On Analyzing Graphs with Motif-Paths

Xiaodong Li[†], Reynold Cheng[†], Kevin Chen-Chuan Chang[‡],
Caihua Shan[†], Chenhao Ma[†], Hongtai Cao[‡]
[†]Department of Computer Science, University of Hong Kong, Hong Kong SAR
[‡]Department of Computer Science, University of Illinois at Urbana-Champaign, USA
{xdli,ckcheng,chshan,chma2}@cs.hku.hk;{kcchang,hongtai2}@illinois.edu

ABSTRACT

Path-based solutions have been shown to be useful for various graph analysis tasks, such as link prediction and graph clustering. However, they are no longer adequate for handling complex and gigantic graphs. Recently, *motif-based analysis* has attracted a lot of attention. A motif, or a small graph with a few nodes, is often considered as a fundamental unit of a graph. Motif-based analysis captures high-order structure between nodes, and performs better than traditional "edge-based" solutions. In this paper, we study *motif-path*, which is conceptually a concatenation of one or more motif instances. We examine how motif-paths can be used in three path-based mining tasks, namely link prediction, local graph clustering and node ranking. We further address the situation when two graph nodes are not connected through a motif-path, and develop a novel defragmentation method to enhance it. Experimental results on real graph datasets demonstrate the use of motif-paths and defragmentation techniques improves graph analysis effectiveness.

PVLDB Reference Format:
Xiaodong Li, Reynold Cheng, Kevin Chen-Chuan Chang, Caihua Shan, Chenhao Ma, Hongtai Cao. On Analyzing Graphs with Motif-Paths. PVLDB, 14(6): 1111-1123, 2021.
doi:10.14778/3447689.3447714

1 INTRODUCTION

Motif-based analysis has recently emerged as an important tool for discovering insight from graphs. A motif, which is also known as higher-order structure or graphlet, is a small subgraph pattern [2, 34]. As pointed out by [5], a motif is a fundamental building block of large and complex networks, and it enables "higher-order semantics" analysis. Fig. 1(a) illustrates several instances of "triangle motif" on a graph G (e.g., 1-2-3, 1-2-4, and 4-7-6). Motifs have been shown to be more effective than traditional "graph-edge-based" solutions in a range of problems, such as link prediction [1, 7], graph clustering [5, 21, 40, 44], and node ranking [45].

Another important line of graph analysis solutions exploits the connectivity between graph nodes. In particular, important paths between two nodes are first extracted, based on which graph analytics are performed. For example, as proposed in [20, 31], a link is predicted to exist between two nodes s and t with a higher probability, if there are many short paths between s and t. Path-based solutions are also commonly used in tasks such as graph clustering, node ranking, and node relationship analysis [27, 28, 44].

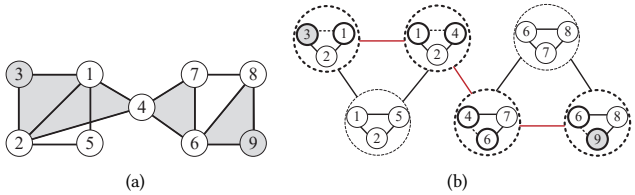


Figure 1: Illustration of (a) Graph G; (b) Higher-order graph of G with triangles. Motif-instances of G are marked in circles and a link between two circles means that they share at least one node. A motif-path between 3 and 9 is highlighted.

In this paper, we ask a question: can we incorporate the use of motifs in path-based solutions, in order to enhance graph analysis effectiveness? For example, given two nodes s and t on a graph G, and a motif τ, it may be useful to find a sequence of τ's graph instances (namely motif-instances) that connect s and t, rather than a sequence of edges which form the traditional paths between s and t. We call this sequence *motif-path* and use the number of motif-instances in the sequence as the path length, following that the length of the traditional path is the number of edges. Fig. 1(a) illustrates the motif-path between nodes 3 and 9, which is a sequence of triangle motif-instances (in gray). Fig. 1(b) shows the *higher-order graph* of G which organizes all the motif-instances of triangle (e.g., nodes 1-2-4) in dashed circles, where the link between two circles means that the corresponding motif-instances share at least one common node. In this paper, we study how motif-paths can be incorporated to three major classes of graph analysis tasks, namely link prediction, local graph clustering, and node ranking. To achieve this goal, we first propose efficient online and offline algorithms to find the motif-paths and then develop the metrics based on motif-paths in several graph mining solutions.

Challenges. To adopt motif-paths in graph analysis tasks, we face two major challenges. First, as we will explain, the *shortest* motif-path between two given nodes needs to be extracted, i.e., the smallest number of motif-instances in the motif-path, as shown in Fig. 1(b). However, finding it directly from the higher-order graph is not possible. On the one hand, the combinatorial number of motif-instances has to be enumerated, the cost of which can be very high even for a moderate-size graph; on the other hand, the higher-order graph can not be materialized because large amount of space needed. To tackle this issue, we develop fast offline and online algorithms,

Proceedings of the VLDB Endowment, Vol. 14, No. 6 ISSN 2150-8097.
doi:10.14778/3447689.3447714

which enable us to find the shortest motif-path between two nodes quickly. The second issue is that the higher-order graph is usually fragmented into many connected components even though the original graph is connected. Then for two given nodes s and t on graph G, a motif-path may not be found between s and t. This can happen when two motif-instances do not have any nodes in common; consequently, there does not exist a motif-path between s and t that can traverse a sequence of motif-instances. This "fragmentation" can affect the effectiveness of our solutions. To tackle this problem, we propose a solution that connects isolated motif-instances with the edges in G while preserving the higher-order graph structures.

Contributions. We first formulate the notions of motif-paths. We then study a framework for incorporating motif-paths in link prediction, local graph clustering, and node ranking. We also develop novel motif-path-based metrics for these applications. With a novel indexing framework, we study efficient offline and online algorithms for finding shortest motif-paths, and address the fragmentation problem with a simple but effective solution that links isolated motif-instances. Extensive experiments on several real graph datasets reveal that compared with existing path-based solutions, our methods achieve an improvement of up to 41% in terms of effectiveness for link prediction, 10% for local graph clustering, and 8% for node ranking. Our solutions are also $5-10\%$ more effective than the state-of-the-art motif-based approaches.

The paper is organized as follows. We review the related work in Section 2. We discuss the definitions and algorithms in Section 3. Section 4 proposes a new motif-path framework for graph mining tasks. We handle the higher order graph fragmentation in Section 5. In Section 6 we evaluate the performance. Section 7 concludes.

2 RELATED WORK

In this section, we survey the work about the motif-based analysis and the path-based analysis respectively in the graph mining area.

Motif-based analysis. Motifs are small building blocks of a large graph [5, 18, 26, 32, 36]. Since motifs can capture high-order structure of a graph, they are used in a number of graph mining tasks, such as graph clustering [5, 21, 40, 44], node ranking [45], and link prediction [1, 7]. For example, motif-based graph clustering extends the notion of graph conductance to motif-conductance, by enumerating motif-instances rather than edges for a cluster [5, 21, 40, 44]; motif-based node ranking employs motif-based authority, which is computed by running PageRank with respect to motifs [45]; motif-based link prediction finds motif-based feature vectors for each missing link, e.g., the number of motif-instances that contains the end points of the missing link [1]. These solutions have been shown to be better than existing "edge-based" approaches, which only use graph edges in the mining process.

Path-based analysis. Another branch of works in graph mining are *path-based* [6, 11, 12, 14, 20, 22, 25, 31], i.e., the path-based information of a graph is used in the mining process. For example, in [20], links are predicted by analyzing the length and number of the paths that connect the pair of nodes of interest. In [31], nodes are clustered based on the shortest path distance. The authors in [6] show that graph nodes can be ranked based on the number of shortest paths passing through them.

Only a few works study the connectivity between two nodes with respect to a given motif. Huang et al. [16] develop a k-truss community model based on the connectivity between triangles. Two nodes are in the same community if they are reachable from each other through a series of adjacent triangle-instances. However, the concept of a path of motif-instances is not proposed, and only triangles are supported. Motif-based random walk is designed to calculate the motif-based conductance and authority, which is further used for clustering [21, 40, 44] and node ranking [45]. To perform a motif-based random walk, these papers first generate a motif-adjacency matrix, where each element in the matrix denotes the number of motif-instances containing the corresponding pair of nodes. Then they run a typical random walk on it. However, as pointed by [40], the time cost of motif-adjacency matrix is high thus only triangles are supported in the current research scope. To our best understanding, previous work has not studied motif-path. In this paper, we propose elegant definitions and fast searching algorithms of motif-path, as well as the use of it in the three fundamental graph mining problems – link prediction, clustering, and node ranking. Since a motif-path captures high-order structure between two nodes, with appropriate adaptations, better effectiveness can be achieved in these graph mining tasks compared with motif-based or path-based solutions. We remark that our solution can support current popular motifs (e.g., 2-5 nodes motifs).

3 MOTIF-PATH

In this section, we introduce the definition and searching algorithms. We define the shortest motif-path search problem in Section 3.1, and develop offline and online algorithms in Section 3.2.

3.1 Problem Definition

Before defining the motif-path, we first introduce *motif-instance* and *motif-connectivity* in graph $G = (V, E)$[1].

Definition 1 (Motif-instance [2]). *Given a graph $G = (V, E)$ and a motif $\tau = (V_\tau, E_\tau)$, the motif-instance $m = (V_m, E_m)$ of τ is a subgraph of G which is isomorphic to τ, denoted as $m \simeq \tau$.*

Definition 2 (Motif-connectivity). *Given two motif-instances $m_s \simeq \tau$ and $m_t \simeq \tau$ in G, m_s and m_t are τ-connected, if there exist a sequence of motif-instances $<m_1, \ldots, m_n>$ in G, where $n \geq 2$, such that $m_1 = m_s, m_n = m_t$, and for $1 \leq i < n, m_i \simeq \tau, V_{m_i} \cap V_{m_{i+1}} \neq \emptyset$. Motif-instances m_i and m_j are τ-adjacent if they are τ-connected with $n = 2$, meaning that m_i and m_j share at least one common node.*

Definition 3 (Higher-Order Graph). *Given $G = (V, E)$ and motif τ, the higher-order graph of G, denoted as $\mathcal{G}$, is an organization of the motif-instances of τ in a graph manner: each node is a motif-instance and there is an edge if the two motif-instances are τ-adjacent.*

As shown in Definition 1, motif-instance is the subgraph of G that is isomorphic[2] to the motif. For example, nodes $(1, 2, 3)$ form the motif-instance of triangle in Fig. 1(a). Two motif-instances of motif τ are τ-connected if there is a sequence of motif-instances between them (i.e., Definition 2). Motif-connectivity, also known as

[1] Unless otherwise stated, the graph is connected, unweighted and undirected.
[2] Here we use node-induced subgraph to check isomorphism, which is a popular setup [2, 30]. The framework can be extended for non-induced subgraphs.

higher-order connectivity is used in searching higher-order components [21] and motif-based random walk [8]. Note that the triangle-connectivity from [9, 16] is a special case of motif-connectivity.

Thus a motif-path between m_i and m_j is a sequence of τ-adjacent motif-instances that link m_i and m_j. Next, we define the length of the motif-path. In the traditional graph, the path length is the number of graph edges on the path. Since the original graph G is a special case of the higher-order graph, e.g., with edge •—• used as the motif. To make the path length from the two presentations consistent, we use the number of the motif-instances on the motif-path as its length. With this definition, it is easy to show that the shortest path distance always equals to the shortest motif-path distance when using edge as the motif pattern.

In this paper, we find shortest motif-path from the higher-order graph, denoted as $\mathcal{P}$, where the number of motif-instances in the sequence is minimized. Also, to apply the motif-path into graph mining tasks, we study the shortest motif-path between nodes $(s,t) \in V \times V$, denoted as $\mathcal{P}_{s,t}$ (i.e., Definition 4). We denote $|\mathcal{P}_{s,t}|$ as the path length, i.e., the number of motif-instances in the sequence. For example, $|\mathcal{P}_{3,9}| = 4$ in Fig. 1(b) (the red path).

Definition 4 (Shortest Motif-path). *Given $G = (V, E)$, $(s,t) \in V \times V$ and motif τ, the shortest motif-path $\mathcal{P}_{s,t}$ is the motif-path with minimum number of motif-instances m_i where:*

(1) $m_i \simeq \tau$, $i = 1, 2, ..., |\mathcal{P}|$;
(2) *Two endpoints* $s \in V_{m_1}$ *and* $t \in V_{m_{|\mathcal{P}|}}$.
(3) $|V_{m_i} \cap V_{m_{i+1}}| \geq 1$, $i = 1, 2, ..., |\mathcal{P}| - 1$.

Why shortest motif-path? First, the rise of higher-order graph [4, 21, 40] calls for the need of developing a theoretical tool to analyze its properties, and a shortest motif-path algorithm is easy to be adapted to quantify the properties of the higher-order graph (e.g., diameter and connected component), which can benefit the mining methods, e.g., the defragmentation in Section 5. Second, the shortest path distance is treated as an important feature in graph mining, and as we will show, the shortest motif-path based solution can obtain good effectiveness by combining this feature with higher-order graph structures. Next, we develop offline and online algorithms to find shortest motif-path efficiently, making it possible to develop graph mining solutions based on shortest motif-paths. Finally, the framework introduced in the paper can be extended to answer several interesting variants. For example, it is interesting to extend "shortest" into an enriched scoring function (e.g., most diverse motif-path), and extend the definition of motif-connectivity (e.g., the neighboring motif-instances should share an edge). The algorithms can also be extended to find motif-paths in labeled graphs, e.g., heterogeneous information networks.

3.2 Shortest Motif-path Searching

However, it is not feasible to directly search shortest motif-path $\mathcal{P}_{s,t}$ from the higher-order graph $\mathcal{G}$. We demonstrate the reason in the baseline algorithm as below.

3.2.1 A Baseline Method. First, we generate the higher-order graph $\mathcal{G}$ by enumerating the motif-instances of τ. It takes $O(\theta h^2)$ operations where $h = |V_\tau|$ and $\theta = \binom{|V|}{h}$. In the worst case, lines 2-5 of Algorithm 1 search all-pairs shortest motif-paths in $\mathcal{G}$, which costs $O(\theta^3)$ by Floyd-Warshall algorithm [38]. We find that it is even not possible to run the baseline for a moderate-size graph because of combinatorial complexity.

3.2.2 A Novel MOD-Index. Since the high complexity of the baseline, it is reasonable to use index to speed up the searching process. However, it is challenging due to the combinatorial number of motif-instances. For example, the number of 5-node motif-instances around the single node can be four to six magnitudes bigger than the size of the original graph [37]. More worse, it is time-consuming to re-compute the index when the user switches the motifs, thus a generic indexing framework for different motifs is required. We notice that even though there are combinatorial number of motif-instances, many of them are in fact redundant in terms of shared substructures. To efficiently organize them, we use the *motif-orbits*.

Algorithm 1 Shortest Motif-Path Baseline (Base)

Input: $G = (V, E)$, τ, $(s, t) \in V \times V$;
Output: $|\mathcal{P}_{s,t}|$
1: $D \leftarrow \infty$, construct the higher-order graph $\mathcal{G}$ from G;
2: $S \leftarrow \{m | s \in V_m, m \simeq \tau\}$, $T \leftarrow \{m | t \in V_m, m \simeq \tau\}$;
3: **for** $m_i \in S$, $m_j \in T$ **do**
4: $\quad \mathcal{P} \leftarrow$shortest sequence of $m \simeq \tau$ linking m_i and m_j in $\mathcal{G}$.
5: $\quad$ Update $D \leftarrow |\mathcal{P}|$ if $D > |\mathcal{P}|$;
6: **return** D.

Definition 5 (Motif-orbit[41]). *Given motif τ, nodes $a \in V_\tau$ and $b \in V_\tau$ are in the same orbit if there is an injective mapping $f : V_\tau \to V_\tau$ with $f_a = b$ and $f_b = a$ such that $(u, v) \in E_\tau \iff (f_u, f_v) \in E_\tau$. Motif-orbit is the motif with seed node on each orbit.*

For example, triangle only has one motif-orbit (e.g., $\tau_{3,1}$ in Fig. 2) and the tailed-triangle has three motif-orbits (e.g., $\tau_{4,1}$, $\tau_{4,2}$ and $\tau_{4,3}$ in Fig. 2). Based on these motif-orbits, we propose the Motif-Orbit-Decomposition Index, namely MOD-Index. It is a triplet $(s, \mathcal{T}, M)$, where $s \in V$ is the seed node and $M = \{m | s \in V_m\}$ is the set of motif-instances to be indexed. $\mathcal{T} = (B, T)$ is the organization of the index, where B is the set of motif-orbits and T is the index structure, e.g., the linkages among different motif-orbits. For each node, we construct a MOD-Index to organize the motif-instances around it.

For each motif-orbit $\tau_{i,j} \in B$, there is a set of motif-instances $M_{i,j} \subseteq M$ attached to it, where $M_{i,j} = \{m | m \simeq \tau_{i,j} \& f_s = seed\}$. Here $f_s = seed$ means that node s from the motif-instance should be mapped to the seed of the motif-orbit. To handle possible queries with different motifs, we need to index all the motif-orbits from the possible motifs. In this paper, we focus on the motifs τ where $|V_\tau| < 6$, which is a common set up in the motif counting and analysis area[3] [2, 10, 15, 37, 39]. For example, Fig. 2 shows part of the index, which indexes seven motifs:

τ_1 •—•, τ_2  , τ_3  , τ_4  , τ_5  , τ_6  and τ_7  .

We describe the building process of the MOD-Index in Algorithm 2. Given the graph G, a set of motifs $\mathcal{M}$, the first step is to construct the organization of motif-orbits $\mathcal{T} = \{B, T\}$. As mentioned before, we detect motif-orbits following Definition 5 (line 1), and organize them in two directions in order to detect common sub-structures (line 2). In the vertical direction, the motif-orbits are organized by the shared common sub-structures (marked by dashed rectangle in

[3] Theoretically speaking, the framework supports arbitrary size of motif, but the number of motifs increases exponentially as the size of motif rises. Thus the motifs of 2-5 nodes provide good tradeoff between the practical usage and complexity [8].

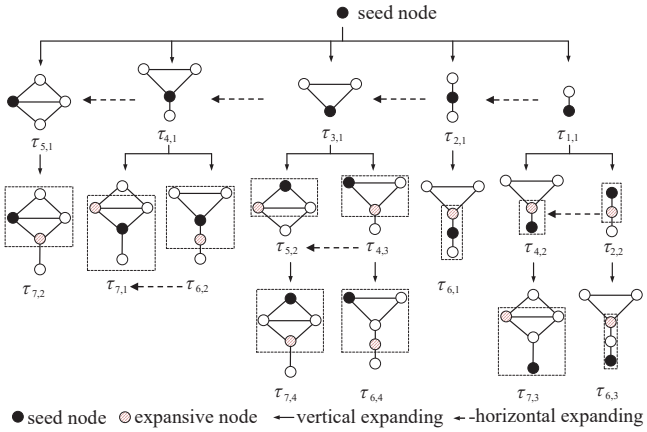


Figure 2: Indexing the orbits of motifs $\{\tau_i | i=1, ..., 7\}$[4].

Fig. 2), e.g., $\tau_{5,2}$ and $\tau_{4,3}$ share $\tau_{3,1}$. To detect the shared common sub-structures, we expand from the seed in each motif-orbit step by step and see if there is any common sub-structure. If so, we merge the corresponding motif-orbits by the same parent. Note that in each step, we only expand one layer from the seed. In the horizontal direction, the motif-orbits of the same parent are organized by the number of edges, e.g., from $\tau_{1,1}$ to $\tau_{2,1}$ and finally $\tau_{5,1}$ (marked by dashed arrow in Fig. 2). There are two benefits for such organizations in the following materializing operations. First, the incremental searching manner reuses the current searching results, which are then extended to bigger motif-orbits. Second, the motif-instances sharing common sub-structures are thus compressed to reduce space cost. Note that $\mathcal{T}$ is independent from the data graph and it can be calculated in constant time since the number of motif-orbits are small and fixed.

Next, we materialize the motif-instances into the corresponding motif-orbits. For each node $s \in V$, we use it as the seed node and initialize a queue Q for it. In the queue, each motif-orbit $\bar{\tau}$ is associated with the matched motif-instances by $\bar{\tau}.ins$ (lines 4-11). For each motif-orbit popped out from the queue, its child motif-orbit $\bar{\tau}'$ will be enumerated in the horizontal order. Then the motif-instances of $\bar{\tau}'$ can be expanded from two direction: its parent $\bar{\tau}$ (vertical direction) or its precursor $\bar{\tau}'.last$ (horizontal direction). We compare the cost of the two directions and use the one with smaller cost to expand. For example, $\tau_{4,1}$ is better expanded from $\tau_{3,1}$ rather than from the seed. We record the expanding trace and thus only need to materialize the new nodes and edges from last motif-instances. There are at most $N^s_{\tau'} = \binom{d_{\max}^{\phi_{\tau'}}}{|V_{\tau'}|}$ motif-instances that contain node s to be materialized, where $\phi_{\tau'}$ is the biggest shortest path distance that starts from the seed in the motif-orbit $\tau' \in B$ (e.g., $\phi_{\tau_{5,1}} = 1$ and $\phi_{\tau_{5,2}} = 2$) and $d_{\max}$ is the maximum degree. Therefore, the space cost of MOD-Index is $\sum_{s \in V} \sum_{\tau' \in B} N^s_{\tau'}$.

We demonstrate the materializing process of τ_6 in Table 1. Given s as node 9 from Fig. 1(a), we find from B that τ_6 has four motif-orbits $\tau_{6,j}, j = 1, ..., 4$, and extract their organizations from T, e.g., Fig. 3(a). It is easy to find that there is no motif-instances of $\tau_{2,1}$, $\tau_{4,1}$ or $\tau_{5,1}$, thus no motif-instances can be materialized for $\tau_{6,1}$ or $\tau_{6,2}$.

[4]Here we use $\tau_{i,j}$ to denote a motif-orbit, where i is the id of the motif and j is the orbit-id of motif τ_i.

Instead, $\tau_{1,1}$ and $\tau_{3,1}$ get materialized in the first layer, then $\tau_{2,2}$ and $\tau_{4,3}$ got materialized in the second layer, and finally $\tau_{6,3}$ and $\tau_{6,4}$ got materialized in the third layer. In this process, we do not need to enumerate all the permutations and terminate the searching process when there is no further instances to be matched. Though we show all the nodes of each motif-instance in Table 1, only new nodes need to be materialized in the index, since the common substructures are already materialized in layers above.

Algorithm 2 MOD-Index Construction (MODC)

Input: $G = (V, E)$, $\mathcal{M}$, k
Output: Φ
1: $B \leftarrow \{\tau_{i,j} | \tau_{i,j}$ is motif-orbit of τ_i, $\tau_i \in \mathcal{M}\}$;
2: $T \leftarrow$organizations of B from vertical and horizontal directions;
3: $\mathcal{T} \leftarrow (B, T)$, $\Phi \leftarrow \emptyset$;
4: **for** $s \in V$ **do**
5: $M \leftarrow \emptyset$, $Q \leftarrow \emptyset$, $seed.ins \leftarrow \{s\}$, $Q.enqueue(seed)$;
6: **while** $Q \neq \emptyset$ **do**
7: $\bar{\tau} \leftarrow Q.dequeue()$, $H \leftarrow \bar{\tau}.ins$, $M \leftarrow M \cup H$;
8: **if** $\bar{\tau}.level < k$ **then**
9: **for** each child $\bar{\tau}'$ of $\bar{\tau}$ in horizontal order **do**
10: $H' \leftarrow$ expanding from $\bar{\tau}.ins$ and $\bar{\tau}.last.ins$;
11: $\bar{\tau}'.ins \leftarrow H'$, $Q.enqueue(\bar{\tau}')$;
12: $\Phi \leftarrow \Phi \cup (s, \mathcal{T}, M)$;
13: **return** Φ.

After materializing the motif-orbits, we can directly query the motif-instances around a specific node, as well as the ones that are τ-adjacent to them, simply by switching the seed node. However, sometimes it is suggested to only materialize part of the index to reach a better tradeoff between the query time and the space cost, especially when the space is limited. In Algorithm 2, we control it by a parameter k, e.g., only materializing the motif-orbits in the top-k layers of $\mathcal{T}$. Then in the online querying process, there are two cases when accessing the MOD-Index: the motif-instances $m \simeq \tau$ can be directly accessed if all motif-orbits of τ are materialized in the top-k layers; otherwise, we need to expand from the current materialized motif-orbits to find $m \simeq \tau$.

Algorithm 3 MOD-Index Query (MODQ)

Input: τ, $q \in V$, Φ
Output: M
1: $\mathcal{M} \leftarrow \{\tau' | \tau'$ is a motif-orbit of $\tau\}$, $M \leftarrow \emptyset$;
2: **for** $\tau' \in \mathcal{M}$ **do**
3: **if** $\phi_{\tau'} \leq \Phi.k$ **then** $M \leftarrow M \cup \Phi_q.\tau'.ins$;
4: **else** $\hat{\tau} \leftarrow$ nearest precursor of τ' in Φ that is materialized;
5: **for** $m \in \Phi_q.\hat{\tau}.ins$ **do**
6: $M \leftarrow M \cup \{m' | m' \simeq \tau' \& m' \in m\}$;
7: **return** M.

We summarize the accessing of MOD-Index in Algorithm 3. Given the query node q and the motif τ, the algorithm returns the set of motif-instances that contain q and isomorphic to τ with the help of the MOD-Index Φ. For the motif-orbit τ' which is materialized in Φ (i.e., $\phi_{\tau'} \leq \Phi.k$ where $\Phi.k$ is the materializing level of Φ), we directly access the motif-instances around node q in $O(1)$, denoted as $\Phi_q.\tau'.ins$ (line 3); for those not materialized, we expand from the current materialized motif-orbits (lines 4-6). To help expand from

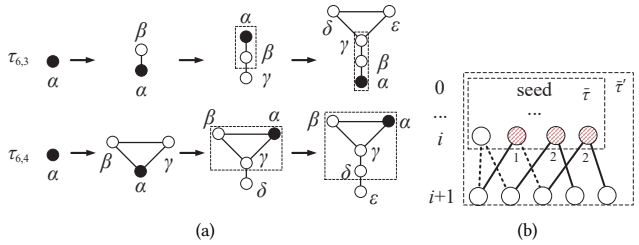


Figure 3: Expanding strategies. (a) Traversal of two branches of the index to find organizations of $\tau_{6,3}$ (top) and $\tau_{6,4}$ (bottom). (b) Detect expansive nodes and their expansive degree by edge-cover from i-th hop to $(i+1)$-th hop from the seed.

current materialized motif-orbits, we define the *expansive nodes*, i.e., the nodes in the motif-orbit whose neighbors will be searched for further expansion. The expansive nodes are marked dashed pink in Fig. 2. For example, given the motif-instances of $\tau_{2,2}$ materialized in the index, the instances of $\tau_{6,3}$ can be found by expanding the corresponding dashed pink node of $\tau_{2,2}$. To detect the expansive nodes automatically, we first rank the nodes in the motif-orbit according to its distance to the seed. Then we only search for the neighbors of the nodes at the bottom of the motif-orbit (e.g., i-th hop of $\bar{\tau}$ in Fig. 3(b)). In other words, the expansive nodes must be the furthest nodes from the seed in the current motif-orbit. Then we use minimum number of edges from these nodes to cover the nodes in the $(i+1)$-th hop of $\bar{\tau}'$, where $\bar{\tau}'$ is the child of $\bar{\tau}$ in $\mathcal{T}$. For example, we can discover all the new nodes by the solid lines in Fig. 3(b), and the endpoints of the solid lines in $\bar{\tau}$ are the expansive nodes of $\bar{\tau}$. In this manner, the expanding strategy does not discover motif-instances repeatedly. Therefore, we do not need to check whether the discovered motif-instances are duplicated (the motif enumerating algorithms that allow duplicates will lead to wrong results in the motif-path based applications), and thus time is saved.

Note that each expansive node is associated with an *expansive degree*, denoting the number of nodes to be expanded. For example, the expansive degree of the expansive nodes in Fig. 3(b) are 1,2,2 respectively. Therefore, each expansion costs $\binom{d_{\max}}{d_e}$ where d_e is the maximum expansive degree. We show in the supplement material that $d_e \leq 2$ in most expansion cases [24]. Since there are at most $\phi_{\tau'}$ times expansion, the expansion costs $O(\binom{d_{\max}}{d_e}^{\phi_{\tau'}})$. Therefore, partial materialization decreases the space cost from $\sum_{s\in V}\sum_{\tau'\in B} N^s_{\tau'}$ to $\sum_{s\in V}\sum_{\tau'\in B_k} N^s_{\tau'}$, in the cost of another $O(\binom{d_{\max}}{d_e}^{\phi_{\tau'}-k})$ expansion in the online querying phase when $\phi_{\tau'} > k$. Here B_k is the set of motif-orbits in the top-k levels of $\mathcal{T}$. Note that we can also use the expansive nodes to construct the MOD-Index, which means the time complexity of MODC is $O(\sum_{s\in V}\sum_{\tau'\in B_k}\binom{d_{\max}}{\tau'.d_e}^k)$ where $\tau'.d_e$ is the expansive degree of motif-orbit τ'. We report the proof in supplemental material [24]. The MOD-index provides more chances for direct access when more layers of $\mathcal{T}$ are materialized, in the cost of more space overhead. As we will show in the experiment part, the time-space tradeoff can be balanced by tuning the parameter k.

[5]Here p.g. denotes pattern graph (the branch of $\mathcal{T}$ in the MOD-Index) and d.g. denotes data graph (the matching results).

Table 1: Demonstration of materializing τ_6 from Fig. 2, with node 9 as the seed and expansive nodes marked bold[5].

$\tau_{6,3}$	p.g.	$\boldsymbol{\alpha}$	→	$\boldsymbol{\alpha\beta}$	→	$\boldsymbol{\alpha\beta\gamma}$	→	$\boldsymbol{\alpha\beta\gamma\delta\epsilon}$
	d.g.	**9**	→	9**8**	→	987	→	terminate.
			→	9**6**	→	967	→	terminate.
					→	964	→	96412
$\tau_{6,4}$	p.g.	$\boldsymbol{\alpha}$	→	$\boldsymbol{\alpha\beta\gamma}$	→	$\boldsymbol{\alpha\beta\gamma\delta}$	→	$\boldsymbol{\alpha\beta\gamma\delta\epsilon}$
	d.g.	**9**	→	98**6**	→	9864	→	98641
							→	98642
			→	96**8**	→	terminate.		

3.2.3 A Heuristic Algorithm. Given the source node s, we first search all motif-instances containing s with the help of the MOD-Index. After that, we use another node in the motif-instance as the seed for searching motif-instances. The process is repeated until target node t is discovered. However, it is possible to detect motif-path containing redundant motif-instances. For example, from node 3 in Fig. 1(a), the 2-hop $(3 \to 1 \to 2)$ and $(3 \to 2 \to 1)$ actually find the same triangle. Removing duplicates is expensive. We avoid adding duplicate motif-instances into motif-path by introducing Lemma 1, with three status marked on each node: *"searched"*, *"discovered"* and *"undiscovered"*. We call a node "searched" if it has been used as the seed and thus all motif-instances containing this node has been searched. A node is marked as "discovered" once this node has been discovered by any motif-instance at the current time. For other nodes, we call it "undiscovered".

LEMMA 1. *An incremental search fulfilling c1, c2 and c3 will find the shortest motif-path without exploring redundant motif-instances.*

- c1. *motif-instance containing any "searched"-node should not be added into any shortest motif-path candidate;*
- c2. *only select node with status "discovered" as next seed;*
- c3. *only add the motif-instances with at least one "undiscovered" node into the shortest motif-path candidate.*

Limited by the space, we put the proof of Lemma 1 in supplemental material [24]. Besides the incremental search manner described above, the searching direction may be far from optimal. In other words, it is essential to develop a method to select better seeds to pass to the MOD-Index for processing in the next step, which may lead the motif-path to reach target node t earlier. To reduce the searching space, we generalize the heuristic search to support shortest motif-path search. It can speed up the searching process by selecting seed smartly from candidates of $\mathcal{P}_{s,t}$. We maintain a priority queue to store the nodes marked as "discovered" currently, and for each node p in the priority queue, we estimate its shortest motif-path distance to t from s via the current "discovered" node p. Then we select the node in the priority queue with shortest estimated motif-path distance as the next seed, since this candidate may obtain higher probability to be the shortest motif-path.

We achieve the shortest motif-path estimation by $|\mathcal{P}^p_{s,t}| = |\mathcal{P}_{s,p}| + h(p,t)$, $h(p,t) \leq |\mathcal{P}_{p,t}|$. Here $\mathcal{P}_{s,p}$ is the shortest motif-path from s to the current "discovered" node p, which is already found out, and $h(p,t)$ is the heuristic function which can estimate the lower bound of $|\mathcal{P}_{p,t}|$. Therefore, the length of shortest motif-path via node p, denoted as $|\mathcal{P}^p_{s,t}|$ can be estimated.

We use $h(p,t) = |P_{p,t}|/\Theta_\tau$ to provide lower bound for $|\mathcal{P}_{p,t}|$, where $|P_{p,t}|$ is the shortest path distance between p and t in graph G, and Θ_τ is the diameter of the motif τ. The function uses the diameter of the motif to cover the shortest path, thus providing a lower-bound to the shortest motif-path distance. Since the shortest-path distance are widely supported in the popular graph databases nowadays, the lower-bound can be calculated efficiently. In this paper, we pre-calculate the shortest-path distances in the offline phase when building the MOD-Index and attach them into the MOD-Index for further queries. The pseudocodes of shortest motif-path search is described in Algorithm 4. According to Lemma 1, SMP will call function MODQ at most $|V|$ times, each with a cost of $O(1)$ with materialized motif-orbits in the MOD-Index, or with a higher expanding cost. We analyze the complexity in supplemental material [24].

Algorithm 4 Shortest Motif-Path (SMP)

Input: $G = (V, E)$, τ, $(s, t) \in V \times V$
Output: $|\mathcal{P}_{s,t}|$
1: $Q \leftarrow \emptyset$, $d \leftarrow \infty$;
2: $Q.enqueue(s)$;
3: **while** $Q \neq \emptyset$ & t is not discovered **do**
4: Rank $p \in Q$ according to $|\mathcal{P}^p_{s,t}|$;
5: $q \leftarrow Q.dequeue()$;
6: $M \leftarrow$MODQ(τ, q, Φ), and remove m from M where m contains "searched" nodes or all node in m are marked "discovered";
7: Mark q as "searched";
8: **for** $v \in V_m$, $m \in M$, v is "undiscovered" **do**
9: Mark v as "discovered";
10: **if** $v = t$ **then return** $|\mathcal{P}_{s,t}|$;
11: $Q.enqueue(v)$;
12: **return** ∞.

3.2.4 A Two-Phase Algorithm. As we will discuss in Section 4, some graph mining tasks ask for all-pairs shortest paths, rather than a single path. Thus we propose a two-phase algorithm which is more efficient when there is a need for all-pairs shortest motif-paths.

In this first phase, we compute the motif-adjacency matrix W with the help of the MOD-Index. We define $W_{i,j} = 1$ if $\exists m \simeq \tau, (v_i, v_j) \in V_m \times V_m, i \neq j$ and otherwise 0. In lemma 2, we prove that the shortest motif-path distance is equal to the shortest path distance in W, denoted as $d_W(s,t)$. Limited by the space, we put the proof in supplemental material [24]. For example, $\mathcal{P}_{3,9}$ is highlighted in Fig. 1(b), and the number of dashed edges in each motif-instance (i.e. triangle) forms $d_W(3,9)$. In this example, $|\mathcal{P}_{s,t}| = d_W(s,t) = 4$.

Lemma 2. $|\mathcal{P}_{s,t}| = d_W(s,t)$.

Lemma 3. *Given motif τ, the construction of its motif-adjacency matrix W only requires materializing one motif-orbit $\bar{\tau}$.*

Thus in the second phase, the all-pairs shortest motif-path distance can be calculated by a traditional unweighted algorithm, e.g., breadth-first search of $|V|$-rounds. Therefore, the key point of the two-phase algorithm is the construction of the motif-adjacency matrix W. As shown in Lemma 3, we only need to match one motif-orbit of τ to build W. Limited by space, we prove it in supplement [24]. To reduce the possible expanding cost, we choose the motif-orbit in the top layer of $\mathcal{T}$, demoted by τ^*, because it is of higher probability to be materialized, and of less expanding cost if not so. For example, $\tau_{6,1}$ in Fig. 2 should be used when τ_6 serves as the query motif. Every time we find a new motif-instance m_{τ^*}, we mark the corresponding elements in the matrix, e.g., $W_{i,j} = W_{j,i} = 1, (i,j) \in E_{m_{\tau^*}}$. Therefore, for the first phase, it costs $O(\binom{d_{\max}^{\phi_\tau}}{|V_\tau|})$ to get $W_{i,j}$ if the motif-orbits of τ are materialized in the index; otherwise, we call MODQ(τ^*, q, Φ) of $O(|V|)$ times to search for the motif-instances and another $O(\binom{d_{\max}^{\phi_\tau}}{|V_\tau|})$ cost to obtain $W_{i,j}$. Note that we pass τ^* to MODQ to make $\mathcal{M} \leftarrow \{\tau^*\}$ (line 1 in Algorithm 3) for speedup. For the second phase, the all-pair shortest path search costs $O(|V|(|V| + |E_W|))$ where $|E_W|$ is the edge set of the motif-adjacency matrix.

4 MOTIF-PATH BASED GRAPH MINING

In this section, we start applying motif-path into graph mining, including missing link prediction (cf. Section 4.1), local graph clustering (cf. Section 4.2) and node ranking (cf. Section 4.3).

4.1 Motif-path based Link Prediction

In this section, we use motif-path to predict the missing link between two nodes, by extending the popular path-based approaches, Katz Index and Graph Distance, into motif-path based versions.

Katz Index (KI) [17]. A potential missing link is likely to have more short paths between the query nodes.

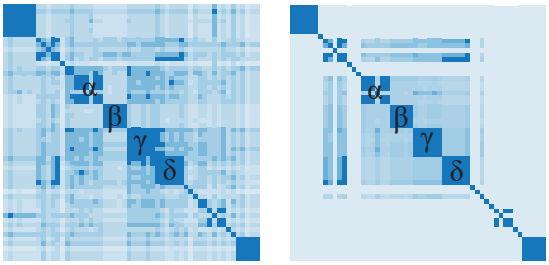
Graph Distance (GD) [20]. A potential missing link is likely to have small shortest path distance between the query nodes.

Given a pair of nodes (x, y) to be predicted, each approach will generate a score $g(x, y)$, denoting the probability that there is a missing link between x and y, which can be formally defined as below: $g_{KI}(x,y) = \sum_{l=1}^{L} \epsilon^{l-1} \cdot |\mathbb{P}^l_{x,y}|$, $g_{GD}(x,y) = \frac{1}{|P_{x,y}|}$;
$g_{MKI}(x,y) = \sum_{l=1}^{L} \epsilon^{l-1} \cdot |\mathbb{MP}^l_{x,y}|$, $g_{MGD}(x,y) = \frac{1}{|\mathcal{P}_{x,y}|}$.

Here $\mathbb{P}^l_{x,y} = \{P_{x,y} || P_{x,y}| = l\}$ is the set of all length-l paths between x and y, and ϵ is the weighting parameter ($\epsilon < 1$). Obviously, the pair of nodes with higher $g(x,y)$ value is more possible to form links. Then we extend KI into *Motif-path-based Katz Index* (MKI), and GD into *Motif-path-based Graph Distance* (MGD). Here $\mathbb{MP}^l_{x,y} = \{\mathcal{P}_{x,y} || \mathcal{P}_{x,y}| = l\}$ is the set of motif-paths between x and y with length l. The pseudocodes of applying motif-path in link prediction is described in Algorithm 5. Note that L and ϵ are the system parameters in MKI. We use SMP to answer MGD (so MGD shares the

Algorithm 5 Motif-path based link prediction

Input: $G = (V, E)$, τ, $(s, t) \in V \times V$ & $(s, t) \notin E$
Output: $g(s, t)$
#Motif-path based Graph Distance (MGD)
$|\mathcal{P}_{s,t}|$ =SMP(G, τ, s, t);
return $g_{MGD}(s, t) = \frac{1}{|\mathcal{P}_{s,t}|}$.
#Motif-path based Katz Index (MKI)
$C_1 \leftarrow \{s\}$;
for $l \leftarrow 1 : L$ **do**
$M \leftarrow$MODQ(τ, p, Φ), $p \in C_l$;
$|\mathbb{MP}^l_{s,t}| \leftarrow |\{t' | t' \in V_m, m \in M, t' = t.\}|$;
$C_{l+1} \leftarrow \{x | x \notin C_l, x \in Vm, m \in M.\}$;
return $g_{MKI}(s, t) = \sum_{l=1}^{L} \epsilon^{l-1} \cdot |\mathbb{MP}^l_{s,t}|$.



(a) (b)

Figure 4: PPI complex discovery in EXTE by (a) shortest path distance and (b) shortest motif-path distance with triangle.

same complexity as SMP) and its variant to answer MKI. For MKI, we search L layers with respect to motifs, each in a round of calculation. In the l-th round, we calculate M, $|\mathbb{MP}^l_{x,y}|$ and update the layer of searched nodes into C_{l+1}. Note that M is the motif-instances that contain $x \in C_l$, and $|\mathbb{MP}^l_{x,y}|$ is calculated by counting the number of motif-instances m such that $t \in V_m \& m \in M$. Since MKI needs to enumerate multiple motif-paths, the node status described in Lemma 1 need to be removed from MKI. For example, the status "searched" should not work in MKI, because MKI may pass the seed node multiple times even though the motif-instances containing it have been enumerated. Therefore, MKI at most calls MODQ for $O(|V|)$ times in each layers of searching. Limited by space, we analyze the time and space complexities in the supplemental material [24].

4.2 Motif-path based Local Graph Clustering

Recently, many studies [40, 44] demonstrate that using motifs as higher-order structure can improve the effectiveness of graph clustering. In this section, we demonstrate that motif-path can improve the effectiveness of ordinary path-based Local Graph Clustering (LGC), which typically finds the k-nearest neighbors as the local cluster. In other words, given a query node $s \in V$, LGC aims to find the local cluster of s by searching k-nearest neighbors of s with respect to shortest path distance.

Algorithm 6 Motif-path based Local Graph Clustering (MLGC)

Input: $G = (V, E)$, τ, k, $s \in V$
Output: C
1: Run SMP(G, τ, s, ∞) until k nodes are marked as "discovered";
2: $C \leftarrow \{c | c$ is marked as "discovered"$\}$;
3: **return** C.

Thus we extend LGC by finding k-nearest neighbors with respect to shortest motif-path distance, namely Motif-path based Local Graph Clustering (MLGC). It is described in Algorithm 6. Therefore, MLGC at most calls function MODQ for $O(k)$ times. Limited by space, we analyze the time and space complexities in supplement [24]. In Fig. 4, we show a protein complex composed of four dense subcomplexes $(\alpha, \beta, \gamma, \delta)$. The value of each pixel is calculated by the pairwise shortest path distance $1/|P_{s,t}|$ (Fig. 4(a)) and shortest motif-path distance $1/|\mathcal{P}_{s,t}|$ (Fig. 4(b)) respectively. Observe that the protein complex is clearly represented based on the motif-path approach while the shortest path based version is noised.

4.3 Motif-path based Node Ranking

Recently it has been proven that the node ranking results can be improved with the involvement of motifs [45]. The authors use motif-based PageRank to measure the authority of the nodes then rank them accordingly. In this paper, we propose a new method to rank the nodes based on motif-path. Centrality is an important property for the graphs, which can select the influential nodes and thus output a proper ranking. Although there are different definitions of centrality, betweenness centrality (namely BET) is seen as one of the most important one [6]. It measures the influence $C(v)$ of a given node v by calculating the times that a node appears in the shortest paths passing it: $C(v) = \sum_{s \neq v \neq t \in V} \frac{\delta_{s,t}(v)}{\delta_{s,t}}$.

Here $\delta_{s,t}$ denotes the number of shortest paths from $s \in V$ to $t \in V$ and $\delta_{s,t}(v)$ denotes the number of shortest paths from s to t that pass v. After changing the shortest path into shortest motif-path in BET, we get the motif-path-based betweenness centrality, namely MBET. Note that MBET requires all-pairs shortest motif-path, thus we use the two-phase algorithm to calculate the result. We describe the pseudocodes in Algorithm 7. It is of the same complexity as the Two-Phase algorithm in Section 3.2.4.

Algorithm 7 Motif-path based Betweenness Centrality (MBET)

Input: $G = (V, E)$, τ, $v \in V$
Output: $C(v)$
1: $\delta_v \leftarrow 0$, $\delta \leftarrow 0$, generate W from G;
2: **for** $s, t \in V \times V$, $s \neq t \neq v$ **do**
3: $l \leftarrow 1$;
4: **while** t is not discovered in W **do**
5: Calculate $\mathbb{P}^l_{s,t}$ on W;
6: **if** $|\mathbb{P}^l_{s,t}| > 0$ **then**
7: $\delta_v \leftarrow \delta_v + |\{P^l_{s,t} | v \in P^l_{s,t}, P^l_{s,t} \in \mathbb{P}^l_{s,t}\}|$;
8: $\delta \leftarrow \delta + |\mathbb{P}^l_{s,t}|$, break;
9: $l \leftarrow l + 1$;
10: **return** $C(v) = \delta_v / \delta$.

5 DEFRAGMENTATION WITH MOTIF-PATH

Recently, it is found that higher-order graph fragmentation is a common issue for higher-order graph [21, 40, 44, 45]. The performance of applications like motif-based clustering degrades when higher-order graph fragmentation occurs [21]. For example, authors in [21] found that the clustering accuracy is raised by 11% on average of the four datasets evaluated, after fixing the motif-fragmentation issue by injecting original graph edges into the higher-order graph. In this section, we study the defragmentation with motif-path.

As shown in Fig. 5, we calculate the size and the diameter (i.e., the longest shortest motif-path distance) for each connected component in the higher-order graph (called *motif-component*) of AMAZ, a popular co-purchasing dataset from Amazon [42]. There is only one point for original graph (marked red) since AMAZ is connected. Although the original graph is connected, the higher-order graph tends to be fragmented into a large number of motif-components and isolated nodes. Also, the motif-components tend to be with bigger diameter values since many weak edges (e.g., the edges

that cannot form triangles) are removed when building the higher-order graph. The connectivity between nodes within different motif-components disappears in the higher-order graph.

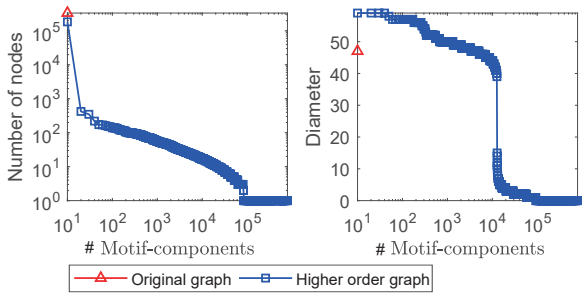


Figure 5: The effect of higher-order graph fragmentation from AMAZ with triangle as the motif pattern.

Recently, researchers from [21] propose a defragmentation approach, which detects motif-components first, then injects the edges from G into the higher-order graph to obtain higher clustering effectiveness. However, this approach has a big influence on the higher-order graph structure, thus is not suitable for use with motif-paths. As we will show in the evaluations, this approach performs bad when using motif-path with it. Also, the approach only supports triangles. Therefore, there is a need to propose a new method to handle the defragmentation issue, in order to find meaningful motif-paths for nodes within different motif-components. To preserve the inner structure of each motif-component, we only inject the edges from the original graph whose endpoints are within different motif-components into the higher-order graph, namely *bridging edges*. For example, the higher-order graph of Fig. 6(a) is broken into two motif-components when triangle is used as the motif pattern. To connect the two motif-components into a connected one, edge $(4, 6)$ is used as the bridging edge and injected into the higher-order graph $\mathcal{G}$ in Fig. 6(b). After injecting the bridging edge $(4, 6)$ from Fig. 6(a) into the higher-order graph $\mathcal{G}$, the shortest motif-path distance becomes $|\mathcal{P}_{3,9}| = 4$. Besides, this approach is more efficient than the approach from [21]. As we will show in Section 6, the number of bridging edges is much smaller than the number of the injected edges from [21], making it applicable for large graphs.

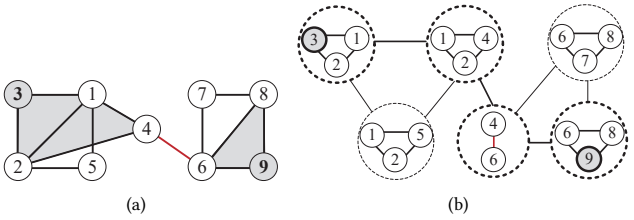


Figure 6: Connect the motif-components by injecting the bridging edges (marked red) from (a) graph G to (b) higher-order graph $\mathcal{G}$. The shortest motif-path between nodes $(3, 9)$ is highlighted. The triangle is used as the motif pattern.

After injecting the edges, we search shortest motif-path, namely enhanced shortest motif-path (ESMP) with respect to both motif-instances and the injected bridging edges. To search the enhanced shortest motif-path, we first search the τ-connected motif-instances, and only employ bridging edges when the target node is not reachable from the source node via motif-instances. In Fig. 6, starting from node 3, we search bridging edges to discover node 9 after searching all the nodes in the motif-component that contains node 3. The pseudocodes are described in Algorithm 8. ESMP calls MODQ at most $O(|V|)$ times during the path search with another $O(|E|)$ searching for bridging edges.

Algorithm 8 Enhanced Shortest Motif-path (ESMP)

Input: $G = (V, E)$, τ, $(s, t) \in V \times V$
Output: $|\mathcal{P}_{s,t}|$
1: $Q \leftarrow \emptyset$, $d \leftarrow \infty$, $Q.enqueue(s)$;
2: **while** $Q \neq \emptyset$ **do**
3: $\quad q \leftarrow Q.dequeue()$, $|\mathcal{P}_{q,t}| \leftarrow$SMP$(G, \tau, q, t)$;
4: $\quad$ **if** $|\mathcal{P}_{q,t}| \neq \infty$ **then**
5: $\quad\quad |\mathcal{P}_{s,t}| \leftarrow |\mathcal{P}_{s,q}| + |\mathcal{P}_{q,t}|$.
6: $\quad\quad$ **if** $|\mathcal{P}_{s,t}| < d$ **then**
7: $\quad\quad\quad d \leftarrow |\mathcal{P}_{s,t}|$;
8: $\quad\quad\quad$ **if** $q = s$ **then return** d.
9: $\quad$ **else**
10: $\quad\quad$ **for** $v \in \{p | p$ is discovered by $q,\ p \neq q\}$ **do**
11: $\quad\quad\quad$ **for** p's neighbor w, w is not discovered yet **do**
12: $\quad\quad\quad\quad Q.enqueue(w)$, $|\mathcal{P}_{s,w}| \leftarrow |\mathcal{P}_{s,p}| + 1$;
13: **return** d.

Note that ESMP is an online algorithm but it may visit the whole higher-order graph if s and t are located in different motif-components. If offline pre-processing is allowed (e.g., MOD-Index construction), a practical approach is to develop ESMP into a two-phase algorithm. In the offline phase, the motif-components are detected and the list of bridging edges is written down. These operations are involved in the process of MOD-Index construction, since the switching of seed nodes is naturally a motif-component detection process. In the online process, given a pair of query nodes (s, t), the algorithm runs in the same manner of SMP if s and t are located in the same motif-component. Otherwise, the algorithm wil enumerate both motif-instances and bridging edges that contain the seed. In other words, motif-instances and bridging edges are treated equally in this case (e.g., Fig. 6(b)).

6 EXPERIMENTS

In this section, we evaluate the efficiency and effectiveness of the proposed algorithms. Two kinds of real-world datasets are used: protein-protein interaction (PPI) networks and social networks.

PPI networks. We use two PPI networks in which nodes denote proteins and edges denote the interactions. GAVI [13] is the PPI network of yeast cell. EXTE [19] is the PPI dataset of bacterias.

Social networks. We use three social networks with ground-truth communities from [42]. DBLP is a co-authorship network from computer science bibliography where two authors are linked if they publish at least one paper together. AMAZ is a co-purchasing network from Amazon, where each node is a product and two nodes are linked if these two products are frequently co-purchased. YOUT

Table 2: Datasets for link predication and local graph clustering (top), and node ranking (bottom).

Name	$\|V\|$	$\|E\|$	Deg.	Θ	$\|E_b^+\|$	Θ_b	$\|E_c^+\|$	Θ_c
GAVI	1,727	7,534	8.7	13	998	15	419K	10
EXTE	3,642	14,300	7.9	10	10K	11	573K	8
DBLP	317,080	1,049,866	6.6	23	106K	25	-	-
AMAZ	334,863	925,872	5.5	47	324K	48	-	-
YOUT	1,134,890	2,987,624	5.3	24	1.2M	24	-	-
Name	$\|V\|$	$\|V_H\|$	$\|E\|$	Θ	$\|E_b^+\|$	Θ_b	$\|E_c^+\|$	Θ_c
$DBLP_1$	54,732	16,556	83,208	37	67K	37	1,976K	32
$DBLP_2$	40,846	13,298	59,591	40	52K	40	478K	36

is a friendship network from YouTube, where each node denotes a user and there is an edge if the two users are friends.

We show the details of these datasets in Table 2, where Θ is the diameter of the original graph. $|E^+|$ is the number of edges injected to enhance the higher-order graph, with triangle as the motif pattern: $|E_b^+|$ is the number of bridging edges injected to the higher-order graph based on motif-path search and Θ_b is the diameter of the enhanced higher-order graph after injecting the bridging edges; $|E_c^+|$ is the number of injected edges by the defragmentation algorithm from [21] and Θ_c is the diameter of the enhanced higher-order graph after injecting these edges. As shown in Table 2, $|E_b^+|$ is much smaller than $|E_c^+|$, which means the bridging edge approach has less modification on the structure of the higher-order graph than the defragmentation approach from [21]. Note that the algorithm from [21] cannot terminate in one day for the social networks, thus we omit the results for these datasets. As shown in the table, Θ_c is much smaller than Θ_b because of the vast number of edges injected. We have implemented the algorithms with the higher-order graph defragmentation approach from [21]. However, this defragmentation method cannot terminate on the large datasets (e.g., DBLP, AMAZ and YOUT). For the mediate-size networks (e.g., GAVI and EXTE), the algorithms based on such defragmentation perform significantly worse than the version based on bridging edges. Limited by the space, we report its results in the supplemental material [24].

We implement the algorithms in Java, and run experiments on a machine with 4-core Intel i7-3770 3.40GHz processor and 16GB of memory. Codes and datasets are released in [23]. Next, we evaluate the efficiency of the algorithms in Section 6.1, and the effectiveness of using motif-path in three applications (link prediction, local graph clustering and node ranking) in Section 6.2, 6.3 and 6.4 respectively. In each application, we test the algorithms from different aspects (see Fig. 7, where BE denotes bridging edge based defragmentation and EM denotes the defragmentation approach from [21]). We illustrate a case study in Section 6.5.

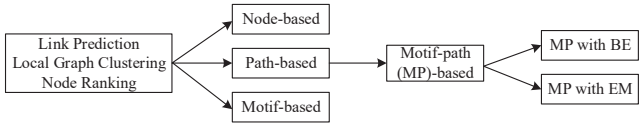


Figure 7: Hierarchy of solutions for the three applications tested in the paper.

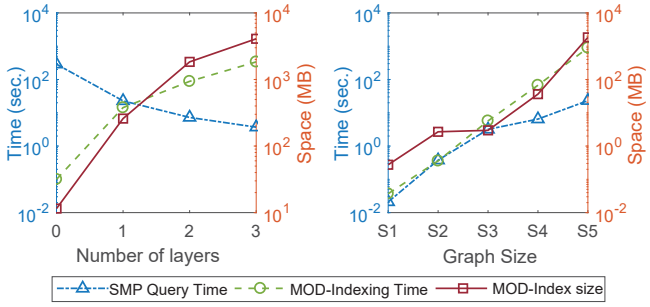


Figure 8: Time-space tradeoff with τ_7 on AMAZ (left) and synthetic graphs (right).

6.1 Efficiency Evaluation

In this section, we report the efficiency and scalability of searching Shortest Motif-Path (SMP) and Enhanced Shortest Motif-Path (ESMP) when tuning the MOD-Index. We skip the baseline algorithm since it even cannot terminate on a moderate-size dataset (e.g., GAVI and EXTE). In this section, the numbers are averaged from 500 random (s, t) queries through the graph.

First, we evaluate the materializing strategy of MOD-Index with motif τ_7, whose motif-orbits are distributed in different layers of the MOD-Index. As shown in Fig. 8, the SMP algorithm's query time is reduced in the online phase when more layers are materialized into the MOD-Index. In general, the indexing time is small, but as more layers are materialized, the space cost rises. Thus in the following sections, we materialize one layer of the MOD-Index, which is lightweight in the time cost and space cost and powerful in saving the query time. Note that the cost of calculating shortest paths are not involved in Fig. 8, since calculating shortest path distances will not change the complexity of MOD-Index construction, and it weights little in the total construction cost; hence we only analyze the materialization cost in this section. Limited by the space, we report the breakdown of the indexing cost in supplement [24].

Next, we evaluate the scalability of the framework by generating five synthetic graphs, namely S_{i-1} with the number of nodes as 2×10^i, $i = 2, 3, 4, 5, 6$. We fix the average degree as 4 and employ Barabási-Albert model [3], a widely used method to simulate real graphs. As the size of graph grows, the indexing time and space cost are increased in the similar trends. Also, we notice that the performance is even better for large graphs (e.g., S_4 and S_5). We guess it is from the fact that large networks are sparser and thus SMP obtains better efficiency with the help of the MOD-Index.

Then we evaluate the efficiency of ESMP on real-world datasets. Note that we assume that the bridging edges have been found out beforehand for ESMP. As shown in Fig. 9, the ESMP algorithm is quite efficient, e.g., most queries are finished in seconds for different motifs and different datasets. Especially, we show the performance of i-cycle and i-clique motifs on AMAZ where $i = 3, 4, 5$. Note that the motif-path based on triangle is more costly since triangle is more significant in the graph and thus more motif-instances need to be enumerated [30]. We also report the two complex motifs τ_6 and τ_7 from Fig. 2, which are more costly to search in both SMP and ESMP. Then we report the query time and MOD-Indexing

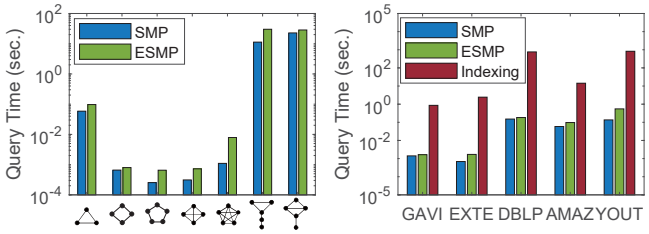


Figure 9: Efficiency evaluation on real world datasets.

time on different real-world datasets when triangle is used. We notice that the cost of ESMP is only a little more than the SMP algorithm. However, as we show in the following sections, higher graph mining effectiveness is obtained when ESMP is used.

6.2 Motif-path based Link Prediction

Following [29, 43], we restrict $L = 4$ and $\epsilon = 0.01$ for both KI and MKI. After the higher-order graph defragmentation by injecting bridging edges, we run MKI and MGD on the enhanced higher-order graph, namely MKI-b and MGD-b respectively. Following the setup of [20], in each iteration we randomly pick a missing link (x^+, y^+) (positive sample) and a non-existent link (x^-, y^-) (negative sample) in the graph and compare their scores, denoted as g^+ and g^- respectively. After c iterations, we denote c_1 as the number of iterations with $g^+ > g^-$ and c_2 as the number of iterations with $g^+ = g^-$. In this section, we set the number of iterations $c = 1000$. Then we use the standard metrics *Area Under Curve* (AUC) [20] to measure the effectiveness: $AUC = \frac{2 \cdot c_1 + c_2}{2 \cdot c}$. Following [20], we independently sample the positive and negative pairs which come from the same shortest-path-distance distribution. In other words, the expected values of $|P_{x^+, y^+}|$ and $|P_{x^-, y^-}|$ are same. Otherwise, positive pairs will be much nearer to each other than the negative pairs, making any predicting method easy to obtain high effectiveness.

We also compare the effectiveness of our methods with the state-of-the-arts, which are generally divided into two classes. First, we evaluate the traditional missing link prediction methods, including Common Neighbors (CN), Jaccard Coefficient (JC), Adaminc/Adar (AA), Preferential Attachment (PA), Friends Measure (FM), Hitting Time (HT) and Rooted PageRank (RPR). Following the standard setup, we use the damping parameter $\alpha = 0.85$ in RPR [1]. Theses methods are easily to be adopted but the AUC score only varies from 0.5 to 0.7 in most cases. Note that HT and RPR are the Markov-chain-based approach to approximate KI by random walk. Second, we evaluate the approaches which employs motif-based features.

Motif-based Common Neighbor (MCN) [7] is the extended work from CN to predict missing links, with scoring function $g_{MCN} = |\Gamma_m(x) \cap \Gamma_m(y)|$, where $\Gamma_m(x) = \{m | x \in m \& m \simeq \tau\}$ denotes the set of motif-instance which contains the node x. We use τ_5, which obtains best performance among the motifs listed in the paper.

Motif-based Link Prediction (MLP) [1] counts the motif-instances around the missing link and generate motif distribution as feature vector. All 3-5 node motifs are used in this work. Then classifiers are trained for prediction. We choose Gradient Boosting (GB), which obtains best effectiveness among all classifiers listed in the paper.

Table 3: MKI/MGD performance with AUC reported. Numbers of top-3 highest are marked bold.

Method	GAVI	EXTE	DBLP	AMAZ	YOUT	Time
CN	0.72	0.56	0.77	0.62	0.54	0.1s
JC	0.70	0.48	0.55	0.52	0.44	0.1s
AA	0.75	0.57	**0.81**	0.65	0.52	0.1s
PA	0.59	0.76	0.64	0.63	0.76	0.2s
FM	0.65	0.65	0.59	0.64	0.69	0.1s
HT	0.60	0.70	0.64	0.59	0.53	0.1s
RPR	0.61	0.51	0.76	0.62	0.48	0.1s
MCN	0.67	0.62	0.75	0.61	0.65	10.9m
MLP+GB	**0.89**	**0.83**	**0.82**	**0.72**	**0.83**	> 24h
KI	0.69	0.60	0.69	0.60	0.63	39.1s
MKI	0.71	0.66	0.74	0.63	0.66	3.7m
MKI-b	**0.76**	**0.87**	**0.77**	**0.73**	**0.76**	4.9m
GD	0.50	0.50	0.50	0.50	0.50	1.2s
MGD	0.67	0.63	0.65	0.66	0.55	52.2s
MGD-b	**0.75**	**0.84**	0.72	**0.81**	**0.87**	71.0s

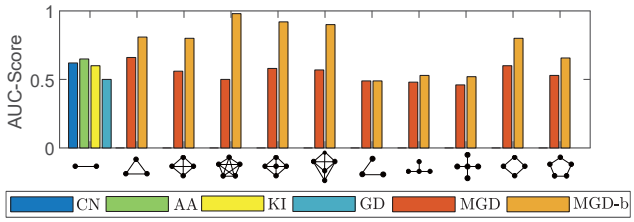


Figure 10: Evaluation of generic motifs (edge, cliques, quasi-cliques, stars and cycles) for link prediction on AMAZ.

As shown in Table 3, the motif-path based approach can achieve pretty high effectiveness among all the methods listed. Compared with the path-based methods (KI and GD), the effectiveness is significantly improved. From the table, the prediction performance can be further improved after the bridging edges are injected (cf. MKI-b and MGD-b). MGD obtains good effectiveness in most datasets, which means that the shortest motif-path distance itself is an important indicator for missing link prediction. Note that MLP+GB obtains outstanding results in most datasets, at the cost of expensive feature learning and classifier training. As shown in the table, our methods are competitive. Here "Time" is averaged from different datasets[6], e.g., Time(CN)=$\sum_{G \in \mathcal{D}} \text{Time}_G(\text{CN}) \times |V_G| / \sum_{G' \in \mathcal{D}} |V_{G'}|$ where $\mathcal{D}$ is the set of datasets and $\text{Time}_G(\text{CN})$ is the running time of CN on dataset G.

In Fig. 10, we evaluate the cliques, quasi-cliques (minimum degree bigger than 2), stars and cycles in the link predication by motif-path based Graph Distance. They are usually called *generic motifs* [8]. As shown in the figure, the involving of motifs do help improve the predication accuracy. However, the fragmentation issue is serious when using 4-clique or 5-clique, leading to low effectiveness in MGD but high effectiveness in MGD-b.

[6]For fair comparison, we assume that there is no motif-orbits materialized in the MOD-index, which means that the triangle-based applications are totally online in the evaluation of Section 6.2, 6.3 and 6.4.

Table 4: MLGC performance with F1-score reported. Numbers of top-3 highest are marked bold.

Method	GAVI	EXTE	DBLP	AMAZ	YOUT	Time
TECTONIC	0.39	**0.44**	-	0.37	-	$> 24h$
MAPPR	0.39	**0.42**	**0.34**	0.35	0.15	$12.4s$
EdMot	0.33	0.38	-	-	-	$> 24h$
LGC	**0.42**	0.36	0.33	**0.63**	**0.17**	$0.9s$
MLGC	**0.41**	0.30	**0.35**	**0.59**	**0.16**	$5.4s$
MLGC-b	**0.42**	**0.38**	**0.35**	**0.65**	**0.23**	$8.9s$

6.3 Motif-path based Local Graph Clustering

In this section, we evaluate the effectiveness of motif-path based local graph clustering (MLGC) and MLGC with bridging edge defragmentation (MLGC-b).

To evaluate the local graph clustering effectiveness, we use Saccharomyces Genome Database (SGD) [7], a well-known protein complex dataset, as the ground truth communities for PPI datasets, in which each protein complex is regarded as a cluster for the nodes in PPI. We also use researcher communities as the ground-truth communities for DBLP, product communities as those for AMAZ, and user communities as those for YOUT [42].

Given a node from the ground truth community P, we find its k-nearest neighbors as the cluster P'. Then precision ($|V_P \cap V_{P'}|/k$), recall ($|V_P \cap V_{P'}|/|V_P|$) and F1-score ($2\times Precision\times Recall/(Precision+Recall)$) are calculated. In Fig. 11, we report them when varying the value of k from 1 to 100. In these datasets, MLGC usually obtains higher effectiveness than ordinary path-based method (LGC), since motif-path can effectively control the searching area of k-nearest neighbor with proper compactness. In all results obtained, MLGC-b outperforms LGC and MLGC. Also, we compare MLGC with the state-of-the-arts, which use motifs for graph clustering. For each method, we follow their default setting to evaluate the performance.

Motif-Aware Graph Clustering (TECTONIC) [40] clusters the graph by reweighed motif-conductance [5] with triangles.

Local Higher-Order Graph Clustering (MAPPR) [44] use motif-based PageRank to perform local graph clustering.

Edge Enhancement Motif Clustering (EdMot) [21] enhances the higher-order graph by injecting edges and partitions it finally.

We use triangle in the metrics above, and report the results in Table 4. For the evaluations which cannot terminate within one day, we mark with a dash ("-"). All numbers are averaged from 500 queries. In the table, we fix $k = 15$ for LGC and all MLGC based approaches. As shown in Table 4, the F1-score of MLGC-b is higher than LGC and MLGC. However, MLGC is beaten by LGC in most datasets. It means that the fragmentation issue of the higher-order graph makes the performance of MLGC degrades.

Compared with the motif-based graph clustering approaches, MLGC-b outperforms the competitors in most datasets. For EXTE, the dataset is sparse, and the ground truth communities are small, thus motif-conductance based approaches obtain advantage. In general, the competitors require more time to answer the query, while the motif-path based approaches are more efficient because of the efficient algorithms to search the k-nearest neighbors.

[7] https://paccanarolab.org/static_content/clusterone/additional_information.html

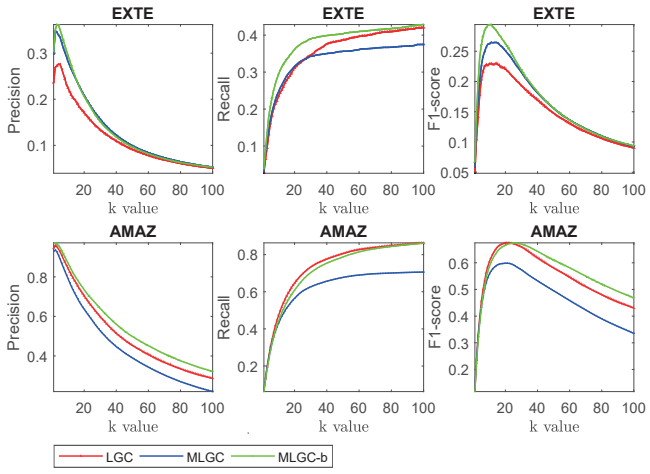


Figure 11: Local graph clustering results on EXTE and DBLP with Precision, Recall and F1-score reported.

6.4 Motif-path based Node Ranking

To evaluate the effectiveness of the ranking result, we compare the top-k users obtained from different methods by Normalized Discounted Cumulative Gain (NDCG), which is a popular metric for ranking quality measurement [45]: $NDCG_k = \frac{DCG_k}{D_k}$, $where\ DCG_k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$. Here rel_i denotes the relevance score of the i-th node and D_k is the DCG_k for the ideal ranking of the nodes. The ideal list is ranked by the relevance score. Higher $NDCG_k$ score means the ranking is closer to the ideal one thus is of higher quality.

Following [45], we use H-index of each author from DBLP as the relevance score. Both DBLP dataset[8] and H-index values are up to 2017. We report the details of the data in Table 2 and release them together with the codes [23]. To increase the percentage of nodes with H-index labels, we generate $DBLP_1$ by removing the edges whose weight is smaller than 7 and $DBLP_2$ by removing the edges whose weight is smaller than 8. In the table, we report the number of authors with H-index crawled (denoted as $|V_H|$), with 16556 and 13298 ground truth nodes in $DBLP_1$ and $DBLP_2$ respectively. We compare our method with metrics as below.

Betweenness Centrality based node ranking (BET) [6] ranks the nodes by their centrality values.

PageRank based node ranking (PR) [45] ranks the nodes by running PageRank on the original graph. We also implement Weighted PageRank (WPR), with the number of coauthored papers on edges.

Motif PageRank based node ranking (MPR) [45] first mixes the adjacency matrix A and motif adjacency matrix W (mixing parameter $\alpha = 0.4$ as suggested in the paper), then ranks the nodes by running PageRank on the mixed adjacency matrix.

Again, we compare BET with MBET-b (rank the nodes by running Algorithm 7 with bridging edges injected) and MBET (no defragmentation). As shown in the first line of Fig. 12, MBET and MBET-b both perform better than BET in $DBLP_1$. However, MBET drops quickly in $DBLP_2$ while MBET-b keeps stable. It is from the

[8] http://konect.uni-koblenz.de/networks/dblp_coauthor

Table 5: Node ranking performance with NDCG reported. Numbers of top-3 highest are marked bold. T denotes the running time (in minutes).

Method	DBLP$_1$				DBLP$_2$			
Top-k	10	30	50	T	10	30	50	T
PR	0.57	0.47	0.46	0.8	0.55	**0.52**	0.47	0.5
WPR	0.61	**0.49**	0.46	3.3	**0.65**	0.50	0.48	2.1
MPR	**0.62**	0.47	**0.47**	0.8	0.59	**0.51**	**0.48**	0.6
BET	0.60	**0.48**	**0.48**	8.4	**0.60**	0.48	**0.50**	5.2
MBET	**0.67**	0.47	0.41	0.5	0.41	0.31	0.32	0.4
MBET-c	0.23	0.27	0.30	36.0	0.18	0.26	0.23	13.5
MBET-b	**0.68**	**0.55**	**0.50**	5.5	**0.63**	**0.54**	**0.51**	4.1

fact that DBLP$_2$ is much sparser than DBLP$_1$ and thus the higher-order graph fragmentation issue is more severe in DBLP$_2$.

Then in the second line of Fig. 12, we compare MBET-b with Motif-based PageRank (MPR) [45] and Edge-based PageRank (PR). As the result shows, MPR twists with PR and both methods fall behind MBET-b. Interestingly, the top-10 ranking results of MBET-b are much better. It may come from the fact that MBET-b preserves higher-order structure thus can filter the noises in the graph, and also keeps good reachability through the graph.

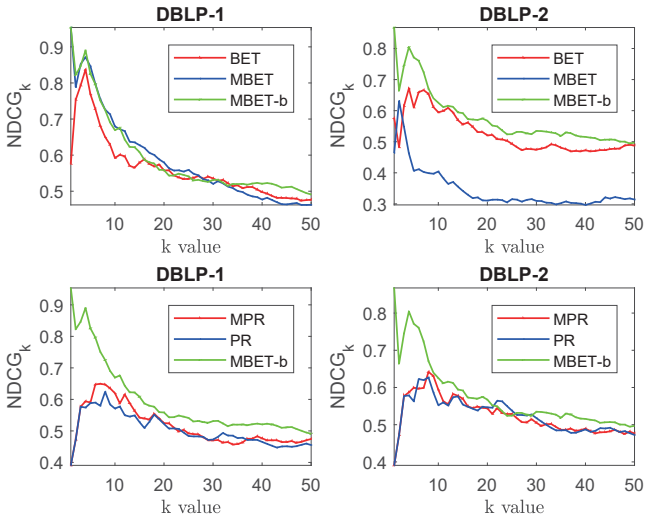


Figure 12: Node ranking comparison on DBLP datasets.

We report more results in Table 5, including the NDCG scores and running time of each method. Motif-based PageRank and weighted PageRank are normally better than the traditional PageRank approach. Also, MBET-b outperforms other approaches in most cases. The evaluation result shows that motif-path is important to reveal higher-order graph semantics and our method to handle the higher-order graph fragmentation is meaningful. We also report the running time of each algorithm. Compared to LGC, MLGC terminates earlier since the fragmentation issue stops its further expansion. MLGC-b needs more time to terminate since the bridging edges are injected and thus obtains better effectiveness.

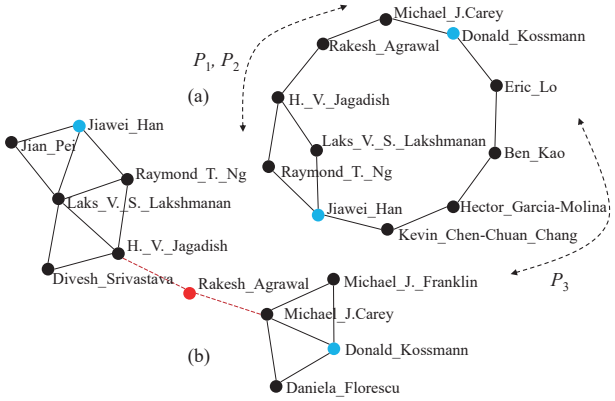


Figure 13: Demonstration of (a) shortest paths and (b) enhanced shortest motif-paths from DBLP$_1$.

6.5 Case Study

Finally, we show that a motif-path can facilitate the analysis of the relationship between two graph nodes. As shown in Fig. 13, we find out three shortest paths and eight shortest motif-paths from DBLP$_1$, given the query nodes as s=Jiawei_Han and t=Donald_Kossmann (marked as blue). Triangle is used as the motif pattern τ. Since the nodes s and t are not τ-connected in the higher-order graph, we inject bridging edges and find the enhanced shortest motif-paths. We draw the subgraph reduced from the nodes and edges within these shortest paths (Fig. 13a) and shortest motif-paths (Fig. 13b) respectively. Note that the bridging edges are marked red dashed. The subgraph in Fig. 13(b) shows that Jiawei Han and Donald Kossmann are connected by two groups: (1) data science researchers closely related to Han; and (2) industrial / system researchers related to Kossmann. These two groups are connected by Rakesh Agrawal. However, this pattern cannot be revealed by the shortest paths.

7 CONCLUSIONS

In this paper, we propose motif-path, which can discover the high-order semantics from the given graph. We develop efficient algorithms to search shortest motif-paths and then use it in graph mining tasks, with evaluations on effectiveness and scalability. Defragmentation, which connects different connected components in the higher-order graph, improves the effectiveness. In the future, we plan to explore more variants of motif-path for more applications.

ACKNOWLEDGMENTS

X. Li, C. Shan, C. Ma and R. Cheng were supported by the Research Grants Council of Hong Kong (RGC Projects HKU 17229116 and 106150091), the University of Hong Kong (Projects 104005858, 104005994), the Innovation and Technology Commission of Hong Kong (ITF project RP/029/18), and the HKU-TCL Joint Research Center for Artificial Intelligence (200009430). K. Chang and H. Cao were supported by the National Science Foundation IIS 16-19302 and IIS 16-33755, Zhejiang University ZJU Research 083650, Futurewei Technologies HF2017060011 and 094013, UIUC OVCR CCIL Planning Grant 434S34, UIUC CSBS Small Grant 434C8U, and IBM-Illinois Center for Cognitive Computing Systems Research (C3SR).

REFERENCES

[1] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. 2019. Link prediction via higher-order motif features. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 412–429.

[2] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. 2015. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining*. IEEE, 1–10.

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[4] Austin R Benson, David F Gleich, and Jure Leskovec. 2015. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 118–126.

[5] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[6] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology* 25, 2 (2001), 163–177.

[7] Jie Cao, Bentian Li, and Xiangquan Gui. 2016. Research on the Influence of Network Motif on Link Prediction. *DEStech Transactions on Computer Science and Engineering* itms (2016).

[8] Xiaowei Chen, Yongkun Li, Pinghui Wang, and John Lui. 2016. A general framework for estimating graphlet statistics via random walk. *Proceedings of the VLDB Endowment* (2016).

[9] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. 277–288.

[10] Vachik S Dave, Nesreen K Ahmed, and Mohammad Al Hasan. 2017. E-CLoG: counting edge-centric local graphlets. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 586–595.

[11] Yixiang Fang, Reynold Cheng, Xiaodong Li, Siqiang Luo, and Jiafeng Hu. 2017. Effective community search over large spatial graphs. *Proceedings of the VLDB Endowment* 10, 6 (2017), 709–720.

[12] Yixiang Fang, Zheng Wang, Reynold Cheng, Xiaodong Li, Siqiang Luo, Jiafeng Hu, and Xiaojun Chen. 2018. On spatial-aware community search. *IEEE Transactions on Knowledge and Data Engineering* 31, 4 (2018), 783–798.

[13] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dümpelfeld, et al. 2006. Proteome survey reveals modularity of the yeast cell machinery. *Nature* 440, 7084 (2006), 631.

[14] Xiaolin Han, Tobias Grubenmann, Reynold Cheng, Sze Chun Wong, Xiaodong Li, and Wenya Sun. 2020. Traffic Incident Detection: A Trajectory-based Approach. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1866–1869.

[15] Tomaž Hočevar and Janez Demšar. 2014. A combinatorial approach to graphlet counting. *Bioinformatics* 30, 4 (2014), 559–565.

[16] 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*. ACM, 1311–1322.

[17] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[18] Arun S Konagurthu and Arthur M Lesk. 2008. On the origin of distribution patterns of motifs in biological networks. *BMC Systems Biology* 2, 1 (2008), 73.

[19] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. 2006. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature* (2006).

[20] Linyuan LÃij and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* 390, 6 (2011), 1150 – 1170.

[21] Pei-Zhen Li, Ling Huang, Chang-Dong Wang, and Jian-Huang Lai. 2019. EdMot: An Edge Enhancement Approach for Motif-aware Community Detection. In *Proceedings of the 25th ACM SIGKDD*. 479–487.

[22] Xiaodong Li. 2019. DURS: A Distributed Method for k-Nearest Neighbor Search on Uncertain Graphs. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 377–378.

[23] Xiaodong Li, Reynold Cheng, Kevin Chang, Caihua Shan, Chenhao Ma, and Hongtai Cao. 2021. Code: On Analyzing Graphs with Motif-Paths. (2021). https://github.com/li20mp/motif-path.

[24] Xiaodong Li, Reynold Cheng, Kevin Chang, Caihua Shan, Chenhao Ma, and Hongtai Cao. 2021. Supplemental Material: On Analyzing Graphs with Motif-Paths. (2021). https://i.cs.hku.hk/~xdli/mpath-sup.pdf.

[25] Xiaodong Li, Reynold Cheng, Yixiang Fang, Jiafeng Hu, and Silviu Maniu. 2018. Scalable evaluation of k-nn queries on large uncertain graphs. In *21st International Conference on Extending Database Technology (EDBT)*. 181–192.

[26] Xiaodong Li, Reynold Cheng, Matin Najafi, Kevin Chang, Xiaolin Han, and Hongtai Cao. 2020. M-Cypher: A GQL Framework Supporting Motifs. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM)*. 3433–3436.

[27] Guanfeng Liu, Yan Wang, and Mehmet A. Orgun. 2010. Optimal Social Trust Path Selection in Complex Social Networks. In *Proceedings of the Twenty-Fourth AAAI 2010*.

[28] Guanfeng Liu, Yan Wang, Mehmet A. Orgun, and Ee-Peng Lim. 2013. Finding the Optimal Social Trust Path for the Selection of Trustworthy Service Providers in Complex Social Networks. *IEEE Trans. Services Computing* (2013).

[29] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. 2009. Similarity index based on local paths for link prediction of complex networks. *Physical Review E* 80, 4 (2009), 046122.

[30] Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, Tobias Grubenmann, Yixiang Fang, and Xiaodong Li. 2019. LINC: a motif counting algorithm for uncertain graphs. *Proceedings of the VLDB Endowment* 13, 2 (2019), 155–168.

[31] Markus Maier, Ulrike V Luxburg, and Matthias Hein. 2009. Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems*. 1025–1032.

[32] Ine Melckenbeeck, Pieter Audenaert, Didier Colle, and Mario Pickavet. 2018. Efficiently counting all orbits of graphlets of any order in a graph using autogenerated equations. *Bioinformatics* 34, 8 (2018), 1372–1380.

[33] Giovanni Micale, Rosalba Giugno, Alfredo Ferro, Misael Mongiovì, Dennis Shasha, and Alfredo Pulvirenti. 2018. Fast analytical methods for finding significant labeled graph motifs. *Data Mining and Knowledge Discovery* 32, 2 (2018), 504–531.

[34] 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.

[35] Paul Monagle, Anthony KC Chan, Neil A Goldenberg, Rebecca N Ichord, Janna M Journeycake, Ulrike Nowak-Göttl, and Sara K Vesely. 2012. Antithrombotic therapy in neonates and children: antithrombotic therapy and prevention of thrombosis: American College of Chest Physicians Evidence-Based Clinical Practice Guidelines. *Chest* (2012).

[36] Kirill Paramonov and James Sharpnack. 2019. Estimating Graphlet Statistics via Lifting. *SIGKDD* (2019).

[37] Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. 2017. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1431–1440.

[38] Kenneth H Rosen and Kamala Krithivasan. 2012. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education.

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

[40] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1451–1460.

[41] Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Jing Tao, and Xiaohong Guan. 2018. SNOD: a fast sampling method of exploring node orbit degrees for large graphs. *Knowledge and Information Systems* (2018), 1–26.

[42] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[43] Yabing Yao, Ruisheng Zhang, Fan Yang, Jianxin Tang, Yongna Yuan, and Rongjing Hu. 2018. Link prediction in complex networks based on the interactions among paths. *Physica A: Statistical Mechanics and its Applications* 510 (2018), 52–67.

[44] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD*. ACM, 555–564.

[45] Huan Zhao, Xiaogang Xu, Yangqiu Song, Dik Lun Lee, Zhao Chen, and Han Gao. 2018. Ranking users in social networks with higher-order structures. In *Thirty-Second AAAI Conference on Artificial Intelligence*.