FirmTruss Community Search in Multilayer Networks

Ali Behrouz[†]
University of British Columbia
alibez@cs.ubc.ca

Farnoosh Hashemi[†]
University of British Columbia
farsh@cs.ubc.ca

Laks V.S. Lakshmanan
University of British Columbia
laks@cs.ubc.ca

ABSTRACT

In applications such as biological, social, and transportation networks, interactions between objects span multiple aspects. For accurately modeling such applications, multilayer networks have been proposed. Community search allows for personalized community discovery and has a wide range of applications in large real-world networks. While community search has been widely explored for single-layer graphs, the problem for multilayer graphs has just recently attracted attention. Existing community models in multilayer graphs have several limitations, including disconnectivity, free-rider effect, resolution limits, and inefficiency. To address these limitations, we study the problem of community search over large multilayer graphs. We first introduce *FirmTruss*, a novel dense structure in multilayer networks, which extends the notion of truss to multilayer graphs. We show that FirmTrusses possess nice structural and computational properties and bring many advantages compared to the existing models. Building on this, we present a new community model based on FirmTruss, called *FTCS*, and show that finding an FTCS community is NP-hard. We propose two efficient 2-approximation algorithms, and show that no polynomial-time algorithm can have a better approximation guarantee unless P = NP. We propose an index-based method to further improve the efficiency of the algorithms. We then consider attributed multilayer networks and propose a new community model based on network homophily. We show that community search in attributed multilayer graphs is NP-hard and present an effective and efficient approximation algorithm. Experimental studies on real-world graphs with ground-truth communities validate the quality of the solutions we obtain and the efficiency of the proposed algorithms.

PVLDB Reference Format:
Ali Behrouz, Farnoosh Hashemi, and Laks V.S. Lakshmanan. FirmTruss Community Search in Multilayer Networks. PVLDB, 16(1): XXX-XXX, 2023.

1 INTRODUCTION

Community detection is a fundamental problem in network science, and has been traditionally addressed with the aim of determining an organization of a given network into subgraphs that express dense groups of nodes well connected to each other [26]. Recently, a query-dependent community discovery problem, called community search (CS) [67], has attracted much attention due to its ability to discover personalized communities. It has several applications like social contagion modeling [72], content recommendation [11], and team formation [28]. The CS problem seeks a cohesive subgraph containing the query nodes given a graph and a set of query nodes.

Significant research effort has been devoted to the study of community search over single-layer large graphs, which have a single type of connection. However, in applications featuring complex networks such as social, biological, and transportation networks, the interactions between objects tend to span multiple aspects. E.g., interactions between people can be social or professional, and professional interactions can in turn differ according to topics. *Multilayer* (ML) *networks* [51], where nodes can have interactions in multiple layers, have been proposed for accurately modeling such applications. Recently, ML networks have gained popularity in an array of applications in social and biological networks and in opinion dynamics [14, 59, 62, 66], due to their more informative representation than single-layer graphs.

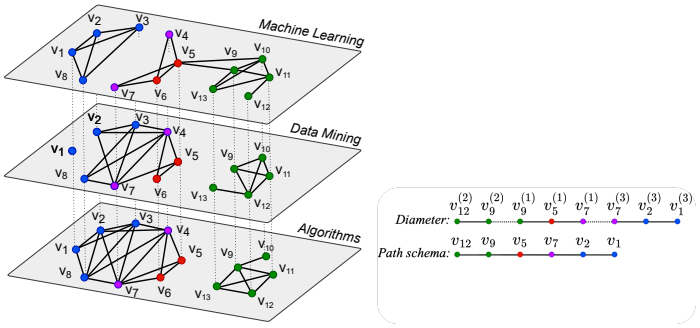
Example 1. *Figure 1(a) is an ML network showing a group of researchers collaborating in various topics, where each layer represents a different connection type, i.e., collaborations in an individual topic.*

To find cohesive communities in single-layer graphs, many models have been proposed, e.g., k-core [67, 68], k-truss [44], k-plex [75], and k-clique [15]. Existing methods for finding cohesive structures in ML networks are inefficient. As a result, there is a lack of practical density-based community models in ML graphs. Indeed, there have been a number of studies on cohesive structures in ML networks [29, 39, 57, 85]. However, they suffer from two main limitations. **(1)** The decomposition algorithms [29, 39, 57] based on these models have an *exponential running time complexity in the number of layers*, making them prohibitive for CS. **(2)** These models have a hard constraint that nodes/edges need to satisfy in *all* layers. It has been noted that ML networks may contain noisy/insignificant layers [29, 36]. These noisy/insignificant layers may be different for each node/edge. Therefore, this hard constraint could result in missing some dense structures [36]. Recently, FirmCore structure [36] in ML graphs has been proposed to address these limitations. However, a connected FirmCore can be disconnected by just removing one edge, and it might have an arbitrarily large diameter. Both of these properties are undesirable for community models.

In addition to the above drawbacks of cohesive structures in ML networks, existing CS methods in ML graphs (e.g., [30, 45, 60]) suffer from some important limitations. **(1)** *Free-rider effect* [77]: some cohesive structure, irrelevant to the query vertices, could be included in the answer community. **(2)** *Lack of connectivity*: a community, at a minimum, needs to be a connected subgraph [44, 78], but existing community models in ML graphs are not guaranteed to be connected. Natural attempts to enforce connectivity in these models lead to additional complications (see § 5.3 for a detailed comparison with previous community models). **(3)** *Resolution Limit* [27]: in a large network, communities smaller than a certain size may not

[†]These authors contributed equally.

Proceedings of the VLDB Endowment, Vol. 16, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX



(a) Multilayer network G (b) Diameter (its path schema) of G

Figure 1: An example of an multilayer collaboration network.

be detected. **(4)** *Failure to scale*: to be applicable to large networks, a community model must admit scalable algorithms. To the best of our knowledge, all existing models suffer from these limitations.

To address the above limitations of existing studies, we study the problem of CS over multilayer networks. First of all, we propose the notion of (k, λ)-*FirmTruss*, based on the truss structure in simple graphs, as a subgraph (not necessarily induced) in which every two adjacent nodes in at least λ individual layers are in at least $k-2$ common triangles within the subgraph. We show that it inherits the nice properties of trusses in simple graphs, viz., uniqueness, hierarchical structure, bounded diameter, edge-connectivity, and high density. Based on FirmTruss, we formally define our problem of *FirmTruss Community Search* (FTCS). Specifically, given a set of query nodes, FTCS aims to find a connected subgraph which **(1)** contains the query nodes; **(2)** is a FirmTruss; and **(3)** has the minimum diameter. We formally show that the diameter constraint in FTCS definition avoids the so-called "free-rider effect".

In real-world networks, nodes are often associated with attributes. For example, they could represent a summary of a user's profile in social networks, or the molecular functions, or cellular components of a protein in protein-protein interaction networks. This rich information can help us find communities of superior quality. While there are several studies on single-layer attributed graphs, to the best of our knowledge, the problem of CS in multilayer attributed networks has not been studied. Unfortunately, even existing CS methods in single-layer attributed graphs suffer from significant limitations. They *require* users to input query attributes; however, users not familiar with the attribute distribution in the entire network, are limited in their ability to specify proper query attributes. Moreover, these studies only focus on one particular type of attribute (e.g., keyword), while most real-world graphs involve more complex attributes. E.g., attributes of proteins can be multidimensional vectors [37]. The recently proposed VAC model [58] for single-layer graphs does not require users to input query attributes, but is limited to metric similarity measures. To mitigate these limitations, we extend our FTCS model to attributed ML graphs, call it AFTCS, and present a novel community model leveraging the well-known phenomenon of network homophily. This approach is based on maximizing the p-mean of similarities between users in a community and does not require users to input query attributes. However, should a user wish to specify query attributes (say for exploration), AFTCS can easily support them. Moreover, it naturally handles a vector of attributes, handling complex features.

Since ML graphs provide more complex and richer information than single-layer graphs, they can benefit typical applications of single-layer CS [22] (e.g., event organization, friend recommendation, advertisement, etc.), delivering better solutions. Below we illustrate an exclusive application for multilayer CS.

Brain Networks. Detecting and monitoring functional systems in the human brain is an important and fundamental task in neuroscience [7, 63]. A brain network (BN) is a graph in which nodes represent the brain regions and edges represent co-activation between regions. A BN generated from an individual subject can be noisy and incomplete, however using BNs from many subjects helps us identify important structures more accurately [53, 60]. A multilayer BN is a multilayer graph in which each layer represents the BN of a different person. A community search method in multilayer graphs can be used to **(1)** identify functional systems of each brain region; **(2)** identify common patterns between people's brains affected by diseases or under the influence of drugs; **(3)** select features in order to discriminate diseased patients from healthy individuals.

We make the following contributions: **(1)** We introduce a novel dense subgraph model for ML graphs, *FirmTruss*, and show that it retains the nice structural properties of Trusses (§ 4). **(2)** We formulate the problem of FirmTruss-based Community Search (FTCS) in ML graphs, and show the FTCS problem is NP-hard and cannot be approximated in PTIME within a factor better than 2 of the optimal diameter, unless P = NP (§ 5). **(3)** We develop two efficient 2-approximation algorithms (§ 6), and propose an indexing method to further improve efficiency (§ 7). **(4)** We extend FTCS to attributed networks and propose a novel homophily-based community model. We propose an exact algorithm for a special case of the problem and an approximation algorithm for the general case (§ 8). **(5)** Our extensive experiments on real-world ML graphs with ground-truth communities show that our algorithms can efficiently and effectively discover communities, significantly outperforming baselines (§ 9). For lack of space, some proofs are sketched. Complete details of all proofs and additional details can be found in [1].

2 RELATED WORK

Community Search. Community search, which aims to find query-dependent communities in a graph, was introduced by Sozio and Gionis [68]. Since then, various community models have been proposed, based on different dense subgraphs [22], including k-core [67, 68], k-truss [3, 42, 44], quasi-clique [15], k-plex [75], and densest subgraph [80]. Wu et al. [77] identified an undesirable phenomenon, called free-rider effect, and propose query biased density to reduce the free-rider effect for the returned community. More recently, CS has also been investigated for directed [23, 24], weighted [83], geo-social [35, 84], temporal [56], multi-valued [55], and labeled [19] graphs. Recently, graph neural network-based CS is studied [31, 47]. All these models are different from our work as they focus on a single type of interactions.

Attributed Community Search. Given a set of query nodes, attributed CS finds the query-dependent communities in which nodes

share attributes [20, 43]. Most existing works on attributed single-layer graphs can be classified into two categories. The first category takes both nodes and attributes as query input [13, 21]. The second category takes only attributes as input, and returns the community related to the query attributes [12, 86]. All these studies **(1)** require users to specify attributes as input, and **(2)** consider only simple attributes (e.g., keywords), limiting their applications. Most recently, Liu et al. [58] introduced VAC in single-layer graphs, which does not require input query attributes. However, they are restricted to metric similarity between users, which can limit applications. All these models are limited to single-layer graphs.

Community Search and Detection in ML Networks. Several methods have been proposed for community detection in ML networks [40, 41, 71]. However, they focus on detecting all communities, which is time-consuming and independent of query nodes. Surprisingly, the problem of CS in ML networks is relatively less explored. Interdonato et al. [45] design a greedy search strategy by maximizing the ratio of similarity between nodes inside and outside of the local community, over all layers. Galimberti et al. [30] adopt a community search model based on the ML **k**-core [6]. Finally, Luo et al. [60] design a random walk strategy to search local communities in multi-domain networks.

Dense Structures in ML Graphs. Jethava et al. [46] formulate the densest common subgraph problem. Azimi et al. [6] propose a new definition of core, **k**-core, over ML graphs. Galimberti et al. [29] propose algorithms to find all possible **k**-cores, and define the densest subgraph problem in ML graphs. Zhu et al. [85] introduce the problem of diversified coherent d-core search. Liu et al. [57] propose the CoreCube problem for computing ML d-core decomposition on all subsets of layers. Hashemi et al. [36] propose a new dense structure, FirmCore, and develop a FirmCore-based approximation algorithm for the problem of ML densest subgraph. Huang et al. [39] define TrussCube in ML graphs, which aims to find a subgraph in which each edge has support $\geq k-2$ in *all selected layers*, which is different from the concept of FirmTruss. ML graphs can be related to heterogeneous graphs, where communities [25, 69, 79] are defined based on meta patterns. However, these models emphasize heterogeneous types of entities connected by different relationships, which is different from the concept of ML graphs.

3 PRELIMINARIES

We let $G = (V, E, L)$ denote an ML graph, where V is the set of nodes, L the set of layers, and $E \subseteq V \times V \times L$ the set of intra-layer edges. We follow the common definition of ML networks [51], and consider inter-layer edges between two instances of identical vertices in different layers. The set of neighbors of node $v \in V$ in layer $\ell \in L$ is denoted $N_\ell(v)$ and the degree of v in layer ℓ is $\deg_\ell(v) = |N_\ell(v)|$. For a set of nodes $H \subseteq V$, $G[H] = (H, E[H], L)$ denotes the subgraph of G induced by H, $G_\ell[H] = (H, E_\ell[H])$ denotes this subgraph in layer ℓ, and $\deg_\ell^H(v)$ denotes the degree of v in this subgraph. Abusing notation, we write $G_\ell[V]$ and $E_\ell[V]$ as G_ℓ and E_ℓ, respectively. We use the following notions in this paper.

Edge Schema. Connections (i.e., relationships) between objects in ML networks can have multiple types; by the *edge schema* of a connection, we mean the connection ignoring its type.

Definition 1 (Edge Schema). *Given an ML network $G = (V, E, L)$ and an intra-layer edge $e = (v, u, \ell) \in E$, the edge schema of e is the pair $\varphi = (v, u)$, which represents the relationship between two nodes, v and u, ignoring its type. We denote by $\mathcal{E}$ the set of all edge schemas in G, $\mathcal{E} = \{(v, u) | \exists \ell \in L : (v, u, \ell) \in E\}$.*

Given an edge schema $\varphi = (v, u)$, we abuse the notation and use φ_ℓ to refer to the relationship between v and u in layer ℓ, i.e., $\varphi_\ell = (v, u, \ell)$, whenever $(v, u, \ell) \in E$.

Distance in ML Networks. For consistency, we use the common definition of ML distance [4] in the literature. However, our algorithms are valid for any definition of distance that is a metric.

Definition 2 (Path in Multilayer Networks). *Let $G = (V, E, L)$ be an ML graph and v_ℓ represent a node v in layer $\ell \in L$. A path in G is a sequence of nodes $\mathcal{P} : v^1_{\ell_1} \rightarrow v^2_{\ell_2} \rightarrow \cdots \rightarrow v^k_{\ell_k}$ such that every consecutive pair of nodes is connected by an inter-layer or intra-layer edge, i.e., $v^i = v^{i+1}$ or $[\ell_i = \ell_{i+1}$ & $(v^i, v^{i+1}, \ell_i) \in E]$. The path schema $\mathfrak{P}$ of $\mathcal{P}$ is obtained by removing inter-layer edges from path $\mathcal{P}$.*

Note that inter-layer edges between identical nodes are used as a penalty for changing edge types in a path. We define the distance of two nodes v and u, $dist(v, u)$, as the length of the shortest path between them. The diameter of a subgraph $G[H]$, $diam(G[H])$, is the maximum distance between any pair of nodes in $G[H]$.[1]

Example 2. *In Figure 1(a), the diameter of ML graph G is 7, corresponding to the path (path schema) in Figure 1(b).*

Density in ML Networks. In this study, we use a common definition of density in multilayer graphs proposed in [30].

Definition 3 (Density). *[30] Given an ML graph $G = (V, E, L)$, a non-negative real number β, the density function is a real-valued function $\rho_\beta : 2^V \rightarrow \mathbb{R}^+$, defined as:*

$$\rho_\beta(S) = \max_{\hat{L} \subseteq L} \min_{\ell \in \hat{L}} \frac{|E_\ell[S]|}{|S|} |\hat{L}|^\beta.$$

Free-Rider Effect. Prior work has identified an undesirable phenomenon known as the "free-rider effect" [77]. Intuitively, if a community definition admits irrelevant subgraphs in the discovered community, we refer to the irrelevant subgraphs as free riders. Typically, a community definition is based on a goodness metric $f(S)$ for a subgraph S: subgraphs with the highest (lowest) $f(.)$ value are identified as communities.

Definition 4 (Free-Rider Effect). *Given an ML graph $G = (V, E, L)$, a non-empty set of query vertices Q, let H be a solution to a community definition that maximizes (resp. minimizes) goodness metric $f(.)$, and H^* be a (global or local) optimum solution when our query set is empty. If $f(H^* \cup H) \geq f(H)$ (resp. $f(H^* \cup H) \leq f(H)$), we say that the community definition suffers from free rider effect.*

Example 3. *For an ML graph $G = (V, E, L)$ and subgraph $H \subseteq G$, define $f(H)$ to be the minimum of the minimum node degrees in H over all its layers. Let $H^* = \arg\max_{H \subseteq G} f(H)$. For any subgraph $H \subseteq G$, we have $f(H^* \cup H) \geq f(H)$. Therefore, this goodness metric $f(.)$ for community search suffers from the free-rider effect.*

[1] For convenience, we refer to both the longest shortest path distance as well as any path with that length as diameter.

Generalized Means. Given a finite set of positive real numbers $S = \{a_1, a_2, \ldots, a_n\}$, and a parameter $p \in \mathbb{R} \cup \{-\infty, +\infty\}$, the generalized mean (*p-mean*) of S is defined as

$$M_p(S) = \left(\frac{1}{|S|} \sum_{i=1}^{|S|} (a_i)^p \right)^{1/p}.$$

For $p \in \{-\infty, 0, +\infty\}$, the mean can be defined by taking limits, so that $M_{+\infty}(S) = \max a_i$, $M_0(S) = (\prod_{i=1}^{|S|} a_i)^{1/|S|}$, and $M_{-\infty}(S) = \min a_i$.

4 FIRMTRUSS STRUCTURE

In this section, we first recall the notion of k-truss in single-layer networks and then present FirmTruss structure in ML networks.

Definition 5 (Support). *Given a single-layer graph $G = (V, E)$, the support of an edge $e = (u, v) \in E$, denoted $sup(e, G)$, is defined as $|\{\triangle_{u,v,w} : u, v, w \in V\}|$, where $\triangle_{u,v,w}$, called triangle of u, v, and w, is a cycle of length three containing nodes u, v, and w.*

The k-truss of a single-layer graph G is the maximal subgraph $H \subseteq G$, such that $\forall e \in H, sup(e, H) \geq (k-2)$. Since each layer of an ML network can be counted as a single-layer network, one possible extension of truss structure is to consider different truss numbers for each layer, separately. However, this approach forces all edges to satisfy a constraint in all layers, including noisy/insignificant layers. This hard constraint would result in missing some dense structures [36]. Next, we suggest FirmTruss, a new family of cohesive structures based on the k-truss of single-layer networks.

Definition 6 (FirmTruss). *Given an ML graph $G = (V, E, L)$, its edge schema set $\mathcal{E}$, an integer threshold $1 \leq \lambda \leq |L|$, and an integer $k \geq 2$, the (k, λ)-FirmTruss of G ((k, λ)-FT for short) is a maximal subgraph $G[J_k^\lambda] = (J_k^\lambda, E[J_k^\lambda], L)$ such that for each edge schema $\varphi \in \mathcal{E}[J_k^\lambda]$ there are at least λ layers $\{\ell_1, ..., \ell_\lambda\} \subseteq L$ such that $\varphi_{\ell_i} \in E_{\ell_i}[J_k^\lambda]$ and $sup(\varphi_{\ell_i}, G_{\ell_i}[J_k^\lambda]) \geq (k-2)$.*

Example 4. *In Figure 1(a), let $k = 4, \lambda = 2$. The union of blue and purple nodes is a $(4, 2)$-FirmTruss, as every pair of adjacent nodes in at least 2 layers are in at least 2 common triangles within the subgraph.*

For each edge schema $\varphi = (u, v) \in \mathcal{E}$, we consider an $|L|$-dimensional support vector, denoted $\mathbf{S}_\varphi$, in which i-th element, $\mathbf{S}_\varphi^i$, denotes the support of the corresponding edge of φ in i-th layer. We define the *Top-λ support* of φ as the λ-th largest value in φ's support vector. Next, we show that not only is the maximal (k, λ)-FirmTruss unique, it also has the nested property.

Property 1 (Uniqueness). *The (k, λ)-FirmTruss of G is unique.*

Property 2 (Hierarchical Structure). *Given a threshold $\lambda \in \mathbb{N}^+$, and an integer $k \geq 0$, the $(k+1, \lambda)$-FT and $(k, \lambda+1)$-FT of G are subgraphs of its (k, λ)-FT.*

Property 3 (Minimum Degree). *Let $G = (V, E, L)$ be an ML graph, and $H = G[J_k^\lambda]$ be its (k, λ)-FT. Then $\forall$ node $u \in J_k^\lambda$, there are at least λ layers $\{\ell_1, ..., \ell_\lambda\} \subseteq L$ such that $deg_{\ell_i}^H(u) \geq k-1, 1 \leq i \leq \lambda$.*

In ML networks, the degree of a node v is an $|L|$-dimensional vector whose i-th element is the degree of node v in i-th layer. Let Top-λ degree of v be the λ-th largest value in the degree vector of v. By Property 3, each node in a (k, λ)-FirmTruss has a Top-λ degree of at least $k-1$. That means, each (k, λ)-FirmTruss is a $(k-1, \lambda)$-FirmCore [36]. Like trusses, a FirmTruss may be disconnected, and we refer to its connected components as *connected FirmTrusses*.

Trusses are known to be dense, cohesive, and stable structures. These important characteristics of trusses make them popular for modeling communities [44]. Next, we discuss the density, closeness, and edge connectivity of FirmTrusses. Detailed proofs of the results and tightness examples can be found in [1], Appendix A.2.

Theorem 1 (Density Lower Bound). *Given an ML graph $G = (V, E, L)$, the density of a (k, λ)-FirmTruss, $G[J_k^\lambda] \subseteq G$, satisfies:*

$$\rho_\beta(J_k^\lambda) \geq \frac{(k-1)}{2|L|} \max_{\xi \in \mathbb{Z}, 0 \leq \xi < \lambda} (\lambda - \xi)(\xi + 1)^\beta.$$

Example 5. *Let $\beta = 0$, in Figure 1(a), the entire graph is a (4,1)-FirmTruss with a density of $\frac{25}{13} \times 1 \approx 1.92$. Theorem 1 provides a lower bound of $\frac{3}{2 \times 3} \times 1 = 0.5$ on the density.*

Theorem 2 (Diameter Upper Bound). *Given an ML graph $G = (V, E, L)$, the diameter of a connected (k, λ)-FirmTruss, $G[J_k^\lambda] \subseteq G$, is no more than $T \times \lfloor \frac{2|J_k^\lambda|-2}{k} \rfloor$, where $T = 1 + \frac{1}{\lfloor \frac{|L|}{|L|-\lambda} \rfloor}$.*

Proof Sketch. We show that if $\mathcal{P}$ is the diameter of the (k, λ)-FT, and $\frac{t}{t+1}|L| > \lambda \geq \frac{t-1}{t}|L|$, then its path schema, $\mathfrak{P}$, has a length at least $\frac{t}{t+1} \times |\mathcal{P}|$. Then we consider every t consecutive edges in the diameter as a block and construct a path, with the same path schema as $\mathcal{P}$ such that edges in each block are in the same layer. Next, we use edge schema supports to bound its length in each block. □

Example 6. *In Figure 1(a), the union of blue and purple nodes is a connected $(4, 2)$-FirmTruss with diameter 2. Theorem 2 provides the upper bound of $\lfloor \frac{4}{3} \times \lfloor \frac{2 \times 6 - 2}{4} \rfloor \rfloor = \lfloor \frac{8}{3} \rfloor = 2$ on its diameter.*

Theorem 3 (Edge Connectivity). *For an ML graph $G = (V, E, L)$, any connected (k, λ)-FirmTruss $G[J_k^\lambda] \subseteq G$ remains connected whenever fewer than $\lambda \times (k-1)$ intra-layer edges are removed.*

5 FIRMTRUSS-BASED COMMUNITY SEARCH

5.1 Problem Definition

In this section, we propose a community model based on FirmTruss in ML networks. Generally, a community in a network is identified as a set of nodes that are densely connected. Thus, we use the notion of FirmTruss for modeling a densely connected community in ML graphs, which inherits several desirable structural properties, such as high density (Theorem 1), bounded diameter (Theorem 2), edge connectivity (Theorem 3), and hierarchical structure (Property 2).

Problem 1 (FirmTruss Community Search). *Given an ML network $G = (V, E, L)$, two integers $k \geq 2$ and $\lambda \geq 1$, and a set of query vertices $Q \subseteq V$, the FirmTruss community search (FTCS) is to find a connected subgraph $G[H] \subseteq G$ satisfying:*

(1) $Q \subseteq H$,
(2) $G[H]$ is a connected (k, λ)-FirmTruss,
(3) diameter of $G[H]$ is the minimum among all subgraphs satisfying conditions (1) and (2).

Here, Condition (1) requires that the community contains the query vertex set Q, Condition (2) makes sure that the community is densely connected through a sufficient number of layers, and Condition (3) requires that each vertex in the community be as close to other vertices as possible, which excludes irrelevant vertices from the community. Together, all three conditions ensure that the returned community is a cohesive subgraph with good quality.

Example 7. *In the graph shown in Figure 1, let v_1 be the query node, $k = 4$, and $\lambda = 2$. The union of purple and blue nodes is a $(4, 2)$-FirmTruss, with diameter 2. The FTCS community removes purple nodes to reduce the diameter. Let v_6 be the query node, $k = 4$, and $\lambda = 1$, the entire graph is a $(4, 1)$-FirmTruss, with diameter 7. The FTCS community removes blue and green nodes to reduce the diameter.*

Why FirmTruss Structure? Triangles are fundamental building blocks of networks, which show a strong and stable relationship among nodes [76]. In ML graphs, every two nodes can have different types of relations, and a connection can be counted as strong and stable if it is a part of a triangle in each type of interaction. However, forcing all edges to be a part of a triangle in every interaction type is too strong a constraint. Indeed, TrussCube [39], which is a subgraph in which each edge has support $k - 2$ in all selected layers, is based on this strong constraint. In Figure 1, the green nodes are densely connected. However, while this subgraph is a $(4, 1)$-FirmTruss, due to the hard constraint of TrussCube, green nodes are a 2-TrussCube, meaning that this model misses it. That is, even if the green subgraph were to be far less dense and have no triangles in it, it would still be regarded as 2-TrussCube. Furthermore, in some large networks, there is no non-trivial TrussCube when the number of selected layers is more than 3 [39]. In addition to these limitations, the exponential-time complexity of its algorithms makes it impractical for large ML graphs. By contrast, FirmTrusses have a polynomial-time algorithm, with guaranteed high density, bounded diameter, and edge connectivity. While FirmCore [36] also has a polynomial-time algorithm, a connected FirmCore can be disconnected by just removing one edge, and it might have an arbitrarily large diameter, which are both undesirable for communities.

5.2 Problem Analysis

Next we analyze the hardness of the FTCS problem and show not only that it is NP-hard, but it cannot be approximated within a factor better than 2. Thereto, we define the decision version of the FTCS, d-FTCS, to test whether G contains a connected FirmTruss community with diameter $\leq d$, that contains Q. Given $\alpha \geq 1$ and the optimal solution to FTCS, $G[H^*]$, an algorithm achieves an α-approximation to FTCS if it outputs a connected (k, λ)-FirmTruss, H, such that $Q \subseteq H$ and $diam(G[H]) \leq \alpha \times diam(G[H^*])$.

Theorem 4 (FTCS Hardness and Non-Approximability). *Not only the d-FTCS problem is NP-hard, but also for any $\epsilon > 0$, the FTCS-problem cannot be approximated in polynomial-time within a factor $(2 - \epsilon)$ of the optimal solution, unless $P = NP$.*

Proof Sketch. Given an instance $G = (V, E)$ of k-CLIQUE, we construct ML graph $G' = (V', E', L)$, where $V = V'$ and $\forall \ell \in L : E'_\ell = E$, and show that if $\exists$ such an approximation algorithm, by running it on G', we can check whether G contains a k-clique in polynomial-time. Proof of NP-hardness of d-FTCS is similar. □

In § 6, we provide a 2-approximation algorithm for FTCS, thus essentially matching this lower bound.

Avoiding Free-rider Effect. We can show:

Theorem 5 (FTCS Free-Rider Effect). *For any multilayer network $G = (V, E, L)$ and query vertices $Q \subseteq V$, there is a solution $G[H]$ to the FTCS problem such that for all query-independent optimal solutions $G[H^*]$, either $H^* = H$, or $G[H \cup H^*]$ is disconnected, or $G[H \cup H^*]$ has a strictly larger diameter than $G[H]$.*

5.3 Comparison of CS Models in ML Networks

We compare FirmTruss with existing CS models for ML networks.

Cohesiveness. In the literature, communities are defined as cohesive, densely connected subgraphs. Hence, cohesiveness, i.e., high density, is an important metric to measure the quality of communities. It is shown that FirmCore can find subgraphs with higher density than the ML **k**-core [36]. Since each (k, λ)-FirmTruss is a $(k - 1, \lambda)$-FirmCore (Property 3), FirmTruss is more cohesive than ML **k**-core. ML-LCD model [45] maximizes the similarity of nodes within the subgrpah. RWM [60] is a random walk-based approach and minimizes the conductance. Both of these models do not control the density of the subgraph. Thus, one node may have degree 1 within the subgraph, allowing non-cohesive structures.

Connectivity. A minimal requirement for a community is to be a connected subgraph. Surprisingly, ML **k**-core, ML-LCD, and RWM (with multiple query nodes) community search models do not guarantee connectivity! Natural attempts to enforce connectivity in these community models lead to additional complications and might change the hardness of the problem. Even after enforcing connectivity, these models can be disconnected by just removing one intra-layer edge, which is undesirable for community models [38]. Our FirmTruss community model forces the subgraph to be connected, and guarantees that after removing up to $\lambda \times (k - 1)$ intra-layer edges, the (k, λ)-FirmTruss is still connected (Theorem 3).

Edge Redundancy. In ML networks, the rich information about node connections leads to repetitions, meaning edges between the same pair of nodes repeatedly appear in multiple layers. Nodes with repeated connections are more likely to belong to the same community [81]. Also, without such redundancy of connections, the tight connection between objects in ML networks may not be represented effectively and accurately. While none of the models ML **k**-core, ML-LCD, and RWM guarantees edge redundancy, in a (k, λ)-FirmTruss, each edge is required to appear in at least λ layers.

Hierarchical Structure. The hierarchical structure is a desirable property for community search models as it represents a community at different levels of granularity, and can also avoid the Resolution Limit problem as is discussed in [27]. While FirmTruss has a hierarchical structure, none of the existing models has this property.

6 FTC ONLINE SEARCH

Given the hardness of the FTCS problem, we propose two online 2-approximation algorithms in top-down and bottom-up manner.

6.1 Global Search

We start by defining query distance in multilayer networks.

Algorithm 1: FTCS Global Search

Input : An ML graph $G = (V, E, L)$, a set of query vertices $Q \subseteq V$, and two integers $k \geq 2$ and $\lambda \geq 1$
Output: A connected (k, λ)-FT containing Q with a small diameter

1 $G_0 \leftarrow$ Find a maximal connected (k, λ)-FirmTruss containing Q; // See Algorithm 2 (or Algorithm 5)
2 $i \leftarrow 0$; $d_{\min} \leftarrow 1$; $d_{\max} \leftarrow \text{dist}_{G_0}(G_0, Q)$; $\mathcal{G} \leftarrow G_0$;
3 **while** $d_{\min} < d_{\max}$ **do**
4 $\quad d_{avg} \leftarrow \lfloor \frac{d_{\min}+d_{\max}}{2} \rfloor$; $G' \leftarrow \mathcal{G}$
5 $\quad S \leftarrow$ set of vertices with $d_{avg} \leq \text{dist}_{G'}(u, Q)$;
6 $\quad$ Delete nodes in S and their incident edges from G' in all layers;
7 $\quad$ Maintain G' as (k, λ)-FirmTruss by removing vertices/edges;
8 $\quad$ **if** $Q \nsubseteq G'$ ***or*** G' *is disconnected* ***or*** $d_{\max} < dist_{G'}(G', Q)$ **then**
9 $\quad\quad d_{\min} \leftarrow 1 + d_{avg}$;
10 $\quad$ **else**
11 $\quad\quad d_{\max} \leftarrow \text{dist}_{G'}(G', Q)$;
12 $\quad\quad$ Let the remaining graph G' as $\mathcal{G}$;
13 **return** $\mathcal{G}$;

Definition 7 (Query Distance). *Given a multilayer network $G = (V, E, L)$, a subgraph $G[H] \subseteq G$, a set of query vertices $Q \subseteq H$, and a vertex set $S \subseteq H$, the query distance of S in $G[H]$, $dist_{G[H]}(S, Q)$, is defined as the maximum length of the shortest path from $u \in S$ to a query vertex $q \in Q$, i.e., $dist_{G[H]}(S, Q) = \max_{u \in S, q \in Q} dist(u, q)$.*

For a graph G, we use $dist_G(u, Q)$ to denote the query distance for a vertex $u \in V$. Previous works (e.g., see [19, 44]) use a simple greedy algorithm which iteratively removes the nodes with maximum distance to query nodes, in order to minimize the query distance. This approach can be inefficient, as it reduces the query distance by just 1 in each iteration, in the worst case. We instead employ a binary search on the query distance of a subgraph.

Algorithm 1 gives the details of the FTCS Global algorithm. It first finds a maximal connected (k, λ)-FirmTruss G_0 containing Q. We keep our best found subgraph in $\mathcal{G}$, through the algorithm. Then in each iteration, we make a copy of $\mathcal{G}$, G', and for each vertex $u \in V[G']$, we compute the query distance of u. Then, we conduct a binary search on the value of d_{avg} and delete vertices with query distance $\geq d_{avg}$ and all their incident edges, in all layers. From the resulting graph we remove edges/vertices to maintain G' as a (k, λ)-FirmTruss (lines 6 and 7). We maintain the (k, λ)-FirmTruss by deleting the edge schemas whose Top-λ support is $< k - 2$. Finally, the algorithm returns a subgraph $\mathcal{G}$, with the smallest query distance.

The procedure for finding the maximal FirmTruss containing Q is given in Algorithm 2. Notice, a (k, λ)-FirmTruss (see Def. 6) is a maximal subgraph $G[J_k^\lambda]$ in which each edge schema $\varphi \in \mathcal{E}[J_k^\lambda]$ has Top-λ support $\geq k - 2$. The algorithm first uses Property 3, and removes all vertices with Top-λ degree $< k - 1$. It then iteratively deletes all instances of disqualified edge schemas in all layers from the original graph G, and then updates the Top-λ support of their adjacent edges. To do this efficiently, we use the following fact:

Fact 1. *If two edge schemas φ and φ' are adjacent in layer ℓ, removing edge schema φ cannot affect Top–$\lambda(\mathbf{S}_{\varphi'})$, unless Top–$\lambda(\mathbf{S}_{\varphi'}) = \mathbf{S}^\ell_{\varphi'}$.*

Thus, in lines 12-20, we update the Top-λ support of those edge schemas whose Top-λ support may be affected by removing φ. Finally, we use BFS traversal from a query node $q \in Q$ to find the connected component including query vertices. We omit the details of FirmTruss maintenance since it can use operations similar to those in lines 8-21 of Algorithm 2.

Algorithm 2: Maximal (k, λ)-FirmTruss containing Q

Input : An ML graph $G = (V, E, L)$, a set of query nodes $Q \subseteq V$, and integers $k \geq 2$ and $\lambda \geq 1$
Output: A maximal connected (k, λ)-FirmTruss containing Q

1 $G' \leftarrow$ Remove all vertices with Top-λ degree less than $k - 1$;
2 Compute $\mathbf{S}^\ell_\varphi = sup(\varphi_\ell, G'_\ell)$ for each edge schema $\varphi \in \mathcal{E}$ and $\ell \in L$;
3 $N, B \leftarrow \emptyset$;
4 **forall** $\varphi \in \mathcal{E}[G']$ **do**
5 $\quad I[\varphi] \leftarrow$ Top-$\lambda(\mathbf{S}_\varphi) + 2$;
6 $\quad$ **if** $I[\varphi] < k$ **then**
7 $\quad\quad N \leftarrow N \cup \{\varphi\}$;
8 **while** $N \neq \emptyset$ **do**
9 $\quad$ Pick and remove $\varphi = (v, u)$ from N;
10 $\quad$ **forall** $(v, w, \ell) \in E[G']$ *and* $I[(v, w)] \geq k$ *and* $\varphi_\ell \in E$ **do**
11 $\quad\quad$ **if** $(u, w, \ell) \in E[G']$ *and* $I[(u, w)] \geq k$ **then**
12 $\quad\quad\quad$ **if** $\mathbf{S}^\ell_{(v,w)} + 2 = I[(v, w)]$ **then**
13 $\quad\quad\quad\quad B \leftarrow B \cup \{(v, w)\}$;
14 $\quad\quad\quad$ **if** $\mathbf{S}^\ell_{(u,w)} + 2 = I[(u, w)]$ **then**
15 $\quad\quad\quad\quad B \leftarrow B \cup \{(u, w)\}$;
16 $\quad\quad\quad \mathbf{S}^\ell_{(v,w)} \leftarrow \mathbf{S}^\ell_{(v,w)} - 1$; $\mathbf{S}^\ell_{(u,w)} \leftarrow \mathbf{S}^\ell_{(u,w)} - 1$;
17 $\quad$ **forall** $\varphi' = (w, t) \in B$ **do**
18 $\quad\quad$ Update $I[\varphi']$;
19 $\quad\quad$ **if** $I[\varphi'] < k$ **then**
20 $\quad\quad\quad N \leftarrow N \cup \{\varphi'\}$;
21 $\quad$ Remove all instance of φ from G' in all layers;
22 $H \leftarrow$ The connected component of G' containing Q;
23 **return** H;

Example 8. *In Figure 1, let $k = 4$, $\lambda = 2$, and $Q = \{v_2\}$. Algorihtm 2 first calculates the support of each edge schema. Next, it removes the edge schema $\varphi = (v_{12}, v_{13})$ in all layers, as its Top-2 support is 0. Next, it updates the support of edge schema adjacent to φ, and iteratively removes all edges between green, red, and purple nodes since their edge schema has Top-2 support less than 2. Finally, the remaining graph, the union of blue of purple nodes, is returned by the algorithm.*

Example 9. *In Figure 1, let $k = 4$, $\lambda = 1$, and $Q = \{v_1\}$. Algorithm 1 starts from the entire graph as G_0. Since the query distance is 7, it sets $d_{avg} = \frac{7+1}{2} = 4$, removes all nodes with query distance ≥ 4, and maintains the remaining graph as $(4, 1)$-FirmTruss. The remaining graph includes blue, purple, and red nodes. Next, it sets $d_{avg} = \lfloor \frac{3+1}{2} \rfloor = 2$, removes all vertices with query distance ≥ 2, and maintains the remaining graph as a $(4, 1)$-FirmTruss, which includes blue nodes. Algorithm 1 terminates and returns this subgraph as the solution.*

Next, we analyze the approximation quality and complexity of the FTCS Global algorithm.

Theorem 6 (FTCS-Global Quality Approximation). *Algorithm 1 achieves 2-approximation to an optimal solution $G[H^*]$ of the FTCS problem, that is, the obtained (k, λ)-FirmTruss, $G[H]$ satisfies*

$$diam(G[H]) \leq 2 \times diam(G[H^*]).$$

Lemma 1. *Algorithm 2 takes $O(\sum_{\ell \in L} |E_\ell|^{1.5} + |E||L| + |E|\lambda \log |L|)$ time, and $O(|E||L|)$ space.*

Algorithm 3: FTCS Local Search

Input : An ML graph $G = (V, E, L)$, a set of query vertices $Q \subseteq V$, and two integers $k \geq 2$ and $\lambda \geq 1$
Output: A connected (k, λ)-FT containing Q with a small diameter

1 $d_{\min} \leftarrow 1$; $d_{\text{mid}} \leftarrow 1$; $G_{\text{out}} \leftarrow \emptyset$; $d_{\max} \leftarrow \infty$; $V' = \emptyset$;
2 **while** $d_{\min} < d_{\max}$ ***and*** $V' \neq V$ **do**
3 $V' \leftarrow Q \cup \{u \in V | \text{dist}_G(u, Q) \leq d_{\text{mid}}\}$;
4 $G' \leftarrow$ Induced subgraph of G by vertices V';
5 $G' \leftarrow$ Find maximal (k, λ)-FirmTruss of G' containing Q;
6 **while** $G' \neq \emptyset$ **do**
7 $N \leftarrow \emptyset$;
8 **for** $u \in V[G']$ **do**
9 **if** $dist_{G'}(u, Q) > d_{mid}$ **then**
10 $N \leftarrow N \cup \{u\}$;
11 **if** $N = \emptyset$ **then**
12 $d_{\max} \leftarrow d_{\text{mid}}$; $d_{\text{mid}} \leftarrow \lfloor \frac{d_{\min}+d_{\max}}{2} \rfloor$;
13 $G_{\text{out}} \leftarrow G'$;
14 **Break**; //Break in the inner **while** loop
15 **else**
16 Delete N and their incidents edges in all layers from G';
17 Maintain G' as (k, λ)-FirmTruss;
18 **if** $G' = \emptyset$ **then**
19 $d_{\min} \leftarrow d_{\text{mid}} + 1$; $d_{\text{mid}} \leftarrow 2 \times d_{\text{mid}}$;
20 **return** G_{out};

Theorem 7 (FTCS-Global Complexity). *Algorithm 1 takes* $O(\gamma(|Q||E[G_0]| + \sum_{\ell \in L} |E_\ell|^{1.5}) + |E||L| + |E|\lambda \log |L|)$ *time, and* $O(|E||L|)$ *space, where* $\gamma = \log\left(dist_{G_0}(G_0, Q)\right)$.

6.2 Local Search

The top-down approach of the Global algorithm may incur unnecessary computations over massive networks. The FTCS Local algorithm (Algorithm 3), presented next, addresses this limitation using a bottom-up approach.

We can first to collect all vertices whose query distances are $\leq d$ into V' (line 3) and then construct G' as the induced subgraph of G by V' (line 4). Next, given d, examine whether G' contains a (k, λ)-FirmTruss whose query distance is d. If such a FirmTruss exists, return it as the solution, and otherwise, increment d by 1 and iterate. One drawback of this approach is that it increases the query distance only by 1 in each iteration, which is inefficient. We instead conduct a binary search on the value of d. One challenge is the lack of upper bound on d. A trivial upper bound, which is the query distance in the entire graph, might lead to considering almost the entire graph in the first iteration. We instead use a doubling search whereby we double the query distance d in every iteration until a solution is found. Then by considering the resulting query distance as an upper bound on d, we conduct a binary search. Algorithm 3 shows the details.

Theorem 8 (FTCS-Local Quality Approximation). *Algorithm 3 achieves 2-approximation to an optimal solution* $G[H^*]$ *of the FTCS problem, that is, the obtained* (k, λ)*-FirmTruss,* $G[H]$ *satisfies*

$$diam(G[H]) \leq 2 \times diam(G[H^*]).$$

Proof Sketch. We first prove that the binary search method finds a solution with a smaller query distance than the optimal diameter solution. Next, by the triangle inequality, we show that the diameter of the found solution is at most twice the optimal. The detailed proof can be found in Appendix A.2, [1]. □

Algorithm 4: FirmTruss Decomposition

Input : An ML graph $G = (V, E, L)$
Output: Skyline FirmTruss index of each edge schema

1 Compute $\mathrm{S}^{\ell}_{\varphi} = sup(\varphi_\ell, G_\ell)$ for each edge schema $\varphi \in \mathcal{E}$ in each layer $\ell \in L$;
2 **forall** $\lambda = 1, 2, \ldots, |L|$ **do**
3 reinitialize supports, $\mathrm{S}^{\ell}_{\varphi}$;
4 **forall** $\varphi \in \mathcal{E}$ **do**
5 $I[\varphi] \leftarrow$ Top-$\lambda(\mathrm{S}_\varphi) + 2$;
6 $B[I[\varphi]] \leftarrow B[I[\varphi]] \cup \{\varphi\}$;
7 **forall** $k = 2, 3, \ldots, |V|$ **do**
8 **while** $B[k] \neq \emptyset$ **do**
9 Pick and remove $\varphi = (v, u)$ from $B[k]$;
10 $\mathsf{SFT}(\varphi) \leftarrow \mathsf{SFT}(\varphi) \cup (k, \lambda)$, $N \leftarrow \emptyset$;
11 **forall** $(v, w, \ell) \in E$ *and* $I[(v, w)] > k$ *and* $\varphi_\ell \in E$ **do**
12 **if** $(u, w, \ell) \in E$ *and* $I[(u, w)] > k$ **then**
13 **if** $\mathrm{S}^{\ell}_{(v,w)} + 2 = I[(v, w)]$ **then**
14 $N \leftarrow N \cup \{(v, w)\}$;
15 **if** $\mathrm{S}^{\ell}_{(u,w)} + 2 = I[(u, w)]$ **then**
16 $N \leftarrow N \cup \{(u, w)\}$;
17 $\mathrm{S}^{\ell}_{(v,w)} \leftarrow \mathrm{S}^{\ell}_{(v,w)} - 1$; $\mathrm{S}^{\ell}_{(u,w)} \leftarrow \mathrm{S}^{\ell}_{(u,w)} - 1$;
18 **forall** $\varphi' = (w, t) \in N$ **do**
19 Remove φ' from $B[I[\varphi']]$;
20 Update $I[\varphi']$;
21 $B[I[\varphi']] \leftarrow B[I[\varphi']] \cup \{\varphi'\}$;
22 Remove all instance of φ from G in all layers;
23 Remove all dominated indices in $\mathsf{SFT}(\varphi)$ for each $\varphi \in \mathcal{E}$;

Theorem 9 (FTCS-Local Complexity). *FTCS-Local algorithm takes* $O(\gamma(|Q||E| + \sum_{\ell \in L} |E_\ell|^{1.5}) + |E||L| + |E|\lambda \log |L|)$ *time, and* $O(|E||L|)$ *space, where* $\gamma = \log\left(dist_{G_0}(G_0, Q)\right)$.

7 INDEX-BASED ALGORITHM

Both online algorithms need to find FirmTruss from scratch. However, for each query set, computing the maximal FirmTruss from scratch can be inefficient for large multilayer networks. In this section, we discuss how to employ FirmTruss decomposition to accelerate our algorithms, by storing maximal FirmTrusses as they are identified into an index structure. We first present our FirmTruss decomposition algorithm and then describe how the index can be used for efficient retrieval of the maximal FirmTruss given a query.

7.1 FirmTruss Decomposition

In this section, we define the Skyline FirmTrussness index. For an edge schema $\varphi \in \mathcal{E}$, we let $FTI(\varphi)$ denote the set $\{(k, \lambda) \mid \varphi$ is in a (k, λ)-FirmTruss$\}$. We will use the following notion of index dominance.

Definition 8 (Index Dominance). *Given two pairs of numbers* (k_1, λ_1) *and* (k_2, λ_2)*, we say* (k_1, λ_1) *dominates* (k_2, λ_2)*, denoted* $(k_2, \lambda_2) \preceq (k_1, \lambda_1)$*, provided* $k_1 \geq k_2$ *and* $\lambda_1 \geq \lambda_2$.

Clearly, $(FTI(\varphi), \preceq)$ is a partial order.

Definition 9 (Skyline FirmTrussness). *Let* $\varphi \in \mathcal{E}$ *be an edge schema. The skyline FirmTrussness of* φ*, denoted* $\mathsf{SFT}(\varphi)$*, contains the maximal elements of* $FTI(\varphi)$.

Algorithm 5: Index-based Maximal FirmTruss Finding

Input : An ML graph $G = (V, E, L)$, a set of query vertices $Q \subseteq V$, SFT indices, and two integers $k \geq 2$ and $\lambda \geq 1$
Output: A maximal connected (k, λ)-FirmTruss containing Q

```
1  G_0 ← ∅; N ← Q;
2  while N ≠ ∅ do
3      Pick and remove u ∈ N;
4      for each unvisited edge schema φ = (u, v) do
5          Mark φ as visited;
6          for each skyline FirmTruss index (k_i, λ_i) ∈ SFT(φ) do
7              if (k, λ) ⪯ (k_i, λ_i) then
8                  add v and u with all their incident edges into G_0;
9                  N ← N ∪ {v};
10 return G_0;
```

In order to find all possible FirmTrusses, we only need to compute the skyline FirmTrussness for every edge schema in a multilayer graph G. To this end, we present the details of FirmTruss algorithm in Algorithm 4. For a given edge schema φ, if Top$-\lambda(\mathbf{S}_\varphi) = k - 2$, then it cannot be a part of a (k', λ)-FirmTruss, for $k' > k$. Therefore, given λ, we can consider Top$-\lambda(\mathbf{S}_\varphi) + 2$ as an upper bound on the FirmTruss index of φ (line 5). In the FirmTruss decomposition, we recursively pick an edge schema φ with the lowest Top$-\lambda(\mathbf{S}_\varphi)$, assign its FirmTruss index as Top$-\lambda(\mathbf{S}_\varphi) + 2$, and then remove it from the graph. After that, to efficiently update the Top-λ support of its adjacent edges, we use Fact 1 (lines 13-16). At the end of the algorithm, we remove all dominated indices in $\mathsf{SFT}(\varphi)$ for each $\varphi \in \mathcal{E}$ to only store skyline indices (line 23). We can show:

Theorem 10 (FirmTruss Decomposition Complexity). *Algorithm 4 takes $O(\sum_{\ell \in L} |E_\ell|^{1.5} + |E||L|^2)$ time.*

7.2 Index-based Maximal FirmTruss Search

Using Algorithm 4, we can find offline all skyline FirmTruss indices for a given edge schema and query vertex set. Next, we start from the query vertices and by using a breadth-first search, check for each neighbor whether its corresponding edge schema has a skyline FirmTruss index that dominates the input (k, λ). Algorithm 5 shows the procedure. We have:

Theorem 11. *Algorithm 5 takes $O(|E[G_0]|)$ time.*

This indexing approach can be used in Algorithm 1 to find the maximal G_0, as well as in Algorithm 3 so that we only need to add edges whose corresponding edge schema has an index that dominates (k, λ). We refer to these variants of Global and Local as iGlobal and iLocal, respectively.

8 ATTRIBUTED FIRMTRUSS COMMUNITY

Often networks come naturally endowed with attributes associated with their nodes. For example, in DBLP, authors may have areas of interest as attributes. In protein-protein interaction networks, the attributes may correspond to biological processes, molecular functions, or cellular components of a protein made available through the Gene Ontology (GO) project [5]. It is natural to impose some level of similarity between a community's members, based on their attributes.

Network homophily is a phenomenon which states similar nodes are more likely to attach to each other than dissimilar ones. Inspired by this "birds of a feather flock together" phenomenon, in social networks, we argue that users remain engaged with their community if they feel enough similarity with others, while users who feel dissimilar from a community may decide to leave the community. Hence, for each node, we measure how similar it is to the community's members and use it to define the homophily in the community.

We show that surprisingly, use of homophily in a definition of attributed community offers an alternative means to avoid the free-rider effect. In this section, we extend the definition of the FirmTruss-based community to attributed ML networks, where we assume each vertex has an attribute vector. In order to capture vertex similarity, we propose a new function to measure the homophily in a subgraph. We show that this function not only guarantees a high correlation between attributes of vertices in a community but also avoids the free-rider effect. Unlike previous work [20, 43, 82], our model allows for continuous valued attributes. E.g., in a PPI network, the biological process associated with a protein may have a real value, as opposed to just a boolean or a categorical value.

Let $\mathcal{A} = \{A_1, ..., A_d\}$ be a set of attributes. An *attributed multilayer network* $G = (V, E, L, \Psi)$, where (V, E, L) is a multilayer network and $\Psi : V \to \mathbb{R}^d_{\geq 0}$ is a non-negative function that assigns a d-dimensional vector to each vertex, with $\Psi(v)[i]$ representing the strength of attribute A_i in vertex v. Let $h(v, u)$ be a symmetric and non-negative similarity measure based on attribute vectors of u and v. E.g., $h(v, u)$ can be the cosine similarity between $\Psi(u)$ and $\Psi(v)$. Let S be a community containing v. We define $h_S(v)$, capturing the aggregate similarity between v and members of S:

$$h_S(v) = \sum_{\substack{u \in S \\ u \neq v}} h(v, u).$$

The higher the value $h_S(v)$ the more similar user v "feels" they are with the community S. While cosine similarity of attribute vectors is a natural way to compute the similarity $h(v, u)$, any symmetric and non-negative measure can be used in its place.

Based on $h_S(v)$, we define the *homophily score* of community S as follows. Let $p \in \mathbb{R} \cup \{+\infty, -\infty\}$ be any number. Then the homophily score of S is defined as:

$$\Gamma_p(S) = \left(\frac{1}{|S|} \sum_{v \in S} h_S(v)^p \right)^{1/p}.$$

The parameter p gives flexibility for controlling the emphasis on similarity at different ends of the spectrum. When $p \to +\infty$ (resp. $p \to -\infty$) , we have higher emphasis on large (resp. small) similarities. This flexibility allows us to tailor the homophily score to the application at hand.

8.1 Attributed FirmCommunity Model

Problem 2 (Attributed FirmTruss Community Search). *Given an attributed ML network $G = (V, E, L, \Psi)$, two integers $k \geq 2, \lambda \geq 1$, a parameter $p \in \mathbb{R} \cup \{+\infty, -\infty\}$, and a set of query vertices $Q \subseteq V$, the attributed FirmTruss community search (AFTCS) is to find a connected subgraph $G[H] \subseteq G$ satisfying:*

(1) $Q \subseteq H$,
(2) $G[H]$ is a connected (k, λ)-FirmTruss,
(3) $\Gamma_p(H)$ is the maximum among all subgraphs satisfying (1) and (2).

Hardness Analysis. Next we analyze the complexity of the AFTCS problem and show that when p is finite, it is NP-hard.

Theorem 12 (AFTCS Hardness). *The AFTCS problem is NP-hard, whenever p is finite.*

Proof Sketch. Finding the densest subgraph with $\geq k$ vertices in single-layer graphs [49] is a hard problem. Given an instance of this problem, $G = (V, E)$, we construct a complete, attributed ML graph and provide an approach to construct an attribute vector of each node such that $\forall$ vertices u, v, $h(u, v) = \frac{1}{2|V|}$ if $(u, v) \in E$, and $h(u, v) = 0$, if $(u, v) \notin E$. So the densest subgraph with $\geq k$ vertices in G is a solution for AFTCS, and vice versa. □

Free-rider Effect. In the following, we show that the AFTCS model avoids the free-rider effect, analogously to Theorem 5.

Theorem 13 (AFTCS Free-Rider Effect). *For any attributed ML network $G = (V, E, L, \Psi)$ and query vertices $Q \subseteq V$, there is a solution $G[H]$ to the AFTCS problem such that for all query-independent optimal solutions $G[H^*]$, either $H^* = H$, or $G[H \cup H^*]$ is disconnected, or $G[H \cup H^*]$ has a strictly smaller homophily score than $G[H]$.*

8.2 Algorithms

In this section, we propose an efficient approximation algorithm for the AFTCS problem. We show that when $p = +\infty$, or $-\infty$, this algorithm finds the exact solution. We can show that our objective function $\Gamma_p(.)$ is neither submodular nor supermodular (proof in [1], Appendix C), suggesting this problem may be hard to approximate, for some values of p.

Peeling Approximation Algorithm. We divide the problem into two cases: (i) $p > 0$, and (ii) $p < 0$. For finite $p > 0$, $\arg\max \Gamma_p(S) = \arg\max \Gamma_p^p(S)$, so for simplicity, we focus on maximizing $\Gamma_p^p(.)$. Similarly, for finite $p < 0$, we focus on minimizing $\Gamma_p^p(.)$. Note that, for any finite p, an α-approximate solution for optimizing $\Gamma_p^p(.)$ provides an $\alpha^{1/p}$-approximate solution for optimizing $\Gamma_p(.)$.

Consider a set of vertices $S \subseteq V$. Our approximation algorithm is to greedily remove nodes $u \in S$ that may improve the objective. Since removing any node $u \in S$ will change the denominator of $\Gamma_p^p(S)$ in the same way, we can choose the node that leads to the minimum (maximum) drop in the numerator. Let us examine the change to the $\Gamma_p^p(S)$ from dropping $u \in S$:

$$\Gamma_p^p(S\backslash\{u\}) = \frac{\sum_{v \in S\backslash\{u\}} h_{S\backslash\{u\}}(v)^p}{|S| - 1} = \frac{1}{|S| - 1}\left(|S| \cdot \Gamma_p^p(S) - \Delta_u(S)\right),$$

where

$$\Delta_u(S) = h_S(u)^p + \left(\sum_{v \in S\backslash\{u\}} h_S(v)^p - [h_S(v) - h(v, u)]^p\right).$$

Notice that $\Delta_u(S)$ represents the exact decrease in the numerator of $\Gamma_p^p(S)$ resulting from removing u. Based on this observation, in Algorithm 6, we recursively remove a vertex with a minimum (maximum) Δ value, and maintain the remaining subgraph as a (k, λ)-FirmTruss. We have the following result:

Theorem 14 (AFTCS-Approx Complexity). *Algorithm 6 takes $O(d|V_0|^2 + t(|V_0| + |E_0|) + \sum_{\ell \in L} |E_\ell|^{1.5} + |E||L| + |E|\lambda \log |L|)$ time, and $O(|E||L| + |V_0|^2)$ space, where t is the number of iterations, V_0 and E_0 are the vertex set and edge set of maximal (k, λ)-FirmTruss.*

Algorithm 6: AFTCS-Approx

Input : An attributed ML graph $G = (V, E, L, \Psi)$, a set of query vertices $Q \subseteq V$, and two integers $k \geq 2$ and $\lambda \geq 1$
Output : A connected (k, λ)-FT containing Q with a large $\Gamma_p(.)$

1 $G_0 \leftarrow$ Find a maximal connected (k, λ)-FirmTruss containing Q;
2 Calculate $h_{V[G_0]}(u)$ for all $u \in V[G_0]$; $i \leftarrow 0$;
3 **while** $Q \subseteq V[G_i]$ **do**
4 **if** $p > 0$ **then**
5 $u \leftarrow \arg\min_{u \in V[G_i]} \Delta_u(V[G_i])$;
6 **else**
7 $u \leftarrow \arg\max_{u \in V[G_i]} \Delta_u(V[G_i])$;
8 Delete vertex u and its incident edges from G_i in all layers;
9 Maintain G_i as (k, λ)-FirmTruss by removing vertices/edges;
10 Let the remaining graph as G_{i+1}; $i \leftarrow i + 1$;
11 **return** $\arg\max_{H \in \{G_0, \ldots, G_{i-1}\}} \Gamma_p(H)$;

As for the approximation quality, we can show the following when $p \geq 1$. The detailed proof and tightness example can be found in [1], Appendix A.2 and B.

Theorem 15 (AFTCS-Approx Quality). *Let $p \geq 1$, Algorithm 6 returns a $(p + 1)^{1/p}$-approximation solution of AFTCS problem.*

Proof Sketch. Let H^* be the optimal solution. Since removing a node $u^* \in H^*$ will produce a subgraph with homophily score at most $\Gamma_p^p(H^*)$, we have $\Gamma_p^p(H^*) \leq \Delta_{u^*}(H^*)$. Next, we show that the first removed node $u^* \in H^*$ by the algorithm cannot be removed by maintaining FirmTruss, so it was a node with a minimum Δ. Then, we use the fact that the minimum value of Δ is less than the average of Δ over all nodes and provide an upper bound of $(p+1)\Gamma_p^p(S)$ for the average of Δ over S. Finally, we show that function $|S|\Gamma_p^p(S)$ is supermodular for $p \geq 1$, and based on its increasing differences property, we conclude the approximation guarantee. □

Remark 1. *(1) How much good can Algorithm 6 work? As p increases, Algorithm 6 has a better approximation factor. In the worst case, ($p = 1$), we get approximation factor = 2, and when $p \to \infty$, our approximation factor has a limit of 1. This limit of the approximation factor intuitively matches the fact that when $p = +\infty$ the optimal solution is trivial to obtain by the maximal FirmTruss. (2) Although our method provides no formal guarantees when $p < 1$, it achieves results of good empirical quality in practice, as validated in our experiments.*

Exact Algorithm when $p = +\infty$, or $-\infty$. The case $p = +\infty$ is straightforward, where we just want to maximize $\Gamma_{+\infty}(S) = \max_{v \in S} h_S(v)$. The solution of this case is the maximal subgraph that satisfies the conditions (1) and (2) in Problem 2. In the $p = -\infty$ case, we want to maximize $\Gamma_{-\infty}(S) = \min_{v \in S} h_S(v)$. We can recursively remove a vertex with minimum value of h_S and maintain the remaining subgraph such that satisfies conditions (1) and (2) in Problem 2. The pseudocode is identical to Algorithm 6, except in lines 5-8, we recursively remove a vertex with a minimum value of h_S. We refer to this modified peeling algorithm as Exact-MaxMin.

Theorem 16 (Correctness of Exact-MaxMin). *Exact-MaxMin returns the exact solution to the AFTCS problem with $p = -\infty$.*

Table 1: Network Statistics

Dataset	$\|V\|$	$\|E\|$	$\|L\|$	Size	#FT	Attribute	GT
Terrorist	79	2.2K	14	17 KB	48	✓	✓
RM	91	14K	10	112 KB	113	✓	✓
FAO	214	319K	364	3 MB	2397		
Brain	190	934K	520	10 MB	1493		✓
DBLP	513K	1.0M	10	16 MB	66	✓	✓
Obama	2.2M	3.8M	3	60 MB	20		
YouTube	15K	5.6M	4	106 MB	372	✓	✓
Amazon	410K	8.1M	4	123 MB	23		✓
YEAST	4.5K	8.5M	4	97 MB	542		
Higgs	456K	13M	4	205 MB	94		
Friendfeed	510K	18M	3	291 MB	320		
StackOverflow	2.6M	47.9M	24	825 MB	1098		
Google+	28.9M	1.19B	4	20 GB	-		

Size: graph size #FT: number of FirmTrusses GT: ground truth

9 EXPERIMENTS

We conduct experiments to evaluate the proposed CS models and algorithms. Additional experiments on efficiency and parameter sensitivity can be found in [1], Appendix F.

Setup. All algorithms are implemented in Python and compiled by Cython. The experiments are performed on a Linux machine with Intel Xeon 2.6 GHz CPU and 128 GB RAM.

Baseline Methods. We compare our FTCS with the state-of-the-art community search methods in ML networks. ML **k**-core [30] uses an objective function to automatically choose a subset of layers and then finds a subgraph such that the minimum of per-layer minimum degrees, across selected layers, is maximized. ML-LCD [45] uses a greedy strategy to maximize the ratio of Jaccard similarity between nodes inside and outside of the local community. RWM [60] sends random walkers in each layer to obtain the local proximity w.r.t. the query nodes and returns the set of nodes with the smallest conductance. We also implemented a baseline based on TrussCube [39], which finds a maximal connected TrussCube containing query nodes. We also compare our approach with CTC [44], which finds the closest truss community in single-layer graphs, and VAC [58], an attributed variant of CTC, also on single-layer graphs. For baselines, we tune the parameters according to their original papers and report the best results.

Datasets. We perform extensive experiments on thirteen real networks [2, 10, 16–18, 29, 32, 50, 52, 54, 61, 64] covering social, genetic, co-authorship, financial, brain, and co-purchasing networks, whose main characteristics are summarized in Table 1. While Terrorist and DBLP datasets naturally have attributes, for RM and YouTube, we chose one of the layers, embedded it using node2vec [34], and used the vector representation of each node as its attribute vector.

Queries and Evaluation Metrics. We evaluate the performance of all algorithms using different queries by varying the number of query nodes, and the parameters k, λ, and p. To evaluate the quality of found communities C, we measure their F1-score to grade their alignment with the ground truth $\tilde{C}$. Here, $F1(C,\tilde{C}) = \frac{2pre(C,\tilde{C})rec(C,\tilde{C})}{pre(C,\tilde{C})+rec(C,\tilde{C})}$, where $pre(C,\tilde{C}) = \frac{|C\cap\tilde{C}|}{|C|}$ and $rec(C,\tilde{C}) = \frac{|C\cap\tilde{C}|}{|\tilde{C}|}$. To evaluate the efficiency, we report the running time. In reporting results, we cap the running time at 5 hours and memory footprint at 100 GB. For index-based methods, we cap the construction time at 24 hours. Unless stated otherwise, we run our algorithms over 100 random query sets with a random size between 1 and 10, and report the average results. We randomly set k and λ to one of the common skyline indices of edge schemas incident to query nodes.

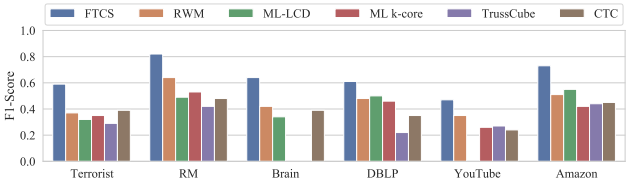


Figure 2: Quality evaluation on ground-truth networks.

Table 2: Evaluation of FTCS with the state-of-the-art methods on datasets without ground truth.

CS Model	FAO		Obama		YEAST		Higgs	
	Density	Diameter	Density	Diameter	Density	Diameter	Density	Diameter
FTCS	**979.71**	**1**	**9.81**	**1.84**	**177.27**	**1.52**	**65.14**	**1.93**
ML **k**-core	-	-	8.13	∞	159.94	∞	59.41	∞
ML-LCD	952.88	1.09	4.87	2.46	-	-	-	-
RWM	911.94	1.12	4.62	3.07	25.45	1.84	24.99	3.16
TrussCube	-	-	4.71	2.03	147.33	1.87	26.89	2.14
CTC	733.85	**1**	5.35	1.99	139.03	1.92	35.18	2.05

Table 3: Evaluation of AFTCS with the state-of-the-art methods on attributed datasets with ground-truth.

CS Model		Terrorist		RM		DBLP		Youtube	
		F1	Density	F1	Density	F1	Density	F1	Density
AFTCS	$p=+\infty$	0.52	**15.29**	0.77	62.35	0.62	8.29	0.45	**11.64**
	$p=2$	0.52	**15.29**	0.79	61.24	0.61	8.22	0.45	**11.64**
	$p=1$	**0.61**	15.22	0.83	**64.31**	0.60	7.91	0.45	11.59
	$p=0$	**0.61**	15.18	0.82	63.98	**0.64**	8.11	0.43	10.88
	$p=-1$	0.59	13.76	0.81	63.19	0.61	8.19	0.44	11.24
	$p=-2$	0.56	13.94	0.81	63.19	0.60	8.03	0.46	11.49
	$p=-\infty$	0.57	14.08	**0.85**	62.46	0.62	7.97	0.46	11.49
FTCS		0.59	10.23	0.84	60.52	0.61	**8.69**	**0.47**	10.36
ML **k**-core		0.35	8.43	0.53	55.98	0.46	5.53	0.26	8.78
ML-LCD		0.32	7.82	0.49	47.26	0.50	6.49	-	-
RWM		0.37	5.45	0.65	39.81	0.48	5.12	0.35	7.46
VAC		0.41	7.51	0.48	52.50	0.35	5.27	0.24	4.34

Quality. We evaluate the effectiveness of different community search models over multilayer networks. Figure 2 reports the average F1-scores of all methods on datasets with the ground-truth community. We observe that our approach achieves the highest F1-score on all networks against baselines. The reason is two-fold. First, in our problem definition, we enforce the minimum-diameter restriction, effectively removing the irrelevant vertices from the result. Second, FirmTruss requires each edge schema to have enough support in a sufficient number of layers, ensuring that the found subgraphs are cohesive and densely connected. While CTC also minimizes the diameter, it is a single-layer approach and misses some structure due to ignoring the type of connections.

We also evaluate all algorithms in terms of other goodness metrics – density ($\beta = 1$), and diameter. Table 2 reports the results on FAO, Obama, YEAST, and Higgs datasets. The results on other datasets are similar, and are omitted for lack of space. We observe that our approach achieves the highest density, and lowest diameter on all networks against baselines.

Since there are no prior models for attributed community search in ML networks, we compare the quality of AFTCS with our ML unattributed baselines as well as VAC. Table 3 reports the F1-score and density of communities found, over four datasets with ground-truth communities. AFTCS consistently beats the baselines. Notice

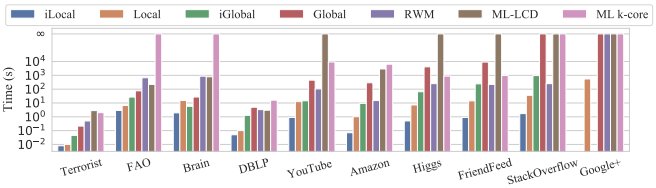
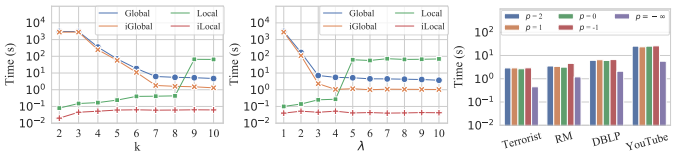


Figure 3: Efficiency Evaluation.



(a) Effect of k (DBLP) **(b) Effect of λ (DBLP)** **(c) Effect of p**

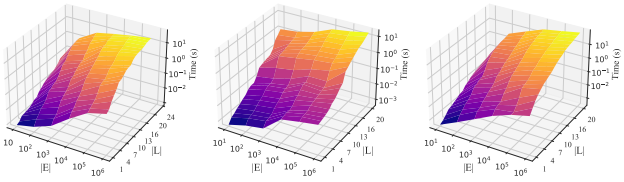
Figure 4: Parameter Sensitivity Evaluation.

that AFTCS has a higher F1-score than FTCS in all but one case, as the existence of both attributes and structure is richer information than only structure. Accordingly, AFTCS is better able to distinguish members from non-members of a ground-truth community.

Efficiency. We evaluate the efficiency of different community search models on multilayer graphs. Figure 3 shows the query processing time of all methods. All of our methods terminate within 1 hour, except Global on the two largest datasets, as it generates a large candidate graph G_0. Our algorithms Local and iLocal run much faster than Online-Global. Overall, iLocal achieves the best efficiency, and it can deal with a search query within a second on most datasets. Local is the only algorithm that scales to graphs containing billions of edges. Bars for iGlobal and iLocal are missing for Google+, as index construction time exceeds our threshold.

Parameter Sensitivity. We evaluate the sensitivity of algorithm efficiency to the parameters k, λ, and p, varying one parameter at a time. Figures 4(a) and (b) show the running time as a function of k and λ on DBLP. The larger k and λ for Global and iGlobal result in lower running time since the algorithms generate a smaller G_0. However, the larger k and λ increase the running time of Local and iLocal since they need to count more nodes in the neighborhood of query nodes to find a (k, λ)-FirmTruss. This also is the reason for the sharp increase of time in both plots. With large k and λ, Local and iLocal need to count nodes farther away, and there is a significant increase in the number of nodes that they need to explore. Figure 4(c) shows the running time as a function of p. We observe that AFTCS-Approx achieves a stable efficiency on different finite values of p. Notice, when $p = -\infty$, this algorithm takes less time as it does not need to calculate $\Delta_u(S)$ for each node and can simply remove the vertex with minimum $h_S(u)$ in each iteration.

Scalability. We test our algorithms using different versions of StackOverflow obtained by selecting a variable #layers from 1 to 24 and also with different subsets of edges. Figure 5 shows the results of the index-based Global, Local Search, and AFTCS-Approx algorithms. The results Global and iLocal are similar, and are omitted for lack of space (see Appendix F [1]). The running time of all approaches scales linearly in #layers. By varying #edges, all algorithms scale gracefully. As expected, the Local algorithm is less sensitive to varying #edges than #layers.



(a) iGlobal **(b) Local Search** **(c) AFTCS-Approx**

Figure 5: Scalability of proposed algorithms with varying the number of layers and the number of edges.

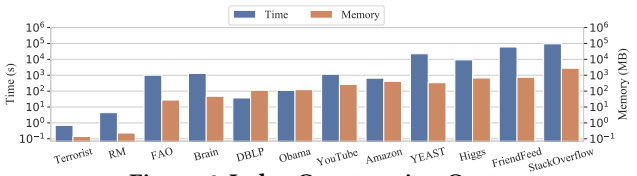
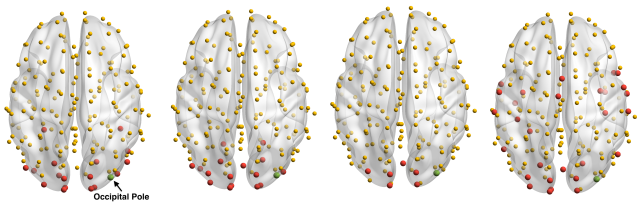


Figure 6: Index Construction Costs.

Index Construction. Figure 6 reports the SFT index construction time and size. The size of indices is more dependent on the structure of a graph than its size. That is, since we store the SFT indices for each edge schema, the size of indices depends on the number of FirmTrusses in the network. For all datasets, the SFT index can be built within 24 hours, and its size is within 2.6× of the original graph size. The result shows the efficiency of SFT index construction.

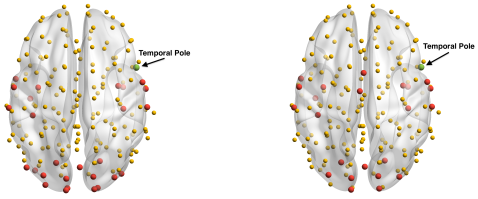
Case Studies: Identify Functional Systems in Brain Networks. Detecting and monitoring functional systems in the human brain is a primary task in neuroscience. However, the brain network generated from an individual can be noisy and incomplete. Using brain networks from many individuals can help to identify functional systems more accurately. A community in a multilayer brain network, where each layer is the brain network of an individual, can be interpreted as a functional system in the brain. In this case study, to show the effectiveness of the FTCS, we compare its detected functional system with ground truth. Here, we focus on the "visual processing" task in the brain. As the "Occipital Pole" is primarily responsible for visual processing [48], we use one of its representing nodes as the query node. Figure 7 reports the found communities by FTCS and baselines. The identified communities are highlighted in red, and the query node is green. Results show the effectiveness of FTCS as the community detected by our method is very similar to the ground truth with F1-score of 0.75. RWM, which is a random walk-based community model, includes many false-positive nodes that cause F1-score of 0.495. On the other hand, some nodes in the boundary region are missed by ML-LCD that caused low F1-score of 0.4. The result of the ML **k**-core is omitted as it does not terminate even before one week. This is also a real-world illustrative example that shows the advantages of FirmTruss over existing models.

Case Studies: Classification on Brain Networks. Behavioral disturbances in attention deficit hyperactivity disorder (ADHD) are considered to be caused by the dysfunction of spatially distributed, interconnected neural systems [33]. In this section, we employ our FTCS to detect common structures in the brain functional connectivity network of ADHD individuals and typically developed (TD) people. Our dataset is derived from the functional magnetic resonance imaging (fMRI) of 520 individuals with the



(a) Ground truth (b) FirmTruss (c) ML-LCD (d) RWM

Figure 7: Detected functional systems.



(a) C_{ADHD} (b) C_{TD}

Figure 8: FirmTruss community in TD and ADHD groups.

Table 4: Results of the ADHD classification task.

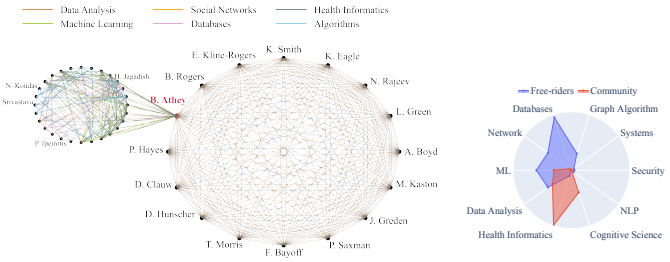
CS Model	Accuracy	Precision	Recall	F1-score
FTCS	**76.56 ± 0.72**	**75.73 ± 1.00**	**83.77 ± 1.21**	**77.54 ± 0.66**
ML-LCD	55.70 ± 1.25	55.43 ± 1.15	78.91 ± 1.64	64.13 ± 1.06
RWM	50.47 ± 0.18	53.03 ± 2.08	55.09 ± 0.41	45.59 ± 1.18

same methodology used in [53]. It contains 190 individuals in the condition group, labeled ADHD, and 330 individuals in the control group, labeled TD. Here, each layer is the brain network of an individual person, where nodes are brain regions, and each edge measures the statistical association between the functionality of its endpoints. Since "Temporal Pole" is known as the part of the brain that plays an important role in ADHD symptoms [65, 70], we use a subset of its representing nodes as the query nodes.

Next, we randomly chose 230 individuals labeled TD and 90 individuals labeled ADHD to construct two multilayer brain networks and then found the FirmTruss communities associated with "Temporal Pole" in each group separately, referred to as C_{TD} and C_{ADHD} in Figure 8. In the second step, for each individual unseen brain network, we find the associated communities to the query nodes using the FTCS model, setting $|L| = 1$. In order to classify an unseen brain network, we calculate the similarity of its found communities with C_{TD} and C_{ADHD} and then predict its label as the label of the community with maximum similarity. Here, we use the overlap coefficient [73] as the similarity measure between two communities.

To ensure that the result is statistically significant, we repeat this process for 1000 trials and report the mean, and its relative standard deviation of accuracy, precision, recall and F1-score in Table 4. Not only does our FTCS outperform baseline community search models, but it also achieves results comparable with the state-of-the-art ADHD classification model [33], based on SVM, which reports an accuracy of 76%. This comparable result is achieved by the FTCS method, which is a white-box and explainable model.

Case Studies: DBLP. We conduct a case study on the DBLP dataset to judge the quality of the AFTCS model and to show the effectiveness of the homophily score in removing free riders. The multilayer



(a) AFTCS Community (b) Average attribute

Figure 9: Case study of DBLP.

DBLP dataset is a collaboration network derived following the methodology in [9]. In this dataset, each node is a researcher, an edge shows collaboration, and each layer is a topic of research. For each author, we consider the bag of words drawn from the titles of all their papers and apply LDA topic modeling [8] to automatically identify 240 topics. The attribute of each author is the vector that describes the distribution of their papers in these 240 topics. We use "Brian D. Athey" as the query node. The maximal $(8, 2)$-FirmTruss, including the query node, has 44 nodes with a minimum homophily score of 0.08, shown in Figure 9(a). The community found by AFTCS ($p = -\infty$) is an $(8, 2)$-FirmTruss with a minimum homophily score of 0.28, which resulted from removing 28 nodes as free-riders. The found community is shown in the larger circle, while the smaller circle shows free riders. We compute the average attributes of community members and free-riders and then cluster their non-zero elements into ten known research topics. Results are shown in Figure 9(b). While researchers in the found community have focused more on "Health Informatics," removed researchers (free-riders) have focused more on "Databases." The connection between these two communities, which results in their union being an $(8, 2)$-FirmTruss, is the collaboration of "Brian D. Athey," from the "Health Informatics" community with some researchers in "Databases" community. AFTCS divides the maximal FirmTruss into two communities with more correlations inside each of them.

10 CONCLUSIONS

We propose and study a novel extended notion of truss decomposition in ML networks, FirmTruss, and establish its nice properties. We then study a new problem of FirmTruss-based community search over ML graphs. We show that the problem is NP-hard. To tackle it efficiently, we propose two 2-approximation algorithms and prove that our approximations are tight. To further improve their efficiency, we propose an index and develop fast index-based variants of our approximation algorithms. We extend the FirmTruss-based community model to attributed ML networks and propose a homophily-based model making use of generalized p-mean. We prove that this problem is also NP-hard for finite value of p and to solve it efficiently, we develop a fast greedy algorithm which has a quality guarantee for $p \geq 1$. Our extensive experimental results on large real-world networks with ground-truth communities confirm the effectiveness and efficiency of our proposed models and algorithms, while our case studies on brain networks and DBLP illustrate their practical utility.

REFERENCES

[1] [n.d.]. https://www.dropbox.com/s/r656czazbl8uxk5/Appendix_firmtruss.pdf?dl=0.

[2] Carlo Abrate and Francesco Bonchi. 2021. Counterfactual Graphs for Explainable Classification of Brain Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery; Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2495–2504. https://doi.org/10.1145/3447548.3467154

[3] Esra Akbas and Peixiang Zhao. 2017. Truss-Based Community Search: A Truss-Equivalence Based Indexing Approach. *Proc. VLDB Endow.* 10, 11 (aug 2017), 1298–1309. https://doi.org/10.14778/3137628.3137640

[4] Alberto Aleta and Yamir Moreno. 2019. Multilayer networks in a nutshell. *Annual Review of Condensed Matter Physics* 10 (2019), 45–62.

[5] M. Ashburner, C.A. Ball, and J.A. Blake et al. 2000. Gene Ontology: Tool for the Unification of Biology. *Nature Genetics* 25, 1 (2000), 25–29.

[6] N. Azimi-Tafreshi, J. Gomez-Garde, and S. N. Dorogovtsev. 2014. k-corepercolation on multiplex networks. *Physical Review E* 90, 3 (Sep 2014).

[7] Bharat B Biswal, Maarten Mennes, Xi-Nian Zuo, Suril Gohel, Clare Kelly, Steve M Smith, Christian F Beckmann, Jonathan S Adelstein, Randy L Buckner, Stan Colcombe, et al. 2010. Toward discovery science of human brain function. *Proceedings of the National Academy of Sciences* 107, 10 (2010), 4734–4739.

[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (mar 2003), 993–1022.

[9] Francesco Bonchi, Aristides Gionis, Francesco Gullo, Charalampos E. Tsourakakis, and Antti Ukkonen. 2015. Chromatic Correlation Clustering. *ACM Trans. Knowl. Discov. Data* 9, 4, Article 34 (jun 2015), 24 pages. https://doi.org/10.1145/2728170

[10] Fabio Celli, F Marta L Di Lascio, Matteo Magnani, Barbara Pacelli, and Luca Rossi. 2010. Social Network Data and Practices: the case of Friendfeed. In *International Conference on Social Computing, Behavioral Modeling and Prediction (Lecture Notes in Computer Science)*. Springer Berlin Heidelberg, New York, NY, USA, –.

[11] Tanmoy Chakraborty, Sikhar Patranabis, Pawan Goyal, and Animesh Mukherjee. 2015. On the Formation of Circles in Co-Authorship Networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 109–118. https://doi.org/10.1145/2783258.2783292

[12] Lu Chen, Chengfei Liu, Kewen Liao, Jianxin Li, and Rui Zhou. 2019. Contextual Community Search Over Large Social Networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, Macau SAR, China, 88–99. https://doi.org/10.1109/ICDE.2019.00017

[13] Lu Chen, Chengfei Liu, Rui Zhou, Jianxin Li, Xiaochun Yang, and Bin Wang. 2018. Maximum Co-Located Community Search in Large Scale Social Networks. *Proc. VLDB Endow.* 11, 10 (2018), 1233–1246. https://doi.org/10.14778/3231751.3231755

[14] Shuntong Chen, Huaien Qian, Yanping Wu, Chen Chen, and Xiaoyang Wang. 2019. Efficient Adoption Maximization in Multi-layer Social Networks. In *2019 International Conference on Data Mining Workshops (ICDMW)*. IEEE, Beijing, China, 56–60. https://doi.org/10.1109/ICDMW.2019.00017

[15] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. Association for Computing Machinery, New York, NY, USA, 277–288.

[16] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi. 2013. The Anatomy of a Scientific Rumor. *Scientific Reports* 3, 1 (2013), –.

[17] M. De Domenico, M. A. Porter, and A. Arenas. 2014. MuxViz: a tool for multilayer analysis and visualization of networks. *Journal of Complex Networks* 3, 2 (Oct 2014), 159–176. https://doi.org/10.1093/comnet/cnu038

[18] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora. 2015. Structural reducibility of multilayer networks. *Nature communications* 6 (2015), 6864.

[19] Zheng Dong, Xin Huang, Guorui Yuan, Hengshu Zhu, and Hui Xiong. 2021. Butterfly-Core Community Search over Labeled Graphs. *Proc. VLDB Endow.* 14, 11 (jul 2021), 2006–2018. https://doi.org/10.14778/3476249.3476258

[20] Yixiang Fang, Reynold Cheng, Yankai Chen, Siqiang Luo, and Jiafeng Hu. 2017. Effective and efficient attributed community search. *The VLDB Journal* 26, 6 (2017), 803–828.

[21] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective Community Search for Large Attributed Graphs. *Proc. VLDB Endow.* 9, 12 (aug 2016), 1233–1244. https://doi.org/10.14778/2994509.2994538

[22] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2019. A Survey of Community Search Over Big Graphs. https://doi.org/10.48550/ARXIV.1904.12539

[23] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2019. Effective and Efficient Community Search Over Large Directed Graphs. *IEEE Transactions on Knowledge and Data Engineering* 31, 11 (2019), 2093–2107.

[24] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2019. Effective and Efficient Community Search Over Large Directed Graphs (Extended Abstract). In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, Macau SAR, China, 2157–2158. https://doi.org/10.1109/ICDE.2019.00273

[25] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (Feb. 2020), 854–867. https://doi.org/10.14778/3380750.3380756

[26] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[27] Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007), 36–41. https://doi.org/10.1073/pnas.0605965104 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.0605965104

[28] Amita Gajewar and Atish Das Sarma. 2012. Multi-skill Collaborative Teams based on Densest Subgraphs. In *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics, California, USA, 165–176. https://doi.org/10.1137/1.9781611972825.15

[29] Edoardo Galimberti, Francesco Bonchi, and Francesco Gullo. 2017. Core Decomposition and Densest Subgraph in Multilayer Networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 1807–1816. https://doi.org/10.1145/3132847.3132993

[30] Edoardo Galimberti, Francesco Bonchi, Francesco Gullo, and Tommaso Lanciano. 2020. Core Decomposition in Multilayer Networks: Theory, Algorithms, and Applications. *ACM Trans. Knowl. Discov. Data* 14, 1, Article 11 (2020), 40 pages. https://doi.org/10.1145/3369872

[31] Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. 2021. ICS-GNN: Lightweight Interactive Community Search via Graph Neural Network. *Proc. VLDB Endow.* 14, 6 (feb 2021), 1006–1018. https://doi.org/10.14778/3447689.3447704

[32] Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, and Dawn Song. 2012. Evolution of Social-Attribute Networks: Measurements, Modeling, and Implications Using Google+. In *Proceedings of the 2012 Internet Measurement Conference* (Boston, Massachusetts, USA) *(IMC '12)*. Association for Computing Machinery, New York, NY, USA, 131–144.

[33] Kristi R. Griffiths, Taylor A. Braund, Michael R. Kohn, Simon Clarke, Leanne M. Williams, and Mayuresh S. Korgaonkar. 2021. Structural brain network topology underpinning ADHD and response to methylphenidate treatment. *Translational Psychiatry* 11, 1 (02 Mar 2021), 150. https://doi.org/10.1038/s41398-021-01278-x

[34] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. arXiv:1607.00653 [cs.SI]

[35] Fangda Guo, Ye Yuan, Guoren Wang, Xiangguo Zhao, and Hao Sun. 2021. Multi-attributed Community Search in Road-social Networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 109–120. https://doi.org/10.1109/ICDE51399.2021.00017

[36] Farnoosh Hashemi, Ali Behrouz, and Laks V.S. Lakshmanan. 2022. FirmCore Decomposition of Multilayer Networks. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 1589–1600. https://doi.org/10.1145/3485447.3512205

[37] Allen L. Hu and Keith C. C. Chan. 2013. Utilizing Both Topological and Attribute Information for Protein Complex Identification in PPI Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 10, 3 (2013), 780–792. https://doi.org/10.1109/TCBB.2013.37

[38] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2016. Querying Minimal Steiner Maximum-Connected Subgraphs in Large Graphs. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 1241–1250. https://doi.org/10.1145/2983323.2983748

[39] Hongxuan Huang, Qingyuan Linghu, Fan Zhang, Dian Ouyang, and Shiyu Yang. 2021. Truss Decomposition on Multilayer Graphs. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, virtual event, 5912–5915. https://doi.org/10.1109/BigData52589.2021.9671831

[40] Ling Huang, Chang-Dong Wang, and Hong-Yang Chao. 2019. Higher-Order Multi-Layer Community Detection. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence* (Honolulu, Hawaii, USA) *(AAAI'19/IAAI'19/EAAI'19)*. AAAI Press, -, Article 1271, 2 pages. https://doi.org/10.1609/aaai.v33i01.33019945

[41] Xinyu Huang, Dongming Chen, Tao Ren, and Dongqi Wang. 2021. A survey of community detection methods in multilayer networks. *Data Mining and Knowledge Discovery* 35, 1 (2021), 1–45.

[42] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying K-Truss Community in Large and Dynamic Graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (Snowbird, Utah, USA) *(SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 1311–1322. https://doi.org/10.1145/2588555.2610495

[43] Xin Huang and Laks VS Lakshmanan. 2017. Attribute-driven community search. *Proceedings of the VLDB Endowment* 10, 9 (2017), 949–960.

[44] Xin Huang, Laks V. S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate Closest Community Search in Networks. *Proc. VLDB Endow.* 9, 4 (dec 2015), 276–287. https://doi.org/10.14778/2856318.2856323

[45] Roberto Interdonato, Andrea Tagarelli, Dino Ienco, Arnaud Sallaberry, and Pascal Poncelet. 2017. Local community detection in multilayer networks. *Data Mining and Knowledge Discovery* 31, 5 (2017), 1444–1479.

[46] V. Jethava and N. Beerenwinkel. 2015. Finding Dense Subgraphs in Relational Graphs. In *Machine Learning and Knowledge Discovery in Databases*, Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, João Gama, Alípio Jorge, and Carlos Soares (Eds.). Springer International Publishing, Cham, 641–654.

[47] Yuli Jiang, Yu Rong, Hong Cheng, Xin Huang, Kangfei Zhao, and Junzhou Huang. 2021. Query Driven-Graph Neural Networks for Community Search: From Non-Attributed, Attributed, to Interactive Attributed. https://doi.org/10.48550/ARXIV.2104.03583

[48] Eileanoir B. Johnson, Elin M. Rees, Izelle Labuschagne, Alexandra Durr, Blair R. Leavitt, Raymund A.C. Roos, Ralf Reilmann, Hans Johnson, Nicola Z. Hobbs, Douglas R. Langbehn, Julie C. Stout, Sarah J. Tabrizi, and Rachael I. Scahill. 2015. The impact of occipital lobe cortical thickness on cognitive task performance: An investigation in Huntington's Disease. *Neuropsychologia* 79 (2015), 138–146. https://doi.org/10.1016/j.neuropsychologia.2015.10.033

[49] Samir Khuller and Barna Saha. 2009. On Finding Dense Subgraphs. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP '09)*. Springer-Verlag, -, 597–608.

[50] Jungeun Kim and Jae-Gil Lee. 2015. Community Detection in Multi-Layer Graphs: A Survey. *SIGMOD Rec.* 44, 3 (dec 2015), 37–48. https://doi.org/10.1145/2854006.2854013

[51] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. 2014. Multilayer networks. *Journal of Complex Networks* 2, 3 (07 2014), 203–271. https://doi.org/10.1093/comnet/cnu016

[52] Jérôme Kunegis. 2013. KONECT: The Koblenz Network Collection. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13 Companion)*. Association for Computing Machinery, New York, NY, USA, 1343–1350. https://doi.org/10.1145/2487788.2488173

[53] Tommaso Lanciano, Francesco Bonchi, and Aristides Gionis. 2020. Explainable Classification of Brain Networks via Contrast Subgraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3308–3318. https://doi.org/10.1145/3394486.3403383

[54] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. 2007. The Dynamics of Viral Marketing. *ACM Trans. Web* 1, 1 (May 2007), 5–es.

[55] Rong-Hua Li, Lu Qin, Fanghua Ye, Jeffrey Xu Yu, Xiaokui Xiao, Nong Xiao, and Zibin Zheng. 2018. Skyline Community Search in Multi-Valued Networks. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 457–472. https://doi.org/10.1145/3183713.3183736

[56] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. 2018. Persistent community search in temporal networks. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, Paris, France, 797–808.

[57] Boge Liu, Fan Zhang, Chen Zhang, Wenjie Zhang, and Xuemin Lin. 2019. CoreCube: Core Decomposition in Multilayer Graphs. In *Web Information Systems Engineering – WISE 2019*, Reynold Cheng, Nikos Mamoulis, Yizhou Sun, and Xin Huang (Eds.). Springer International Publishing, Cham, 694–710.

[58] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, Dallas, Texas, USA, 937–948. https://doi.org/10.1109/ICDE48307.2020.00086

[59] Xueming Liu, Enrico Maiorino, Arda Halu, Kimberly Glass, Rashmi B. Prasad, and Joseph Loscalzo. 2020. Robustness and lethality in multilayer biological molecular networks. *Nature Communications* 11 (2020), –.

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

[61] Elisa Omodei, Manlio De Domenico, and Alex Arenas. 2015. Characterizing interactions in online social networks during exceptional events. *Frontiers in Physics* 3 (2015), 59. https://doi.org/10.3389/fphy.2015.00059

[62] Kaiyan Peng, Zheng Lu, Vanessa Lin, Michael R. Lindstrom, Christian Parkinson, Chuntian Wang, Andrea L. Bertozzi, and Mason A. Porter. 2021. A Multilayer Network Model of the Coevolution of the Spread of a Disease and Competing Opinions. arXiv:2107.01713 [cs.SI]

[63] Jonathan D. Power, Alexander L. Cohen, Steven M. Nelson, Gagan S. Wig, Kelly Anne Barnes, Jessica A. Church, Alecia C. Vogel, Timothy O. Laumann, Fran M. Miezin, Bradley L. Schlaggar, and Steven E. Petersen. 2011. Functional network organization of the human brain. *Neuron* 72, 4 (17 Nov 2011), 665–678. https://doi.org/10.1016/j.neuron.2011.09.006 S0896-6273(11)00792-6[PII].

[64] N. Roberts and S. Everton. 2011. The Noordin Top Terrorist Network Data. (2011). http://www.thearda.com/Archive/Files/Descriptions/

[65] Jacqueline F. Saad, Kristi R. Griffiths, Michael R. Kohn, Simon Clarke, Leanne M. Williams, and Mayuresh S. Korgaonkar. 2017. Regional brain network organization distinguishes the combined and inattentive subtypes of Attention Deficit Hyperactivity Disorder. *NeuroImage: Clinical* 15 (2017), 383–390. https://doi.org/10.1016/j.nicl.2017.05.016

[66] Pramod Shinde and Sarika Jalan. 2015. A multilayer protein-protein interaction network analysis of different life stages in Caenorhabditis elegans. *EPL (Europhysics Letters)* 112, 5 (dec 2015), 58001. https://doi.org/10.1209/0295-5075/112/58001

[67] Mauro Sozio and Aristides Gionis. 2010. The Community-Search Problem and How to Plan a Successful Cocktail Party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington, DC, USA). Association for Computing Machinery, New York, NY, USA, 939–948.

[68] Mauro Sozio and Aristides Gionis. 2010. The Community-Search Problem and How to Plan a Successful Cocktail Party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington, DC, USA) *(KDD '10)*. Association for Computing Machinery, New York, NY, USA, 939–948. https://doi.org/10.1145/1835804.1835923

[69] Yizhou Sun and Jiawei Han. 2013. Mining Heterogeneous Information Networks: A Structural Analysis Approach. *SIGKDD Explor. Newsl.* 14, 2 (apr 2013), 20–28. https://doi.org/10.1145/2481244.2481248

[70] Yunkai Sun, Lei Zhao, Zhihui Lan, Xi-Ze Jia, and Shao-Wei Xue. 2020. Differentiating Boys with ADHD from Those with Typical Development Based on Whole-Brain Functional Connections Using a Machine Learning Approach. *Neuropsychiatric disease and treatment* 16 (10 Mar 2020), 691–702. https://doi.org/10.2147/NDT.S239013 PMC7071874[pmcid].

[71] Andrea Tagarelli, Alessia Amelio, and Francesco Gullo. 2017. Ensemble-Based Community Detection in Multilayer Networks. *Data Min. Knowl. Discov.* 31, 5 (sep 2017), 1506–1543. https://doi.org/10.1007/s10618-017-0528-8

[72] Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. 2012. Structural diversity in social contagion. *Proceedings of the National Academy of Sciences* 109, 16 (2012), 5962–5966. https://doi.org/10.1073/pnas.1116502109 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1116502109

[73] MK Vijaymeena and K Kavitha. 2016. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal* 3, 2 (2016).

[74] Jia Wang and James Cheng. 2012. Truss Decomposition in Massive Networks. *Proc. VLDB Endow.* 5, 9 (2012), 812–823. https://doi.org/10.14778/2311906.2311909

[75] Yue Wang, Xun Jian, Zhenhua Yang, and Jia Li. 2017. Query Optimal k-Plex Based Community in Graphs. (2017). https://doi.org/10.1007/s41019-017-0051-3

[76] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (01 Jun 1998), 440–442. https://doi.org/10.1038/30918

[77] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust Local Community Detection: On Free Rider Effect and Its Elimination. *Proc. VLDB Endow.* 8, 7 (feb 2015), 798–809. https://doi.org/10.14778/2752939.2752948

[78] Jaewon Yang and Jure Leskovec. 2012. Defining and Evaluating Network Communities Based on Ground-Truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics* (Beijing, China) *(MDS '12)*. Association for Computing Machinery, New York, NY, USA, Article 3, 8 pages. https://doi.org/10.1145/2350190.2350193

[79] Yixing Yang, Yixiang Fang, Xuemin Lin, Wenjie Zhang, and Yixiang Fang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 901–912. https://doi.org/10.1109/ICDE48307.2020.00083

[80] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2018. Index-Based Densest Clique Percolation Community Search in Networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 5 (2018), 922–935. https://doi.org/10.1109/TKDE.2017.2783933

[81] Xuemeng Zhai, Wanlei Zhou, Gaolei Fei, Weiyi Liu, Zhoujun Xu, Chengbo Jiao, Cai Lu, and Guangmin Hu. 2018. Null model and community structure in multiplex networks. *Scientific reports* 8, 1 (2018), 1–13.

[82] Zhiwei Zhang, Xin Huang, Jianliang Xu, Byron Choi, and Zechao Shang. 2019. Keyword-centric community search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 422–433.

[83] Zibin Zheng, Fanghua Ye, Rong-Hua Li, Guohui Ling, and Tan Jin. 2017. Finding weighted k-truss communities in large networks. *Information Sciences* 417 (2017), 344–360. https://doi.org/10.1016/j.ins.2017.07.012

[84] Qijun Zhu, Haibo Hu, Cheng Xu, Jianliang Xu, and Wang-Chien Lee. 2017. Geo-Social Group Queries with Minimum Acquaintance Constraint. arXiv:1406.7367 [cs.DB]

[85] Rong Zhu, Zhaonian Zou, and Jianzhong Li. 2018. Diversified Coherent Core Search on Multi-Layer Graphs. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 701–712. https://doi.org/10.1109/ICDE.2018.00069

[86] Yuanyuan Zhu, Qian Zhang, Lu Qin, Lijun Chang, and Jeffrey Xu Yu. 2018. Querying Cohesive Subgraphs by Keywords. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. https://doi.org/10.1109/ICDE.2018.00141

A PROOFS

A.1 Properties

Property 1 (Uniqueness).

Proof. Suppose J_k^λ and $J_k^{'\lambda}$ are two distinct (k,λ)-FirmTrusses of G. By Definition 6, J_k^λ is a maximal subgraph such that $\forall \varphi \in \mathcal{E}[J_k^\lambda]$ there are $\geq \lambda$ layers where the supports of φ in each of these layers is $\geq k-2$. Similarly, $J_k^{'\lambda}$ is a maximal subgraph with the same property. Then the subgraph $J_k^\lambda \cup J_k^{'\lambda}$ trivially satisfies the FirmTruss conditions, contradicting the maximality of J_k^λ and $J_k^{'\lambda}$. □

Property 2 (Hierarchical Structure).

Proof. (a) The property follows from the definition of FirmTruss and this fact that in a subgraph, every edge schema $\varphi \in \mathcal{E}$ that has at least $k-1$ supports, also has at least $k-2$ supports. (b) Similarly, if edge schema φ in at least $\lambda+1$ layers has no less than $k-2$ supports, in at least λ layers it has at least $k-2$ supports as well. □

Property 3 (Minimum Degree).

Proof. Given an edge schema $(v,u) \in \mathcal{E}$, where $u, v \in V$, since (v,u) in at least λ layers has at least $k-2$ supports, u and v has at least $k-2$ common neighbours in at least λ layers. Therefore, each has degree at least $k-1$ in at least λ layers. □

A.2 Theorems and Lemmas

Theorem 1 (Density Lower Bound).

Proof. First, we provide a lower bound for the mean of average degree over all layers: $\frac{\sum_{\ell=1}^{|L|} |E_\ell[J_k^\lambda]|}{|L||J_k^\lambda|}$. We can rewrite the numerator as half of the sum of degrees of each node in each layer. In fact,

$$\sum_{\ell=1}^{|L|} |E_\ell[J_k^\lambda]| = \frac{1}{2}\left(\sum_{\ell=1}^{|L|}\sum_{u\in J_k^\lambda} \deg_\ell^{J_k^\lambda}(u)\right). \tag{1}$$

Since J_k^λ is a FirmTruss, each edge schema $\varphi = (v,u)$ in at least λ layers has at least $k-2$ supports. Accordingly, v and u each in at least λ layers have at least $k-1$ neighbours. Therefore,

$$\sum_{\ell=1}^{|L|}\sum_{u\in J_k^\lambda} \deg_\ell^{J_k^\lambda}(u) \geq \lambda(k-1)|J_k^\lambda|. \tag{2}$$

Based on Equations (1) and (2), we can conclude that the mean of average degree over all layers is at least $\frac{\lambda(k-1)}{2|L|}$. Since it is the average over all layers, there exist a layer $\ell_1 \in L$ such that:

$$\frac{|E_{\ell_1}[J_k^\lambda]|}{|J_k^\lambda|} \geq \frac{\lambda(k-1)}{2|L|}.$$

Now, ignoring this layer, and exploiting the definition of J_k^λ, each edge schema $\varphi = (v,u)$ in at least $\lambda-1$ layers has at least $k-2$ supports. Accordingly, v and u each in at least $\lambda-1$ layers have at least $k-1$ neighbours. Therefore, the mean of average degree over remaining layers, $L \setminus \{\ell_1\}$, is at least $\frac{(\lambda-1)(k-1)}{2|L|}$. Again, we can conclude that there is a layer $\ell_2 \neq \ell_1$ such that $\frac{|E_{\ell_2}[J_k^\lambda]|}{|J_k^\lambda|} \geq \frac{(\lambda-1)(k-1)}{2|L|}$. So there is a subset of L, $\hat{L} = \{\ell_1, \ell_2\}$, with two layers such that each layer has density at least $\frac{(\lambda-1)(k-1)}{2|L|}$. By iterating this process, we can conclude that $\exists \tilde{L} \subseteq L$, such that each layer $\ell \in \tilde{L}$ has density at least $\frac{(\lambda-|\tilde{L}|+1)(k-1)}{2|L|}$. Based on the definition of the density in ML graphs, we can conclude that:

$$\rho_\beta(J_k^\lambda) \geq \frac{(\lambda-|\tilde{L}|+1)(k-1)}{2|L|}|\tilde{L}|^\beta.$$

The right hand side of the above inequality is a function of $|\tilde{L}|$, and since the above inequality is valid for each arbitrary integer $1 \leq |\tilde{L}| \leq \lambda$, $\rho_\beta(J_k^\lambda)$ is not less than the maximum of this function exhibited by these values. □

Theorem 2 (Diameter Upper Bound).

Proof. Let path

$$\mathcal{P} : v_{\ell_1}^1 \to v_{\ell_2}^2 \to \cdots \to v_{\ell_\alpha}^\alpha,$$

be the diameter of $G[J_k^\lambda]$, and for each edge schema $\varphi \in \mathcal{E}[J_k^\lambda]$, we define $L_\varphi = \{\ell'_1, \ldots, \ell'_{\lambda'}\}$ as the set of layers such that in each layer ℓ'_i, $\varphi_{\ell'_i}$ has at least $k-2$ supports. Note that $\lambda' \geq \lambda$. Let $t \in \mathbb{N}$ such that $\frac{t}{t+1}|L| > \lambda \geq \frac{t-1}{t}|L|$, then we show that there is a path

$$\mathcal{P}' : w_{\ell''_1}^1 \to w_{\ell''_2}^2 \to \cdots \to w_{\ell''_{\alpha'}}^{\alpha'},$$

such that its path schema is the same as the path schema of $\mathcal{P}$, and for each $0 \leq r \leq \lfloor \frac{|\mathcal{P}'|}{t} \rfloor - 1$ all $\{w^{rt+1}, w^{rt+2}, \ldots, w^{rt+t}\}$ are in the same layer ℓ_r^* such that:

$$\ell_r^* \in \bigcap_{i=1}^{t-1} L_{(v^{rt+i}, v^{rt+i+1})}.$$

To show that, let $\mathfrak{P}$ be the path schema of $\mathcal{P}$, and $u^1, u^2, \ldots, u^t$ be the t consecutive vertices in the path schema. For each edge schema $\varphi^i = (u^i, u^{i+1})$, we know that there are at least λ layers $\ell \in L_{\varphi^i}$ in which φ_ℓ^i has at least $k-2$ support. Thus, since $\lambda \geq \frac{t-1}{t}|L|$, based on the pigeonhole principle there is a layer ℓ^* such that all φ^i in ℓ^* has at least $k-2$ supports. Therefore, we have constructed path $\mathcal{P}'$ such that for each $0 \leq r \leq \lfloor \frac{|\mathfrak{P}|+1}{t} \rfloor - 1$:

(1) vertices $\{u^{rt+1}, u^{rt+2}, \ldots, u^{rt+t}\}$ of the path are in the same layer ℓ_r^*,
(2) we have an inter-layer edge in the path that $u_{\ell_r^*}^{rt+t} \to u_{\ell_{r+1}^*}^{rt+t}$.

Due to the number of inter-layer edges, which is at most $\frac{|\mathfrak{P}|}{t}$, the length of the $\mathcal{P}'$ is at most $(1+1/t)|\mathfrak{P}|$.

Next we show that the length of $\mathfrak{P}$ is bounded by $\lfloor \frac{2|J_k^\lambda|-2}{k} \rfloor$. Given the part of $\mathfrak{P}$, $\mathfrak{P}_r : u^{rt+1} \to u^{rt+2} \to \cdots \to u^{rt+t}$, that all edges are in layer ℓ_r^*, for any $0 \leq r \leq \lfloor \frac{|\mathfrak{P}|+1}{t} \rfloor$. We know that each edge $(u^{rt+i}, u^{rt+i+1}, \ell_r^*)$ has at least $k-2$ supports. Thus, each pair (u^{rt+i}, u^{rt+i+1}) has at least $k-2$ common neighbours in layer ℓ_r^*. We divide the set of common neighbours into two sets A_i^r and B_i^r: each A_i^r holds those vertices not in $\mathfrak{P}_r$ that are adjacent to both u^{rt+i} and u^{rt+i+1}, and to no other vertices in $\mathcal{P}'_r$; each B_i^r holds those vertices that are adjacent to all three of u^{rt+i-1}, u^{rt+i} and

u^{rt+i+1}. Note that the B_i^r sets are disjoint, else we can find a path with smallest length. Let

$$|N_{\ell_r^*}| = |\mathfrak{P}_r| + 1 + \sum_{i=0}^{|\mathfrak{P}_r|-1} |A_i^r| + \sum_{i=1}^{|\mathfrak{P}_r|-1} |B_i^r|.$$

For each i, $|A_i^r| = k-2-|B_i^r|$ and also $|B_i^r| + |B_{i+1}^r| \leq k-2$. Thus:

$$|N_{\ell_r^*}| \geq 1 + |\mathfrak{P}_r|(k-1) - \lfloor \frac{|\mathfrak{P}_r|}{2} \rfloor (k-2)$$

$$\Rightarrow \sum_r |N_{\ell_r^*}| \geq \lfloor \frac{|\mathfrak{P}|+1}{t} \rfloor + (k-1) \sum_r |\mathfrak{P}_r| - (k-2) \sum_r \lfloor \frac{|\mathfrak{P}_r|}{2} \rfloor$$

$$\geq \lfloor \frac{|\mathfrak{P}|+1}{t} \rfloor + |\mathfrak{P}|(k-1) - \lfloor \frac{|\mathfrak{P}|}{2} \rfloor (k-2)$$

Note that each node cannot be counted in two $N_{\ell_r^*}$, else we can find a path with smallest length. Therefore,

$$|J_k^\lambda| \geq \sum_r |N_{\ell_r^*}| \geq \lfloor \frac{|\mathfrak{P}|+1}{t} \rfloor + |\mathfrak{P}|(k-1) - \lfloor \frac{|\mathfrak{P}|}{2} \rfloor (k-2)$$

$$\Rightarrow |\mathfrak{P}| \leq \lfloor \frac{2|J_k^\lambda| - 2}{k} \rfloor$$

Overall, we have:

$$(1+1/t)\lfloor \frac{2|J_k^\lambda| - 2}{k} \rfloor \geq (1+1/t)|\mathfrak{P}| \geq |\mathcal{P}'| \geq |\mathcal{P}|.$$

The last (right) inequality comes from the fact that the start and end of paths $\mathcal{P}$ and $\mathcal{P}'$ are the same. Accordingly, if the length of $\mathcal{P}'$ is less than $\mathcal{P}$, then $\mathcal{P}'$ is the shortest path between the start and end nodes, which contradicts the assumption that $\mathcal{P}$ is the diameter. □

Theorem 3 (Edge Connectivity).

Proof. Let C be a minimal set of intra-layer edges whose removal will divide the $G[J_k^\lambda]$ into separate components, and $e = (u, v, \ell) \in C$. When all of the edges in C are removed, the vertices v and u will be in separate components, else the removal of e was unnecessary, violating the minimality of C. Since $(v, u) \in \mathcal{E}(J_k^\lambda)$ there are at least λ layers $\hat{L} = \{\ell_1, \ldots, \ell_\lambda\}$ in which v and u are joined by at least $k-2$ independent paths of length 2. Therefore, in each layer $\ell_i \in \hat{L}$, at least $k-1$ edges must be removed to place v and u in separate components. Therefore, $|C| \geq \lambda(k-1)$. □

Theorem 4 (FTCS Hardness and Non-Approximability).

Proof. We directly prove non-approximibility. The proof of NP-hardness follows a similar construction.

Assume there exists a polynomial time $(2-\epsilon)$-approximation algorithm for the FTCS problem, for any $\epsilon \in (0, 1)$. Let $\mathcal{A}$ be an algorithm that provides a solution H with an approximation factor $(2-\epsilon)$ of the optimal solution H^*. Let the set of query nodes be $Q = \emptyset$. Let $G = (V, E)$ be an instance of Maximum Clique Decision problem. Construct $G' = (V', E', L)$, where $V' = V$, $|L| = \lambda$, and for each $\ell \in L : E'_\ell = E$.

$(\Rightarrow)$: Clearly, if $\mathcal{A}$ outputs a solution $G'[H]$ that is a (k, λ)-FirmTruss with diameter 1, then $G_\ell[H]$ is clearly a clique of size at least k for each $\ell \in L$, which implies G contains a clique of size $\geq k$.

$(\Leftarrow)$: Suppose $diam(G'[H]) \geq 2$, so:

$$2diam(G'[H^*]) > (2-\epsilon)diam(G'[H^*]) \geq diam(G'[H]) \geq 2.$$

Since diameter is an integer, we deduce that $diam(G'[H^*]) \geq 2$. In this case, $G'_\ell[H^*]$ cannot possibly contain a clique of size k in any layer, for if it did, that clique would be the optimal solution to the FTCS problem with diameter 1. Therefore, since $G'_\ell[H^*] = G[H^*]$, we can distinguish between the YES and NO instances of the Maximum Clique Decision problem in polynomial-time, which is a contradiction.

To show the NP-hardness of d-FTCS problem, we reduce the Maximum Clique (decision version) to d-FTCS problem. By setting $d = 1$, the instance construction and proof is similar to the proof of non-approximability. □

Theorem 5 (FTCS Free-Rider Effect).

Proof. Let $G[H]$ be the maximal subgraph among all optimal solutions to the FTCS problem, and $G[H^*]$ be any query-independent optimal solution. We prove this theorem by contradiction. If $G[H^*] \setminus G[H] \neq \emptyset$, $G[H \cup H^*]$ is a connected (k, λ)-FirmTruss containing Q, and $diam(G[H \cup H^*]) \leq diam(G[H])$, then $G[H \cup H^*]$ is an optimal solution to the FTCS problem while $G[H] \subseteq G[H \cup H^*]$, violating the maximality of $G[H]$. □

Theorem 6 (FTCS-Global Quality Approximation).

Proof. Motivate by [44], we first prove:

$$dist_{G[H]}(H, Q) \leq dist_{G[H^*]}(H^*, Q).$$

Algorithm 1 outputs a sequence of intermediate graphs (in line 12) $\hat{G} = \{G_0, G_1, \ldots, G_{i-1}\}$, which are (k, λ)-FirmTruss containing query vertices Q. Therefore, we have $G_{i-1} \subseteq \cdots \subseteq G_1 \subseteq G_0 \subseteq G$ and also since the optimal solution is also a (k, λ)-FirmTruss, $G[H^*] \subseteq G_0$. We consider two cases:

(I) $G[H^*] \not\subseteq G_{i-1}$. Suppose the first deleted vertex $u^* \in H^*$ happens in graph G_s, this vertex must be deleted because of the distance constraint but not the FirmTruss maintenance. Accordingly, $dist_{G_s}(G_s, Q) = dist_{G_s}(u^*, Q) = dist_{G_s}(H^*, Q)$. Since H has the smallest query distance in $\hat{G}$ and $G[H^*] \subseteq G_s$, we have:

$$dist_{G[H]}(H, Q) \leq dist_{G_s}(G_s, Q) = dist_{G_s}(H^*, Q) \leq dist_{G[H^*]}(H^*, Q)$$

(II) $G[H^*] \subseteq G_{i-1}$. We prove this case by contradiction. Assume that $dist_{G_{i-1}}(G_{i-1}, Q) > dist_{G_{i-1}}(H^*, Q)$, then there exist a vertex $u^* \in G_{i-1} \setminus H^*$ with the largest query distance $dist_{G_{i-1}}(u^*, Q) = dist_{G_{i-1}}(G_{i-1}, Q) > dist_{G_{i-1}}(H^*, Q)$. In the next iteration, Algorithm 1 will delete u^* from G_{i-1}, and maintain the FirmTruss structure of G_{i-1}. Since $G[H^*] \subseteq G_{i-1}$, the output of this iteration is a feasible FirmTruss. This contradicts that G_{i-1} is the last feasible FirmTruss. Therefore, $dist_{G_{i-1}}(G_{i-1}, Q) \leq dist_{G_{i-1}}(H^*, Q)$, so we have:

$$dist_{G[H]}(H, Q) \leq dist_{G_{i-1}}(G_{i-1}, Q) \leq dist_{G_{i-1}}(H^*, Q)$$
$$\leq dist_{G[H^*]}(H^*, Q).$$

From above we have proved that $dist_{G[H]}(H, Q) \leq dist_{G[H^*]}(H^*, Q)$. Not that based on the triangle inequality

$$diam(G[H]) \leq 2dist_{G[H]}(H, Q).$$

Thus, we can conclude that:

$$diam(G[H]) \leq 2dist_{G[H]}(H,Q) \leq 2dist_{G[H^*]}(H^*,Q) \leq 2diam(G[H^*]).$$

□

Lemma 1 (Time Complexity of Maximal (k,λ)-FirmTruss).

Proof. Using a heap, we can find the λ-th largest element of vector S in $O(|L| + \lambda \log |L|)$ time so lines 4-7 take $O(|E|(|L| + \lambda \log |L|))$ time. We remove each edge schema exactly one time from the buckets, taking time $O(|E|)$. For $\ell \in L$, let $nb^{\ell}_{\geq}(u) = \{v|(v,u,\ell) \in E, deg_{\ell}(v) \geq deg_{\ell}(u)\}$, lines 10-16 take $O(\sum_{(v,u)\in\mathcal{E}} \sum_{\ell\in L} deg_{\ell}(u)|nb^{\ell}_{\geq}(u)|)$. For each layer ℓ, we know that $|nb^{\ell}_{\geq}(u)| \leq 2\sqrt{|E_{\ell}|}$ [74], so we can conclude that lines 10-16 take $O(\sum_{\ell\in L} |E_{\ell}|^{1.5})$. Finally, to update the Top-λ support for a given edge schema we need $O(|L|)$, so lines 17-20 takes $O(|E||L|)$ time. In this algorithm, we only need to store and keep the support vector of each edge schema so it takes $O(|E||L|)$ space. □

Theorem 7 (FTCS-Global Complexity).

Proof. First, Algorithm 1 finds the maximal (k,λ)-FirmTruss of G, which takes $O(\sum_{\ell\in L} |E_{\ell}|^{1.5} + |E||L| + |E|\lambda \log |L|)$ time as analyzed in Lemma 1. Then, it iteratively deletes the vertex with the largest query distance starting from G_0 with $|E_0|$ edges. In each iteration, the algorithm needs to compute the query distance using $O(|Q||E_0|)$ time and to perform FirmTruss maintenance using $O(\sum_{\ell\in L} |E_{\ell}|^{1.5})$. Therefore, for t iterations the time cost is $O(t(|Q||E_0| + \sum_{\ell\in L} |E_{\ell}|^{1.5}))$. In total, FTCS-Global algorithm takes $O(t(|Q||E_0| + \sum_{\ell\in L} |E_{\ell}|^{1.5}) + |E||L| + |E|\lambda \log |L|)$ time. The space complexity of FTCS-Global algorithm is dominated by finding G_0, which takes $O(|E||L|)$ space (Lemma 1). □

Theorem 8 (FTCS-Local Quality Approximation).

Proof. By contradiction, we first show that $dist_{G[H]}(H,Q) \leq dist_{G[H^*]}(H^*,Q)$. Let $dist_{G[H]}(H,Q) > dist_{G[H^*]}(H^*,Q)$, so there exists another FirmTruss with smallest diameter, so when $d = dist_{G[H^*]}(H^*,Q)$, the algorithm must find a FirmTruss with query distance of d, which is contradiction as $d = dist_{G[H]}(H,Q)$ is the smallest value of d that the algorithm finds a non-empty FirmTruss. Thus, $dist_{G[H]}(H,Q) \leq dist_{G[H^*]}(H^*,Q)$. Based on triangle inequality, $diam(G[H]) \leq 2dist_{G[H]}(H,Q)$. Overall:

$$diam(G[H]) \leq 2dist_{G[H]}(H,Q) \leq 2dist_{G[H^*]}(H^*,Q) \leq 2diam(G[H^*]).$$

□

Theorem 9 (FTCS-Local Complexity).

Proof. The proof is similar to analysis of time complexity of FTCS-Global algorithm. □

Theorem 10 (FirmTruss Decomposition Complexity).

Proof. By efficient computing of Top−λ degree for all values of λ, lines 4-6 take $O(|E||L| \log |L|))$ time. Given λ, we remove each edge schema exactly one time from the buckets, so lines 9-10 take $O(|E||L|)$. For $\ell \in L$, let $nb^{\ell}_{\geq}(u) = \{v|(v,u,\ell) \in E, deg_{\ell}(v) \geq deg_{\ell}(u)\}$. So for a given λ, lines 11-17 take $O(\sum_{(v,u)\in\mathcal{E}} \sum_{\ell\in L} deg_{\ell}(u)|nb^{\ell}_{\geq}(u)|)$. For each layer ℓ, we know that $|nb^{\ell}_{\geq}(u)| \leq 2\sqrt{|E_{\ell}|}$ [74], so we can conclude that lines 11-17 take $O(\sum_{\ell\in L} |E_{\ell}|^{1.5})$. Finally, to update the buckets for a given edge schema we need $O(|L|)$, so line 20 takes $O(|E||L|^2)$ time. □

Theorem 12 (AFTCS Hardness).

Proof. We reduce the problem of finding the densest subgraph with at least k vertices in single-layer graphs. Here, for simplicity, we assume that $p = 1$, we can easily extend this proof to any finite p. Given a single-layer graph $G = (V,E)$, construct an instance $G' = (V',E',L,\Psi)$, where $V' = V$, and for each $\ell \in L$, G'_{ℓ} is a complete graph over V'. Assume that $V' = \{u_1, u_2, \ldots, u_n\}$, where $n = |V'|$. For each vertex $u_i \in V'$, we construct a vector of size $n^2 + n$ as their attribute vector as follows:

$$\Psi(u_i)_j = \begin{cases} 1 & \text{if } 1 \leq j \leq n^2 \text{ and } (u_i, u_{j-(i-1)\times n}) \in E \\ & \text{or } (u_j, u_{i-(j-1)\times n}) \in E \\ \sqrt{2n - 2\mathrm{Nb}(u_i)} & \text{if } j = n^2 + i \\ 0 & \text{otherwise} \end{cases},$$

where $\mathrm{Nb}(u)$ is the number of u's neighbours in G. Now for each two vertices u and v that $(u,v) \in E : h(u,v) = \frac{1}{2n}$ and if $(u,v) \notin E : h(u,v) = 0$. Now, a solution to AFTCS on G' is the densest subgraph with at least k vertices in G. Also, each densest subgraph with at least k vertices is a (k,λ)-FirmTruss with the maximum homophily score. □

Theorem 13 (AFTCS Free-Rider Effect).

Proof. Let $G[H]$ be the maximal subgraph among all optimal solutions to the AFTCS problem, and $G[H^*]$ be any query-independent optimal solution. We prove this theorem by contradiction. If $G[H^*] \setminus G[H] \neq \emptyset$, $G[H \cup H^*]$ is a connected (k,λ)-FirmTruss containing Q, and $\Gamma_p(G[H \cup H^*]) \geq \Gamma_p(G[H])$, then $G[H \cup H^*]$ is an optimal solution to the AFTCS problem while $G[H] \subseteq G[H \cup H^*]$, violating the maximality of $G[H]$. □

Theorem 14 (AFTCS-Approx Complexity).

Proof. The time complexity of finding G_0 is $O(\sum_{\ell\in L} |E_{\ell}|^{1.5} + |E||L| + |E|\lambda \log |L|)$ (Lemma 1). Let t be the number of iterations, and V_0 and E_0 be the vertex set and edge set of G_0. The computation of homophily score for all vertices takes $O(d|V_0|^2)$ time, where d is the dimension of the attribute vector. The time complexity of maintenance algorithm is $O(t|V_0| + |E_0|)$ in t iterations, since each edge in G_0 is removed at most one time in all iteration and in each iteration we need to check the Top-λ support of each edge schema. Since in each iteration we delete at least one node and its incident edges in all layers, which is at least $(k-1) \times \lambda$, the total number of iterations t is $O(\min\{|V_0| - k, \frac{|E_0|}{\lambda.(k-1)}\})$. □

Theorem 15 (AFTCS-Approx Quality Approximation). We first mention two useful lemmas:

Lemma 2. *Given $a, b \in \mathbb{R}^+$ and $a \geq b$, and a real value $p \geq 1$:*

$$a^p - (a-b)^p \leq pa^{p-1}b.$$

Proof. Given $p \geq 1$ and $a, b \in \mathbb{R}^+$, let $f(x) = x^p, \ \forall\, x \in \mathbb{R}$. Based on the mean value theorem, there is a $c \in [a-b, a]$ such that:

$$f'(c) = \frac{f(a) - f(a-b)}{b} \Rightarrow pa^{p-1} \geq pc^{p-1} = f'(c) = \frac{a^p - (a-b)^p}{b}$$

□

Lemma 3. *Given $p \geq 1$, function $\Xi(S) = |S|\Gamma_p^p(S) = \sum_{v \in S} h_S(v)^p$ is supermodular.*

Proof. We know that function $f(x) = x^p$ for $p \geq 1$ and $x \geq 0$ is increasing convex function. Since function $h_S(v)$ is a modular function, its combination with f is supermodular. Accordingly, p-th power of $h_S(v)$ is a supermodular function. Finally, we know that the summation of supermodular functions is supermodular. □

Based on lemmas 2 and 3, we can prove the theorem:

Proof. Let H^* be the optimal solution and H be the output of Algorithm 6. Since H^* is the optimal solution, removing a node u^* will produce a subgraph with homophily score at most $\Gamma_p^p(H^*)$, therefore, we have:

$$\frac{1}{|H^*| - 1}\left(|H^*|\Gamma_p^p(H^*) - \Delta_{u^*}(H^*)\right) \geq \Gamma_p^p(H^*)$$
$$\Rightarrow \Gamma_p^p(H^*) \leq \Delta_{u^*}(H^*).$$

Note that u^* is an arbitrary vertex in H^* and $H^* \backslash \{u^*\}$ is not necessarily a (k, λ)-FirmTruss. Since $|S|\Gamma_p^p(S)$ is a supermodular function for $p \geq 1$, when $H^* \subseteq S, \Delta_{u^*}(H^*) \leq \Delta_{u^*}(S)$ (increasing differences). Let S denote the subgraph maintained by the algorithm right before the first node $u^* \in H^*$ is removed. This node cannot be removed by maintaining the subgraph (line 10) as $u^* \in H^*$ and $H^* \subseteq S$ is a (k, λ)-FirmTruss. Since u^* is the first node to be removed from H^*, $H^* \subseteq S$ and so $\Delta_{u^*}(H^*) \leq \Delta_{u^*}(S)$. Since $p \geq 1$, in each iteration we remove a node with minimum value of Δ, $u^* = \arg\min_{u \in S} \Delta_u(S)$, so $\Delta_{u^*}(S)$ is smaller than the average value of $\Delta_u(S)$ across $u \in S$. Therefore, we have:

$$\Gamma_p^p(H^*) \leq \Delta_{u^*}(H^*) \leq \Delta_{u^*}(S) \leq \frac{1}{|S|}\sum_{u \in S} \Delta_u(S)$$
$$= \frac{1}{|S|}\left(\sum_{u \in S} h_S(u)^p + \sum_{u \in S}\sum_{v \in S/\{u\}} h_S(v)^p - [h_S(v) - h(v,u)]^p\right)$$
$$\leq \frac{1}{|S|}\left(\sum_{u \in S} h_S(u)^p + \sum_{u \in S}\sum_{v \in S/\{u\}} p h_S(v)^{p-1} h(v,u)\right)$$
$$= \frac{1}{|S|}\left(\sum_{u \in S} h_S(u)^p + p\sum_{u \in S} h_S(v)^p\right) = (p+1)\Gamma_p^p(S)$$

The one before last step follows from the inequality in Lemma 2. Accordingly, $\Gamma_p(H^*) \leq (p+1)^{1/p}\Gamma_p(S)$. Since $S \in \{G_0, G_1, \ldots, G_{i-1}\}$, and H is the output of the algorithm, we have:

$$\Gamma_p(H^*) \leq (p+1)^{1/p}\Gamma_p(S) \leq (p+1)^{1/p}\Gamma_p(H).$$

□

Theorem 16 (Correctness of Exact-MaxMin AFTCS).

Proof. Let H be the found subgraph by Exact-MaxMin algorithm and H^* be the optimal solution, we prove this theorem by contradiction. If $H \neq H^*$, let s be the first time that a vertex $u^* \in H^*$ is removed. Accordingly, $H^* \subseteq G_s$ and since u^* is removed, it must be removed when we remove a vertex with a minimum value of h_S but H^* is a subgraph with the maximum minimum homophily score. Thus, $\Gamma_{-\infty}(H^*) \leq h_{H^*}(u^*) \leq h_{G_s}(v)$ for any $v \in V[G_s]$. Therefore, $\Gamma_{-\infty}(H^*) \leq \Gamma_{-\infty}(G_s)$, which is contradiction. □

B TIGHTNESS EXAMPLES

Density Lower Bound. For simplicity assume that $\beta = 0$ (for other values of β the example is similar), for a given k and λ, we construct a multilayer graph $G = (V, E, L)$, such that $|L| = \lambda$ and for each $\ell \in L$, G_ℓ is a k-clique. In this case, $\rho_\beta(G) = \frac{k-1}{2}$ and lower bound for the density of G is $\frac{k-1}{2\lambda} \times \lambda = \frac{k-1}{2}$. Which shows that the lower bound is tight.

Diameter Upper Bound. First, the diameter of a connected k-truss with n vertices in a single layer graph is no more than $\lfloor \frac{2n-2}{k} \rfloor$, and this is a tight upper bound. Therefore, for a given k, there exists a single layer k-truss, H_k, with diameter $\lfloor \frac{2n-2}{k} \rfloor$. Now, for any arbitrary k and λ, we construct a multilayer graph $G = (V, E, L)$, such that $|L| = \lambda$ and for each $\ell \in L$, $G_\ell = H_k$. In this case $T = 1$ and diameter of G is $\lfloor \frac{2|G|-2}{k} \rfloor$.

Edge Connectivity. For a given k, λ, we construct a multilayer graph $G = (V, E, L)$, such that $|L| = \lambda$ and for each $\ell \in L$, G_ℓ is a k-clique. To make G disconnected, we have to make G_ℓ for all $\ell \in L$ disconnected. Since the edge connectivity of a k-clique is $k-1$, we have to remove at least $\lambda \times (k-1)$ intra-layer edges to make G disconnected.

Approximation Factor of AFTCS-Approx. In the proof of Theorem 12, we see the following corollary:

Corollary 1. *Given a set of tuples $\mathcal{E} \subseteq V \times V$, we can construct attribute vectors for each $u \in V$ such that $\forall (u,v) \in \mathcal{E} : h(u,v) = c$, and $\forall (u,v) \notin \mathcal{E} : h(u,v) = 0$, where c is a constant number.*

Given a complete multilayer graph $G = (V, E, L)$, we want to construct the set of tuples $\mathcal{E} \subseteq V \times V$ such that AFTCS-Approx returns an approximation solution that asymptotically approaches $(p+1)^{1/p}$ of the exact solution of AFTCS problem.

For a subgraph S, we let $\delta(S) = \min_{u \in S} \Delta_u(S)$. In order to construct the tightness example, we assume that $V = V_1 \cup V_2$, and so $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$ such that $\mathcal{E}_1 \subseteq V_1 \times V_1$ and $\mathcal{E}_2 \subseteq V_2 \times V_2$. Now we need to construct each V_1 and V_2, and so $\mathcal{E}_1$ and $\mathcal{E}_2$ accordingly. Given an arbitrary number $q \in \mathbb{N}$, we let V_1 be a set of nodes with size $q+1$ and also $\mathcal{E}_1 = V_1 \times V_1$. Accordingly, based on Corollary 1, we have:

$$\delta(V_1) = \min_{u \in V_1} \Delta_u(V_1) = (cq)^p + qc^p(q^p - (q-1)^p), \tag{3}$$
$$\Gamma_p(V_1) = cq \tag{4}$$

Now let k and t be two arbitrary integers such that $t < \frac{k}{2}$, we consider a set of nodes $V_2 = \{u_1, u_2, \ldots, u_k\}$ such that $V_1 \cap V_2 = \emptyset$, and $|V_2| = k$. In order to construct $\mathcal{E}_2$, for each node $u_i \in V_2$, we add (u_i, u_j) to $\mathcal{E}_2$ for each $i+1 \leq j \leq \max\{i+t, k\}$. Accordingly,

we can see:

$$\delta(V_2) = \Delta_{u_1}(V_2) = (ct)^p + c^p \left(\sum_{i=2}^{t+1} (t+i-1)^p - (t+i-2)^p \right)$$
$$= (ct)^p + c^p \left((2t)^p - t^p \right) = c^p (2t)^p, \quad (5)$$

and

$$\Gamma_p(V_2) = \frac{c}{\sqrt[p]{k}} \left(\sum_{i=1}^{t} (t+i-1)^p + \sum_{i=t+1}^{k-t} (2t)^p + \sum_{i=k-t+1}^{k} (t+i-1)^p \right)^{1/p} \quad (6)$$

Note that q is an arbitrary number. So, we let $q = \lceil \frac{2t}{(p+1)^{1/p}} \rceil + 1$, and accordingly, we have:

$$\delta(V_1) = (cq)^p + qc^p (q^p - (q-1)^p) \geq c^p (2t)^p = \delta(V_2). \quad (7)$$

Now consider a complete attributed multilayer graph $G = (V_1 \cup V_2, E, L, \Psi)$. Let $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$, based on Corollary 1, we can construct Ψ such that $\forall (u,v) \in \mathcal{E} : h(u,v) = c$, and $\forall (u,v) \notin \mathcal{E} : h(u,v) = 0$, where c is a constant number. Since $\delta(V_1) \geq \delta(V_2)$, AFTCS-Approx algorithm starts from removing all nodes from V_2, while based on the value of $\Gamma_p(V_1)$ and $\Gamma_p(V_2)$, we know that the exact solution is the induced subgraph by V_2. Therefore, the found solution by AFTCS-Approx algorithm is the induced subgraph by V_1. In order to see the approximation ratio for this example, we need to calculate $\frac{\Gamma_p(V_2)}{\Gamma_p(V_1)}$. To this end, we have:

$$\frac{\Gamma_p(V_2)}{\Gamma_p(V_1)} = \frac{\sqrt[p]{\left(\sum_{i=1}^{t} (t+i-1)^p + \sum_{i=t+1}^{k-t} (2t)^p + \sum_{i=k-t+1}^{k} (t+i-1)^p \right)}}{q \times \sqrt[p]{k}}$$
$$\geq \frac{\sqrt[p]{(1-\frac{2t}{k})(2t)}}{q} \geq \left(\sqrt[p]{1-\frac{2t}{k}} \right) \left(\frac{\sqrt[p]{p+1}}{1+\frac{\sqrt[p]{p+1}}{t}} \right) = M_p(k,t)$$

Let $t \in O(k)$ and fixed $p \geq 1$, we have $\lim_{k \to \infty} M_p(k,t) = \sqrt[p]{p+1}$, which means the approximation factor converges to $\sqrt[p]{p+1}$.

C PROPERTIES OF HOMOPHILY SCORE FUNCTION

Positive Influence of Similar Vertices. The first property of homophily score is positive influence of similar nodes. As we discussed, $\Delta_u(S)$ is the value that droping $u \in S$ changes the numerator of the homophily score:

$$\Gamma_p^p(S/\{u\}) = \frac{1}{|S|-1} \left(|S| \Gamma_p^p(S) - \Delta_u(S) \right),$$

where

$$\Delta_u(S) = h_S(u)^p + \left(\sum_{v \in S/\{u\}} h_S(v)^p - [h_S(v) - h(v,u)]^p \right). \quad (8)$$

Similarly, $\Delta_u(S \cup \{u\})$ is the value that adding u to S channges the numerator of the homophily score. Next fact, illustrates when adding a vertex can increase the homophily score.

Fact 2. *Adding a vertex u to a subgraph S increases the homophily score of subgraph S iff $\Delta_u(S \cup \{u\})^{1/p} > \Gamma_p(S)$.*

Example 10. *Let $p = 1$, Fact 2 states that adding a node u with node-homophily score $h_S(u) > \frac{1}{2|S|} \sum_{v \in S} h_S(v)$ to subgraph S, increases the homophily score of the subgraph.*

Negative Influence of Dissimilar Vertices. Adding dissimilar vertices to the community decreases the homophily score.

Fact 3. *Adding a vertex u to a subgraph S decreases the homophily score of subgraph S iff $\Delta_u(S \cup \{u\})^{1/p} < \Gamma_p(S)$.*

Example 11. *Let $p = 1$, Fact 3 states that adding a node u with node-homophily score $h_S(u) < \frac{1}{2|S|} \sum_{v \in S} h_S(v)$ to subgraph S, decreases the homophily score of the subgraph.*

Non-monotone Property. Non-monotonicity is a desirable property for the community model. Assume that if the attribute correlation measure for the community model is monotone and increasing, then always the maximal FirmTruss is the optimal solution. At the same time, we know that maximal FirmTruss may include some free riders. The introduced homophily score function is non-monotone: as we discussed in Facts 2 and 3, adding a vertex might increase or decrease the homophily score.

Non-submodularity and Non-supermodularity. Optimization problems over submodular or supermodular functions lend themselves to efficient approximation. We thus study whether our homophily score function is submodular or supermodular with respect to the set of vertices. By a contradictory example, we show that our homophily score function is neither submodular nor supermodular.

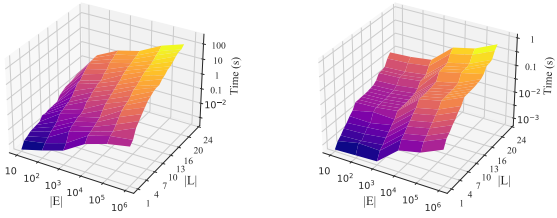
Example 12. *Consider a graph G with $V = \{u_1, u_2, v_1, v_2\}$. Let $p = 1$. Assume that $h(v_1, u_2) = h(v_2, u_1) = 0.1$, $h(v_2, u_2) = 0.5$, $h(v_1, u_1) = 0.2$, $h(u_1, u_2) = 0.3$ and $h(v_1, v_2) = 0$. Now consider the sets $S = \{u_1\}$ and $T = \{u_1, u_2\}$. Let us compare the marginal gains $\Gamma_p(S \cup \{v^*\}) - \Gamma_p(S)$ and $\Gamma_p(T \cup \{v^*\}) - \Gamma_p(T)$, from adding the new vertex $v^* \notin T$ to S and T. Suppose $v^* = v_1$, then we have $\Gamma_p(S \cup \{v_1\}) - \Gamma_p(S) = (2 \times 0.2)/2 - 0 = 0.2 > \Gamma_p(T \cup \{v_1\}) - \Gamma_p(T) = 2(0.1+0.2+0.3)/3 - (2 \times 0.3)/2 = 0.1$, violating supermodularity. On the other hand, suppose $v^* = v_2$. Then we have $\Gamma_p(S \cup \{v_2\}) - \Gamma_p(S) = (2 \times 0.1)/2 - 0 = 0.1 < \Gamma_p(T \cup \{v_2\}) - \Gamma_p(T) = 2(0.1+0.5+0.3)/3 - (2 \times 0.3)/2 = 0.3$, which violates submodularity. Thus, this function is non-submodular and non-supermodular.*

D REPRODUCIBILITY

All algorithm implementations and code for our experiments are available at Github.com/joint-em/FTCS.

E DATASETS

We perform extensive experiments on thirteen real networks including social, genetic, co-authorship, financial, and co-purchasing networks, whose main characteristics are summarized in Table 1. Using the raw Noordin terrorist network data [64], we constructed the multilayer terrorist relationship network. Each node in this network represents a terrorist, and each layer represents a type of relationship (e.g., friendships, organizations, educational institutions, businesses, and religious institutions, etc.). RM [50] has 10 networks, each with 91 nodes. Nodes represent phones and one edge exists if two phones detect each other under a mobile network. Each network describes connections between phones in a



(a) Global (b) iLocal

Figure 10: Scalability of Global and iLocal algorithms with varying the number of layers and the number of edges.

month. Phones are divided into two communities according to their owners' affiliations. Brain is derived from the functional magnetic resonance imaging (fMRI) of 520 individuals with the same methodology used in [53]. It contains 190 individuals in the condition group, labelled as ADHD and 330 individuals in the control group, labelled as TD. Here, each layer is the brain network of an individual, nodes are brain regions, and an edge measures the statistical association between the functionality of nodes. Each community is a functional system in the brain. YEAST is a biological network, in which the layers correspond to interaction networks of genes in Saccharomyces cerevisiae (which was obtained via a synthetic genetic-array methodology) and correlation-based networks in which genes with similar interaction profiles are connected to each other [17]. FAO represents different types of trade relationships among countries in which layers represent products, nodes are countries and edges at each layer represent import/export relationships among countries [18]. Obama network represent re-tweeting, mentioning, and replying among Twitter users, focusing on Barack Obama's visit in 2013 [61]. DBLP is a co-authorship network until 2014 that is used in [29]. Amazon is a co-purchasing network, in which each layer corresponds to one of its four snapshots between March and June 2003 [54]. Higgs is a two-layer network where the first layer represents tracking the spread of news about the discovery of the Higgs boson on Twitter, and the second layer is for the following relation [16]. Friendfeed contains commenting, liking, and following interactions among users of Friendfeed [10]. StackOverflow represents user interactions from the StackExchange site where each layer corresponds to interactions in one hour of the day [52]. Finally, Google+ is a billion-scale network consisting of 4 Google+ snapshots between July and October 2011 [32].

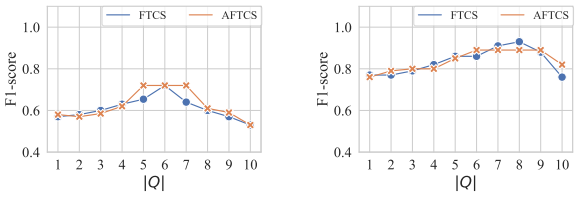
F ADDITIONAL EXPERIMENTS

F.1 Scalability of Other Algorithms

We test our algorithms using different versions of StackOverflow obtained by selecting a variable #layers from 1 to 24 and also with different subsets of edges. Figure 10 shows the results of the Global, and iLocal algorithms. The running time of these algorithms scales linearly in #layers. By varying #edges, all algorithms scale gracefully. As expected, the iLocal algorithm is less sensitive to varying #edges than #layers.

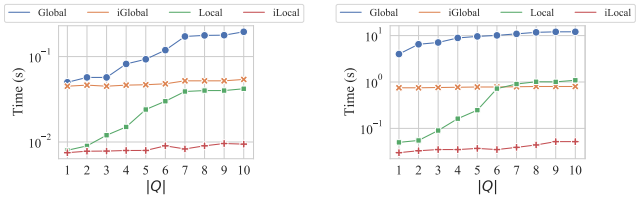
F.2 Effect of Query Set Size

Effectiveness. We evaluate the effect of the query set size on the quality of found communities by FTCS and AFTCS. Figure 11(a) and (b) show the F1-score as a function of query set size, $|Q|$, on Terrorist and RM datasets. We observe that increasing the size of the query set first results in increasing F1-score, but after it achieves its maximum, the F1-score decreases. The reason is increasing $|Q|$ increases F1-score as long as $|Q|$ is less than the size of the ground-truth community. If $|Q|$ exceeds the size of the community, the F1-score starts to decrease.



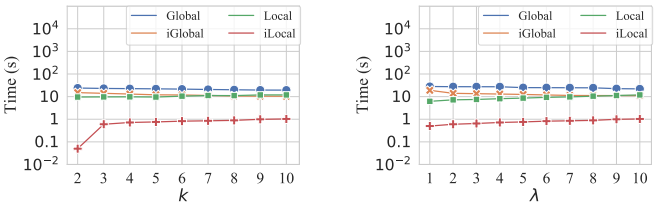
(a) Terrorist dataset (b) RM dataset

Figure 11: Effect of the query set size, |Q|, on F1-score.



(a) Terrorist dataset (b) DBLP dataset

Figure 12: Effect of the query set size, |Q|, on running time.



(a) Effect of k (b) Effect of λ

Figure 13: Effect of k and λ on running time (Brain).

Efficiency. We evaluate the effect of the query set size on the efficiency of proposed algorithms. Figure 12(a) and (b) show the query time as a function of query set size, $|Q|$, on Terrorist and DBLP datasets. We observe that increasing the size of the query set results in increasing the running time. Increasing the query set size results in increasing the running time as our algorithms need to explore more nodes in the neighborhood of query nodes.

F.3 Effect of k and λ on Efficiency

Figure 13 shows the result of the effect of varying k and λ on running time. As we discussed earlier, the larger k and λ for Global algorithms results in less running time since algorithms generate a smaller G_0. However, the larger k and λ increases the running time of Local algorithms since it needs to count more nodes in the neighbourhood of query nodes to find a (k, λ)-FirmTruss.

Table 5: Case study of DBLP: Effect of k and λ on Quality.

Query node		$k=2$		$k=3$		$k=4$		$k=5$		$k=6$		$k=7$		$k=8$		$k=9$	
		$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$	$\lambda=1$	$\lambda=2$
Jiawei Han	G_0 Size	952305	952305	367471	308285	231953	104779	148625	29297	100171	18	70236	17	50111	-	37476	-
	Community Size	**440**	**440**	**284**	**264**	**225**	**102**	**169**	**34**	**113**	**10**	**83**	**10**	**60**	-	**34**	-
	#Free-riders	951865	951865	367187	308021	231728	104677	148456	29233	100058	8	70153	7	50051	-	37442	-
	F1-Score	0.27	0.27	0.39	0.41	0.47	0.78	0.57	0.63	0.75	0.26	0.85	0.26	0.94	-	0.63	-
Samuel Madden	G_0 Size	952305	952305	367471	308285	231953	104779	148625	29297	100171	5107	70236	17	50111	-	37476	-
	Community Size	**210**	**210**	**173**	**152**	**144**	**99**	**122**	**59**	**113**	**17**	**100**	**17**	**83**	-	**67**	-
	#Free-riders	952095	952095	367298	308133	231809	104680	148543	29238	100058	5090	70136	0	50028	-	37409	-
	F1-Score	0.44	0.44	0.52	0.57	0.59	0.75	0.66	0.97	0.69	0.44	0.75	0.44	0.84	-	0.93	-
Yoshua Bengio	G_0 Size	952305	952305	367471	308285	231953	104779	148625	29297	100171	17	70236	17	50111	17	37476	17
	Community Size	**116**	**116**	**109**	**69**	**81**	**34**	**58**	**23**	**36**	**17**	**23**	**17**	**17**	**17**	**17**	**17**
	#Free-riders	952189	952189	367362	308216	231872	104745	148567	29274	100135	0	70213	0	50094	0	37459	0
	F1-Score	0.32	0.32	0.34	0.48	0.43	0.75	0.55	0.98	0.72	0.89	0.98	0.89	0.89	0.89	0.89	0.89

F.4 Effect of k and λ on Found Communities

Since k and λ are given by the user, one might ask what happens if inappropriate values of k and λ are given? We conduct a case study on the DBLP dataset to evaluate the effect of k and λ, and to show the effectiveness of the FTCS in removing free riders. Table 5 shows the G_0 size, the size of found community by FTCS, and the # free-riders for different values of k and λ, and for different query nodes, "Jiawei Han", "Samuel Madden", "Yoshua Bengio". These results show that giving inappropriate values of k and λ results in increasing the size of G_0. However, since FTCS minimizes the diameter, it is able to remove far nodes from the query node and always finds a community with appropriate size and quality. Accordingly, FTCS is robust to inappropriate values of k and λ.

We know that given $\lambda \in \mathbb{N}^+$ and $k \geq 0$, the $(k+1,\lambda)$-FT and $(k,\lambda+1)$-FT of a multilayer graph are subgraphs of its (k,λ)-FT. Another property of FTCS that is shown in Table 5 is its hierarchical structure. This is an important property for a community model as by varying the value of k and λ it can find communities at different levels of granularity.

Remark 2. *How is a user supposed to choose the parameters? The reported experiments on the effect of k, λ, and p offer some guidance. Figure 4 in the paper shows the effect of k, λ, and p on the efficiency. Based on this experiment, the value of p does not affect the efficiency much and the running times are comparable for different values of p. Also, our iLocal algorithm shows a stable trend, whereby varying k and λ has a negligible impact on the running time. On the other hand, in the Global and iGlobal algorithms we see a decreasing trend and in the Local algorithm, we see an increasing trend, as k, λ are increased. Thus, when an index is available, iLocal is clearly the algorithm of choice as it has a stable performance fairly independent of the parameters. Otherwise, one can use the Global algorithm for large values of k and λ and use Local algorithm for small values of k and λ. Table 5 shows the effect of k and λ on the quality of the communities found from the DBLP dataset. We measured the size of the largest connected (k,λ)-truss found, the number of free-riders removed, the final community size, as well as the F1 score w.r.t. the ground truth. These results show the robustness of our approach even in many cases when different values of k and λ are chosen.*

In sum, a user can start with some initial values for the parameters p, k, λ, subsequently modify them, and explore the various extracted communities and choose the one that best matches their needs. Our experiments show that either using iLocal (if index is available) or using Global (if k, λ are large), or Local (if k, λ are small), each resulting community can be extracted in about a second, permitting efficient exploration. In terms of quality, our experiments show that even the worst choice of p leads to a quality that is better than all the baselines.

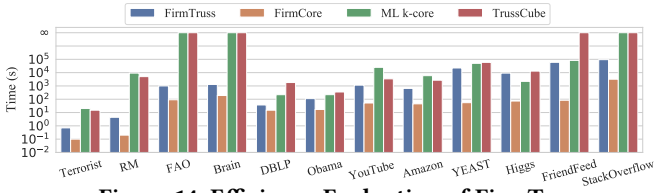


Figure 14: Efficiency Evaluation of FirmTruss.

F.5 Efficiency of FirmTruss Decomposition

We compare the efficiency of FirmTruss decomposition algorithm with the state-of-the-art dense strcture mining algorithms, FirmCore [36], ML **k**-core [29], and TrussCube [39], in multilayer graphs. Figure 14 shows the running time of these algorithms on different datasets. FirmTruss outperforms all other algorithms, except FirmCore, and can be scaled to graphs with hundred thousands of edges. As we previously discussed, although FirmCore is more efficient that FirmTruss, FirmTrusses are denser than FirmCores. It is a trade-off between density and efficiency.

G COMPARISON OF DIFFERENT ML COMMUNITY SEARCH MODELS

Example 13. *In Figure 1, the green nodes are densely connected and intuitively they form a community. Let v_{10} be the query node, we expect a good CS model to return the green nodes as the corresponding community of v_{10}. Let $k = 4$ and $\lambda = 1$, we can see that FTCS returns the green nodes as a community. For TrussCube,[2] when the number of selected layer is 2, or 3, it returns the entire graph as the community of v_{10}, which includes many false positive nodes. Note that, choosing only 1 layer is equivalent to a simple truss in a simple graph. Let $\beta = 1$, then k-core returns a subgraph including $\{v_1, v_2, v_3, v_8, v_9, v_{10}, v_{11}, v_{13}\}$ as the community, which misses several nodes in the expected community (i.e., the green nodes). Let $\beta = 3$, ML k-core returns a subgraph including all nodes as the community, which includes many false positives. Finally, ML-LCD returns $\{v_9, v_{10}, v_{11}\}$ as the corresponding community, which misses several green nodes.*

[2]There is no prior community search model based on TrussCube. Here we consider a baseline that finds the maximal connected TrussCube containing query nodes.