Efficient Maximal Frequent Clique Enumeration in Multilayer Networks

Han Wang[1], Renjie Sun[2], Yongye Li[1(✉)], Chen Chen[3], Xiaoyang Wang[4], and Ying Zhang[1]

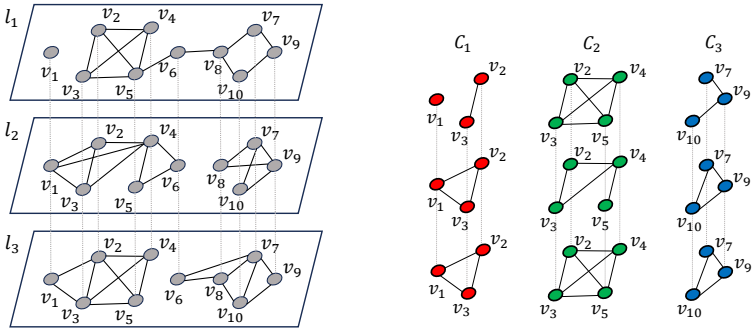
[1] Zhejiang Gongshang University, China
{hanw.zjgsu,yongyeli.zj}@gmail.com, ying.zhang@uts.edu.au
[2] East China Normal University, China
renjie.sun@stu.ecnu.edu.cn
[3] University of Wollongong, Australia
chenc@uow.edu.au
[4] University of New South Wales, Australia
xiaoyang.wang1@unsw.edu.au

Abstract. Clique as a fundamental model within graph theory, has been extensively studied in various single-layer graphs. However, in multilayer (ML) graphs, which provide a more expressive representational framework, research in this area remains relatively limited. In this paper, we propose a novel model, named (k,λ)-frequent clique ((k,λ)-FC), designed to capture complex patterns of interactions that across various domains in ML graph. Given a ML graph G, a node set H is a (k, λ)-FC if i) $|H| \geq k$, and ii) H is a clique in at least λ layers. We aim to enumerate all the maximal (k, λ)-FCs, which is proved to be NP-hard. To tackle the problem, we introduce the concept of the projection graph and develop a merged-based search method based on it. To further enhance efficiency, we implement optimizations in graph reduction, branch pruning, and intersection acceleration. Extensive experiments on 10 real-world ML graphs are conducted to demonstrate the efficiency and effectiveness of the proposed model and techniques.

Keywords: Multilayer graph · Clique · Frequency · Graph mining

1 Introduction

In graph theory, a clique is defined as a subset of nodes where every pair of distinct nodes is adjacent [6,22]. A clique is termed maximal if it cannot be expanded by any other nodes. Enumerating all such maximal cliques within a graph [2] is crucial for uncovering densely connected substructures that often carry significant insights in network analysis [26]. The multilayer (ML) graph [11] is a critical data structure that facilitates the simultaneous representation of multiple relationships, making it particularly apt for modeling a wide range of real-world. For instance, routes from several airlines in air-transportation network [5] and various tissue types or environments in gene co-expression networks [13]. While



(a) Multilayer graph (b) Frequent cliques

Fig. 1. A multilayer graph with 3 layers and the corresponding $(3, 2)$-frequent cliques

maximal clique enumeration (MCE) has been extensively studied within single-layer graphs [1,15,20], its application to ML graphs remains largely unexplored.

Integrating MCE into ML graphs presents a significant challenge. An intuitive strategy is to divide the ML graph into several independent layers and perform MCE on each layer separately. However, the approach abandons the joint search properties of ML graph. Therefore, it is necessary to propose a targeted clique-based model which is compatible with both. To the best of our knowledge, current ML models are primarily based on classical k-core, k-truss, and γ-quasi-clique. Specifically, the k-core based include Firmcore [10] and multi-layer core [8,14,27]; k-truss based model has FirmTruss [3]; and γ-quasi-clique based models, namely cross-layer quasi-clique [16]. However, the clique has a unique structure, unlike other dense subgraph models, which allow multiple forms in different layers. Such difference makes it challenging to draw inspiration from prior literature. Furthermore, though existing ML models perform well in capturing dense structures in ML graphs, they do not account for the stability of these structures. Take Firmcore as an example in Fig. 1(a). The node set $S = \{v_1, v_2, v_3, v_4, v_5\}$ forms a $(2, 2)$-Firmcore. However, the link (v_1, v_5) is not stable, as it is not supported by any layer.

To address the above limitations, we introduce the concept of frequency from subgraph pattern search [28] and propose a novel model named (k, λ)-frequent clique ((k, λ)-FC) in ML graphs. Specifically, given a size constraint k and a frequency constraint λ, (k, λ)-FC is a complete subgraph in at least λ layers. (k, λ)-FC ensures that each pair of nodes is reliably connected, as every pair of nodes is supported by at least λ types of links. For example, C_1, C_2 and C_3 in Fig. 1(b) are three $(3, 2)$-FCs since their sizes are at least 3, and each forms a clique in 2 layers. The following are two representative applications.

Community formation in social networks. Taking users with accounts on multiple social channels, such as Facebook, Instagram, and Twitter, for instance, a group forming a community exclusively on Facebook may be coincidental. In contrast, if a group establishes communities on both Facebook and Instagram, it is more likely to evolve into a closely connected community on Twitter in the future. In general, frequent groups feature greater stability, being less prone to

dissolve on social media where they initially formed communities and more likely to form communities on platforms where they have not yet done so.

Biological networks analysis. In gene interaction networks, frequent cliques represent genomes that form complete graphs across various levels, such as metabolism, regulation, and physical interaction. Identifying these genomes is crucial for pinpointing core nodes in the biological network, offering valuable insights into the network's topological structure. Moreover, genomes exhibiting multilayer complete graphs demonstrate stability across multiple relationships. This inherent robustness enhances their ability to adapt to dynamic environmental changes and respond to signal disturbances effectively.

Challenges and Contributions. In this paper, we study the maximal (k, λ)-frequent clique enumeration over ML graphs. To the best of our knowledge, there is currently no method that can address the problem. The main challenges of the proposed problem lie in the following aspects. Firstly, we prove that our problem is NP-hard, implying it is non-trivial to enumerate all the (k, λ)-FCs. Secondly, the efficiency of algorithms on large graphs is significantly hampered by numerous invalid nodes and search branches. The main contributions are summarized below.

- In this paper, we first propose the (k, λ)-frequent clique model on ML graphs, and formally define the problem of maximal (k, λ)-frequent clique enumeration, which is proven to be NP-hard.
- To solve the problem, a baseline algorithm is first constructed. Considering the redundant operations of the baseline algorithm, we merge ML graphs as projection graphs and propose a merge search algorithm based on it.
- To further scale for dense and large ML graphs, we develop the three optimization techniques: *i*) graph reduction before the exhaustive search, *ii*) search branch pruning optimization, and *iii*) intersection acceleration.
- Comprehensive experiments are conducted over 10 real-world ML datasets to verify the advantages of the proposed techniques and model.

2 Preliminaries

2.1 Problem definition

We consider a multilayer (ML) graph $G = (V, E, L)$, where V is the set of nodes, L is the set of layers, and $E = \{E_1, E_2, \cdots, E_{|L|}\}$, each E_ℓ representing the set of edges in layer ℓ. (u, v, ℓ) denote an edge between nodes u and v in E_ℓ. The set of neighbors of node $v \in V$ in layer $\ell \in L$ is denoted as $N_\ell(v)$ and the degree of v in layer ℓ is $deg_\ell(v) = |N_\ell(v)|$. For a set of nodes $S \subseteq V$, $G[S] = (S, E[S])$ denotes the subgraph of G induced by S. In layer ℓ, the subgraph induced by S is represented as $G_\ell[S] = (S, E_\ell[S])$, and the degree of node v within this subgraph is denoted by $deg_\ell^S(v)$. For notation simplicity, we use G_ℓ to replace $G_\ell[V]$.

Definition 1 (k-clique). *Given a single-layer graph G_ℓ and an integer k, a k-clique C ($|C| \geq k$) is a subset of V such that $G_\ell[C]$ is a complete graph.*

A k-clique C is *maximal* if we cannot find a larger k-clique C' such that $C \subsetneq C'$. In ML graphs, the frequency of a subgraph is the number of layers in which the subgraph appears. In this paper, we use $SL(H)$ to denote the specific layers where the subgraph H exists.

Definition 2 (Support layers). *Given an ML graph $G = (V, E, L)$ and a subgraph H, the support layers of H, denoted by $SL(H)$, is a subset of L such that H is a subgraph of G_ℓ for each layer $\ell \in SL(H)$.*

We define $SL(H) = L$ if H is an empty subgraph or contains only one node and refer to $|SL(H)|$ as the frequency of H. For an edge e in G, $SL(e) = \{\ell \in L | e \in E_\ell\}$. A reliable cohesive community in ML graphs should be densely connected (i.e., k-clique) and free from contingency (i.e., support layers). Thus, we propose the novel model, called (k, λ)-frequent clique, by incorporating frequency into dense subgraph.

Definition 3 ((k, λ)-frequent clique ((k, λ)-FC)). *Given an ML graph $G = (V, E, L)$ and two positive integers k and λ ($1 \leq \lambda \leq |L|$), a subset $H \subseteq V$ is a (k, λ)-FC if H is a k-clique in at least λ layers (i.e., $|SL(H)| \geq \lambda$).*

A (k, λ)-FC H is *maximal* if none of the proper supersets of H that is a (k, λ)-FC. In addition to providing flexible frequency guarantees, referring to the definition of ML diameter in [3], the diameter of FC is always 1, which makes it avoid the free-rider [25] effect existing in community search.

Problem statement. Given an ML graph $G = (V, E, L)$ and two positive integers k and λ, in this paper, we aim to develop an efficient algorithm to enumerate all the maximal (k, λ)-FCs.

Example 1. Reconsider the ML graph G in Fig. 1(a). Suppose $k = 3$ and $\lambda = 2$, there are three maximal $(3, 2)$-FCs in G, which illustrated in Fig. 1(b), i.e., $C_1 = \{v_1, v_2, v_3\}$ with 2 support layers $SL(C_1) = \{l_2, l_3\}$, $C_2 = \{v_2, v_3, v_4, v_5\}$ with 2 support layers $SL(C_2) = \{l_1, l_3\}$ and $C_3 = \{v_7, v_9, v_{10}\}$ with 2 support layers $SL(C_3) = \{l_2, l_3\}$.

2.2 Problem properties

Theorem 1. *Given a (k, λ)-FC H of G, any subgraph H' of H with $|H'| \geq k$ is also (k, λ)-FC of G.*

Proof. Since H is a k-clique in each layer within $SL(H)$, all nodes in H are fully connected in these layers. Consequently, any subgraph $H' \subseteq H$ with $|H'| \geq k$ will inherit this fully connected structure in each layer of $SL(H)$. Thus, H' itself forms a k-clique in each of these layers, satisfying the required conditions.

Theorem 2. *The maximal (k, λ)-FC enumeration problem is NP-hard.*

Algorithm 1: Baseline algortihm

Input : $G = (V, E, L)$: ML graph, k: size constraint, λ: frequency constraint
Output : $\mathcal{R}$: all the maximal (k, λ)-FCs

1 $\mathcal{R} \leftarrow \varnothing, \mathcal{P} \leftarrow \varnothing$;
2 **for each** $\ell \in L$ **do**
3 $\quad \mathcal{P}_c \leftarrow$ enumerate all the maximal k-cliques from G_ℓ;
4 $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_c$;
5 $\mathcal{P} \leftarrow$ remove non-maximal k-cliques from $\mathcal{P}$;
6 **for each** $P \in \mathcal{P}$ **do**
7 $\quad$ **if** $|SL(P)| \geq \lambda$ **then**
8 $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{P\}$;
9 $\quad$ **else**
10 $\quad\quad \mathcal{R}_c \leftarrow$ enumerate all the maximal (k, λ)-FC within P;
11 $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_c$;
12 $\mathcal{R} \leftarrow$ remove non-maximal (k, λ)-FCs from $\mathcal{R}$;
13 **return** $\mathcal{R}$;

Proof. When the minimum support layer $\lambda = 1$, the maximal frequent clique enumeration problem degenerates to the maximal clique enumeration problem [2,4] in single-layer graph G_ℓ on each layer $\ell \in L$, which is NP-hard. Moreover, when $\lambda > 1$ and $|L| > 2$, the maximal frequent clique enumeration problem is more complex than the former situation. This is because the corresponding solution must simultaneously satisfy both size and frequency constraints. Therefore, the problem proposed in this paper is also NP-hard.

3 Baseline

In this section, we propose a baseline algorithm to address our research problem, analyze its time complexity and discuss its limitations.

Baseline algorithm. It is easy to observe that any maximal (k, λ)-FC must be contained in a maximal k-clique. Thus, we propose a baseline method to enumerate all the maximal (k, λ)-FCs within ML graphs, which first identify all maximal k-cliques within each layer, and then extract all maximal (k, λ)-FCs within it. The pseudocode of the baseline method is shown in Algorithm 1. Firstly, we enumerate all the maximal k-cliques for each layer $\ell \in L$, and store them in $\mathcal{P}$ (lines 2-4). Given that the enumeration procedure is executed independently across each layer, it is necessary to filter out non-maximal k-cliques from $\mathcal{P}$ (line 5). For each maximal k-clique P in $\mathcal{P}$, if the number of its support layers no less than λ, P is a (k, λ)-FC (lines 7-8). Otherwise, we enumerate all the subset of P to extract all the maximal (k, λ)-FCs within P, and store them in $\mathcal{R}$ (lines 10-11). Finally, all the maximal (k, λ)-FCs can be obtained by eliminating the non-maximal results (line 12).

Time complexity. The time comlexity of Algorithm 1 is bounded in $O(|L||V|3^{|V|/3} + \sum_{i=1}^{c} p_i \cdot 2^{p_i} + t^2 \cdot z) \approx O(|L||V|3^{|V|/3} + \sum_{i=1}^{c} p_i \cdot 2^{p_i})$. Algorithm 1 first enumerates all maximal k-cliques on each layer in $O(|L|3^{|V|/3})$. Enumer-

Algorithm 2: Merge search algorithm

Input : $G = (V, E, L)$: ML graph, k: size constraint, λ: frequency constraint
Output : $\mathcal{R}$: all the maximal (k, λ)-FCs

$\mathcal{R} \leftarrow \varnothing$;
Enum$(\varnothing, V, \varnothing, L)$;
return $\mathcal{R}$;
Function Enum$(R, P, X, \mathcal{L})$
if $P = \varnothing$ **then**
  if $X = \varnothing$ **and** $|R| \geq k$ **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{R\}$;
  return;
for each $v \in P$ **do**
  $P' \leftarrow P \cap N(v)$;
  for each $u \in P'$ **do**
    if $|SL(R \cup \{v, u\})| < \lambda$ **then**
      $P' \leftarrow P' \setminus \{u\}$;
  get X' similar as P';
  Enum$(R \cup \{v\}, P', X', SL(R \cup \{v\})$;
  $P \leftarrow P \backslash \{v\}; X \leftarrow X \cup \{v\}$;

ating maximal (k, λ)-FCs from each maximal k-cliques runs in $O(\sum_{i=1}^{c} p_i \cdot 2^{p_i})$ where c is the number of maximal k-cliques and p_i is the subset of one of maximal k-clique. For maximality check, it runs in $O(t^2 \cdot z)$ where t and z are the number of (k, λ)-FCs and the maximum number in each (k, λ)-FC, and we have $t^2 \cdot z \ll |L||V|3^{|V|/3}$ or $t^2 \cdot z \ll \sum_{i=1}^{c} p_i \cdot 2^{p_i}$.

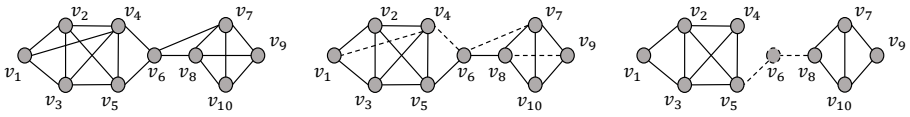
Limitations. Although the baseline algorithm can find all the maximal (k, λ)-FCs within an ML graph, it is incapable of processing real-world networks within a reasonable time. This is because enumerating all the maximal k-cliques is NP-hard [9], and there could be a large number of maximal k-cliques in an ML graph. In addition, it is also challenging to find all the maximal (k, λ)-FCs within a k-clique, and check whether a (k, λ)-FC is maximal.

4 Optimization

4.1 Merge search algorithm

Given the redundancy generated by the baseline algorithm's sequential approach of enumerating k-cliques and subsequently accessing their frequency, we propose a merge search algorithm that integrates the two steps to enhance efficiency. This method simultaneously enumerates k-cliques and evaluates their frequency adherence during the enumeration. To realize this approach, we introduce the concept of the projection graph.

Definition 4 (Projection graph). *Given an ML graph $G = (V, E, L)$, the projection graph of G is denoted by $\mathcal{G} = (V, \mathcal{E})$, where $\mathcal{E} = \{(u, v) \mid \exists \ell \in L$ s.t. $(u, v, \ell) \in E_\ell\}$, which represents the union of all layers in G.*



(a) Projection graph $\mathcal{G}$ (b) Edge pruning (c) Node pruning

Fig. 2. Running example for graph reduction

Merge search algorithm. Let $N(v)$ denote the set of nodes adjacent to v in the projection graph $\mathcal{G}$. Algorithm 2 shows the details of the proposed merge search method. We initialize $\mathcal{R}$ as empty and invoke the recursive procedure Enum to enumerate all the maximal (k, λ)-FCs (lines 1-2). Enum maintains four sets R, P, X and $\mathcal{L}$, where R is the temporary result set, P is the set of possible candidate nodes to expand R, X is the excluded set to make sure the returned result is maximal, and $\mathcal{L}$ is the support layer set of R. At the beginning of each recursion, if P and X are both empty, and the size of R is no smaller than k, R is a maximal (k, λ)-FC (lines 5-8). To expand R with the nodes in P, we iteratively add a new node v from P to R and obtain P' by removing the non-neighbor of v from P (line 10). Then, the node $u \in P'$ with $|SL(R \cup \{v, u\})|$ smaller than λ can also be removed since u cannot form a (k, λ)-FC with $R \cup \{v\}$ (lines 11-13). A similar process is used to derive X' (line 14). In line 15, we use the new parameters to enter the recursion and repeat the process until P is empty. When that recursion ends we move the currently visited node v from P to X.

4.2 Graph reduction method

In the ML graph, many unpromising nodes and edges cannot exist in any (k, λ)-FCs. Therefore, we propose two pruning rules to filter those unpromising nodes and edges from the search space.

Edge pruning. Based on the definition of (k, λ)-FC, it is easily obtained that the frequency of each edge e in a (k, λ)-FCs must be no less than λ, i.e., $SL(e) \geq \lambda$. Hence, we can safely remove the edges e from all layers of the ML graph if $|SL(e)| < \lambda$. Then, we derive the following lemma. Due to the simplicity and clarity of this lemma, we omit the proof of it.

Lemma 1. *Given an ML graph G and two positive integers k and λ, an edge e cannot belong to any (k, λ)-FCs of G if $|SL(e)| < \lambda$.*

Node pruning. Given an ML graph G, many unpromising nodes are incapable of existing in any (k, λ)-FCs, especially after losing connections formed by non-frequent edges. We can filter such nodes by preserving the $(k-1)$-core of the corresponding projection graph $\mathcal{G}$ based on the fact that k-cliques are included in $(k-1)$-core [17,7]. The concept of k-core is defined as follows.

Definition 5 (k-core). *Given a single-layer graph G_ℓ and an integer k, the k-core of G_ℓ denoted by C_k^ℓ is a maximal subgraph in which every node has no least than k neighbors.*

Algorithm 3: GR(G, k, λ)

```
   Input  : G = (V, E, L): ML graph, k: size constraint, λ: frequency constraint
   Output : Updated ML graph
 1 remove all edges with |SL(e)| < λ from each layer;
 2 for each ℓ ∈ L do
 3     compute the k-core C_k^ℓ in G_ℓ;
 4     for each v ∈ C_k^ℓ do
 5         CS_k(v) ← CS_k(v) ∪ {ℓ};
 6 Q ← all nodes with |CS_k(·)| < λ;
 7 while Q ≠ ∅ do
 8     v ← Q.pop();
 9     for each ℓ ∈ CS_k(v) do
10         for u ∈ N_ℓ(v) and u ∉ Q do
11             deg_ℓ(u) ← deg_ℓ(u) − 1;
12             if deg_ℓ(u) = k − 1 then
13                 CS_k(u) ← CS_k(u) \ {ℓ};
14                 if |CS_k(u)| < λ then
15                     Q.push(u);
16     remove v from G;
```

While using k-core on the projection graph to filter nodes is effective, it does not take into account the frequency requirement. Hence we propose a novel structure to tightly filter the search space. We first introduce the concept of (k, λ)-set, and the details of the pruning method based on (k, λ)-set are presented in the subsequent lemma.

Definition 6 ((k, λ)-set). *Given an ML graph $G = (V, E, L)$ and two integers k and λ, a subset $C \subseteq V$ is (k, λ)-set if it satisfies the following conditions: i) every node $v \in C$ belongs to the k-core on at least λ layers, and ii) there is no superset of C satisfies i).*

Lemma 2. *Given an ML graph $G = (V, E, L)$ and two integers k and λ, every maximal (k, λ)-FC must be the subset of (k-1, λ)-set.*

Proof. From the definition of (k, λ)-FC, we can infer that any (k, λ)-FC is $(k-1)$-core on at least λ layers. This implies that every node in (k, λ)-FCs belongs to the $(k-1)$-core on at least λ layers, thereby satisfying the $(k-1, \lambda)$-set constraint for nodes. Hence, the lemma holds.

The details of graph reduction with edge and node pruning are shown in Algorithm 3. All edges with $|SL(e)| < \lambda$ are first removed from the ML graph by edge pruning (line 1). Then, we extract the (k, λ)-set on the remaining graph (lines 2-16). Specifically, we compute the k-core C_k^ℓ for each single-layer graph G_ℓ (line 3). If node v exists in a C_k^ℓ, we use $CS_k(v)$ to store the corresponding layer ℓ (lines 4-5). A queue $\mathcal{Q}$ is used to maintain all nodes with $|CS_k(\cdot)|$ smaller than λ (line 6). Then we iteratively delete the nodes in $\mathcal{Q}$ and update $CS_k(\cdot)$

for other nodes until all the remaining nodes are contained in at least λ k-cores (lines 7-16). Specifically, for each popped node v, we iterative every layer ℓ in $CS_k(v)$, and decrease the degree of v's neighbor u in G_ℓ by 1 (lines 8-11). If the updated degree of u equals $k-1$, the corresponding layer ℓ is removed from $CS_k(u)$ (lines 12-13). Furthermore, if the size of $CS_k(u)$ falls below λ, u is added to $\mathcal{Q}$ for further processing (lines 14-15).

Example 2. Suppose $k = 4$ and $\lambda = 2$, we illustrate the graph reduction process on the projection graph. *i*) Edge pruning: In Fig. 2(b), the edges represented by dashed lines are pruned since their frequencies are 1. *ii*) Node pruning: This step is implemented on the basis of *i*), allowing further nodes to be filtered. The k-core of each layer is computed iteratively, and we obtain $|CS_k(v_6)| = 0 < 2$, leading to the inclusion of v_6 in $\mathcal{Q}$. Then v_6 is removed and the graph is updated. At this point, no residual nodes violate the definition of (k, λ)-set, marking the end of the node pruning phase. In this process, unpromising nodes and their connected edges are represented by dashed lines, as shown in Fig. 2(c).

Time complexity. The overall time cost of Algorithm 3 stands at $O(|\mathcal{E}| + |L||V| + |E|)$. Because we can filter unpromising edges in $O(|\mathcal{E}|)$ and compute per-layer k-cores in $O(|L||V| + |E|)$. In the worst case, every node will need to be deleted, and each time a node is deleted, its related information in all layers will be updated, which can be completed in $O(|V| + |E|)$.

4.3 Branch pruning method

To further enhance the performance of Algorithm 2, we propose two techniques specifically aimed at reducing search branches.

Heuristic search order. Given two branches $B(R, P, X, \mathcal{L})$ and $B'(R', P', X', \mathcal{L}')$ in Algorithm 2, where B' is derived from B by moving a node from P to R, the choice of the moved node determines the size P'. In general, a smaller $|P'|$ leads to better algorithm performance, as it results in fewer direct sub-branches derived from B' and a shallower recursion depth starting from B'. For nodes that would result in a larger $|P'|$, we can defer their processing since some of their neighbors will be moved to X'. But, how can we identify nodes that are suitable for prioritized processing? Before the details of heuristic search order, we first introduce the concept of aggregated degree to evaluate nodes.

Definition 7 (Aggregated degree). *The aggregated degree $Deg(v)$ of a node v in an ML graph is defined as the sum of its degrees across all layers, i.e., $Deg(v) = \sum_{\ell \in L} deg_\ell(v)$.*

A larger aggregated degree indicates a weaker pruning capability of the node on the candidate set P. Therefore, the heuristic search order is defined by ordering the nodes in non-decreasing aggregated degree, i.e., larger aggregated degree nodes are positioned later in the sequence.

Pivot-based pruning. The pivot technique [1] is widely used in the algorithms for clique enumeration on normal graphs, which prunes the search branch by

avoiding the processing of the neighbors of the selected pivot node. However, applying this technique directly to our problem presents some challenges. Specifically, the neighbors of the pivot on the projected graph may not be adjacent to the pivot at every layer in $\mathcal{L}$.

By observing the search process of maximal (k, λ)-FC in Algorithm 2, we find that $|R|$ gradually increases while $|\mathcal{L}|$ decreases with increasing recursion depth. When $|\mathcal{L}| = \lambda$, $\mathcal{L}$ remains constant, only R changes. At this point, the support layers of any (k, λ)-FCs containing R are $\mathcal{L}$. Thus, we can remove each edge e in the search space with $\mathcal{L} \not\subseteq SL(e)$ and then use the pivot technique on the remaining graph.

4.4 Intersection acceleration method

In lines 12 and 15 of Algorithm 2, to obtain the new support layers, we need to compute the intersection between $SL(R)$ and $SL(e)$ for each newly added edge e. The time complexity of the intersection operation for two ordered sets is $O(|\mathcal{L}_1| + |\mathcal{L}_2|)$, where $\mathcal{L}_1$ and $\mathcal{L}_2$ are two layer sets. Therefore, the intersection operation is time-consuming when the ML graph has many layers or high density.

To address this issue, we introduce the bitwise "AND" operation as an alternative to set intersection. For a given layer set $\mathcal{L} \subseteq L$, we first encode $\mathcal{L}$ into a binary with $|L|$ bit: if $l_i \in \mathcal{L}$, the i-th bit is set to 1; otherwise, it is set to 0. To enable efficient bitwise "AND" operations, the binary encoding is divided into blocks of size $\mathcal{X}$, resulting in $\lceil |L|/\mathcal{X} \rceil$ blocks. Each block is then converted into a $2^{\mathcal{X}}$-ary representation, producing a bitmap denoted as $BM(\mathcal{L})$. Based on this, bitmap-based intersection is performed as described in Equation 1 with a time complexity of $O(\lceil |L|/\mathcal{X} \rceil)$.

$$BM(\mathcal{L}_1 \cap \mathcal{L}_2) = \{a_i \& b_i \mid a_i \in BM(\mathcal{L}_1),\ b_i \in BM(\mathcal{L}_2)\}, \tag{1}$$

where $1 \leq i \leq \lceil |L|/\mathcal{X} \rceil$.

Example 3. Suppose there are two layer subsets $\mathcal{L}_1 = \{l_1, l_2\}$ and $\mathcal{L}_2 = \{l_2, l_3\}$ of $L = \{l_1, l_2, l_3\}$, and $\mathcal{X} = 2$. The binary of $\mathcal{L}_1$ and $\mathcal{L}_2$ are $\{0(0), 3(11)\}$ and $\{1(1), 2(10)\}$, respectively, where each block correspond to a $2^{\mathcal{X}}$-ray value. Performing the bitwise "AND" operation block by block, the result of $BM(\mathcal{L}_1 \cap \mathcal{L}_2)$ is $\{0(0\&1), 2(3\&2)\}$.

5 Experiments

5.1 Experiment Setup

Algorithms. To the best of our knowledge, there is no existing research on enumerating all maximal (k, λ)-FCs within ML graphs. To evaluate the effectiveness of different modules, the following algorithms are implemented and evaluated in our experiments.

– BASE: the baseline algorithm proposed in Section 3 (Algorithm 1).

Table 1. Statistics of datasets

Dataset	$\|V\|$	$\|E\|$	$\|\mathcal{E}\|$	$\|L\|$	**Domin**	(k, λ)
D1 (RM)	91	14,289	2,906	10	Financial	(13,4)
D2 (Homo)	18,222	153,922	137,659	7	Genetic	(7,2)
D3 (Sacchcere)	6,570	247,152	223,542	7	Genetic	(7,2)
D4 (FAO)	214	268,339	9,420	364	Co-purchasing	(8,80)
D5 (Brain)	190	933,920	17,955	520	Brain	(6,60)
D6 (DBLP)	513,629	1,015,808	888,353	10	Co-authorship	(7,2)
D7 (Wiki)	1,140,149	3,025,079	2,787,967	24	Social	(5,4)
D8 (Yeast)	4,458	8,473,997	8,327,981	4	Genetic	(8,2)
D9 (Higgs)	456,631	13,706,153	13,094,377	4	Social	(8,2)
D10 (Stack)	2,601,977	63,497,050	36,233,450	24	Social	(8,5)

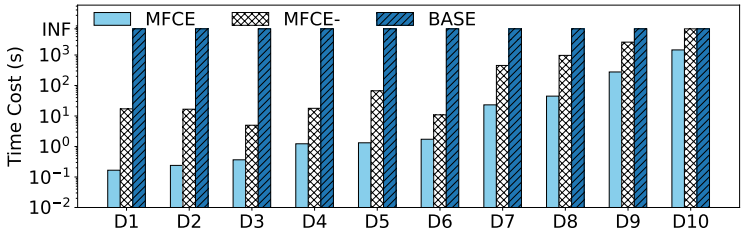


Fig. 3. Efficiency evaluation on all datasets

- MFCE-: the merge search algorithm developed in Section 4.1 (Algorithm 2).
- MFCE: MFCE- equipped with all the optimizations proposed in Section 4.
- MFCE-GR: MFCE without the graph reduction technique.
- MFCE-BP: MFCE without the branches pruning method.
- MFCE-BM: MFCE without the intersection acceleration via bitmap.

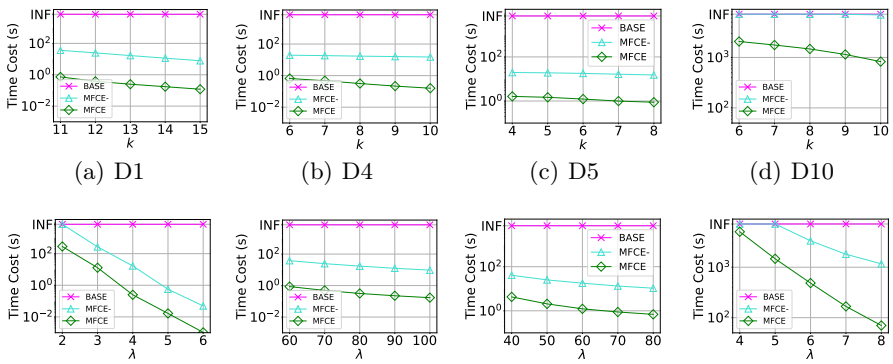
Datasets. We conduct the experiments on 10 real-world ML datasets sourced from SNAP[5] and [3,10,12]. These datasets encompass a wide range of domains, whose details are shown in Table 1. Among them, D7 and D10 are converted from temporal graphs by splitting the time window into different time intervals, and others are real ML graphs.

Parameters and workloads. We conduct the experiment by varying parameters k and λ, whose default values are shown in the last column of Table 1. All the programs are implemented in standard C++. All experiments are conducted on a server with an Intel Xeon 2.10GHz CPU and 256 GB of RAM. When the running time of algorithms exceeds 2 hours, we set them as **INF**.

5.2 Algorithms Evaluation

Exp-1: Experiments over all datasets. We present the response time of MFCE, MFCE-, and BASE across all datasets with default settings, as shown

[5]https://snap.stanford.edu/



(a) D1 (b) D4 (c) D5 (d) D10

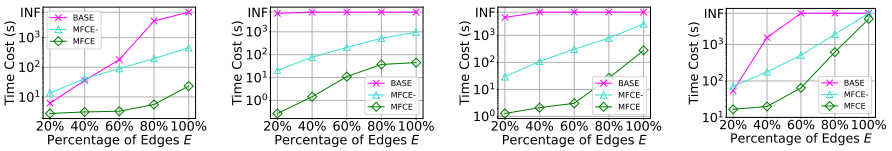
(e) D1 (f) D4 (g) D5 (h) D10

Fig. 4. Response time by varying parameters

in Fig. 3. Notably, BASE fails to complete on all datasets. This failure can be attributed to two main reasons. First, there is a significant imbalance in the distribution of edges within the ML graphs; approximately 80% of the edges are concentrated in 20% of the layers. For example, in dataset D1, some layers contain over 2K edges. Running the baseline algorithm BASE on such high-density layers can result in excessive recursion depth and memory overflow. Second, the approach of BASE, which involves enumerating all cliques before verifying their frequencies, results in considerable redundant processing. In contrast, MFCE and MFCE- significantly outperform BASE across all datasets. Our merged search method reduces numerous time-costly operations compared to the traditional layer-by-layer search approach. Besides, MFCE is at least 10 times faster than MFCE-, underscoring the effectiveness of our proposed optimizations in Sections 4.2, 4.2 and 4.4.

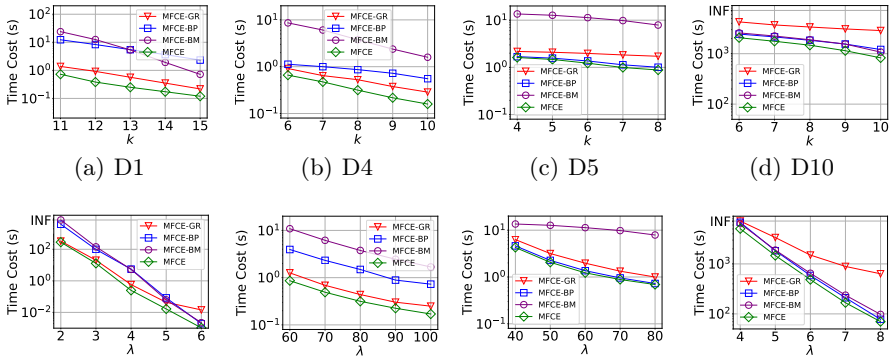
Exp-2: Response time by varying parameters. In this experiment, we independently vary k and λ, and report the response time of BASE, MFCE- and MFCE in Fig. 4. Note that the larger $\frac{|E|}{|\mathcal{E}|}$ implies the bigger range of k and λ. Hence we select four datasets with the largest $\frac{|E|}{|\mathcal{E}|}$, i.e., D1, D4, D5 and D10 to assess the parameter sensitivity and differences between our algorithms. Figs. 4(a) to 4(d) illustrate the impact of varying k while holding λ constant. With an increase in k, we observe a decrease in running time across all algorithms due to the more stringent constraints applied. Both MFCE and MFCE- consistently outperform BASE, underscoring the efficiency of our merged search method in eliminating redundant computations. Compared to MFCE-, MFCE can achieve up to an order of magnitude speedup, which proves the effectiveness and generalization of our optimizations. Similar trends and conclusions are observed when varying λ with k fixed, as depicted in Figs. 4(e) to 4(h).

Exp-3: Evaluation of algorithm scalability. In this part, we study the scalability of BASE, MFCE-, and MFCE on the four largest datasets, D7, D8, D9, and D10. Specifically, we randomly sample four subgraphs with 20%, 40%, 60%, and 80% edges for each dataset. Fig. 5 reports the response time of algorithms



(a) D7 (b) D8 (c) D9 (d) D10

Fig. 5. Scalability evaluation by sampling edges



(a) D1 (b) D4 (c) D5 (d) D10

(e) D1 (f) D4 (g) D5 (h) D10

Fig. 6. Ablation study

with the default parameters. Before 60% in D7 and D9, the runtime of MFCE increases slightly when the sampling ratio increases. This is because graph reduction results in an empty search space, MFCE operates exclusively on the graph reduction process. After the point of 60%, the increase of MFCE's runtime becomes more pronounced, as maximal (k, λ)-FCs clique enumeration is NP-hard. Overall, both MFCE- and MFCE demonstrate robust scalability, as they handle large-scale datasets without a notable rise in computational cost.

Exp-4: Ablation studies. To assess the individual effectiveness of the proposed three optimization methods, we conducted comparative analyses among MFCE-GR, MFCE-BP, MFCE-BM and MFCE with the results presented in Fig. 6. Our observations include the following: *i*) When comparing MFCE with MFCE-GR, the graph reduction technique demonstrates a more significant acceleration, particularly on sparse and large-scale graphs such as D10. *ii*) The branches pruning method excels in handling dense datasets, such as D1 and D4, optimizing performance by effectively reducing unnecessary computational overhead. *iii*) The intersection acceleration by bitmap shows a pronounced speedup on datasets with a large number of layers, like D4 and D5, achieving an order of magnitude speedup in comparison between MFCE and MFCE-BM.

Exp-5: Evaluation of graph reduction. To evaluate the performance of the graph reduction method, we conduct the experiments on the datasets D1, D4, D5 and D10 by varying k and λ, respectively. The results are shown in Fig. 7. Except in Fig. 7(e), both the runtime and the ratio of pruned edges tend to increase with

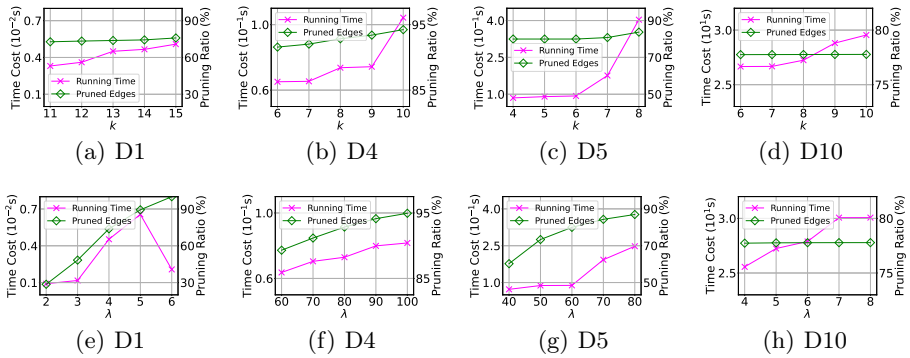


Fig. 7. Evaluation of graph reduction

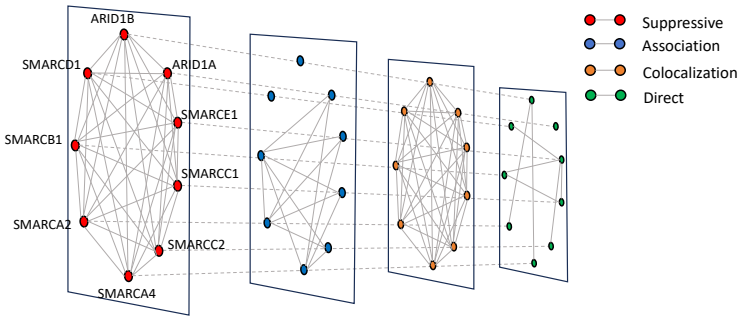


Fig. 8. Case study on D2

higher values of k and λ. This is because, the edge pruning process (Lemma 1) at $\lambda = 6$ results in the removal of nearly all edges, leading to the subsequent direct removal of all nodes due to unmet constraints. This obviates the need for further checks and updates on the remaining nodes. Increasing λ demonstrates a more substantial pruning effect compared to increasing k, indicating that there are many dense but infrequent structures within general ML graphs. Within the same dataset, constraints based on high frequency prune the graph more effectively than those based on larger size, indicating that ML graphs contain more unstable structures than sparse ones.

Exp-6: Case study. To demonstrate the effectiveness of our model, we present a maximal frequent clique of Homo dataset (D2) with $k = 9$ and $\lambda = 2$ in Fig. 8. For clarity, we display only the layers where this $(9, 2)$-FC is present. This frequent clique consists of 9 human genes whose support layer includes "Suppressive" and "Colocalization". The presence of genes forming a clique in both these layers, as opposed to only in "Suppressive" provides compelling evidence of highly coordinated regulatory interactions among them. This configuration suggests that these genes likely form the core components of a highly modular complex and play crucial roles in cellular functioning. Their dysfunction could significantly influence the onset and progression of diseases. Notably, previous

models have not detected this structure, highlighting the superior capability of our approach in uncovering complex biological networks.

6 Related Work

Graphs are widely used to model entity relationships in many fields [23,18,24]. As the densest subgraph in graph theory, clique is widely applied across diverse fields and has been extended to other versions suitable for specific graphs, e.g., [1,20,19,21]. Abidi et al. [1] investigated the problem of maximal biclique enumeration in bipartite graphs. Sun et al. [20] proposed signed clique, which aimed to better capture the cohesiveness within signed graphs, and Qin et al. [17] studied the problem of mining periodic communities in temporal graph based on σ-clique. However, no variants of clique specifically designed for ML graphs have been proposed yet. The joint mining of multiple datasets within ML graphs has the potential to reveal interesting and novel structures. For instance, Galimberti et al. [8] pioneered the study of core decomposition via core-based model in ML graph. Other researchers explored similar models, such as d-CC [27] and FirmCore [10]. To address the free-rider effect, Behrouz et al. [3] proposed a truss-based model, FirmTruss, and explored the problem of the FirmTruss-based community search. As discussed, such models may involve numerous unstable structures. In addition, the proposed techniques cannot be extended to support our problem.

7 Conclusion

Maximal clique enumeration is a fundamental problem in graph theory, yet the majority of existing research predominantly concentrates on single-layer graphs. In this paper, we propose a novel model, named (k, λ)-frequent clique, to better characterize the cohesive subgraph in ML graphs. We show that the problem of enumerating all the maximal (k, λ)-FCs is NP-hard. To solve the problem, a baseline algorithm is first constructed. Then, to scale for large ML graphs, a merge search algorithm and three optimization techniques are developed. Finally, extensive experiments are conducted on 10 real-world graphs to demonstrate the efficiency and effectiveness of the proposed model and techniques.

Acknowledgments. This work was supported by ARC DP240101322, DP230101445, UoW R6288.

References

1. Abidi, A., Zhou, R., Chen, L., Liu, C.: Pivot-based maximal biclique enumeration. In: IJCAI (2020)
2. Akkoyunlu, E.A.: The enumeration of maximal cliques of large graphs. SIAM Journal on Computing (1973)
3. Behrouz, A., Hashemi, F., Lakshmanan, L.V.: Firmtruss community search in multilayer networks. PVLDB (2022)

4. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM (1973)
5. Cardillo, A., Gómez-Gardenes, J., Zanin, M., Romance, M., Papo, D., Pozo, F.d., Boccaletti, S.: Emergence of network features from multiplexity. Scientific reports (2013)
6. Chen, C., Wu, Y., Sun, R., Wang, X.: Maximum signed θ-clique identification in large signed graphs. TKDE (2021)
7. Chen, C., Zhu, Q., Sun, R., Wang, X., Wu, Y.: Edge manipulation approaches for k-core minimization: Metrics and analytics. TKDE (2021)
8. Galimberti, E., Bonchi, F., Gullo, F.: Core decomposition and densest subgraph in multilayer networks. In: CIKM (2017)
9. Hartmanis, J.: Computers and intractability: a guide to the theory of np-completeness. Siam Review (1982)
10. Hashemi, F., Behrouz, A., Lakshmanan, L.V.: Firmcore decomposition of multilayer networks. In: WWW (2022)
11. Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multilayer networks. Journal of complex networks (2014)
12. Kunegis, J.: Konect: the koblenz network collection. In: WWW (2013)
13. Li, W., Liu, C.C., Zhang, T., Li, H., Waterman, M.S., Zhou, X.J.: Integrative analysis of many weighted co-expression networks using tensor computation. PLoS computational biology (2011)
14. Liu, D., Wang, R.A., Zou, Z., Huang, X.: Fast multilayer core decomposition and indexing. In: ICDE (2024)
15. Pan, M., Li, R.H., Zhang, Q., Dai, Y., Tian, Q., Wang, G.: Fairness-aware maximal clique enumeration. In: ICDE (2022)
16. Pei, J., Jiang, D., Zhang, A.: On mining cross-graph quasi-cliques. In: KDD (2005)
17. Qin, H., Li, R.H., Yuan, Y., Wang, G., Yang, W., Qin, L.: Periodic communities mining in temporal networks: Concepts and algorithms. TKDE (2020)
18. Sima, Q., Yu, J., Wang, X., Zhang, W., Zhang, Y., Lin, X.: Deep overlapping community search via subspace embedding. Proceedings of the ACM on Management of Data (2025)
19. Sun, R., Wu, Y., Chen, C., Wang, X., Zhang, W., Lin, X.: Maximal balanced signed biclique enumeration in signed bipartite graphs. In: ICDE (2022)
20. Sun, R., Wu, Y., Wang, X., Chen, C., Zhang, W., Lin, X.: Clique identification in signed graphs: a balance theory based model. TKDE (2023)
21. Sun, R., Wu, Y., Wang, X., Chen, C., Zhang, W., Lin, X.: Efficient balanced signed biclique search in signed bipartite graphs. TKDE (2023)
22. Sun, R., Zhu, Q., Chen, C., Wang, X., Zhang, Y., Wang, X.: Discovering cliques in signed networks based on balance theory. In: DASFAA (2020)
23. Wang, J., Wu, Y., Wang, X., Zhang, Y., Qin, L., Zhang, W., Lin, X.: Efficient influence minimization via node blocking. PVLDB (2024)
24. Wu, Y., Sun, R., Wang, X., Wen, D., Zhang, Y., Qin, L., Lin, X.: Efficient maximal frequent group enumeration in temporal bipartite graphs. PVLDB (2024)
25. Wu, Y., Jin, R., Li, J., Zhang, X.: Robust local community detection: on free rider effect and its elimination. VLDB (2015)
26. Yu, H., Paccanaro, A., Trifonov, V., Gerstein, M.: Predicting interactions in protein networks by completing defective cliques. Bioinformatics (2006)
27. Zhu, R., Zou, Z., Li, J.: Diversified coherent core search on multi-layer graphs. In: ICDE (2018)
28. Zou, Z., Li, J., Gao, H., Zhang, S.: Mining frequent subgraph patterns from uncertain graph data. TKDE (2010)