A PROOFS

A.1 Property 1 (Uniqueness)

Proof. We prove this by contradiction. Suppose there are two distinct $(\mathcal{F}, \lambda, k)$-FoTrusses H and H'. According to Definition 2, for all edge schema $(u, v) \in E(H)$ ($(u, v) \in E(H')$ resp.), we have the uphold layer set of (u, v) $\mathcal{L} = uh(u, v, H, k)$ ($\mathcal{L} = uh(u, v, H', k)$ resp.) satisfying $|\mathcal{L}| \geq \lambda$ and $\mathcal{F} \subseteq \mathcal{L}$. Therefore, for each $(u, v) \in E(H) \cup E(H')$, we have its uphold layer set satisfying $|\mathcal{L}| \geq \lambda$ and $\mathcal{F} \subseteq \mathcal{L}$. The graph edge induced by $E(H) \cup E(H')$ should also be a $(\mathcal{F}, \lambda, k)$-FoTruss. It contradicts the maximality of H and H'. Thus, the property holds. □

A.2 Property 2 (Hierarchy)

Proof. Suppose H is a $(\mathcal{F}, \lambda, k+1)$-FoTruss, and $(u, v) \in E(H)$ is an edge schema in H. According to Definition 2, we have the uphold layer set $\mathcal{L} = uh(u, v, H, k+1)$ satisfying $|\mathcal{L}| \geq \lambda$ and $\mathcal{F} \subseteq \mathcal{L}$. Suppose the uphold layer set under k is $\mathcal{L}' = uh(u, v, H, k)$ of edge schema (u, v). Then we have for each uphold layer $l \in L$, (u, v, l) is contained in at least $k+1$ triangles, so (u, v, l) should also be contained in at least k triangles. Then for each $l \in L$ we have $l \in L'$. We have $L \subseteq L'$ and $|\mathcal{L}'| \geq \lambda$ and $\mathcal{F} \subseteq \mathcal{L}'$. This means each edge schema in the $(\mathcal{F}, \lambda, k+1)$-FoTruss is also an edge schema in the $(\mathcal{F}, \lambda, k)$-FoTruss. Thus, the property holds. □

A.3 Property 3 (Containment)

Proof. Let H_1 be a $(\mathcal{F}_1, \lambda_1, k)$-FoTruss and let H_2 be a $(\mathcal{F}_2, \lambda_2, k)$-FoTruss. By Definition 2, we have the uphold layer set of each edge schema $(u, v) \in E(H_2)$ $\mathcal{L} = uh(u, v, H_2, k)$ satisfying $|\mathcal{L}| \geq \lambda_2 \geq \lambda_1$ and $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \mathcal{L}$. So all the edge schemas in $E(H_2)$ satisfy the background and focus constraint of $(\mathcal{F}_1, \lambda_1, k)$. Thus, the property holds. □

A.4 Property 4 (Minimum Degree)

Proof. Suppose H is a $(\mathcal{F}, \lambda, k)$-FoTruss, and $(u, v) \in E(H)$ is an edge schema in H. According to Definition 2, we have the uphold layer set $\mathcal{L} = uh(u, v, H, k)$ satisfying $|\mathcal{L}| \geq \lambda$ and $\mathcal{F} \subseteq \mathcal{L}$. In each layer $l \in \mathcal{L}$, we have u and v contained in at least $k-2$ triangles, meaning that u and v have at least $k-2$ common neighbors. Thus $|N_l(v, G)| \geq k-1$. □

A.5 Theorem 1 (Decomposition Hardness)

Proof. As will be presented in Section 3, there exist algorithms that solve the FoTruss decomposition problem in $f(|L|) \cdot |E|^{O(1)}$ time, where f is a function of $|L|$ that is independent of $|E|$. Thus, the theorem holds. □

A.6 Theorem 2 (Density)

This proof is the same as the proof of density lower bound in FirmTruss [3].

A.7 Theorem 3 (Edge Connectivity)

Proof. Suppose after removing the minimal set of edges S, we will get two connected components C_1 and C_2, where there is no edge (u, v, l) that holds $u \in C_1$, $v \in C_2$, and $l \in L$. We calculate how many edges will be removed to get C_1 and C_2, i.e., $|S|$.

If $|C_1| = 1$ or $|C_2| = 1$. Let's assume $|C_1| = 1$, which means there is only one vertex $u \in C_1$. Before the removal, C_1 and C_2 are triangle-connected, which means there are at least two vertices $v_1, v_2 \in C_2$, where edge schemas $(u, v_1), (u, v_2), (v_1, v_2)$ exist. Meanwhile, $\tau(u, v_1, \mathcal{F}, \lambda) \geq k$ and $\tau(u, v_2, \mathcal{F}, \lambda) \geq k$ hold. So there will be at least λ layers $\mathcal{L} = \{l_1, \ldots, l_\lambda\}$ where u, v_1 (u, v_2, resp.) has at least $k-3$ common neighbors except v_2 (v_1, resp.). These common neighbors w all belong to C_2 because $|C_1| = 1$. So there will be (u, w, l_i) edge that needs to be removed. In each layer $l_i \in \mathcal{L}$, there will be at least $(k-3)+2$ edges that need to be removed. In total, $|S| \geq \lambda(k-1)$.

If $|C_1| \neq 1$ and $|C_2| \neq 1$. According to triangle-connectivity, there will be two vertices $u_1, u_2 \in C_1$ and $v_1, v_2 \in C_2$, where edge schemas $(u_1, v_1), (u_1, v_2), (u_2, v_1), (u_1, u_2), (v_1, v_2)$ exist. Meanwhile, $\tau(u_1, v_1, \mathcal{F}, \lambda) \geq k$, $\tau(u_1, v_2, \mathcal{F}, \lambda) \geq k$, and $\tau(u_2, v_1, \mathcal{F}, \lambda) \geq k$ hold. Similarly, there will be at least λ layers $\mathcal{L}_1 = \{l_1, \ldots, l_\lambda\}$ where u_1, v_1 (u_1, v_2, resp.) has at least $k-3$ common neighbors except v_2 (v_1, resp.). We can assume these common neighbors w all belong to C_2 to make $|S|$ small. Meanwhile, there will be at least λ layers $\mathcal{L}_2 = \{l_1, \ldots, l_\lambda\}$ where u_1, v_1 (u_2, v_1, resp.) has at least $k-3$ common neighbors except u_2 (u_1, resp.). In each layer $l_i \in \mathcal{L}_1$, there will be at least $(k-3)$ edges that need to be removed. In each layer $l_i \in \mathcal{L}_2$, there will also be at least $(k-3)$ edges that need to be removed. In total, $|S| \geq 2\lambda(k-3) + 3\lambda$.

In summary, $|S| \geq \lambda(k-1)$. □

A.8 Theorem 4 (Diameter)

Research on triangle-connected truss communities in single-layer graphs [55] has proven that the size and diameter of a triangle-connected k-truss satisfy the following relationship:

Theorem 6. If d is the diameter of a triangle-connected k-truss with n vertices, then $d \leq \lfloor \frac{2n}{k+1} \rfloor$

Proof. Suppose $\mathcal{P} = \{(v_1, l_1) \to (v_2, l_2) \to \cdots \to (v_p, l_p)\}$ is a path of the triangle-connected $(\mathcal{F}, \lambda, k)$-FoTruss H. We can split $\mathcal{P}$ into two sub-paths: $\mathcal{P}_1$ contains all the adjacent (v_i, l_i) and (v_{i+1}, l_{i+1}) in path $\mathcal{P}$ where $v_i \neq v_{i+1}$. While $\mathcal{P}_2$ contains all the adjacent (v_i, l_i) and (v_{i+1}, l_{i+1}) in path $\mathcal{P}$ where $v_i = v_{i+1}$ and $l_i \neq l_{i+1}$. In other words, $\mathcal{P}_1$ represents the intra-layer edges in the path, while $\mathcal{P}_2$ represents the inter-layer edges in the path. We have $diam(H) = min_{\mathcal{P} \in H}(|\mathcal{P}|)$ and $|\mathcal{P}| = |\mathcal{P}_1| + |\mathcal{P}_2|$. Next, we construct a $\mathcal{P}'$ composed of $\mathcal{P}_1'$ and $\mathcal{P}_2'$ whose upper bound can be determined.

We first construct a $\mathcal{P}_1'$ with an upper bound. We can merge all the edge schemas in $H = (V, L, E_L)$ into a single-layer graph $H' = (V, E)$, where $E = \cup_{(u,v,l) \in E_L} \{(u, v)\}$. According to the definition of triangle-connected $(\mathcal{F}, \lambda, k)$-FoTruss, H' is a triangle-connected k-truss because each edge schema (u, v) is contained in at least $k-2$ triangles, and each edge schema is triangle-connected in H'. So we can denote the diameter of H' as $\{v_1 \to \cdots \to v_d\}$. And we can construct $\mathcal{P}_1' = \{(v_1, l_1) \to \cdots \to (v_d, l_d)\}$, who satisfies $|\mathcal{P}_1'| \leq \lfloor \frac{2(V(H))}{k+1} \rfloor$ according to Theorem 6.

Next, we can construct $\mathcal{P}_2'$ and $\mathcal{P}'$ based on $\mathcal{P}_1'$. Suppose $\mathcal{P}_1^* = \{(v_1, l_1^*) \to (v_2, l_2^*) \to \cdots \to (v_d, l_d^*)\}$ where the layer set $L^* = \{l_1^*, \ldots l_d^*\}$ is the optimal layer set such that $|\bigcup_{l_i \neq l_{i+1}} \{(v_i, l_i), (v_{i+1}, l_{i+1})\}|$ is the smallest. In other words, using the layer set L^*, we can construct the shortest multilayer path $\mathcal{P}'$ between v_1 and v_d. We mark this shortest path as $\mathcal{P}^*$, and the corresponding $\mathcal{P}'_2$ as $\mathcal{P}^*_2$. If $|\mathcal{F}| \geq 1$, we can set all the $l_i \in L^*$ to be $l \in \mathcal{F}$. Thus, $|\mathcal{P}^*_2| = 0$.

If $|\mathcal{F}| = 0$, we can not select a layer for all the layers in L^*. There have to be some inter-layer edges in the path, i.e., $|\mathcal{P}^*_2| \neq 0$. According to the definition of FoTruss, the size of the uphold layer set of each (v_i, v_{i+1}) edge schema is greater than λ. We study the maximum α where uphold layer set of $\{(v_i, v_{i+1}), \ldots, (v_{i+\alpha-1}, v_{i+\alpha})\}$ can share a common layer for the worst case. According to the pigeonhole principle, we have $\alpha = \lfloor \frac{|L|}{|L|-\lambda} \rfloor$. So in the worst case, there can be a continuous sequence of α edge schemas $(v_i, v_{i+1}) \in \mathcal{P}^*_1$ with a layer l such that all (v_i, v_{i+1}, l) exist. Thus, $|\mathcal{P}^*_2| \leq \frac{1}{\alpha}|\mathcal{P}'_1|$.

In summary, we have

$$\begin{aligned} diam(H) &= min_{\mathcal{P} \in H}(|\mathcal{P}|) \leq |\mathcal{P}^*| = |\mathcal{P}^*_1| + |\mathcal{P}^*_2| \\ &= |\mathcal{P}^*_1| \leq \lfloor \frac{2|V(H)|}{k+1} \rfloor \end{aligned}$$

if $|\mathcal{F}| \geq 1$ and

$$\begin{aligned} diam(H) &= min_{\mathcal{P} \in H}(|\mathcal{P}|) \leq |\mathcal{P}^*| = |\mathcal{P}^*_1| + |\mathcal{P}^*_2| \\ &\leq (1 + \frac{1}{\alpha})|\mathcal{P}^*_1| \\ &\leq (1 + \frac{1}{\lfloor \frac{|L|}{|L|-\lambda} \rfloor}) \lfloor \frac{2|V(H)|}{k+1} \rfloor \end{aligned}$$

if $|\mathcal{F}| = 0$. □

A.9 Lemma 1

Proof. Suppose we have two combos $(\mathcal{F}_1, \lambda_1)$ and $(\mathcal{F}_2, \lambda_2)$ where $(\mathcal{F}_1, \lambda_1) \sqsubseteq (\mathcal{F}_2, \lambda_2)$ and $(\mathcal{F}_1, \lambda_1)$ combo is trivial. For any $k > 2$, the $(\mathcal{F}_2, \lambda_2, k)$-FoTruss is a subgraph of $(\mathcal{F}_1, \lambda_1, k)$-FoTruss. Since $(\mathcal{F}_1, \lambda_1)$ combo is trivial, for any $k > 2$ the $(\mathcal{F}_1, \lambda_1, k)$-FoTruss is an empty graph. So for any $k > 2$ the $(\mathcal{F}_2, \lambda_2, k)$-FoTruss is also an empty graph. The $(\mathcal{F}_2, \lambda_2)$-trussness of all edge schemas $(u, v) \in E(G)$ is 2, which means the $(\mathcal{F}_2, \lambda_2)$ combo is a trivial combo. □

A.10 Lemma 2

Proof. Let $k_1 = \tau(\mathcal{F}_1, \lambda_1, (u, v))$ and $k_2 = \tau(\mathcal{F}_2, \lambda_2, (u, v))$. Suppose H_2 is the $(\mathcal{F}_2, \lambda_2, k_2)$-FoTruss. From Property 3, we have $(\mathcal{F}_2, \lambda_2, k_2)$-FoTruss is a subgraph of $(\mathcal{F}_1, \lambda_1, k_2)$-FoTruss becuase $(\mathcal{F}_1, \lambda_1) \sqsubseteq (\mathcal{F}_2, \lambda_2)$. By definition of trussness, k_1 is the largest k such that $(\mathcal{F}_1, \lambda_1, k)$-FoTruss contains (u, v). The trussness k_1 is at least k_2 beacuse $(\mathcal{F}_1, \lambda_1, k_2)$-FoTruss contains (u, v). Thus, the lemma holds. □

A.11 Lemma 3

Proof. For all combo $(\mathcal{F}_2, \lambda_2)$, $\tau(\mathcal{F}_2, \lambda_2, (u, v)) \geq 2$. While according to lemma 2, $\tau(\mathcal{F}_2, \lambda_2, (u, v)) \leq \tau(\mathcal{F}_1, \lambda_1, (u, v))$. Since (u, v) is trivial w.r.t. $(\mathcal{F}_1, \lambda_1)$, $\tau(\mathcal{F}_1, \lambda_1, (u, v)) = 2$. We both have $\tau(\mathcal{F}_2, \lambda_2, (u, v)) \geq 2$ and $\tau(\mathcal{F}_2, \lambda_2, (u, v)) \leq 2$, so $\tau(\mathcal{F}_2, \lambda_2, (u, v)) = 2$, which means (u, v) is trivial w.r.t. $(\mathcal{F}_2, \lambda_2)$. □

A.12 Property 5

Proof. According to the definition, we have

$$\begin{aligned} \Delta &= C^+ - C \\ &= O(|L|\log|L|(\sum_{l \in L}(|E_l(G)| - |E_l(H)|)^{1.5} \\ &\quad + (|E(G)| - |E(H)|)) - \sum_{l \in L}|E_l(H)|^{1.5}) \\ &= O(|L|\log|L|\sum_{l \in L}(|E_l(G)| - |E_l(H)|)^{1.5} - \sum_{l \in L}|E_l(H)|^{1.5}) \\ &= O(|L|\log|L||L|^{1.5}(|E(G)| - |E(H)|)^{1.5} - |L|^{1.5}|E(H)|^{1.5}) \\ &= O(|E(G)|^{1.5}|L|^{1.5}(|L|\log|L|(\frac{|E(G)| - |E(H)|}{|E(G)|})^{1.5} - \\ &\quad (\frac{|E(H)|}{|E(G)|})^{1.5})) \\ &= O(|E(G)|^{1.5}|L|^{1.5}(|L|\log|L|p^{1.5} - (1-p)^{1.5})) \\ &= k_1 p^{1.5} - k_2(1-p)^{1.5} + b \end{aligned}$$

Since k_1, k_2 and b are all greater than 0, Δ is a monotonically increasing function of p, where $p \in [0, 1]$. It is worth noting that our analysis is merely an estimation of the cost, not an exact calculation. This function can be approximated by a line with respect to p within the domain $[0, 1]$. For convenience in calculation, we use linear regression with respect to p to approximate this function during regression analysis. □

A.13 Lemma 4

Proof. (**Irreflexivity.**) Given a k-partial class P, $P \nless P$ because $k \nless k$.

(**Antisymmetry.**) Given a k_1-partial class P_1 and a k_2-partial class P_2. If $P_1 < P_2$, we have $k_1 < k_2$, Thus, $k_2 \nless k_1$, which means $P_2 \nless P_1$.

(**Transitivity.**) Suppose P_1 is a k_1-partial class of a triangle-connected $(\mathcal{F}, \lambda, k_1)$-FoTruss H_1, P_2 is a k_2-partial class of a triangle-connected $(\mathcal{F}, \lambda, k_2)$-FoTruss H_2, and P_3 is a k_3-partial class of a triangle-connected $(\mathcal{F}, \lambda, k_3)$-FoTruss H_3. If $P_1 < P_2$ and $P_2 < P_3$, then $k_1 < k_2 < k_3$. So we have H_2 is a subgraph of H_1, and H_3 is a subgraph of H_2. Meanwhile, $\forall (u_1, v_1) \in P_1, (u_3, v_3) \in P_3$, there exist at least one $(u_2, v_2) \in P_2$, s.t. (u_1, v_1) is triangle-connected to (u_2, v_2) in H_1, and (u_2, v_2) is triangle-connected to (u_3, v_3) in H_2. Since H_2 is a subgraph of H_1, so (u_2, v_2) must be triangle-connected to (u_3, v_3) in H_1. According to the definition of triangle connectivity, (u_1, v_1) is triangle-connected to (u_3, v_3) in H_1. Thus, $P_1 < P_3$.

In summary, since the relationship of containment of partial classes possesses the properties of irreflexivity, antisymmetry, and transitivity, this relationship is a strict partial order. □

A.14 Lemma 5

Proof. We prove by contradiction that the Hasse diagram is a forest. Suppose the Hasse diagram is not a forest. Then there exists a node in the diagram representing a k-partial class P whose indegree is more than 2. Suppose two of them are k_1-partial class P_1 and k_2-partial class P_2.

If $k_1 \neq k_2$, suppose $k_1 < k_2 < k$. Then $\forall (u_1, v_1) \in P_1, (u_2, v_2) \in P_2$, there exists a $(u, v) \in P$ s.t. (u_1, v_1) is triangle-connected to

(u_2, v_2) through (u, v). So there should be $P_1 < P_2 < P$. There should not be a link between P_1 and P according to the definition of the Hasse diagram, which leads to a contradiction.

If $k_1 = k_2 < k$. We have $P_1 < P$ and $P_2 < P$. Then $\forall(u_1, v_1) \in P_1, (u_2, v_2) \in P_2$, there exists a $(u, v) \in P$ s.t. (u_1, v_1) is triangle-connected to (u_2, v_2) through (u, v). So there should be a $P' = P_1 \cup P_2$ that also holds $P' < P$. The existence of P' contradicts the definition of a partial class, where each partial class is a maximal subset of the edge schema set $E(H)$ of the $(\mathcal{F}, \lambda, k)$-FoTruss H.

In summary, since the indegree of all partial classes cannot exceed 2, the Hasse diagram must be a forest. □

A.15 Lemma 6

Proof. We first prove each T_x of T represents a TFC. Suppose x is a k-partial class of $(\mathcal{F}, \lambda)$. For each node x_i in T_x, suppose it is a k_i-partial class. Then a subgraph H is formulated by all the edge schemas containing x_i and x. According to the definition of partial class, we have $\forall(u, v) \in E(H), \tau(\mathcal{F}, \lambda, u, v) \geq k$. Meanwhile, for all triangle schemas $\triangle$ and $\triangle'$ in H, they are triangle-connected because they are linked in T_x. The maximum of H is guaranteed because each partial class is maximal, and edge schemas from other partial classes not in T_x cannot be added into H because they are not triangle-connected. Thus, each T_x of T represents a TFC.

Next, we prove each TFC is represented by a T_x of T. For a TFC subgraph H, the edge schemas in $E(H)$ can be categorized into a series of partial classes according to Definition 9. Each of these partial classes reflects a specific node in the tree. Otherwise, the fact that TFC is the maximal subgraph or that the partial class is the largest subset would be violated. Since a TFC is triangle-connected, these partial classes can form a tree T'. This tree is a subtree of T because the tree nodes not in the subtree are not triangle-connected to $E(H)$, while T' can not lose nodes in the subtree because H is the maximal subgraph. Thus, each TFC is represented by a T_x of T. □

A.16 Theorem 5

Proof. Each edge schema is contained in only one tree node T_x. Suppose there is an edge schema $(u, v) \in E(G)$ that is contained in two tree nodes T_{x1} and T_{x2}. We have the triangle-connected components of T_{x1} and T_{x2} should be triangle-connected by edge schema (u, v), so x_1 and x_2 should be the same tree node. So the number of all the tree nodes $|T_x|$ and the sum of all $|E(T_x)|$ are both smaller than $|E(G)|$.

In the worst case, each edge schema is a partial class. Each partial class is a node in the index, so the links between nodes can not exceed the links in the minimum spanning tree of all edges in $E(G)$. So the sum of both $|P(T_x)|$ and $|C(T_x)|$ should both be smaller than $|E(G)|$.

In summary, the size of the index tree should not exceed $O(|E(G)|)$. □

B COMPLEXITIES

B.1 Complexity of the Decomposition Framework (Section 3.1)

For the initialization before the peeling, sup can be calculated once for all combos, counting triangles for each edge will take $O(\sum_{l \in L} |E_l|^{1.5})$ [49]. While it takes $O(|E||L| \log |L|)$ to initialize $\tau(u, v)$ for each combo. The running time of Algorithm 2 is dominated by the time for updating the trussness using Algorithm 1 in lines 9 and 12. During the peeling, each edge $(u, v, l) \in E_L$ is removed only once. For each edge (u, v, l) removed by line 8 and 11, each common neighbor $w \in N_l(u) \cap N_l(v)$ may trigger an update using Algorithm 1. Suppose we have $deg(u, l) \leq deg(v, l)$. Then each common neighbor $w \in N_l(u) \cap N_l(v)$ can be found by searching if each $w \in N_l(u)$ has an edge $(v, w, l) \in E_l$. Let $N_l^{\geq}(u)$ be $\{v | v \in N_l(u), deg(v, l) \geq deg(u, l)\}$. We have $|N_l^{\geq}(u)| \leq 2\sqrt{|E_l|}$ for any $u \in V$ [49]. Thus, the loop in lines 6–13 is executed for at most $\sum_{l \in L} \sum_{u \in V} (deg(u, l) \cdot |N_l^{\geq}(u)|) = O(\sum_{l \in L} |E_l|^{1.5})$ times. For the worst case, each loop needs to update the trussness, thus the loop in lines 5–13 takes $O(|L| \log |L| \sum_{l \in L} |E_l|^{1.5})$ time. Putting all together, the time complexity of Algorithm 2 is $O(|L| \log |L| \cdot (|E| + \sum_{l \in L} |E_l|^{1.5}))$. For all the combos, according to equation 2, the time complexity is $O(2^{|L|} |L|^2 \log |L| \cdot (|E| + \sum_{l \in L} |E_l|^{1.5}))$.

B.2 Complexity of TFCS-Online (Algorithm 5)

Algorithm 5 is dominated by Line 7, which enumerates all the triangles once in all the triangle-connected components containing v_q. Similar to the analysis in Section 3.1, the enumeration will take $O(|E(H)|^{1.5})$, where H is the subgraph induced by all the triangle-connected components.

B.3 Complexity of FoTruss Index Construction (Algorithm 6)

The complexity of the peeling process (Line 1) is analyzed in Section 3. The rest part of Algorithm 6 is dominated by the UF operations. Each triangle will be enumerated only once in Line 7 and Line 11, which will both cost $O(|E(G)|^{1.5})$ time. For each enumeration, three UF operations are followed (Lines 8–9, Line 12, and Line 19), which will each cost $O(\alpha(|E(G)|))$ for a single operation ($\alpha()$ is the inverse Ackermann function) [7]. So Lines 6–12 and Lines 18–21 will cost $O(|E(G)|^{1.5} \alpha(|E(G)|))$ time in total. Each edge schema will be enumerated once in Line 13, so Lines 13–17 will cost $O(|E(G)|)$ time. Thus, the overall complexity is $O(|E(G)|^{1.5} \alpha(|E(G)|))$.

C ALGORITHMS

C.1 Query-Driven Index

As shown in Algorithm 8, assume $\mathcal{T}$ is the set of returned indices with budget b. We use an arbitrary `Metric()` to measure the performance of a combo on historical queries (Line 4). This metric can be the density or diameter of the multilayer graph, or any other defined community evaluation metric. We first fill $\mathcal{T}$ with the indices corresponding to the first b enumerated combos (Lines 5–7). When the number of indices in $\mathcal{T}$ reaches b, we compare the combo that performs the worst on historical queries in $\mathcal{T}$ with the currently enumerated combo and keep the one that performs better

Algorithm 8: `BuildQueryDrivenIndex`

Input: A multilayer graph $G = (V, L, E_L)$, a maximum index budget b, a set of sampled queries Q, and a metric function to evaluate a community `Metric()`.
Output: A set of FoTruss indices $\mathcal{T}$.

1 $\mathcal{T} \leftarrow \{\}$; $qu \leftarrow \{\}$; qu.enqueue$((\emptyset, 1))$;
2 **for** $(\mathcal{F}, \lambda) \leftarrow qu$.dequeue() **do**
3     $T_{(\mathcal{F},\lambda)} \leftarrow$ `BuildIndex`$(G, \mathcal{F}, \lambda)$;
4     $AM_{(\mathcal{F},\lambda)} \leftarrow$ average `Metric`(v_q) for $v_q \in Q$;
5   **if** $|\mathcal{T}| < b$ **then**
6         $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{(\mathcal{F},\lambda)}\}$;
7   **for** $(\mathcal{F}', \lambda') \in$ `enumChildCombos`$(\mathcal{F}, \lambda)$ **do** qu.enqueue$((\mathcal{F}', \lambda'))$;
8   **else**
9   Find index of $(\mathcal{F}', \lambda')$ in $\mathcal{T}$ whose $AM_{(\mathcal{F},\lambda)}$ is worst;
10   **if** $AM_{(\mathcal{F}',\lambda')}$ *is worse than* $AM_{(\mathcal{F},\lambda)}$ **then**
11   Replace $T_{(\mathcal{F}',\lambda')}$ by $T_{(\mathcal{F},\lambda)}$;
12   **for** $(\mathcal{F}', \lambda') \in$ `enumChildCombos`$(\mathcal{F}, \lambda)$ **do** qu.enqueue$((\mathcal{F}', \lambda'))$;
13 **return** $\mathcal{T}$;

on historical queries (Lines 8–12). Only if an enumerated combo replaces a previous combo, combos that are stricter than it will be enumerated (Line 12).

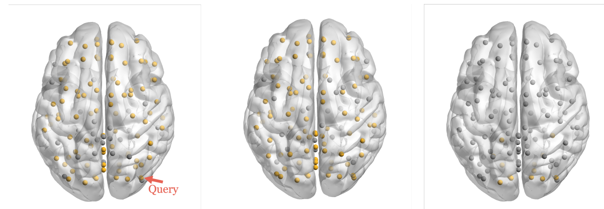
D ADDITIONAL EXPERIMENTS

D.1 Case Study on Brain Networks

We use the brain network derived from a previous work [32]. These brain networks focus on resting-state functional MRI (rs-fMRI) data, which measures brain activity by detecting changes in blood flow and oxygenation. The temporal coherence in blood fluctuations suggests the brain functions as a network, with regions as vertices and functional connectivity as edges. Given the large volume of MRI data, voxel aggregation, and various preprocessing strategies are used to reduce dimensionality and minimize noise.

The dataset adopts the AAL atlas which divides the brain into 116 ROIs (Region of Interest). For each patient, the dataset formulates an undirected unweighted graph of 116 vertices. Each edge (u, v) means for the individual, the two regions u and v correlate. We sampled 10 individuals from the datasets to form a multilayer graph. It should be pointed out that such signals are heavily subject to noise caused by different confounding factors.

Detecting functional systems in the brain is an important task in neuroscience. The previous works [3, 39] use multilayer graphs to avoid these noises. However, these methods cannot study the functional brain regions of interested individuals. They can only investigate the common functional regions across multiple individuals. In the brain, the Occipital Pole, located at the extreme posterior end of the cerebral hemisphere, near the back of the skull, is responsible for visual processing. We selected a query vertex from the Occipital Pole, as shown in the figure 14(a). As shown in figure 14(b), when performing a truss community search on a single layer, the result includes the entire brain rather than being limited to the Occipital Pole due to the presence of noise in individual fMRI. In contrast, TFCS effectively completes this task. We can set the focus layer to be the layer formulated by the corresponding individual of interest, and set $\lambda = 2$, meaning we want to use the region connections from other individuals to filter the noise. As shown in figure 14(c), the community identified by TFCS lies in the Occipital Pole. When using the same parameters as TFCS, the community detected by FirmTruss fails to target the individual of interest. Meanwhile, the result also includes the entire brain rather than being limited to the Occipital Pole.



(a) FirmTruss. (b) Truss on a single layer. (c) TFCS.

Figure 14: Detected functional systems.