is a YES-instance of the Maximum Clique decision problem.

Extending multilayer graphs to handle other graph instances. Note that our multilayer graph is a general graph type, which can be used to equivalently model other graph instances, e.g., a pillar multilayer graph is our multilayer graph with the same typed vertices; a multiplex graph is our multilayer graph MG with the same set of vertices at all layers, i.e., $V_1 = V_2 = \ldots = V_l$; one single-layer homogeneous graphs $G_1(V_1, E_1)$ is our multilayer graph MG for $|\mathcal{L}(\mathsf{MG})| = 1$; a bipartite graph $G_{12}(V_1, V_2, E_{12})$ is our multilayer graph MG for $E_1 = E_2 = \emptyset$ and $|\mathcal{L}(\mathsf{MG})| = 2$, and so on. In addition, our full-layer connected community model can be extended to path-layer connected community model, which makes a relaxation of NP-hard problem to another problem that can be done in polynomial-time (Section V).

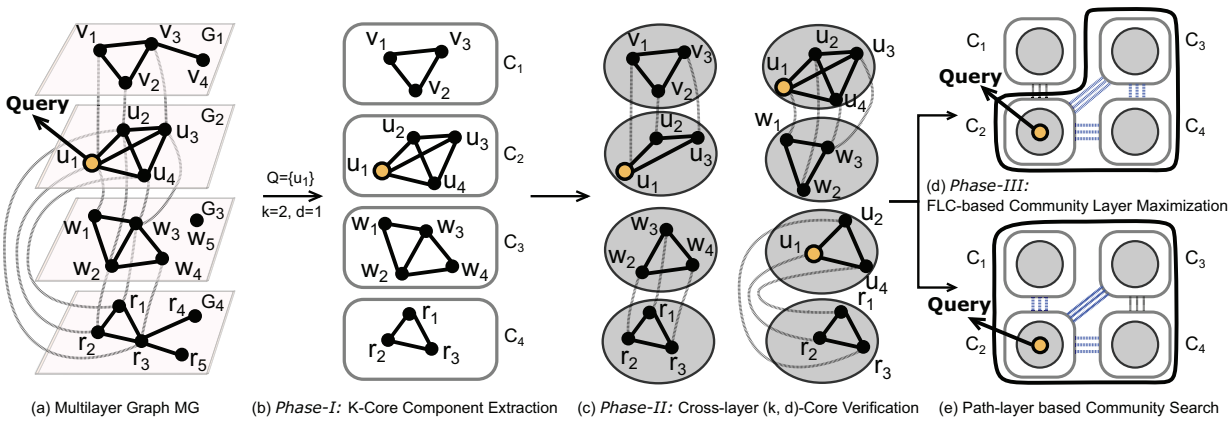


Fig. 3. Multilayer Community Search Framework.

IV. The MCS Framework

In this section, we propose a novel MCS framework to address the MCS-problem, which finds a connected multilayer community H containing the input query vertices Q meanwhile achieving the maximum layer $|\mathcal{L}(H)|$.

A. Solution Overview

In the following, we present an overview of MCS framework. The key idea of MCS consists of three phases: (1) extracting k-core components at all layers, (2) identifying the strong cross-layer connectivity for any two k-core components at different layers, and (3) starting from Q to search fully-connected multilayer community H. The detailed phases are described below.

- **Phase I: k-core component extraction**. For each layer graph G_i, we apply the core decomposition [47] on G_i and then use BFS to identify all connected k-core components.
- **Phase II: cross-layer (k,d)-core validation**. We propose a (k,d)-core decomposition algorithm to check whether there exists a (k,d)-core for each pair of k-core components at two different layers.
- **Phase III: FLC-based multilayer community refinement for layer maximization**. After determining the strong cross-layer connectivity for all possible layers, we obtain a set of valid components and prune those disqualified candidates for FLC-based communities. Then, we start from k-core components containing Q to expand and refine to an exact community answer with the maximum number of layers.

Fig. 3 shows an overview of our MCS framework. The detailed algorithm of MCS is presented in Algorithm 1. Note that we design the (k,d)-core index computed offline to help improve the efficiency (Algorithm 7), which can accelerate the above online query processing for fast MCS search.

B. Phase-I: K-Core Component Extraction for All Layers

We present the Phase-I of k-core component extraction for all layers in Algorithm 2. The algorithm has two main steps. For each layer graph G_i where $1 \le i \le |\mathcal{L}(\mathsf{MG})|$, we first apply the core decomposition algorithm [47] to remove all disqualified vertices v with $\deg_{G_i}(v) < k$. Thus, we obtain the k-core subgraph of G_i (lines 3-5). After that, the algorithm uses the breadth-first search (BFS) to split the vertices in G_i into independent components of k-core as $\{C_i^1, C_i^2, ..., C_i^t\}$, where each C_i^x represents the x-th connected k-core in layer G_i and $1 \le x \le t$ (line 6). Finally, the procedure assigns all k-core components for all layers into the set $\mathcal{C}$ and returns $\mathcal{C}$ (lines 7-8).

Algorithm 1: MCS Framework

Input: Multilayer graph $\mathsf{MG} = (V_M, E_M, \mathcal{L})$, parameters k, d, and query vertices Q
Output: Multilayer community H

1 Extract k-core connected components $\mathcal{C}$ in all layers by invoking Algorithm 2;
2 Verify the strong cross-layer connectivity between two different layered components $C_i, C_j \in \mathcal{C}$, and store the valid pair (C_i, C_j) into SLC_p by invoking Algorithm 3;
3 Start from query Q to find a valid full-layer connected community H with the largest $|\mathcal{L}(H)|$ by invoking Algorithm 4;
4 **return** H;

Algorithm 2: K-Core Component Extraction

Input: Multilayer graph MG and parameter k
Output: $\mathcal{C} = \{C_i^x$: the k-core components at layer G_i for $1 \le i \le |\mathcal{L}(\mathsf{MG})|\}$

1 $\mathcal{C} \leftarrow \emptyset$;
2 **for** *each layer* G_i *where* $1 \le i \le |\mathcal{L}(\mathsf{MG})|$ **do**
3 **while** $\exists v \in V_i$ *with* $\deg_{G_i}(v) < k$ **do**
4 Remove v and its incident edges from G_i;
5 All remaining vertices satisfy the k-core degree;
6 Find all maximal connected k-core components $\{C_i^1, \ldots, C_i^t\}$ by applying BFS at layer G_i;
7 $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_i^1, \ldots, C_i^t\}$;
8 **return** $\mathcal{C}$;

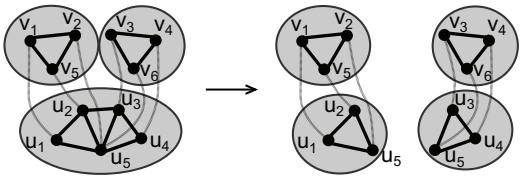


Fig. 4. An example of splitted (k, d)-core.

C. Phase-II: Cross-layer (k, d)-Core Verification

Next, we present the Phase-II of cross-layer (k, d)-core verification in Algorithm 3. The algorithm aims at finding all qualified pairs of layers satisfying the *strong cross-layer connectivity (SLC)*. Thus, for each pair of k-core components $C_i, C_j \in \mathcal{C}$, it needs to verify the existence of a (k, d)-core between them (lines 1-6). If (C_i, C_j) satisfy the SLC, we add the pair (C_i, C_j) into answer SLC_p (line 5). Note that the main procedure invoked by Algorithm 3 is *ComputeSLC*, which computes valid (k, d)-cores (lines 7-21). It starts from an induced cross-layer subgraph C_{ij} formed by C_i, C_j in MG (line 9). The algorithm continues to remove vertices with internal layer degree less than k, or vertices with cross-layer degree less than d, along with their incident edges, which is to find a (k, d)-core C_{ij} (lines 10-11). However, the remaining C_{ij} above can be disconnected, which violates the SLC constraint. We further check the connectivity of those components as follows (lines 12-20). First, if both C_i and C_j are unique k-core components, they are valid and added to the valid set KD (lines 13-16). Otherwise, the algorithm iterates over each connected component in C_i and C_j, recursively calling *ComputeSLC* to add valid results to KD (lines 17-20). If the *ComputeSLC* function returns a non-empty set, it indicates a valid SLC pair, which is finally added to our answer SLC_p.

Fig. 4 gives an example of splitting a $(2, 1)$-core subgraph to two $(2, 1)$-core subgraphs when extracting a connected (k, d)-core. The connectivity constraint confirms the high cohesiveness of our (k, d)-core model.

D. Phase-III: FLC-based Community Layer Maximization

Based on the obtained components $\mathcal{C}$ and the layer connectivity pairs SLC_p from Phases-I&II, we present the Phase-III of FLC-based community layer maximization in Algorithm 4. The key to MCS is to find a multilayer community with the maximum number of layers. Thus, the general idea of Algorithm 4 is to treat each k-core component of $\mathcal{C}$ as *single vertex* and the connectivity pairs SLC_p as the set of *edges*, which finally discovers the maximum "clique" as the community answer H. The algorithm initializes an empty set H (line 1) and updates $\mathcal{C}$ by removing the disqualified k-core components by SLC_p (line 2). Next, it extracts the set of components containing all query vertices Q as $\mathcal{C}_Q$ (line 3). If one pair of components $C_i, C_j \in \mathcal{C}_Q$ does not satisfy the SLC condition, i.e., $(C_i, C_j) \notin \mathsf{SLC}_p$, the algorithm can early terminate and return empty set which indicates no valid community H (lines 4-6). Otherwise, we explore and find a valid community H (lines 7-10). Next, we initialize H as $\mathcal{C}_Q$ and update the remaining components $\mathcal{C} \setminus \mathcal{C}_Q$, which represents the candidate components that have not yet been included in the multilayer community (line 8). Then, it calls the procedure of FindMaxFLC to recursively explore different combinations of components to find the maximum multilayer community (lines 11-20). It starts with an unvisited component C_i and adds it to multilayer community H by checking the FLC connectivity (lines 15-19). The procedure stops when all components are visited in the exploration process. Finally, the algorithm returns a multilayer community H with the maximum layers $|\mathcal{L}(H)|$.

Algorithm 3: Cross-layer (k, d)-Core Verification

Input: MG, integers k, d, and k-core components $\mathcal{C}$
Output: A set of valid component pairs satisfying SLC: $\mathsf{SLC}_p = \{(C_i^x, C_j^y) : C_i^x, C_j^y \in \mathcal{C}\}$

1 $\mathsf{SLC}_p \leftarrow \emptyset$;
2 **for** $\forall C_i^x \in \mathcal{C}$ *and* $\forall C_j^y \in \mathcal{C}$ *with* $i \neq j$ **do**
3 // Check SLC between components C_i^x, C_j^y by calling the procedure ComputeSLC(.);
4 **if** *ComputeSLC*$(C_i^x, C_j^y) \neq \emptyset$ **then**
5 $\mathsf{SLC}_p \leftarrow \mathsf{SLC}_p \cup \{(C_i^x, C_j^y)\}$;
6 **return** SLC_p;
7 **procedure** ComputeSLC(C_i, C_j)
8 $KD \leftarrow \emptyset$;
9 Extract a cross-layer induced subgraph of MG formed by two sets of vertices C_i and C_j, denoted as C_{ij};
10 **while** $\exists v \in C_i \cup C_j$ *having internal layer degree less than* k *or cross-layer degree less than* d *in* C_{ij} **do**
11 Remove v and its incident edges from C_{ij};
12 **if** $C_i \neq \emptyset$ *and* $C_j \neq \emptyset$ **then**
13 Identify all connected components in the remaining graph of two layers C_i and C_j;
14 **if** *both* C_i *and* C_j *have one unique component* **then**
15 // Indicating the holding SLC: $C_i \overset{C_{ij}}{\longleftrightarrow} C_j$;
16 $KD \leftarrow KD \cup \{(C_i, C_j)\}$;
17 **else**
18 **foreach** *component* $C_i' \subseteq C_i$ **do**
19 **foreach** *component* $C_j' \subseteq C_j$ **do**
20 $KD \leftarrow KD \cup$ComputeSLC(C_i', C_j');
21 **return** KD;

Fig. 3 gives an example of the whole three phases. Given a multilayer graph MG, query $Q = \{u_1\}$, parameters $k = 2$, $d = 1$, the MCS framework first extracts all k-core components $\mathcal{C}$ in MG in Fig. 3(b), then verifies the (k, d)-core existence between each pair of k-core components in Fig. 3(c), and finally computes the full-layer connected community with three layers in Fig. 3(d).

Algorithm 4: FLC-based Community Layer Maximization

Input: MG, integers k, d, query vertices Q, components $\mathcal{C}$, and all valid SLC pairs SLC_p
Output: FLC-based multilayer community H

```
1  Initialize the answer H ← ∅;
2  Extract the candidate components C = {C_i : (C_i,C_j)∈SLC_p} by Algorithm 3;
3  Extract a set of components C_Q ⊆ C containing all query vertices Q, i.e., Q ⊆ ⋃_{C∈C_Q} C;
4  if ∃C_i, C_j ∈ C_Q such that (C_i, C_j) ∉ SLC_p then
5      //No valid multilayer community containing Q;
6      return ∅;
7  else
8      Update the community H ← C_Q; Update the candidate components: C ← C \ C_Q;
9      H ←FindMaxFLC(H, C);
10     return H;
11 procedure FindMaxFLC(H, C_unvisited)
12     Initialize MaxH ← H;
13     foreach component C_i ∈ C_unvisited do
14         // Starting from H to expand community layers by picking a component C_i;
15         Check the layer connectivity FLC between C_i and existing components' layers in H;
16         if ∀C_j ∈ H with (C_i, C_j) ∈ SLC_p then
17             H' ← FindMaxFLC(H ∪ C_i, C_unvisited \ C_i);
18             if |L(H')| > |L(MaxH)| then
19                 MaxH ← H';
20     return MaxH;
```

Complexity analysis. We denote the number of vertices, edges, layers and components in MG as $n = |V_M|$, $m = |E_M|$, $l = |\mathcal{L}(\mathsf{MG})|$, and $c = |\mathcal{C}|$ respectively. For $1 \le i < j \le l$, we denote vertex size and edge size of G_i, and cross-layer edge size between G_i and G_j as $n_i = |V(G_i)|$, $m_i = |E(G_i)|$, and $m_{ij} = |E(G_{ij})|$ respectively. The total number of internal edges at all layers is $m_l = \sum_{i=1}^{l} m_i$, and the number of cross-layer edges is $m_s = \sum_{1\le i<j\le l} m_{ij}$. Thus, $m = m_l + m_s$. Algorithm 2 involves iterating through each layer and removing vertices and their incident edges with internal layer degrees less than k, which takes $O(\sum_{i=1}^{l} m_i) = O(m_l)$ time. The BFS-based splitting procedure takes $O(\sum_{i=1}^{l}(m_i + n_i)) = O(m_l)$ time. The overall time complexity of Algorithm 2 is $O(m_l)$. Algorithm 3 within the ComputeSLC procedure takes $O(m_{c_i} + m_{c_j} + m_{c_{ij}})$ time for extracting the cross-layer induced subgraph. The overall time complexity can be expressed as $O(\sum_{1\le i\le c}\sum_{1\le j\le c}(m_{c_i} + m_{c_j} + m_{c_{ij}})) = O(\sum_{1\le i\le c}(m_l + m_l + m_s)) = O(cm)$. The time complexity of Algorithm 4 depends on the size of the input SLC_p and the number of valid k-core components $\mathcal{C}$. It involves recursive calls and iterations over sets of components. The time complexity for checking the connectivity in line 15 is $O(l^2)$. The total time complexity of Algorithm 4 is $O(l^2 \cdot \binom{c}{l}) = O(l^2 2^c)$. In summary, the total time complexity of MCS framework is $O(m_l + cm + l^2 2^c) = O(cm + l^2 2^c)$, and the space complexity is $O(m)$.

V. Relaxed Problem and Fast Algorithms

Due to the NP-hardness of MCS-problem as shown in Section III, it leads to highly expensive computation of Algorithm 1 to find exact answers. To improve the efficiency, we propose a fast MCS algorithm in this section. First, we relax the strict constraint of *full-layer connectivity* to a new version of *path-layer connectivity*. Based on the path-layer connectivity, we reformulate our MCS-problem to a relaxation problem of path-layer MCS, which can be optimally addressed in polynomial time. Leveraging the answer of path-layer based multilayer community as an upper bound, we develop a new MCS algorithm to further accelerate the search process.

A. The Relaxed pMCS-problem and Analysis

We begin with a new definition of path-layer connectivity (PLC) as follows.

Definition 4 (Path-layer Connectivity)**.** For a given multilayer subgraph $H \subseteq \mathsf{MG}$, two layers H_i and H_j with $i, j \in \mathcal{L}(H)$, $i \neq j$, we say that H_i and H_j has the *path-layer connectivity*, denoted as $H_i \overset{H}{\rightsquigarrow} H_j$, *if and only if* there exists a path $(H_{p_1}, \ldots, H_{p_r})$ such that every pair of layers $(H_{p_x}, H_{p_{x+1}})$ where $1 \le x < r$, is *strong cross-layer connected*, i.e., $H_{p_x} \overset{H}{\longleftrightarrow} H_{p_{x+1}}$, $p_1 = i$, and $p_r = j$.

Based on the path-layer connectivity, we make a relaxation of MCS-problem as the new pMCS-problem.

Problem 3 (pMCS-problem)**.** *Given a multilayer graph* MG*, a set of query vertices Q, two parameters k and d, the problem of path-layer based multilayer community search is to find a connected subgraph* $H \subseteq \mathsf{MG}$ *satisfying four constraints:*

1) *H contains all query vertices Q;*
2) *$\forall i \in \mathcal{L}(H)$, H_i is a k-core;*
3) *$\forall i, j \in \mathcal{L}(H)$, the path-layer connectivity $H_i \overset{H}{\rightsquigarrow} H_j$ always holds;*
4) *$|\mathcal{L}(H)|$ is maximized.*

As we can see, pMCS-problem has the same three constraints of MCS-problem, in terms of the query-dependent personalization, core-dense internal layers, and the optimized objective of cross-layer maximization. The only one difference lies on the constraint of *path-layer connected cross-layers* in replace of *full-connected cross-layers*. Fig. 3(e) gives an example of search result by applying the pMCS-problem model.

Comparing MCS-problem and pMCS-problem, if we regard a connected k-core component at each layer as a vertex, then MCS-problem is corresponding to the maximum clique search for the largest layers, while pMCS-problem can be reformulated as finding out the largest component containing

Algorithm 5: Path-layer based Community Search

Input: MG, integers k, d, and query vertices Q
Output: Path-layer based multilayer community H_{path}

1 Extract k-core components $\mathcal{C}_Q$ containing the query Q;
2 Initialize community: $H_{path} \leftarrow C_i$ for any one $C_i \in \mathcal{C}_Q$;
3 Enlarge community H_{path} through cross-layer neighbors in BFS manner by expanding components that satisfy SLC;
4 **if** $C_Q \not\subseteq H_{path}$ **then**
5 **return** $\emptyset$;
6 **else**
7 **return** H_{path};

Q. However, the efficient search of relaxed pMCS-problem is also challenging due to the complexity of multilayer graph structure. Specifically, each layer may have several eligible connected k-core components, which have many cross-layer connections for one k-core component.

Path-based community search algorithm. We present Algorithm 5 to find a path-based multilayer community with the maximum number of layers. It first identifies the k-core components of $\mathcal{C}$ containing query Q as $\mathcal{C}_Q$ (line 1). Then, it initializes H_{path} as one component of $\mathcal{C}_Q$ (line 2). It then starts the BFS search through the cross-layer neighbors of vertices in $\mathcal{C}_Q$, expanding more components and enlarging the number of layers by verifying whether they satisfy the SLC constraint (line 3). If the final community H_{path} involves the complete query $\mathcal{C}_Q$, the algorithm terminates with a feasible answer H_{path}; otherwise, there is no answer of connected community containing Q (lines 4-7).

B. Fast FLC*-based Community Layer Maximization*

In this section, we propose a fast algorithm for maximizing FLC-based community layer under the framework in Algorithm 1, which leverages the path-layer community answer in Algorithm 5 for reducing search space.

FastMCS algorithm. We propose Algorithm 6 to fast search community using a binary search strategy, which takes the upper bound of path-layer community by Algorithm 5 and the lower bound of query-based components $\mathcal{C}_Q$ (lines 2-5). Indeed, Algorithm 6 accelerates Algorithm 4 by replacing the lines 8-9 of Algorithm 4. It begins by running Algorithm 5 to compute the upper bound l_{max}, which is the maximum number of layers in community H_{path} (lines 2-3). Next, it finds $\mathcal{C}_Q$ to contain all the query vertices Q and sets the lower bound l_{min} as the number of layers in $\mathcal{C}_Q$. Moreover, H is initially assigned to C_Q (lines 4-5). Next, the algorithm binary searches a possible answer by enumerating all candidate communities S_h with h-layers $|\mathcal{L}(H)| = h$, where the middle value $h = (l_{max} + l_{min})/2$ (lines 7-8). It then checks the FLC connectivity of S_h (line 9). If such a FLC holds, l_{min} is updated to h for finding a community with more layers, and H is updated to S_h; otherwise, l_{max} is updated to h as the

Algorithm 6: Fast Community Layer Maximization

Input: MG, integers k, d, query vertices Q, components $\mathcal{C}$, and all valid SLC pairs SLC_p
Output: FLC-based multilayer community H

1 //Replace the following steps with lines 8-9 in Algorithm 4;
2 Apply Algorithm 5 to find path-layer based community H_{path} containing Q; Filter $\mathcal{C}$ by H_{path};
3 Set the upper bound of layers: $l_{max} = |\mathcal{L}(H_{path})|$;
4 Initialize the community $H \leftarrow \mathcal{C}_Q$;
5 Set the lower bound of layers: $l_{min} = |\mathcal{L}(H)|$;
6 **while** $l_{min} < l_{max}$ **do**
7 $h \leftarrow \lfloor \frac{l_{min}+l_{max}}{2} \rfloor$;
8 Enumerate the set of h components $S_h \subseteq \mathcal{C}$ at different layers as $|\mathcal{L}(S_h)| = h$;
9 Check the FLC connectivity of S_h;
10 **if** $\exists C_i, C_j \in S_h$ *with* $(C_i, C_j) \notin \mathsf{SLC}_p$ **then**
11 $l_{max} \leftarrow h$;
12 **else**
13 $l_{min} \leftarrow h$, $H \leftarrow S_h$;
14 **return** H;

FLC does not hold (lines 10-13). Finally, it returns a multilayer community H when $l_{min} \geq l_{max}$ (lines 6-14).

Complexity analysis. Algorithm 5 invokes Algorithms 2 and 3 to verify the SLC constraint, which has a time complexity of $O(cm)$. The community H_{path} is initialized and extended by expanding components using the SLC in a BFS manner, which has a time complexity of $O(cm + m) = O(cm)$ in total. The total space complexity of Algorithm 5 is $O(m)$. Algorithm 6 first computes the upper bound of layer number by applying Algorithm 5. It then does the binary search which takes $O(\log \hat{l})$ time in the worst case, where $\hat{l}$ denotes the upper bound of layer number. During the while loop, it enumerates the possible FLC combinations and checks the SLC of components in S_h, which takes $O(\log \hat{l} \cdot \hat{l}^2 \cdot \binom{\hat{c}}{\hat{l}}) = O(\hat{l}^2 2^{\hat{c}} \log \hat{l})$ time in total, where $\hat{c}$ is the number of filtered k-core components. In summary, the complexity of the fast MCS algorithm takes $O(cm + \hat{l}^2 2^{\hat{c}} \log \hat{l})$ time in $O(m)$ space, and we have $\hat{l} \leq l$ and $\hat{c} << c$ in practice.

VI. (k, d)-CORE INDEXING

In this section, we design a novel (k, d)-core index to accelerate MCS process. Specifically, we first present an algorithm for (k, d)-core decomposition and indexing. This algorithm precomputes all possible (k, d) pairs for cross-layer edges over every pair of layers in MG and stores them into an index of compact data structure. Then, we illustrate how we apply the (k, d)-core index to help improve efficiency of our proposed MCS algorithms in previous sections.

(k, d)-core index. In a single-layer graph G, the *coreness* of a vertex v is represented as $\delta(v)$, referring to the maximum value of k such that a non-empty k-core contains v [15], [25]. In our MCS-problem, for two layers i and j, each cross-layer edge

Algorithm 7: (k,d)-Core Index Construction

Input: Multilayer graph $\mathsf{MG} = (V_M, E_M, \mathcal{L})$
Output: the (k,d)-core index $\Phi(\mathsf{MG}) = \{\Phi(\langle u,v\rangle) : \mathcal{L}(u) \neq \mathcal{L}(v), u, v \in V_M\}$

1 $\Phi(\mathsf{MG}) \leftarrow \emptyset$;
2 **for** $k \leftarrow 1$ *to* k_{max} **do**
3 Compute the connected k-core components $\mathcal{C}$ by invoking Algorithm 2;
4 **for** $\forall C_i^x \in \mathcal{C}$ *and* $\forall C_j^y \in \mathcal{C}$ *with* $i \neq j$ **do**
5 $d_{max} \leftarrow 0$;
6 **if** $k = 1$ **then**
7 **while** *ComputeSLC*$(C_i^x, C_j^y, k, d_{max}) \neq \emptyset$ **do**
8 $d_{max} \leftarrow d_{max} + 1$;
9 **else**
10 $d_{max} \leftarrow \Phi^{(k-1)}(\langle u,v\rangle)$;
11 **while** *ComputeSLC*$(C_i^x, C_j^y, k, d_{max}) = \emptyset$ **do**
12 $d_{max} \leftarrow d_{max} - 1$;
13 **for** $\forall (u,v), u \in C_i^x, v \in C_j^y$ **do**
14 $\Phi(\langle u,v\rangle) \leftarrow \Phi(\langle u,v\rangle) \cup \{(k, d_{max})\}$;
15 **return** $\{\Phi(\langle u,v\rangle) : \mathcal{L}(u) \neq \mathcal{L}(v), u, v \in V_M\}$;

$(u,v) \in E_{ij}$ can belong to a set of various (k,d)-cores. We develop a notation (k,d)-*coreness* Φ to store all the possible *non-dominant* (k,d) pairs for cross-layer edges $e = (u,v)$ to determine whether there exists a (k,d)-core H such that $(u,v) \in E_{ij}^H$. Before introducing the (k,d)-coreness Φ, we first give a definition of $\Phi^k(\langle u,v\rangle)$ to denote the maximum coreness of value d for a given value k such that a (k,d)-core contains cross-layer edge (u,v). Obviously, there is no (k,d')-core H' containing (u,v) with $d' > d = \Phi^k(\langle u,v\rangle)$. Based on $\Phi^k(\langle u,v\rangle)$, we give a definition of (k,d)-coreness.

Definition 5 ((k,d)-coreness)**.** Given two layers G_i and G_j, the (k,d)-coreness of a cross-layer edge (u,v) in E_{ij}, denoted as $\Phi(\langle u,v\rangle)$, is a set of unique (k,d) pairs, such that for any pair of $(k,d) \in \Phi(\langle u,v\rangle)$, $\Phi^k(\langle u,v\rangle) = d$. Moreover, for any $(k,d) \in \Phi(\langle u,v\rangle)$, there exists no $(k',d') \in \Phi(\langle u,v\rangle)$ such that both $k' \geq k$ and $d' \geq d$ hold.

As a result, the (k,d)-core index of multilayer graph MG is $\Phi(\mathsf{MG}) = \{\Phi(\langle u,v\rangle) : \mathcal{L}(u) \neq \mathcal{L}(v), (u,v) \in E_M\}$ to store all cross-layer edges' coreness by the Def. 5.

Example 6. *Fig. 2(b) shows a multilayer $(2,1)$-core subgraph H_a cross two layers G_2 and G_3. Thus, $\{(2,1)\} \subseteq \Phi(\langle u_1, w_1\rangle)$. There does not exist a (k,d')-core containing (u_1,w_1), where $d' > 1$, or a (k',d)-core containing (u_1,w_1) where $k' > 2$.*

(k,d)-core index construction. We build the (k,d)-core index by computing the (k,d) coreness Φ for all cross-layer edges and extending the (k,d) value pairs to its corresponding cross-layer connected k-core components. Formally, given two layers i and j, the (k,d)-core index of edge (u,v) in $H_{ij}(H_i, H_j, E_{ij})$, contains a value pair (k,d) if and only if there exists (u',v') in H_{ij}, where $(k,d) \in \Phi(\langle u',v'\rangle)$, both H_i and H_j are connected k-cores. The computations of (k,d)-coreness and (k,d)-core index take the same procedure using the same time cost. Our (k,d)-core index keeps a compact index structure for saving space costs.

We present the procedure of (k,d)-core index construction in Algorithm 7. The algorithm starts to iterate all possible values of k from 1 to k_{max}. Then, it checks the components C_i^x and C_j^y in each pair of layers G_i and G_j (lines 3-14). It applies ComputeSLC algorithm to determine whether (k,d)-cores exist in a pair of connected k-core components C_i^x and C_j^y (lines 4-12). After computing all non-dominant pairs of (k,d) values, the corresponding (k,d)-corenesses are kept into $\Phi(\langle u,v\rangle)$ for each valid cross-layer edge (u,v) (lines 13-14). Finally, it stores the (k,d)-corenesses of each cross-layer edge (u,v) in the (k,d)-core index $\Phi(\mathsf{MG})$ (line 15).

Complexity analysis of (k,d)-core index construction. We represent the maximum coreness as $\hat{k}$. Overall, the index size of $\Phi(\mathsf{MG})$ is bounded by $O(\hat{k} \cdot m_s + m_l) = O(\hat{k}m)$. Next, we analyze the time complexity of Algorithm 7. For every two components in layers G_i and G_j, it invokes the core decomposition algorithm of ComputeSLC to check two components' SLC, which takes $O(m_{c_i} + m_{c_j} + m_{c_{ij}})$ time. Thus, the overall time complexity of Algorithm 7 is $O(\hat{k} \cdot \sum_{1\leq i\leq c}\sum_{1\leq j\leq c}(m_{c_i} + m_{c_j} + m_{c_{ij}})) = O(\hat{k} \cdot \sum_{1\leq i\leq c}(m_l + m_l + m_s)) = O(\hat{k}cm)$. The total space complexity of index construction is $O(\hat{k}m)$.

(k,d)-core index-based MCS algorithms. The proposed (k,d)-core index can be easily used to help accelerate two main procedures for both full-layer-based MCS algorithms and pMCS-problem algorithms. First, the connected k-core component extraction procedure in Algorithm 2 computes the k-cores by scanning the neighborhood of vertex, which can be replaced by checking the largest k value in the (k,d)-core index. Second, we can address all SLC pairs and PLC pairs computed in Algorithm 3 and Algorithm 5 by checking whether there exists a cross-layer edge with a valid (k,d)-core index between a pair of k-core components. It speeds up the time-consuming procedure for cross-layer (k,d)-core verification.

VII. Experiments

In this section, we conduct experiments to evaluate the performance of our proposed MCS model and algorithms on real-world multilayer graph datasets.

A. Experimental Setup

Setup. We implemented all algorithms in C++. All experiments were performed on a server with an Intel Xeon Gold 6330 2.0 GHz CPU and 1T RAM, running 64-bit Oracle Linux 8.8. Each test is obtained by averaging over the experimental results of 100 query samples.

Datasets. We evaluate the proposed algorithms on nine datasets of real multilayer graphs in Table II, including the

TABLE II
NETWORK STATISTICS

Dataset	n	m	m_l	m_s	l
DBLP	41,892	661,883	280,707	381,176	2
Twitter	47,280	535,062	445,287	89,775	3
6ng	4,500	29,984	9,000	20,984	5
9ng	6,750	44,980	13,500	31,480	5
Citeseer (CS)	15,533	68,376	56,548	11,828	3
Yeast	4,458	8,500,745	8,473,997	26,748	4
FAO	214	14,456,470	318,346	14,138,124	364
FriendFeed (FF)	510,338	20,204,534	18,673,520	1,531,014	3
Venetie	206	21,310	19,955	1,355	43

collaboration networks (DBLP, Citeseer [10]), social networks (Twitter, Friendfeed, Venetie [48]), news groups (6ng, 9ng), protein-protein networks (Yeast [14]), and agriculture data (FAO [14]). The last four datasets are a variety of MGs called multiplex graphs, which have the same vertices in all layers and only have internal edges; we add the additional cross-layer edges to make vertices one-to-one mapping between each layer. The statistics of these graphs are presented in Table II, where $n = |V|$ denotes the total number of distinct vertices in the whole MG, $m = |E|$ denotes the total number of edges, $m_l = |E(L)|$ represents the number of internal layer edges, $m_s = |E(C)|$ denotes the number of cross-layer edges, and $l = |\mathcal{L}(\mathsf{MG})|$ represents the layer number.

Compared algorithms. We compare seven algorithms including our five algorithms against two state-of-the-art competitors RWM [13] and FirmTruss [14] as follows.

- RWM: is a random-walk-based approach for local multilayer community search [13].
- FirmTruss: is truss-based community search approach in multiplex networks [14].
- Naive-MCS: is our baseline method in Algorithm 1 using Algorithm 4 for full-layered community enumerations.
- Path-MCS: is the path-layer community search algorithm using Algorithm 5.
- MCS: is our fast approach for full-layer community search in Algorithm 1 equipped Algorithm 6.
- Path-iMCS: is our index-based Path-MCS approach using (k, d)-core index for accelerating efficiency.
- iMCS: is our index-based improved approach of MCS.

Evaluation metrics. We first evaluate the efficiency of all algorithms on various datasets by comparing the running time. Note that FirmTruss runs only on the last 4 multiplex networks due to the nature design of algorithms. Next, we evaluate the quality of community by comparing the number of layers in community answers, i.e., $|\mathcal{L}(H)|$. A large number of layers $|\mathcal{L}(H)|$ indicates that the search result contains more types of entities, reflecting a more informative community. To further evaluate the approaches, we apply our algorithm to synthetic datasets by generating a set of fixed-sized graphs and varying the number of layers $|\mathcal{L}(\mathsf{MG})|$. We randomly select 100 queries for each task and randomly set a possible (k, d) pair for each query. We evaluate the parameter sensitivity by varying different (k, d) pairs. Furthermore, we compare the index size and index construction time to analyze the compactness and efficiency of (k, d)-*core index* vs FirmTruss *index*.

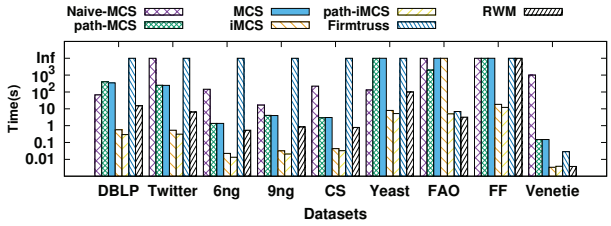


Fig. 5. Efficiency evaluation for all algorithms on all datasets.

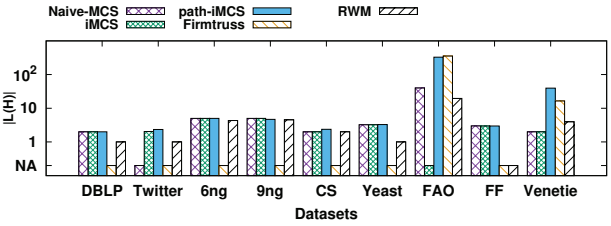


Fig. 6. Quality evaluation for all algorithms on all datasets.

B. Performance Evaluation

Exp-1: Running time of algorithms on all datasets. Fig. 5 shows the running time of two baseline algorithms RWM [13], FirmTruss [14] and our five algorithms on nine real-world datasets. Note that the FirmTruss can only execute on the last four datasets due to the restriction of its model. Note that if a query is not completed within 1200 seconds, we will report "Inf". As shown in Fig. 5, our (k, d)-core index-based algorithms significantly improve the efficiency reflected in both small and large datasets. Our Path-iMCS algorithm outperforms all other algorithms for seven datasets. Path-iMCS is slightly slower than RWM and our iMCS method for FAO and Venetie. Our iMCS approach performs similarly to our Path-iMCS approach except for the densest dataset FAO. Our (k, d)-core index-based algorithms have demonstrated a superior performance on the largest dataset, FriendFeed, with an impressive response time of around 10 seconds. In contrast, RWM and FirmTruss have failed to generate comparable results. This outcome underscores the efficiency of our algorithmic approach in multilayer community search.

Exp-2: Quality evaluation. Fig. 6 evaluates the layer number of resulting community for all algorithms. A larger layer number represents various entity types in the densely connected community. Note that our (k, d)-core index-based algorithm can find the same result as our no-index methods. Our Path-iMCS method performs the best on most datasets. FirmTruss identifies communities with the largest number of layers on FAO while it always costs more time among all competitors in Fig. 5. Even though RWM has a competitive running time for some datasets in Fig. 5, it always fails to find an informative multilayer community cross various layer entities in practice. Fig. 6 demonstrates a high-quality effectiveness of our Path-iMCS approach on most datasets.

Exp-3: Parameter sensitivity evaluation by varying (k, d) pairs. Fig. 7 and Fig. 8 show the running time of our two (k, d)-core index-based algorithms iMCS and Path-iMCS by

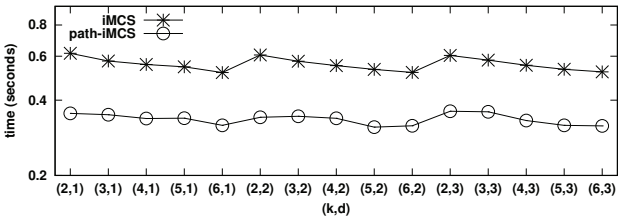


Fig. 7. Efficiency evaluation by varying parameters (k, d) on Twitter.

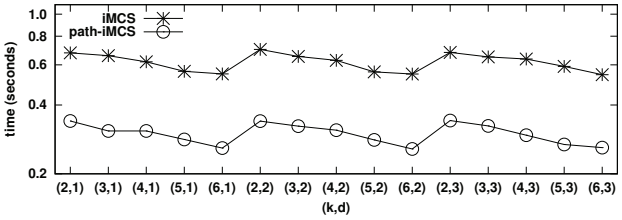


Fig. 8. Efficiency evaluation by varying parameters (k, d) on DBLP.

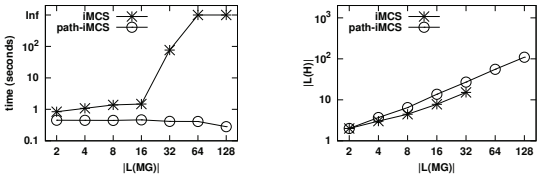
varying (k, d) pairs on Twitter and DBLP datasets, respectively. We vary the setting of query parameters (k, d) in $\{2, 3, 4, 5, 6\} \times \{1, 2, 3\}$. The results show that our Path-iMCS method always achieves a smaller running time. With the increase of k, the running time of both methods decreases, as it may decrease the number of connected k-cores when the community becomes denser. Moreover, with the increase of d, the running time remains stable; this is caused by using our (k, d)-core index, which can help check out (k, d)-core components in linear time. As a result, our methods have stable efficiency performance over different settings of (k, d) pairs on different datasets.

Exp-4: Scalability evaluation by varying the number of layers $|\mathcal{L}(\mathsf{MG})|$. We generate synthetic datasets to test the scalability of our proposed algorithms. Specifically, we generate a multilayer graph consisting of 100k of vertices and 497k edges, where the nodes follow power-law degree distribution. The probability of adding a triangle after adding a random edge is 0.99. We fix the graph size and then randomly assign vertices into different layers with a list of increase numbers $\{2, 4, 8, 16, 32, 64, 128\}$. As shown in Fig. 9(a), the running time of iMCS approach increases significantly and fails on MGs with $|\mathcal{L}(\mathsf{MG})| \geq 64$, which is because of the NP-hardness of our problem. In contrast, with the layer number increase, the computation time of Path-iMCS approach stays stable with the help of pre-computed (k, d)-core index, which decreases the computation cost, showing the superiority of the relaxation property of our path-based model. Fig. 9(b) represents the increased number of community layers found by our algorithms with the graph layer number increased. By varying the layer number $|\mathcal{L}(\mathsf{MG})|$, our Path-iMCS approach demonstrates high scalability on both efficiency and effectiveness, which shows the goodness of our well-designed relaxed path-connected multilayer model.

Exp-5: (k, d)-core index size and construction time. We evaluate the (k, d)-core index size and construction time. We compare our method to the state-of-the-art FirmTruss index in

TABLE III
A COMPARISON OF INDEX SIZE AND CONSTRUCTION TIME.

	DBLP	Twitter	6ng	9ng	CS	Yeast	FAO	FF	Venetie
Graph Size (MB)	8.99	9.43	0.377	0.59	1.71	117	211	357	0.286
(k, d)-core Index Size (MB)	4.15	7.32	0.25	0.376	1.1	336	523	307	0.417
FirmTruss Index Size (MB)	NA	NA	NA	NA	NA	1380	2600	2670	2.8
(k, d)-core-indexing Time (h)	3.63	3.44	0.005	0.009	0.01	29.02	4.27	98.39	0.001
FirmTruss-indexing Time (h)	NA	NA	NA	NA	NA	15.54	0.71	11.55	0.008



(a) Running Time (b) Layer Count $|\mathcal{L}(H)|$

Fig. 9. Running time and layer number evaluation by varying $|\mathcal{L}(\mathsf{MG})|$ on synthetic datasets.

Table III. The indexes are built offline and stored in memory. We can see that the (k, d)-core index is at most three times the original graph size $|G|$, and has a similar size to original graph, which is compact and very competitive. The FirmTruss index, available for the last four datasets, is at most 13 times the original graph. It takes much more space than ours. On the other hand, the construction time of (k, d)-core index takes longer than the construction time of FirmTruss index; this is because our index computes a large size of (k, d)-core index while we store it in a compact structure to save space. Therefore, as shown in Fig. 5, Fig. 7, Fig. 8, our (k, d)-core index-based approach provides high-quality community results and runs significantly faster than Naive-MCS without using an index, even when compared to FirmTruss, especially on large graphs and general multilayer graphs.

VIII. CONCLUSION

This paper proposes a (k, d)-core based community search problem on multilayer graphs, which finds a multilayer community H connected by (k, d)-cores containing query vertices to achieve the largest number of cross-layers. We consider two-fold definitions of full-layer and path-layer connectivities for cross-layer relationships. We show this MCS-problem under full-layer connectivity is NP-hard and propose two methods of exact exploration and heuristic search to find answers. We develop a fast search algorithm to identify path-layer-based communities and refine them to full-layer answers. Furthermore, we develop a novel (k, d)-core index of multilayer core that effectively captures essential (k, d)-core structure, which significantly speeds up the MCS algorithm up to 1000 times. Extensive experiments on nine real-world multilayer graphs demonstrate the robustness and scalability of our proposed algorithms by comparing them to two state-of-the-art algorithms.

ACKNOWLEDGMENT

This work is supported by Hong Kong RGC Grant Nos. 22200320, 12200021, 12201923, C2004-21GF, and 12201520. Xin Huang is the corresponding author.

REFERENCES

[1] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks," *Physics Reports*, vol. 544, no. 1, pp. 1–122, 2014.

[2] E. Davoodijam, N. Ghadiri, M. L. Shahreza, and F. Rinaldi, "Multigbs: A multi-layer graph approach to biomedical summarization," *JBI*, vol. 116, p. 103706, 2021.

[3] B. Oselio, A. Kulesza, and A. O. Hero, "Multi-layer graph analysis for dynamic social networks," *J-STSP*, vol. 8, no. 4, pp. 514–523, 2014.

[4] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *TKDE*, vol. 26, no. 1, pp. 97–107, 2013.

[5] E. Estrada, *The structure of complex networks: theory and applications*. Oxford University Press, USA, 2012.

[6] J. Kim and J.-G. Lee, "Community detection in multi-layer graphs: A survey," *SIGMOD*, vol. 44, no. 3, pp. 37–48, 2015.

[7] J. Lei, K. Chen, and B. Lynch, "Consistent community detection in multi-layer network data," *Biometrika*, vol. 107, no. 1, pp. 61–73, 2020.

[8] J. D. Wilson, J. Palowitch, S. Bhamidi, and A. B. Nobel, "Community extraction in multilayer networks with heterogeneous community structure," *JMLR*, vol. 18, no. 1, pp. 5458–5506, 2017.

[9] Z. Xu, S. Zhang, Y. Xia, L. Xiong, J. Xu, and H. Tong, "Destine: Dense subgraph detection on multi-layered networks," in *CIKM*, 2021, pp. 3558–3562.

[10] D. Liu and Z. Zou, "gcore: Exploring cross-layer cohesiveness in multi-layer graphs," *PVLDB*, vol. 16, no. 11, pp. 3201–3213, 2023.

[11] X. Huang, L. V. Lakshmanan, J. Xu, and H. Jagadish, *Community search over big graphs*. Springer, 2019, vol. 14.

[12] X. Liu and Z. Zou, "Common-truss-based community search on multi-layer graphs," in *ADMA*. Springer, 2023, pp. 277–291.

[13] D. Luo, Y. Bian, Y. Yan, X. Liu, J. Huan, and X. Zhang, "Local community detection in multiple networks," in *SIGKDD*, 2020, pp. 266–274.

[14] A. Behrouz, F. Hashemi, and L. V. Lakshmanan, "Firmtruss community search in multilayer networks," *PVLDB*, vol. 16, no. 3, pp. 505–518, 2022.

[15] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *VLDBJ*, vol. 29, no. 1, pp. 353–392, 2020.

[16] Y. Tang, J. Li, N. A. H. Haldar, Z. Guan, J. Xu, and C. Liu, "Reliable community search in dynamic networks," *PVLDB*, vol. 15, no. 11, pp. 2826–2838, 2022.

[17] T. Xu, Z. Lu, and Y. Zhu, "Efficient triangle-connected truss community search in dynamic graphs," *PVLDB*, vol. 16, no. 3, pp. 519–531, 2022.

[18] J. Kim, S. Luo, G. Cong, and W. Yu, "Dmcs: Density modularity based community search," in *SIGMOD*, 2022, pp. 889–903.

[19] Y. Jiang, Y. Fang, C. Ma, X. Cao, and C. Li, "Effective community search over large star-schema heterogeneous information networks," *PVLDB*, vol. 15, no. 11, pp. 2307–2320, 2022.

[20] Y. Zhou, Y. Fang, W. Luo, and Y. Ye, "Influential community search over large heterogeneous information networks," *PVLDB*, vol. 16, no. 8, pp. 2047–2060, 2023.

[21] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *PVLDB*, vol. 13, no. 6, pp. 854–867, 2020.

[22] Z. Wang, Y. Yuan, X. Zhou, and H. Qin, "Effective and efficient community search in directed graphs across heterogeneous social networks," in *ADC*. Springer, 2020, pp. 161–172.

[23] B. Liu, F. Zhang, C. Zhang, W. Zhang, and X. Lin, "Corecube: Core decomposition in multilayer graphs," in *WISE*. Springer, 2019, pp. 694–710.

[24] H. Huang, Q. Linghu, F. Zhang, D. Ouyang, and S. Yang, "Truss decomposition on multilayer graphs," in *IEEE BigData*. IEEE, 2021, pp. 5912–5915.

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

[26] Y. Fang, R. Cheng, Y. Chen, S. Luo, and J. Hu, "Effective and efficient attributed community search," *VLDBJ*, vol. 26, no. 6, pp. 803–828, 2017.

[27] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *SIGMOD*, 2014, pp. 1311–1322.

[28] X. Huang and L. V. Lakshmanan, "Attribute-driven community search," *PVLDB*, vol. 10, no. 9, pp. 949–960, 2017.

[29] L. Yuan, L. Qin, W. Zhang, L. Chang, and J. Yang, "Index-based densest clique percolation community search in networks," *TKDE*, vol. 30, no. 5, pp. 922–935, 2017.

[30] Y. Wang, X. Jian, Z. Yang, and J. Li, "Query optimal k-plex based community in graphs," *Data Science and Engineering*, vol. 2, pp. 257–273, 2017.

[31] Z. Dong, X. Huang, G. Yuan, H. Zhu, and H. Xiong, "Butterfly-core community search over labeled graphs," *PVLDB*, vol. 14, no. 11, pp. 2006–2018, 2021.

[32] Y. Sadikaj, Y. Velaj, S. Behzadi, and C. Plant, "Spectral clustering of attributed multi-relational graphs," in *SIGKDD*, 2021, pp. 1431–1440.

[33] S. Huang, H. Weng, and Y. Feng, "Spectral clustering via adaptive layer aggregation for multi-layer networks," *JCGS*, vol. 32, no. 3, pp. 1170–1184, 2023.

[34] M. El Gheche, G. Chierchia, and P. Frossard, "Orthonet: multilayer network data clustering," *TSIPN*, vol. 6, pp. 152–162, 2020.

[35] M. De Domenico, *Multilayer Networks: Analysis and Visualization: Introduction to muxViz with R*. Springer Nature, 2022.

[36] S. Gurukar, N. Pancha, A. Zhai, E. Kim, S. Hu, S. Parthasarathy, C. Rosenberg, and J. Leskovec, "Multibisage: A web-scale recommendation system using multiple bipartite graphs at pinterest," *PVLDB*, vol. 16, no. 4, pp. 781–789, 2022.

[37] J. Kim, T. Guo, K. Feng, G. Cong, A. Khan, and F. M. Choudhury, "Densely connected user community and location cluster search in location-based social networks," in *SIGMOD*, 2020, pp. 2199–2209.

[38] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *SIGKDD Explor. Newsl.*, vol. 14, no. 2, pp. 20–28, 2013.

[39] A. G. Carranza, R. A. Rossi, A. Rao, and E. Koh, "Higher-order clustering in complex heterogeneous networks," in *SIGKDD*, 2020, pp. 25–35.

[40] J. Zhang, S. Wang, Y. Sun, and Z. Peng, "Prerequisite-driven fair clustering on heterogeneous information networks," *PACMMOD*, vol. 1, no. 2, pp. 1–27, 2023.

[41] H. Zhao, Y. Zhou, Y. Song, and D. L. Lee, "Motif enhanced recommendation over heterogeneous information network," in *CIKM*, 2019, pp. 2189–2192.

[42] T. Xiang, A. Li, Y. Ji, and D. Li, "Knowledge based prohibited item detection on heterogeneous risk graphs," in *SIGKDD*, 2023, pp. 5260–5269.

[43] G. Wang, N. Ivanov, B. Chen, Q. Wang, T. Nguyen, and Q. Yan, "Graph learning for interactive threat detection in heterogeneous smart home rule data," *PACMMOD*, vol. 1, no. 1, pp. 1–27, 2023.

[44] Y. Yan, D. Luo, J. Ni, H. Fei, W. Fan, X. Yu, J. Yen, and X. Zhang, "Local graph clustering by multi-network random walk with restart," in *PAKDD*. Springer, 2018, pp. 490–501.

[45] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *TKDE*, vol. 29, no. 1, pp. 17–37, 2016.

[46] X. Jian, Y. Wang, and L. Chen, "Effective and efficient relational community detection and search in large dynamic heterogeneous information networks," *PVLDB*, vol. 13, no. 10, pp. 1723–1736, 2020.

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

[48] 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," *PNAS*, vol. 113, no. 48, pp. 13 708–13 713, 2016.