Motif Paths: A New Approach for Analyzing Higher-order Semantics between Graph Nodes

Xiaodong Li[†], Tsz Nam Chan[†], Reynold Cheng[†], Caihua Shan[†], Chenhao Ma[†], Kevin Chang[‡]
[†]*Department of Computer Science, University of Hong Kong, Hong Kong SAR*
[‡]*Department of Computer Science, University of Illinois at Urbana-Champaign, USA*
[†]{*xdli, tnchan, ckcheng, chshan, chma2*}*@cs.hku.hk;* [‡]*kcchang@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 propose *motif-path*, which is conceptually a concatenation of one or more motif instances. We find that employing the shortest motif-paths in path-based solutions can significantly improve the accuracy of graph analysis. Because finding shortest motif-paths involves exploring an combinatorial number of motifs, we study an efficient solution. Experimental results on real and synthetic graphs show that our proposed solution can efficiently find shortest motif-paths, and the effectiveness is also better than existing path-based methods.**

***Index Terms*—motif-path, incremental search, graph analysis**

I. Introduction

Understanding the relation between two entities is an important and fundamental task in graph analytics. A common approach is to find paths between two nodes in a network and use the path information for graph analysis tasks like link prediction [1] and local graph clustering [2]. Some data analysts also find the path relation important in different networks [3], [4].

For example, Figure 1(a) illustrates the Linkedin network, which shows how Donald (Donald Kossmann, professor in ETH Zurich previously) is related to Ruibang (Ruibang Luo, assistant professor in HKU). A possible way is to find the shortest path between them, i.e., {Donald $\rightarrow$ Eric $\rightarrow$ Ken $\rightarrow$ Nikos $\rightarrow$ Ruibang}. However, this path is long, and does not give a clear intuition about how Donald and Ruibang are connected. To have a better understanding/interpretation of the relation between two nodes in graph, we can regard some nodes in this graph as a group as they have a close relationship (similar semantic meaning). For example, {Eric, Hong Va, Ken, Jiannong} and {Ken, Ben, Ruibang, Nikos} can be regarded as two research groups in different institutions (HKU and HKPolyU). Therefore, instead of finding a long path, we can conceptually "group" different segments of the path in order to give a more intuitive explanation of the relationship between Donald and Ruibang: {ETH Zurich research group $\rightarrow$ HKPolyU research group $\rightarrow$ HKU research group} (cf. Figure 1(a)), which can be easily interpreted by the users.

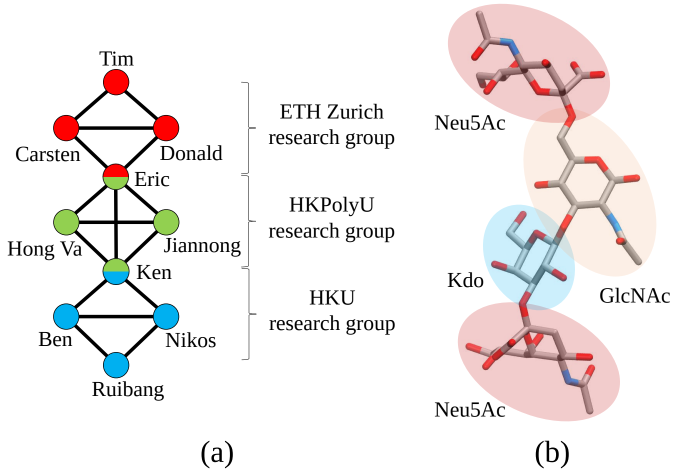


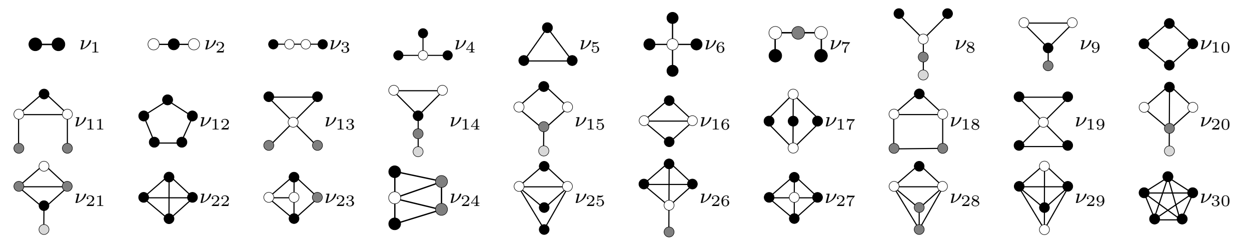
Fig. 1: Motif-path from (a) Linkedin network with research groups as motifs (◇,◇) and (b) glycan network with acids as motifs (Neu5Ac, Kdo, ClcNNAc).

On the other hand, some biologists also regard the biological compounds, e.g., branched glycan, as a path with a sequence of subcomponents, e.g., different types of acids (cf. Figure 1(b)). As such, finding the path, which is connected by different subcomponents, can be used to detect some particular or even new biological compounds, e.g., Heparin, which can identify diseases like heart attacks [5].

Therefore, in this paper, we ask a question: how to capture this "higher-order path" between two nodes? To answer this question, we adopt motif, a small subgraph usually with only a few nodes [6]–[9] (e.g. Table I) to represent the structures (cf. Figure 1) in this path, termed *motif-path*. Even though motif has been extensively studied in existing works [2], [10]–[17], as far as we know, this is the first work to incorporate both the concepts of motif and path to find the relationship between two nodes. Due to the extensive use of the shortest path in existing literatures and software (e.g., Neo4j [18]) for path-based graph analysis, we mainly focus on finding the shortest motif-path in this paper.

We show that the shortest motif-paths can be easily adopted to some path-based graph analytical tasks, e.g., link prediction [1] and local graph clustering [2]. Moreover, the effectiveness of these tasks can be significantly improved with the use of

TABLE I: All k-node motifs ranking by number of edges, $k = 2, 3, 4, 5$. For each motif, nodes of same color (black, white, dark gray and light gray) are in the same node-orbit[1].



motif-paths. In particular, we show that finding the shortest motif-path can achieve up to 40% and 25% improvement in both link prediction and local graph clustering tasks respectively, compared with the ordinary path-based approach.

The above "motif-path-based" tasks require the repeated execution of many shortest-path queries. However, retrieving the shortest motif-path is time-consuming. We found that given a motif, finding the shortest motif-path between two nodes takes $O\left(\binom{|V|}{k}^2\right)$ time, where $|V|$ and k denote the graph size and the number of nodes of the motif respectively. This high time complexity prohibits the usage of motif-path in large graphs. To solve this problem, we propose a general framework for finding the shortest motif-path based on a novel index, called *motif-tree*. We further develop an efficient heuristic bidirectional search algorithm to boost the efficiency performance. Our proposed methods can efficiently find the shortest motif-path in both real and synthetic datasets, which can achieve at least three-order-of-magnitude faster than the baseline method.

In this paper, we first review the related work in Section II and introduce the preliminaries in Section III. Then, we define the concepts of motif-path and the shortest motif-path search problem in Section IV. Next, we develop the shortest motif-path searching framework in Section V. We evaluate the performance of methods in Section VI. Section VII concludes.

II. Related Work

Motif-path searching problem is the generalization of path searching problem between source and target nodes, which involves three concepts, (1) motif search (cf. Section II-A) (2) motif-connectivity (cf. Section II-B) and (3) path-based graph analysis (cf. Section II-C).

A. Motif search

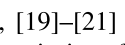
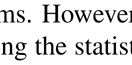
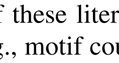
Motifs (e.g., Table I) are regarded as small building blocks [6]–[9] of large and complex graphs [6], [8]. Some popular motifs include k-path (), k-star ( ), k-cycle () and k-clique (), where $k = 3, 4, 5$. Due to its usefulness for a wide range of applications, many literatures [6]–[9], [19]–[21] study the motif-discovery problems. However, the majority of these literatures focus on obtaining the statistics of motifs, e.g., motif counting problem, which is further used for tasks like graph alignment [22] and clustering [2]. In addition, most of them only study small motifs (e.g., motif with 3 or 4 nodes).

TABLE II: Summary of existing motif-connectivities

Existing motif-connectivity	Motif	Connectivity
k-truss [15]	triangle	1 edge
k-gonal chain [27]	triangle	1 node
node k-cycle connectivity [14]	k-paths	1 node
edge k-cycle connectivity [14]	k-paths	1 edge
k-clique connectivity [16]	k-cliques	$k-1$ nodes
α-k-clique connectivity [16]	k-cliques	α nodes
motif random walk [24]	d-node motifs	$d-1$ nodes

Since motifs can capture the high-order structure among the nodes, many graph mining tasks start employing motifs to improve the effectiveness, such as node classification [17], graph clustering [2], [13], [23], node ranking [12] and graph embedding [10], [11].

B. Motif-connectivity

Motif-connectivity function, which is generalized from edge-connectivity, has been utilized in existing work. Huang et al. [15] develop the k-truss community model, which is based on the triangle connectivity function. Cui et al. [16] propose k-cliques connectivity to detect overlapping communities. Batagelj et al. [14] further propose edge/node k-cycle connectivity to decompose graphs. In addition, many motif-connectivity functions, e.g., d-node motifs connectivity, are adopted in the literatures of the graph analysis using motif-based random walk [24]–[26].

However, most of these works can only support a few motifs or motif-connectivity functions, and they cannot find a path of motifs. In this paper, we propose the generalized framework to support a wide range of motif-connectivity functions, including those functions in Table II.

C. Path-based Graph Analysis

Path-based methods have been extensively used for graph analysis in existing work. In link prediction task, missing link can be predicted by analyzing the length and number of the paths that link the query nodes [1], [28]. In graph clustering task, nodes can be clustered based on the shortest path distance [29]. In this work, we show that motif-path can capture high-order structure between two nodes, compared with traditional path-based methods [1], [28], [29], which can achieve better effectiveness in these two graph mining tasks.

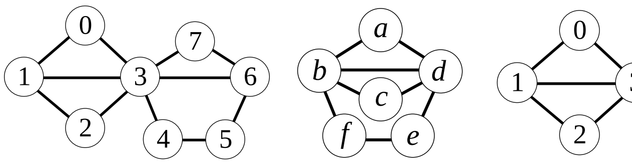
III. Preliminaries

In Section III-A, we first review motif-instance, which is the basic element of motif-path. Then in Sections III-B we introduce motif-connectivity, which controls the compactness of motif-path.

A. Motif-instance

To find a motif-path, we first need to match the subgraphs with the given motifs. In this paper we focus on node-induced subgraph, which is a popular setup in motif discovery area [9]. Figure 2c shows an example of the node-induced subgraph of graph G_1 in Figure 2a. Observe that G_3 must contain all the edges $\{(0,1),(1,2),(2,3),(0,3),(1,3)\}$ in the original graph G_1.

Definition 1: **(Node-induced subgraph)** Given a graph $G = (V, E)$ and a set of nodes $V_m \subseteq V$, the node-induced subgraph $m = (V_m, E_m)$ is the subgraph of G that $(u,v) \in E_m \iff (u,v) \in E$, denoted as $m \in G$.



(a) G_1 (b) G_2 (c) G_3

Fig. 2: Examples of two graphs G_1 and G_2, which are two possible ways to link two motifs and , and the corresponding node-induced subgraph G_3 for nodes $\{0,1,2,3\}$.

Definition 2: **(Graph isomorphism)** Graph $G_1 = (V_1, E_1)$ is isomorphic to G_2, denoted as $G_1 \simeq G_2$, if there is an injective mapping $f : V_1 \to V_2$ such that $(u,v) \in E_1$ if and only if $(f_u, f_v) \in E_2$.

Definition 3: **(Motif-instance)** [9] Given a graph $G = (V, E)$ and a motif $\nu = (V_\nu, E_\nu)$, the motif-instance $m = (V_m, E_m)$ of ν is a subgraph of G, such that:

$$m \in G \quad \text{and} \quad m \simeq \nu. \tag{1}$$

For example, $G_3 \simeq m_{\{a,b,c,d\}}$ in Figure 2 with mapping $\{0,1,2,3\} \to \{a,b,c,d\}$. Also, $G_3 \in G$. Thus G_3 is a motif-instance of motif .

B. Motif-connectivity

To discover a meaningful motif-path, we need to define how two motif-instances are connected. Also, we want to propose a generic tool to unify all the motif-connectivities mentioned in Table II. For example, Figure 2a and 2b are two cases, which are connected by two motifs , based on different motif-connectivity functions.

Definition 4: **(Motif-connectivity)** Given the connectivity threshold δ and two motif-instances $m_1 \simeq \nu_1$ and $m_2 \simeq \nu_2$, we define $c_\delta(m_1, m_2)$ as the motif-connectivity function to indicate whether m_1 and m_2 are connected.

Some representative motif-connectivity functions include:

1) **Node-based motif-connectivity [14], [16]:**

$$c_\delta(m_1, m_2) = \begin{cases} 1 & \text{if } |V_{m_1} \cap V_{m_2}| \geq \delta; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

2) **Edge-based motif-connectivity [15]:**

$$\bar{c}_\delta(m_1, m_2) = \begin{cases} 1 & \text{if } |E_{m_1} \cap E_{m_2}| \geq \delta; \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

For example, G_1 and G_2 in Figures 2 are formed by connecting two motif-instances with c_1 and $\bar{c}_3$ respectively. Note that this motif-connectivity function is controlled by the connectivity threshold δ. Larger δ makes two motif-instances connected more compactly.

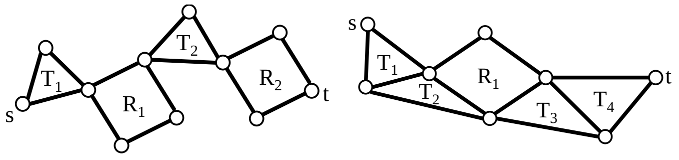
IV. Motif-path

In this paper, we model motif-path as a sequence of motif-instances from source node s to target node t, which are isomorphic to any motifs in the given motif-set $\mathcal{M}$, while the neighboring motif-instances must satisfy the motif-connectivity function.

Definition 5: **(Motif-path)** Given a graph $G = (V, E)$, a source node s, a target node t and parameters $(\mathcal{M}, \delta)$, the motif-path $\mathcal{P}$ is a sequence of motif-instances m_i where:

1) $\exists\, \nu \in \mathcal{M}$, $m_i \simeq \nu$, $i = 1, 2, ..., |\mathcal{P}|$;
2) Two endpoints $s \in V_{m_1}$ and $t \in V_{m_{|\mathcal{P}|}}$.
3) $c/\bar{c}_\delta(m_i, m_{i+1}) = 1$, $i = 1, 2, ..., |\mathcal{P}| - 1$.

Note that the user can choose either c or $\bar{c}$ for a node-based or edge-based version. Figure 3 shows how to find the valid motif-path between the nodes s and t, using the motif-set $M = \{ , \}$ and $\delta = 1$, which means two consecutive motif-instances only share one node and one edge in Figure 3a and 3b respectively.



(a) (b)

Fig. 3: Examples of valid motif-path between the source node s and target node t with (a) node-based motif-connectivity (b) edge-based motif-connectivity with $\mathcal{M} = \{ , \}, \delta = 1$.

Due to the extensive use of the shortest path in existing literatures and software (e.g., Neo4j [18]), we focus on finding the shortest motif-path in this paper, i.e., the motif-path with minimum number of motif-instances, denoted as $\mathcal{P}^*_{s,t}$. We use notation $P^*_{s,t}$ as the ordinary shortest path distance from s to t. Note that there are other interesting variants of motif-path, where we introduce them as future work in the last section.

For example, given $\mathcal{M} = \{ , \}$ and $\delta = 1$ in Figure 4a, the node-based shortest motif-path is $\mathcal{P}^*_{s,t} = \{T_1 \to R_1 \to R_2\}$ with $|\mathcal{P}^*_{s,t}| = 3$.

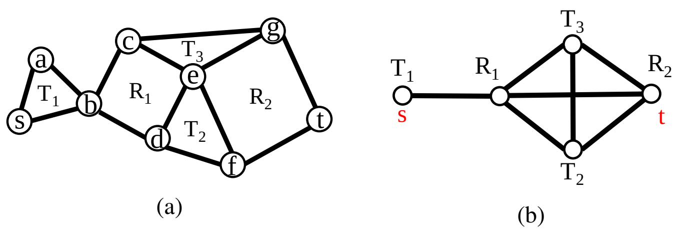


Fig. 4: Example of (a) a graph G and (b) its motif-graph, where each node represents a motif-instance in G.

A. *Baseline method*

Even though there is no existing literature for studying how to find the shortest motif-path between two given nodes s and t, we can simply convert the original graph into a higher-order view, called motif-graph, in which each node represents a motif-instance and two nodes in this motif-graph are connected once the corresponding motif-instances m_1 and m_2 satisfy the motif-connectivity function.

Figure 4b shows the motif-graph for Figure 4a, using $\mathcal{M} = \{△, ◇\}$ and node-based connectivity function with $\delta = 1$. Observe that we can still find the correct shortest motif-path $T_1 \rightarrow R_1 \rightarrow R_2$ by directly using the traditional shortest path searching algorithm in this motif-graph. Note that there may be several motif-instances that contain node s or t, thus it is actually a set-to-set shortest path search problem, with source set $S = \{i | s \in m_i\}$ and target set $T = \{j | t \in m_j\}$.

We regard this baseline method as a two-phase algorithm: (a) Discover all motif-instances in the graph G based on $\mathcal{M}$ and find the linkages between each possible pair of motif-instances based on $c/\bar{c}_\delta$; (b) Run set-to-set shortest path search on the motif-graph generated.

However, there are three major drawbacks of this approach. First, it is time-consuming to build the motif-graph. Actually, searching for all motif-instances in G for a given motif is already a time-consuming process [6], [9], [24], let alone to enumerate the linkages between all motif-instances. Second, the users may need to discover different kinds of motif-paths by tuning $(\mathcal{M}, \delta)$. Then this baseline method has to generate different motif-graphs for different queries. In our experiments, memory overflow occurs even for graphs of moderate size with this baseline approach. Third, the set-to-set shortest path search is also time consuming, as the size of the motif-graph can be much larger than the original graph.

Here, we analyze the time complexity of the shortest motif-path searching problem. In step (a), combinational number of k-node subgraph candidates needs to be enumerated and checked for isomorphism, which takes $O(\sum_{v_i \in \mathcal{M}} \binom{|V|}{k_i} k_i^2)$ operations, where $k_i = |\nu_i|, \nu_i \in \mathcal{M}$. Then, we need to check whether each pair of motif-instances is connected based on the motif-connectivity function, which takes $O\left(a\binom{\sum_{v_i \in \mathcal{M}} \binom{|V|}{k_i}}{2}\right)$ operations. Here, we let a be the cost to search whether two given motif-instances share δ nodes/edges. As a remark, the space complexity of this generated motif-graph is $O\left(\binom{\sum_{v_i \in \mathcal{M}} \binom{|V|}{k_i}}{2}\right)$ in the worst case. In step (b), it takes another $O((\sum_{v_i \in \mathcal{M}} \binom{|V|}{k_i})^2)$ operations to find a shortest path in the generated motif-graph, by using breadth-first search method [30]. Since the motif is usually small, e.g., $k = 3, 4, 5$, the total time complexity is $O(\binom{|V|}{\hat{k}}^2)$, where $\hat{k} = \max_{\nu \in \mathcal{M}} |\nu|$.

V. SMP Incremental Search

Due to the complexity of constructing the motif-graph, we search shortest motif-path incrementally. Observe that this method does not need to explore the full motif-graph.

We show the overview of shortest motif-path search framework in Figure 5. From the source node s, we iteratively expand from a selected seed. To identify motif-instances around the seed, we extract the patterns of motifs in $\mathcal{M}$ called motif-template, and organize them into an index called motif-tree (Section V-A and V-B). We stop expanding if the target node is discovered by the motif-instances. Otherwise, we select another seed and continue expanding. To make a smart choice on the next seed and reduce the searching area, we develop a heuristic bidirectional algorithm (Section V-C).

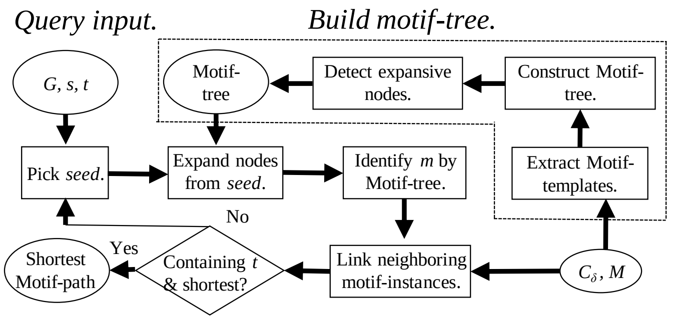


Fig. 5: Overview of shortest motif-path search.

A. *Search motif-instances around s*

Given a source node s as well as a motif set $\mathcal{M}$, we first explore how to search for the motif-instances around s efficiently. However, it is challenging due to the combinational complexity. For example, the number of 5-node motif-instances around the single node can be four to six magnitudes bigger than the size of original graph [31]. Even though there are many possible combinations, many of them are in fact redundant. In the following, we illustrate how to efficiently search motif-instances around the single node.

1) Identify non-redundant motif-templates: Note that each node in the motif has the possibility to become the source node s, and thus there are different searching patterns according to the different positions of s in the motif. To model this process, we introduce node-orbit [32], which can be used here to avoid finding redundant motif-instances.

Definition 6: **(Node-orbit)** [32] Given a motif ν, the nodes $a \in V_\nu$ and $b \in V_\nu$ are within the same node-orbit if there is an injective mapping $f : V_\nu \rightarrow V_\nu$ with $f_a = b$ and $f_b = a$, such that $(u, v) \in E_\nu$ if and only if $(f_u, f_v) \in E_\nu$.

As an example, we treat G_3 in Figure 2c as the motif. Observe that once we impose the constraint $f_0 = 2$ and $f_2 = 0$, we can find the injective mapping which fulfills Definition 6. Therefore, nodes 0 and 2 are in the same orbit. Similarly, nodes 1 and 3 are also in the same orbit. However, nodes 0 and 1 are not in the same orbit, since there is no injective mapping which fulfills Definition 6. Table I shows different colors of nodes, which represent different node-orbits, for each motif.

Here we denote s as the first *seed* and regard each motif with seed node as a *motif-template*, denoted as $\bar{\nu}$. For those seed nodes in the same node-orbit (cf. Definition 6), the corresponding motif-templates are redundant to each other. For example, the seed nodes of Figure 6c and Figure 6e are redundant since the two seeds (black nodes) are in the same node-orbit. Therefore, it is enough to use either one of them to search for the motif-instances around s.

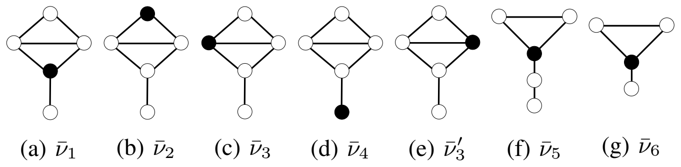


Fig. 6: (a-d) All motif-templates of , with seed as black node. (e) Redundant motif-template of $\bar{\nu}_3$. (f) A motif-templates of . (g) Common sub-structures shared by $\bar{\nu}_1$ and $\bar{\nu}_5$.

2) Build motif-tree on top of motif-templates: After we obtain all those non-redundant motif-templates, one basic method is to search for all motif-instances around the seed, i.e., we need to check if each possible candidate of motif-instance is isomorphic to any one of the motif-template from $\mathcal{M}$. However, checking all the motif-templates one-by-one can be time-consuming, especially for the large size of motif-set $\mathcal{M}$, which can generate large number of motif-templates.

Observe from Figure 6, once we search motif-instances for $\mathcal{M} = \{ , \}$, two examples of motif-templates $\bar{\nu}_1$ and $\bar{\nu}_5$ (Figure 6a and 6f) are generated, and they share the same sub-structure $\bar{\nu}_6$ (Figure 6g). As such, we can search motif-instances m that is isomorphic to $\bar{\nu}_6$ first, and then expand m to find the motif-instances that is isomorphic to $\bar{\nu}_1$ and $\bar{\nu}_5$. In this manner, we can avoid repeatly searching same common sub-structure, which is shared by different motif-templates, such that it can reduce searching time. Based on these shared motif-substructures of different motif-templates, we can construct an index called *motif-tree*.

Motif-tree is constructed in an top-down manner. From the seed node (node s at the current stage), we check the sub-structures of each motif-template layer by layer. The common substructures are combined into the same branch in the motif-tree. For example, for motif-template $\bar{\nu}_{21,2}$ and $\bar{\nu}_{14,3}$ in Figure 7, we find that they share the same one-layer substructures, that is, $\bar{\nu}_{5,1}$, but second-hop substructures ($\bar{\nu}_{16,1}$ and $\bar{\nu}_{9,2}$) are different, so we only merge the first level in motif-tree, and generate two branches in the following levels. Here the i-th layer is the set of nodes in motif-template whose shortest path distance to seed are i.

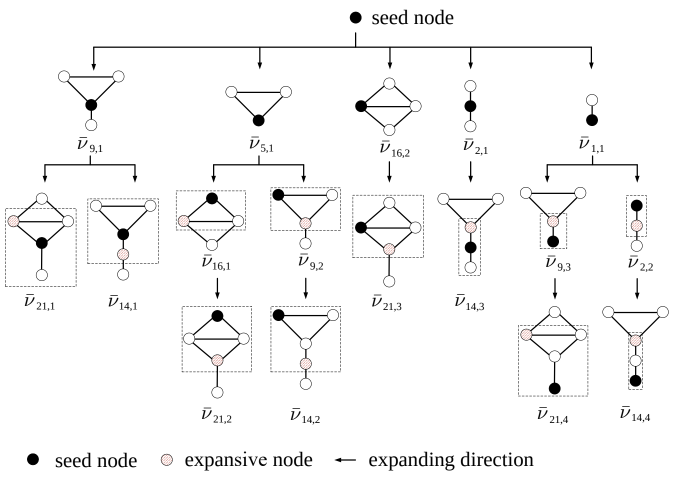


Fig. 7: Example of motif-tree using $\mathcal{M} = \{ , \}$ around a single node. Here $\bar{\nu}_{i,j}$ denotes j-th motif-template of the i-th motif in Table I.

Figure 7 shows the motif-tree for $\mathcal{M} = \{ , \}$, where the leaf nodes are all generated motif-templates of these two motifs. The general idea of searching motif-instances around the seed is to explore each branch in the motif-tree and filter the branch if there is no motif-instance that is isomorphic to the shared sub-structure.

Using Figure 8a as an example, we aim to search for the motif-instances around node s in Figure 4a with $\mathcal{M} = \{ , \}$. Observe that the common sub-structures $\bar{\nu}_{9,1}$, $\bar{\nu}_{16,2}$ and $\bar{\nu}_{2,1}$ in Figure 7 do not match any motif-instances around s. As such, we can safely filter these three branches, which means that we do not need to search the child nodes of $\bar{\nu}_{9,1}$, $\bar{\nu}_{16,2}$ or $\bar{\nu}_{2,1}$. Compared with the basic method, which searches for the motif-instances using all motif-templates, motif-tree can help reduce the redundant search, thus speeding up the searching process.

3) Efficient motif-tree traversal: During the motif-tree traversal, we need to maintain the mapping from the motif-templates to current possible candidates of motif-instances. For example, one possible mapping from $\bar{\nu}_{14,2}$ of Figure 8a to graph G in Figure 4a is $(\alpha \to s, \beta \to a, \gamma \to b, \delta \to c, \epsilon \to e)$. Below we introduce how to find the mappings by an node expanding manner.

In each iteration of motif-tree traversal, it expands one hop from the motif-template of the parent node $\bar{\nu}_p$ to the child node $\bar{\nu}_c$. In this process, we search for the neighbors of nodes in $\bar{m}_p$ where $\bar{m}_p \simeq \bar{\nu}_p$, and check if the new discovered subgraph is isomorphic to $\bar{\nu}_c$. If so, a new subgraph $\bar{m}_c \simeq \bar{\nu}_c$ is discovered.

Given $\bar{m}_p \simeq \bar{\nu}_p$, we follow a two-phase algorithm to discover the new motif-instances $\bar{m}_c$ such that $\bar{m}_c \simeq \bar{\nu}_c$, by expanding $\bar{m}_p$ smartly.

Phase A. First, we discover possible candidates of $\bar{m}_c$ via node expansion from $\bar{m}_p$. Given $\bar{m}_p \simeq \bar{\nu}_p$, we expand $\bar{m}_p$

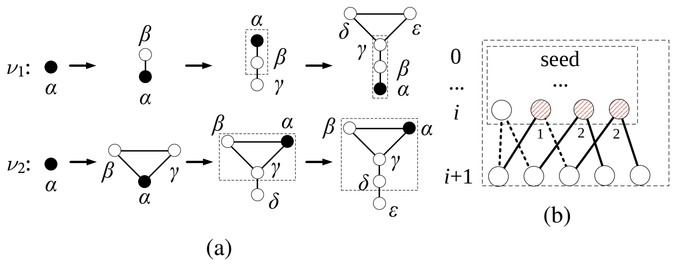


Fig. 8: Expanding strategies. (a) Traversal of two branches of motif-tree to find motif-template $\bar{\nu}_{14,4}$ (top) and $\bar{\nu}_{14_2}$ (bottom). (b) Detect expansive nodes and their expansive degree by edge-cover from i-th hop to $(i+1)$-th hop from the seed.

by searching the neighbors of nodes in $\bar{m}_p$. To avoid finding duplicates of $\bar{m}_c$, we carefully select the nodes in $\bar{m}_p$ that we search for neighbors, called *expansive nodes*. We show in Figure 8 about the expanding strategy.

We first rank nodes in $\bar{m}_p$ according to its distance to the seed. Then we only need to search for the neighbors of the nodes at the bottom of $\bar{m}_p$ (i-th hop in Figure 8(b)). In other words, the expansive nodes must be the furthest nodes from the seed in $\bar{m}_p$. Then, we use minimum number of edges from these nodes to cover the nodes in m_c to be discovered. For example, we can discover all the new nodes in $\bar{m}_c$ by the solid lines in Figure 8(b). The end-points in $\bar{m}_p$ of these solid lines are the expansive nodes. They are marked as dashed pink nodes in the paper. In this manner, this expanding strategy does not discover motif-instance repeatedly and thus, we do not need to check whether the discovered motif-instances are duplicated or not, and thus time is saved.

Note that expansive nodes may have different number of nodes to expand, denoted as *expansive degree*. For example, the expansive degree of the expansive nodes in Figure 8(b) are (1, 2, 2) respectively.

Phase B. Then, we check subgraph isomorphism on subgraph expanded from the expansive nodes in phase A. Note that we check the linkages for the new discovered nodes and maintain the mappings which satisfy the graph isomorphism to $\bar{\nu}_c$. Then for next round expanding in this branch of motif-tree, we expand all possible mappings maintained currently. Note that after expansion from the expansive nodes, each mapping should discover at least one new node, which is not discovered yet by any motif-instance.

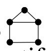
For example, we show the expanding steps of Figure 8a in Table III, from node s in Figure 4a as the seed. The expansive nodes are in bold. To save space, we denote motif-tree branch as MTB and graph mappings as GM. As shown in the table, we only need to enumerate eight entries following the two branches of motif-tree in Figure 8, and two entries terminate in the early stage. Also, several common sub-structures can be shared between several entries and thus time is saved.

TABLE III: Expanding examples of $\bar{\nu}_1$ and $\bar{\nu}_2$ in Figure 8a, with s in Figure 4a as the seed.

$\bar{\nu}_{14,4}$	MTB	$\boldsymbol{\alpha}$	$\rightarrow$	$\alpha\boldsymbol{\beta}$	$\rightarrow$	$\alpha\beta\boldsymbol{\gamma}$	$\rightarrow$	$\alpha\beta\gamma\delta\epsilon$
	GM	$\boldsymbol{s}$	$\rightarrow$	$s\boldsymbol{a}$		terminate.		
			$\rightarrow$	$s\boldsymbol{b}$	$\rightarrow$	$sb\boldsymbol{c}$	$\rightarrow$	$sbceg$
					$\rightarrow$	$sb\boldsymbol{d}$	$\rightarrow$	$sbdef$
$\bar{\nu}_{14,2}$	MTB	$\boldsymbol{\alpha}$	$\rightarrow$	$\alpha\beta\boldsymbol{\gamma}$	$\rightarrow$	$\alpha\beta\gamma\boldsymbol{\delta}$	$\rightarrow$	$\alpha\beta\gamma\delta\epsilon$
	GM	$\boldsymbol{s}$	$\rightarrow$	$sa\boldsymbol{b}$	$\rightarrow$	$sab\boldsymbol{c}$	$\rightarrow$	$sabce$
							$\rightarrow$	$sabcg$
					$\rightarrow$	$sab\boldsymbol{d}$	$\rightarrow$	$sabde$
							$\rightarrow$	$sabdf$
			$\rightarrow$	$sb\boldsymbol{a}$		terminate.		

B. Search motif-instances around the seed

In previous section, we only focus on searching the motif-instances around a single node s. However, as stated in Section III-B and Table II, there are different motif-connectivity functions in which searching motif-instances around the single node (cf. Section V-A) is not enough to handle these cases.

For example, given a motif-instance $m_1 = \{s, a, b, d, f\}$, containing the source node s in Figure 9, we aim at finding a motif-path with $\mathcal{M} = \{$,  $\}$. Figure 9a and 9b shows two cases where edge-based motif-connectivity and node-based motif-connectivity are used respectively, both with $\delta = 1$. When using the node-based motif-connectivity function, as shown in Figure 9b, the seed to be selected after node s is node f, since the motif-instance containing f finally reaches node t. Therefore, the algorithms in Section V-A can be reused.

However, we cannot directly apply the same process in Figure 9a, if an edge-based connectivity function is used. A simple adaption is to select a node from the edge as the new seed and find motif-instances which must contain another node of the edge. However, this method (1) finds more possible candidates of motif-instances and (2) needs to check for the correct results, which can incur higher computational cost. To make the motif-instance searching process more efficient for this case, we need to consider finding motif-instances around a more complex seed, instead of a single node.

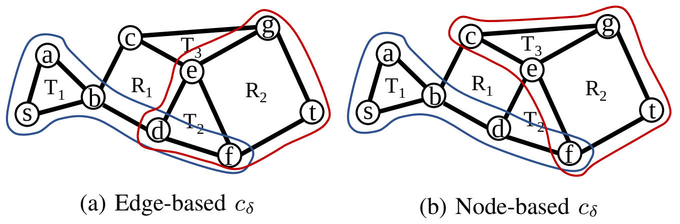


Fig. 9: Different seed from the motif-connectivity function with $\delta = 1$. (a) Seed is edge (d, f). (b) Seed is node f.

In this section, we consider arbitrary number of nodes or edges as the seed, and build the motif-tree for different types of seeds. First, we identify the non-redundant motif-templates (cf. Section V-A1) for a complex seed by generalizing Definition 6 into node/edge-set-orbit.

Definition 7: (Node/Edge-set-orbit) Given a motif ν, two sets of nodes (edges) $S \subset V_\nu$ ($S \subset E_\nu$) and $T \subset V_\nu$ ($T \subset E_\nu$) are within the same node-set-orbit (edge-set-orbit) if there is an injective mapping $f : V_\nu \to V_\nu$ with $\forall a \in S, f_a = b$ and $f_b = a, b \in T$, and $\forall b \in T, f_b = a$ and $f_a = b, a \in S$, such that $(u, v) \in E_\nu$ if and only if $(f_u, f_v) \in E_\nu$.

Then we only select one motif-template from each orbit. For example, Figure 10 shows all motif-templates for single edge as the seed and two-node-set as the seed. Once we obtain the non-redundant motif-templates, we can build the motif-tree using the same manner by checking the shared sub-structure (cf. Section V-A2).

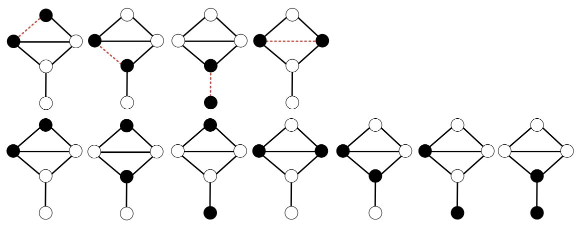


Fig. 10: Motif templates considering seeds (in black) of 1 edge (top) and 2 nodes (bottom) for motif .

C. Motif-path incremental search

After we discover all possible motif-instances around the seed, we can construct a motif-path $\mathcal{P}$ based on these motif-instances. Using Figure 4a as an example, for $\mathcal{M} = \{ , \}$ and $\delta = 1$ with node-based motif-connectivity, we first discover motif-instance T_1 containing seed s. After that, we use another node (a or b) as next seed and discover other motif-instances. The process is repeated until target node t is discovered.

However, there are two drawbacks with this method. First, it is possible to detect motif-paths containing redundant motif-instances. Second, the incremental searching direction for the motif-path may be far from the optimal direction, which leads to the shortest motif-path distance.

Therefore, it will be meaningful to filter some shortest motif-path candidates at the very first stage. In this section, we address these two issues in Sections V-C1 and V-C2. To make it more clear, we focus on node-based motif-connectivity, i.e., the seed is a single node. However, these techniques can be easily extended to multiple edge-based motif-connectivity by changing node-based seed selection into edge based seed selection. We also evaluate motif-paths based on different motif-connectivity functions in Section VI.

1) Motif-path filtering by node status: During the motif-path searching process, we mark each node in the graph into one of the three status: *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 covered by any motif-instance at the current time. For other nodes, we call it "undiscovered". We develop following incremental searching strategies.

c1. motif-instance containing any node with status "searched" 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 candidates.

Proof: For c1, there is no need to add motif-instances containing a "searched" node since all motif-instances around node marked as "searched" have been found and added into candidates for $\mathcal{P}^*_{s,t}$. For c2, for the motif-instances on candidates for $\mathcal{P}^*_{s,t}$, there are only two status of the nodes: "searched" and "discovered". We only select "discovered" node as next seed because the "searched" nodes have been used as seed before and thus using them as next seed will find duplicates on the candidates. For c3, in the incremental search manner, $\mathcal{P}^*_{s,v}$ is found for the "undiscovered" node v when v is covered by any motif-instance for the first time. Therefore, we only add motif-instances which contain at least one node marked as "undiscovered" to push the incremental search forward. ■

2) Heuristic bi-directional search: Besides the incremental search manner described in Section V-C1, the searching direction may be far from optimal. In other words, it is essential to develop a method to select better seeds in the next step which may lead the motif-path to reach target node t earlier.

To reduce the search space, we generalize the bidirectional heuristic search [33] 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 has the biggest probability to be be shortest motif-path.

We achieve the shortest motif-path estimation by defining heuristic function as below. 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 shortest motif-path via node p, denoted as $\mathcal{P}^p_{s,t}$ can be estimated.

$$|\mathcal{P}^p_{s,t}| = |\mathcal{P}^*_{s,p}| + h(p,t), h(p,t) \le |\mathcal{P}^*_{p,t}| \tag{4}$$

In the following, we propose the heuristic function $h(p,t)$, which fulfills the requirements above. First we launch a single-sourced shortest path search from node t and mark each node p with its shortest path distance to target node t, that is, $|P^*_{t,p}|$. Then every time a new node p is discovered, we check for $P^*_{t,p}$ and calculate $h(p,t) = \frac{|P^*_{t,p}|}{\max_{\nu \in \mathcal{M}} \Theta(\nu)}$. Here $\Theta(\nu)$ is the diameter of ν. This heuristic function actually uses the motif with largest diameter to cover the shortest path distance, thus providing a lower-bound to the shortest motif-path.

Then we launch the same process introduced above from both node s (forward search) and t (reverse search) for bi-directional search. The algorithm terminates when the two searching processes meet, that is,

$$\exists p \in G, |\mathcal{P}^*_{s,t}| = |\mathcal{P}^*_{s,p}| + |\mathcal{P}^*_{t,p}|. \tag{5}$$

From the property of unweighted bi-directional search, node p is the first meeting point of forward search and reverse search.

VI. Evaluations

In our experiments, we evaluate the efficiency and effectiveness of searching motif-paths on real-world and synthetic graphs. In Section VI-A, we test the efficiency of our proposed motif-path searching methods. Then, we adapt two path-based graph mining tasks, which are missing link prediction (cf. Section VI-B) and local graph clustering (cf. Section VI-C) into motif-path-based versions. Compared to the path-based results, the effectiveness is significantly improved by the motif-path-based approach.

Two kinds of real-world datasets are used: protein-protein interaction (PPI) networks and social networks. We show the details of these datasets in Table IV.

TABLE IV: Statistics of datasets.

Name	$\|V\|$	$\|E\|$	Degree	Diameter
GAVI	1,855	7,669	8.3	13
KCOR	2,708	7,123	5.3	11
EXTE	3,672	14,317	7.8	10
DBLP	317,080	1,049,866	6.6	23
AMAZ	334,863	925,872	5.5	47
YOUT	1,134,890	2,987,624	5.3	24
SYNT	200∼2M	400∼4M	4	10∼20

PPI networks. We use five PPI networks in which nodes denote proteins and edges denote the interactions between proteins. GAVI [34] is the PPI network of yeast cell. KCOR [35] is the core PPI network of another bacterias, and EXTE [35] is the extended interaction dataset of KCOR, which contains less reliable interactions but its coverage is higher.

Social networks. We use three social networks with ground-truth communities from [36]. DBLP is a co-authorship network from computer science bibliography where two authors are linked if they publish at least one paper together. Similarly, 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 is a friendship network from Youtube, where each node denotes a user and there is an edge if the two users are friends.

Synthetic networks. To test the scalability of the algorithm, we generate several synthetic graphs (SYNT in Table IV), with the number of nodes as 2×10^i, with $i = 2, 3, 4, 5, 6$. We fix the average degree as 4 and employ Barabási-Albert model [37], a widely used method to simulate real graphs.

TABLE V: Motif-path searching algorithms.

Algo.	Techniques used.	Sec.
BASE	Enumerate motif-graph by VF3.	IV-A
IMS	Motif-tree & motif-path filtering.	V-C1
HBS	IMS & Heuristic bi-directional search.	V-C2

A. Efficiency Evaluation

In this paper, we develop three algorithms to search for shortest motif-paths: Motif-graph enumeration with VF3 (BASE), Incremental motif-path search (IMS) and Heuristic bi-directional search (HBS). We summarize the technical details of them into Table V.

BASE searches for the motif-path by enumerating the motif-graph needed. However, as introduced in Section IV-A, this method is expensive and easy to cause memory overflows. So we adapt VF3 [38], the state-of-the-art subgraph isomorphism algorithm, to speed up the enumerating process. For the following evaluations, we randomly sample 100 (s, t)-pairs and measure the average response time.

First, we evaluate the efficiency of the three algorithms by varying the size of $\mathcal{M}$, in order to test the efficiency change when supporting more motif types on a single motif-path. Here, we fix $\delta = 1$ and use the node-based connectivity function. Motifs in Table I are added into $\mathcal{M}$ in the ascending order of their edge numbers. Observe from Figure 11(a), the response time increases when $|\mathcal{M}|$ increases, since larger $\mathcal{M}$ incurs longer time for searching motif-instances (more motif-instances match with the motifs in $\mathcal{M}$). The result shows that both IMS and HBS are 3-5 magnitudes faster than BASE, which cannot terminate when there are thousands of edges in the graph. With motif-tree and the incremental searching scheme, IMS and HBS largely reduce the searching space where baseline suffers, and thus enable the usage of motif-paths in many applications.

Then with the synthetic graphs, we show the scalability of the three methods in Figure 11(b). Observe that both methods can be finished in a proper response time for different graph sizes, which means our method is scalable. In general, the response time increases once the graph size increases. However, since motif-path composed of dense motif-instances, e.g., and are difficult to form long paths, which means the searching is more likely to terminate than other motifs, making the curve only rise a little bit, or even drop as graph size increases.

Finally, we fix $|\mathcal{M}| = 1$ and test how the motif-connectivity functions affects the efficiency performance by evaluating parameter δ. Here we test both node-based motif-connectivity c_δ and edge-based motif-connectivity $\bar{c}_\delta$ and vary δ from 1 to 4. Observe from Figure 12, once we increase δ, the response time normally decreases. The main reason is that the larger the value of δ, the harder the two motif-instances to be concatenated (more nodes or edges need to be shared between two motif-instances). However, bigger δ may generates more motif-instances, thus making the red line increase first and then drop.

B. Motif-path based Link Prediction

In this section, we utilize motif-path to predict the missing link between two nodes, by extending the widely-used path-based methods, Katz Index and Graph Distance, into motif-path-based versions. We also compare the effectiveness with other methods, including motif-based methods and embedding-based methods.

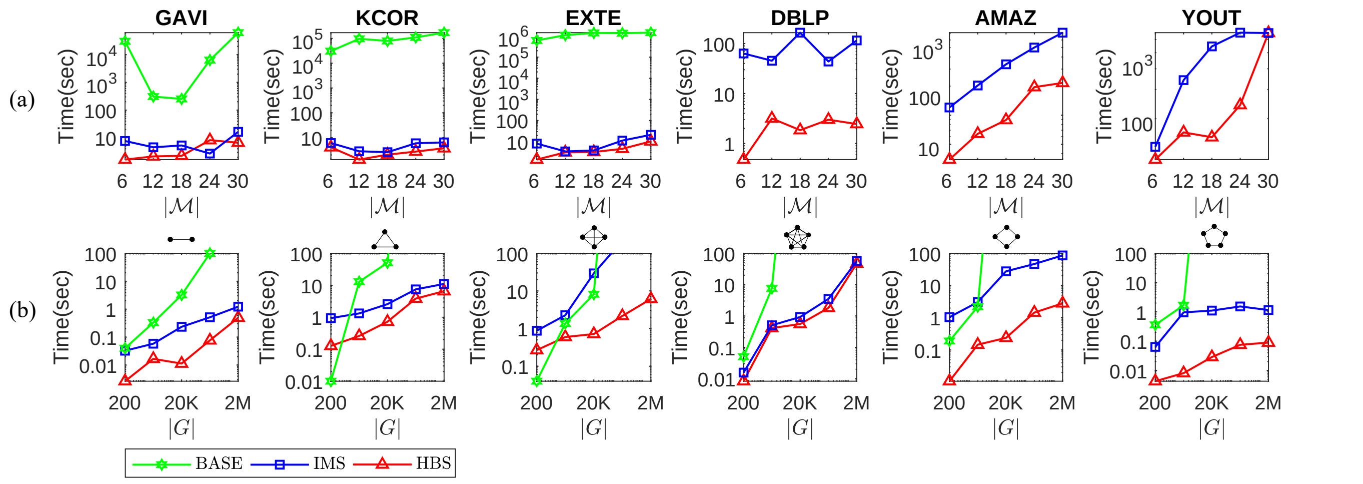


Fig. 11: Response time of searching motif-paths on (a) real-world datasets when varying $|\mathcal{M}|$ (b) synthetic graphs when varying $|G|$.

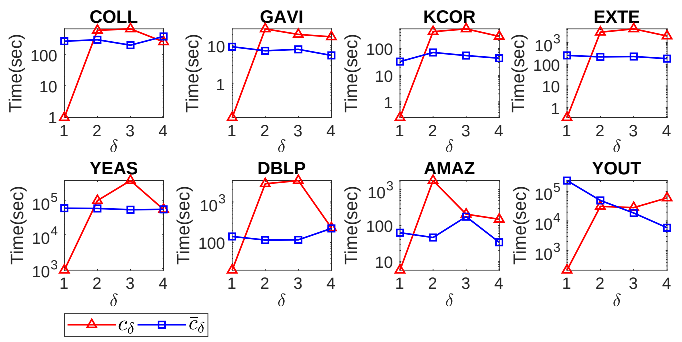


Fig. 12: Response time of searching motif-paths on real-world graphs when varying δ from 1 to 4.

Katz Index (KI) [28] A potential missing link is likely to have more short paths between the end nodes (x, y):

$$g_{KI}(x,y) = \sum_{l=1}^{L} \epsilon^{l-1} \cdot |\mathbb{P}^l_{x,y}|, \tag{6}$$

where $\mathbb{P}^l_{x,y} = \{P^l_{x,y}\}$ is the set of all length-l paths between x and y, and ϵ is the weighting parameter with $\epsilon < 1$.
Graph Distance (GD) [1] A potential missing link is likely to have small shortest path distance between (x, y):

$$g_{GD}(x,y) = \frac{1}{|P^*_{x,y}|}. \tag{7}$$

Obviously, the pair with higher g value has bigger potential to form links. Then we extend KI into *Motif-path-based Katz Index* (MKI), denoted as

$$g_{MKI}(x,y) = \sum_{l=1}^{L} \epsilon^{l-1} \cdot |\mathbb{MP}^l_{x,y}|, \tag{8}$$

where $\mathbb{MP}^l_{x,y} = \{\mathcal{P}^l_{x,y}\}$ is the set of motif-paths between x and y with length l. Following the trend [39], [40], we restrict $L = 4$ and $\epsilon = 0.1$ for both KI and MKI. Also, we extend GD into *Motif-path-based Graph Distance* (MGD), denoted as

$$g_{MGD}(x,y) = \frac{1}{|\mathcal{P}^*_{x,y}|}. \tag{9}$$

Following the setting of [1], 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 $c = 250$.

Then we can utilize the standard metrics *Area Under Curve* (AUC) and *Accuracy* (ACC) [1] to measure the effectiveness:

$$AUC = \frac{2 \cdot c_1 + c_2}{2 \cdot c} \text{ and } ACC = \frac{c_1}{c}. \tag{10}$$

Following [1], we independently sample the positive and negative pairs which come from the same shortest-path-distance distribution. Otherwise, positive pairs will be much nearer to each other than the negative pairs, making any predicting method easy to obtain high effectiveness.

In Figure 13, we first evaluate the diversity of motif-path. As shown in Figure 13(a), AUC/ACC score changes as $|\mathcal{M}|$ increases. Here we use node-based motif-connectivity with $\delta = 1$, and employ k-cliques and k-cycles, that is, $\bar{\mathcal{M}} = \{\ ,\ ,\ ,\ ,\ \}$. With each $|\mathcal{M}|$, we report the highest AUC/ACC that MKI can achieves, denoted as AUC*/ACC*. In other words, $\text{AUC}^*_i/\text{ACC}^*_i = \max_{\mathcal{M}\subseteq\bar{\mathcal{M}}} \text{AUC}_i/\text{ACC}_i, |\mathcal{M}| = i$. Generally, optimal $|\mathcal{M}|$ appears in the middle of the curve. It may come from the fact that two implicit friends/proteins can be linked by a few different motifs, but too many diverse communities/complexes within the path can damage the coherence between the corresponding end nodes.

Note that we omit the effectiveness of GD in Figure 13 since it always appears at the bottom of the figure, with ACC around

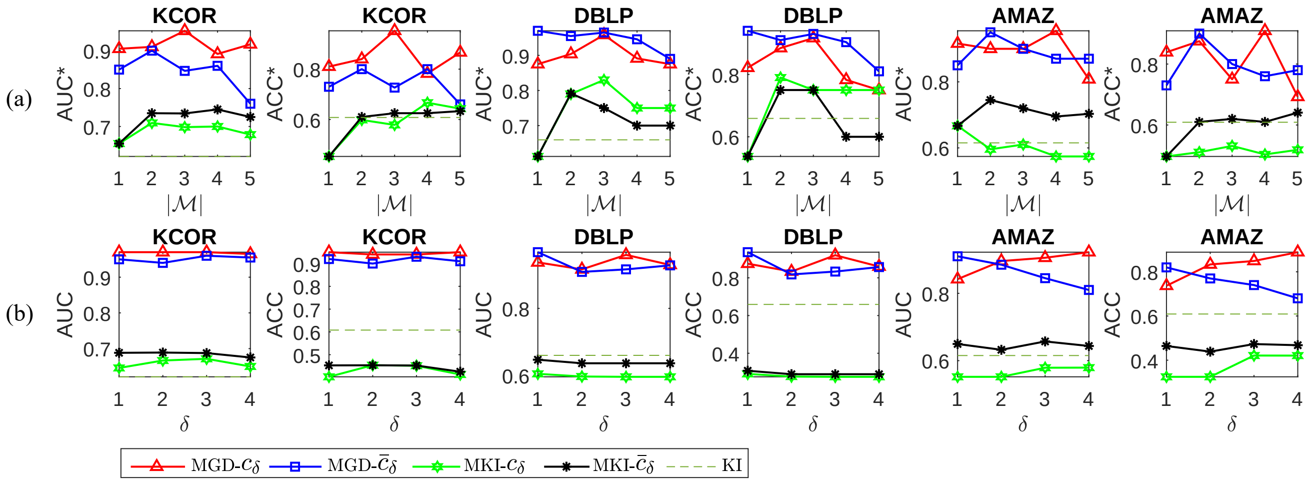
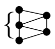


Fig. 13: Effectiveness of MKI and MGD when varying (a) $|\mathcal{M}|$ and (b) δ.

0.1 and AUC around 0.5 (see Table VI). Obviously, MKI and MGD outperforms the ordinary path-based KI and GD in most datasets. Also, MGD obtains higher effectiveness than MKI in most cases, meaning that $|\mathcal{P}^*_{x,y}|$ itself as an indicator is critical in missing link prediction tasks. In general, edge-based MKI/MGD is better than the node-based version, since edge-based motif-path is more compact and thus can find out more robust pairs as missing links.

Then in Figure 13(b), we fix $\mathcal{M} = \{$$\}$ and vary δ for both node-based and edge-based motif-connectivity functions. With a proper δ, motif-path can be of optimal compactness and thus obtain best predicting results. The optimal δ sometimes appears in the middle of the curve, since a compact motif-path is meaningful to reveal cohesive relationship, but too compact motif-path can be difficult to be found, thus losing the ability to predict missing links. An interesting result is that in AMAZ, the trend of node-based MGD (MGD-c_δ) and edge-based MGD (MGD-$\bar{c}_\delta$) are different, which means that with the same δ, it may not be compact enough for a node-based motif-path, but already too compact for edge-based version.

We also compare the effectiveness of MKI with the state-of-the-art works, 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 [41]. Theses methods are easily to be adopted but the AUC score only varies from 0.5 to 0.7 in most cases.

Then we evaluate the complex approaches, with either embedding features or motif-features.

Motif-based Common Neighbor (MCN) [42] is the extended work from CN to predict missing links, with scoring function $g_{MCN} = |\Gamma_m(x) \cap \Gamma_m(y)|$, where $\Gamma_m(x) = \{m | x \in m \text{ and } m \simeq \nu, \nu \in \mathcal{M}\}$ denotes the set of motif-instance which contains the node x. Triangle is used in this work.

TABLE VI: MKI/MGD performance with AUC reported.

Method	GAVI	KCOR	EXTE	DBLP	AMAZ
CN	0.72	0.58	0.56	0.79	0.62
JC	0.70	0.51	0.48	0.55	0.52
AA	0.76	0.58	0.57	0.81	0.65
PA	0.59	0.72	0.76	0.64	0.63
FM	0.65	0.65	0.65	0.59	0.64
HT	0.60	0.62	0.70	0.64	0.59
RPR	0.61	0.52	0.51	0.76	0.62
MCN	0.67	0.63	0.62	0.58	0.61
MLP+GB	**0.95**	**0.91**	**0.83**	**0.82**	0.72
DW+GB	0.65	0.66	0.67	0.81	**0.95**
KI	0.66	0.65	0.64	0.66	0.66
MKI-t	0.72	0.66	0.67	0.66	0.68
MKI-cc	0.78	0.75	0.77	**0.83**	0.75
GD	0.50	0.50	0.50	0.50	0.50
MGD-t	**0.81**	**0.83**	**0.85**	0.73	**0.79**
MGD-cc	**0.93**	**0.92**	**0.94**	**0.95**	**0.88**

Motif-based Link Prediction (MLP) [41] counts the motif-instances around the missing/non-existent link and generate motif distribution as feature vector. All k-motifs are used in this work, $k = 3, 4, 5$. Then classifiers are trained for prediction. We choose Gradient Boosting (GB), which obtains best effectiveness among all classifiers listed in the paper.

Deepwalk (DW) [43] is believed as an graph embedding approach which can extract elaborate feature vectors for the missing/non-existent links. Similarly, we train the Gradient Boosting to obtain missing link prediction results.

As shown in Table VI, our motif-path approach can achieve pretty high effectiveness among all the methods listed. Compared with the ordinary path-based methods (KI and GD), the effectiveness is significantly improved with affordable extra searching time, either with a triangle (MGD-t and MKI-t) or with k-cycles/cliques $\bar{\mathcal{M}}$ (MGD-cc and MKI-cc). Compared with the complex approaches, our methods are also competitive while avoiding the expensive training process.

C. Motif-path based Local Graph Clustering

Recently, many studies [2], [13] demonstrate that using motifs as higher-order structure can improve the effectiveness

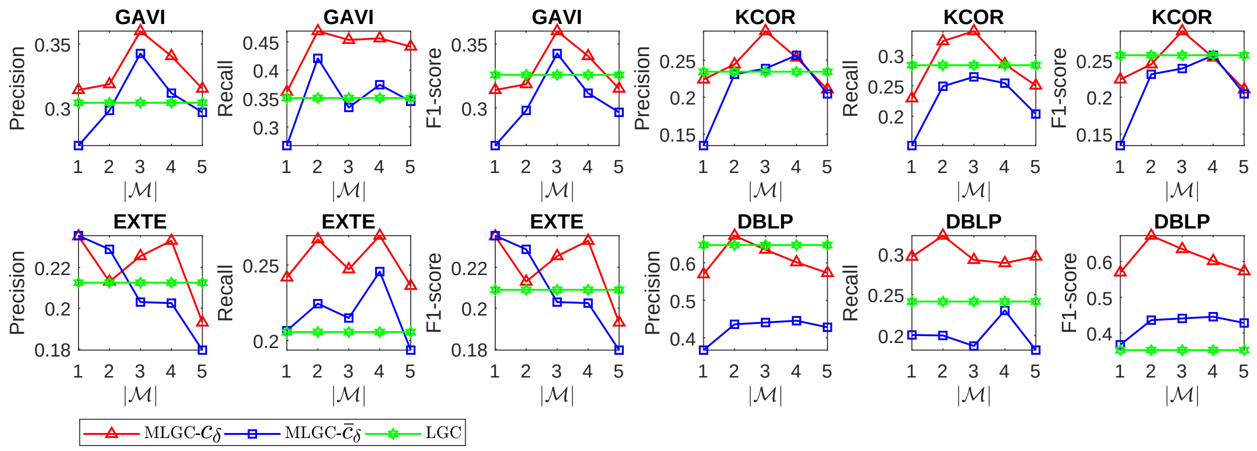
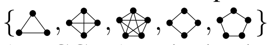


Fig. 14: Local graph clustering results on four real-world datasets with Precision, Recall and F1-score reported.

of graph clustering. In this section, we demonstrate that our motif-path can improve the effectiveness of ordinary path-based local graph clustering. In order to launch the comparison, we extend the widely-used Local Graph Clustering (LGC) into a motif-path-based version, namely MLGC. LGC aims to find the cluster for a given query node by searching for k-nearest neighbors, so we extend LGC by finding k-nearest neighbors with shortest motif-path distance, rather than ordinary shortest path distance.

In this experimental setting, we aim to find the protein-complex/author-community with a given protein/author. Similar like the setup of link prediction, we use $\bar{\mathcal{M}} =$  and test the node-based version (MLGC-c_δ) and edge-based version (MLGC-$\bar{c}_\delta$). Here we fix $k = 50$ and $\delta = 1$.

To evaluate the local graph clustering effectiveness, we use a well-known protein complex dataset, called MIPS [44], as the ground truth for PPI datasets, in which each protein complex can be regarded as a cluster for the nodes in PPI. We also use researcher communities as the ground-truth for DBLP [36].

In Figure 14, we report the precision, recall and F1-score of the effectiveness comparison results. Normally our method (MLGC) 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 among the cluster numbers, compared to the ordinary path (e.g., Figure 1). Also, we notice that node-based motif-connectivity obtains higher effectiveness, especially in DBLP. It may come from the fact that edge-based motif-path limits the searching area thus damages the clustering effectiveness. Similar like the link prediction curves, the optimal $|\mathcal{M}|$ usually appears in the middle, meaning that a proper diversity of motif types on motif-path is critical in developing the clustering effectiveness.

Observations. Within the above two graph mining tasks, motif-path significantly improves effectiveness compared to the ordinary path-based approaches. Also, we notice that a single motif usually cannot obtain the optimal effectiveness, thus the scheme we proposed is meaningful to support efficient search with diverse motifs on a single motif-path.

Given a new graph, it will be a problem of choosing parameters δ and $\mathcal{M}$. Here we propose three observations. First, motif-path with two or three motifs usually has better performance than a single motif, so it would be useful to test several combinations of motifs in a small sampled subgraph, and then use the motif set in the graph mining task. Second, the effectiveness is usually linked with the frequency/significance of the motif employed [24], [42]. We also observe that the performance of motif-path usually has strong correlation to the frequency of motifs in the graph. For example, KCOR and EXTE have bigger proportion of triangles than AMAZ, and thus MKI-t and MGD-t have better effectiveness on these networks than AMAZ in Table IV. Finally, the performance of motif-path is usually influenced by the sparsity of the graph. It worth trying bigger δ on dense graphs but on sparse graphs, it may be more safe to chose a smaller δ with node-based motif-connectivity.

VII. Conclusions

In this paper, we propose the novel concept, called motif-path, which can discover the high-order semantics between two nodes in graph and further propose the shortest motif-path problem. Our accuracy experiements show that, by combining the shortest motif-path, with different path-based graph mining tasks, we can significantly boost the accuracy performance, e.g., 40% and 25% improvement for link prediction and local graph clustering respectively.

On the other hand, due to the high time and space complexity for finding the shortest motif-path, we further develop a general incremental search framework with the bidirectional search algorithm and a novel tree-index, motif-tree, to significantly boost the efficiency performance. Our efficient experiments show that our proposed methods can significantly outperform the baseline method by at least three-order-of-magnitude.

In the future, we plan to investigate how the concept of motif-path can be adopted on heterogeneous information networks. In addition, except for only finding the shortest motif-path, we will also explore how other types of motif-path problems (e.g., top-k shortest motif-paths) can be used in different applications.

REFERENCES

[1] L. L and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150 – 1170, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037843711000991X

[2] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 555–564.

[3] G. Liu, Y. Wang, and M. A. Orgun, "Optimal social trust path selection in complex social networks," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1751

[4] G. Liu, Y. Wang, M. A. Orgun, and E. Lim, "Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks," *IEEE Trans. Services Computing*, 2013. [Online]. Available: https://doi.org/10.1109/TSC.2011.58

[5] P. Monagle, A. K. Chan, N. A. Goldenberg, R. N. Ichord, J. M. Journeycake, U. Nowak-Göttl, and S. K. Vesely, "Antithrombotic therapy in neonates and children: antithrombotic therapy and prevention of thrombosis: American college of chest physicians evidence-based clinical practice guidelines," *Chest*, 2012.

[6] N. H. Tran, K. P. Choi, and L. Zhang, "Counting motifs in the human interactome," *Nature communications*, vol. 4, p. 2241, 2013.

[7] J. Ugander, L. Backstrom, and J. Kleinberg, "Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 1307–1318.

[8] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[9] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield, "Efficient graphlet counting for large networks," in *2015 IEEE International Conference on Data Mining*. IEEE, 2015, pp. 1–10.

[10] R. A. Rossi, N. K. Ahmed, and E. Koh, "Higher-order network representation learning," in *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018.

[11] Y. Yu, Z. Lu, J. Liu, G. Zhao, and J. Wen, "Rum: Network representation learning using motifs," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, April 2019, pp. 1382–1393.

[12] H. Zhao, X. Xu, Y. Song, D. L. Lee, Z. Chen, and H. Gao, "Ranking users in social networks with higher-order structures," in *AAAI*, 2018, pp. 232–240. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16122

[13] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher, "Scalable motif-aware graph clustering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1451–1460.

[14] V. Batagelj, P. Doreian, N. Kejzar, and A. Ferligoj, *Understanding large temporal networks and spatial networks: Exploration, pattern searching, visualization and network evolution*. John Wiley & Sons, 2014, vol. 2.

[15] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1311–1322.

[16] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang, "Online search of overlapping communities," in *SIGMOD*. ACM, 2013.

[17] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*, 2011, pp. 115–148. [Online]. Available: https://doi.org/10.1007/978-1-4419-8462-3_5

[18] N. Developers, "Neo4j," *Graph NoSQL Database [online]*, 2012.

[19] Y. Zhang and S. Parthasarathy, "Extracting analyzing and visualizing triangle k-core motifs within networks," in *ICDE*, 2012, pp. 1049–1060. [Online]. Available: https://doi.org/10.1109/ICDE.2012.35

[20] C. E. Tsourakakis, "The k-clique densest subgraph problem," in *WWW*, 2015, pp. 1122–1132. [Online]. Available: https://doi.org/10.1145/2736277.2741098

[21] J. Hu, R. Cheng, K. C.-C. Chang, A. Sankar, Y. Fang, and B. Y. Lam, "Discovering maximal motif cliques in large heterogeneous information networks," in *ICDE*, 2019.

[22] T. Milenković, W. L. Ng, W. Hayes, and N. Pržulj, "Optimal network alignment with graphlet degree vectors," *Cancer informatics*, 2010.

[23] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[24] X. Chen, Y. Li, P. Wang, and J. Lui, "A general framework for estimating graphlet statistics via random walk," *Proceedings of the VLDB Endowment*, 2016.

[25] P. Wang, J. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan, "Efficiently estimating motif statistics of large networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 2, p. 8, 2014.

[26] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 91–100.

[27] V. Batagelj and M. Zaveršnik, "Short cycle connectivity," *Discrete Mathematics*, vol. 307, no. 3-5, pp. 310–318, 2007.

[28] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[29] M. Maier, U. von Luxburg, and M. Hein, "Influence of graph construction on graph-based clustering measures," in *NIPS*, 2008, pp. 1025–1032. [Online]. Available: http://papers.nips.cc/paper/3496-influence-of-graph-construction-on-graph-based-clustering-measures

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. [Online]. Available: http://mitpress.mit.edu/books/introduction-algorithms

[31] A. Pinar, C. Seshadhri, and V. Vishal, "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, 2017, pp. 1431–1440.

[32] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting," *Bioinformatics*, vol. 30, no. 4, pp. 559–565, 2014.

[33] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of a*," *J. ACM*, vol. 32, no. 3, pp. 505–536, 1985. [Online]. Available: https://doi.org/10.1145/3828.3830

[34] A.-C. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. J. Jensen, S. Bastuck, B. Dümpelfeld *et al.*, "Proteome survey reveals modularity of the yeast cell machinery," *Nature*, vol. 440, no. 7084, p. 631, 2006.

[35] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis *et al.*, "Global landscape of protein complexes in the yeast saccharomyces cerevisiae," *Nature*, 2006.

[36] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[37] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[38] V. Carletti, P. Foggia, A. Saggese, and M. Vento, "Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with vf3," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 804–818, 2017.

[39] L. Lü, C.-H. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks," *Physical Review E*, vol. 80, no. 4, p. 046122, 2009.

[40] Y. Yao, R. Zhang, F. Yang, J. Tang, Y. Yuan, and R. Hu, "Link prediction in complex networks based on the interactions among paths," *Physica A: Statistical Mechanics and its Applications*, vol. 510, pp. 52–67, 2018.

[41] G. Abuoda, G. D. F. Morales, and A. Aboulnaga, "Link prediction via higher-order motif features," *arXiv preprint arXiv:1902.06679*, 2019.

[42] J. Cao, B. Li, and X. Gui, "Research on the influence of network motif on link prediction," *DEStech Transactions on Computer Science and Engineering*, no. itms, 2016.

[43] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[44] H.-W. Mewes, C. Amid, R. Arnold, D. Frishman, U. Güldener, G. Mannhaupt, M. Münsterkötter, P. Pagel, N. Strack, V. Stümpflen *et al.*, "Mips: analysis and annotation of proteins from whole genomes," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D41–D44, 2004.