Efficient Maximal Motif-Clique Enumeration over Large Heterogeneous Information Networks

Yingli Zhou
The Chinese University of Hong Kong, Shenzhen
yinglizhou@link.cuhk.edu.cn

Yixiang Fang*
The Chinese University of Hong Kong, Shenzhen
fangyixiang@cuhk.edu.cn

Chenhao Ma
The Chinese University of Hong Kong, Shenzhen
machenhao@cuhk.edu.cn

Tianci Hou
The Chinese University of Hong Kong, Shenzhen
tiancihou@link.cuhk.edu.cn

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

ABSTRACT

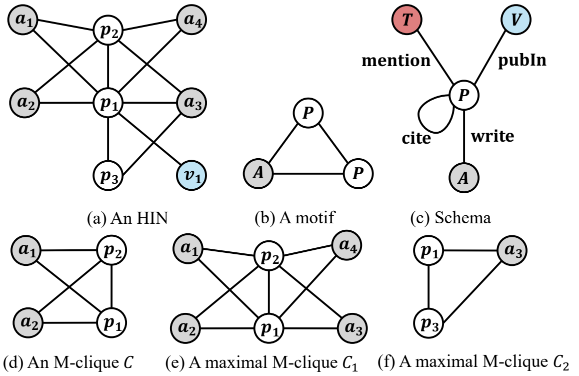
In the heterogeneous information network (HIN), a motif-clique is a "complete graph" for a given motif (or a small connected graph) that could capture the desired relationship in the motif. The maximal motif-cliques of HINs have found various applications in community discovery, recommendation, and biological network analysis. The state-of-the-art algorithm for enumerating maximal motif-cliques may have to explore all possible subgraphs of a maximal motif-clique and check whether a maximal motif-clique has been enumerated at each recursive step, which is very time-consuming. To improve the efficiency of enumeration, in this paper, we develop efficient algorithms for maximal motif-clique enumeration over large HINs. We first introduce an order-based framework to avoid duplicated enumeration, which results in lower time complexity compared to the existing algorithm. We then propose a pivot-based pruning strategy, which significantly reduces the search space. We further optimize the process of identifying the candidate sets and locating the subgraphs containing the maximal motif-cliques. Extensive experiments on five real-world HINs demonstrate that our proposed algorithm achieves high efficiency and is up to three orders of magnitude faster than the state-of-the-art algorithm.

PVLDB Reference Format:
Yingli Zhou, Yixiang Fang, Chenhao Ma, Tianci Hou, and Xin Huang. Efficient Maximal Motif-Clique Enumeration over Large Heterogeneous Information Networks. PVLDB, 17(11): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/EnderturtleOrz/VLDB2024-Mclique.

1 INTRODUCTION

Heterogeneous information networks (HINs) are networks with multiple typed objects and multiple typed links denoting different semantic relations. These graph data sources are prevalent in various domains, including bibliographic networks [73, 76], co-purchasing networks [74, 95], and knowledge graphs [32]. For example, Figure 1(a) illustrates an example HIN of the DBLP bibliographic network, consisting of four authors (i.e., $a_1, \cdots, a_4$), three papers (i.e., p_1, p_2, p_3), and one venue (i.e., v_1).



(a) An HIN (b) A motif (c) Schema
(d) An M-clique C (e) A maximal M-clique C_1 (f) A maximal M-clique C_2

Figure 1: An example HIN of DBLP network.

In this paper, we study the efficient solutions for the problem of **M**aximal **M**otif-**C**lique **E**numeration (or MMCE problem) over large HINs [39, 48]. As a fundamental graph mining topic, maximal clique enumeration (MCE) has gained tremendous attention [10, 15, 22–24, 30, 61, 78], but little has been paid to MCE on HINs. Recently, the MCE on HINs [39, 48] has been studied by using a motif to capture the desired relationship in it. A motif, a.k.a. higher-order structure or graphlet, is a small connected subgraph or pattern. As pointed out by [20, 40, 59, 64, 65, 69, 80], a motif is a fundamental building block of large and complex networks, and it enables "higher-order semantics" analysis for HINs. Figure 1(b) depicts a triangle motif, describing that an author writes two papers and one paper cites the another one. The "higher-order semantics" is a widely-used concept in graph mining [4, 53, 65, 92], indicating the complex multi-hop relationships captured by a motif, instead of an edge with direct connection. Given a motif, a motif-clique (M-clique) [39, 48] is a "complete graph" for capturing the higher-order semantics in the motif. Conceptually, the M-clique is a generalization of the classic clique for HINs, representing a user's defined patterns (motifs) rather than edges; as such, it is a "complete subgraph" according to the motif. For example, if a motif $\mathcal{M}$ is an edge linked by two vertices of the same type, then the M-clique is a clique in a homogeneous network; if $\mathcal{M}$ is an edge linked by two vertices of different types, then the M-clique actually is a biclique [1, 18] in a bipartite graph.

Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$, a vertex set R is called an M-clique, if for any subset T of its vertices, that contains the same

*Yixiang Fang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:XX.XX/XXX.XX

number of vertices and the same types of vertices as $\mathcal{M}$, the induced subgraph of T is subgraph isomorphic to $\mathcal{M}$. For example, Figure 1(d) shows an M-clique C, since both subsets $\{a_1, p_1, p_2\}$ and $\{a_2, p_1, p_2\}$ satisfy the above constrains. An M-clique is maximal if it is not a subgraph of any other M-clique. Figures 1(e) and (f) present two maximal M-cliques for the above motif in the HIN of Figure 1(a). Similar to the classic maximal clique enumeration (MCE) in homogeneous networks that has been applied to various applications like social network analysis [47, 62], financial network analysis [8], and biological network analysis[44], the MMCE on HINs has also found many real-world applications, to name a few:

• **Community discovery [39].** In social networks (e.g., Facebook), the communities can be modeled as maximal M-cliques, by using a specific motif as a guide. Our experiments in Section 7.4 show that the maximal M-cliques are effective for revealing communities in DBLP network. For example, given a motif that represents two authors who have published the same papers, we can discover the maximal M-cliques [39], which can be viewed as close-knit communities, since all members have a close collaborative relationship.

• **Bundle recommendation [13].** The MMCE solutions can be used to enhance the performance of bundle recommendation models, since a bundle can often be described by a motif. As shown in our case study in Section 7.4, the MMCE algorithms can improve the performance of SOTA bundle recommendation method BGCN [13], by first finding maximal M-cliques of motifs of bundles and then augmenting the networks of users, items, and bundles.

• **Biological network analysis [48].** In a biological graph, the maximal M-cliques can disclose new side effects of a drug, and potential drugs for healing diseases. For example, in the HIN of diseases, genes, and drugs, discovering the maximal M-cliques under specified motifs (e.g., a drug node and a disease node linking to the same gene node) could be used to study medicine characteristics and find potential side effects of drugs [48].

While it is very useful, the MMCE problem is NP-hard [39]. Hu et al. attempted to solve it by proposing a recursive backtracking algorithm META [39] based on the Bron-Kerbosch (BK) algorithm [10], which is a classic MCE algorithm for homogeneous graphs. It first identifies all subgraphs to which the motif is isomorphic. Here, the sets of vertices of these subgraphs are also called the *motif instances* of the motif. Then, for each motif instance, it iteratively enumerates all the maximal M-cliques that contain it by following a node expansion process. However, META is very costly, because to enumerate all the maximal M-cliques containing a specific motif instance, it needs to explore all the possible M-cliques that contain it. Besides, it treats checking whether each maximal M-clique has been enumerated as a subset query problem, i.e., if the M-clique R includes a motif instance Γ that is already visited, then the expansion of R can be skipped. Although it can avoid outputting duplicated maximal M-cliques, it is still inefficient since adding each new vertex to an M-clique requires a subset query.

We notice that to avoid duplicated search and reduce the time complexity of MCE, the previous algorithms of the classic MCE problem often define a search order for the vertices [18, 25, 27, 30, 50, 56]. The main idea is that for each vertex v, only its neighbors with an order greater than v need to be included in the enumeration process, so the overall search space can be dramatically reduced. However, the search order above cannot be directly applied to MMCE problem either. This is because, unlike MCE, MMCE needs to iteratively find maximal M-cliques containing each motif instance, rather than a single vertex, and defining an order for these motif instances is not straightforward. Besides, the BK algorithm can be significantly speeded up by exploiting the pivot principle [10, 30, 61, 78], whose key idea is that every maximal clique must contain either a vertex p or a non-neighbor of p. Thus, during the MCE expansion process, if a vertex p (known as a pivot vertex) is selected, then all its neighbors can be pruned in the current recursion, thus significantly reducing search space. Unfortunately, we demonstrate that the pivot principle cannot be directly applied to MMCE, because in each recursion, the neighbors of a pivot vertex may form a larger M-clique with the current M-clique, so it cannot be skipped.

Our technical contributions. To resolve the above issues, we first propose a new elegant order-based framework that can incorporate different kinds of vertex orders. Under the order-based framework, in the enumeration process, for each motif instance Γ, we only take into account the larger-order vertices. Here, a larger-order vertex is a vertex, if its order is higher than the maximum order of vertices that are with the same type in Γ. By exploiting the orders, the search space can be reduced dramatically.

We also design a novel pivot principle, called motif-pivot, which is very different from the previous pivot principle. The key idea of motif-pivot is that when a pivot vertex p is selected, we first identify a set $\mathcal{I}(p)$ of vertices, called M-clique precedence set, for reducing the search space. More precisely, in each recursion, any M-clique in $R \cup \mathcal{I}(p)$ can be enlarged by p. Since $R \cup \mathcal{I}(p)$ does not contain any solutions, the enumeration algorithm can safely prune the vertices in $\mathcal{I}(p)$ from the candidate set, which contains all the possible vertices that are in the maximal M-cliques. However, given a pivot vertex p, detecting the M-clique precedence set $\mathcal{I}(p)$ is an NP-hard problem, as we will prove later. To tackle this challenge, we propose a fast algorithm to quickly identify an approximate subset of $\mathcal{I}(p)$. As shown in our experiments later, this approximate set is very close to the original set, and it also can be quickly identified.

In addition, to further improve the efficiency, we introduce two non-trivial optimization techniques. The first one is to quickly identify the candidate set for each motif instance, and the second one is a graph reduction approach for eliminating edges and vertices that are not included in any maximal M-cliques.

By combining the above techniques together, we obtain a fast **P**ivot and **O**rder-based **M**MCE **A**lgorithm, also called POMA. We have performed extensive efficiency evaluation on five real-world large HINs. The results show that POMA achieves higher efficiency and scalability than the state-of-the-art algorithm META on all datasets, and is up to three orders of magnitude faster than META. We have also conducted two case studies on real-world HINs which show that enumerating maximal M-cliques could be useful in community discovery and bundle recommendation.

Contributions. In summary, our main contributions are:

- We propose an elegant order-based search framework to avoid duplicated computation;
- We develop a novel pivot principle to reduce the search space which significantly speedups the process of MMCE;
- We propose non-trivial optimization techniques to quickly identify the candidate set for each motif instance and locate the subgraphs containing the maximal M-cliques.
- We conduct experiments on five real-world large HINs to demonstrate the efficiency and scalability of our algorithm.

Outline. We formally present the MMCE problem in Section 2. Section 3 analyzes the limitations of the state-of-the-art algorithm.

The overall POMA algorithm and two optimizations are presented in Section 4. The order-based and pivot-based of our POMA algorithm are detailed in Sections 5 and 6, respectively. We report the experimental results in Section 7. We review the related works in Section 8 and conclude in Section 9. For lack of space, all the proofs in this paper are included in the technical report [93].

2 PROBLEM DEFINITION

In this section, we formally present the definition of MMCE problem. Table 1 summarizes the notations frequently used in this paper.

Table 1: Notations and meanings.

Notation	Meaning
$\mathcal{H} = (\mathcal{V}, \mathcal{E}, \psi, \phi)$	An HIN with vertex set $\mathcal{V}$, edge set $\mathcal{E}$, vertex type mapping function ψ, edge type mapping function ϕ
$\mathcal{M} = (\mathcal{V}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}}, \psi, \phi)$	A motif with vertex set $V_{\mathcal{M}}$, edge set $\mathcal{E}_{\mathcal{M}}$, vertex type mapping function ψ, edge type mapping function ϕ
Γ	A motif instance of $\mathcal{M}$ in $\mathcal{H}$
Φ	All motif instances of $\mathcal{M}$ in $\mathcal{H}$
$\mathcal{N}(v)$	The set of neighbors of v in $\mathcal{H}$: $\mathcal{N}(v) = \{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}$
$\mathcal{N}_{\mathcal{M}}(v)$	The set of neighbors of v in $\mathcal{M}$: $\mathcal{N}_{\mathcal{M}}(v) = \{u \in \mathcal{V}_{\mathcal{M}} \mid (u, v) \in \mathcal{E}_{\mathcal{M}}\}$
Δ	The number of motif instances of $\mathcal{M}$ in $\mathcal{H}$
$\mathcal{D}_u^R$	The set of vertices in R dominated by u
$\mathcal{H}[T]$	The subgraph of $\mathcal{H}$ induced by vertices set T
h	The number of vertex types in $\mathcal{M}$
$\mathcal{P}_R$	The candidate pivot set of R

Definition 1 (HIN [76]). *An HIN is an undirected graph* [1] *$\mathcal{H} = (\mathcal{V}, \mathcal{E}, \psi, \phi)$ with a vertex type mapping function $\psi : \mathcal{V} \rightarrow \mathcal{A}$, and an edge type mapping function $\phi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ is the set of vertex types, $\mathcal{R}$ is the set of edge types, each vertex $v \in \mathcal{V}$ belongs to a vertex type $\psi(v) \in \mathcal{A}$, each edge $e \in \mathcal{E}$ belongs to an edge type $\phi(e) \in \mathcal{R}$, $|\mathcal{A}| \geq 1$, $|\mathcal{R}| \geq 1$, and $|\mathcal{A}| + |\mathcal{R}| > 2$.*

An HIN often follows a schema, or a graph defined over vertex types $\mathcal{A}$ and edge types $\mathcal{R}$, denoted by $T_G = (\mathcal{A}, \mathcal{R})$. The HIN schema describes all allowable edge types between vertex types. Figure 1(b) gives the HIN schema of DBLP network. We use $h = |\mathcal{A}|$ to denote the number of vertex types in $\mathcal{A}$, and use $\mathcal{A}_i$ to denote the i-th vertex type in $\mathcal{A}$. Denoted by $\mathcal{N}(v) = \{u \in \mathcal{V} | (u, v) \in \mathcal{E}\}$, the set of neighbors of v in $\mathcal{H}$.

Given an HIN $\mathcal{H}$, a motif is a small connected HIN $\mathcal{M} = (\mathcal{V}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}}, \psi, \phi)$. Notice that a valid motif must adhere to the constraints imposed by $\mathcal{H}$'s schema, including adhering to vertex types defined by the schema and ensuring that edges follow the vertex relationships permitted by the schema. Denote by $\mathcal{N}_{\mathcal{M}}(v) = \{u \in \mathcal{V}_{\mathcal{M}} \mid (u, v) \in \mathcal{E}_{\mathcal{M}}\}$, a set of neighbors of v in $\mathcal{M}$. Since a motif is often small, it cannot capture the cohesive relationship among many vertices. Thus, people are more interested in motif-based subgraphs.

Definition 2 (Subgraph isomorphism [66]). *A motif $\mathcal{M}$ is subgraph isomorphic to an HIN $\mathcal{H}$, if there exists an injective mapping τ: $\mathcal{V}_{\mathcal{M}} \rightarrow \mathcal{V}$, s.t., $\forall u \in \mathcal{V}_{\mathcal{M}}$, $\psi(u) = \psi(\tau(u))$ and $\forall u, v \in \mathcal{V}_{\mathcal{M}}$, if $(u, v) \in \mathcal{E}_{\mathcal{M}}$, then $(\tau(u), \tau(v)) \in \mathcal{E}$ and $\phi(u, v) = \phi(\tau(u), \tau(v))$, where $\tau(u)$ is the vertex to which u is mapped.*

Clearly, $\mathcal{M}$ is subgraph isomorphic to $\mathcal{H}$ iff $\mathcal{M}$ is isomorphic to a subgraph (not necessarily induced) of $\mathcal{H}$.

Definition 3 (Type-matched vertex set [39]). *Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$, a vertex set $T \subseteq \mathcal{V}$ is a type-matched vertex set of $\mathcal{M}$, if T and $\mathcal{M}$ have the same number of vertices (i.e., $|T| = |\mathcal{V}_{\mathcal{M}}|$), and there exists a bijection $\tau : T \rightarrow \mathcal{V}_{\mathcal{M}}$, s.t., $\forall u \in T, \psi(u) = \psi(\tau(u))$.*

In the case without ambiguity, we will simply use "match-set" to denote "type-matched vertex set". If a motif $\mathcal{M}$ is subgraph isomorphic to the induced subgraph of a match-set T, then T is called a motif instance of $\mathcal{M}$.

Definition 4 (Motif-clique [39, 48, 83]). *Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$, a vertex set R is a motif-clique (M-clique), if for each match-set T in R, $\mathcal{M}$ is subgraph isomorphic to the induced subgraph $\mathcal{H}[T]$.*

An M-clique R is maximal if there does not exist any other vertex set R' in $\mathcal{H}$ such that R' is an M-clique and $R \subsetneq R'$. Since a maximal M-clique often contains an exponential number of small M-cliques, enumerating all M-cliques will be extremely costly. Thus, we focus on enumerating maximal M-cliques rather than all M-cliques. We now formally present the definition of MMCE problem [39, 48].

Problem 1 (MMCE problem [39, 48]). *Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$, enumerate all the maximal M-cliques of $\mathcal{M}$ in $\mathcal{H}$.*

Example 1. *Consider the HIN and motif in Figure 1. There are five match-sets of $\mathcal{M}$, i.e., $T_1 = \{a_1, p_1, p_2\}$, $T_2 = \{a_2, p_1, p_2\}$, $\cdots$, $T_5 = \{a_3, p_1, p_2\}$. Clearly, all these match-sets are motif instances of $\mathcal{M}$, because for each match-set T_i, $\mathcal{M}$ is subgraph isomorphic to $\mathcal{H}[T_i]$. Then, we can find that there are two maximal M-cliques $R_1 = \{a_1, a_2, a_3, a_4, p_1, p_2\}$ and $R_2 = \{a_3, p_1, p_2\}$ in the HIN, as depicted in Figures 1 (d) and (e) respectively.*

3 EXISTING APPROACH

In this section, we review existing algorithm META [39] for solving the MMCE problem, and then discuss its limitations.

3.1 The META algorithm

To solve the MMCE problem, Hu et al. [39] proposed a recursive backtracking algorithm META based on the classic BK algorithm [10], as shown in Algorithm 1. The main idea is to maintain three disjoint sets R, C, and X in the recursive enumeration procedure, where R is an M-clique, C is a set of candidates that can be added to R to form a larger M-clique, and X is a set of vertices that have already been explored from C. In each recursion, the three sets keep the invariance that $R \cup \{v\}$ is an M-clique if $v \in C \cup X$, and $R \cup \{v\}$ is not an M-clique if $v \notin C \cup X$. The algorithm recursively processes the vertices in C to expand the current M-clique R, until C=$\emptyset$.

In Algorithm 1, it first computes all the motif instances of $\mathcal{M}$ (line 1). Then, for each motif instance Γ, it invokes the procedure GetMMC to enumerate all the maximal M-cliques that contain it (lines 2-5). GetMMC recursively processes each vertex $u \in C$ to expand the current M-clique R. In each iteration, it updates the sets C and X using function Refine to keep the invariance that $R \cup \{u\}$ is also an M-clique when $u \in C' \cup X'$ (lines 15-16). Afterwards, it removes u from C and adds it into X (line 12). Finally, GetMMC stops when all the vertices in C are processed.

Lemma 3.1 ([39]). *Given an HIN $\mathcal{H}$ with n vertices and a motif $\mathcal{M}$,* META *costs $O(\gamma + \alpha n!)$ time, where γ and α denote the time cost of finding all motif instances of $\mathcal{M}$ in $\mathcal{H}$ and checking whether a vertex can enlarge the current M-clique, respectively.*

[1] In this paper, we follow the [39] and focus on undirected HINs for simplicity, but our techniques can be readily extended to handle the directed HINs.

Algorithm 1: META ($\mathcal{H}$, $\mathcal{M}$) [39]

```
   input  : An HIN $\mathcal{H}$ and a motif $\mathcal{M}$
   output : All the maximal M-cliques of $\mathcal{M}$ in $\mathcal{H}$
1  $\Phi \leftarrow$ compute all motif instances of $\mathcal{M}$ in $\mathcal{H}$ ;
2  foreach $\Gamma \in \Phi$ do
3      $\Gamma' \leftarrow \{u | u \in \mathbf{V}\backslash\Gamma \wedge \mathcal{N}(u) \cap \Gamma \neq \emptyset\}$;
4      $C \leftarrow$ Refine($\Gamma, \Gamma'$);
5      GetMMC($\Gamma, C, \emptyset$) ;
6  Function GetMMC($R, C, X$):
7      if $C \cup X = \emptyset$ then report $R$ as a maximal M-clique;
8      foreach $u \in C$ do
9          $C' \leftarrow$ Refine($R \cup \{u\}, C$);
10         $X' \leftarrow$ Refine($R \cup \{u\}, X$);
11         GetMMC($R \cup \{u\}, C', X'$);
12         $X \leftarrow X \cup \{u\}$;
13 Function Refine($R, C$):
14     $C' \leftarrow \emptyset$;
15     foreach $u \in C$ do
16         if $R \cup \{u\}$ is an M-clique then $C' \leftarrow C' \cup \{u\}$;
17     return $C'$
```

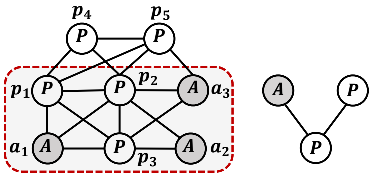
Lemma 3.2 ([39]). *Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$,* META *costs $O(\Delta)$ space where Δ is the number of motif instances of $\mathcal{M}$ in $\mathcal{H}$.*

Note that verifying whether a vertex can be added to an existing M-clique to generate a larger one is also NP-hard [39]. To alleviate this issue, Hu et al. [39] proposed a pruning criterion by introducing the concept of dominance relationships among vertices.

Definition 5 (Dominance [39]). *Given an HIN $\mathcal{H}$, an M-clique R and two vertices $u \in \mathcal{V}\backslash R$ and $v \in R$ with $\psi(u) = \psi(v)$, v is dominated by u if $(\mathcal{N}(v) \cap R) \subseteq (\mathcal{N}(u) \cap R)$.*

Let $\mathcal{D}_v^R$ denote the set of vertices in R dominated by v. As proved in [39], if $1 + |\mathcal{D}_v^R| > t$, where t denotes the number of vertices in $\mathcal{M}$ with type $\psi(v)$, we can directly append v from C to R without performing any additional checks.

In this case, for any match-set $T \subseteq R \cup \{u\}$ containing u, $T \cup \{v\} \setminus \{u\}$, i.e., replacing u with any $v \in \mathcal{D}_u^R$, is a motif instance, which means T is also a motif instance. Besides, Hu et al. presented an early stop strategy and set-trie structure to avoid redundant searches. For lack of space, we omit the details.



(a) An HIN (b) A motif

Figure 2: An example illustration the limitations of META.

3.2 Limitations of META

A major limitation of META is that to find a maximal M-clique, it has to explore all the possible non-maximal M-cliques contained by it, which is very costly. For motif instance, consider the HIN (shown in the shaded region), and motif in Figure 2. To enumerate all maximal M-cliques containing the motif instance $\Gamma = \{a_1, p_1, p_2\}$, META has to explore a total number of seven M-cliques, which respectively include the (2^3-1) subsets of $\{a_2, a_3, p_3\}$, but there is only one maximal M-clique containing Γ. Meanwhile, when a new vertex is added into R to expand the current M-clique, META needs to check if this M-clique is already checked (i.e., before line 10 in Algorithm 1), which is also time-consuming. Our later experiments show that META is very inefficient for processing large HINs (e.g., on a DBLP dataset with two million edges, it cannot enumerate all the maximal M-cliques for 100 motifs within 30 days). Hence, there is much room for improvement.

Opportunities: The most of the existing MCE algorithms [18, 25, 27, 30, 50, 56] utilize a fixed total search order to ensure that each maximal clique is enumerated only once, which dramatically shrinks the search space and also reduces the time complexity. The key idea is that for each vertex v, only its neighbors with an order greater than v need to be included in the enumeration process. However, the search order above cannot be directly applied to the MMCE problem either. This is because, unlike MCE, MMCE needs to iteratively find maximal M-cliques containing each motif instance, rather than a single vertex. Hence, defining an order for these motif instances is not easy.

Besides, the pivot principle has been widely used to accelerate MCE [10, 30, 78]. The key idea is that any maximal clique of a graph either contains a vertex v or a non-neighbor of v. *In other words, the clique containing only the neighbors of v must not be a maximal clique, because v can be added into it to form a larger one.* Thus, if a vertex v in $C \cup X$, called a pivot vertex, is selected, then all neighbors of $v \in C$ can be skipped to expand the current clique R, thus significantly reducing the number of recursive calls. Unfortunately, the pivot principle cannot be directly applied to the MMCE problem, since the key property for maximal cliques above does not hold for maximal M-cliques — *the maximal M-clique may not contain v but only contains its neighbor.* Reconsider the HIN and motif in Figure 2, for example, the vertex set $\{a_1, p_1, p_2, p_4\}$ is an M-clique. It does not contain vertex p_5 but only contains one of its neighbor p_4, since $\{a_1, p_4, p_5\}$ is not a motif instance of the motif.

Inspired by the discussions above, we develop a novel **P**ivot and **O**rder-based **M**MCE **A**lgorithm POMA in the following sections.

4 OUR POMA ALGORITHM

In this section, we first give an overview of our POMA algorithm, and then introduce the two optimization techniques.

4.1 Overview of POMA

Algorithm 2 gives an overview of POMA which sequentially performs the following five steps:

(1) Removing vertices and edges that cannot be included in any M-cliques by reduceHIN from the HIN (see Section 4.2);
(2) Computing all motif instances of motif $\mathcal{M}$ in the HIN by using a subgraph isomorphism algorithm (e.g., VF3 [12]);
(3) Identifying candidate sets by selectCand (see Section 4.3);
(4) For each motif instance Γ, imposing an order constraint to reduce the redundant computation (see Section 5);
(5) For each motif instance Γ, finding all the maximal M-cliques containing it recursively with motif-pivot principle (see Section 6);

Next, we present the algorithms for Steps (1) and (3) in Sections 4.2 and 4.3 respectively. The two key techniques of ordering strategy and pivot principle in Steps (4) and (5) will be discussed in Sections 5 and 6 respectively. In Step (2), any subgraph isomorphism algorithm can be used and we use the SOTA algorithm VF3 [12].

Algorithm 2: POMA($\mathcal{H}$, $\mathcal{M}$)

```
input  : An HIN $\mathcal{H}$ and a motif $\mathcal{M}$
output : All maximal M-cliques of $\mathcal{M}$ in $\mathcal{H}$
$\mathcal{H} \leftarrow$ reduceHIN($\mathcal{H}$,$\mathcal{M}$) ;                 // See Section 4.2
$\Phi \leftarrow$ compute all motif instances of $\mathcal{M}$ in $\mathcal{H}$;
$\lambda \leftarrow$ sort the vertices by an ordering strategy;
$C \leftarrow$ selectCand($\mathcal{H}$, $\Phi$) ;                   // See Section 4.3
foreach $\Gamma \in \Phi$ do
    $C, X \leftarrow$ Order-Divide($C[\Gamma], \Gamma, \lambda$) ;      // See Section 5
    MP-MMC($\Gamma, C, X$) ;                                // See Section 6
```

4.2 Graph reduction

We introduce two reduction rules to eliminate some vertices and edges that cannot be included in any M-cliques.

- **Rule 1.** For each vertex $v \in \mathcal{V}$, if there does not exist a vertex $v' \in \mathcal{V}_{\mathcal{M}}$ such that $\psi(v) = \psi(v')$, then we can remove v from $\mathcal{H}$ directly, since it is not included in any M-clique.

Similarly, for each edge $e \in \mathcal{E}$, if there does not exist an edge $e' \in \mathcal{E}_{\mathcal{M}}$ such that $\phi(e) = \phi(e')$, then we can remove e from $\mathcal{H}$ directly.

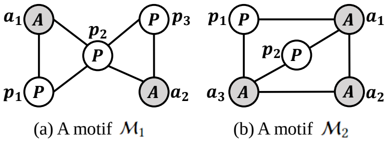
Next, we present two definitions before introducing Rule 2.

Definition 6 (Motif orbit [53]). *Given a motif $\mathcal{M}$, two vertices $u, v \in \mathcal{V}_{\mathcal{M}}$ are in the same orbit, if there is an automorphism mapping $\tau : \mathcal{V}_{\mathcal{M}} \rightarrow \mathcal{V}_{\mathcal{M}}$ with $\tau(u) = v$.*

Note that if vertex u is in the same orbit as vertex v, and v is in the same orbit as vertex x, then u and x must be in the same orbit.

Definition 7 (Orbit type). *Given a motif $\mathcal{M}$, a vertex type γ is said to be an orbit type, if $\mathcal{M}$ contains either a single vertex of type γ or all vertices of type γ belong to the same orbit.*

Note that for any two vertices x, y of the same orbit type in the motif, we can assert that the neighbors of x and y have the same number of vertices of each type in $\mathcal{M}$. Otherwise, x and y cannot be on the same orbit, which contradicts our assumption.



(a) A motif $\mathcal{M}_1$ (b) A motif $\mathcal{M}_2$

Figure 3: Illustration of the orbit type.

Example 2. *In the motif $\mathcal{M}_1$ of Figure 3, "author" is an orbit type, since there exists an automorphism mapping $\tau : \mathcal{V}_{\mathcal{M}_1} \rightarrow \mathcal{V}_{\mathcal{M}_1}$ with $\tau(a_1) = a_2$, $\tau(p_1) = p_3$, and $\tau(p_2) = p_2$. However, "paper" is not an orbit type, because p_2 has a larger degree than p_1 and p_3.*

- **Rule 2.** Given two vertices $v \in \mathcal{V}$, $x \in \mathcal{V}_{\mathcal{M}}$, where $\psi(v) = \psi(x)$ and $\psi(x)$ is an orbit type in the motif $\mathcal{M}$, if there exists a vertex type γ such that

$$|\{u \in \mathcal{N}(v) \mid \psi(u) = \gamma\}| < |y \in \mathcal{N}_{\mathcal{M}}(x) \mid \psi(y) = \gamma\}|, \qquad (1)$$

then we can remove v from $\mathcal{H}$ because it is not included in any maximal M-clique.

By combining the aforementioned two rules, when a motif $\mathcal{M}$ is specified, we can locate the maximal M-cliques of $\mathcal{M}$ in a small subgraph of $\mathcal{H}$ by iteratively removing vertices and edges that violate the above rules from $\mathcal{H}$. We refer to this graph reduction algorithm as reduceHIN and provide the details in the technical report [93].

4.3 Fast candidate set calculation

To compute new candidates, we can use the dominance relationship between two vertices to significantly boost efficiency. Nonetheless, when $R = \Gamma$, the number of vertices that can be dominated by vertices in R is quite limited. Consequently, the process of determining the candidate set C for a motif instance Γ remains computationally intensive. Recall that each vertex u in the candidate set satisfies that $\Gamma \cup \{u\}$ forms an M-clique. To solve this computational bottleneck, we propose a novel algorithm that effectively computes the candidates set for all motif instances in polynomial time. Before introducing our algorithm, we give the following definition.

Definition 8 (Motif instance Neighbor). *Given an HIN $\mathcal{H}$, a motif $\mathcal{M}$, and two motif instances Γ_1, Γ_2 of $\mathcal{M}$, Γ_1 and Γ_2 are motif instance neighbor if $|\Gamma_1 \backslash \Gamma_2| = 1$, and $\Gamma_1 \cup \Gamma_2$ is also an M-clique of $\mathcal{M}$. Let $\mathcal{S}[\Gamma]$ denote the set of motif instance neighbors of Γ.*

Lemma 4.1 shows how to identify the candidate set.

Lemma 4.1. *Given a motif instance Γ of $\mathcal{M}$, the candidate set C of Γ can be identified by gathering the different vertices from all motif instance neighbors of Γ using the following equation:*

$$C = \bigcup_{\Gamma_i \in \mathcal{S}[\Gamma]} \{\Gamma_i \backslash \Gamma\} \qquad (2)$$

According to Lemma 4.1, we can directly compute the candidate set for a motif instance Γ by identifying its motif instance neighbors. The following three steps outline the process: (1) compute a motif instances set $\mathcal{T}[\Gamma]$, s.t., $\forall T \in \mathcal{T}[\Gamma]$, $|T \backslash \Gamma| = 1$; (2) for each motif instance $T \in \mathcal{T}[\Gamma]$, denoting $u = T \backslash \Gamma$, if all match-sets in $\Gamma \cup \{u\}$ containing u are also motif instances of $\mathcal{M}$, T is a motif instance neighbor of Γ; (3) the candidate set for Γ is computed based on Lemma 4.1. The detailed steps are illustrated in Algorithm 3.

Algorithm 3: selectCand($\mathcal{H}$, Φ)

```
   input  : An HIN $\mathcal{H}$ and a motif instance set $\Phi$ of motif $\mathcal{M}$
   output : The candidate sets for all motif instances in $\Phi$
 1 $C \leftarrow \emptyset$ ;                                   // The candidate sets
 2 foreach $\Gamma \in \Phi$ do
 3     $\mathcal{T}[\Gamma] \leftarrow \{T | (|T \backslash \Gamma| = 1) \wedge T \in \Phi\}$;
 4     foreach $T \in \mathcal{T}[\Gamma]$ do
 5         $u \leftarrow T \backslash \Gamma$;
 6         $\mathcal{Y} \leftarrow$ all match-sets in $\Gamma \cup u$ that contain $u$;
 7         if all match-sets in $\mathcal{Y}$ are within $\mathcal{T}[\Gamma]$ then
 8             $\mathcal{S}[\Gamma] \leftarrow \mathcal{S}[\Gamma] \cup T$ ; // motif instance neighbors
 9     $C[\Gamma] \leftarrow \bigcup_{\Gamma_i \in \mathcal{S}[\Gamma]} \{\Gamma_i \backslash \Gamma\}$;
10 return $C$;
```

We first initialize C=$\emptyset$ and use it to save the candidate sets (line 1). Then, we compute the candidate set for each motif instance Γ in Φ one by one. Specifically, for each motif instance Γ, we compute a motif instance set $\mathcal{T}[\Gamma]$ (line 3). Next, for each motif instance $T \in \mathcal{T}[\Gamma]$, we denote $u = \{T \backslash \Gamma\}$, and then, we calculate all match-sets, denoted as $\mathcal{Y}$, that are formed by adding u into Γ. If all these match-sets in $\mathcal{Y}$ are also motif instances of $\mathcal{M}$, T is a motif instance neighbor of Γ (lines 5-8). Afterward, we can compute the candidate set of Γ (line 9). Finally, after iterating all motif instances in Φ, we stop and return C (line 10).

Example 3. *Reconsider the HIN (depicted in the shaded region), and motif in Figure 2, taking $\Gamma = \{a_1, p_1, p_2\}$ as an example. There* *are four motif instances that only have a one-vertex difference from Γ, namely $\Gamma_1 = \{a_1, p_1, p_3\}$, $\Gamma_2 = \{a_1, p_2, p_3\}$, $\Gamma_3 = \{a_2, p_1, p_2\}$, and $\Gamma_4 = \{a_3, p_1, p_2\}$, that differ from Γ by only one vertex. Therefore, we have $\mathcal{T}[\Gamma] = \{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4\}$. For the motif instance $\Gamma_1 \in \mathcal{T}$, we can compute $\mathcal{Y} = \{\{a_1, p_1, p_3\}, \{a_1, p_2, p_3\}\}$. Notably, all match-sets in $\mathcal{Y}$ are also motif instances of $\mathcal{M}$ (i.e., Γ_1, and Γ_2). Hence, Γ_1 is a motif instance neighbor of Γ. After scanning all motif instances in $\mathcal{T}[\Gamma]$,we can identify all the motif instance neighbors of Γ, represented as $\mathcal{S}[\Gamma]$ = $\{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4\}$. Therefore, we can compute the candidate set for Γ, $C[\Gamma] = \{a_2, a_3, p_3\}$.*

Theorem 4.2. *Algorithm 3 correctly computes the candidate sets for all motif instances of $\mathcal{M}$.*

Proof sketch. For any $Y \in \mathcal{T}[\Gamma]$, let $u = (Y \backslash \Gamma)$. Then, all match-sets in $\Gamma \cup \{u\}$ are also motif instances of $\mathcal{M}$, since $\Gamma \cup \{u\}$ is an M-clique. Hence, any $Y \in \mathcal{T}[\Gamma]$ and all motif instance neighbors of Γ can be computed by Algorithm 3. □

Lemma 4.3. *The total time cost of* `selectCand` *is $O(|\mathcal{V}| \cdot \Delta)$, where Δ is the number of motif instances of $\mathcal{M}$ in $\mathcal{H}$.*

Proof sketch. In the worst case, for each motif instance $\Gamma \in \Phi$, we need $O(|\mathcal{V}| \cdot |\mathcal{V}_{\mathcal{M}}|)$ time to compute its motif instance neighbor, where $|\mathcal{V}_{\mathcal{M}}|$ is the number of vertices in the motif, which can be considered as a constant in practice. □

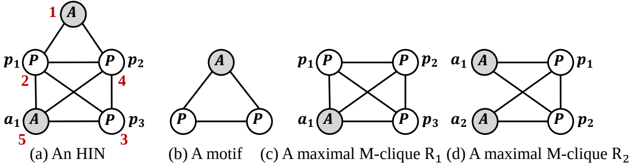
In addition, we summarize the time, and space complexities of each step of `META` and `POMA`, and the limitations of `POMA` in our technical report [93].

5 AN ORDER-BASED SEARCH FRAMEWORK

In this section, we introduce a novel ordering technology that can be used to avoid the duplicate enumeration of maximal M-cliques.

5.1 Our order-based search framework

The main idea of the order-based search framework in the MCE problem is that for each vertex v, only its neighbors with an order greater than v need to be included in the enumeration process, so the overall search space can be dramatically reduced, where degree order [50, 84, 90] and degeneracy order [14, 18, 25] are commonly used orders. While order-based techniques for improving the performance of MCE have been well studied, their utilization for accelerating the MMCE process remains unexplored. Unfortunately, those vertex order techniques cannot be directly applied to MMCE, because MMCE needs to iteratively enumerate maximal M-cliques containing each motif instance, rather than a single vertex. Hence, it is desired to design a new order-based search framework for the MMCE problem.



(a) An HIN (b) A motif (c) A maximal M-clique R_1 (d) A maximal M-clique R_2

Figure 4: Illustrating the limitations of existing orders.

- **A failed attempt.** The key challenge in designing an order-based search framework lies in determining how to compare the order of vertices in a motif instance with that of a vertex in its candidate set. The initial idea is to use either the maximum or minimum order value of all the vertices within a motif instance to denote its order. However, both methods have inherent limitations. We illustrate the limitations via Figure 4, where the order of each vertex is displayed below it in red, and there are four motif instances in the graph, i.e., $\Gamma_1 = \{a_1, p_1, p_2\}$, $\Gamma_2 = \{a_1, p_2, p_3\}$, $\Gamma_3 = \{a_1, p_1, p_3\}$, and $\Gamma_4 = \{a_2, p_1, p_2\}$.

  (1) *If the maximum order is adopted, some maximal M-cliques will be missed.* Assume that we first detect all maximal M-cliques containing Γ_1, with $C = \{a_2, p_3\}$ and the maximum vertex order of Γ_1 is 5. As all vertices in C have orders less than 5, R_1 cannot be obtained. Similarly, in the process of detecting the maximal M-cliques containing other motif instances, such as Γ_2, Γ_3, and Γ_4, R_1 remains undiscovered.

  (2) *Using the minimum order will lead to redundant computations.* When we detect maximal M-cliques containing Γ_1, and Γ_1 has an order of 2, the candidate set C is $\{p_3\}$. As a result, R_1 will be enumerated. Similarly, when detecting maximal M-cliques containing Γ_3, R_1 will also be enumerated. Hence, the advantages of the order-based search strategy will not be fully utilized.

- **A new search framework.** To ensure each maximal M-clique is enumerated only once, we introduce a novel order-based search framework. The main idea of our new search framework is that for each motif instance Γ of $\mathcal{M}$, we only consider larger-order vertices in the candidate set of Γ. Here, a larger-order vertex is a vertex, if its order is higher than the maximum order of the vertices in Λ_Γ with the same type. Let $\Lambda_\Gamma[\cdot]$ denote the maximum order of the vertices in each vertex type in Γ:

$$\Lambda_\Gamma[\gamma] = \max\{\lambda(u) | u \in \Gamma \wedge \psi(u) = \gamma\}, \tag{3}$$

where γ is a vertex type in Γ. Consequently, no result will be missed and each maximal M-clique is enumerated only once, which will be proved in Section 5.3. Based on the above analysis, we develop a new order-divide algorithm using any given vertex order, called `Order-Divide`, in Algorithm 4.

Algorithm 4: `Order-Divide` (C, Γ, λ)

```
  input  : A candidate set $C$, a motif instance $\Gamma$, and a vertex order $\lambda$
  output : A new candidate set $C'$ and not set $X$
1 $\Lambda_\Gamma[\cdot] \leftarrow \emptyset$;
2 foreach $v \in \Gamma$ do
3     $\Lambda_\Gamma[\psi(v)] \leftarrow \max\{\lambda(u) | u \in \Gamma \wedge \psi(u) = \psi(v)\}$;
4 $C' \leftarrow \{u | u \in C \wedge \lambda(u) > \Lambda_\Gamma[\psi(u)]\}$;
5 $X \leftarrow \{u | u \in C \wedge \lambda(u) < \Lambda_\Gamma[\psi(u)]\}$;
6 return $C'$ and $X$;
```

Specifically, we first compute the maximum order of the vertices for each vertex type in a motif instance Γ (lines 2-3). Next, we divide the candidate set C into the new candidate set C' and the not set X by only considering the larger-order vertices in C (lines 4-5). Finally, the new sets C' and X are returned (line 6).

Example 4. *Reconsider the HIN and the motif instances in Figure 4 with $\Gamma_1 = \{a_1, p_1, p_2\}$, $\Gamma_2 = \{a_1, p_2, p_3\}$, $\Gamma_3 = \{a_1, p_1, p_3\}$, and $\Gamma_4 = \{a_2, p_1, p_2\}$. Taking $\Gamma_3 = \{a_1, p_1, p_3\}$ as an example, we have $\Lambda_{\Gamma_3}[A] = 5$ and $\Lambda_{\Gamma_3}[P] = 3$, where 5 represents the order of a_1 (i.e., $\lambda(a_1) = 5$) and 3 represents the maximum value of $\lambda(p_1)$ and $\lambda(p_3)$. By running Algorithm 4, we get $C = \{a_2, p_2\}$, $C' = \{p_2\}$, and $X = \{a_2\}$ for Γ_3. The detailed information is provided in Table 2.*

Table 2: Illustrating the `Order-Divide`.

Motif instances	$\Lambda_{\Gamma_i}[\cdot]$	C	C'	X
Γ_1=$\{a_1, p_1, p_2\}$	$[5,4]$	$\{a_2, p_3\}$	$\emptyset$	$\{a_2, p_3\}$
Γ_2=$\{a_1, p_2, p_3\}$	$[5,4]$	$\{a_2, p_1\}$	$\emptyset$	$\{a_2, p_1\}$
Γ_3=$\{a_1, p_1, p_3\}$	$[5,3]$	$\{a_2, p_2\}$	$\{p_2\}$	$\{a_2\}$
Γ_4=$\{a_2, p_1, p_2\}$	$[1,4]$	$\{a_1\}$	$\{a_1\}$	$\emptyset$

5.2 Ordering heuristics

As shown in Algorithm 4, different vertex orders may have significant effect on the efficiency of MMCE. However, finding the optimal order is an NP-hard problem [27], so in the literature, some heuristic orders have been proposed, such as degree order and degeneracy order, which are described as follows:

• **Degree order.** The degree order [50, 84, 90] is the simplest one, where the vertices are arranged in ascending order based on their degrees. By using the degree order, the size of the largest candidate set is no larger than the maximum degree.

• **Degeneracy order.** Another simple yet efficient ordering is to use the degeneracy order [25, 30], which is widely used in MCE algorithms. Such an order can be obtained by repeatedly peeling a vertex of the minimum degree in the remaining subgraph, where the vertex removing order is exactly the degeneracy order. Such a peeling procedure can be done in $O(m+n)$, where m and n denote the number of edges and vertices in the graph, respectively. In the HIN, we can obtain a degeneracy order by ignoring the vertex and edge types and running classic core decomposition algorithm [3]. Since the degeneracy order is theoretically better than the degree order [30], so we employ it in this paper.

5.3 Theoretical correctness analysis for `POMA`

Lemma 5.1. *For each motif instance Γ of $\mathcal{M}$, Algorithm 2 can enumerate all the maximal M-cliques containing it without missing any results.*

Proof sketch. Suppose there exists a maximal M-clique R containing Γ that cannot be enumerated. We can construct a new motif instance Γ' by selecting vertices of each type with the minimum order from R, and then R can be enumerated by the recursive enumeration starting from Γ', which contradicts our assumption. □

Lemma 5.2. *Algorithm 2 ensures that each maximal M-clique will only be enumerated exactly once.*

Proof sketch. Proof this by contradiction, supposing a maximal M-clique R can be enumerated twice by two motif instances Γ_1 and Γ_2. However, whether Λ_{Γ_1} is dominated by Λ_{Γ_2} or not, R cannot be enumerated twice. □

Lemma 5.3. *The total time cost of* `POMA` *is $O(\gamma + \alpha\Delta 2^{\beta})$, where $\beta = h \times \delta$, δ is the degeneracy number of the input HIN, h is the number of vertex types in the input motif, Δ is the number of motif instance of $\mathcal{M}$ in $\mathcal{H}$, and the other variables have the same meanings as those in Lemma 3.1.*

Proof sketch. In the worst-case scenario, each motif instance Γ necessitates the consideration of up to 2^{β} branches in Algorithm 2. For each branch, we need $O(\alpha)$ time to check whether a vertex can expand to the M-clique. □

Discussions. Although our ordering framework shares some general purpose with those of classic MCE algorithms [30, 61, 78], it significantly differs from them in three aspects:

- *Order definitions:* In MCE algorithms, the order is defined based on comparing vertices, while in `POMA`, the order is defined based on comparing both vertices and motif instances.
- *Order-based pruning strategies:* In MCE, when enumerating the maximal cliques containing a vertex v, only the neighbours of v with larger orders than v need to be considered. In contrast, in `POMA`, we find maximal M-cliques containing each motif instance, rather than a single vertex, so we only consider "larger-order" vertices. Here, a larger-order vertex is a vertex whose order is higher than the maximum order of the vertices with the same type in this motif instance.
- *Correctness proofs.* In MCE problem, the correctness is evident as each maximal clique can be enumerated starting from the vertex with the smallest order in this clique. However, for MMCE problem, we need to theoretically prove Lemma 5.1 and 5.2 to confirm the correctness.

Lemma 5.4. *Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$,* `POMA` *costs $O(\Delta)$ space, where Δ is the number of motif instance of $\mathcal{M}$ in $\mathcal{H}$.*

Proof sketch. In the process of enumerating all maximal M-cliques, all the motif instances would be stored in the memory. □

6 PIVOT-BASED ENUMERATION TECHNIQUES

As discussed in Section 3, the classical pivot principle cannot be directly applied to the MMCE problem. In this section, we present a novel pivot principle for accelerating the MMCE process.

6.1 The motif-pivot principle

Recall that in the MCE on homogeneous graphs, we can just use the neighbors of the pivot to find the vertices that can be pruned. However, in our MMCE, pruning neighbors from the candidate set is not correct. To figure out the vertices to prune, we introduce the concept of M-clique precedence.

Definition 9 (M-clique precedence $\prec$). *Given an HIN $\mathcal{H}$, a motif $\mathcal{M}$, an M-clique R, a candidate set C of vertices, and two vertices $u, v \in C$, v has M-clique precedence over u, denoted by $v \prec u$, if the following conditions are met:*

- *$R \cup \{u\}$ is an M-clique due to $u \in C$, and $R \cup \{u, v\}$ is also an M-clique;*
- *if there exists an edge $(x, y) \in \mathcal{E}_{\mathcal{M}}$ satisfying $\psi(x) = \psi(u) \wedge \psi(y) = \psi(v)$, we have $(u, v) \in \mathcal{E}$.*

Let $\mathcal{I}(u)$ denote the set of vertices in C that have M-clique precedence over u, i.e., $\mathcal{I}(u)$=$\{v \in C | v \prec u, u \in C\}$. In each recursion of MMCE, the pivot vertex p should be able to enlarge the M-cliques formed in $R \cup \mathcal{I}(p)$, which will be further explained in Lemma 6.1. To ensure this, we introduce the concept of candidate pivot set.

Definition 10 (Candidate pivot set). *Given an HIN $\mathcal{H}$, a motif $\mathcal{M}$, and an M-clique R, we define $\mathcal{P}_R$ as a candidate pivot set of R if, for every vertex v in $\mathcal{P}_R$, the condition $1 + |\mathcal{D}_v^R| > t$ holds, where $\mathcal{D}_v^R$ denotes the set of vertices in R dominated by v, and t represents the number of vertices in $\mathcal{M}$ with type $\psi(v)$.*

It is noted that $\mathcal{P}_R$ can be empty, indicating that we cannot prune any search space in this case. However, in practice, the case when

$\mathcal{P}_R = \emptyset$ is not common, and as shown in our experiments, the pivot technique can significantly avoid redundant computation. Based on the definitions above, we give the following lemma:

Lemma 6.1. *For any M-clique R and a vertex u in $\mathcal{P}_R$, any maximal M-clique $R' \supseteq R$ must include either u or one of the vertices that are not M-clique precedence over u, as otherwise, R' could be enlarged by adding u.*

Proof sketch. We assume that a maximal M-clique R' does not contain vertex u or any vertices that are not M-clique precedence over u. In other words, R' satisfies $R \subseteq R' \subseteq R \cup \mathcal{I}(u)$. Then, $R' \cup \{u\}$ is also an M-clique, since we each match-set T in $R' \cup \{u\}$ containing u is also a motif instance. □

Next, based on Lemma 6.1, we formally introduce our motif-pivot principle. Given a maximal M-clique, there are only two cases: it either contains the pivot vertex p or one of the vertices not in $\mathcal{I}(p)$. If an M-clique does not contain p, then it can only be enlarged by vertices from $C \backslash \mathcal{I}(p)$. In this situation, the algorithm can safely prune vertices in $\mathcal{I}(p)$ from C.

6.2 Efficiently approximating $\mathcal{I}(p)$

Based on the discussion above, for each pivot vertex p, we need to compute its M-clique precedence set $\mathcal{I}(p)$. However, as stated in [39], determining whether a vertex can be incorporated into an existing M-clique to form a larger one is an NP-hard problem. Consequently, computing $\mathcal{I}(p)$ is also an NP-hard problem, and it is not feasible to directly obtain. Luckily, we can quickly find a subset of $\mathcal{I}(p)$ via the dominance relationship. However, this only allows a limited number of vertices to be included in $\mathcal{I}(p)$. To better approximate $\mathcal{I}(p)$, we propose additional conditions to include more vertices in $\mathcal{I}(p)$ in the following lemma.

Lemma 6.2 (Approximating $\mathcal{I}(p)$). *Given an HIN $\mathcal{H}$, a motif $\mathcal{M}$, an M-clique R and a pivot vertex p, a vertex v in C that satisfies at least one of the following conditions is M-clique precedence over p:*

- ***Condition 1.*** $1 + |\mathcal{D}_v^{R\cup\{p\}}| > t$.
- ***Condition 2.*** *$\psi(p) = \psi(v)$, and there is only one vertex in $\mathcal{M}$ with type $\psi(p)$.*
- ***Condition 3.*** *Let $L = \{x | x \in R' \wedge \psi(x) = \psi(v)\}$, and $\left(\bigcup_{x\in L} (\mathcal{N}(x) \cap R')\right) \subseteq (\mathcal{N}(v) \cap R')$, where $R' = R \cup \{p\}$;*

Proof sketch. The Condition 1 is guaranteed by Lemma 6.4 in [39]. For Condition 2, since all new match-sets will contain v but not p, and $R \cup \{v\}$ is an M-clique, $R \cup \{p, v\}$ is an M-clique. For Condition 3, whether the match-set T in $R' \cup \{v\}$ includes v but not p, or both p and v, T is a motif instance. □

The above lemma can be easily incorporated into our pivot principle, as discussed in Section 6.1. Specifically, in each recursive call, for the current M-clique R, the algorithm first selects a pivot vertex p from the $\mathcal{P}_R$. To compute $\mathcal{I}(p)$, we applied Lemma 6.2 to quickly detect an approximated subset of $\mathcal{P}(p)$ without subgraph isomorphism checking. The following example illustrates that Lemma 6.2 can better approximate $\mathcal{I}(p)$ than the dominance relationship.

Example 5. *Consider the HIN and motif in Figure 2. Assume that $R = \{a_1, p_1, p_2\}$, $C = \{a_2, a_3, p_3, p_4, p_5\}$, and $p = a_2$. Here, we need to compute $\mathcal{I}(p)$. However, if we only rely on the dominance relationship (i.e., Condition 1 of Lemma 6.2), none of the vertices in C satisfy this condition. According to Condition 2, a_3 can be appended into $\mathcal{I}(p)$, since there is only one author in the motif. In addition, based on Condition 3, p_3 can also be added to $\mathcal{I}(p)$, as the neighbors of p_1 and p_2 are also neighbors of p_3.*

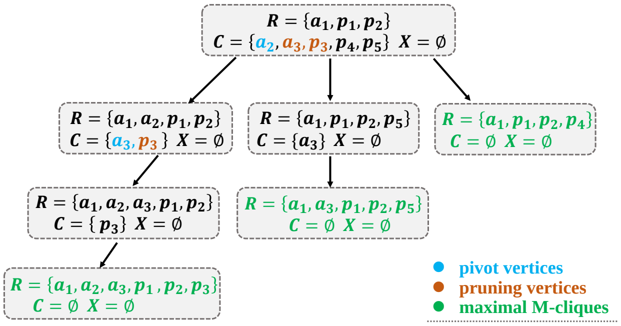

$R = \{a_1, p_1, p_2\}$
$C = \{a_2, a_3, p_3, p_4, p_5\}$ $X = \emptyset$
$R = \{a_1, a_2, p_1, p_2\}$
$C = \{a_3, p_3\}$ $X = \emptyset$
$R = \{a_1, p_1, p_2, p_5\}$
$C = \{a_3\}$ $X = \emptyset$
$R = \{a_1, p_1, p_2, p_4\}$
$C = \emptyset$ $X = \emptyset$
$R = \{a_1, a_2, a_3, p_1, p_2\}$
$C = \{p_3\}$ $X = \emptyset$
$R = \{a_1, a_3, p_1, p_2, p_5\}$
$C = \emptyset$ $X = \emptyset$
$R = \{a_1, a_2, a_3, p_1, p_2, p_3\}$
$C = \emptyset$ $X = \emptyset$
● pivot vertices
● pruning vertices
● maximal M-cliques

Figure 5: An enumeration tree for the HIN in Figure 2.

To compute $\Gamma(p)$ exactly, we need to check whether a vertex v can expand an M-clique via VF3, which invokes a subgraph isomorphism algorithm. There is no constant factor approximation algorithm for it [66]. Our approximation algorithm can quickly detect a subset of $\Gamma(p)$, which costs $O(n \cdot d_{max})$ time to check whether a vertex can be added into $\Gamma(p)$, while the exact algorithm requires $O(n!)$ time in the worst case. Besides, the approximate $\mathcal{I}(u)$ can also be used to efficiently update C and X. Specifically, in Algorithm 1, after adding vertex u into R, we can compute the $\mathcal{I}(u)$ based on Lemma 6.2. Then, the vertices in $\mathcal{I}(u) \cap C$ and $\mathcal{I}(u) \cap X$ can safely be added to C' and X', respectively, without any extra checks.

6.3 A pivot-based enumeration algorithm

In this subsection, we develop a pivot-based enumeration algorithm, called MP-MMC, that fully integrates the proposed motif-pivot principle, as shown in Algorithm 5. Specifically, we first compute the candidate pivot set $\mathcal{P}_R$ of R, and then select a pivot vertex from $\mathcal{P}_R$ (lines 3-4). Note that if $\mathcal{P}_R$ has multiple vertices, we select the one with the maximum degree as the pivot vertex. Next, we compute the M-clique precedence set $\mathcal{I}(p)$ based on the Lemma 6.2 (lines 5-8). Then, we iterate over the vertices in $C \backslash \mathcal{I}(p)$ to expand the current M-clique R (lines 9-13). Example 6 further illustrates how MP-MMC reduces redundant computation.

Algorithm 5: MP-MMC (R, C, X)

```
   input  : An HIN $\mathcal{H}$ and a motif $\mathcal{M}$
   output : All maximal M-cliques of $\mathcal{M}$ in $\mathcal{H}$
1  if $C \cup X = \emptyset$ then report $R$ as a maximal M-clique;
2  $\mathcal{P}_R \leftarrow$ compute the candidate pivot set of $R$;
3  $p \leftarrow$ select a pivot vertex from $\mathcal{P}_R$;
4  $R' \leftarrow R \cup \{p\}$;
5  foreach $u \in C$ do
6      if $\exists (x, y) \in \mathcal{E}_\mathcal{M}, \psi(x) = \psi(p), \psi(y) = \psi(u)$ then
7          if $(p, u) \notin \mathcal{E}$ then continue ;
8      if Lemma 6.2 is satisfied then $\mathcal{I}(p) \leftarrow \mathcal{I}(p) \cup \{u\}$ ;
9  foreach $u \in C \backslash \mathcal{I}(p)$ do
10     $C' \leftarrow$ Refine$(R \cup u, C)$;
11     $X' \leftarrow$ Refine$(R \cup u, X)$;
12     MP-MMC$(R \cup u, C', X')$;
13     $X \leftarrow X \cup \{u\}$;
```

Example 6. *Given an HIN $\mathcal{H}$ and a motif $\mathcal{M}$ in Figure 2, assume that $\Gamma = \{a_1, p_1, p_2\}$. The initial parameters of* MP-MMC *are $R = \{a_1, p_2, p_2\}$, $C = \{a_2, a_3, p_3, p_4, p_5\}$, and $X = \emptyset$ respectively. Assume that the pivot vertex of* MP-MMC *is a_2 (marked in blue in Figure 5). Then, $\{a_3, p_3\}$ (marked in orange in Figure 5) can be pruned from C, as all the vertices are M-clique precedence over a_2. By our pivot technique, only vertices in $\{a_2, p_4, p_5\}$ are used to expand the current M-clique R. When R is expanded with the vertex a_2, the candidate set is updated to $\{a_3, p_3\}$, which will be checked in the next recursive call. In the next recursion, suppose the selected pivot vertex is a_3. Then, p_3 can be pruned from C. Thus, the algorithm only picks a_3 to expand R. The procedure continues until all vertices in C of the top recursion have been processed. The enumeration tree is shown in Figure 5.*

textbfDiscussions. Our motif-pivot principle is different from the pivot strategies of existing MCE methods [10, 30, 61, 78] in two aspects:

- *Pivot-based pruning strategies:* In MCE algorithms, when a pivot p is selected, the neighbours of p can be pruned from C. However, in MMCE, such a pruning set no longer exists since a maximal M-clique may not contain the pivot p, but only contain its neighbours. Thus, we introduce the M-clique precedence relationship, then use the M-clique precedence set of p, $\mathcal{I}(p)$, as the pruning set, and finally propose an efficient algorithm to quickly approximate $\mathcal{I}(p)$.
- *Pivot selection processes.* In existing MCE algorithms, the pivot vertex is directly selected from the candidate set or not set. However, in MMCE problem, for an M-clique R, we can only select the pivot vertex from the candidate pivot set of R. In Lemma 6.1, we theoretically prove that when a pivot vertex p is selected, the M-cliques in $R \cup \mathcal{I}(p)$ can be enlarged by p.

7 EXPERIMENTS

We now present the experimental results. Section 7.1 discusses the setup. We discuss the efficiency results in Sections 7.2 and 7.3. We present the two case studies in Section 7.4.

7.1 Setup

Datasets. We use five real HINs: *Instacart*[2] [39], *WordNet*[3] [75], *DBLP*[4] [32], *DBpedia*[5] [91], and *Freebase*[6] [91]. Their statistics, including the numbers of vertices, edges, vertex types, edge types, and the degeneracy values δ, are reported in Table 3. *Instacart* is a co-purchasing network, where each product has a type showing its category (e.g., "personal care" and "beverages"), and each edge between two products means they have been purchased together more than 200 times. *WordNet* is a large lexical database, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. *DBLP* includes publication records in computer science areas, and the vertex types are authors, papers, venues and topics. *DBpedia* contains the data extracted from wikipedia infoboxes using the mapping-based extraction. *Freebase* contains all the entities and relations in the music domain.

Table 3: Datasets used in our experiments.

Dataset	Vertices	Edges	Vertex types	Edge types	δ	Motifs
Instacart	49,688	12,770	21	237	30	100
WordNet	76,853	240,798	5	25	10	100
DBLP	881,039	2,247,195	4	7	14	100
DBpedia	8,970,120	71,403,844	414	79,397	52	100
Freebase	347,463,729	1,110,001,528	10, 801	620,307	168	100

Queries. For each dataset, to generate a motif, we perform a random walk on the data graph to obtain a connected subgraph, following the approach in [7, 75]. For evaluation purposes, we generate five motif sets for each dataset, which contain 100 motifs with 3, 4, 5, 6, and 7 vertices respectively, in line with the range of motif sizes (from 3 to 7) typically encountered in various real-world applications [35, 59, 87, 92]. The default motif size is 4. The structures of motifs used in our paper are shown in our technical report [93]. For example, a motif with two users liking the same genre of movie or item shows value for recommendation purposes, and it can be found in both DBpedia and Freebase [60, 82, 94]. We implement all the algorithms in C++ with STL used and run experiments on a machine having an Intel(R) Xeon(R) Gold 6338R 2.0GHz CPU and 512GB of memory, with Ubuntu installed. If an algorithm cannot finish in 30 days, we mark its runtime as **INF**.

7.2 Overall efficiency results

In this section, we compare the efficiency of POMA and META.

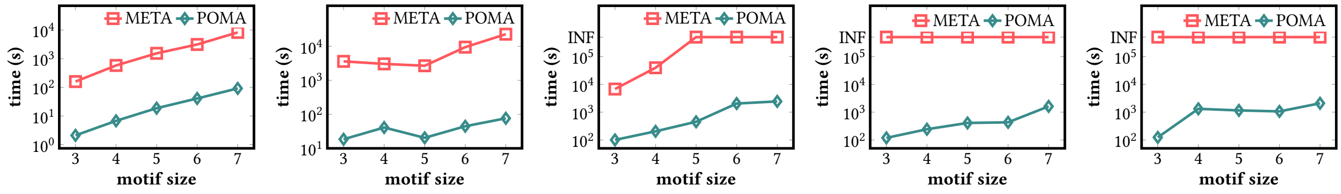
1. Effect of motif sizes. Figure 6 compares the efficiency of these two algorithms by varying motif sizes. Clearly, POMA is up to three orders of magnitude faster than META, because it not only ensures that a maximal M-clique will not be enumerated multiple times but also dramatically reduces the search space, while META includes numerous redundant computation. Besides, as motif size becomes larger, the running time of all algorithms generally increases, since a larger motif means checking subgraph isomorphism is more time-consuming. POMA, on average, only requires a few tens of seconds to enumerate all maximal M-cliques for motifs with three vertices. In addition, on the largest two datasets, META could not complete its run for any motif within one month.

2. Scalability test. For each HIN, we first randomly select 20%, 40%, 60%, 80%, and 100% of its edges and then obtain five sub-HINs induced by these edges, respectively. We then report the average efficiency of POMA and META on these sub-HINs in Figure 7. The time costs of both POMA and META scale almost exponentially with the number of vertices in the graph, but the growth rate of the curve of POMA is smaller, so POMA exhibits better scalability and META.

3. Comparing the search space of META and POMA. To measure the size of search space, we count the total number of branches that need to be enumerated by META and POMA (i.e., the number of nodes in the recursive tree), we use "N/A" to denote that the algorithm could not be finished within one month. Figure 8 shows the results on five datasets. Clearly, POMA is more effective for reducing the search space than META, thus achieving higher efficiency.

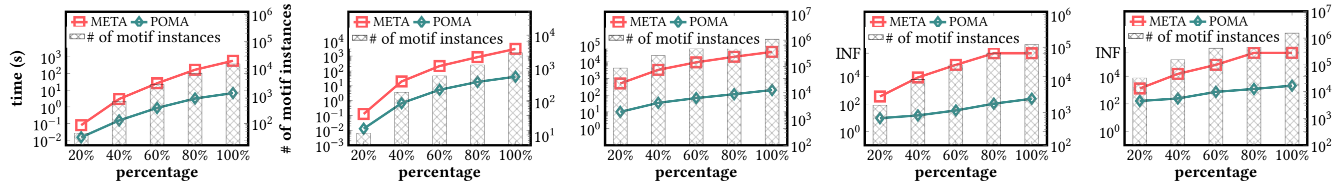
4. Comparing the space usage of META and POMA. We examine the memory usage of META and POMA on all datasets with motif size 4, reporting their results in Figure 9. The META and POMA have the same space complexity $O(\Delta)$, where Δ denotes the number of motif instances. Nevertheless, in practice, POMA usually takes slightly more memory than META, since it adapts a string hash operation in the selectCand step. This is an optimization technique at the

[2] https://www.instacart.com/datasets/grocery-shopping-2017
[3] https://wordnet.princeton.edu/
[4] https://www.aminer.cn/citation
[5] https://wiki.dbpedia.org/Datasets
[6] http://freebase-easy.cs.uni-freiburg.de/dump/



(a) Instacart (b) WordNet (c) DBLP (d) DBpedia (e) Freebase

Figure 6: Efficiency results of MMCE algorithms.



(a) Instacart (b) WordNet (c) DBLP (d) DBpedia (e) Freebase

Figure 7: Scalability test for MMCE algorithms.

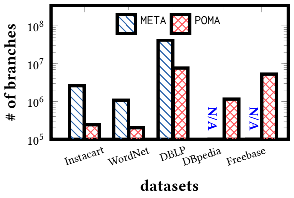


Figure 8: Search space.

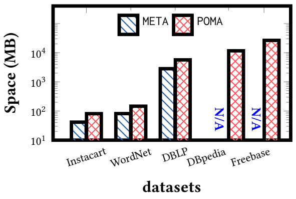


Figure 9: Memory usage.

implementation level, which will increase the constant size of the space required by POMA, but will not increase its space complexity.

7.3 Detailed analysis of POMA

We extensively evaluate and analyze POMA from different angles.

1. Time cost of different steps in POMA. Recall that POMA sequentially performs the following five steps: (1) reducing the original HIN (reduceHIN), (2) computing all the motif instances (VF3 [12]), (3) identifying the candidate sets selectCand, (4) dividing the candidate sets and not vertex sets Order-Divide, and (5) performing the recursive search (MP-MMC). Figure 10 shows the time cost of these five steps (assuming the graph has been loaded into memory) on five datasets. We see that MP-MMC is the most computationally expensive step on all datasets. On DBpedia and Freebase datasets, the time cost of reduceHIN is significantly larger than that on other datasets. This is because these two datasets contain a much larger number of vertex types, allowing more vertices to be pruned since each motif includes a limited number of diverse vertex types. Besides, running the VF3 algorithm and Order-Divide only account for a relatively small portion of the total time during the execution of POMA, so we should focus on optimizing other steps.

2. Ablation study. To evaluate the effect of four key steps, we design four variants by adding reduceHIN, selectCand, ordering framework, and motif-pivot into META, which are denoted by META+R, META+C, META+O, and META+P respectively. We then run META and these variants on all datasets and report the efficiency results in Figure 11. We can see that POMA significantly outperforms the baseline META and the other variants. Moreover, these four variants are faster than META, further substantiating the efficacy of our four proposed techniques, especially on larger HINs. We also design other four variants by removing the four key steps above from POMA, and report their results in our technical report [93].

3. Effect of graph reduction technique. We present the average numbers of vertices and edges before and after the graph reduction for different motifs across all datasets. Clearly, our graph reduction method can substantially reduce the size of the original HIN. The details are shown in our technical report [93].

4. Effect of different vertex ordering strategies. In this experiment, we consider two widely used vertex orders: degree order and degeneracy order. Clearly, both orders significantly outperform the case without vertex ordering, and the degeneracy order usually achieves better performance, and results are shown in [93].

5. Efficiency of the $\mathcal{I}(p)$ approximation algorithm. To show the efficiency of the $\mathcal{I}(p)$ approximation algorithm, we denote the POMA that does not employ the algorithm proposed in Section 6.2 as POMA*. Then, we report the sum running time and the number of pruning vertices of POMA and POMA* on all datasets in Table 4. It is noted that if the algorithm cannot finish within three days, we mark its running time as ≥ 259,200s. Clearly, POMA significantly outperforms POMA* on larger HINs, as it obviates the need for subgraph isomorphism checking. More specifically, on the two largest datasets (DBpedia and Freebase), POMA* cannot finish within three days. Besides, from a theoretical perspective, POMA can only prune a subset of vertices that are pruned by POMA*. However, in practice, we observe that POMA can prune nearly the same number of vertices as POMA*, which underscores the practical effectiveness of POMA.

Dataset	Instacart	WordNet	DBLP	DBpedia	Freebase
POMA*	1,804.5s	1,877.1s	6,262.3s	≥ 259,200s	≥ 259,200s
POMA	136.2s	827.1s	4,117.9s	4,866.3s	12,097.8s
Speedup	13.2 ×	14.4 ×	1.5 ×	≥ 53 ×	≥ 21.4 ×
POMA*	36.5	117.0	6.37	–	–
POMA	35.6	117.0	6.37	161.8	50.0
ratio	97.5 %	100.0 %	100.0 %	–	–

Table 4: Comparing different pivot techniques.

6. The parallel version of POMA. Recall that in POMA, we need to enumerate the maximal M-cliques containing each motif instance, so the enumeration for all the motif instances can be performed

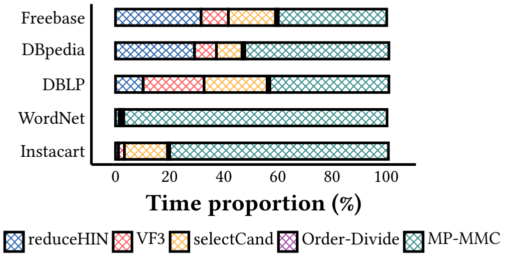


Figure 10: Proportion of the time cost of each step in POMA**.**

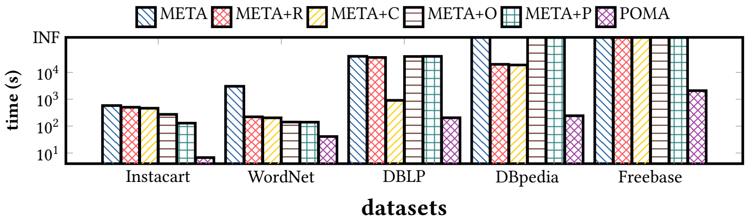
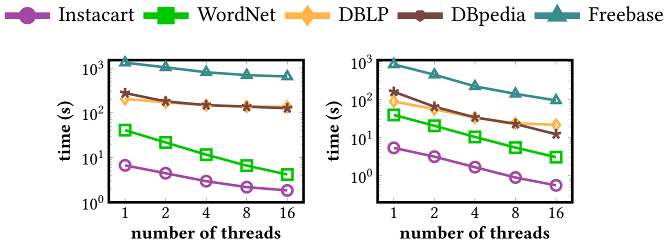


Figure 11: Efficiency comparison of MMCE algorithms.



(a) Efficiency of POMA (b) Efficiency of MP-MMC in POMA

Figure 12: Effect of the number of threads.

	Instacart	**WordNet**	**DBLP**	**DBpedia**	**Freebase**
γ	0.15 s	0.19 s	46.13 s	19.48 s	6.52 s
α	3.25 us	6.34 us	3.24 us	1.03 us	0.17 us

Table 5: Actual running time of γ and α.

in parallel. Specifically, we have implemented a parallel version of POMA by paralleling the MP-MCC step, denoted by POMA-Par, in which each thread is scheduled for enumerating the maximal M-cliques containing a specific motif instance. Figure 12(a) shows the time cost of the five steps of POMA and POMA-Par, and Figure 12(b) only shows the running time of MP-MMC step, by varying the number of threads from 1 to 16 across all datasets. Clearly, with an increasing number of threads, the overall runtime of POMA-Par exhibits a reduction, and if MP-MCC takes up a large portion of the whole running time, POMA-Par shows a strong parallel scalability. For example, on WordNet dataset, using 16 threads allows POMA-Par to achieve self-speedups of 13 times.

7. Statistical of the hyper-parameters. We mainly use five parameters, i.e., γ, α, Δ, δ, and β. For γ and α, they denote the time costs of computing all motif instances and checking whether a vertex can expand to M-clique respectively. Table 5 reports their actual running time costs. For Δ, δ, and β, they represent the average number of motif instances in the HIN, the degeneracy value of the graph, and the degeneracy multiplying the number of vertex types, respectively. The values of δ are included in Table 3. We report the values of Δ in Figure 7, which range from a few to two million. We observe that the running time of META and POMA on all datasets is proportional to the motif instance numbers since we need to enumerate the maximal M-cliques for each motif instance, which is aligned with our complexity analysis in Section 5.3.

7.4 Case studies

In this section, we present two case studies on real-world HINs.

Table 6: Community quality analysis on Instacart.

Method	Diameter	Similarity			
		Babies	Household	Breakfast	Frozen
R-com	3.45	0.15	0.22	0.22	0.19
POMA	**2.01**	**0.45**	**0.53**	**0.49**	**0.42**

(1) Community detection. As the SOTA method of HIN community detection, R-com [43] takes input from a set of relational constraints which can be regarded as a motif, and finds communities with multiple vertex types. In this experiment, we compare the quality of communities detected by POMA and R-com on Instacart and DBLP datasets. The Specifically, we first randomly select 20 motifs with each having 4 vertices. Then, for each motif, we run POMA to detect the corresponding maximal M-cliques, and R-com to get the communities to adhere to the relation constraints in the motif. Finally, we compare the quality of communities in terms of community member similarity and closeness of communities, which are used for measuring community quality [31, 41, 43]. For the former one, we compute the Jaccard similarity for different types of vertices by following the method in [43]; for the latter one, we compute the diameter by following the method in [32, 41]. As shown in Table 6, we observe that the communities of POMA achieve both higher similarity values and smaller diameters than those of R-com on Instacart dataset. The results on DBLP dataset are shown in our technical report [93]. Thus, POMA is able to detect communities with higher quality.

Table 7: Statistics of datasets for bundle recommendation.

Dataset	# of vertices			# of edges	
	#User (U)	#Item (I)	#Bundle (B)	#U-I	#U-B
Netease	18,528	123,628	22,864	1,128,065	302,303
Youshu	8,039	32,770	4,771	138,515	51,377

Table 8: The Recall@K values on BGCN **and** BGCN-M**.**

Dataset	**Method**	Recall@10	Recall@20	Recall@30	Recall@40
Netease	BGCN	0.0369	0.0642	0.0845	0.1013
	BGCN-M	**0.0391**	**0.0650**	**0.0863**	**0.1033**
Youshu	BGCN	0.1596	0.2410	0.2984	0.3416
	BGCN-M	**0.1600**	**0.2463**	**0.3018**	**0.3453**

Table 9: The NDCG@K values on BGCN **and** BGCN-M**.**

Dataset	**Method**	NDCG@10	NDCG@20	NDCG@30	NDCG@40
Netease	BGCN	0.0202	0.0274	0.0321	0.0356
	BGCN-M	**0.0209**	**0.0278**	**0.0327**	**0.0362**
Youshu	BGCN	0.0934	0.1165	0.1303	0.1398
	BGCN-M	**0.0955**	**0.1198**	**0.1331**	**0.1427**

(2) Bundle recommendation. Our proposed algorithm POMA can be used to enhance the performance of the SOTA bundle recommendation method BGCN, which aims to recommend a bundle of items to a single user [11, 13, 38]. BGCN [13] trains a Graph Neural Networks (GNN)-based recommendation model using three networks, including the user-bundle interaction network, user-item interaction network, and bundle-item affiliation network, and then predict whether a user will interact with a certain bundle. However, due to exposure bias and item diversity imbalance [16, 86], the bundle-item affiliation network cannot capture the precise relationships between bundles and items, which may cause some items to be overlooked and not considered for the bundle, leading to sparse connections between bundles and items.

To alleviate the sparsity issue above, we propose to detect the maximal M-cliques from the network of items, and then use these maximal M-cliques to argument the bundle-item affiliation network, thereby enhancing the performance of BGCN. Specifically, we first build a co-purchasing network between items, where each edge between two items means they have been purchased together more than 5 times. Then, we use POMA to enumerate maximal M-cliques by using motifs formed by items in the bundles. Next, in the bundle-item affiliation network, we link items in the same maximal M-cliques to their respective bundles. Finally, we run BGCN on the augmented networks, and denote this approach by BGCN-M.

We have compared BGCN and BGCN-M on two real-world datasets, i.e., Netease and Youshu, whose statistics are shown in Table 7. The former one was collected [11] from the largest music platform [7] in China, which enables users to bundle songs with a specific theme, while the latter one was collected by [19] from Youshu platform, a Chinese book review site [8], where each bundle is a list of books selected by some users. We use Recall@K and NDCG@K [38, 86] to judge the performance of the ranking list, where Recall@K measures the ratio of test bundles that have been contained by the top-K ranking list, while NDCG@K complements Recall by assigning higher scores to the hits at higher positions of the list. As shown in Tables 8 and 9, BGCN-M significantly outperforms BGCN in terms of Recall@K and NDCG@K. This is because it uses maximal M-cliques to augment the bundle-item affiliation network, which alleviates the sparsity issue by more precisely capturing the relationships between bundles and items.

8 RELATED WORKS

This section mainly reviews the existing works of maximal clique enumeration (MCE) in both homogeneous and heterogeneous graphs.

• **MCE in homogeneous graphs.** As a fundamental problem for graph data analysis, the MCE problem has garnered plenty of research attention. The well-known solutions to the MCE problem are the classic BK algorithm [10] and its pivot-based variants [10, 30, 61, 78]. The BK algorithm [10], proposed by Bron and Kerbosch in 1973, enumerates maximal cliques in a recursive backtracking manner. Tomita et al. [78] showed that the time overhead to the MCE problem is optimal in the worst case using their pivot technique. Eppstein et al. [30] further derived a tighter worst-case time complexity for MCE based on the degeneracy order [54]. Naudé et al. [61] revised the pivot algorithm by refining the pivot selection process. Besides, the I/O efficient algorithms [22, 23], distributed algorithms [89], parallel [28, 70] algorithms, and output-sensitive algorithms [15, 24, 57] have also been developed for the MCE problem. In addition, many variants of the MCE problem have been formulated for different types of homogeneous graphs, such as temporal [81], signed [21, 51], uncertain networks [25, 52], and efficient solutions have also been studied.

• **MCE in heterogeneous graphs.** Generally, the heterogeneous graphs can be classified as bipartite graphs and other general HINs [31]. On bipartite graphs, the problem of maximal bi-clique enumeration (MBCE) has received much attention. David Eppstein [29] developed a linear MBCE algorithm for any graph of bounded arboricity. In [68], the MBCE problem is solved by exhaustively listing subsets of vertices in one layer, subsequently identifying vertices in the other layer as their common neighbors, and then identifying the bicliques. Li et al. [49] applied efficient algorithms for mining closed patterns to the MBCE problem. Zhang et al. [96] proposed an algorithm iMBEA by fusing backtracking with a branch-and-bound framework. Recently, several approaches [1, 18] have employed the pivot principle to speed up the MBCE, and a special case of the MBCE problem, called maximum biclique search, has attracted much attention [17, 56, 71]. Besides, MCE on multi-partite graphs has also been studied [63]. For general HINs, Hu et al. [39] introduced the concept of motif-clique and proposed the META algorithm for the MMCE problem. However, META cannot efficiently process large HINs, calling for a faster solution.

In addition, many other works also find maximal solutions, such as maximal frequent itemset [5, 6, 34, 36, 37, 79], maximal independent set [33, 45, 55, 77, 88], maximal k-plex [2, 26, 42, 85], etc.

9 CONCLUSIONS

In this paper, we investigate the problem of efficient maximal motif-clique enumeration (MMCE) over large HINs. The existing MMCE algorithm, following the classic BK algorithm, which explores all the possible subgraphs of a maximal motif-clique (M-clique) and checks whether each maximal M-clique has been enumerated at each recursive step, is very time-consuming. To improve the efficiency of MMCE, we introduce an order-based framework and propose a pivot-based pruning strategy. We further propose a series of optimization techniques. Our experimental results on five real large HINs show that our algorithm achieves up to three orders of magnitude faster than the state-of-the-art algorithm. In the future, we will develop a more generic definition of M-clique, that can incorporate various kinds of graph isomorphism, such as subgraph monomorphism [67], supergraph isomorphism [72], and graph homomorphism [66]. We will also study how to efficiently enumerate maximal M-cliques efficiently on large dynamic HINs since many real-world HINs are evolving over time.

ACKNOWLEDGMENTS

This work was supported in part by NSFC under Grant 62102341 and 62302421, Guangdong Talent Program under Grant 2021QN02X826, and Shenzhen Science and Technology Program under Grants JCYJ20220530143602006 and ZDSYS 20211021111415025. This work was supported by Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010001), and Hong Kong RGC grants No. GRF/12201923 and CRF/C2003-23Y. This work was also supported by Basic and Applied Basic Research Fund in Guangdong Province under Grant 2023A1515011280, and the Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen.

[7] https://music.163.com

[8] http://www.yousuu.com

REFERENCES

[1] Aman Abidi, Rui Zhou, Lu Chen, and Chengfei Liu. 2020. Pivot-based Maximal Biclique Enumeration.. In *IJCAI*. 3558–3564.

[2] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V Hicks. 2011. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research* 59, 1 (2011), 133–142.

[3] Vladimir Batagelj and Matjaz Zaversnik. 2003. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[4] Christoph Benzmüller, Chad E Brown, and Michael Kohlhase. 2004. Higher-order semantics and extensionality. *The Journal of Symbolic Logic* 69, 4 (2004), 1027–1088.

[5] Thomas Bernecker, Reynold Cheng, David W Cheung, Hans-Peter Kriegel, Sau Dan Lee, Matthias Renz, Florian Verhein, Liang Wang, and Andreas Zuefle. 2013. Model-based probabilistic frequent itemset mining. *Knowledge and Information Systems* 37 (2013), 181–217.

[6] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Zuefle. 2009. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 119–128.

[7] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. 2016. Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*. 1199–1214.

[8] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. 2005. Statistical analysis of financial networks. *Computational statistics & data analysis* 48, 2 (2005), 431–443.

[9] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2017. Counting graphlets: Space vs time. In *Proceedings of the tenth ACM international conference on web search and data mining*. 557–566.

[10] Coenraad Bron and Joep Kerbosch. 1973. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM* 16, 9 (1973), 575–576.

[11] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding factorization models for jointly recommending items and user generated lists. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 585–594.

[12] Vincenzo Carletti, Pasquale Foggia, Alessia Saggese, and Mario Vento. 2017. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 804–818.

[13] Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Bundle recommendation with graph convolutional networks. In *Proceedings of the 43rd international ACM SIGIR conference on Research and development in Information Retrieval*. 1673–1676.

[14] Lijun Chang. 2019. Efficient maximum clique computation over large sparse graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 529–538.

[15] Lijun Chang, Jeffrey Xu Yu, and Lu Qin. 2013. Fast maximal cliques enumeration in sparse graphs. *Algorithmica* 66 (2013), 173–186.

[16] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[17] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2021. Efficient exact algorithms for maximum balanced biclique search in bipartite graphs. In *Proceedings of the 2021 International Conference on Management of Data*. 248–260.

[18] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2022. Efficient maximal biclique enumeration for large sparse bipartite graphs. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1559–1571.

[19] Liang Chen, Yang Liu, Xiangnan He, Lianli Gao, and Zibin Zheng. 2019. Matching user with item set: Collaborative bundle recommendation with deep attention network.. In *IJCAI*. 2095–2101.

[20] Xiaowei Chen, Yongkun Li, Pinghui Wang, and John CS Lui. 2016. A General Framework for Estimating Graphlet Statistics via Random Walk. *Proceedings of the VLDB Endowment* 10, 3 (2016).

[21] Zi Chen, Long Yuan, Xuemin Lin, Lu Qin, and Jianye Yang. 2020. Efficient maximal balanced clique enumeration in signed networks. In *Proceedings of The Web Conference 2020*. 339–349.

[22] James Cheng, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. 2011. Finding maximal cliques in massive networks. *ACM Transactions on Database Systems (TODS)* 36, 4 (2011), 1–34.

[23] James Cheng, Linhong Zhu, Yiping Ke, and Shumo Chu. 2012. Fast algorithms for maximal clique enumeration with limited memory. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1240–1248.

[24] Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. 2016. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[25] Qiangqiang Dai, Rong-Hua Li, Meihao Liao, Hongzhi Chen, and Guoren Wang. 2022. Fast maximal clique enumeration on uncertain graphs: A pivot-based approach. In *Proceedings of the 2022 International Conference on Management of Data*. 2034–2047.

[26] Qiangqiang Dai, Rong-Hua Li, Hongchao Qin, Meihao Liao, and Guoren Wang. 2022. Scaling Up Maximal k-plex Enumeration. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 345–354.

[27] Maximilien Danisch, Oana Balalau, and Mauro Sozio. 2018. Listing k-cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference*. 589–598.

[28] Apurba Das, Seyed-Vahid Sanei-Mehri, and Srikanta Tirthapura. 2018. Shared-memory parallel maximal clique enumeration. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. IEEE, 62–71.

[29] David Eppstein. 1994. Arboricity and bipartite subgraph listing algorithms. *Information processing letters* 51, 4 (1994), 207–211.

[30] David Eppstein, Maarten Löffler, and Darren Strash. 2013. Listing all maximal cliques in large sparse real-world graphs. *Journal of Experimental Algorithmics (JEA)* 18 (2013), 3–1.

[31] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (2020), 353–392.

[32] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 6 (2020), 854–867.

[33] Mohsen Ghaffari. 2016. An improved distributed algorithm for maximal independent set. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 270–277.

[34] Gösta Grahne and Jianfei Zhu. 2005. Fast algorithms for frequent itemset mining using fp-trees. *IEEE transactions on knowledge and data engineering* 17, 10 (2005), 1347–1362.

[35] Saket Gurukar, Sayan Ranu, and Balaraman Ravindran. 2015. Commit: A scalable approach to mining communication motifs from dynamic networks. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 475–489.

[36] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery* 15, 1 (2007), 55–86.

[37] Jiawei Han, Jian Pei, and Hanghang Tong. 2022. *Data mining: concepts and techniques*. Morgan kaufmann.

[38] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[39] Jiafeng Hu, Reynold Cheng, Kevin Chen-Chuan Chang, Aravind Sankar, Yixiang Fang, and Brian YH Lam. 2019. Discovering maximal motif cliques in large heterogeneous information networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 746–757.

[40] Xiaocheng Hu, Yufei Tao, and Chin-Wan Chung. 2013. Massive graph triangulation. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. 325–336.

[41] 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*. 1311–1322.

[42] Said Jabbour, Nizar Mhadhbi, Badran Raddaoui, and Lakhdar Sais. 2022. A declarative framework for maximal k-plex enumeration problems. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS*.

[43] Xun Jian, Yue Wang, and Lei Chen. 2020. Effective and efficient relational community detection and search in large dynamic heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 10 (2020), 1723–1736.

[44] Rui Jiang, Zhidong Tu, Ting Chen, and Fengzhu Sun. 2006. Network motif identification in stochastic networks. *Proceedings of the National Academy of Sciences* 103, 25 (2006), 9404–9409.

[45] David S Johnson, Mihalis Yannakakis, and Christos H Papadimitriou. 1988. On generating all maximal independent sets. *Inform. Process. Lett.* 27, 3 (1988), 119–123.

[46] Benoit Jolicoeur, Erin E Chapman, Alison Thompson, and William D Lubell. 2006. Pyrrole protection. *Tetrahedron* 62, 50 (2006), 11531–11563.

[47] Jure Leskovec and Julian Mcauley. 2012. Learning to discover social circles in ego networks. *Advances in neural information processing systems* 25 (2012).

[48] Boxuan Li, Reynold Cheng, Jiafeng Hu, Yixiang Fang, Min Ou, Ruibang Luo, Kevin Chen-Chuan Chang, and Xuemin Lin. 2020. Mc-explorer: Analyzing and visualizing motif-cliques on large networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1722–1725.

[49] Jinyan Li, Guimei Liu, Haiquan Li, and Limsoon Wong. 2007. Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms. *IEEE Transactions on Knowledge and Data Engineering* 19, 12 (2007), 1625–1637.

[50] Ronghua Li, Sen Gao, Lu Qin, Guoren Wang, Weihua Yang, and Jeffrey Xu Yu. 2020. Ordering Heuristics for k-clique Listing. *Proc. VLDB Endow.* (2020).

[51] Rong-Hua Li, Qiangqiang Dai, Lu Qin, Guoren Wang, Xiaokui Xiao, Jeffrey Xu Yu, and Shaojie Qiao. 2018. Efficient signed clique search in signed networks. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 245–256.

[52] Rong-Hua Li, Qiangqiang Dai, Guoren Wang, Zhong Ming, Lu Qin, and Jeffrey Xu Yu. 2019. Improved algorithms for maximal clique search in uncertain networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1178–1189.

[53] Xiaodong Li, Reynold Cheng, Kevin Chen-Chuan Chang, Caihua Shan, Chenhao Ma, and Hongtai Cao. 2021. On analyzing graphs with motif-paths. *Proceedings of the VLDB Endowment* 14, 6 (2021), 1111–1123.

[54] Don R Lick and Arthur T White. 1970. k-Degenerate graphs. *Canadian Journal of Mathematics* 22, 5 (1970), 1082–1096.

[55] Michael Luby. 1985. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*. 1–10.

[56] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2020. Maximum biclique search at billion scale. *Proceedings of the VLDB Endowment* (2020).

[57] Kazuhisa Makino and Takeaki Uno. 2004. New algorithms for enumerating all maximal cliques. In *Algorithm Theory-SWAT 2004: 9th Scandinavian Workshop on Algorithm Theory, Humlebæk, Denmark, July 8-10, 2004. Proceedings 9*. Springer, 260–272.

[58] Dror Marcus and Yuval Shavitt. 2012. Rage–a rapid graphlet enumerator for large networks. *Computer Networks* 56, 2 (2012), 810–819.

[59] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[60] Cataldo Musto, Pierpaolo Basile, and Giovanni Semeraro. 2019. Embedding knowledge graphs for semantics-aware recommendations based on dbpedia. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*. 27–31.

[61] Kevin A Naudé. 2016. Refined pivot selection for maximal clique enumeration in graphs. *Theoretical Computer Science* 613 (2016), 28–37.

[62] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* 435, 7043 (2005), 814–818.

[63] Charles A Phillips, Kai Wang, Erich J Baker, Jason A Bubier, Elissa J Chesler, and Michael A Langston. 2019. On finding and enumerating maximal and maximum k-partite cliques in k-partite graphs. *Algorithms* 12, 1 (2019), 23.

[64] Ali Pinar, Comandur Seshadhri, and Vaidyanathan Vishal. 2017. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings of the 26th international conference on world wide web*. 1431–1440.

[65] Nataša Pržulj and Noël Malod-Dognin. 2016. Network analytics in the age of big data. *Science* 353, 6295 (2016), 123–124.

[66] Kenneth H Rosen. 2007. *Discrete mathematics and its applications*. The McGraw Hill Companies,.

[67] Michael Rudolf. 1998. Utilizing constraint satisfaction techniques for efficient graph pattern matching. In *International Workshop on Theory and Application of Graph Transformations*. Springer, 238–251.

[68] Michael J Sanderson, Amy C Driskell, Richard H Ree, Oliver Eulenstein, and Sasha Langley. 2003. Obtaining maximal concatenated phylogenetic data sets from large sequence databases. *Molecular biology and evolution* 20, 7 (2003), 1036–1042.

[69] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. 2018. Butterfly counting in bipartite networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2150–2159.

[70] Matthew C Schmidt, Nagiza F Samatova, Kevin Thomas, and Byung-Hoon Park. 2009. A scalable, parallel algorithm for maximal clique enumeration. *Journal of parallel and distributed computing* 69, 4 (2009), 417–428.

[71] Eran Shaham, Honghai Yu, and Xiao-Li Li. 2016. On finding the maximum edge biclique in a bipartite graph: a subspace clustering approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 315–323.

[72] Haichuan Shang, Ke Zhu, Xuemin Lin, Ying Zhang, and Ryutaro Ichise. 2010. Similarity search on supergraph containment. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, 637–648.

[73] Chuan Shi, Xiangnan Kong, Philip S Yu, Sihong Xie, and Bin Wu. 2012. Relevance search in heterogeneous networks. In *EDBT*. 180–191.

[74] Chuan Shi, Chong Zhou, Xiangnan Kong, Philip S Yu, Gang Liu, and Bai Wang. 2012. Heterecom: a semantic-based recommendation system in heterogeneous networks. In *KDD*. 1552–1555.

[75] Shixuan Sun and Qiong Luo. 2020. In-memory subgraph matching: An in-depth study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1083–1098.

[76] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.

[77] Robert Endre Tarjan and Anthony E Trojanowski. 1977. Finding a maximum independent set. *SIAM J. Comput.* 6, 3 (1977), 537–546.

[78] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical computer science* 363, 1 (2006), 28–42.

[79] Yongxin Tong, Lei Chen, Yurong Cheng, and Philip S Yu. 2012. Mining Frequent Itemsets over Uncertain Databases. *Proceedings of the VLDB Endowment* 5, 11 (2012).

[80] Ngoc Hieu Tran, Kwok Pui Choi, and Louxin Zhang. 2013. Counting motifs in the human interactome. *Nature communications* 4, 1 (2013), 2241.

[81] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. 2016. Computing maximal cliques in link streams. *Theoretical Computer Science* 609 (2016), 245–252.

[82] Michael Matthias Voit and Heiko Paulheim. 2021. Bias in Knowledge Graphs–an Empirical Study with Movie Recommendation and Different Language Editions of DBpedia. *arXiv preprint arXiv:2105.00674* (2021).

[83] Chenxu Wang, Minnan Luo, Zhen Peng, Yixiang Dong, and Huaping Liu. 2023. Heterogeneous graph attention network with motif clique. *Neurocomputing* 555 (2023), 126608.

[84] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2019. Vertex Priority Based Butterfly Counting for Large-scale Bipartite Networks. *PVLDB* (2019).

[85] Bin Wu and Xin Pei. 2007. A parallel algorithm for enumerating all the maximal k-plexes. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 476–483.

[86] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[87] Stephan Wuchty, Zoltán N Oltvai, and Albert-László Barabási. 2003. Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nature genetics* 35, 2 (2003), 176–179.

[88] Mingyu Xiao and Hiroshi Nagamochi. 2017. Exact algorithms for maximum independent set. *Information and Computation* 255 (2017), 126–146.

[89] Yanyan Xu, James Cheng, Ada Wai-Chee Fu, and Yingyi Bu. 2014. Distributed maximal clique computation. In *2014 IEEE International Congress on Big Data*. IEEE, 160–167.

[90] Jianye Yang, Yun Peng, Dian Ouyang, Wenjie Zhang, Xuemin Lin, and Xiang Zhao. 2023. (p, q)-biclique counting and enumeration for large sparse bipartite graphs. *The VLDB Journal* (2023), 1–25.

[91] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and efficient truss computation over large heterogeneous information networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 901–912.

[92] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 555–564.

[93] Chenhao Ma Tianci Hou Xin Huang Yingli Zhou, Yixiang Fang. 2023. Efficient Maximal Motif-Clique Enumeration over Large Heterogeneous Information Networks (technical report). https://github.com/EnderturtleOrz/VLDB2024-Mclique/blob/master/VLDB2024-Mclique-Technique-Report.pdf.

[94] Xiao Yu, Hao Ma, Bo-June Hsu, and Jiawei Han. 2014. On building entity recommender systems using user click log and freebase knowledge. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 263–272.

[95] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM conference on Recommender systems*. 347–350.

[96] Yun Zhang, Charles A Phillips, Gary L Rogers, Erich J Baker, Elissa J Chesler, and Michael A Langston. 2014. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. *BMC bioinformatics* 15 (2014), 1–18.

A ADDITIONAL ALGORITHM

Algorithm 6: `reduceHIN`($\mathcal{H}$, $\mathcal{M}$)

```
input  : An HIN $\mathcal{H}$ and a motif $\mathcal{M}$
output : A small HIN $\mathcal{H}'$
1 $T_V, T_E \leftarrow$ vertex and edge types in $\mathcal{V}_\mathcal{M}$ and $\mathcal{E}_\mathcal{M}$ respectively;
2 remove vertices and edges whose types are not in $T_V$ and $T_E$ from
    $\mathcal{H}$ respectively ;                                  // Use rule 1
3 foreach $v \in \mathcal{V}$ and $\psi(v)$ is orbit type respectively do
4     foreach $\gamma \in T_V$ do
5         $a \leftarrow |\{u \in \mathcal{N}(v) | \psi(u) = \gamma\}|$ ;
6         $b \leftarrow |\{y \in \mathcal{N}_\mathcal{M}(x) | x \in \mathcal{V}_\mathcal{M} \wedge \phi(x, y) = \gamma\}|$ ;
7         if $a < b$ then remove $v$ from $\mathcal{V}$ ;          // Use rule 2
8 return A small HIN $\mathcal{H}'$ ;
```

We first collect different vertex and edge types in $\mathcal{V}_\mathcal{M}$ and $\mathcal{E}_\mathcal{M}$ respectively (line 1). Next, we follow Rule 1 to remove vertices and edges whose types are not in T_V and T_E from $\mathcal{H}$ respectively (line 2). Then, for each vertex $v \in \mathcal{V}$ of orbit type, we iteratively verify if there exists a vertex type $\gamma \in T_V$ that satisfies Rule 2. If such a vertex type exists, we can remove v from $\mathcal{V}$ (lines 3-7). Finally, we return a small HIN $\mathcal{H}'$ (line 8).

META			POMA		
Step	**Time**	**Space**	**Step**	**Time**	**Space**
VF3	$O(\gamma)$	$O(\Delta)$	`reduceHIN`	$O(n+m)$	$O(n+m)$
			VF3	$O(\gamma)$	$O(\Delta)$
GetMMC	$O(\alpha \cdot n!)$	$O(\Delta)$	`selectCand`	$O(\Delta \cdot n)$	$O(\Delta)$
			`Order-Divide`	$O(\Delta \cdot n)$	$O(n)$
			`MP-MMC`	$O(\alpha\Delta \cdot 2^n)$	$O(\Delta)$
Total	$O(\gamma + \alpha \cdot n!)$	$O(\Delta)$	Total	$O(\gamma + \alpha\Delta \cdot 2^\beta)$	$O(\Delta)$

Table 10: Time and space complexities of MMCE algorithms.

B COMPLEXITIES AND LIMITATIONS

We compare the time, and space complexities of each step of `META` and `POMA` in Table 10. Clearly, `POMA` takes less time than `META`, since `POMA` uses `MP-MMC` rather than `GetMMC` to enumerate maximal M-cliques containing each motif instance, its total time cost is reduced from $O(\alpha \cdot n!)$ to $O(\alpha\Delta \cdot 2^n)$. Particularly, since the ordering framework of `POMA` ensures that each motif instance contains at most β vertices in its candidate set, the time cost of `MP-MMC` is reduced from $O(\alpha\Delta \cdot 2^n)$ to $O(\alpha\Delta \cdot 2^\beta)$. As for space complexities, both `POMA` and `META` need to store all motif instances in memory. The detailed analysis for `POMA` is in Sections 5 and 6.

While `POMA` significantly outperforms `META`, it has the following limitations: Firstly, the definition of M-clique is based on induced subgraph isomorphism, restricting our solution to only support this kind of isomorphism. Secondly, `POMA` is designed for processing static graphs, so it cannot be applied to dynamic graphs directly. In the future, we will address the above issues.

C PROOFS OF LEMMAS AND THEOREMS

Theorem 4.1. Algorithm 6 does not affect the correctness of `POMA`.

Proof. The first rule is very straightforward. For the second rule, we need to prove that there does not exist a motif instance Γ of $\mathcal{M}$ containing v. More specifically, we can prove this by contradiction. Assume that there is a motif instance Γ of $\mathcal{M}$ containing v, and τ denotes the injective mapping: $\mathcal{V}_\mathcal{M} \rightarrow \mathcal{V}$, and let $\tau(x) = v$, where $x \in \mathcal{V}_\mathcal{M}$. Clearly, $\mathcal{N}(v)$ has at least the same number of vertices of each type as $\mathcal{N}_\mathcal{M}(x)$ since v is an injective mapping to x. This observation contradicts the second rule. □

Theorem 4.3. Algorithm 3 correctly computes the candidate sets for all motif instances of $\mathcal{M}$.

Proof. Here, we need to prove that all vertices in the candidate set of Γ can be computed by Algorithm 3. Lemma 4.1 states that if all motif instance neighbors of Γ can be computed, then its candidate set can also be computed. For any $Y \in \mathcal{T}[\Gamma]$, let $u = (Y \backslash \Gamma)$. We have that all match-sets in $\Gamma \cup \{u\}$ are also motif instances of $\mathcal{M}$, since $\Gamma_1 \cup Y = \Gamma \cup \{u\}$ is an M-clique. Consequently, any $Y \in \mathcal{T}[\Gamma]$ can be found by Algorithm 3. Hence, the lemma holds. □

Lemma 4.4. The total time cost of `selectCand` is $O(|\mathcal{V}| \cdot \Delta)$, where Δ is the number of motif instances of $\mathcal{M}$ in $\mathcal{H}$.

Proof. In the worst case, for each motif instance $\Gamma \in \Phi$, the computation of $\mathcal{T}[\Gamma]$ may take up to $O(|\mathcal{V}| \cdot |\mathcal{V}_\mathcal{M}|)$ time. Besides, for each motif instance T in $\mathcal{T}[\Gamma]$, we need $O(|\mathcal{V}_\mathcal{M}|)$ time to check if T is a motif instance neighbor of Γ, and the size of $\mathcal{T}[\Gamma]$ is up to $|\mathcal{V}|$. In practice, $|\mathcal{T}[\Gamma]|$ is typically very small, and the number of vertices in a motif, i.e., $|\mathcal{V}_\mathcal{M}|$, can be treated as a constant. As a result, by summarizing the above cost, we obtain Lemma 4.3. □

Lemma 5.1. For each motif instance Γ of $\mathcal{M}$, Algorithm 2 can enumerate all the maximal M-cliques containing it without missing any results.

Proof. We prove this by contradiction. Suppose there exists a maximal M-clique R containing Γ that cannot be enumerated. There must exist a vertex u in S such that $\lambda(u) \leq \Lambda_\Gamma[\psi(u)]$. Next, we can construct a new motif instance Γ' by selecting vertices of each type with the minimum order from R such that $\Gamma' = \{v_1, v_2, \cdots, v_{|\mathcal{V}_\mathcal{M}|}\}$ satisfies $\Lambda_\Gamma[\psi(v)] < \lambda(v)$, for each $v \in R \backslash \Gamma'$. R can be enumerated by the recursive enumeration process starting from Γ' as all vertices in R will not be put into the not set X by Algorithm 4. This contradicts our assumption. Therefore, the lemma holds. □

Lemma 5.2. For each motif instance Γ of $\mathcal{M}$, Algorithm 2 can enumerate all the maximal M-cliques containing it without missing any results.

Proof. We prove this by contradiction. Suppose there exists a maximal M-clique R which is enumerated twice from the two motif instances Γ_1 and Γ_2, respectively. Without loss of generality, we assume Λ_{Γ_1} is dominated by Λ_{Γ_2}, which means $\Lambda_{\Gamma_1}[\gamma] \leq \Lambda_{\Gamma_2}[\gamma]$ for each type γ. There exists a vertex $v \in \Gamma_1 \setminus \Gamma_2$ such that $\lambda(v) < \Lambda_{\Gamma_2}[\psi(v)]$. Hence, R cannot be enumerated starting from Γ_2. Similarly, if Λ_{Γ_1} and Λ_{Γ_2} cannot dominate each other. We can also find a vertex $v \in \Gamma_1 \setminus \Gamma_2$ such that $\lambda(v) < \Lambda_{\Gamma_2}[\psi(v)]$, and R cannot be enumerated starting from Γ_2. Since both cases contradict our assumption, the lemma holds. □

Lemma 5.3. The total time cost of `POMA` is $O(\gamma + \alpha\Delta 2^\beta)$, where $\beta = h \times \delta$, δ is the degeneracy number of the input HIN, h is the number of vertex types in the input motif, Δ is the number of motif instance of $\mathcal{M}$ in $\mathcal{H}$, and the other variables have the same meanings as those in Lemma 3.1.

Proof. In the worst-case scenario, each motif instance Γ necessitates the consideration of up to 2^β branches in Algorithm 2. Specifically, a motif instance Γ has h different types, and for each type, up to δ vertices could be incorporated into the candidate set C.

Hence, at most $2^{h\times\delta}$ branches need to be processed. Consequently, the time complexity of Algorithm 2 is $O(\gamma + \alpha\Delta 2^{\beta})$. □

Lemma 6.1.For any M-clique R and a vertex u in $\mathcal{P}_R$, any maximal M-clique $R' \supseteq R$ must include either u or one of the vertices that are not M-clique precedence over u, as otherwise, R' could be enlarged by adding u.

Proof. We prove this by contradiction. Assume that R' does not contain vertex u or any vertices that are not M-clique precedence over u. In other words, R' satisfies $R \subseteq R' \subseteq R \cup \mathcal{I}(u)$. Now, consider adding vertex u to R', let $S = R' \cup \{u\}$, and then, we claim that each match-set T in S containing u is also a motif instance of $\mathcal{M}$. Here, we can always identify a motif instance of Γ in R' such that $|\Gamma \backslash T| = 1$, and we denote $x = \Gamma \backslash T$. Besides, we have $\forall y \in \mathcal{N}(x) \cap S$, if $\exists (a,b) \in \mathcal{E}_{\mathcal{M}}, \psi(a) = \psi(x) \wedge \psi(b) = \psi(y)$, then $(u,y) \in \mathcal{E}$. This is because that

(1) all of x's neighbors in R are also u's neighbors, since u is selected from $\mathcal{P}_R$;
(2) $\forall y \in \mathcal{N}(x) \cap \mathcal{I}(u) \cap R'$, if $\exists (a,b) \in \mathcal{E}_{\mathcal{M}}, \psi(a) = \psi(u) \wedge \psi(b) = \psi(y)$, then $(u,y) \in \mathcal{E}$, as outlined in Definition 9.

Hence, T is also a motif instance of $\mathcal{M}$, which means S is a maximal M-clique. This contradicts our assumption that R' is a maximal M-clique. Hence, the lemma holds. □

Lemma 6.2 (Approximating $\mathcal{I}(p)$)). Given an HIN $\mathcal{H}$, a motif $\mathcal{M}$, an M-clique R and a pivot vertex p, a vertex v in C that satisfies at least one of the following conditions is M-clique precedence over p:

- **Condition 1.** $1 + |\mathcal{D}_v^{R\cup\{p\}}| > t$.
- **Condition 2.** $\psi(p) = \psi(v)$, and there is only one vertex in $\mathcal{M}$ with type $\psi(p)$.
- **Condition 3.** Let $L = \{x | x \in R' \wedge \psi(x) = \psi(v)\}$, and $\left(\bigcup_{x\in L} (\mathcal{N}(x) \cap R')\right) \subseteq (\mathcal{N}(v) \cap R')$, where $R' = R \cup \{p\}$;

Proof. We will prove each condition separately:

(1) Condition 1 is guaranteed by Lemma 6.4 in [39].
(2) When adding v to $R \cup \{p\}$, all new match-sets will contain v but not p since $\mathcal{M}$ has only one vertex with type $\psi(p)$. Since $R \cup \{v\}$ is an M-clique, $R \cup \{p, v\}$ is an M-clique.
(3) When adding vertex v to R', we have the following observations:
    - if a match-set T in $R' \cup \{v\}$ includes v but not p, T must be a motif instance, since $R \cup \{v\}$ forms an M-clique.
    - if T includes p and v, we can always find a vertex $x \in R' \backslash T$, such that $T \backslash \{v\} \cup \{x\}$ is a motif instance. Besides, all neighbors of x are also neighbors of v in $T \backslash \{v\}$. Hence, T is a motif instance.

    Therefore, $R' \cup \{v\}$ is also an M-clique, and the lemma holds.

□

Lemma 6.3. Replace `GetMMC` with `MP-MMC` (line 6) in Algorithm 1, the total running time of this algorithm is $O(\gamma + \alpha\Delta 2^{n})$, where all variables have the same meanings as those in Lemma 5.3.

Proof. In the worst case, for each motif instance Γ, there are at most 2^n branches to be considered in Algorithm 1. Thus, the time complexity of Algorithm 1 is $O(\gamma + \alpha\Delta 2^{n})$. □

Remark. With our pivot-based pruning techniques, the practical performance of our algorithm can be quite efficient, which is further demonstrated in our experiments.

Table 11: Similarity between the maximal M-cliques.

id	Instacart				
	lower quartile	median	upper quartile	maximum	mean
1	5.88%	9.09%	12.00%	32.00%	9.19%
2	0.00%	0.00%	5.26%	14.29%	2.31%
3	11.11%	13.33%	16.13%	39.29%	13.86%
4	3.85%	5.00%	7.41%	21.43%	5.55%
5	0.00%	0.00%	3.23%	16.67%	1.65%
6	0.00%	2.63%	3.85%	11.11%	2.10%
7	1.39%	3.13%	5.45%	10.19%	3.46%
8	0.00%	0.00%	3.23%	18.18%	1.56%
9	0.00%	0.00%	7.69%	25.00%	4.49%
10	0.00%	0.00%	0.00%	25.00%	1.97%
11	0.00%	0.00%	5.56%	28.57%	3.03%
12	3.57%	4.17%	6.45%	18.75%	4.82%
13	0.00%	3.70%	6.25%	25.00%	3.53%
14	0.00%	0.00%	0.00%	10.00%	1.39%
15	0.00%	0.00%	3.03%	16.67%	1.54%
16	0.00%	0.00%	0.00%	11.11%	0.15%
17	0.00%	0.00%	0.00%	12.50%	0.92%
18	0.00%	3.57%	5.56%	20.00%	3.46%
19	0.00%	0.00%	0.00%	0.00%	0.00%
20	2.78%	3.85%	4.69%	17.62%	4.32%
id	WordNet				
	lower quartile	median	upper quartile	maximum	mean
1	0.00%	0.00%	0.000%	8.33%	0.03%
2	0.00%	0.00%	0.00%	0.00%	0.00%
3	0.00%	0.00%	0.00%	0.00%	0.00%
4	0.00%	0.00%	0.00%	4.00%	0.67%
5	0.00%	0.00%	0.00%	14.29%	0.89%
6	0.00%	0.00%	0.00%	12.50%	2.08%
7	0.00%	0.00%	0.00%	0.00%	0.00%
8	0.00%	0.00%	0.00%	5.88%	1.09%
9	0.00%	0.00%	0.00%	0.00%	0.00%
10	0.00%	0.00%	0.00%	0.00%	0.00%
11	0.00%	0.00%	0.00%	0.00%	0.00%
12	0.00%	0.00%	0.00%	0.00%	0.00%
13	0.00%	0.00%	0.00%	4.35%	0.01%
14	0.00%	0.00%	0.00%	0.00%	0.00%
15	0.00%	0.00%	0.00%	0.00%	0.00%
16	0.00%	0.00%	0.00%	0.00%	0.00%
17	0.00%	0.00%	0.00%	0.00%	0.00%
18	0.00%	0.00%	0.00%	0.00%	0.00%
19	0.00%	0.00%	0.00%	0.00%	0.00%
20	0.00%	0.00%	0.00%	0.00%	0.00%

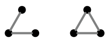
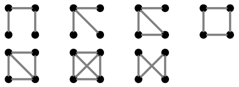
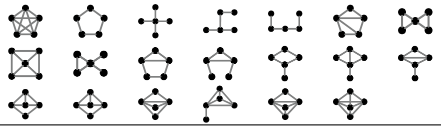
D ADDITIONAL EXPERIMENTS

Table 12 shows the structures of motifs with 3, 4, and 5 vertices used in our experiments and the previous works that use these motifs.

1. Similarity of the maximal M-cliques.

To measure the difference between the maximal M-cliques, we calculate the similarity between all pairs of maximal M-cliques on the Instacart and WordNet datasets. We calculate the similarity between all pairs of maximal M-cliques on the Instacart and WordNet datasets. Specifically, we employ the "Jaccard similarity" metric to evaluate the similarity between any two maximal M-cliques R_1 and

Table 12: Structures of the 3-, 4- and 5-vertices motifs.

3-vertex [39, 48, 58–60, 64, 65, 82, 94]	
4-vertex [27, 39, 48, 64, 69, 84]	
5-vertex [9, 20, 27, 39, 46, 64, 90]	

R_2, i.e.,:

$$sim(R_1, R_2) = \frac{|R_1 \cap R_2|}{|R_1 \cup R_2|}, \tag{4}$$

where $|R_1 \cap R_2|$ is the number of vertices shared by R_1 and R_2, and $|R_1 \cup R_2|$ is the total number of unique vertices in R_1 and R_2. Table 7 presents the percentile of the maximal M-clique's similarity on Instacart and WordNet. The average similarity values between any two maximal M-cliques for each motif vary from 0 to 0.139, and the average values for all motifs on these two datasets are 0.029 and 0.002 respectively, indicating that these maximal M-cliques are very different.

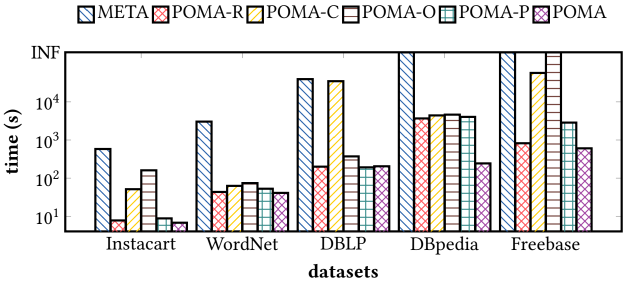


Figure 13: Efficiency comparison of MMCE algorithms.

2. Ablation study. To evaluate the effect of four key steps, we design four variants by removing reduceHIN, selectCand, motif-pivot, and ordering framework from POMA, which are denoted by POMA-R, POMA-C, POMA-P, and POMA-O, respectively. We then run META, POMA, and these variants on all datasets and report the efficiency results in Figure 13.

We make the following observations and analyses: (1) For the first three datasets, both POMA-R and POMA demonstrate similar performance. However, for the rest two datasets, POMA is at least 10 times faster than POMA-R. Thus, the usage of reduceHIN indeed allows us to locate smaller sub-HINs instead of performing MMCE directly on large HINs. (2) POMA is much faster than POMA-C, meaning that selectCand can quickly identifying the candidate set for each motif instance. For example, on the DBLP dataset, POMA is almost two orders of magnitude faster than POMA-C, which further shows that identifying the candidate set plays a crucial role in enhancing the overall efficiency of the process. (3) POMA-P runs slower than POMA. For example, on the DBpedia dataset, POMA-P runs over five times slower than POMA because the deploying pivot technique can avoid much redundant computation, thereby significantly improving the efficiency. (4) Without the order-based framework, POMA-O runs much slower than POMA. For example, on the Freebase dataset, POMA is almost three orders of magnitude faster than POMA-O. Note that POMA-O uses the set-trie tree structure [39] to avoid enumerating the duplicated maximal m-cliques, which is very time-consuming. (5) All these four variants significantly outperform the baseline META, further substantiating the efficacy of our four proposed techniques, especially on larger HINs.

Table 13: The static information of original and reduced HINs.

Dataset	# Vertices		# Edges	
	Original	*Reduced*	*Original*	*Reduced*
Instacart	49,688	294.9	12,770	1,194.6
WordNet	76,853	22,371.7	240,798	34,245
DBLP	881,039	515,794.8	2,247,195	1,076,192.5
DBpedia	8,970,120	65,489.5	71,403,844	125,361.3
Freebase	347,463,729	1,561,926.4	1,110,001,528	1,437,902.7

3. Effect of graph reduction technique. We present the average numbers of vertices and edges before and after the graph reduction for different motifs across all datasets. in Table 13. Clearly, our graph reduction method can substantially reduce the size of the original HIN. More specifically, on the DBpedia and Freebase datasets, the scale of the graph is substantially reduced, and the numbers of vertices and edges removed are significantly higher than those on other datasets. This indicates that these two datasets require more time for the reduceHIN operation, which also explains the results shown in Figure 10. Thus, by confining the enumeration process to smaller subgraphs, we can improve the POMA's efficiency.

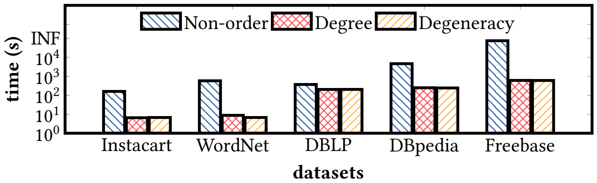


Figure 14: Effect of different vertex ordering strategies.

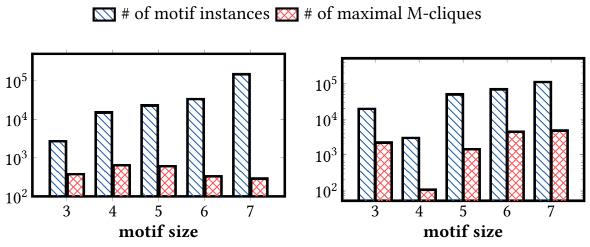
4. Case studies. We present two case studies on real-world HINs:

Table 14: Community quality analysis on DBLP.

Method	Diameter	Similarity			
		Author	Paper	Venue	Topic
R-com	4.92	0.25	0.19	0.21	0.25
POMA	**2.18**	**0.60**	**0.30**	**0.48**	**0.48**

(1) Community discovery. We consider the DBLP co-authorship network and use POMA to find groups of researchers and papers by using a motif, i.e., "Author-Paper-Author", which is also a meta-path [76], describing the co-authorship between two authors. Intuitively, we wish to find cohesive heterogeneous communities, each of which includes both authors and papers that are intensively linked together. Figure 15(c) depicts a maximal M-clique based on the motif, showing that the four researchers "Jiawei Han", "Jian Pei", "Yizhou Sun", and "Philip S. Yu" have very close collaboration on four papers in Figure 15(b).

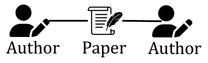
(2) Product recommendation. We use the Instacart e-commerce network which includes various products. We consider a 4-vertex motif with three types of items (i.e., "beverages", "daily egg", and "snack"), as shown in Figure 16(a). The intuition of this motif is to find out those beverages and daily egg products that are frequently bought together with snacks for better product promotion. Figure



(a) Instacart

(b) WordNet

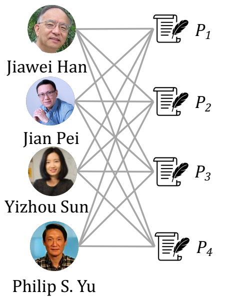
Figure 17: Numbers of motif instances and maximal M-cliques.



(a) A motif $\mathcal{M}_1$

Id	Title
P_1	PathSim: Meta Path-Based Top-k ...
P_2	PathSelClus: Integrating Meta Path ...
P_3	HIN: the past, the present, and the future
P_4	Mining knowledge from interconnected ...

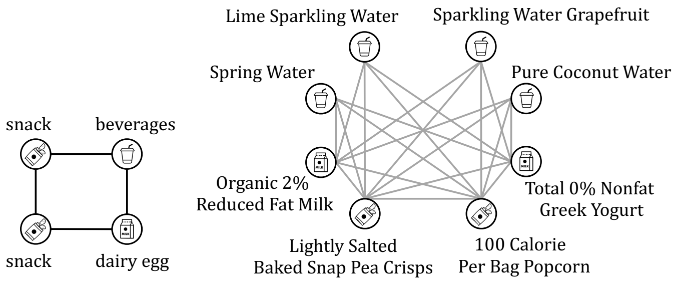
(b) Papers



(c) A maximal M-clique for $\mathcal{M}_1$

Figure 15: A case study on DBLP co-authorship network.

16(b) shows a maximal M-clique found by POMA, which is highly connected and reflects the related items interesting to customers. Thus, the maximal M-clique can be used in product promotion (e.g., a bundle containing two snacks, one beverage, and one daily egg extracted from this maximal M-clique might be sold at a discounted price).



snack
beverages
snack
dairy egg
Lime Sparkling Water
Sparkling Water Grapefruit
Spring Water
Pure Coconut Water
Organic 2% Reduced Fat Milk
Total 0% Nonfat Greek Yogurt
Lightly Salted Baked Snap Pea Crisps
100 Calorie Per Bag Popcorn

(a) A motif $\mathcal{M}_1$

(b) A maximal M-clique for $\mathcal{M}_1$

Figure 16: A case study on Instacart e-commerce network.

In addition, we would like to bring to the readers' attention that a recent demo system [48] has shown the various applications of maximal M-cliques in biological network analysis. As shown in [48], the maximal M-clique can be used to disclose new side effects of a drug and potential drugs for healing diseases on a large biological network, facilitate the analysis and visualization of a biological network, support disease subtyping, drug mechanism investigation, and drug repurposing. An online video demonstrating how the maximal M-clique works for the above applications is available at https://www.dropbox.com/s/vkalumc28wqp8yl/demo.mov.

4. The numbers of motif instances and maximal M-cliques. We examine the numbers of motif instances and maximal M-cliques for motifs with vertex sizes ranging from 3 to 7 on Instacart and WordNet datasets (for lack of space, we omit the numbers for other datasets). Figure 17 presents the average numbers for motifs with different sizes. We see that the average motif instance numbers are often below 100,000, while the average maximal M-clique numbers are much smaller since they are usually around 1,000.