Received 19 January 2023, accepted 11 March 2023, date of publication 16 March 2023, date of current version 22 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3257887

RESEARCH ARTICLE

Mining Top-k Frequent Patterns in Large Geosocial Networks: A MNIE-Based Extension Approach

CHANGBEN ZHOU, JIAN XU, MING JIANG, DONGHANG TANG, AND SHENG WANG
School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Changben Zhou (201050060@hdu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61572165.

ABSTRACT Frequent pattern mining (FPM) has played an important role in many graph domains, such as bioinformatics and social networks. In this paper, we focus on geo-social graphs, a kind of social network augmented by geographical information. However, in addition to the exponential time complexity of the problem, we face the challenge of efficient subgraph retrieval since we are interested in patterns in a specific region in such a network. For this reason, we formulate the top-k FPM problem in large geo-social networks. Specifically, we devise a novel framework for subgraph retrieval and FPM mining with a series of optimizations. First, we propose a neighboring-aware R-tree (NaR-Tree) index structure to alleviate the challenge of retrieving subgraphs from a large graph. NaR-Tree is a variant of R-tree in which each nonleaf tree node further maintains some edge statistics information for the rectangle related to it. Second, we define the concept of minimum image-based support of edges (MNIE). With the help of the NaR-Tree and MNIE-based pattern extension approach, a mining algorithm that addresses the problem of exponential candidate patterns is proposed. We also present a lazy retrieval strategy to reduce the frequency of subgraph retrieval. Finally, we adopt an edge sampling approach to further accelerate the mining process. Extensive experiments on real-world and synthesized datasets are conducted to demonstrate the effectiveness and efficiency of our solution.

INDEX TERMS Frequent pattern mining, geo-social network, NaR-tree, edge sampling.

I. INTRODUCTION

With the near-ubiquitous diffusion of location-aware mobile devices and ubiquitous internet access, new applications such as group-based activity planning [1], POI(Point of Interest) recommendation [2], and geo-crowdsourcing [3] are emerging. These applications combine both location and social relations to generate valuable search results for either business or social goods. However, to make these services more intelligent, it is better to reveal patterns or fundamental relations among users. To this end, we investigate frequent pattern mining in this paper, a central problem that plays a critical role in all graph-related applications since graphs are typically used to model relations among users in a geo-social network.

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

The goal of frequent subgraph mining is to find subgraphs whose appearances are top-k frequent [4], [5] or exceed a user-defined threshold [6], [7]. We argue that frequent subgraph mining finds its real applications in tasks such as activity planning or POI recommending.

Consider the collaboration graph G of Fig. 1 and a user who is interested in mining collaborations among researchers in a region to promote cooperative research among institutes in the area. Typically, the nodes in such graphs represent authors, and they are labeled as DM (data mining), KD (knowledge discovery) or IR (information retrieval) researchers, each of which has its spatial information generally described by coordinates. The edges represent interactions or cooperation between these researchers. Frequent subgraphs can be used here to show collaborations among authors having the same interest in work. To reveal more detailed relations, we progressively reduce the frequency threshold or change the input core pattern until interdisciplinary collaborations are discovered.

Another example is POI recommending. Suppose we have users and enjoyable places in a graph; check-in data in these places represent the user interest in them. Then, we can use this graph to mine the frequent visit mode to facilitate POI recommendation. That is, when recommending a place or planning a trip, the historic pattern found regarding a user's label (e.g., young or old, man or women), POI label (e.g., cinema, restaurant, gym) and checking records will give customers a better reference.

The literature evaluates the frequency of a subgraph S in a graph G by looking for isomorphisms of S in G [5], [6], [7]. Isomorphisms mean exact matches of node labels and edges in the subgraph with a pattern. The problem is well studied in a general graph, which can be a collaboration graph without geo-attributes [7], bioinformatics network [4] or knowledge graph [5]. In this paper, different from previous work, we address a new kind of geo-social frequent pattern mining in networks with constraints on a spatial and fixed label space, hereafter called GsFPM for brevity. A GsFPM mining is a top-k query that takes three arguments: $(\Lambda; L; c; k)$, where Λ is the spatial constraint, L is the label constraint, and c is a core pattern defined in L. The spatial constraint Λ is a range constraint, and L imposes that all nodes appear as a pattern instance with a label $l \in L$. We argue that spatial and fixed label space constraints are reasonable assumptions in real life since the geo-social network is so large that it is not necessary to explore the relations among all labels in the whole graph. Additionally, a GsFPM query starts from a core pattern, and the process can be easily extended to explore all patterns in the label space constrained by L.

Mining frequent patterns in geo-social networks faces the same challenges as previous works; the first is the exponentially large number of candidate patterns. The possible frequent patterns are obtained from the combination of all feasible nodes and edges in the network. The size of the search space is obviously huge and is exponential in the cardinality of the nodes and edges. The second challenge comes from evaluating the frequency of each candidate pattern. An obvious definition of support for a pattern is the number of its occurrences in the input graph [8]. However, such a definition is not anti-monotone since there are cases where a subgraph appears fewer times than its extension, which results in not being allowed to develop methods that effectively prune the search space [7].

In addition to the above two challenges, to conduct GsFPM in a large geo-social network, we face a third problem: how to retrieve a specific subgraph with spatial and label space constraints efficiently.

Due to the additional constraint, traditional frequent pattern mining algorithms [6], [7] cannot be directly applied to GsFPM. That is, we extract the desired subgraph first and then apply it to achieve the results. Thus, when GsFPM is processed in a large network, considerable time is spent on obtaining the nodes and edges that satisfy both spatial and label constraints.

In this work, we propose a novel framework for the GsFPM problem, which addresses the above three challenges. Specifically, we first devise a neighboring-aware R-tree (NaR-Tree) to organize the spatial and social relationships of each node in the geo-social network. It helps to access subgraphs efficiently in constraint areas and filter out nodes with undesirable labels, thus accelerating the speed of subgraph retrieval. We modify node-based MNI to edge-based MNIE), which counts the minimum number of unique edges. Different from the previous frequent pattern mining algorithm, which extends the candidate pattern based on the frequency of edge appearance, we also devise a MNIE-based extension approach and lazy retrieval strategy to speed up mining. Finally, we propose an edge sampling-based approach to accelerate mining. To the best of the authors' knowledge, this is the first work on frequent pattern mining in geo-social networks.

In summary, our main contributions are as follows:

- We present the GsFPM problem, which can be used in many geo-social network analytical applications.
- We construct a basic solution that integrates the most advanced frequent pattern mining techniques.
- We devise NaR-Tree to alleviate the challenge of retrieving subgraphs from a large graph and a MNIE-based extension approach to alleviate the challenge of exponential candidate patterns. We also adapt a lazy retrieval strategy to reduce retrieval times.
- Extensive experiments on real-world and synthesized datasets are conducted to demonstrate the effectiveness and efficiency of our solution. We verify that our algorithm is far more efficient than the basic solution and can be scaled to networks with millions of nodes.

The rest of the paper is organized as follows. Section II surveys related work. Section III introduces the fundamental concepts of frequent patterns and formally defines the GsFPM problem. Section IV proposes an R-tree-based baseline solution. Section V presents our mining framework with several optimization algorithms. Section VII presents the experimental evaluation, and Section VIII presents the conclusions.

II. RELATED WORK

A. TRANSACTIONAL MINING

Transactional mining and single graph mining are two forms of FPM. In transactional mining, mining algorithms require relatively small graphs as inputs [6], [9], [10], [11], [12], [13]. A pattern's frequency is determined by the number of graph transactions in which it appears, regardless of how often it appears in one transaction.

The FSG [9] algorithm performs a breadth-first search, and candidate subgraphs are generated by combining frequent subgraphs. Actually, AGM [10] and FSG adopt an improved version of the Apriori-based approach, which combines small frequent patterns together to create new candidate patterns. Their join strategy can exclude many infrequent subgraphs. However, the limitations of these approaches are the complex combinations and high memory consumption. The gSpan [6] is a popular frequent subgraph mining algorithm that is based on pattern modeling. Its extensions of a subgraph must appear before, which avoids the generation of nonexistent graphs. To efficiently identify duplicate subgraphs, gSpan calculates a DFScode for each searched subgraph. If two graphs are isomorphic, their minimum DFScodes will be equal. The gSpan algorithm overcomes the limitations of the breadth-first approaches by only keeping subgraphs associated with the recursive call currently in progress. The FPGraphMiner [11] proposes a structure named BitCode (a bit vector) to group subgraphs. Therefore, its mining process uses depth-first search without backtracking, which is highly efficient. It is true that the FPM is useful in practice, but it is difficult to set the minimum support threshold. A high threshold will cause few patterns to be found, while a low threshold will result in a large number of meaningless patterns.

To address this issue, a top-*k* frequent subgraph mining problem was proposed. The threshold for the mining process can be set without any prior knowledge. TGP [12] is the first algorithm for this problem, and it gradually raises the minimum support threshold, which is initially set to 0. To find the top-*k* patterns, TGP generates all patterns, which is obviously inefficient. To address this problem, the FS^3 [13] algorithm, adopting the Markov Chain Monte Carlo method, was proposed to trade precision for efficiency. Hence, FS^3 may return infrequent patterns.

B. SINGLE GRAPH MINING

The input data for single-graph frequent pattern mining is a single large graph [5], [7], [14], [15], [16], [17]. The frequency of a pattern is based on its occurrences in the single graph. SIGRAM [14] stores frequent embeddings to extend for finding larger subgraphs. However, storing embeddings consumes considerable memory, and the computational cost of SIGRAM is high due to the use of MIS. GRAMI [7] is an algorithm that does not store all embeddings and takes advantage of the features of MNI. In contrast to previous approaches, GRAMI only identifies the minimal set of instances that satisfy the frequency threshold without enumerating all instances. GRAMI adopts DFScode in gSpan so that no duplicate subgraphs are found in the result. Both directed and undirected graphs are supported by GRAMI, which also proposes an approximate version.

However, GRAMI cannot mine patterns in multirelational graphs. MuGram [15] performs a depth-first search starting from frequent edges to handle this case. ScaleMine [16] divide mining tasks among separate CPU cores by optimizing GraMi. Another algorithm SSIGRAM [17] divides tasks among threads on a cluster in distributed systems.

However, threshold-based FPM has difficulty setting a suitable threshold. To address this issue, FastPat [5] studies a core-based top-*k* frequent pattern discovery problem. FastPat proposes an upper bound of MNI and an index structure called meta-index. It also designs a join-based approach to efficiently compute the MNI.

C. SUPPORT MEASURES

In the process of FPM in a single graph, how to measure the frequency of a pattern is a key problem. Intuitively, the frequency should be determined by the occurrence of patterns, while this strategy can lead to an exponentially large search space. MIS [18] is an anti-monotonic support measure, but it has been found to be NP-hard, which makes it unsuitable for practical application. MNI [19] has been extensively adopted in the FPM literature, which is the most practical measurement, and the calculation can be done in linear time. However, the MNI support lacks intuitiveness, so [8] proposes two support measures: MI and MVC, which combine the advantages of existing approaches.

D. SPATIAL INDEX

Spatial indices are used to efficiently retrieve multidimensional objects or objects with spatial extents. Hierarchical tree-based spatial index structures [20], [21] are a popular type of spatial index that can be classified into two categories: (a) Grow-and-Post trees [22] such as R-tree [23] and R*-tree [24], which extend the B-tree structure. The main idea is to recursively partition the spatial data based on spatial proximity clustering [25]. (b) Space-Partitioning trees include the Quad-tree [26] and k-d tree [27], which recursively decomposes the space into disjoint partitions [25].

The augmented spatial index is another type of spatial index that is augmented with information to resolve different categories of geospatially related queries [28], [29]. The keyword-first index finds documents that satisfy both spatial and textual requirements by first locating them based on keywords and then by their location. The RTree-first index, used in a reversed manner, is another augmented spatial index. KR*-tree [29] is extended from R-Tree, and keywords can be stored in its internal tree nodes. Instead of directly using keywords, IR^2-tree [28] extends R-Tree with signature files. Similar to IR^2-tree, IR-Tree [30] is proposed for solving top-*k* geospatial and textual document search problems, and it stores document summaries such as term frequency and document frequency. Due to the complexity of graph data, it cannot be implemented for geospatial graphs.

III. PRELIMINARIES

This section reviews some basic knowledge of geo-social networks. The mathematical notation used in this paper is summarized in Table 1. Then, we introduce R-tree, which is used to index nodes, and the minimum image-based support (MNI), which is used to measure pattern frequency. With these concepts, we formally define the GsFPM problem.

TABLE 1. Summary of notation.

Symbol	Description
G_s	A geo-social graph
u, v	Nodes
(u, v)	Edge connecting node u and node v
V	Node set
E	Edge set
$\|V\|, \|E\|$	The size of V and E
$L(.)$	Label function associating each node with a label
L	Label constraint
Λ	Spatial constraint
c	Input core pattern
$P(V_p, E_p)$	Pattern with node set V_p and edge set E_p
$P.freq$	Frequency of pattern P
$S(V_s, E_s)$	Subgraph with node set V_s and edge set E_s
MNIE(P)	MNIE value of pattern P
MNI(P)	MNI value of pattern P
PE(P)	Pattern elements of Pattern P
NTS(S)	Node types of subgraph S
ETS(S)	Edge types of subgraph S
Q_{gs}	GsFPM query in a network
f	Bijection function between the subgraph and pattern

A. PROBLEM STATEMENT

Definition 1 (Geo-Social Networks): A geo-social network $G_s(V, E, L)$ is an undirected and labeled graph. V represents a set of nodes, and $E(E \subseteq V \times V)$ represents a set of edges. $L(.)$ is a label function such that every node $v \in V$ is associated with a label indicating its type, i.e., $L(v) \in L$.

The geographical attribute or location for a node v consists of two attributes of node v, i.e., $v.x$ and $v.y$, which are the x and y coordinates of node v, respectively.

Definition 2 (Pattern): A pattern $P(V_p, E_p)$ is a weakly connected subgraph obtained from pattern space, in which each node $v' \in Vp$ is assigned a label $L(v')$, and each edge $e' \in Ep$.

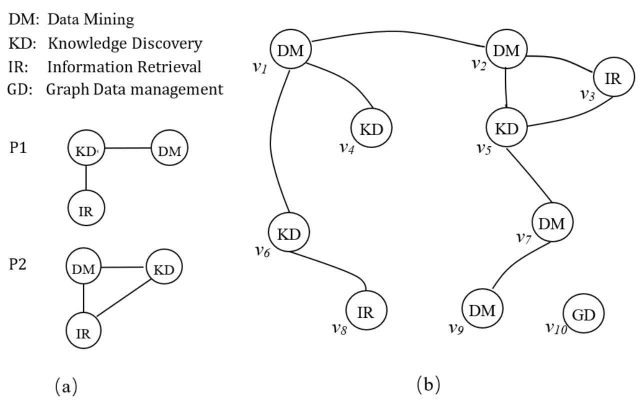

DM: Data Mining
KD: Knowledge Discovery
IR: Information Retrieval
GD: Graph Data management
P1
P2
KD
DM
IR
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}
GD
(a)
(b)

FIGURE 1. (a) Patterns P_1 and P_2. (b) A collaboration graph G; nodes correspond to authors (labeled with their field of work).

Example 1: Fig. 1 (a) shows two patterns: P_1: (DM)-(KD)-(IR) and P_2: (DM)-(KD)-(IR)-(DM), indicate cooperations among data mining, knowledge discovery and information retrieval scientists. It should be noted that not every pattern that exists in pattern space will be found in a geo-social graph. It is obvious that pattern (KD)-(IR)-(KD) does not appear in the graph shown in Fig. 1(b).

Definition 3 (Instance): An instance of pattern$P(V_p, E_p)$ is a subgraph $S(V_s, E_s) \in G_s$, for which the subgraph isomorphism of $P(V_p, E_p)$ to $S(V_s, E_s)$ is satisfied. There exists a bijection function $f : V_p \rightarrow V_s$ satisfying:

- $|V_p| = |V_s|$ and $|E_p| = |E_s|$;
- $L_p(v') = L_s(f(v'))$;
- $\forall(u', v') \in E_p \implies (f(u'), f(v')) \in E_s$
- $\forall(u', v') \in E_s \implies (f^{-1}(u'), f^{-1}(v')) \in E_p$

If there exists an instance of pattern P in G_s, that means mapping function f between the subgraph and pattern needs to map each node and edge in P to a distinct node and edge, and the label of each node in the subgraph also has the same label as the nodes in the pattern. Therefore, a pattern can be considered a general schema for relations among some nodes.

Example 2: Fig. 1(b) shows three instances for P_1: (v_1-v_6-v_8), (v_2-v_5-v_3), and (v_7-v_5-v_3). For pattern P_2, there is only one instance, (v_2-v_3-v_5-v_2).

Definition 4 (Geo-Social Frequent Pattern Mining Problem): A GsFPM query in a network $G_s(V, E, L)$ represented as $Q_{gs} = (\Lambda; L; c; k)$, where Λ is a spatial constraint (range constraint in this paper), L denotes label constraint, k is an integer and c is a user specified core, which is obtained from label space constrained by L, returns the top-k frequent patterns derived from pattern space.

Example 3: Take Fig. 1(b) as an example. If we relax Λ to include the whole graph, L = {DM, IR, KD}, k = 2 and c: (DM)-(KD), a GsFPM query returns top-1 pattern (DM)-(DM)-(KD), since it appears four times in the graph, in the form of (v_1-v_2-v_5), (v_2-v_1-v_4), (v_2-v_1-v_6), and (v_9-v_7-v_5). It returns (DM)-(KD)-(IR) as the top-2 pattern.

B. MINIMUM IMAGE-BASED SUPPORT (MNI)

Intuitively, evaluating the frequency of a pattern P is a matter of counting the number of instances it has in the geo-social network. Each instance in the network is an isomorphism of pattern P. To find all instances, it is necessary to perform an exhaustive search in the network. Because the subgraph isomorphism problem has been shown to be NP-hard [31], many algorithms [32], [33], [34], [35] have been introduced to speed up the search for isomorphism subgraphs.

To support practical subgraph mining and efficient frequent subgraph evaluation, several antimonotone metrics have been proposed in the literature. Reference [19] introduces minimum image-based support, [14] proposes the concept of maximum independent sets, and [8] presents a framework that brings together minimum image-based and overlap graph-based support measures. Since MNI is widely used in the literature, we adopt it as our frequency measure [7], [14], [36].

Definition 5 (Minimum Image-Based Support(MNI) [19]): Given pattern $P(V_p, E_p)$ and geo-social network $G_s(V, E, L)$, minimum image-based support is the number of unique nodes in G_s to which a node from pattern P is mapped. Suppose pattern P has m isomorphisms in G_s, i.e.,$f_1, f_2, f_3, ..f_m$, where f is a mapping function, and its corresponding image set is $M(v) = f_1(v), f_2(v), f_3(v) \dots, f_m(v)$. MNI is defined as: $MNI(P) = \min\{|M(v)| \text{ for all } v \in V_p\}$.

Example 4: The pattern P: (DM)-(KD) corresponds to 4 instances in Fig. 1(b). For the nodes in $V_p = (v'_{DM}, v'_{KD})$, we have their image sets as $M(v'_{DM}) = \{v_1, v_2, v_7\}$ and $M(v'_{KD}) = \{v_4, v_5, v_6\}$. Hence, the MNI of P in G_s is $MNI(P) = \min\{|M(v'_{DM})|, |M(v'_{KD})|\} = MNI\{3, 3\} = 3$.

It is easy to see that MNI satisfies downward closure, and [19] shows that MNI's antimonotonicity allows efficient pruning during the search of instances for a pattern.

IV. R-TREE-BASED BASIC SOLUTION

We consider a scenario described in [37], in which the user's spatial and social information are stored separately on external disk storage. Dealing with spatial information, it is popular to use R-tree [38], a tree data structure for storing location data indices in an efficient manner. R-trees are highly useful for spatial data queries and storage for many real-life applications [25], [39]. A basic approach for GsFPM query processing on an R-Tree index of user locations is as follows.

For a $Q_{gs} = (\Lambda; L; c; k)$, we first find all nodes located inside Λ (range constraint) via R-tree, filter out nodes with labels not in L and then start from core pattern c to conduct frequent pattern mining. We now present our basic solution for the GsFPM query, which is summarized in Algorithm 1. In general, we enumerate all patterns extended from the core pattern c in the retrieved subgraph and evaluate the MNI values of these patterns. Specifically, two priority queues are used to check the candidate patterns and maintain the current top-k patterns. The algorithm evaluates every pattern in the min-priority queue to ensure that these patterns meet requirements. Line 6 prunes the candidate patterns if their MNI upper bounds are lower than the min-priority queue's top pattern – the current k_{th} frequent pattern.

Algorithm 1 GsFPM-B
```
Input: Graph $G_s$, spatial constraint $\Lambda$, label set $L$, core pattern $c$
    and integer $k$
Output: Top-$k$ frequent patterns extended from $c$
 1: $Gs$ = Retrieve nodes with constraint from $G_s$
 2: Initialize candidate pattern set - max-priority queues $q_1$
 3: Initialize top-$k$ pattern set - min-priority queues $q_2$
 4: while $q_1$ is not empty do
 5:     $P_{cur}$ = EXTRACT-MAX($q_1$).
 6:     if $P_{cur}.freq$ > MINMUN($q_2$).$freq$ then
 7:         $P_{cur}.freq \leftarrow$ MNIEvaluate($P_{cur}$)
 8:         if $P_{cur}.freq$ > MINMUN($q_2$).$freq$ or SIZE($q_2$) $\le k$ then
 9:             INSERT($q_2$, $P_{cur}$)
10:             Extension($G_s$,$P_{cur}$, $q_1$)
11: return $q_2$
```

In Algorithm 1, Line 5 removes and returns the element of q_1 with the largest key, and MINIMUM() returns the element of q_2 with the smallest key and compares it with the current pattern MNI(Line 6). Then, we evaluate the MNI value of these patterns (Line 7) with MNIEvaluate(), a function adapted from [7], which is the state-of-the-art technique for MNI computation. It works as a constraint satisfaction problem and checks whether each node of a pattern has threshold instances in the graph and does not return an exact MNI value. However, MNIEvaluate() is a modified version of the original version and returns the exact MNI value after searching all instances. The same approach was adopted by [5]. Line 9 inserts pattern P_{cur} into the set q_2 where the top-k results are kept. Algorithm 1 returns the final results at Line 11.

Algorithm 2 Extension(G,P,Q)
```
Input: A Graph $G$, a pattern $P$, candidate pattern set $Q$
Output: All frequent patterns that extend $P$.
1: for each $(e, v)$ in $G$ do
2:     if $(e, v)$ can be used to extend $P$ then
3:         Generates a new pattern $P'$ from $P$
4:         $P'.freq \leftarrow P.freq$
5:         Put $P'$ into $Q$
```

To enumerate all possible subgraphs in a graph, the edge-growing method [6], [7] starts with an edge and gradually adds neighboring edges to extend the graph. Algorithm 2 enumerates all candidate patterns that can be extended from the core pattern c, and it avoids duplicate patterns by using the DFScode technique of gSpan [6].

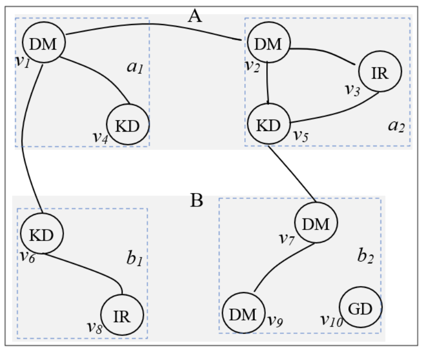
V. THE MNIE-BASED TOP-k FREQUENT PATTERN MINING FRAMEWORK

This section proposes a novel framework to efficiently address the GsFPM problem in a geo-social network.

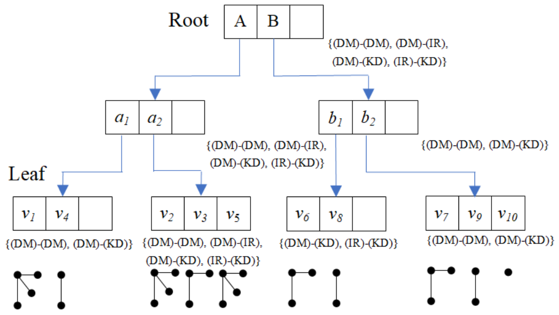
Obviously, the approach of the basic solution adopted is inefficient for $Q_{gs} = (\Lambda; L; c; k)$ in some scenarios. For example, there is a high chance of keeping the irrelevant node in a sparse graph while searching for frequent patterns. As we can see from Fig. 1, node v_{10} is not connected with any other nodes and cannot be used to construct any pattern. The problem will worsen when extending the pattern from a core. Many nodes and edges in which the query is not of interest or is not closely related to the core pattern will be included in the search subgraph.

The second problem with the basic solution is that it always checks every edge in the graph to extend the pattern. A typical approach to extend the pattern or add edges to the pattern is according to their frequency of appearance in the graph. Edges with higher frequency enjoy more opportunity to be added to candidate patterns. However, as we revealed in Section VI, pattern extension based on the knowledge of the whole graph is not always appropriate for us, since what are interested in in this work are patterns growing from a core. The edges that should be used in extension are edges that appear frequently around instances presented.

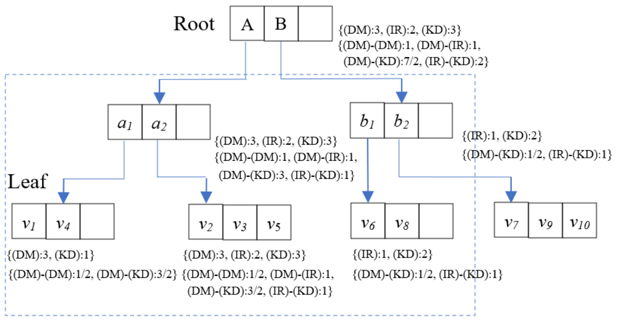
For the two reasons above, we devise a novel index structure extended from R-tree: neighboring-aware R-tree, which allows us to conduct node and edge filtering during subgraph extraction from the original networks. We also introduce a new extension approach, which extends the candidate pattern based on the knowledge of the subgraph around



(a) MBRs of graph G in Fig. 1(b)



(b) NaR-Tree index structure


Root
A
B
{(DM):3, (IR):2, (KD):3}
{(DM)-(DM):1, (DM)-(IR):1, (DM)-(KD):7/2, (IR)-(KD):2}
a_1
a_2
b_1
b_2
{(DM):3, (IR):2, (KD):3}
{(DM)-(DM):1, (DM)-(IR):1, (DM)-(KD):3, (IR)-(KD):1}
{(IR):1, (KD):2}
{(DM)-(KD):1/2, (IR)-(KD):1}
Leaf
v_1 v_4
v_2 v_3 v_5
v_6 v_8
v_7 v_9 v_{10}
{(DM):3, (KD):1}
{(DM)-(DM):1/2, (DM)-(KD):3/2}
{(DM):3, (IR):2, (KD):3}
{(DM)-(DM):1/2, (DM)-(IR):1, (DM)-(KD):3/2, (IR)-(KD):1}
{(IR):1, (KD):2}
{(DM)-(KD):1/2, (IR)-(KD):1}

(c) Example of NaR-Tree search

FIGURE 2. **NaR-Tree index structure and search process.**

instances previously found. With NaR-Tree and subgraph-based extension in hand, we have our frequent pattern searching framework.

A. NEIGHBORING-AWARE R-TREE

Since Q_{gs} involves spatial and social constraints, both spatial and social relations should be indexed simultaneously to expedite its processing. However, traditional R-tree only indexes the spatial information of the nodes. In this subsection, we first introduce the structure of NaR-Tree and then present details of the search algorithm.

From the example shown in Fig. 2(a), we can make an easy observation: when we extract unique edges in a graph and its subgraph, the edge types from the graph include edge types from the subgraph. In this figure, we have unique edge types {(DM)-(KD), (DM)-(IR), (IR)-(KD), (DM)-(DM)} for the whole graph and {(IR)-(KD), (DM)-(DM)} for the subgraph constrained by rectangle B. In the following discussion, we use Edge Types (ET) to denote unique edge types in a rectangle region or subgraph. With this observation, we have the NaR-Tree index.

A NaR-Tree is a variant of the R-tree in which each nonleaf tree node further maintains some edge type information for the rectangle related to it, and each graph node indexed by leaf tree nodes maintains its neighbor information. Fig. 2(b) exemplifies a NaR-Tree, which satisfies the following properties in addition to the traditional R-tree:

- Graph nodes indexed by NaR-Tree keep information of their neighbors.
- Index record in a leaf node has ET information of nodes included in the minimum bounding rectangle (MBR) of this node.
- Index record in a nonleaf node has ET information to keep the union of its children.

Example 5: Fig. 2(b) shows an instance of NaR-Tree. For leaf node a_1, a_2, we have $ET(S_{a_1})$={DM-DM, DM-KD}, $ET(S_{a_2})$ = {DM-DM, DM-KD, DM-IR, IR-KD}; for both of them, DM-DM is an outgoing edge. For MBR A, we have $ET(S_A) = ET(S_{a_1}) \cup ET(S_{a_2})$ = {DM-DM, DM-KD, DM-IR, IR-KD}.

To build a NaR-Tree, we adopt a bottom-up approach in our implementation. We first construct a standard R-tree based on the location of the nodes. Then, from edges connected with each graph node, we add ET information into its leaf node entry. For a nonleaf entry e, let $e_1, e_2, e_3, \ldots, e_m$ (m is the minimum number of entries that will fit in one NaR-Tree node); then, $ET(S_e)$ can be computed by recursively applying the union operation among $e_1, e_2, e_3, \ldots, e_m$.

The purpose of devising NaR-Tree is to retrieve the subgraph for a special pattern. While a subgraph has its edge types, a pattern also has edge types. To make a difference between them, we have Definition 6.

Definition 6 (Pattern Element): Pattern elements are edge types that appear in a pattern $P(V_p, E_p)$. Pattern elements (PE) are presented as a set of ordered pairs of labels. The order of labels is their alphabetical order.

To extract a subgraph of a pattern, we first broke the pattern into elements, then together with spatial and label constraints, we filtered out nodes and edges not related to the pattern. For example, when we want to retrieve a subgraph for a pattern (DM)-(KD)-(IR), we first obtain pattern elements {(DM)-(KD), (IR)-(KD)} and then search the NaR-Tree shown in Fig. 2(b). Since there is no such edge type in rectangle b_2, nodes and edges are filtered out.

With NaR-Tree, we propose the following search algorithm.

Algorithm 3 returns a subgraph and statistical information of this subgraph. A NaR-Tree also works as a temporal data structure that is used to store node and edge type information for the returned subgraph. We declare two collection variables Node Type of Subgraph (NTS) and Edge Type of Subgraph (ETS) to keep the maximal degree of a special label mapped

Algorithm 3 NaR-Tree Search(T, S, Λ, L, $PE(P)$)

```
Input: Root node of a NaR-Tree $T$, spatial constraint $\Lambda$, pattern $P$
Output: A subgraph $S$ satisfy $\Lambda$, $L$ and $c$ constraint, Updated $T$.
1:  if $T$ is not a leaf then
2:      for each child $e_i$ of $T$ do
3:          if $e_i$.MBR overlaps $\Lambda$ && $PE(P) \cap T.ET \neq \varnothing$ then
4:              NaR-Tree Search($e_i$, $S$, $\Lambda$, $L$, $PE(P)$)
5:              Update $T.NTS$ and $T.ETS$
6:  else
7:      for each node $u$ of $T$ do
8:          for each edge $(u,v)$ of $u$ do
9:              if $u.location$ overlaps $\Lambda$ && $v.location$ overlaps $\Lambda$ &&
                $L(u) \in L$ && $L(v) \in L$ && $(L(u), L(v)) \in PE(P)$ then
10:                 $S = S \cup \{(u,v)\}$
11:                 Update $T.NTS$ and $T.ETS$
```

nodes and the unique edge numbers of pattern elements mapped. The purpose of this information will be introduced in detail in section V-B.

When the algorithm finds a qualified node, as Line 9 indicates, both nodes at the end of an edge are mapped to labels in L, and its type $(L(u), L(v))$ (suppose it is in the order of $L(u)$, $L(v)$) is included in $PE(P)$. Then, it will be added to set S, which is used to keep the returned subgraph. For example, the algorithm encounters two edges in the rectangle b_1, as shown in Fig. 2(c), (DM)-(KD) and (IR)-(KD). The unique occurrence of (IR)-(KD) is 1. (DM)-(KD) is the type for an outgoing edge. The occurrence in this situation is 1/2. Node types in this rectangle are (IR):1 and (KD):2, where the number indicates maximal degree for the label IR is 1 (occurred in v_8) and maximal degree for label KD is 2 (occurred in v_6). Finally, the algorithm returns a subgraph S. At the root node T of NaR-Tree, we have the maximal degree for each label and a unique edge number for each type of edge.

We have Lemma 1 and Lemma 2 for the correctness of Algorithm 3.

Lemma 1 (Instance Integrity of Subgraph S): Subgraph S returned by Algorithm 3 includes all instances of pattern P constrained by Λ and L.

Proof: We now prove the lemma by contradiction. Suppose that an instance I is not included in S. Then, there exists an edge $(u,v) \in I$, for which $(L(u), L(v)) \in PE(P)$ or $(L(v), L(u)) \in PE(P)$. This contradicts Line 10 in Algorithm 3, which has exhausted every edge of u. ■

Lemma 2 (Extension Integrity of Subgraph S): Subgraph S returned by Algorithm 3 includes all possible extensions of pattern P constrained by Λ and L.

Proof: The same as that of Lemma1. ■

B. MNIE-BASED PATTERN EXTENSION

As stated before, the usual constraint employed in frequent pattern mining is minimum support, especially MNI support. MNI in Definition 5 is a single node-based support measure that avoids potentially expensive maximal independent set computations compared to other support measures [8], [14]. It is obvious that pattern elements enforce more constraints than node-based measures since there is certain topology information with them. In this section, we modify node-based MNI to edge-based MNI, which count the minimum number of unique edges an element of the pattern is mapped to.

Definition 7 (Minimum Image of Edge-Based Support (MNIE)): Given pattern $P(V_p, E_p)$ and geo-social network $G_s(V, E, L)$, the minimum image of edge-based support is the unique edge number in G_s to which an element from pattern P is mapped. MNIE is defined as: $MNIE(P) = min\{|\, M(u',v')|: \forall\, u',v' \in P,\, M$ is a subgraph isomorphic mapping (instance) of P in G_s $\}$.

In the following discussion, we refer to MNI introduced in Definition 5 as the minimum image of node-based support (MNIN). With MNIE, we can tighten its upper bound by removing disqualified edges from a graph by checking the types of edges. The following lemma shows that MNIE is an antimonotonic support measure and can be used as the minimum support constraint.

Lemma 3 (Antimonotonicity of MNIE): Given a geo-social network $G_s(V, E, L)$, consider a pattern P and its extension P'. It holds that $MNIE(P) \geq MNIE(P')$.

Proof: Suppose P' is extended by adding an element (X,X'); there are two situations concerning this extension: $(X,X') \in PE(P)$ and $(X,X') \notin PE(P)$. When $(X,X') \in PE(P)$, we have $MNIE(P) = MNIE(P')$. When $(X,X') \notin PE(P)$, we have $MNIE(P) > MNIE(P')$ from Definition 7. ■

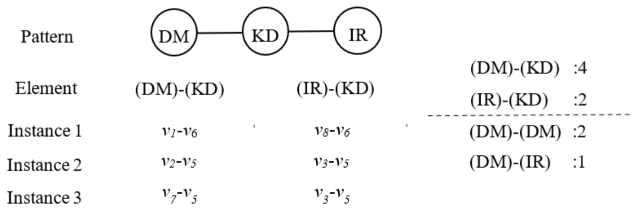


FIGURE 3. An example of the MNIE bound.

Example 6: Fig. 3 shows that pattern (DM)-(KD)-(IR) has two elements: (DM)-(KD) and (IR)-(KD). Unique edge numbers for these two elements are 3 and 2. Hence, we have MNIE value 2 for this pattern. There are two possible extensions to (DM)-(KD)-(IR): (DM)-(DM) and (DM)-(IR). After we extend the pattern, either for pattern (DM)-(DM)-(KD)-(IR) or (DM)-(KD)-(IR)-(DM), the MNIE value is less than or equal to the original value.

During frequent pattern mining, after retrieving subgraph S for a pattern from G_s, we have obtained NTS and ETS, as we introduced in the last subsection. These two values are the maximal degree of a special label mapped node and the unique edge numbers of pattern elements (or MNIE value). They can be used in pattern extension. We have Algorithm 4 to account for this procedure.

The difference between Algorithm 2 and Algorithm 4 is in Line 2 of the algorithm. We do not expand edge types with MNIE value less than the currently found k-th MNIE in the top-k queue to pattern P which is under consideration. Thus, we eliminate many unfeasible candidate patterns. When extending a pattern, we also check whether it is practical to expand by examining the maximal degree of nodes

Algorithm 4 Extension-S(S, P, Q)

```
Input: A subgraph $S$, a pattern $P$, candidate pattern set $Q$
Output: All frequent patterns that extend $P$
1: for each type $e_T$ in $S$ do
2:   if MNIE($e_T$) $> k_{th}$ MNIE in $q_2$ && $e_T$ can be used to extend $P$ then
3:     Generates a new pattern $P'$ from $P$
4:     $P'.freq \leftarrow P.freq$
5:     Put $P'$ into $Q$
```

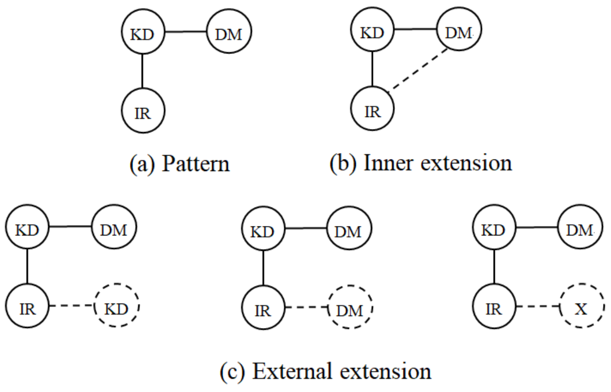


FIGURE 4. **Extension modes of a pattern.**

maintained in *NTS*. Algorithm 4 also uses the DFScode form from gSpan to remove duplicate candidates.

C. LAZY RETRIEVAL OF THE SUBGRAPH FOR A PATTERN

It is easy to observe that there are two types of extension: extension inside a pattern and outward extension. Fig. 4 illustrates these situations. Suppose P' is an extension of pattern P, and it is an extension inside of P. This means that when obtaining a subgraph for P' from G_s, we do not add more nodes or edges to the subgraph for P. In regard to an outward extension, there are two situations: with a pattern element already existing or with a new pattern element. Fig. 4 (c) shows these two situations. Either the other side of the edges appears as a node with a label available in pattern P or not, and we must retrieve the subgraph once again from the graph. The reason is that we do not check the node at the other side of the edge when retrieving the subgraph for P. Based on the observations above, we have Algorithm 5, which adopts a lazy retrieval strategy.

Algorithm 5 follows the approach adopted by GRAMI. The algorithm first conducts lazy retrieval of the subgraph for a pattern. Then, it searches all instances for each element in P (Line 4-11) to return the MNIE value. The optimization details used by GRAMI are not presented in the algorithm. We will show in the experiment section that the lazy retrieval approach effectively avoids frequent subgraph creation while obtaining frequent patterns.

The GsFPM-M follows the steps of the basic solution. Algorithm 6 describes the overall procedure of our mining framework. First, we construct a NaR-Tree for the graph (Line 1) and then modify Algorithm 1 with MNIE Evaluate() (Line 8) and Extension-S() (Line 11). We have Theorem 1 for the correctness of Algorithm 6.

Algorithm 5 MNIE Evaluate (T, S, P)

```
Input: NaR-Tree $T$, Subgraph $S$, Pattern $P$
Output: MNIE value from pattern $P$
1: if $PE(P) \nsubseteq ETS(S)$ then
2:   obtain $S$ from NaR-Tree searching
3: $occurrence = \infty$
4: for each element $e \in PE(P)$ do
5:   $tmp = 0$
6:   for each instance's edge $i_e$ of $e$ in $V$ do
7:     if $i_e$ is already marked then
8:       $tmp$++
9:     else
10:      if an instance of $P$ found then
11:        Mark all edges of this instance to $P$'s elements
12:        $tmp$++
13:  if $occurrence > tmp$ then
14:    $occurrence = tmp$
15: return $occurrence$
```

Algorithm 6 GsFPM-M

```
Input: Graph $G_s$, spatial constraint $\Lambda$, label set $L$, core pattern $c$ and integer $k$
Output: Top-$k$ frequent patterns extended from $c$
1: Construct NaR-Tree $T$ from $G_s$
2: Initialize subgraph $S$
3: Initialize candidate pattern set - max-priority queues $q_1$
4: Initialize top-$k$ pattern set - min-priority queues $q_2$
5: while $q_1$ is not empty do
6:   $P_{cur}$ = EXTRACT-MAX($q_1$).
7:   if $P_{cur}.freq >$ MINMUN($q_2$).$freq$ then
8:     $P_{cur}.freq \leftarrow$ MNIE Evaluate($T$,$S$,$P_{cur}$)
9:     if $P_{cur}.freq >$ MINMUN($q_2$).$freq$ or SIZE($q_2$) $\leq k$ then
10:      INSERT($q_2$, $P_{cur}$)
11:      Extension-S($S$,$P_{cur}$,$q_1$)
12: return $q_2$
```

Theorem 1 (Correctness of the MNIE-Based GsFPM Algorithm): The MNIE-based GsFPM algorithm returns an exact top-k frequent pattern with Λ, L and c constraints.

Proof: To show that MNIE-based GsFPM works correctly, we use the following loop invariant: At the start of each iteration of the while loop of lines 5, each pattern i ($1 \leq i \leq k$) queued in q_2 is the top-k frequent pattern in G_s with Λ, L and c constraint. This invariant is true prior to the first loop iteration, each iteration of the loop maintains the invariant, and the invariant shows correctness when the loop terminates. ∎

1) INITIALIZATION

Prior to the first iteration of the loop, the queue is empty. Then, the invariant remains.

2) MAINTENANCE

To see that each iteration maintains the loop invariant, observe that subgraph S obtained from the graph contains all instances according Lemma 1. Then, we have an exact MNIE value for this pattern. If the MNIE value is greater than the current k-th pattern, it will be inserted into q_2, and a series of candidate patterns will be created for it. According to Lamma 2, all possible patterns will be examined. If it is a feasible candidate, it will be put into q_1. This ensures the loop invariance for the next iteration.

3) TERMINATION

At termination, by the loop invariant, the patterns in q_2 are top-k frequent patterns.

4) COMPLEXITY ANALYSIS

The time complexity of Algorithm 6 consists of several parts. First is the construction of NaR-Tree. Its time complexity is $O(|V| + |E|)$. The second is retrieving a subgraph for a pattern. When spatial constraint Λ and label set L are relaxed to the whole graph, the time complexity is $O(|V| + |E|)$. The third part is instance enumerating for each pattern candidate. Let n be the number of nodes in the pattern; then, the time complexity is $O(|V|^{n-1})$ [7]. The last part is the number of candidate patterns explored. Algorithm 6 improves the efficiency mainly by reducing pattern candidates, the retrieving times and the size of the subgraph that is needed for a pattern.

VI. ACCELERATING MINING VIA EDGE SAMPLING

Geosocial networks have grown explosively in recent years, and there are millions or even billions of nodes and edges in such graphs. As a result, various graph sampling techniques have been proposed for accelerating mining of these complex and evolving networks [40]. Under the sampling approach, only a small subset of the nodes or edges from the original graph is selected for further processing.

Many graph sampling algorithms have been proposed in the literature [41], [42]. In particular, uniform random edge sampling (URE) and nonuniform random edge sampling (NURE) are two approaches that have been extensively studied. URE sampling scans a graph and puts each edge under scanning into the sampled graph with a constant probability. On the other hand, NURE samples edges with different probabilities.

In this work, we adopt a sample and hold approach [43], [44] that falls in the middle of URE and NURE. The approach takes a strategy of keeping the sampling probability of an arriving edge as a function of those already sampled. Consider a random process $H_i = \{H_i : i \in [|E|]\}$ where $H_i = 1$ if edge e_i is selected, and $H_i = 0$ otherwise. F_i denotes the set of possible outcomes $\{H_1, \cdots, H_i\}$. Thus, we have

$$P[k_i \text{ is sampled} | \{H_1, \cdots, H_{i-1}\}] = E[H_i | F_{i-1}] = p_i \tag{1}$$

where $p_i \in (0, 1]$ is a random probability that is determined by the previous sampling result. For $N_E = |E|$, an unbiased estimate would be

$$\hat{N_E} = \sum_{e_i \in \hat{E}} \frac{1}{p_i} \tag{2}$$

TABLE 2. Dataset details.($K = 10^3$,$M = 10^6$).

Dataset	Nodes	Edges	Distinct node labels	Average degree
Citeseer	3.3K	4.7K	6	2.86
GitHub	37.7K	289K	30	15.33
Brightkite	58.2K	214K	50	7.35
Mico	100K	1.08M	29	21.61
Twitch	168K	6.8M	21	80.87
Twitter	11.3M	85.3M	100	15.08

Following this approach, an arriving edge is sampled with probability q when its nodes match a node that currently has been sampled. If there is no match, the edge is stored with probability p. With this simple and single-pass approach, we have Algorithm 7.

```
Algorithm 7 NaR-Tree Search-Sampling($T$,$S$,$\Lambda$,$L$,$PE(P)$)
Input: Root node of a NaR-Tree $T$, spatial constraint $\Lambda$, pattern $P$
Output: A subgraph $S$ satisfy $\Lambda$, $L$ and $c$ constraint, Updated $T$
1: (same with Algorithm 3)
2: if $u \in V_s$ or $v \in V_s$ then
3:     $prob = p$
4: else
5:     $prob = q$
6: $S = S \cup \{(u, v)\}$ with probability $prob$
7: Update $T.NTS$ and $T.ETS$
```

VII. EXPERIMENTS

In this section, we experimentally evaluate the efficiency and effectiveness of our proposed algorithms by conducting extensive experiments on six real-world datasets. The experimental settings are introduced in Section VII-A. The efficiency of our algorithms is measured in terms of the search area, running time, core pattern size and the number of constraint labels. We then conduct experiments to measure the effectiveness of our optimization techniques in Section VII-C. All experiments are run on a machine with an Intel i5 2.50 GHz CPU and 8 GB main memory (Ubuntu Linux 64-bit 14.04 LTS). All algorithms are compiled with Java (JDK v1.8.0_271).

A. EXPERIMENTAL SETTING

1) DATASETS

We use six real-world datasets in TABLE 2 for our experiment: (a) Citeseer [45]: Citeseer is a directed graph with 3.3K publications(nodes) which models the citation information between publications. Nodes represent publications and are labeled with a computer science area. Edges represent the citations between them; (b) GitHub [46]: A large social network of GitHub developers that was collected from the public API. Nodes are developers, and edges are mutual follower relationships between them. The node labels are extracted based on location, and e-mail address; (c) Brightkite [46]: A friendship network that was collected using public API. Edges represent friendship between users. The original graph does not contain labels, so we added labels to the nodes randomly. The number of distinct labels was set to 50 and

TABLE 3. Description of parameter.

Parameter	Values
Search area	15000, 20000, 25000, **30000**, 35000
Value k	1, 5, **10**, 15, 20
Node size	2, **3**, 4, 5, 6

the randomization follows a Gaussian distribution; (d) Mico: This dataset models Microsoft coauthorship information, which is a dense graph. Nodes represent authors and are labeled with the author's field of interest, edges represent collaboration between two authors. Mico was crawled by [7]; (e) Twitch [47]: A Twitch users social network collected from the public API, which models the mutual follower relationships between users. Each node has a single label representing the user's language, with EN(English) dominating; (f) Twitter [48]: This graph models the social news of Twitter. Nodes represent Twitter users, and edges represent an interaction between two users. The number of distinct labels was set to 100 and the randomization follows a Gaussian distribution. For fair comparison, each running time reported is averaged over 50 randomly generated core patterns. TABLE 2 shows the detailed information of these six datasets.

2) ALGORITHMS

We conduct extensive studies to compare the following 4 solutions.

- GsFPM-B: The basic solution (Algorithm 1), which utilizes the techniques from GRAMI and gSpan.
- GsFPM-R: This develops the search procedure in GsFPM-B through the NaR-Tree structure (Algorithm 3).
- GsFPM-S: This improves GsFPM-R by adapting the MNIE and MNIE Based Pattern Extension in Section V-B.
- GsFPM-M: Our mining framework (Algorithm 6) improves GsFPM-S by employing a lazy retrieval strategy.

3) PARAMETER SETTINGS

Table 3 shows the tested parameter settings, where the default values are highlighted in bold. In particular, to preserve the characteristics of each dataset, we choose 30000(km^2) as the default search area. Since the numbers of distinct labels are different among datasets, the label constraint depends on the specific dataset. To analyze the effect of each parameter, we vary one parameter while setting the rest to default values.

B. EFFICIENCY EVALUATION

Extensive experiments are conducted in this subsection to verify the efficiency of the proposed algorithms.

1) EFFECT OF SEARCH AREA

We first assess the efficiency of the solutions for the GsFPM problem by varying the search area. The running times for the four solutions are shown in Fig. 5. The results show that the running time clearly increases as the search area grows, especially in Mico. This is because the graph scale and density are much higher for a larger search area. It is obvious that GsFPM-B has the worst performance. We found that the performance of GsFPM-M is the best among the 4 solutions, with a speedup of 5× to 11× over the basic solution. Furthermore, the performance gap between GsFPM-S and GsFPM-R narrows when the search area grows for the Mico dataset. This is because the full graph of Mico is a dense graph in which many candidate patterns are valid [5] and cannot be pruned by Algorithm 4.

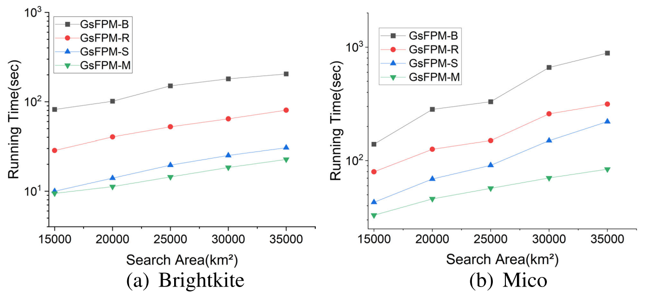


FIGURE 5. Effect of searching area.

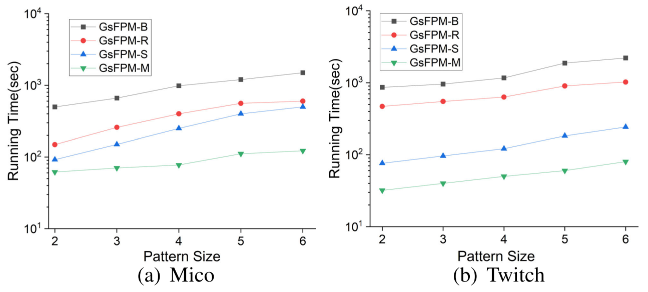


FIGURE 6. Effect of core pattern size.

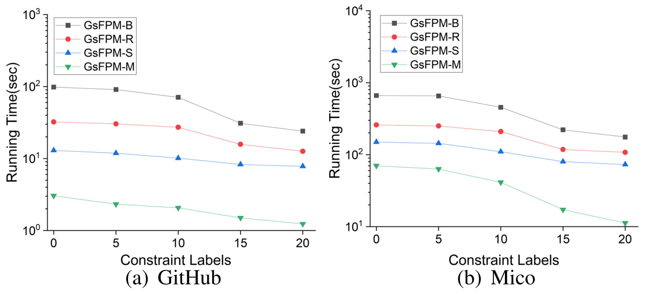


FIGURE 7. Effect of label constraint.

2) EFFECT OF K

We then evaluate the performance by adjusting the value of k. As illustrated in Fig. 8, the running time increases with increasing k. This is because a larger k means a larger result set and more candidate patterns. It can be seen that the performance of GsFPM-M is the best among the four solutions and achieves a speedup of 11× to 32×. We notice that GsFPM-R performs better in graphs with more distinct labels because

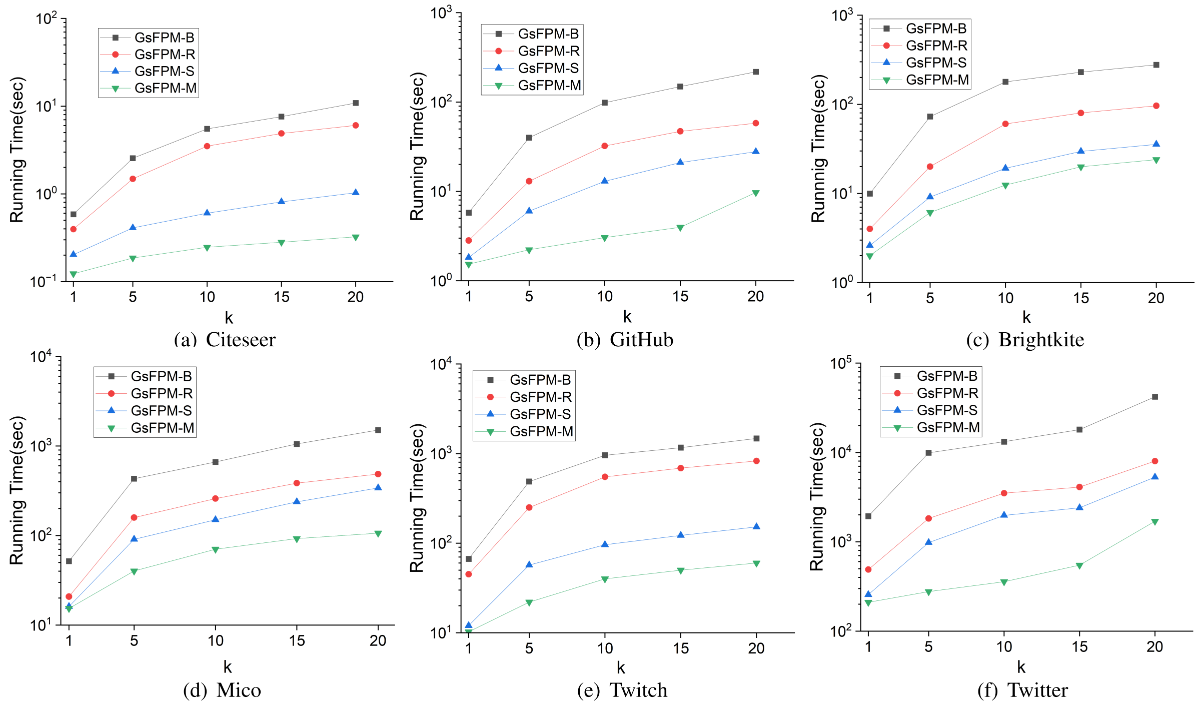


FIGURE 8. Effect of k.

NaR-Tree can retrieve smaller graphs by casting off labels that are not contained in the current pattern. As we mentioned before, dense graphs such as Mico contain almost every pattern's instance, which makes it difficult for our GsFPM-S solution to prune candidate patterns. In addition, we set users' languages as the node label in dataset Twitch. It is obvious that users in different languages have strong independence, so unlike Mico, a great number of candidate patterns can be pruned by GsFPM-S.

3) EFFECT OF CORE PATTERN SIZE

Next, we evaluate the performance by adjusting the size of the input core pattern. Fig. 6 shows the results on Mico and Twitch. The results show that the running time gradually increases as the size of core pattern grows. This is because a larger size of core pattern means more potential candidate patterns. As illustrated in Fig. 6, GsFPM-M has the lowest time cost among all four solutions and on both datasets again. For Mico and Twitch, it is at least 8× and 23× faster than GsFPM-B. Additionally, in Mico, GsFPM-S performs slightly better than GsFPM-R, as shown in Fig. 6(a). However, the performance gain of GsFPM-M over GsFPM-S is significant. The reason is that the lazy retrieval strategy adapted by GsFPM-M can reduce the number of times subgraphs are retrieved because many candidate patterns generated by GsFPM-S do not contain pattern elements that have not appeared before.

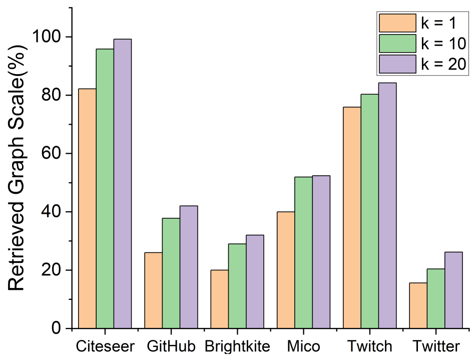


FIGURE 9. Effectiveness of retrieving optimization.

4) EFFECT OF LABEL CONSTRAINT

We then examine the influence of the number of constraint labels in GitHub and Mico. Both datasets contain nearly the same number of labels. As illustrated in Fig. 7, the running time gradually decreases as the number of constraint labels increases. This is because excluding the constraint labels will result in a smaller graph and fewer candidate patterns. However, GsFPM-R and GsFPM-S are more suitable for graphs with many labels, which leads to a widening of the performance gap between GsFPM-M and other solutions (i.e., GsFPM-B, GsFPM-R and GsFPM-S). The reason is that fewer labels will significantly reduce the number of times graphs are retrieved by our lazy retrieval strategy.

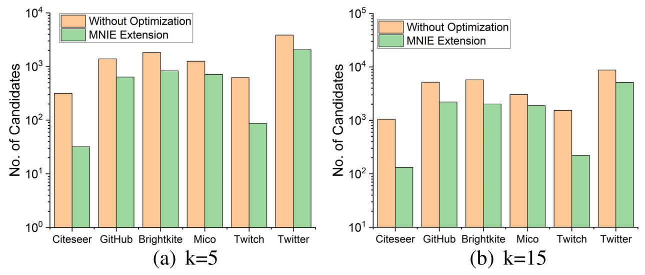


FIGURE 10. **Effectiveness of MNIE extension.**

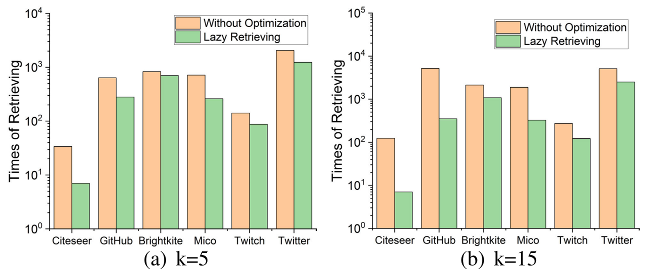


FIGURE 11. **Effectiveness of Lazy retrieving.**

C. EFFECTIVENESS EVALUATION

1) EFFECTIVENESS OF RETRIEVING OPTIMIZATION

Retrieving graphs aims to reduce the graph scale and further reduce the running time. In Fig. 9, we compare the scale of the retrieval graph with different datasets to show the effectiveness of retrieval optimization. The effectiveness of subgraph retrieval becomes worse with increasing k, because larger value of k leads to more candidate patterns. It is clear that datasets with more distinct labels achieve better retrieval performance, especially on BrightKite and Twitter. This is because with more distinct labels, more nodes in other labels will be removed in the retrieval process. As a result, the scale of the retrieval graph is reduced. However, retrieval performance in Twitch is only better than Citeseer, which only contains 6 distinct labels. The reason is that we set the user's language as the node label in Twitch, and the label EN dominates the label set. Therefore, other labels can only slightly impact the scale of the retrieved graph.

2) EFFECTIVENESS OF MNIE PATTERN EXTENSION

Fig. 10 illustrates the effectiveness of the MNIE-based pattern extension on six datasets. We compare the number of candidate patterns generated by two pattern enumeration algorithms, i.e., Algorithm 2 and Algorithm 4. As shown in Fig. 10, the number of candidate patterns is markedly reduced (by 1.75×-9.26×) using MNIE pattern extension. As we have mentioned before, MNIE pattern extension sets an upper bound for each pattern and reduces the running time by pruning invalid candidate patterns. We can observe that it works better in sparser datasets because many candidate patterns are valid and cannot be pruned in dense graphs such as Mico. However, MNIE pattern extension also performs well in Twitch, a dense graph. Because of the practical significance of Twitch, there is little overlap between users of different niche languages.

3) EFFECTIVENESS OF THE LAZY RETRIEVAL STRATEGY

Fig. 11 shows the retrieval times of the two different strategies for different values of k. As a result of lazy retrieval, there is a significant reduction in retrieval times on every dataset, especially on Citeseer, GitHub and Mico. Obviously, datasets with fewer label types usually have fewer retrieval times. This is because there are fewer label combinations to choose. Recall that the lazy retrieval strategy does not retrieve with pattern elements that already exist. As shown in Fig. 11, the lazy search strategy performs the worst on Brightkite. The reason is that Brightkite contains 50 label types, which leads to poor optimization.

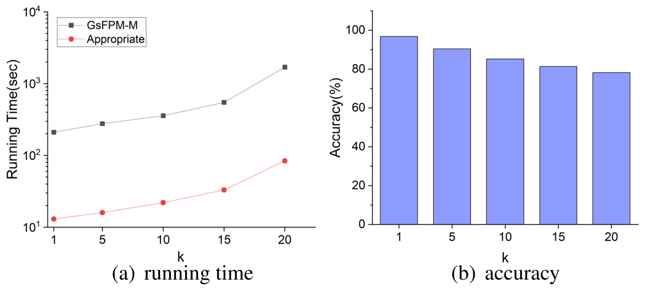


FIGURE 12. **Effectiveness of appropriate solution.**

4) EFFECTIVENESS OF THE APPROXIMATE ALGORITHM

Finally, we evaluate the effectiveness of our approximate solution. Fig. 12(a) and (b) show the running time and accuracy comparison of GsFPM-M and the edge sampling-based approximate solution on Twitter. The running time of the approximate solution achieves a speedup of 16.15× to 20.24× over GsFPM-M. For the accuracy, an approximate solution can achieve 78% accuracy on Twitter, which proves the effectiveness of our edge sampling-based solution.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we study the top-k FPM problem in geosocial networks. We introduce a novel framework for subgraph retrieval and MNIE-based mining with a series of optimizations. The NaR-Tree index structure is first proposed to improve mining efficiency and effectiveness. Together with the index, a MNIE-based candidate pattern extension strategy is devised to alleviate the exponentially increaing number of candidates. We also introduce subgraph lazy retrieval and edge-based graph sampling approaches to accelerate mining. Extensive experiments on real social networks are conducted to test the performance of the algorithms, and the experimental results strongly corroborate the superiority and effectiveness of our approaches.

The limitations of our work is the framework can only be applied to a single static graph, which is difficult to meet the timeliness required in the field of data analysis. With this observation, there are some opportunities for theoretical and experimental research. (1) Time dimension can be considered into research. Future work can focus on extending algorithms and index structures to dynamic graphs, which is desirable for many applications. (2) Design algorithmic frameworks for pattern mining that can be applied to other data types such as graph databases, trajectory networks, etc.

REFERENCES

[1] Y. Li, R. Chen, J. Xu, Q. Huang, H. Hu, and B. Choi, "Geo-social K-cover group queries for collaborative spatial computing," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 1510–1511.

[2] E. Wang, Y. Jiang, Y. Xu, L. Wang, and Y. Yang, "Spatial-temporal interval aware sequential POI recommendation," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 2086–2098.

[3] X. Chen, Y. Zhao, K. Zheng, B. Yang, and C. S. Jensen, "Influence-aware task assignment in spatial crowdsourcing," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*. IEEE, 2022, pp. 2141–2153.

[4] A. Prateek, A. Khan, A. Goyal, and S. Ranu, "Mining top-*k* pairs of correlated subgraphs in a large network," *Proc. VLDB Endowment*, vol. 13, no. 9, pp. 1511–1524, May 2020.

[5] J. Zeng, U. L. Hou, X. Yan, M. Han, and B. Tang, "Fast core-based top-*k* frequent pattern discovery in knowledge graphs," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Apr. 2021, pp. 936–947.

[6] X. Yan and J. Han, "GSpan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 721–724.

[7] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis, "GraMi: Frequent subgraph and pattern mining in a single large graph," *Proc. VLDB Endowment*, vol. 7, no. 7, pp. 517–528, Mar. 2014.

[8] J. Meng and Y.-C. Tu, "Flexible and feasible support measures for mining frequent patterns in large labeled graphs," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 391–402.

[9] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proc. IEEE Int. Conf. Data Mining*, Nov./Dec. 2001, pp. 313–320.

[10] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *Principles of Data Mining and Knowledge Discovery*, D. A. Zighed, J. Komorowski, and J. Żytkow, Eds. Berlin, Germany: Springer, 2000, pp. 13–23.

[11] R. Vijayalakshmi, N. Rethnasamy, J. F. Roddick, M. Thilaga, and P. Nirmala, "FP-GraphMiner—A fast frequent pattern mining algorithm for network graphs," *J. Graph Algorithms Appl.*, vol. 15, no. 6, pp. 753–776, 2011.

[12] Y. Li, L. Quan, R. Li, and D. Duan, "TGP: Mining top-*k* frequent closed graph pattern without minimum support," in *Proc. Int. Conf. Adv. Data Mining Appl.*, 2010, pp. 537–548.

[13] T. K. Saha and M. A. Hasan, "FS^3: A sampling based method for top-*k* frequent subgraph mining," *Stat. Anal. Data Mining, ASA Data Sci. J.*, vol. 8, no. 4, pp. 245–261, 2015.

[14] M. Kuramochi and G. Karypis, "Finding frequent patterns in a large sparse graph," *Data Mining Knowl. Discovery*, vol. 11, no. 3, pp. 243–271, Nov. 2005.

[15] V. Ingalalli, D. Ienco, and P. Poncelet, "Mining frequent subgraphs in multigraphs," *Inf. Sci.*, vols. 451–452, pp. 50–66, Jul. 2018.

[16] E. Abdelhamid, I. Abdelaziz, P. Kalnis, Z. Khayyat, and F. Jamour, "ScaleMine: Scalable parallel frequent subgraph mining in a single large graph," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Nov. 2016, pp. 716–727.

[17] F. Qiao, X. Zhang, P. Li, Z. Ding, S. Jia, and H. Wang, "A parallel approach for frequent subgraph mining in a single large graph using spark," *Appl. Sci.*, vol. 8, no. 2, p. 230, Feb. 2018.

[18] N. Vanetik, E. Gudes, and S. E. Shimony, "Computing frequent graph patterns from semistructured data," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, p. 458.

[19] B. Bringmann and S. Nijssen, "What is frequent in a single graph?" *Proc. 12th Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*, 2008, pp. 858–863.

[20] P. Rigaux, M. Scholl, and A. Voisard, *Spatial Databases: With Application to GIS*. San Mateo, CA, USA: Morgan Kaufmann, 2002.

[21] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. San Mateo, CA, USA: Morgan Kaufmann, 2006.

[22] D. B. Lomet, "Grow and post index trees: Role, techniques and future potential," in *Proc. 2nd Symp. Adv. Spatial Databases (SSD)*, Zurich, Switzerland. Springer, 2005, pp. 181–206.

[23] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.

[24] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R^*-tree: An efficient and robust access method for points and rectangles," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, May 1990, pp. 322–331.

[25] Y. Sun and M. Sarwat, "Riso-tree: An efficient and scalable index for spatial entities in graph database management systems," *ACM Trans. Spatial Algorithms Syst.*, vol. 7, no. 3, pp. 1–39, Sep. 2021.

[26] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Inf.*, vol. 4, no. 1, pp. 1–9, 1974.

[27] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[28] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Apr. 2008, pp. 656–665.

[29] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems," in *Proc. 19th Int. Conf. Sci. Stat. Database Manage. (SSDBM)*, Jul. 2007, p. 16.

[30] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D.-C. Lee, and X. Wang, "IR-tree: An efficient index for geographic document search," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 585–599, Apr. 2011.

[31] J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee, "An in-depth comparison of subgraph isomorphism algorithms in graph databases," *Proc. VLDB Endowment*, vol. 6, no. 2, pp. 133–144, Dec. 2012.

[32] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976.

[33] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

[34] P. Zhao and J. Han, "On graph query optimization in large networks," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 340–351, Sep. 2010.

[35] X. Ren and J. Wang, "Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs," *Proc. VLDB Endowment*, vol. 8, no. 5, pp. 617–628, Jan. 2015.

[36] M. Fiedler and C. Borgelt, "Subgraph support in a single large graph," in *Proc. 7th IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Oct. 2007, pp. 399–404.

[37] N. Armenatzoglou, S. Papadopoulos, and D. Papadias, "A general framework for geo-social query processing," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 913–924, 2013.

[38] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1984, pp. 47–57.

[39] Q. Zhu, H. Hu, C. Xu, J. Xu, and W.-C. Lee, "Geo-social group queries with minimum acquaintance constraints," *VLDB J.*, vol. 26, no. 5, pp. 709–727, Oct. 2017.

[40] G. Preti, G. De Francisci Morales, and M. Riondato, "MaNIACS: Approximate mining of frequent subgraph patterns through sampling," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1348–1358.

[41] P. Hu and W. C. Lau, "A survey and taxonomy of graph sampling," 2013, *arXiv:1308.5865*.

[42] R. Gao, H. Xu, P. Hu, and W. C. Lau, "Accelerating graph mining algorithms via uniform random edge sampling," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[43] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 1446–1455.

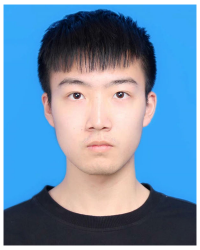
[44] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Trans. Comput. Syst.*, vol. 21, no. 3, pp. 270–313, 2003.

[45] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI Conf. Artif. Intell.*, vol. 29. New York, NY, USA, 2015, pp. 4292–4293.

[46] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *J. Complex Netw.*, vol. 9, no. 2, 2019, Art. no. cnab014.

[47] B. Rozemberczki and R. Sarkar, "Twitch gamers: A dataset for evaluating proximity preserving and structural role-based node embeddings," 2021, *arXiv:2101.03091*.

[48] R. Zafarani and H. Liu. (2009). *Social Computing Data Repository at ASU*. [Online]. Available: http://socialcomputing.asu.edu



CHANGBEN ZHOU received the B.E. degree in computer science and technology from Shaanxi University of Science and Technology, Xi'an, China, in 2020. Since 2020, he has been with the Laboratory of Internet and Network Security, Hangzhou Dianzi University, Hangzhou. His current research interests include data mining and social network analysis.

JIAN XU received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2004. He is currently a Full Professor with Hangzhou Dianzi University, Hangzhou. His research interests include data mining, machine learning, and social network analysis.

MING JIANG received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2004. He is currently a Full Professor with Hangzhou Dianzi University, Hangzhou. His research interests include natural language processing and semantic web.

DONGHANG TANG received the B.E. degree in software engineering from Henan University of Engineering, in 2021. Since 2021, he has been with the Laboratory of Internet and Network Security, Hangzhou Dianzi University, Hangzhou. His current research interests include data analysis and social network analysis.

SHENG WANG received the B.E. degree in the Internet of Things engineering from Information Engineering College, Hangzhou Dianzi University, Hangzhou, China, in 2020. Since 2021, he has been with the Laboratory of Internet and Network Security, Hangzhou Dianzi University. His current research interests include data analysis and sequence pattern mining.