Received December 20, 2019, accepted January 2, 2020, date of publication January 8, 2020, date of current version June 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964785

Searching Correlated Patterns From Graph Streams

MING JIN[1], MEI LI[2], YU ZHENG[3], AND LIANHUA CHI[4]
[1]School of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3010, Australia
[2]College of Information Engineering, Northwest A&F University, Yangling 712100, China
[3]Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia
[4]School of Engineering and Mathematical Sciences, La Trobe University, Bundoora, VIC 3086, Australia

Corresponding author: Mei Li (limei@nwsuaf.edu.cn)

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2452018147, in part by the Doctoral Scientific Research Foundation under Grant Z109021709, and in part by the Key Research and Development Program of Shaanxi under Grant 2019ZDLNY07-06-01.

ABSTRACT Mining the correlation has attracted widespread attention in the research community because of its advantages in understanding the dependencies between objects. In this paper, a correlated graph pattern searching scheme has been proposed, that is, provided with a query g as a structured pattern (*i.e.*, a graph), our algorithm is capable of retrieving the top-k graphs that most likely correlated with g. Traditional methods treat graph streams as static records, which is computational infeasible or ineffective because of the complexity of searching correlated patterns in a dynamic graph stream. In this paper, by relying on sliding windows to separate graph streams in chunks, we propose a Hoe-PGPL algorithm to handle the top-k correlated patterns searching from a dynamic perspective. Our algorithm applies Hoeffding bound and two-level (Sliding window level and local batch level) candidate inspection to discover potential graph candidates and determine the similarity of these candidates without double-checking the previous stream. Theoretical analysis shows that our method can guarantee the quality of the returned answers, and our experiments also present that Hoe-PGPL has an excellent performance with aspects of precision, recall, runtime, and resource consumption.

INDEX TERMS Data streams, graph streams, correlated structure pattern query, Pearson correlation coefficient.

I. INTRODUCTION

Correlation mining has attracted widespread attention in the research community because of its peculiarity and superiority in finding potential correlations between different patterns. Many research works have been conducted on correlated patterns mining in various applications, such as financial transaction databases [45], [68], [70], quantitative databases [11], and time-series data streams [40]. The research interest has recently extended to graph streams where elements are connected by relying on structural relations [26], [28]–[30]. A structural pattern is, therefore, equivalent to a graph represented as nodes that interconnected by edges.

The correlation between two structural patterns (graphs) is a measure of the similarity in their occurrence distribution. Provided with a query g and a graph stream, Correlated Graph Search (CGSearch) aims to find structural patterns with the Pearson correlation coefficient at least θ, which is a user predefined threshold. An example of this has been shown in Fig 1 where $\theta = 1$. Although this type of correlated subgraph pattern is found to be important in various graph representations, it expects the user to predefine a threshold to work. However, it is usually challenging to specify such a threshold, since a small value may lead to many correlation graphs, while a large value may lead to very few results. Recently, a version of the top-k algorithm has been proposed to identify salient correlation subgraphs with the highest correlation with g from the database [28], [43], [48].

However, these works [26], [28] are limited to static graph streams only. In reality, the data streams are continuously generated or change within/across various applications. For example, on a website such as eBay, there are millions of active users every day [5], and we can use graphs to represent their browsing records. On the other hand, their activities may change dynamically and different from other users so that the analysis of correlated graphs of user's browsing

The associate editor coordinating the review of this manuscript and approving it for publication was Shirui Pan.

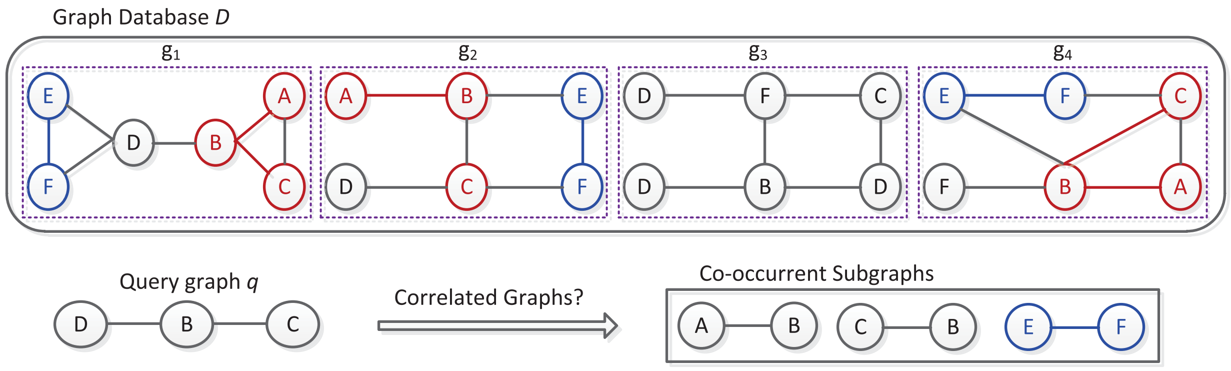

FIGURE 1. A example illustrated correlated graph query. Given a database $D = \{g_1, g_2, g_3, g_4\}$ and a query graph q, correlated graph query is able to discover some subgraphs which has similar occurrence distributions with q. For instance, the answer subgraph "E-F" is a co-occurrent subgraph of the query subgraph "A-B-C".

patterns will enable website owners to better understand user needs and improve their services. Another example of the dynamic graph stream can be found in the biomedical domain. The association of chemical compounds may change constantly during a chemical reaction because the structures of chemicals change dynamically. Analysis of these compounds (which can be regarded as correlated graphs) can find out some important substructures, which may help to discover new drugs [51]. In our experiments, we will also demonstrate a case study of structural patterns searching of scientific publications, which uses keywords and citation relationships to retrieve papers highly correlated to a query pattern (whereas traditional non-structural pattern search can only use keywords for query). All these examples demonstrate a clear need for searching structural patterns from graph streams.

Recently, Shirui and Xingquan proposed a CGStream-based method to query correlated graphs in a data stream scenario [42]. While CGStream can return the correlated graphs in an approximate but effective way, it expects users to define a threshold value θ, which indicates the minimum correlation scores between query g and retrieved patterns in graph streams. Notice that in real-world applications, such a threshold value is usually challenging to define and may differ significantly for different streams and query patterns. In order to make the structure pattern search more applicable to real-life usages, we propose to address top-k correlated graph queries for data streams in a dynamic way. In this article, we focus on finding the top-k correlated graphs, rather than retrieving correlation graphs with correlation values above a given threshold θ. The main challenges, in this case, are as follows:

- Graph correlation query combines NP-complete subgraph isomorphism examinations. In this case, storing and calculating the frequency and correlation of each subgraph is intricate in streams.
- Recalculating all correlations is very time-consuming because the correlation within graphs are constantly changing in streams over time.
- Streaming schemes require algorithms to return answers promptly.

Exhaustive search is a straightforward method to address our problem, which handles graph streams by using sliding window approaches, then the hidden correlations are examined by searching for the top-k correlated structural patterns based on the CGSearch [25], [26] or TopCor [28], [30] algorithm in each window. Although the exhaustive approach guarantees that the results are complete and accurate, it is computationally infeasible since it is time-consuming to search correlated structural patterns in different windows repeatedly. A compromise is to allow the system to retrieve the results in an approximate but highly reliable way, with much faster query speed than exhaustive search.

In this paper, we propose a correlated pattern searching algorithm based on Hoeffding bound [20] and data streams. This algorithm returns the top-k most correlated graphs of the query graph in a sliding window covering multiple consecutive batches of stream records. More precisely, Hoeffding bound [20] has been applied to each batch to determine a series of possible correlated graphs. Then a novel scheme named *global-local inspection* has been proposed to keep track of candidate information based on two different subschemes: Potential Local Lists (PLs) and Potential Global List (PG). Although the potential global list PG can be utilized to roughly estimate the candidate's real correlation value from a global perspective, the potential local lists PLs enable us to estimate the correlation value from a local perspective, which is more reliable. By carefully manipulating the PG and PLs, we can predict each candidate's actual correlation with a relatively small deviation. Theoretical analysis shows that the correlation value of each candidate in the PG is close to its real correlation so that the output quality can be guaranteed and bounded with aspects of the accuracy and recall of queries. Our experimental results in section 5 indicate that Hoe-PGPL is far more accurate and efficient when compared with an exhaustive search method with aspects of time, memory consumption, precision, recall.

The rest of the paper is structured as follows. Preliminaries and the problem definition are presented in Section 2. Our Hoeffding bound based algorithm is presented in Section 3. In Section 4, we provide a bound on the expectation of precision and recall. Experimental results are shown in Section 5, following by a case study of correlated graphs from DBLP graph streams in Section 6. In Section 7 and 8, we give a review of the related works and summarize the paper.

II. PRELIMINARIES AND PROBLEM DEFINITION

A. PRELIMINARIES

We mainly consider connected, labeled, and undirected graphs in this paper, where a typical graph is defined as $g = (V, E, \ell)$. In this representation, V, E, and ℓ are the vertices set, edge set, and labeling function, respectively. The labeling function aims to attach labels to an edge or a node in a graph. Given two graphs $g_1 = (V_1, E_1, \ell_1)$ and $g_2 = (V_2, E_2, \ell_2)$, a subgraph isomorphism from g_1 to g_2 is an injective function $f\colon V_1 \rightarrow V_2$, such that $\forall(u, v) \in E_1$, we have $(f(u), f(v)) \in E_2$, $\ell_1(u) = \ell_2(f(u))$, $\ell_1(v) = \ell_2(f(v))$, $\ell_1(u, v) = \ell_2(f(u), f(v))$. g_1 is a subgraph of g_2 if a subgraph isomorphism exists from g_1 to g_2, where subgraph isomorphism is a NP-complete problem [13].

For a set of graphs G_i, the projected set in terms of graph g denotes as $S_g^{G_i} = \bigcup\{g' | g \subseteq g', g' \in G_i\}$. The frequency F_g and support $SP(g)$ of the projected set is denoted as $F_g = |S_g^{G_i}|$ and $SP(g) = |S_g^{G_i}|/|G_i|$. For a set $\bullet$, its cardinality has been denoted as $|\bullet|$. Given g_1 and g_2 in G_i, their joint frequency is the number of graphs in G_i that have both g_1 and g_2, which denotes as $F_{g_1\,g_2} = |S_{g_1}^{G_i} \bigcap S_{g_2}^{G_i}|$. Accordingly, their joint support is $SP(g_1, g_2) = |S_{g_1}^{G_i} \bigcap S_{g_2}^{G_i}|/|G_i|$.

B. PROBLEM DEFINITION

Provided with a streaming graph G and a query g, we mainly focus on determining top-k correlated structural patterns (graphs) from G regards to g. We assume that graph data comes in batches, and a sliding window $D = \{G_{1+i-w}, G_{2+i-w}, \cdots, G_i\}$ is defined to cover a continuous area of the streaming graph since G describes a dynamic graph stream which changes continuously. In this case, a batch of graphs has been denoted as G_j, $i \geq j \geq 1 + i - w$, and the latest batch of streaming graphs has been denoted as G_i. After that, our task is searching for the top-k correlated patterns with g and the highest Pearson correlation in the most current w batches. Figure 2 shows a top-k correlated graph query with a query graph g and a window size w, which equals to 3.

Definition 1 (Pearson Correlation Coefficient): Provid with graphs g_1 and g_2, which individual and joint supports are defined as $SP(g_1)$, $SP(g_2)$, and $SP(g_1, g_2)$ over F graphs, respectively. The Pearson correlation coefficient $\phi(g_1, g_2)$ is denotes as [45], [61]:

$$\phi(g_1, g_2) = \frac{SP(g_1, g_2) - SP(g_1)SP(g_2)}{\sqrt{SP(g_1)(1 - SP(g_1))SP(g_2)(1 - SP(g_2))}} \tag{1}$$

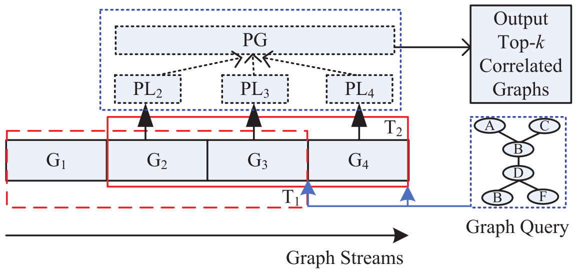

FIGURE 2. Searching top-k correlated patterns based on sliding windows over data streams. Batches G_1, G_2, and G_3 have been covered by a sliding window (red rectangular range) at timestamp T_1. G_4 is a newly arrived batch at timestamp T_2, which means that it is the most recent batch, the sliding window updates to cover G_2, G_3, and G_4 (solid red rectangle). We use two different lists (PG and PLs) to store the potential graphs, and results will be returned from the PG in each sliding window.

$\phi(g_1, g_2))$ is 0 when $SP(g_1)$ or $SP(g_2)$ is equal to 0 or 1, and the range of $\phi(g_1, g_2))$ is between 0 and 1 because in this paper, we only consider the positive correlations between graphs.

The Pearson correlation coefficient can transform into another representation over a set with F graphs in terms of frequency [68]:

$$\phi(g_1, g_2) = \frac{FF_{g_1 g_2} - F_{g_1}F_{g_2}}{\sqrt{F_{g_1}(F - F_{g_1})F_{g_2}(F - F_{g_2})}} \tag{2}$$

F_{g_1}, F_{g_2}, and $F_{g_1\,g_2}$ denote the number of graphs with g_1, g_2, and g_1 plus g_2 over F graphs, respectively. For clarity, the correlation between g_1 and g_2 is represented as $\phi_D(g_1, g_2)$ when they come from the entire window D; On the other hand, the correlation is $\phi_L(g_1, g_2)$ if these variables are collected from a local batch of graphs G_j.

III. PROPOSED METHOD

In a data stream scenario, we aim to return the top-k correlated subgraphs in each sliding window. A straightforward way is to apply CGSearch [25], [26] or TopCor [28], [30] algorithm to each sliding window whenever a new graph batch arrives. However, this is computationally ineffective because recomputing the correlation of each subgraph from scratch in the sliding window is expensive, especially for a large window size.

Another way is to have a set of possible candidates over streams, whenever a new batch of graphs flow in, we update the statistic frequency information (according to Eq. 2) for each candidate g in D (F_{g_1}, F_{g_2}, and $F_{g_1 g_2}$) instead of restarting to mine from the whole sliding window. Under such circumstances, determining what type of candidates should be discovered and how to maintain these candidates are of great significance. In this paper, Hoeffding bound [20] has been applied in each batch to elect a set of candidates, then we rely on a *global-local inspection* scheme to maintain these potential graphs with the PG and PLs. In this section, we first state the Hoeffding bound [20] for candidate generation and then introduce our algorithms with two levels of candidate lists.

A. HOEFFDING BOUND FOR CANDIDATE GENERATION

In [58], Rong *et. al* use a Chernoff bound to discover the top-k frequent itemsets over data stream, where they assume the presence of each item/itemset at each timestamp over streams is a random boolean variable (*i.e.*, if an item/itemset appears at a timestamp, the random variable is 1, otherwise 0). In our application, the correlation over a stream at each timestamp is a real number within range [0, 1], so we employ Hoeffding bound [20], which is also widely used in pieces of literature on algorithm ranking and data stream classification for candidate generation.

Suppose there is a series of independent drawing points $\{\phi_1, \phi_2, \cdots, \phi_n\}$, each point ϕ_i is in the range of $[a, b]$. The Hoeffding boundary [20] provides a certain probability that guarantees for the statistical estimation of the underlying data. More explicitly, the estimated mean $\bar{r} = \frac{1}{n}\sum_{i=}^{n} \phi_i$ if r is the expected value of these points. For any ϵ, Hoeffding bound states that [20], [43],

$$\Pr\{|r - \bar{r}| \geq \epsilon\} \leq 2e^{-\frac{2n\epsilon^2}{R^2}} \tag{3}$$

In above formula, n and R denote the number of accumulated observations and the range of each observation ϕ_i, respectively. If we let $\delta = 2e^{-\frac{2n\epsilon^2}{R^2}}$ be the right side of Eq.(3), the Hoeffding bound [20] indicates that the estimated average $\bar{r}$ exceeds $\pm\epsilon$ of r with a probability no greater than δ; Otherwise speaking, $\bar{r}$ is within the expected ϵ. From Eq.(3), we know that [42]

$$\epsilon = \sqrt{\frac{R^2 \ln(2/\delta)}{2n}} \tag{4}$$

In the problem settings of this paper, we mainly focus on positive correlations and fix the range of $R = 1$. We assume the streaming data arrives in batches, and the correlation is randomly distributed over data streams. For each batch, we measure the estimated mean $\bar{r} = \phi_L(g_1, g_2)$, and the correlation $\phi_D(g_1, g_2)$ in a sliding window is no less than $\phi_L(g_1, g_2) - \epsilon$ with probability $1 - \delta$. Generally speaking, we compute $\phi_L(g_1, g_2)$ in batches over $|G_i|$ graphs, and this result should equals to the average of $\phi_L(g_1, g_2)$.

In each batch of the data stream, a set of potential top-k correlated graphs can be mined by using the Hoeffding bound [20]. Specifically, instead of query from the whole batch, we aim to identify a series of candidates from the *projected database* $S_{g_2}^{G_i}$, which is a subset of the batch data G_i including all graphs containing query g_2. Because the size of $S_{g_2}^{G_i}$ is much smaller than the original batch of data, the employment of $S_{g_2}^{G_i}$ can greatly reduce the time consumption [26], [28], [30]. For each candidate graph g_1, we requires its correlation value is within a small range of the kth value, *i.e.*, $\phi_L(g_1, g_2) > \phi_L - 2\epsilon_s$. In this case, $\epsilon_s = \sqrt{\frac{R^2 \ln(2/\delta)}{2|G_i|}}$ and $\phi_L(g_1, g_2)$ denotes the kth correlation value (notice that candidates are sorted based on their correlation values in descending order) with query graph g_2. Specificly, the method in [28], [30] or CGSearch algorithm [25], [26]

TABLE 1. Notations with Respect to Hoeffding Bound for Candidate Generation.

ϕ_L	The k^{th} correlation value in a local batch of data
ϕ_D	The k^{th} correlation value in the sliding window
$\phi_L(g_1, g_2)$	The correlation of graph g_1 with g_2 in a local batch
$\phi_D(g_1, g_2)$	The correlation of graph g_1 with g_2 in the sliding window
$\epsilon_s = \sqrt{\frac{R^2 \ln(2/\delta)}{2\|G_i\|}}$	Error in a local batch
$\epsilon_w = \sqrt{\frac{R^2 \ln(2/\delta)}{2w\|G_i\|}}$	Error in the sliding window

can be used to find the kth correlation value. In this paper, we summarise the primarily used notations of Hoeffding bound in table 1 [20].

B. TWO LEVELS OF LISTS IN GLOBAL-LOCAL INSPECTION SCHEME

After finding the possible top-k correlated structural patterns in different batches, they need to be maintained carefully. Here we introduce two different lists to manipulate these patterns (graphs).

- **Potential Global List (PG):** For each candidate in sliding window D, its global frequency information (F_{g_1}, F_{g_2}, and $F_{g_1}g_2$) is stored in the PG where graphs are sorted based on their correlation values in descending order. The top-k correlated graph query may simply return the final answers by retrieving the top-k graphs in the PG. The frequency of each candidate should be updated accordingly whenever D is changed, *e.g.*, adding or excluding a data batch.
- **Potential Local Lists (PLs):** The local frequency information of each candidate (F_{g_1}, F_{g_2}, and $F_{g_1}g_2$) is recorded in a set of PLs where we apply a list PL_j, $j \in [1 + i - w, i]$ to keep the retrieved graphs from G_j.

To mine a graph stream effectively, we cannot keep track on all of the information about the data stream. Usually, the complexity is caused by a situational pattern τ, which is less significant in historical observations but may become significant in the future. However, the historical information τ may not be stored since τ is less significant in previous observations unless we are allowed to access the historical stream data, which is prohibitive. As a result, we define an emerging candidate pattern as follows.

Definition 2 (Emerging Candidate Patterns): Provide with a query graph g_1 and a sliding window D that covers w consecutive data chunks $D = \{G_{i-w}, G_{1+i-w}, \cdots, G_{i-1}\}$, where G_{i-1} is the most recent data chunk. Assume a new data batch G_i arrives, an emerging candidate pattern denotes a pattern τ, whose correlation value to g_1 in G_i is significant with respect to a predefined threshold (*i.e.*, $\phi_L(\tau, g_1) \geq \phi_L - 2\epsilon_s$), whereas τ does not exist in D's potential global candidate list.

Because an emerging pattern τ does not exist in the PG of the previous sliding window, τ's true correlated value to g_1

in a new sliding window $D' = \{G_{i-w+1}, \cdots, G_{i-1}, G_i\}$ will need to be estimated, which may, however, introduce bias. In the following, we first discuss some possible solutions and then present our *global-local inspection* scheme to handle this problem.

- **Up-to-date Batch only Estimation:** A straightforward way of estimating an emerging pattern τ's correlation to the query graph g_1 is to use τ's correlation in the most current batch ($\phi_L(\tau, q)$) as an estimation of the whole sliding window. As in our experiments, such a simple scheme is not accurate and may result in significant errors in the query results.
- **PG-only Estimation:** An alternative approach is to utilize the PG to estimate the correlation of candidates from a global perspective. Specifically, if an emerging pattern τ is not in the PG (*i.e.,* $\tau \notin PG$) but correlate with the most current batch (*i.e,* $\tau \in PL_i$), which means that in previous batches, $\phi_D(\tau, g_1)$ is comparatively low in the sliding window, so τ cannot be previously held in the PG. If a candidate $\tau \notin PG$ and τ_{min} denotes the graph with the minimum correlation value $\phi_D(\tau_{min}, g_1)$ in the PG, we know that in previous batches $\phi_D(\tau, g_1) < \phi_D(\tau_{min}, g_1)$. Therefore, τ can be estimated by using τ_{min}'s frequency information in previous batches. τ is usually a small value (not the correlated patterns in the current window) but may have the most significant correlation in previous blocks.
- **PLs-only Estimation:** We can also employ a set of PLs to calculate the emerging candidate pattern's frequency. Specifically, if a candidate $\tau \notin PG$, we can search over the PL_{1+i-w} to PL_{i-1} to estimate its frequency in previous batches. On the one hand, the frequency information in G_j can be obtained directly if $\tau \in PL_j$, $j \in [1 + i - W, i - 1]$. On the contrast, if $\tau \notin PL_j$, then the minimum correlation in PL_j is larger than its correlation in G_j. If τ_{min} denotes the graph in PL_j with the minimum correlation value, the frequency of τ in G_j can be estimated by using the frequency information of τ_{min}. By leveraging PLs, we are hopefully getting a more precise correlation estimation for each emerging candidate pattern.

Global-local inspection scheme: Because PG and PLs each has its unique advantages to estimate the correlation value from either global or local perspectives, our global-local inspection scheme integrates PLs and PG together, and the detailed procedures are illustrated in Algorithm 1.

Firstly, the estimated frequency for each candidate τ can be obtained by going through PL_{1+i-w} to PL_{i-1} from a local perspective, as shown in Algorithm 1 from steps 3 to 10, which minimizes the gap between the correlation values in $PL_j, j \in [1+i-w, i-1]$ and their actual values in G_j. Specifically, τ in batch G_j has been estimated by using the frequency of g_2' in step 11. The data structure of $list(F_\tau)$, $list(F_{g_2})$, and $list(F_{\tau g_2})$ in this step will be introduced in Sec. 3.3. When

Algorithm 1 *Merge*(PL_i, PG, τ)

```
1  $\tau_{min} = \arg\min_{g_0 \in PG} \phi_D(g_0, g_1)$;
2  for $\tau \in PL_i$ and $PG$ do
3      for $PL_j, j \in [1 + i - w, i - 1]$ do
4          if $\tau \in PL_j$ then
5              $g_2' = \tau$;
6          end
7          else
8              $g_2' = \arg\min_{g_2 \in PL_j} \phi_L(g_2, g_1)$;
9          end
10     end
11     Insert the statistics of $g_2'$ in batch $G_j$ to $list(F_\tau)$,
       $list(F_{\tau g_2})$, and $list(F_{g_2})$;
12     if $\phi_D(\tau_{min}, g_2) < \phi_D(\tau, g_2)$ then
13         Further correct the statistics of $\tau$;
14     end
15     Add $\tau$ into PG;
16 end
```

we get the frequency of τ from PLs, this information can be examined from a global perspective in the PG, as shown in Algorithm 1 from steps 12 to 14. If τ_{min} denotes the graph with the minimum correlation in the PG and a renewed correlation value $\phi_D(\tau, g_2)$ is greater than $\phi_D(\tau_{min}, g_2)$, we require $\phi_D(\tau, g_2) < \phi_D(\tau_{min}, g_2)$ from a global perspective so that a further correction of the frequency information of g_2 will be needed. In our implementation, we replace the frequency of τ with the frequency of τ_{min} in G_j if we find $\tau \notin PL_j, j \in [1+i-w, i-1]$. By doing this, we can add newly generated candidates into the PG and accurately estimate its real correlation with the query graph g_1. Consequently, the ranking of the PG can be more reliable, and the ground truth can also be returned in a more precise way.

C. HOE-PGPL ALGORITHM

Our Hoe-PGPL algorithm for continuous correlated subgraph queries from data streams is listed in Algorithm 2. The framework takes multiple parameters as inputs. To be more specific, G is a continuous graph stream; k specifies the number of returned graphs; w indicates the sliding windows size with aspects of a number of batches; parameter m controlling the capacity of the PG.

We first initialize the PG and PLs as empty sets in step 1, and the loop in step 2 represents a stream processing cycle. As long as new graph data arrives constantly, the stream processing cycle will repeat. When a new chunk G_i is collected, the most antiquated data batch will be discarded by Hoe-PGPL, which only applies G_i for the mining purposes. After this, in each sliding window, Hoe-PGPL will return the top-k correlated graphs A_{g_2}, and this process continues as long as the stream data continuously flow.

Algorithm 2 Hoe-PGPL Algorithm

```
Input :
        $G = \{G_1, \cdots, G_i, \cdots\}$: Graph streams;
        $k$ : number of returned correlated graphs;
        $w$ : size of the sliding window;
        $m$ : maximize number of candidates in PG;
1  $PG = \emptyset$, $PLs = \emptyset$;
2  while $G! = \emptyset$ do
3      $G_i \leftarrow$ a new graph batch;
4      $G \leftarrow G/G_i$ ;
5      $S_{g_1}^{G_i} = \bigcup\{g_2 | g_1 \subseteq g_2, g_2 \in G_i\}$, $F_{g_1} = |S_{g_1}^{G_i}|$;
6      Retrieve top-$k$ correlated graphs in $G_i$ from $S_{g_1}^{G_i}$;
7      Get the $k^{th}$ correlation value $\phi_L$ and error $\epsilon_s$;
8      $PL_i \leftarrow \bigcup\{g_2 | \phi_L(g_2, g_1) \geq \phi_L - 2\epsilon_s, g_2 \subseteq g_2', g_2' \in G_i\}$;
9      for $g_2 \in PG$ do
10         Increasing $list(F_{g_2})$, $list(F_{g_2 g_1})$, and $list(F_{g_1})$;
11         if $|list(F_{g_2})| > w$ then
12             Remove the first element from $list(F_{g_2})$, $list(F_{g_2 g_1})$, and $list(F_{g_1})$;
13         end
14     end
15     $Merge(PL_i, PG, \tau)$;
16     Recalculate top-$k$ correlated patterns based on PG and insert to $A_{g_2}$;
17     if $|PG| > m * k$ then
18         calculate the bound $\epsilon_w = \sqrt{\frac{R^2 \ln(2/\delta)}{2w|G_i|}}$;
19         Get the $k^{th}$ correlation value in PG, $\phi_D$;
20         $T \leftarrow \bigcup\{g_2 | \phi_D(g_2, g_1) < \phi_D - 2\epsilon_w, g_2 \in PG\}$;
21         $PG \leftarrow PG - T$;
22     end
23     Output $A_{g_2}$ in the current sliding window;
24     $i \leftarrow i + 1$;
25 end
```

The Hoe-PGPL algorithm has four parts: (i) Candidate generation based on Hoeffding bound [20] (steps 5 to 8); (ii) Adjusting the frequency information in the PG (steps 9 to14); (iii) Adding newly generated candidates to the PG (step 15). Notice that this is a crucial subprocedure, and it is shown in Algorithm 1; (iv) Pruning the PG (steps 16 to 22). As parts (i) and (iii) have been discussed in previous subsections, we focus on the other two parts in this subsection.

1) UPDATING THE FREQUENCY INFORMATION IN PG

In each batch and window D, we need to refresh the frequency information for each candidate $g_2 \in PG$, *i.e.*, F_{g_1}, F_{g_2} and $F_{g_1\ g_2}$. Steps 9 to 14 in Algorithm 2 list the maintenance of candidates that already exist in the PG. We first introduce our data structure to record the frequency information then describe the increasing and the decreasing procedures to maintain the PG. To check the calculation of the Pearson correlation value, each candidate in the PG is represented by a five-dimensional tuple as follows:

$$< g_2, F, list(F_{g_1}), list(F_{g_1 g_2}), list(F_{g_2}) >$$

In the above tuple, g_2 is the graph, F is the total number of stream graphs after g_2 is inserted into the global candidates set. $list(F_{g_1})$ is an array list, with each of its elements recording the frequency of g_1 in each batch of the sliding window. Similarly, $list(F_{g_1 g_2})$ and $list(F_{g_2})$ store information about the $F_{g_1 g_2}$ and F_{g_2} in each batch, respectively. The increasing and decreasing procedures are as follows:

Increasing Procedure: for each candidate $g_2 \in PG$, if it exists in the PL_i, we update its frequency information directly, otherwise, we search dataset $S_{g_1}^{G_i}$ to get $F_{g_2 g_1}$, and search the whole batch to get F_{g_2} information. All these frequency statistics are inserted to $list(F_{g_2})$, $list(F_{g_2 g_1})$, and $list(F_{g_1})$, respectively.

Decreasing Procedure: We need to decrease the frequency information for candidates in the PG when an antiquated data batch is deleted from the sliding window. As we store each candidate in a five-dimensional tuple, which helps facilitate the decreasing process in a fast way, *i.e.*, we only need to remove the first element in the array list of F_{g_2}, $F_{g_2 g_1}$, and F_{g_1} whenever a batch of the graph becomes outdated.

2) PRUNING PG

If $|PG|$ is becoming considerably large, it may be time-consuming to maintain and search from the PG, which will, in turn, require pruning for PG (steps 11 to 15 in Algorithm 2). If the number of candidates in the PG exceeds a predefined threshold, *i.e.*, $m*k$, Hoe-PGPL simply removes the candidates whose correlation value is less than $\phi_D - 2\epsilon_w$ to ensure that the maximum number of candidates in PG is $m*k$.

IV. PRECISION AND RECALL BOUND ANALYSIS

Because our algorithm is an approximation based method, in this section, we study its theoretical bound in terms of query precision and recall.

Suppose the target top-k correlated graphs are denoted as T_{g_2}, and the calculated graphs A_{g_2} are returned by Algorithm 2, then the **precision** and **recall** can be calculated with $|T_{g_2} \bigcap A_{g_2}|/|A_{g_2}|$ and $|T_{g_2} \bigcap A_{g_2}|/|T_{g_2}|$ [58], respectively.

Basically, we need to use two different estimations in our algorithm: the k^{th} correlation values in a local batch (denoted by ϕ_L), the sliding window (denoted by ϕ_D), and the estimated correlation for each emerging candidate τ. Either of estimation may contribute to the overall error.

Firstly, a bound has been setted on the possible bias of the k^{th} correlation value in the local batch (ϕ_L) and sliding window (phi_D) in Lemma 1, where we show that ϕ_L is likely to be close to ϕ_D. Next, in Lemma 2, we state that the accurate correlated graph's correlation value in a batch $\phi_L(g_2, g_1)$ will be larger than $\phi_D - \epsilon_s$ with a high probability. Combining Lemma 1 and 2, we derive Theorem 1 to attest that our algorithm will guarantee the storing of the accurate correlated graphs at a certain confidence level. In other words, a real top-k correlated graph will be stored in the PG and PL_i with a certain probability value. As our global-local inspection scheme can estimate each candidate accurately, we finally state our bound on precision and recall in Theorem 2 based on Theorem 1.

Lemma 1: Let the k^{th} correlation values in a local batch and the sliding window are ϕ_L and ϕ_D, respectively. The probability that $\phi_L \geq \phi_D + \epsilon_s$ is at most δ, where $\epsilon_s = \sqrt{\frac{R^2 \ln(2/\delta)}{2|G_i|}}$, and δ is a user specified confidence threshold.

Proof: From Hoeffding bound, we have $\phi_L(g_2, g_1) \geq \phi_D(g_2, g_1) + \epsilon_s$ with probability at most δ. Suppose graph g_2 is the k^{th} correlated graph in a local batch, *i.e.*, $\phi_L(g_2, g_1) = \phi_L$, then with probability at most δ,

$$\phi_L \geq \phi_D(g_2, g_1) + \epsilon_s \tag{5}$$

Case 1: if g_2 is also the k^{th} graph in sliding window, which means $\phi_D(g_2, g_1) = \phi_D$. Then the lemma holds.

Case 2: if g_2 is not one of the real top-k correlated graph in the sliding window, then $\phi_D(g_2, g_1) < \phi_D$. From Eq.(5), we know with probability $p' \leq \delta$, $\phi_L \geq \phi_D + \epsilon_s$.

Case 3: If g_2 is one of the top-k correlated graphs in the sliding window, then $\phi_D(g_2, g_1) \geq \phi_D$. As g_2 is the k^{th} correlated graph in a local batch, there should be a set of graphs G_A that for each graph $g_2' \in G_A$, $\phi_L(g_2', g_1) \geq \phi_L$. With probability at most $p_z \leq \delta$,

$$\phi_L(g_2', g_1) \geq \phi_D(g_2', g_1) + \epsilon_s \tag{6}$$

Case 3a: If the real top-k correlated graphs in the sliding window are the graphs in G_A, then there is a graph g_2' in G_A which is the k^{th} correlated graph in the sliding window, *i.e.*, $\phi_D(g_2', g_1) = \phi_D$. As $\phi_L(g_2', g_1) > \phi_L$ and $\phi_D(g_2', g_1) = \phi_D$. According to Eq (6), we know that, $p \leq p_z \leq \delta$ and $\phi_L \geq \phi_D + \epsilon_s$ with a probability p.

Case 3b: At least one graph g_2' in set G_A is not the real top-k correlated graphs, then $\phi_D(g_2', g_1) < \phi_D$. As $\phi_L(g_2', g_1) > \phi_L$ and $\phi_D(g_2', g_1) < \phi_D$, substituting in Eq (6), we know that, with probability p, $p \leq p_z \leq \delta$, $\phi_L \geq \phi_D + \epsilon_s$. □

Lemma 2: If g_2 is one of the true top-k correlated graph in the sliding window, the probability that $\phi_L(g_2, g_1) \geq \phi_D - \epsilon_s$ is at least $1 - \delta$.

Proof: If g_2 is the k^{th} correlated graph in the sliding window ($\phi_D(g_2, g_1) = \phi_D$), the lemma follows according to the Hoeffding bound ($\phi_L(g_2, g_1) \geq \phi_D(g_2, g_1) - \epsilon_s$ is at least $1 - \delta$). If g_2 is not the k^{th} correlated graph, then $\phi_D(g_2, g_1) \geq \phi_D$, the bound also follows. □

As the k^{th} correlation values in a local batch (ϕ_L) is close to the k^{th} correlation values in a sliding window (ϕ_D) (Lemma 1), and each true top-k graph's correlation value is greater than $\phi_D - \epsilon_s$ with a high probability (Lemma 2), combining these two inequations, we can guarantee that each true correlated graph will be stored by our algorithm with a high probability, which is shown in Theorem 1.

Theorem 1: If g_2 is one of the real top-k correlated graphs, then PL_i stores g_2 with probability higher than $(1 - \delta)^2$, and the PG stores g_2 with probability higher than $(1 - \delta)^2$.

Proof: According to Lemma 2, if g_2 is one of the top-k correlated graph in the sliding window, then the probability that $\phi_L(g_2, g_1) \geq \phi_D - \epsilon_s$ is at least $1 - \delta$. From Lemma 1, the probability that $\phi_L \geq \phi_D + \epsilon_s$ is at most δ. In another word, $\phi_D \geq \phi_L - \epsilon_s$ is at least $1 - \delta$. Thus, combining the two inequalities, we have $\phi_L(g_2, g_1) \geq \phi_D - \epsilon_s \geq \phi_L 2\epsilon_s$, and the probability is at least $(1 - \delta)^2$. Note that in each batch, we retrieve those graphs whose correlation is above $\phi_L - 2\epsilon_s$ and store them to the candidate list PL_i. In other words, if g_2 is one of the top-k correlated graphs in the sliding window, it is held by a local list PL_i at least $(1 - \delta)^2$. As we insert the newly discovered graphs into the PG, and we use a similar technologies to maintain the PG, which store g_2 with a probability also higher than $(1 - \delta)^2$. □

Since we design two levels of candidate lists to determine the real correlation of each newly emerging candidate, its estimated correlation will approach its actual correlation. So if a real top-k correlated graph is held in the PG, it will be returned. Then our algorithm has such a bound in terms of expectations of precision and recall, stated in Theorem 2.

Theorem 2: The expectations of recall and precision of our proposed Hoe-PGPL algorithm are both at least $(1 - \delta)^2$.

Proof: Because Hoe-PGPL returns k highest correlated graphs, in our circumstance, the precision and recall will be the same according to our definition. This is because $|A_{g_2}| = |T_{g_2}| = k$ [1]. Given that the correlation of each candidate g_2 with query graph g_1 in the PG approaches to the true correlation, the true answer will be returned only if it is held in the PG. From Theorem 1, we know that the probability of storing a true correlated graph is at least $(1 - \delta)^2$. In other words, each true top-k correlated graph will be returned at a probability at least $p_r = (1 - \delta)^2$. Suppose the number of true top-k correlated graphs returned by our algorithm is t, then t follows a binomial distribution in a total number of k, *i.e.*, $(t; k, p_r) = Pr(T = t) = C_k^t p_r^t (1 - p_r)^{(k-t)}$. The expectation of this binomial distribution is kp_r, variance is $kp_r(1 - p_r)$. Thus the expectation of recall is $kp_r/k = p_r = (1 - \delta)^2$. □

V. EXPERIMENTAL RESULTS

In this section, we illustrate our experiments, as well as the associated results, in detail. The streaming dataset we experimented is NCI Open Database Compounds [2], which contains a series of compound structures of illness data. The official dataset consists of around 250,000 graphs, which has been reduced to around 230,000 after the preprocessing (*e.g.* delete disconnected graphs).

We use two metrics, known as the precision and the recall, to measure the performance of our algorithm. If there are

[1]For simplicity, we assume $|A_{g_2}| = |T_{g_2}| = k$ in the analysis. In our implementation, $|A_{g_2}|$ may be larger than k, as there may be several correlation values equal to ϕ_D. Consequently, precision and recall are slightly different in our experiments.

[2]https://cactus.nci.nih.gov/ncidb2/download.html

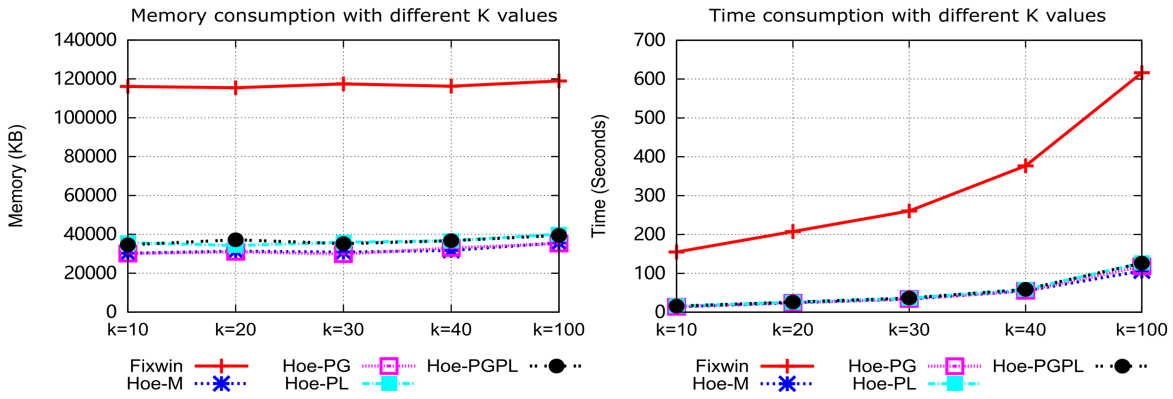

FIGURE 3. Memory and time consumption with various k values.

no false negatives in results provided by our algorithms, the recall should equal to 1; Similarly, if there are no false positives in the returned results, the precision should be 1.

Our algorithm has been compared with an exhaustive search strategy, known as **FixWin**, with aspects of memory consumption and running efficiency. FixWin is an accurate method that can return all correct results with no false positives and negatives. However, this approach is computationally infeasible because it demands a large amount of computing capacity and memory so that it is not an optimal choice for the majority of stream-based applications. For example, in our experiments, FixWin will abandon the existing work and turn to search the top-k correlated structural patterns from the beginning regardless of whether a new batch of graphs comes.

In addition to FixWin, we also compare our algorithm with three simple solutions to justify the effectiveness of our *global-local inspection* scheme. The three simple solutions represent three different approaches to use PG and PLs to estimate the correlation values of newly emerging candidates.

- **Estimate from the most current batch (denoted as Hoe-M):** A newly emerging candidate is inserted into the PG directly, and its correlation value in the most current batch is used as an estimation of the candidate's correlation in the sliding window.
- **Use PG-only (Hoe-PG):** Use the PG to estimate the correlation from a global perspective.
- **Use PLs-only (Hoe-PLs):** Use PLs to estimate the correlation from a local perspective.

There are fifty graphs we used in our experimented query graphs, and for each of them, it has been randomly selected and associated with a support value between 0.02 and 0.05 in data streams. If there are α batches of graphs in total, the average precision for a certain query can be calculated by $\overline{Precision} = \frac{1}{\alpha}\sum_{i=1}^{\alpha} P_i$. In this equation, P_i is the precision in a sliding window $D = \bigcup\{G_j | i \geq j \geq 1 + i - w\}$. Similarly, the average recall, time, and memory consumption are calculated in the same way. Note that for those fifty graphs, we need to calculate the above metrics and average them again as our final metrics.

For each batch, we rely on CGSearch [25], [26] to discover a set of top-k correlated graphs over data streams. In the first batch, we need to lower down the correlation threshold for CGSearch to get the top-k answers step-by-step and then insert these graphs into both PL_1 and PG. Suppose ϕ_D is the k^{th} correlation value in the PG, we can initially use ϕ_D as a correlation threshold to get the potential top-k correlated graphs in the next batch over data streams. By doing so, we can discover a series of candidates in different batches effectively. An alternative method is to employ TopCor [28], [30] to find the candidates within different batches. It is worth noting that both CGSearch and TopCor can be integrated into our Hoe-PGPL algorithm, and their selection not affects the fairness of comparison in our experiments.

The performance of our algorithm has been examined with different parameters, and the default values are $\delta = 0.03$, $k = 20$, $m = 4$, $w = 8$, and $|G_i| = 4000$.

A. EXPERIMENTS WITH DIFFERENT K VALUES

To examine the performance of our algorithm for various k values, we adjust this value from 10 to 100. Fig. 3 compares the runtime and memory consumption, and Fig. 4 shows the average precision and recall with different k values.

1) SYSTEM RUNTIME AND MEMORY CONSUMPTION

With regard to memory consumption, Fig. 3 tells us that FixWin consumes around three times more memory comparing to the Hoeffding bound based methods since FixWin has to keep the graph data within an entire window, but others methods only store the latest batch of data and candidates in PLs or/and PG. Among these Hoeffding bound based methods, Hoe-M and Hoe-PG utilize a little smaller memory comparing to Hoe-PLs and Hoe-PGPL. This is because Hoe-M and Hoe-PG only employ the PG to store the candidates, while Hoe-PLs and Hoe-PGPL additionally use PLs to estimate the correlation values.

Similarly, with regard to the runtime performance, Hoe-PGPL and other Hoeffding bound based strategies are several times faster than the exhaustive search, as shown in Fig. 3. Meanwhile, the time consumption of the Hoeffding bound based methods are very close to each other. This result

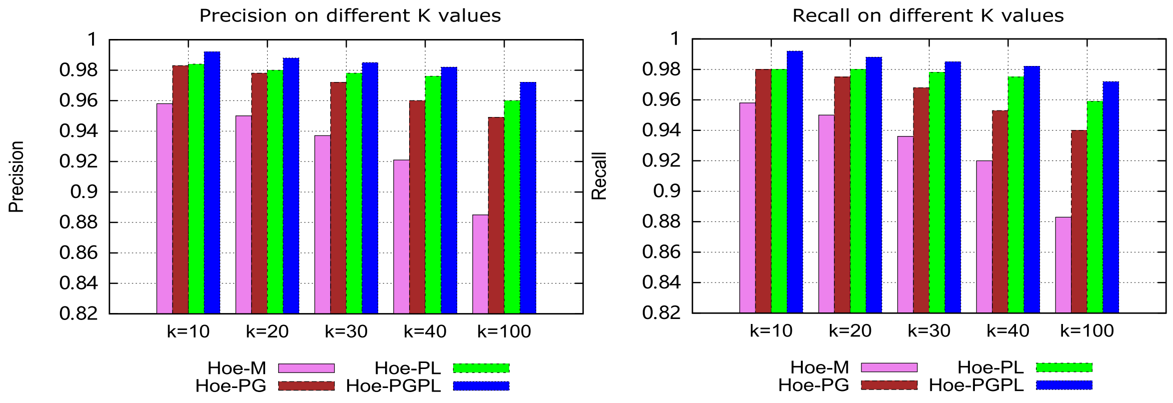

FIGURE 4. Precision and recall with various *k* values.

shows that including PLs (like Hoe-PLs and Hoe-PGPL do) will not significantly increase the time consumption compared to the methods using PG only (Hoe-M and Hoe-PG). This is because the majority runtime for all these methods is taken by the process of finding the top-k correlated structural patterns in batches by CGSearch [25], [26], which involves two time-consuming procedures, including mining frequent subgraphs by gspan [63] and checking candidate frequency to get F_{g_1}.

The results in Fig. 3 reveal that compared to exhaustive search method FixWin, Hoeffding bound based methods are several times more effective in terms of runtime and memory consumptions. In addition, all Hoeffding bound bases methods are close to each other with aspects of their memory and time consumptions because Hoeffding bound based methods are highly overlapping with each other, for clarity of presentation, we will mainly focus on the Hoe-PGPL in the following subsections.

2) QUERY PRECISION AND RECALL

In Fig. 4, we report the effectiveness of our global-local inspection scheme in terms of query precision and recall. From Fig. 4(A), it is clear that Hoe-M has the worst performance among these methods. This is because Hoe-M simply uses the correlation value in the most current batch as an estimation of the sliding window for each emerging candidate pattern. Because an emerging pattern's correlation value in the current batch may be relatively large, this estimation will introduce severe bias and affects the candidate ranking in the PG, which in turn results in inaccurate query results. In comparison, Hoe-PG, Hoe-PLs, and Hoe-PGPL integrate additional estimation techniques to take PG or/and PLs into consideration, thus can significantly improve the system performance. While Hoe-PG uses the PG only to estimate the correlation values roughly, Hoe-PLs makes use of a set of local candidate list PLs, which results in more accurate estimations than Hoe-PG. Because PG and PLs each have their unique advantage to estimate the correlation value from either global or local perspectives, Hoe-PGPL combines their strength and yields a better result in terms of query performance and recall. Since the correlation value estimated by Hoe-PGPL is much more reliable and more close to its true value, according to Theorem 2, the precision and recall are theoretically bounded. The result in Fig. 4 confirms that Hoe-PGPL is always better than other methods in both precision and recall.

Note that in the experiment with increasing k values, the precision and recall values drop for all algorithms. This is because with a large k, the ϕ_D (the k^{th} correlation value in PG) will be small, and the number of candidates in the range $[\phi_D-\epsilon, \phi_D+\epsilon]$ may increase dramatically compared to a large ϕ_D. For instance, when $\phi_D = 0.85$, there may be only ten candidates in a range of $[0.85-\epsilon_w, 0.85+\epsilon_w]$. When $\phi_D = 0.65$, there may be 235 of candidates in a range of $[0.65-\epsilon_w, 0.65+\epsilon_w]$. It is challenging to differentiate these candidates as their ranking order in the PG may change dramatically even if a small error estimation exists in the PG. Nevertheless, it is shown in Fig. 4 that, when Hoe-M drops dramatically in performance in $k = 100$, Hoe-PGPL only decreases slightly, which reveals the robustness of our Hoe-PGPL algorithm. As both precision and recall of Hoe-PGPL are approaching to 1, it shows that Hoe-PGPL is a highly accurate algorithm.

In summary, the above experiments and observations conclude that (i) Hoeffding bound based methods can significantly reduce the memory and time consumption comparing to the exhaustive search method. (ii) Hoe-PGPL scheme outperforms its peers in terms of precision and recall by compromising a tiny amount of runtime and memory as it considers not only the potential global candidates (PG) but also the potential local candidates (PLs).

B. EXPERIMENTS WITH VARIOUS BATCH SIZES $|G_i|$

Fig. 5 and Fig. 6 shows the performance of our algorithm with different batch sizes varies from $|G_i| = 3000, 5000, 7000$, to 9000.

From the results shown in Fig. 5 and Fig. 6, with the increasing of the batch size, the precision and recall are increasing for all the methods. This is because increasing the batch size will decrease ϵ_s ($\epsilon_s = \sqrt{\frac{R^2 \ln(2/\delta)}{2|G_i|}}$).

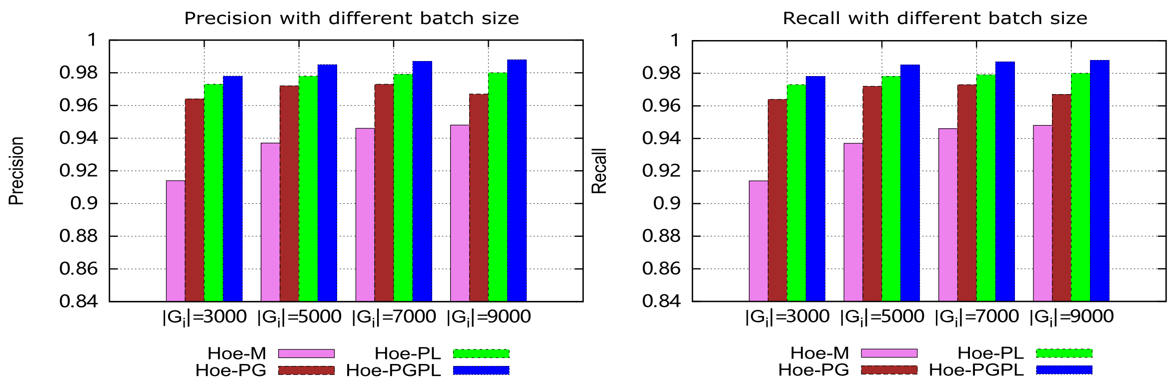

FIGURE 5. **Precision and recall with various batch size.**

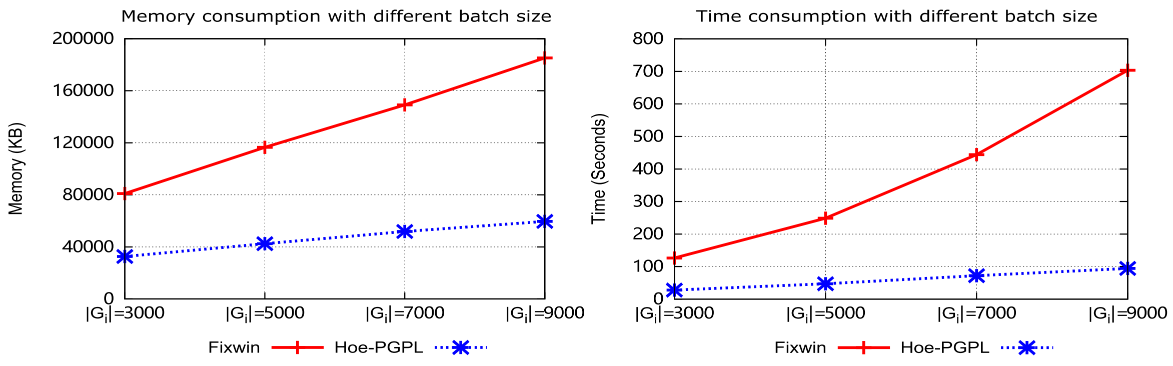

FIGURE 6. **Memory and time consumption with various batch size.**

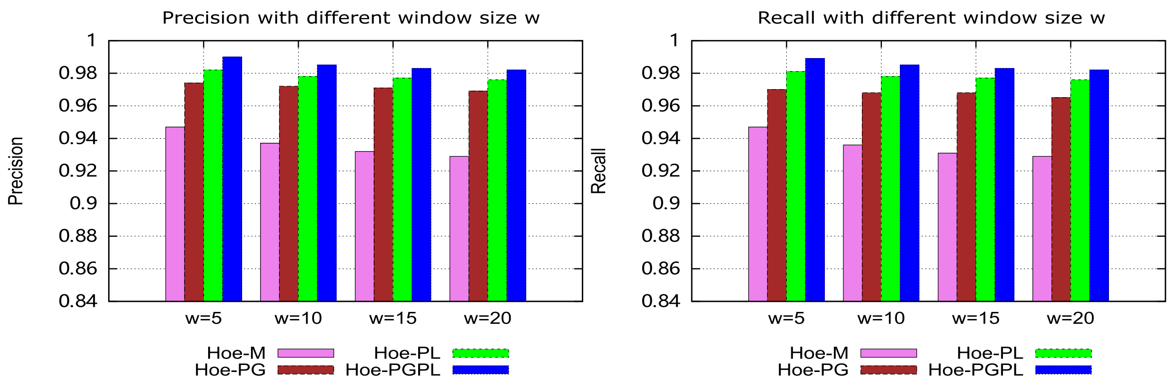

FIGURE 7. **Precision and recall with various window size *w*.**

In other words, the correlation of each candidate in PGs will be more close to its true correlation values, resulting in a better performance. However, the time and memory consumption will also climb because the number of graphs that need to be processed is increasing in each batch. Nonetheless, Fig. 5 and 6 show that Hoe-PGPL is much more effective comparing to FixWin in terms of memory and time complexity, which is also superior other methods with aspects of precision and recall.

C. EXPERIMENTS WITH VARIOUS SLIDING WINDOW SIZES

Fig. 5 and Fig. 6 shows the performance of our algorithm with different sliding window sizes w varies from 5 to 20.

From Fig. 7 and Fig. 8, it is obvious that with the increasing of the window size, all methods experience a decline of the query precision and recall values. This is because that the larger the window size, the correlation of each candidate in the PG will be more far away from its true correlation. Intuitively, $\phi_L(g_2, g_1)$ in a batch will be much nearer to $\phi_D(g_2, g_1)$ in a sliding window if $|G_i|$ is closer to the total number of graphs in the sliding window (i.e., $w|G_i|$). It is also shown in Fig. 8 that our Hoe-PGPL outperforms the exhaustive method significantly regards to system runtime and memory consumption. When increasing the window size, the runtime taken by Hoe-PGPL remains the same. However, the runtime of FixWin increase dramatically. For instance, with $w = 20$, Hoe-PGPL only needs about 50 seconds to return the answer,

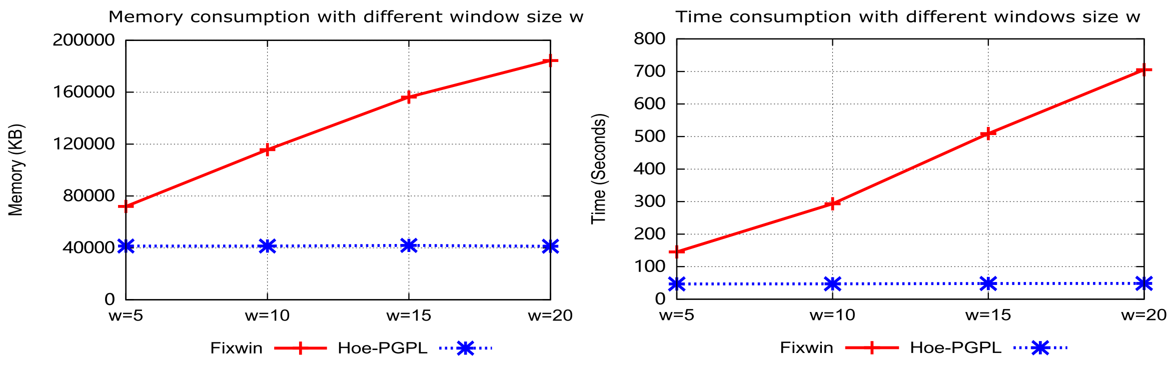

FIGURE 8. Memory and time consumption with various window size *w*.

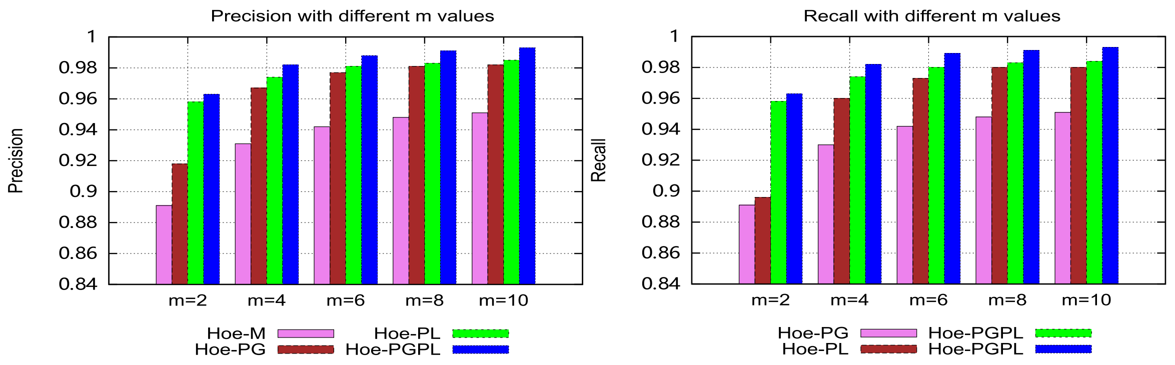

FIGURE 9. Precision and recall with various *m* value.

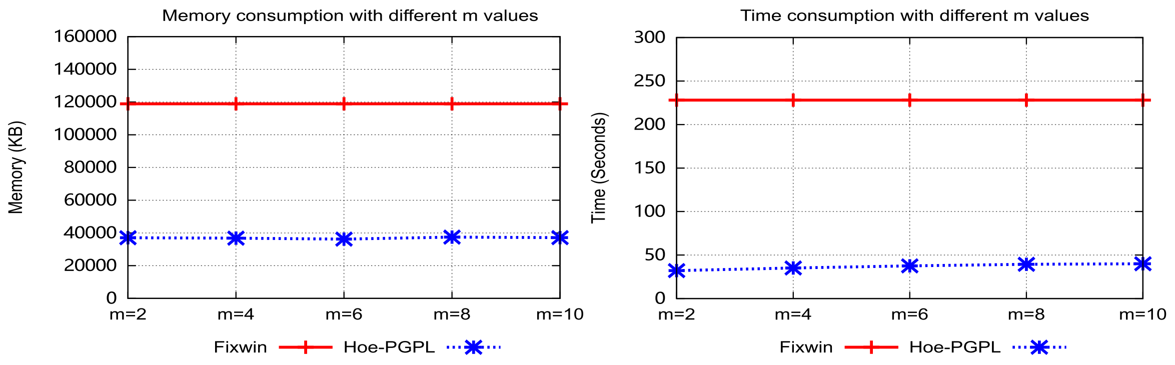

FIGURE 10. Memory and time consumption with various *m* value.

whereas FixWin requires about 700 seconds. In this case, Hoe-PGPL outperforms FixWin an order of magnitude.

D. EXPERIMENTS WITH DIFFERENT POTENTIAL CANDIDATE LIST SIZE

In Fig. 9 and 10, the performance of the algorithm with different sizes of potential candidate list (PG) are compared, where we vary the *m* values from 2 to 10, to change the size of the PG.

The results in Fig. 9 show that with a smaller *m* value, the Hoe-PG's performance deteriorates significantly, with only about 0.92 and 0.89 in precision and recall, respectively. In this case, Hoe-PG is significantly worse than the methods involving local PLs (Hoe-PG and Hoe-PGPL). Meanwhile, Hoe-PG is only slightly better than Hoe-M, which uses the most current batch to estimate the correlation values. This result demonstrates the power of employing local PLs into the algorithms. While *m* value continuously increases, the precision and recall values of all methods improve because the increasing of the PG size allows more candidates to be stored in the PG list. As a result, fewer candidates need to estimate their correlation values, which in turn produces more accurate results. In an extreme case, if the size of PG is unlimited, we will record information for all patterns in

TABLE 2. DBLP dataset used in experiments.

Categories	Descriptions	#Paper	#Graphs
DBDM	ACL, ECAI, AAAI, ICDT, SIGMOD, NIPS, COLT, KDD, KR, ACML, ACL, ICML, ECML, IJCAI, PAKDD	18870	10089

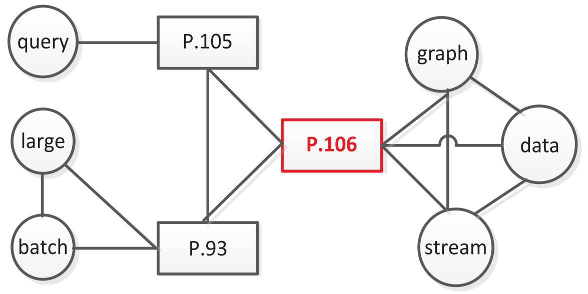

FIGURE 11. Graph representation of the paper P.106 (red-marked rectangular box) in the DBLP database. Rectangular boxes denote the identifier of papers, circles (nodes) denote various keywords, and those solid lines (edges) denote different citation relationships. Notice that for each rectangular box, the nodes are fully connected with each other.

the graph stream. The algorithm will then become 100% accurate, yet it requires unbounded storage space (for PG) and a significant amount of time for checking the PG list. In our experiments, when varying the value of m from 2 to 10, Hoe-PGPL consistently outperforms others in respect of precision and recall, and there is no notable increase in terms of the system runtime and memory consumptions.

VI. CASE STUDY ON DBLP GRAPH STREAM

In this section, we use the DBLP graph streams to analyze the correlated graphs discovered by our Hoe-PGPL algorithm.

DBLP Graph Stream: DBLP database [3] contains bibliographic data on the majority of computer science journals and proceedings [71]. For each instance in this database, it consists of several attributes, *i.e.*, article title, authors, abstract, and unique ID [39], [53]. In our case, we choose a series of conferences on Database and Data Mining conferences and summarised in Table 2. We use the paper published on these conferences as sources to build a graph stream.

In our case study, graphs are papers recorded in the DBLP database, keywords or unique identifiers of these papers are represented as nodes, and citation relationships across these papers, or the relationships between titles, authors, and keywords, are defined as edges. In detail, we define that (i) The unique identifier of a paper is denoted as a node; (ii) Every keyword in a paper is also denoted as a node; (iii) There is an edge between paper P_A and paper P_B if the citation relationship exists between P_A and P_B; (iv) Nodes are connected with each other, *i.e.*, given a paper, the unique identifier and keywords are linked with each other. Particularly, keywords in a given paper are fully connected in this scenario. Fig. 11 shows a graph example for a DBLP paper.

[3]http://arnetminer.org/citation

In our study, we use "data-streams" as a one-edge query graph, and retrieve top-20 correlated subgraphs from the whole graph streams. As correlated graph query can return subgraphs with similar distribution of the query graph, we expect that this structure pattern query can help obtain following meaningful results: (i) keywords frequently appear with "data" and "streams", which may indicate some data stream properties, challenges, research directions, and methods, etc. and (ii) some state-of-the-art literature related to data streams, which have been publicly cited in the DBLP benchmark.

Fig. 12 shows 10 correlated graphs returned by our Hoe-PGPL algorithms. From the correlated graphs, we can observe that many research papers related to data streams indeed consider the time-changing (g_1, g_6, and g_{10}), high speed (g_2, g_3, g_4, and g_{10}), or concept drifts (g_8 and g_9) of data streams. According to the results showing in g_5 and g_7, query over data streams is also a popular research direction in the past years. Subgraph g_9 shows that the classifier ensemble-based method is a popular method to handle data streams.

The results in Fig. 12 also show that some popularly cited papers in data stream research have been retrieved in our query results. For instance, P1 ([16]) is actually the first incremental learning algorithm to address data stream classification using decision tree methods. P2 ([23]) is the follow-up work of P1 ([16]), which intends to capture the time-changing properties of data streams. P4 ([57]) is the state-of-the-art classifier ensemble learning algorithm to handle concept drift over data streams. P3 ([15]) is another state-of-the-art algorithm addressing the query task from data streams. All these four papers (P1, P2, P3, and P4) represent some early works on data streams. According to Google Scholar statistics on March 17, 2013, the citations of these papers are 905, 828, 306, and 692, respectively.

The above case study indicates that correlated structure pattern search is indeed useful to discover some interesting patterns inside streams. When referring to academic publication streams (such as DBLP), it can retrieve research challenges, directions, methodologies, and state-of-the-art literature related to a given query. For chemical compounds analysis, it may help discover some fundamental properties or similar substructures of the provided query graph, which may eventually help discover new drugs.

VII. RELATED WORK

The work did in this paper is related to correlation mining, correlated structural patterns (graphs) mining, data stream/ graph stream mining, and correlated graph stream query.

Correlation mining has attracted many research interests and been widely studied in multiple domains, like biomedicine and economics. For example, correlation mining has been extensively examed and adopted in market-basket database [47], [59], [61], [68]. Although the methods in these works are designed to mine the correlation based on Pearson correlation coefficient, there are other strategies

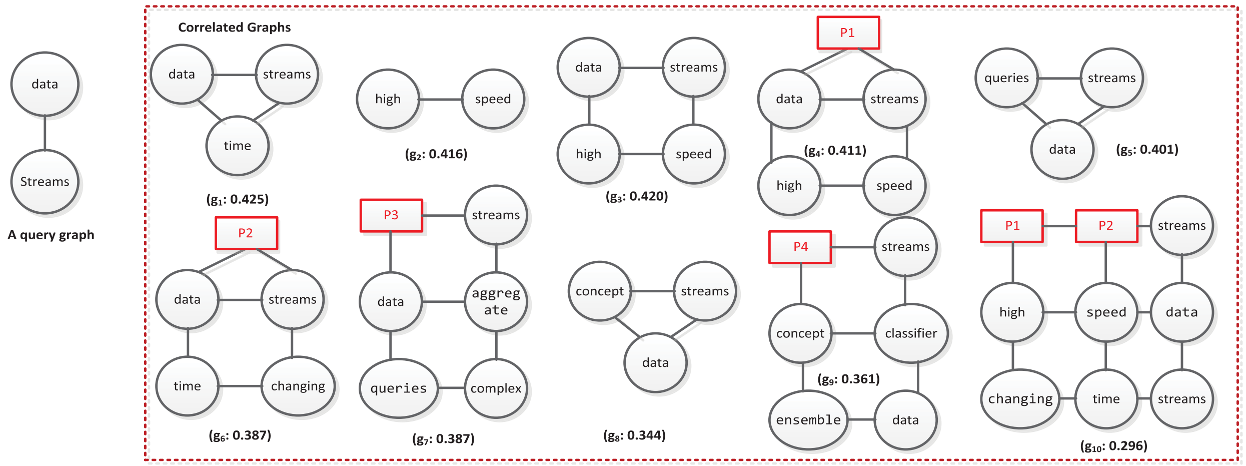

FIGURE 12. A query graph ("data-streams") with one edge and part of its top-20 correlated graphs discovered from the DBLP graph streams. The values in the brackets are the correlation values. Note that the keywords should be fully connected. For clear presentation, we omit some edges. P1, P2, P3, P4 are some reference papers, i.e., P1 - [16], P2 - [23], P3 - [15], P4 - [57].

that have been widely studied, for example, χ^2 test [7], h-confidence [60], and *m*-pattern measure [36].

As for graph databases, correlation mining is an active research topic. For example, CGSearch mines the correlated graphs by filtering the correlation above a predefined threshold [25], [26], and TopCor mines the correlated graphs by searching the graphs with the most significant correlation [28], [30]. These two algorithms handle a query by relying on a certain query g_1, but the works in [27], [32] did it in a different way: It mines all of the correlated structural pattern pairs in a graph database. In contrast, our algorithm is designed for dynamic graph databases, instead of limited to static databases.

Similar to graph databases, data stream has been an active research field in the past few years [17], [18], [23], [24], [41], [55], [67], [69]. In data stream environments, many methods exist to mine interesting patterns, *e.g.*, frequent items/itemsets [9], [34], [38], [58], [64], [65]. The study in [58] proposes to employ Chernoff bound to mine the top-*k* frequent itemsets from a data stream, whereas our research utilizes a Hoeffding bound to discover top-*k* correlated graphs and employs a global-local inspection scheme to ensure high accuracy for correlation estimation. For graph mining in a data stream scenario, some works on graph search [10], [56], frequent graph pattern mining [6], [46], graph classifications [2], [12], [33], [44], graph clustering [1], [54], [62], and outlier detections [3], [19], [66] have also been done.

For the correlated graph search from data stream settings, [42] previously proposed an algorithm CGStream to query the correlated graphs whose correlations are higher than a predefined value θ. CGStream treats query graphs as operators, which constantly query subgraph patterns related to itself when going through the graph stream. To reduce the computational cost, CGStream sets up a set of *outlooks (special time stamps)* over streams, and the mining task is only triggered at the outlooks. The Hoe-PGPL algorithm is different to CGStream algorithm in two aspects: (i) CGStream is a threshold based algorithm (*i.e.*, it retrieves graphs with correlation values higher than θ), and Hoe-PGPL is a top-*k* based algorithm (*i.e.*, it returns the top-*k* most correlated graphs only). (ii) CGStream stores all graphs in the sliding window to perform repeatedly mining at each outlook, while Hoe-PGPL only stores the potential candidates in two potential lists (PG and PLs) and discard all the graphs in the sliding window after processing.

VIII. CONCLUSION

In this paper, we have studied searching for the top-*k* correlated patterns by using a sliding window approach to cover graph streams over multiple consecutive batches. We believe that in a dynamic data stream scenario, simply exhaustively searching for the top-*k* correlated patterns requires storing the entire window of graph data and repeating the query process, which is computationally infeasible and memory consumptive. In our proposed method, each candidate's correlation in the PG can be accurately calculated by applying Hoeffding bound and a global-local inspection scheme that integrates with PLs and PG. Theoretical analysis proves that this method can guarantee the quality of retrieval results. On the other hand, experimental results show that, in terms of time and memory consumption, our algorithm is much more efficient comparing an exhaustive search method and has a relatively good performance with aspects of precision and recall.

REFERENCES

[1] C. C. Aggarwal, Y. Zhao, and P. S. Yu, "On clustering graph streams," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2010.
[2] C. C. Aggarwal, "On classification of graph streams," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2011.
[3] C. C. Aggarwal, Y. Zhao, and P. S. Yu, "Outlier detection in graph streams," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 399–409.
[4] A. R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley, 1999.
[5] A. Bifet, "Mining big data in real time," *Informatica*, vol. 37, no. 1, 2013.
[6] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà, "Mining frequent closed graphs on evolving data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 591–599.

[7] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," *ACM SIGMOD Rec.*, vol. 26, no. 2, pp. 265–276, Jun. 1997.

[8] F. Cajori, *A History of Mathematical Notations*, vols. 1–2. New York, NY, USA: Dover, 1993.

[9] H. Chen, L. Shu, J. Xia, and Q. Deng, "Mining frequent patterns in a varying-size sliding window of online transactional data streams," *Inf. Sci.*, vol. 215, pp. 15–36, Dec. 2012.

[10] L. Chen and C. Wang, "Continuous subgraph pattern search over certain and uncertain graph streams," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 8, pp. 1093–1109, Aug. 2010.

[11] C. Cheung and F. Li, "A quantitative correlation coefficient mining method for business intelligence in small and medium enterprises of trading business," *Expert Syst. Appl.*, vol. 39, no. 7, pp. 6279–6291, Jun. 2012.

[12] L. Chi, B. Li, and X. Zhu, "Fast graph stream classification using discriminative clique hashing," in *Proc 17th Pacific–Asia Conf Knowl. Discovery Data Mining (PAKDD)*, 2013, pp. 225–236.

[13] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory Comput. (STOC)*, 1971, pp. 151–158.

[14] X. Ding, X. Lian, L. Chen, and H. Jin, "Continuous monitoring of skylines over uncertain data streams," *Inf. Sci.*, vol. 184, no. 1, pp. 196–214, Feb. 2012.

[15] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Processing complex aggregate queries over data streams," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2002, pp. 61–72.

[16] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2000, pp. 71–80.

[17] D. M. Farid, L. Zhang, A. Hossain, C. M. Rahman, R. Strachan, G. Sexton, and K. Dahal, "An adaptive ensemble classifier for mining concept drifting data streams," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5895–5906, Nov. 2013.

[18] M. Garofalakis, J. Gehrke, and R. Rastogi, *Data Stream Management: Processing High-Speed Data Streams*. Berlin, Germany: Springer, 2016.

[19] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[20] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc.*, vol. 58, no. 301, pp. 13–30, Mar. 1963.

[21] N. Homem and J. P. Carvalho, "Finding top-k elements in data streams," *Inf. Sci.*, vol. 180, no. 24, pp. 4958–4974, Dec. 2010.

[22] L. J. Hubert, "Matching models in the analysis of cross-classifications," *Psychometrika*, vol. 44, no. 1, pp. 21–41, Mar. 1979.

[23] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2001, pp. 97–106.

[24] A. U. Kamath, A. Mahalingam, and J. H. Brauker, "Systems and methods for replacing signal data artifacts in a glucose sensor data stream," U.S. Patent 8 260 393, Sep. 4, 2012.

[25] K. Kamfa, M. Y. Waziri, M. Mamat, M. A. Mohamed, and P. Liza, "A new modified three term CG search direction for solving unconstrained optimization problems," *J. Adv. Res. Model. Simul.*, vol. 1, no. 1, pp. 23–30, 2018.

[26] Y. Ke, J. Cheng, and W. Ng, "Correlation search in graph databases," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2007, pp. 390–399.

[27] Y. Ke, J. Cheng, and J. X. Yu, "Efficient discovery of frequent correlated subgraph pairs," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 239–248.

[28] Y. Ke, J. Cheng, and J. X. Yu, "Top-k correlative graph mining," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2009, pp. 1038–1049.

[29] B. Kolleri, "Event correlation and association using a graph database," U.S. Patent App. 15/256 416, Mar. 8, 2018.

[30] G. S. Latsiou and A. N. Papadopoulos, "Incremental discovery of top-k correlative subgraphs," in *Proc. 15th Panhellenic Conf. Inform.*, Sep. 2011, pp. 56–60.

[31] C. Lee, O. Zaiane, H. Park, J. Huang, and R. Greiner, "Clustering high dimensional data: A graph-based relaxed optimization approach," *Inf. Sci.*, vol. 178, no. 23, pp. 4501–4511, Dec. 2008.

[32] M. Letsios, O. D. Balalau, M. Danisch, E. Orsini, and M. Sozio, "Finding heaviest k-subgraphs and events in social media," in *Proc. IEEE 16th Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2016, pp. 113–120.

[33] B. Li, X. Zhu, L. Chi, and C. Zhang, "Nested subtree hash kernels for large-scale graph classification over streams," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 399–408.

[34] N. Li, W. Qardaji, D. Su, and J. Cao, "PrivBasis: Frequent itemset mining with differential privacy," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1340–1351, Jul. 2012.

[35] C. Liang, Y. Zhang, P. Shi, and Z. Hu, "Learning very fast decision tree from uncertain data streams with positive and unlabeled samples," *Inf. Sci.*, vol. 213, pp. 50–67, Dec. 2012.

[36] S. Ma and J. Hellerstein, "Mining mutually dependent patterns," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2002, pp. 409–416.

[37] O. Maron and W. Andrew Moore, "Hoeffding races: Accelerating model selection search for classification and function approximation," in *Proc. NIPS*, 1993, pp. 59–66.

[38] S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data," in *Proc. IEEE Int. Conf. Big Data*, Oct. 2013, pp. 111–118.

[39] C. Moreira, P. Calado, and B. Martins, "Learning to rank academic experts in the DBLP dataset," *Expert Syst.*, vol. 32, no. 4, pp. 477–493, Aug. 2015.

[40] A. Mueen, S. Nath, and J. Liu, "Fast approximate correlation for massive time-series data," in *Proc. Int. Conf. Manage. Data SIGMOD*, 2010, pp. 171–182.

[41] S. Pan, Y. Zhang, and X. Li, "Dynamic classifier ensemble for positive unlabeled text stream classification," *Knowl. Inf. Syst.*, vol. 33, no. 2, pp. 267–287, Nov. 2012.

[42] S. Pan and X. Zhu, "CGStream: Continuous correlated graph query for data streams," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 1183–1192.

[43] S. Pan and X. Zhu, "Continuous top-k query for graph streams," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM*, 2012, pp. 2659–2662.

[44] S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Graph stream classification using labeled and unlabeled graphs," in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013.

[45] P. Sedgwick, "Pearson's correlation coefficient," *Bmj*, vol. 345, p. e4483, 2012.

[46] E. Shen and T. Yu, "Mining frequent graph patterns with differential privacy," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2013, pp. 545–553.

[47] Z. Shen, M. A. Cheema, X. Lin, W. Zhang, and H. Wang, "Efficiently monitoring top-k pairs over sliding windows," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 798–809.

[48] K. Shin, T. Eliassi-Rad, and C. Faloutsos, "CoreScope: Graph mining using k-core analysis—Patterns, anomalies and algorithms," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 469–478.

[49] J. G. Skellam, "The frequency distribution of the difference between two Poisson variates belonging to different populations," *J. Roy. Stat. Soc. A*, vol. 109, no. 3, p. 296, 1946.

[50] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2001, pp. 377–382.

[51] I. Takigawa and H. Mamitsuka, "Graph mining: Procedure, application to drug discovery and recent advances," *Drug Discovery Today*, vol. 18, nos. 1–2, pp. 50–57, Jan. 2013.

[52] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, "Sliding window-based frequent pattern mining over data streams," *Inf. Sci.*, vol. 179, no. 22, pp. 3843–3865, Nov. 2009.

[53] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 990–998.

[54] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014.

[55] D. V. Wie and P. J. Brody, "Automated real-time data stream switching in a shared virtual area communication environment," U.S. Patent 7 844 724, Nov. 30, 2010.

[56] C. Wang and L. Chen, "Continuous subgraph pattern search over graph streams," in *Proc. ICDE*, 2009, pp. 393–404.

[57] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2003, pp. 226–235.

[58] R. C.-W. Wong and A. W.-C. Fu, "Mining top-K frequent itemsets from data streams," *Data Mining Knowl Discovery*, vol. 13, no. 2, pp. 193–217, Sep. 2006.

[59] H. Xiong, M. Brodie, and S. Ma, "TOP-COP: Mining TOP-K strongly correlated Pairs in large databases," in *Proc. 6th Int. Conf. Data Mining (ICDM)*, Dec. 2006, pp. 1162–1166.

[60] H. Xiong, P.-N. Tan, and V. Kumar, "Hyperclique pattern discovery," *Data Mining Knowl. Discovery*, vol. 13, no. 2, pp. 219–242, Sep. 2006.

[61] H. Xiong, S. Shekhar, P.-N. Tan, and V. Kumar, "Exploiting a support-based upper bound of Pearson's correlation coefficient for efficiently identifying strongly correlated pairs," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2004, pp. 334–343.
[62] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2012, pp. 505–516.
[63] X. Yan and J. Han, "GSpan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, Jun. 2003, pp. 721–724.
[64] J. Yu, Z. Chong, H. Lu, Z. Zhang, and A. Zhou, "A false negative approach to mining frequent itemsets from high speed transactional data streams," *Inf. Sci.*, vol. 176, no. 14, pp. 1986–2015, Jul. 2006.
[65] J. Yu, Z. Chong, H. Lu, and A. Zhou, "False positive or false negativemining frequent itemsets from high speed transactional data streams," in *Proc. 13th Int. Conf. Very Large Data Bases*, vol. 30, 2004, pp. 204–215.
[66] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, "On anomalous hotspot discovery in graph streams," in *Proc. IEEE 13th Int. Conf. Data Mining*, Dec. 2013, pp. 1271–1276.
[67] P. Zhang, J. Li, P. Wang, B. J. Gao, X. Zhu, and L. Guo, "Enabling fast prediction for ensemble models on data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 177–185.
[68] W. Zhou and H. Xiong, "Volatile correlation computation: A checkpoint view," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 848–856.
[69] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 6, pp. 1607–1621, Dec. 2010.
[70] Y. Zhu, L. Qin, J. X. Yu, and H. Cheng, "Finding top-k similar graphs in graph databases," in *Proc. 15th Int. Conf. Extending Database Technol. (EDBT)*, 2012, pp. 456–467.
[71] O. R. Zaiane, J. Chen, and R. Goebel, "DBconnect: Mining research community on DBLP data," in *Proc. 9th WebKDD 1st SNA-KDD Workshop Web Mining Social Netw. Anal. (WebKDD/SNA-KDD)*, 2007, pp. 74–81.

MING JIN received the master's degree in information technology from the University of Melbourne, Parkville, Melbourne, Australia, in 2019. Since November 2018, he has been working in the field of industrial data mining and software development. His researches focus on time series analysis, graph neural networks (GNNs), data mining, and machine learning.

MEI LI received the Ph.D. degree from Northwest A&F University, in 2016. She is currently an Associate Professor with Northwest A&F University. Her research interests include data mining and machine learning.

YU ZHENG received the B.S. and M.S. degrees in computer science from Northwest A&F University, China, in 2008 and 2011, respectively. She is currently a Research Assistant with Monash University. Her research interests include image classification, data mining, and machine learning.

LIANHUA CHI received the dual Ph.D. degrees in computer science from the University of Technology Sydney, Australia, and the Huazhong University of Science and Technology, Wuhan, China, in 2015. She was a Postdoctoral Research Scientist with the IBM Research Melbourne. She has been a Lecturer of computer science with La Trobe University, since 2018. Her current research interests include graph stream data mining and big data hashing. She was a recipient of the Best Paper Award in PAKDD, in 2013.