World Wide Web
https://doi.org/10.1007/s11280-019-00686-w

Mining top-k sequential patterns in transaction database graphs

A new challenging problem and a sampling-based approach

Mingtao Lei[1] · Lingyang Chu[2] · Zhefeng Wang[3] · Jian Pei[2] · Caifeng He[4] · Xi Zhang[1] · Binxing Fang[1]

Received: 25 September 2018 / Revised: 15 February 2019 / Accepted: 25 April 2019 /
Published online: 03 May 2019

Abstract
In many real world networks, a vertex is usually associated with a transaction database that comprehensively describes the behaviour of the vertex. A typical example is a social network, where the behaviours of every user are depicted by a transaction database that stores her daily posted contents. Specifically, a transaction database consists of a collection of *transactions*, where each transaction corresponds to a piece of tweet. For each transaction, it consists of a set of items, where each item may correspond to a keyword or a piece of video clip contained in this tweet. To model such type of scenario, we propose the novel notion of the *transaction database graph*, where each vertex is associated with a transaction database. Every path of the graph is a sequence of vertices that induces multiple sequences of transactions. The sequences of transactions induced by all of the paths in the graph form an extremely large sequence database. Finding frequent sequential patterns from such sequence database discovers interesting subsequences that frequently appear in many paths of the network. Our goal is to find the top-k frequent sequential patterns in the sequence database induced from a transaction database graph. However, it is challenging since the sequence database induced by a transaction database graph is too large to be explicitly induced and stored, and finding the top-k frequent sequential patterns is #P-hard. To tackle this problem, we propose an efficient two-step sampling algorithm that approximates the top-k frequent sequential patterns with the provable quality guarantee. Extensive experimental results on synthetic and real-world data sets demonstrate the effectiveness and efficiency of our method.

Keywords Graph · Sequential pattern mining · Sampling · Approximation

✉ Xi Zhang
zhangx@bupt.edu.cn

Extended author information available on the last page of the article.

1 Introduction

Graphs and networks are popularly used to model advanced applications, such as social network analysis and communication network fault detection. More often than not, rich data exists in such graph and network applications. Consequently, graphs and networks in such applications have to be enriched by capturing more information in vertices and edges, such as labeled graphs [33] and attributed graphs [18], where in a labeled graph each vertex is associated with a unique label, and in an attributed graph each vertex is associated with an attribute vector. Mining various types of patterns in labeled graphs and attributed graphs has enjoyed interesting applications [9, 13, 30].

In some advanced graph applications, a vertex may contain much more information than just a label or an attribute vector. Such rich information on each vertex more often than not is better captured by a transaction database. For example, in content-rich social networks, such as Twitter, YouTube, WeChat and DBLP, a vertex modelling a user can be associated with a transaction database that stores multiple transactions of contents, where each transaction stores a tweet, a video clip, a post or a publication. As another example, in a road network of point of interests (POIs), each vertex representing a POI can be associated with a transaction database of visitor comments, where each transaction stores the keywords of one visitor comment.

We also give an example in communication networks, where each communication node is associated with an error log. When network faults occur, many communication nodes will produce error messages collected by their error logs. One specific error in one node may affect a related node and make it produce error messages too. Obviously, these error logs cannot merely be represented as labels or attribute vectors as they have very rich contents. To address this issue, a novel graph structure is required. In particular, a vertex represents a communication node, and a directed edge between two vertices indicates the data transmission. Each vertex can be associated with a transaction database representing the error log, and each transaction in the transaction database consists of history error codes.

To better model such graphs with rich information in vertices, we propose a novel notion of *transaction database graph*, where each vertex is associated with a transaction database. Comparing with the label and the attribute vector on each vertex, the transaction database associated with each vertex naturally and concisely captures much more valuable information, such as the co-occurrences of items and the frequencies of patterns (i.e., set of items).

Finding interesting patterns in a transaction database graph with rich contents in vertices may lead to informative analytic results. For example, by mining sequential patterns (i.e., frequent subsequences) from all possible random walks in a social network with rich contents in vertices, one may find interesting interaction patterns among users. Specifically, in the first example of a social network given above, where each vertex, as a user, is associated with a transaction database storing the daily posts composed by the user, and each transaction stores a set of keywords of one post. In this transaction database graph, a sequential pattern $\langle$(AI, Alpha Go) $\rightarrow$ (deep learning, deepmind) $\rightarrow$ (AI, medical doctor)$\rangle$ indicates that it happens frequently that a user who writes about AI and Alpha Go connects to another user who writes about deep learning and deep mind, and further connects to a third user who writes about AI and medical doctor. It is likely that those topics may stimulate one and another. In the example of communication networks given above, finding interesting patterns from the error logs can facilitate the discovery of error patterns among related communication nodes, and thus enables the administrators to better diagnose the network and identify the problems to fix. For example, a sequential pattern $\langle$(Error

Code 001, Error Code 002) $\rightarrow$ (Error Code 003) $\rightarrow$ (Error Code 004)$\rangle$ indicates that it happens frequently that one node who produces Error Code 001 and Error Code 002 makes a neighboring node produce Error Code 003, and further makes a third node produce Error Code 004.

The task illustrated in the above examples is very different from traditional sequential pattern mining. Specifically, instead of searching for patterns from a sequence database, here we are given a transaction database graph, where each vertex is associated with a transaction database, and our goal is to find sequential patterns that are frequently induced by all possible random walks of the whole graph.

As we want to find sequential patterns, a natural question is whether many existing sequential pattern mining methods, such as PrefixSpan [17], can be extended to solve this problem? Unfortunately, there is no a straightforward way to apply existing methods on a transaction database graph, since none of those methods take a transaction database graph as input and the number of transaction sequences induced by all possible random walks of a transaction database graph is exponential with respect to the number of vertices.

To the best of our knowledge, our study is the first to tackle the problem of finding top-k sequential patterns in transaction database graphs. Our major idea is to maintain a sample of random transaction sequences so that we can approximate the top-k patterns with provable quality guarantees. We make several contributions.

First, we formulate the problem of mining top-k sequential patterns in transaction database graphs. A transaction database graph in our problem is defined as a graph where each vertex is associated with a transaction database and a transaction is a set of items. Different from the traditional sequential pattern mining problem that aims to find frequent patterns in a given sequence database, we are interested in finding frequent sequential patterns in the extremely large sequence database that is induced by all possible random walks of the whole transaction database graph. The major challenge of our problem is that the sequence database of a large transaction database graph is usually too large to be explicitly induced and stored. By reducing from the conventional sequential pattern mining problem, we prove that our problem is #P-hard.

Second, we propose an exact sequential pattern finding algorithm, which is also used as a baseline. By fixing the path length l and an integer k, we first collect all of the length-l transaction sequences and then apply one of the state-of-the-art mining techniques to obtain the exact top-k patterns. However, when the size of the transaction database graph increases, this approach suffers from the expensive time and space cost, since the total number of transaction sequences is exponential with respect to the number of vertices.

Third, we carefully design a two-step sampling framework that significantly improves the mining efficiency. We first sample a set of transaction sequences from the transaction database graph by a two-step sampling algorithm. Using the sampled transaction sequences, we design an unbiased estimator to approach the frequencies of sequential patterns in the transaction database graph with provable guarantees on the estimation error. Based on the estimator, we introduce the weighting mechanism for the sampled transaction sequences.

Last, we conduct extensive experiments on both synthetic and real-world data sets. The results demonstrate the effectiveness and efficiency of the proposed method. We also conduct a case study to show meaningful sequential patterns mined from the Aminer [27].

The rest of the paper is organized as follows. Section 2 reviews the related work. We formulate the problem and present a baseline in Section 3. In Section 4, we develop our two-step sampling method and give the upper bound of sample complexity for our proposed algorithm. A systematic empirical study is reported in Section 5. We conclude our work in Section 6.

2 Related work

To the best of our knowledge, mining sequential patterns in transaction database graphs is a new problem, which has not been touched in literature. It is related to sequential pattern mining and sampling methods.

2.1 Sequential pattern mining

Sequential pattern mining is a well-studied subject in data mining, which was first introduced by [1]. It finds all frequent subsequences from a database of sequences and has enjoyed many applications, such as analyzing shopping patterns [24], classification [32] and understanding user behavior [36]. The Apriori-based algorithms, such as GSP [26] and SPADE [34], improve the efficiency of sequential pattern mining [1] by reducing the search space using the anti-monotonicity of sequential patterns. Distributed methods were also proposed to accelerate the mining processing [11]. To avoid generating candidate sequences, Han et al. [12] proposed FreeSpan. Later, Pei et al. [17] developed PrefixSpan. The two methods mine sequential patterns by sequence database projection and pattern growth.

Choosing an appropriate support threshold for sequential pattern mining is challenging in practice [10]. Tzvetkov et al. [31] proposed TSP, which aims to mine top-k frequent closed sequential patterns whose lengths pass a threshold. Closed sequential patterns are concise representations of sequential patterns.

Liu et al. [16] proposed to proactively reduce the representation of sequences to uncover significant, hidden temporal structures. Their key idea was to decrease the level of granularity of symbols in sequences by clustering the symbols. They claimed that, in this way, symbolic sequences would be represented as numerical sequences or point clouds in the Euclidean space which reduces the time cost. However, this reduction does not change the hardness of the sequential pattern mining problem and may lose some interesting patterns due to the information loss caused by embedding.

There are many existing sequential pattern mining methods. A thorough review of the subject is far beyond the capacity of this paper. Dong and Pei [8] provided a comprehensive review. Mining sequential patterns from transaction database graphs is a new problem. As explained in Section 1, the existing methods cannot be directly applied on a transaction database graph.

2.2 Sampling methods

Sampling and approximation methods [7, 28] have been widely used in pattern mining to tackle large data sets or expedite mining. Sampling methods have been widely used in pattern mining [4] and association rule mining [29], which reduce the amount of computation dramatically and achieve provable quality guarantees.

Raïssi and Poncelet [20] proposed to ease the sequential pattern mining operations by modeling the data as a continuous and potentially infinite stream. They used Hoeffding concentration inequalities to prove a lower bound of the sample size. For reducing static database access by constructing a random sample before mining, they extended the static sampling analysis to the data stream model. They also conducted a simple replacement algorithm using an exponential bias function to regulate the sampling. Another study about static sampling for patterns was proposed by [22]. They applied the statistical concept of *Vapnik-Chervonenkis (VC) dimension* to develop a novel technique that provides tight bounds on the sample size. The resulting sample size was linearly dependent on the VC-dimension of a range of space associated with the dataset to be mined. However, their method cannot be straightforwardly extended to sample transaction sequences from a transaction database graph, since it is not applicable to deal with the transaction sequences.

Progressive sampling methods are also widely used. Pietracaprina et al. [19] proposed to mine top-k frequent itemsets through progressive sampling. They first presented an upper bound on the sample size. Then they devised a progressive sampling approach that extracted the top-k frequent itemsets from increasingly larger samples until suitable stopping conditions were met or the upper bound was hit. Riondato and Upfal [23] introduced another progressive sampling method with *Rademacher Averages* [2] for mining frequent itemsets. This work studied the trade-off between the approximation quality and the sample size using concepts and results from *statistical learning theory* [5], where the stopping condition was based on bounds to the empirical Rademacher average of the problem. However, these methods focused on the conventional frequent itemset mining problem, and cannot be directly extended to find sequential patterns in transaction database graphs.

Sampling transaction sequences from a transaction database graph is also related to graph path sampling methods [21, 35]. For example, Zhang et al. [35] proposed Panther to measure the similarity between vertices by sampling paths with random walk. However, Panther does not consider transaction databases on vertices, thus it cannot be straightforwardly extended to sampling transaction sequences from the transaction database graphs.

3 Problem definition and baseline

In this section, we define the problem formally and then present a baseline. We also show that the problem is indeed #P-hard.

3.1 Problem definition

Let I be a set of *items*. An *itemset* X is a subset of I, that is, $X \subseteq I$. A *transaction* is a tuple $T = (tid, X)$, where tid is a unique transaction-id and X an itemset. A transaction $T = (tid, X)$ is said to *contain* itemset Y if $Y \subseteq X$. In such a case, we overload the subset symbol and write $Y \subseteq T$. A *transaction database* $\mathcal{T}$ is a set of transactions.

A *transaction database graph* is a directed graph, denoted by $G = (V, E, \mathbb{T})$, where V is a set of vertices, $E = \{(u, v) \mid u, v \in V\} \subseteq V \times V$ is a set of directed edges and $\mathbb{T}$ is a set of transaction databases. Each vertex $v \in V$ is associated with a transaction database $\mathcal{T}_v \in \mathbb{T}$. For the sake of simplicity, hereafter a transaction database graph is also called a *graph*.

A *(directed) path* p in a directed graph G is a sequence of vertices $p = \langle v_1, \ldots, v_h \rangle$, where $(v_i, v_{i+1}) \in E$ $(1 \leq i < h)$, and the *length* of the path is $len(p) = h - 1$, the number of edges in the path. In this paper, we consider simple paths only, that is, a vertex appears in a path at most once. For two vertices v_i, v_j in G, the *distance* from v_i to v_j, denoted by $dist(v_i, v_j)$, is the length of the shortest path from v_i to v_j in G. For a vertex $v \in V$, the *l-neighborhood* of v is $N_l(v) = \{v_i \mid dist(v, v_i) \leq l\}$. We define the *$l$-th order degree* of v as the number of vertices of distance l from v, that is,

$$d_l(v) = \begin{cases} 1 & \text{if } l = 0 \\ \|\{v_i \mid dist(v, v_i) = l\}\| & \text{if } l > 0 \end{cases} \tag{1}$$

Apparently, for $l > 0$, $d_l(v) = \|N_l(v) - N_{l-1}(v)\|$.

We are interested in sequential patterns carried by paths in a graph. Formally, a *transaction sequence* in G, denoted by $ts = \langle T_1, \ldots, T_h \rangle$, is a sequence of transactions such that there exists a path $p = \langle v_1, \ldots, v_h \rangle$ and $T_i \in \mathcal{T}_{v_i}$ for $1 \leq i \leq h$. In such the case, we say ts is *supported* by p and the length of ts is $len(ts) = h - 1$.

A *sequential pattern* $s = \langle X_1, \ldots, X_h \rangle$ is a series of itemsets and the length of the sequential pattern is $len(s) = h - 1$. A transaction sequence $ts = \langle T_1, \ldots, T_h \rangle$ *contains* a sequential pattern $s = \langle X_1, \ldots, X_h \rangle$ if $X_i \subseteq T_i$ for $1 \leq i \leq h$. A *sequential pattern* is also called a *pattern*.

Denote D_l as the set of all length-l transaction sequences supported by paths in G, the *frequency* of pattern s in D_l, denoted by $f(s)$, is the proportion of unique transaction sequences in D_l that contain s.

Example 1 (Concepts) Figure 1 shows a graph G, where each vertex is associated with a transaction database.

G has 7 vertices, which are v_1, v_2, v_3, v_4, v_5, v_6 and v_7. We take vertex v_1 as an example. The transaction database associated with v_1 contains three transactions T_1, T_2 and T_3. Specifically, T_1 contains three items, which are i_1, i_2 and i_3. In particular, in the example of a social network, v_1 and v_2 would represent users, and the edge (v_1, v_2) indicates v_1 follows v_2. The transaction T_1 corresponds to a piece of tweet, and the item i_1 can represent a keyword or a video clip in the tweet.

The *1*-, *2*- and *3-neighborhoods* of v_1 are $N_1(v_1) = \{v_2, v_3\}$, $N_2(v_1) = \{v_2, v_3, v_4, v_5, v_6\}$ and $N_3(v_1) = \{v_2, v_3, v_4, v_5, v_6, v_7\}$, respectively. The first, second, and third degrees of v_1 are $d_1(v_1) = 2$, $d_2(v_1) = 3$ and $d_3(v_1) = 1$, respectively. Starting from v_1, there are 6 paths, $p_1 = \langle v_1, v_2 \rangle$, $p_2 = \langle v_1, v_3 \rangle$, $p_3 = \langle v_1, v_2, v_4 \rangle$, $p_4 = \langle v_1, v_2, v_5 \rangle$, $p_5 = \langle v_1, v_3, v_6 \rangle$ and $p_6 = \langle v_1, v_2, v_4, v_7 \rangle$. Since p_6 is the only length-3 path in G, the length-3 transaction sequence set D_3 in G consists of all the transaction sequences supported by p_6.

Figure 2 shows the transaction sequences in D_3.

Now, we are ready to define the *Top-k Sequential Pattern Mining problem in Graph (TSPMG) as follows.*

Problem 1 (TSPMG) Given a graph G, an integer $k > 0$ and a fixed length $l > 0$, the problem of finding the top-k sequential patterns is to find the top-k patterns of length l that have the largest frequencies in D_l of G.

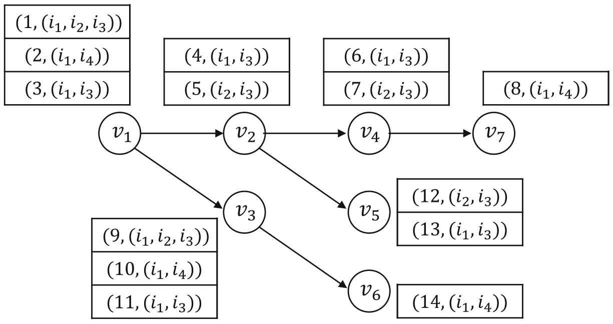


Figure 1 An example of a graph G, where each vertex is associated with a transaction database

Figure 2 The set of length-3 transaction sequences D_3 of the graph G in Figure 1

$$
\begin{gathered}
\langle(i_1,i_2,i_3)(i_1,i_3)(i_1,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_2,i_3)(i_1,i_3)(i_2,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_2,i_3)(i_2,i_3)(i_1,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_2,i_3)(i_2,i_3)(i_2,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_4)(i_1,i_3)(i_1,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_4)(i_1,i_3)(i_2,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_4)(i_2,i_3)(i_1,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_4)(i_2,i_3)(i_2,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_3)(i_1,i_3)(i_1,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_3)(i_1,i_3)(i_2,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_3)(i_2,i_3)(i_1,i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_3)(i_2,i_3)(i_2,i_3)(i_1,i_4)\rangle
\end{gathered}
$$

Example 2 (TSPMG) Consider the graph G in Figure 1. Let $k = 9$ and $l = 3$. The top-9 length-3 patterns, which have the largest frequencies in D_3 (shown in Figure 2), are listed in Figure 3.

Table 1 summarizes the frequently used notations.

3.2 Hardness of the TSPMG problem

In this section, we analyze the hardness of the TSPMG problem. We define the *Sequential Pattern Counting problem in Graph* (SPCG) and prove that the SPCG problem is #P-hard. In the sequel, we prove that the TSPMG problem is at least as hard as the SPCG problem.

Problem 2 (SPCG) Given a transaction database graph G, a threshold $\alpha > 0$ and a fixed length $l > 0$, the problem of counting sequential patterns is to count the number of length-l patterns with frequencies larger than α in D_l of G.

Figure 3 The top-9 length-3 frequent sequential patterns in D_3 of the graph G in Figure 1

$$
\begin{gathered}
\langle(i_1)(i_3)(i_3)(i_1)\rangle\\
\langle(i_1)(i_3)(i_3)(i_4)\rangle\\
\langle(i_1)(i_3)(i_3)(i_1,i_4)\rangle\\
\langle(i_3)(i_3)(i_3)(i_1)\rangle\\
\langle(i_3)(i_3)(i_3)(i_4)\rangle\\
\langle(i_3)(i_3)(i_3)(i_1,i_4)\rangle\\
\langle(i_1,i_3)(i_3)(i_3)(i_1)\rangle\\
\langle(i_1,i_3)(i_3)(i_3)(i_4)\rangle\\
\langle(i_1,i_3)(i_3)(i_3)(i_1,i_4)\rangle
\end{gathered}
$$

Table 1 Frequently used notations

Notation	Description
$dist(v_i, v_j)$	The distance from v_i to v_j.
$d_i(v)$	The i-th order degree of v.
p	A *(directed) path* in G, which is a sequence of vertices.
$f(s)$	The real frequency of pattern s.
$\hat{f}(s)$	The unbiased pattern frequency estimator.
D_l	The set of all length-l transaction sequences in G.
P_l	The set of all length-l paths in G.
S_l	The set of sampled length-l transaction sequences.
I	The set of all items in G.

Theorem 1 *The SPCG problem is #P-hard.*

Proof (sketch)

Given a sequence database SDB, for each sequence in the database that contains a list of transactions, we can construct a graph that contains a single path such that a node is set up for each transaction and the nodes are linked into a path according to the order of transactions in the sequence. Moreover, for a single path $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_{l+1}$, we add edges (T_i, T_j) for $i < j$. $\frac{l(l-1)}{2}$ edges are added on a path of length l. In this way, a sequence in SDB is transformed into a directed acyclic graph (DAC).

Then, we combine all the DACs into a transaction database graph G. That is, the graph is a collection of DACs, one representing a sequence in SDB. Thus, each node contains a transaction database where there is only one transaction. Apparently, this reduction step takes polynomial time.

It can be shown easily that a subsequence is a sequential pattern in SDB if and only if it is a sequential pattern in the transaction database graph constructed. $\square$

Theorem 2 *The TSPMG problem is at least as hard as the SPCG problem.*

Proof We prove this by a Cook reduction from the SPCG problem. Denote by $\Theta(D_l, k)$ the set of top-k length-l patterns in G. Let $s_k = \Theta(D_l, k) \setminus \Theta(D_l, k-1)$ be the pattern with the k-th largest frequency $f(s_k)$ in D_l of G.

Given the oracle of the TSPMG problem, we can search for the parameter k that satisfies $f(s_k) > \alpha \geq f(s_{k+1})$ by querying the oracle multiple times. Such k is exactly the answer to the SPCG problem.

Let I be the set of all items in G. We have $k \in [1, 2^{|I|(l+1)}]$. Thus, the time of a binary search for k is in $\mathcal{O}(|I|(l+1))$. For each k, we can obtain s_k and s_{k+1} by querying the oracle to get $\Theta(D_l, k-1)$, $\Theta(D_l, k)$ and $\Theta(D_l, k+1)$; $f(s_k)$ and $f(s_{k+1})$ can be computed by linearly searching D_l in $\mathcal{O}(|V|^{(l+1)}C^{(l+1)})$ time. Therefore, the overall time to solve the SPCG problem is in $\mathcal{O}(|I|(l+1)|V|^{(l+1)}C^{(l+1)})$. Recall that l is the constant path length, we know that the SPCG problem is Cook reducible to the TSPMG problem. $\square$

3.3 Baseline

Given a graph G, how can we find interesting patterns on directed paths? Intuitively, with a given path length l, we can enumerate all paths in G and then extract all transaction sequences supported by the paths. Then, by applying traditional sequential pattern mining techniques [1, 17], we can find the exact top-k sequential patterns among transaction sequences. Algorithm 1 gives the details of this method.

Algorithm 1 The baseline method.

Input: A transaction database graph G, path length l and k.
Output: The set of top-k length-l sequential patterns $\Theta(D_l, k)$.

```
1: Initialize length-$l$ transaction sequence set $D_l \leftarrow \emptyset$.
2: for each length-$l$ path $p$ in $G$ do
3:     for each transaction sequence $ts$ induced by $p$ do
4:         $D_l \leftarrow D_l \cup ts$.
5:     end for
6: end for
7: Apply sequential pattern mining techniques on $D_l$ to obtain $\Theta(D_l, k)$.
8: return $\Theta(D_l, k)$.
```

The volume of D_l, denoted by $|D_l|$, is in $\mathcal{O}(|V|^{(l+1)}C^{(l+1)})$, where $|V|$ is the number of vertices in G and $C = \max_{v_i \in V} |\mathcal{T}_{v_i}|$ is the maximum number of transactions in the transaction database of a single vertex of G. As a result, the baseline method is computationally expensive due to the large amount of time and space needed to enumerate and store the transaction sequences in D_l.

In the next section, we tackle this problem with an efficient sampling method that significantly reduces the number of transaction sequences needs to search, and thus achieves high-quality approximation results.

4 A fast sampling-based method

The key idea of our fast sequential pattern mining method is to first estimate the pattern frequencies using a set of transaction sequences sampled from G, then obtain the top-k patterns using the estimated pattern frequencies. In this section, we first introduce our *two-step sampling framework* that randomly samples transaction sequences. Then, we provide an unbiased estimator to estimate pattern frequency using the sampled transaction sequences. Last, we prove the upper bound of the complexity for the sampling-based algorithm.

4.1 A two-step sampling framework

The two-step sampling framework involves a path sampling method that uniformly samples a set of length-l paths from G and a transaction sequence sampling method that uniformly samples transaction sequences from each sampled path.

Notice that, although the first step sampling perform uniform sampling of the paths on the graph and the second step sampling perform uniform sampling of transaction sequence on the sampled paths, respectively, the overall two-step sampling is not uniform with respect to the length-l transaction sequences, since the numbers of transactions are different in different vertices. As a result, we cannot directly mine frequent sequential patterns from the sampled transaction sequences by simply using the support of patterns in the sampled transaction sequences. However, as illustrated in Section 4.2, we can still mine the top-k sequential patterns by estimating the pattern frequencies with bounded estimation error using a carefully designed unbiased estimator.

Denote by P_l the set of all length-l paths in G. The *path sampling method* samples a length-l ($l \geq 1$) path $p = \langle v_1, \dots, v_q, \dots, v_{l+1} \rangle$ from P_l by progressively sampling an ordered sequence of $(l+1)$ vertices, where each vertex v_q in p is sampled with probability

$$\mathbb{P}(v_q) = \begin{cases} \mathbb{P}(v_1) = \frac{d_l(v_1)}{\sum_{v_j \in V} d_l(v_j)} & q = 1 \\ \mathbb{P}(v_q|v_{q-1}) = \frac{d_{l-q+1}(v_q)}{d_{l-q+2}(v_{q-1})} & 2 \leq q \leq (l+1) \end{cases} \tag{2}$$

Algorithm 2 gives the pseudocode of the path sampling step. We prove that Algorithm 2 conducts a uniform sampling on paths.

Algorithm 2 Path sampling method.

```
Input: Graph $G$ and path length $l$.
Output: A length-$l$ path $p$.
1: $p \leftarrow \emptyset$.
2: for $q = 1 \rightarrow l$ do
3:     Sample vertex $v_q$ with the probability given by Equation 2.
4:     $p \leftarrow p \cup v_q$.
5: end for
6: return $p$.
```

Theorem 3 *Given a graph G and a path length l, Algorithm 2 uniformly samples a path p from P_l.*

Proof Since P_l is the set of all length-l paths in G, to show Algorithm 2 is uniform, we need to prove that the probability of sampling a length-l path p is $\frac{1}{|P_l|}$.

Denote by $\mathbb{P}(p)$ the probability of sampling a path $p = \langle v_1, \dots, v_{l+1} \rangle$ in G. According to Algorithm 2 and (2), we have

$$\begin{aligned} \mathbb{P}(p) &= \mathbb{P}(v_1) \times \mathbb{P}(v_2) \times \cdots \times \mathbb{P}(v_{l+1}) \\ &= \mathbb{P}(v_1) \times \mathbb{P}(v_2|v_1) \times \cdots \times \mathbb{P}(v_{l+1}|v_l) \\ &= \frac{d_l(v_1)}{\sum_{v_j \in V} d_l(v_j)} \times \frac{d_{l-1}(v_2)}{d_l(v_1)} \times \cdots \times \frac{1}{d_1(v_l)} \\ &= \frac{1}{\sum_{v_j \in V} d_l(v_j)} \\ &= \frac{1}{|P_l|} \end{aligned} \tag{3}$$

The theorem holds. □

Algorithm 2 uniformly samples one path from all length-l paths in G. To sample a set of length-l paths with replacement, we simply run Algorithm 2 multiple times.

Next, we introduce the transaction sequence sampling method. Denote by $p^i = \langle v_1^i, \dots, v_q^i, \dots, v_{l+1}^i \rangle$ the i-th length-l path sampled by Algorithm 2, the *transaction sequence sampling method* in Algorithm 3 uniformly samples a transaction sequence ts^i with probability $\frac{1}{\prod_{q=1}^{l+1} |\mathcal{T}_{v_q{}^i}|}$ from all transaction sequences induced by p^i.

Algorithm 3 Transaction sequence sampling method.

```
Input: The $i$-th sampled length-$l$ path $p^i = \langle v_1^i, \dots, v_q^i, \dots, v_{l+1}^i \rangle$.
Output: A sampled transaction sequence $ts^i$.
1: $ts^i \leftarrow \emptyset$.
2: for $q = 1 \rightarrow (l+1)$ do
3:     Sample transaction $T_q^i$ from vertex $v_q^i$ with the probability $\frac{1}{|T_{v_q^i}|}$.
4:     $ts^i \leftarrow ts^i \cup T_q^i$.
5: end for
6: return $ts^i$.
```

4.2 Unbiased pattern frequency estimator

In this section, we first define some useful notations. Then, we introduce an unbiased estimator of pattern frequency.

For a length-l path $p^i = \langle v_1^i, \dots, v_q^i, \dots, v_{l+1}^i \rangle$, we assign a unique index to each transaction sequence supported by p^i, and denote by ts_j^i the j-th transaction sequence supported by p^i. Apparently, $M^i = \prod_{i=1}^{l+1} |\mathcal{T}_{v_q^i}|$ is the number of all transaction sequences supported by p^i.

We further define a random variable $Y_j^i(h)$

$$Y_j^i(h) = \begin{cases} 1 & \text{if the } h\text{-th sample draws } ts_j^i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $1 \leq h \leq |S_l|$ is the index of a sample and $|S_l|$ is the volume of S_l. Since ts_j^i is drawn with probability $\frac{1}{|P_l| M^i}$, we have $\mathbb{E}(Y_j^i(h)) = \frac{1}{|P_l| M^i}$.

Now we present an unbiased pattern frequency estimator. Denote by P_l the set of all length-l paths in graph G. For a pattern $s = \langle X_1 \dots, X_q, \dots, X_{l+1} \rangle$, an estimator of the real pattern frequency $f(s)$ is

$$\hat{f}(s) = \frac{\sum_{i=1}^{|P_l|} \sum_{j=1}^{M^i} \sum_{h=1}^{|S_l|} Y_j^i(h) W_j^i}{\frac{|S_l|}{|P_l|} \sum_{i=1}^{|P_l|} M^i} \tag{5}$$

where W_j^i is defined as

$$W_j^i = \begin{cases} M^i & s \subseteq ts_j^i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Note that, since $\mathbb{P}(s \subseteq ts_j^i) = f(s)$, we have $\mathbb{E}(W_j^i) = f(s) M^i$.

Now, we prove that $\hat{f}(s)$ is an unbiased estimator of $f(s)$.

Theorem 4 *For a pattern s, $\hat{f}(s)$ is an unbiased estimator of $f(s)$.*

Proof First, since $Y_j^i(h)$ and W_j^i are independent random variables, we have

$$\mathbb{E}(Y_j^i(h) W_j^i) = \mathbb{E}(Y_j^i(h)) \mathbb{E}(W_j^i) = \frac{f(s)}{|P_l|}$$

Then, we have

$$
\begin{aligned}
\mathbb{E}[\hat{f}(s)] &= \frac{\sum_{i=1}^{|P_l|}\sum_{j=1}^{M^i}\sum_{h=1}^{|S_l|}\mathbb{E}(Y_j^i(h)W_j^i)}{\frac{|S_l|}{|P_l|}\sum_{i=1}^{|P_l|}M^i} \\
&= \frac{\sum_{i=1}^{|P_l|}\sum_{j=1}^{M^i}\frac{|S_l|}{|P_l|}f(s)}{\frac{|S_l|}{|P_l|}\sum_{i=1}^{|P_l|}M^i} \\
&= f(s)
\end{aligned}
$$

The proof follows. □

Using the estimator $\hat{f}(s)$ (5) to calculate the top-k most frequent patterns is efficient. In our problem, we only care about the order of the pattern frequencies. As we have shown above, the denominator of $\hat{f}(s)$, that is, $\frac{|S_l|}{|P_l|}\sum_{i=1}^{|P_l|}M^i$, won't affect the order of pattern frequencies. Therefore, the patterns can be sorted by the nominator of $\hat{f}(s)$, which is the weighted sum of the count of transactions in S_l that contains pattern s. For example, for a pattern s, if it is contained in ts^1 and ts^2, the weighted sum is equal to $M^1 + M^2$. We can thus derive the weight of the transaction sequence ts^i supported by p^i as $\phi(ts^i) = M^i$.

The *two-step sampling framework* is summarized in Algorithm 4, which samples a length-l transaction sequence set S_l from G in two steps. First, we uniformly sample m length-l paths from G using Algorithm 2. Then, for each sampled path p^i, we uniformly sample one transaction sequence ts^i by using Algorithm 3 and compute its weight $\phi(ts^i)$. At last, we output S_l, which consists of all the sampled transaction sequences and their corresponding weights.

The time complexity of calculating degrees with Equation 1 is $\mathcal{O}(|E|)$, where $|E|$ denotes the number of edges. Algorithm 2 samples m length-l transaction sequences and it needs $\mathcal{O}(ml)$ time. Algorithm 3 samples transaction sequences from sampled paths and it also needs $\mathcal{O}(ml)$ time. Thus, the overall time complexity of Algorithm 4 is $\mathcal{O}(|E| + 2ml)$.

Algorithm 4 Two-step sampling framework.

```
Input: Graph $G$, path length $l$ and sample size $m$.
Output: A set $S_l$ of sampled length-$l$ transaction sequences with their corresponding weights.
1: for each $v_j \in V$ do
2:     Calculate degrees $\{d_q(v_j) | 1 \le q \le l\}$ using Equation 1.
3: end for
4: $\psi \leftarrow \emptyset$, $S_l \leftarrow \emptyset$
5: for $i = 1 \rightarrow m$ do
6:     Sample a path $p^i = \langle v_1^i, \ldots, v_{l+1}^i \rangle$ from $G$ using Algorithm 2.
7:     $\psi \leftarrow \psi \cup p^i$.
8: end for
9: for each $p^i$ in $\psi$ do
10:    Sample a transaction sequence $ts^i$ from $p^i$ using Algorithm 3.
11:    $S_l \leftarrow S_l \cup (ts^i, \phi(ts^i))$.
12: end for
13: return $S_l$.
```

4.3 Bounding the sample size

In this section, we analyze the approximation error of $\hat{f}(s)$ in Theorem 6 and derive an upper bound of sample complexity for the proposed sampling-based algorithm in Theorem 7 and Theorem 8.

Theorem 5 (**Chernoff's inequality** [6]) *Let $Z_1, \ldots, Z_n$ be independent random variables. They need not have the same distribution. Assume that $0 \leq Z_i \leq 1$ always, for each i. Let $Z = Z_1 + \ldots + Z_n$. Write $\mu = \mathbb{E}[Z] = \mathbb{E}[Z_1] + \ldots + \mathbb{E}[Z_n]$. Then for any $\varepsilon \geq 0$,*

$$\mathbb{P}[|Z - \mu| \geq \varepsilon\mu] \leq 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

Theorem 6 *Given a set of length-l sampled transaction sequences S_l and an arbitrary pattern s, for a fixed threshold $\varepsilon \geq 0$, we have:*

$$\mathbb{P}(|\hat{f}(s) - f(s)| \geq \varepsilon) \leq 2\exp(-\frac{\varepsilon^2 a}{3}|S_l|)$$

where $a = \frac{1}{|P_l|M^*}\sum_{i=1}^{|P_l|} M^i$ and $M^* = \max_i M^i$.

Proof We define a random variable $U_j^i(h) = Y_j^i(h)W_j^i$. According to (4) and (6), we have $U_j^i(h) \in [0, M^i]$. Thus, $\frac{U_j^i(h)}{M^*} \in [0, 1]$.

Now, consider random variable

$$U = \sum_{i=1}^{|P_l|}\sum_{j=1}^{M^i}\sum_{h=1}^{|S_l|} U_j^i(h).$$

Applying Theorem 5, it follows

$$\mathbb{P}\left(|\frac{U}{M^*} - \frac{\mathbb{E}(U)}{M^*}| \geq t\frac{\mathbb{E}(U)}{M^*}) \leq 2\exp(-\frac{t^2}{3}\frac{\mathbb{E}(U)}{M^*}\right)$$

Since $\mathbb{E}(U) = f(s)\frac{|S_l|}{|P_l|}\sum_{i=1}^{|P_l|} M^i$ and $\frac{U}{\frac{|S_l|}{|P_l|}\sum_{i=1}^{|P_l|} M^i} = \hat{f}(s)$ (see (5)), it follows

$$\mathbb{P}(|\hat{f}(s) - f(s)| \geq tf(s)) \leq 2\exp\left(-\frac{t^2 f(s)}{3}\frac{|S_l|}{|P_l|M^*}\sum_{i=1}^{|P_l|} M^i\right)$$

Since $f(s) \in [0, 1]$, it follows

$$\mathbb{P}(|\hat{f}(s) - f(s)| \geq tf(s)) \leq 2\exp\left(-\frac{t^2 f^2(s)}{3}\frac{|S_l|}{|P_l|M^*}\sum_{i=1}^{|P_l|} M^i\right)$$

Let $\varepsilon = tf(s)$ and $a = \frac{1}{|P_l|M^*}\sum_{i=1}^{|P_l|} M^i$. It follows

$$\mathbb{P}(|\hat{f}(s) - f(s)| \geq \varepsilon) \leq 2\exp\left(-\frac{\varepsilon^2 a}{3}|S_l|\right)$$

The proof follows. □

In Theorem 7, we further derive an upper bound of sample complexity for the proposed sampling-based algorithm to make sure for each pattern s, $|\hat{f}(s) - f(s)| \leq \varepsilon$ holds with a high probability.

Theorem 7 *Denote by Q_l the set of all length-l patterns in G. If $|S_l| \geq \frac{12}{\varepsilon^2 a} ln \frac{2|Q_l|}{\delta}$, then $\mathbb{P}(\forall s \in Q_l, |\hat{f}(s) - f(s)| < \frac{\varepsilon}{2}) \geq 1 - \delta$.*

Proof Since Q_l is the set of all length-l patterns in G, by Theorem 6 and the Union Bound [3], it follows

$$\mathbb{P}(\exists s \in Q_l, |\hat{f}(s) - f(s)| \geq \frac{\varepsilon}{2}) \leq 2|Q_l| \exp\left(-\frac{\varepsilon^2 a}{12}|S_l|\right)$$

Thus, we have

$$\mathbb{P}(\forall s \in Q_l, |\hat{f}(s) - f(s)| < \frac{\varepsilon}{2}) \geq 1 - 2|Q_l| \exp\left(-\frac{\varepsilon^2 a}{12}|S_l|\right)$$

Since $|S_l| \geq \frac{12}{\varepsilon^2 a} ln \frac{2|Q_l|}{\delta}$, it follows that

$$1 - 2|Q_l| \exp\left(-\frac{\varepsilon^2 a}{12}|S_l|\right) \geq 1 - \delta$$

Thus, we have

$$\mathbb{P}\left(\forall s \in Q_l, |\hat{f}(s) - f(s)| < \frac{\varepsilon}{2}\right) \geq 1 - \delta$$

The proof follows. □

The bound of $\frac{12}{\varepsilon^2 a} ln \frac{2|Q_l|}{\delta}$ in Theorem 7 is an upper bound of sample complexity for the proposed sampling-based algorithm, since the Union Bound [3] is applied to bound the approximation errors of pattern frequencies of all patterns in Q_l. As demonstrated later in the experiments, in practice we can achieve a good estimation quality with a sample size that is smaller than the bound in Theorem 7. Next, we analyze the relationship between $|S_l|$ and path length l in Theorem 8.

Theorem 8 *Denote by I the set of all items in G, and by Q_l the set of all length-l patterns in G. If $|S_l| \geq \frac{12|I|(l+1)+12}{\varepsilon^2 a} ln \frac{2}{\delta}$, then $\mathbb{P}(\forall s \in Q_l, |\hat{f}(s) - f(s)| < \frac{\varepsilon}{2}) \geq 1 - \delta$.*

Proof Since Q_l is the set of all patterns in G, we have

$$|Q_l| \leq 2^{|I|(l+1)}$$

By substituting the above inequality into the condition $|S_l| \geq \frac{12}{\varepsilon^2 a} ln \frac{2|Q_l|}{\delta}$ of Theorem 7, the theorem follows. □

According to Theorem 8, when $|S_l| \geq \frac{12|I|(l+1)+12}{\varepsilon^2 a} ln \frac{2}{\delta}$, the estimated pattern frequency $\hat{f}(s)$ for each pattern s will have a high probability to be close to its real frequency $f(s)$. Therefore, we can use a sequential pattern mining method, such as PrefixSpan [17], to extract sequential patterns from the set of sampled transaction sequences. The ranking on the estimated pattern frequencies will be a good approximation of the ranking of real pattern frequencies. As a result, we can obtain the high-quality approximation of top-k sequential patterns using the pattern frequencies estimated from the set of sampled transaction sequences.

Comparing to the baseline method that enumerates the exponential number of transaction sequences in D_l, the proposed sampling method significantly reduces the number of required transaction sequences and achieves guaranteed approximation quality.

5 Experiments

In this section, we analyze the effectiveness and efficiency of the proposed algorithms on synthetic and real-world networks.

5.1 Evaluation datasets

The following datasets are used.

– **SYN** is a synthetic dataset. We build two synthetic datasets, denoted by SYN1 and SYN2, which are generated by the IBM Quest Synthetic Data Generator.[1] Both SYN1 and SYN2 consist of the same graph structure with 28 edges and 24 vertices. The difference between SYN1 and SYN2 lies in the distribution of transaction databases. In SYN1, each vertex has 20 transactions. In SYN2, each vertex with degree d has $8(d+1)$ transactions in the associated transaction database.
– **Flight** is a flight routing network published in OpenFlights website.[2] We build our graph by treating each airport as a vertex and each airline as a directed edge. Each transaction on the vertex is an entry of routes, where each route contains the information of source airport, destination airport, codeshare, stops, equipment, and so on. The top-k sequential patterns in the dataset show interesting flight services following the airlines.
– **CN** is a collaboration network built from Aminer dataset [27]. We build our graph datasets by treating each paper as a vertex and each citation as a directed edge. Each transaction database of a vertex is a set of topics, where each topic is represented by a set of keywords extracted from the abstracts of the paper. The top-k sequential patterns in these datasets reveal interesting patterns of topic changes in paper citations and relations among research topics. We build two citation network datasets, denoted by CN1 and CN2, that have different sizes.

The statistics of the datasets are listed in Table 2, where "#Vertices" represents the number of vertices, "#Edges" represents the number of edges, "#Trans" represents the number of unique transactions in G, "Average #items/trans" represents the average number of items in a transaction, "#S for $l = 1$" and "#S for $l = 2$" represent the number of all the transaction sequences for $l = 1, 2$.

5.2 Comparison methods and evaluation metrics

The following comparison methods are used.

– **TSPMG** is a full implementation of the proposed approach, which uses Algorithm 4 to sample weighted transaction sequences and then applies PrefixSpan [17] to extract frequent sequential patterns with the set of sampled transaction sequences.

[1] http://www.almaden.ibm.com/cs/quest/syndata.html
[2] https://openflights.org/data.html

Table 2 The statistics of the datasets. The symbol "-" indicates that we cannot obtain the statistics owing to the limited capacity of our main memory

Datasets	#Vertices	#Edges	#Transactions	Average #items/trans	#S for $l = 1$	#S for $l = 2$
SYN1	24	28	480	5.00	11,200	184,000
SYN2	24	28	420	5.00	8,828	141,312
Flight	28	488	2,524	4.00	44,013	2,369,477
CN1	16,198	60,856	120,741	6.73	–	–
CN2	324,228	1,809,469	3,340,335	6.59	–	–

- **TSPMG-P** differs from TSPMG in that it adopts random walks [15] instead of Algorithm 2 to sample paths. The random walk first selects a vertex from the graph randomly, and then randomly selects a neighbor of the previous selected vertex. This process iterates until the path length reaches l.
- **TSPMG-S** differs from TSPMG in that the weight of each sampled transaction sequence is 1.
- **TSPMG-B** is a baseline introduced in Algorithm 1, which also uses PrefixSpan [17] to extract frequent sequential patterns in the mining stage.

To evaluation these methods, we use the following evaluation metrics. We denote $\mathcal{G}$ as the ground truth ranked list of the real top-k patterns and by $\mathcal{L}$ the ranked list of patterns produced by the proposed sampling-based method. Let $\mathcal{G}(k)$ and $\mathcal{L}(k)$ be the ranked lists of the top-k patterns in $\mathcal{G}$ and $\mathcal{L}$, respectively.

- **Mean Estimation Error (ME)** is the average error between real pattern frequency $f(s)$ and estimated pattern frequency $\hat{f}(s)$ in (5). Denote s_i as the i-th pattern in ranked list $\mathcal{G}(k)$, the ME at rank-k is computed as

$$ME(k) = \frac{\sum_{i=1}^{k} |f(s_i) - \hat{f}(s_i)|}{k} \tag{7}$$

- **Average Precision (AP)** [25] is widely used to evaluate the similarity between two ranked lists in the field of information retrieval. Denote by $R(i)$ an indicator function that equals to 1 if the pattern at rank i in $\mathcal{L}$ is contained in $\mathcal{G}$, the average precision of $\mathcal{L}$ when using $\mathcal{G}(k)$ as the ground truth is

$$AP(k) = \frac{\sum_{i=1}^{|\mathcal{L}|} \frac{|\mathcal{L}(i) \cap \mathcal{G}(k)|}{|\mathcal{L}(i)|} R(i)}{|\mathcal{G}(k)|} \tag{8}$$

- **Ranking Similarity (RS)** [14] quantifies the degree of similarity between two ranked lists at rank k. The ranking similarity of $\mathcal{G}$ and $\mathcal{L}$ at rank k is

$$RS(k) = \frac{|\mathcal{G}(k) \cap \mathcal{L}(k)|}{k} \tag{9}$$

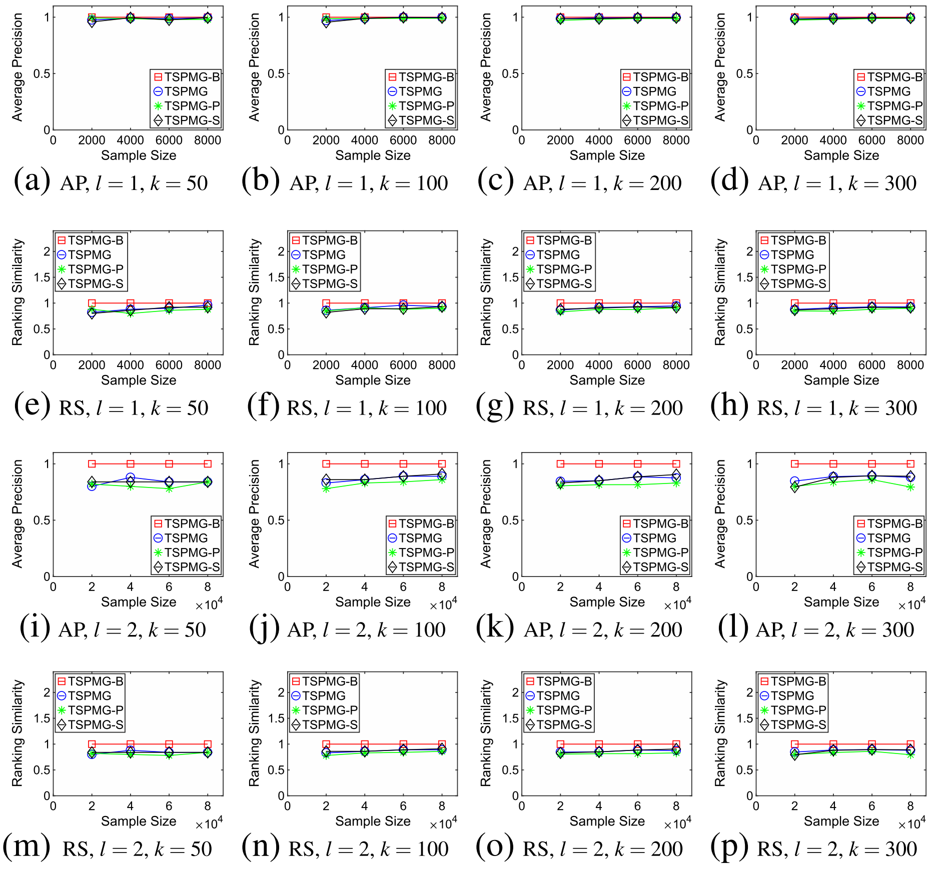
- **Time Cost (TC, in milliseconds) and Space Cost (SC, in megabytes)** are used to measure the running time and main memory usage respectively. Specifically, we use **STC** and **MTC** (resp. **SSC** and **MSC**) to represent the sampling time and mining time (resp. sampling memory usage and mining memory usage) respectively. Since the sizes of SYN1, SYN2 and Flight are small, we omit the results of the memory usage on them.

5.3 Implementation details

All algorithms were implemented in Java and compiled with JRE 9. All experiments were performed on a Windows 10 system with 64GB main memory and 4.00 GHz CPU. We set $l = 1, 2$ and $k = 50, 100, 200, 300$ respectively. The sample sizes are set as follows. Denote S_l as the set of the sampled transaction sequences for path length l and $|S_l|$ as the volume. For SYN1 and SYN2, $|S_1| = 2 \times 10^3, 4 \times 10^3, 6 \times 10^3, 8 \times 10^3$ and $|S_2| = 2 \times 10^4, 4 \times 10^4, 6 \times 10^4, 8 \times 10^4$. For Flight, $|S_1| = 1 \times 10^4, 2 \times 10^4, 3 \times 10^4, 4 \times 10^4$ and $|S_2| = 2 \times 10^5, 4 \times 10^5, 6 \times 10^5, 8 \times 10^5$. For CN1 and CN2, $|S_2| = |S_3| = 2 \times 10^5, 4 \times 10^5, 6 \times 10^5, 8 \times 10^5$.

5.4 The results on synthetic datasets

In this subsection, we introduce the experimental results on the synthetic datasets. Since SYN1 and SYN2 are small, we are able to run TSPMG-B to obtain the exact pattern frequencies of all patterns and the exact ranked lists of the top-k sequential patterns, respectively. The exact pattern frequencies are used to calculate the ME of the proposed sampling-based



(a) AP, $l = 1, k = 50$ (b) AP, $l = 1, k = 100$ (c) AP, $l = 1, k = 200$ (d) AP, $l = 1, k = 300$

(e) RS, $l = 1, k = 50$ (f) RS, $l = 1, k = 100$ (g) RS, $l = 1, k = 200$ (h) RS, $l = 1, k = 300$

(i) AP, $l = 2, k = 50$ (j) AP, $l = 2, k = 100$ (k) AP, $l = 2, k = 200$ (l) AP, $l = 2, k = 300$

(m) RS, $l = 2, k = 50$ (n) RS, $l = 2, k = 100$ (o) RS, $l = 2, k = 200$ (p) RS, $l = 2, k = 300$

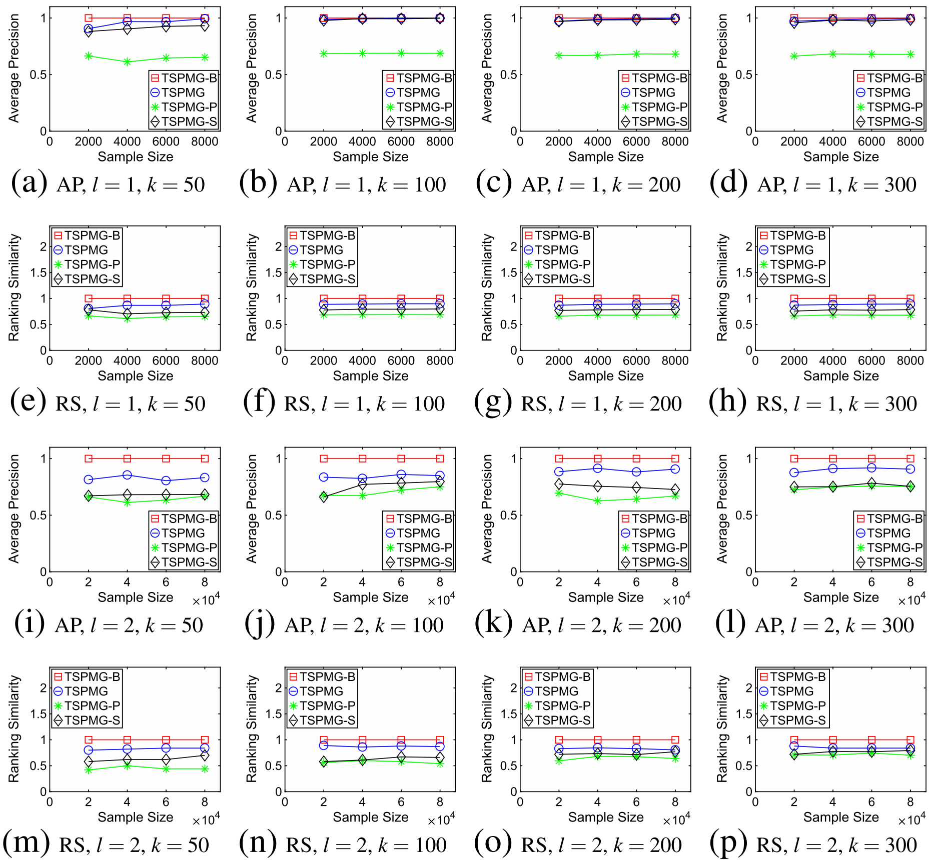
Figure 4 The AP and RS on the SYN1 dataset for $l = 1$ and $l = 2$

method. The exact ranked lists of patterns are used as the ground truth to evaluate the performance of the sampling-based method in AP and RS.

5.4.1 The effectiveness on the synthetic datasets

Figures 4a-p and 5a-p show AP and RS of TSPMG, TSPMG-B, TSPMG-P and TSPMG-S on SYN1 and SYN2 for different path lengths and values of k. AP and RS of TSPMG-B are 1.0 as TSPMG-B produces the exact results, while the other methods work with sampling methods and thus their values of AP and RS are lower than 1.0.

For SYN1, when $l = 1$, TSPMG, TSPMG-P and TSPMG-S have similar AP and RS. The reason is that, in the first step of our two-step sampling method, the path sampling of TSPMG, TSPMG-P and TSPMG-S when $l = 1$ is to sample edges uniformly. And in the second step, since each vertex of the graph has the same number of transactions, the sampled transaction sequences have the same weights. Thus, TSPMG, TSPMG-P and TSPMG-S can achieve similar performances. Similarly, when $l = 2$, TSPMG and TSPMG-S have similar AP and RS but TSPMG-P has lower AP and RS than TSPMG and TSPMG-S. The reason is that, TSPMG-P uses random walks rather than uniform sampling as the path sampling



(a) AP, $l = 1, k = 50$ (b) AP, $l = 1, k = 100$ (c) AP, $l = 1, k = 200$ (d) AP, $l = 1, k = 300$

(e) RS, $l = 1, k = 50$ (f) RS, $l = 1, k = 100$ (g) RS, $l = 1, k = 200$ (h) RS, $l = 1, k = 300$

(i) AP, $l = 2, k = 50$ (j) AP, $l = 2, k = 100$ (k) AP, $l = 2, k = 200$ (l) AP, $l = 2, k = 300$

(m) RS, $l = 2, k = 50$ (n) RS, $l = 2, k = 100$ (o) RS, $l = 2, k = 200$ (p) RS, $l = 2, k = 300$

Figure 5 The AP and RS on the SYN2 dataset for $l = 1$ and $l = 2$

Table 3 ME of pattern frequency in dataset SYN1

$l = 1$	$\|S_1\|$	2,000	4,000	6,000	8,000
	ME	2.56×10^{-3}	6.57×10^{-4}	6.15×10^{-4}	5.79×10^{-4}
$l = 2$	$\|S_2\|$	20,000	40,000	60,000	80,000
	ME	1.79×10^{-4}	1.53×10^{-4}	1.46×10^{-4}	1.06×10^{-4}

method. For SYN2, when $l = 1, 2$, TSPMG has higher AP and RS than both TSPMG-P and TSPMG-S, as different vertices in SYN2 have different number of transactions. This verifies the effectiveness of Algorithm 4.

In summary, when $l = 1$ and each vertex has the same number of transactions, TSPMG, TSPMG-P and TSPMG-S can achieve similar performances for mining top-k sequential patterns in transaction database graphs. For other path lengths, when each vertex has the same number of transactions, TSPMG and TSPMG-S can achieve similar performances. In addition, in other cases, only TSPMG can achieve a good approximation. We thus can conclude that, TSPMG is robust and can accurately approximate the ranked list of patterns with varying path lengths and varying number of transactions on each vertex.

Tables 3 and 4 show the ME at rank-50 of the sampling-based method in SYN1 and SYN2, respectively. For both $l = 1$ and $l = 2$, the ME of the sampling-based method monotonically decreases when the sample size $|S_l|$ increases. This is because, according to Theorems 6 and 7, a larger sample size leads to a smaller estimation error of the pattern frequencies.

As shown in Table 3, the sampling method achieves an estimation error of 2.56×10^{-3} with a sample size $|S_l| = 2,000$, which is much smaller than the bound of $|S_l|$ given in Theorem 7. The difference between the bound of $|S_l|$ and the practical sample size is not really surprising, because the bound in Theorem 7 is an upper bound of sample complexity for the proposed sampling-based algorithm.

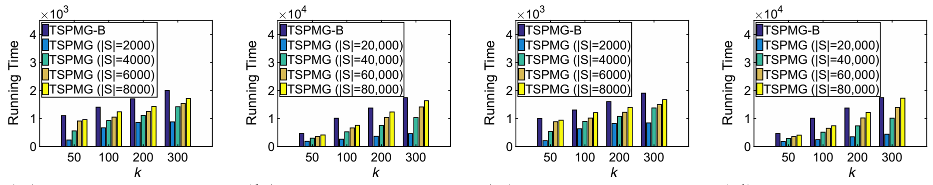
5.4.2 The efficiency on the synthetic datasets

Figure 6 shows the total time cost with different values of k on the SYN1 and SYN2 datasets for $l = 1$ and $l = 2$. As TSPMG has achieved much better AP and RS than TSPMG-P and TSPMG-S, we only report the time cost of TSPMG. For the same value of k, since a larger sample size leads to a larger time cost in transaction sequence sampling and sequential pattern mining, the time cost of TSPMG increases when the sample size increases. For different values of k, the time cost of TSPMG increases when k increases, since a larger k leads to a larger time cost in mining stage. We can also observe that TSPMG can achieve a speedup against the baseline TSPMG-B, especially when the sample size is small.

In summary, on the synthetic datasets TSPMG significantly reduces the number of required transaction sequences and achieves good approximation quality.

Table 4 ME of pattern frequency in dataset SYN2

$l = 1$	$\|S_1\|$	2,000	4,000	6,000	8,000
	ME	2.89×10^{-3}	2.37×10^{-3}	2.27×10^{-3}	1.33×10^{-3}
$l = 2$	$\|S_2\|$	20,000	40,000	60,000	80,000
	ME	2.07×10^{-4}	1.87×10^{-4}	1.66×10^{-4}	1.43×10^{-4}



(a) TTC, SYN1, $l = 1$ (b) TTC, SYN1, $l = 2$ (c) TTC, SYN2, $l = 1$ (d) TTC, SYN2, $l = 2$

Figure 6 The TTC, which is the sum of STC and MTC, with different values of k on the SYN1 and SYN2 datasets for $l = 1$ and $l = 2$. $|S|$ denotes the sample size

5.5 Results on real-world datasets

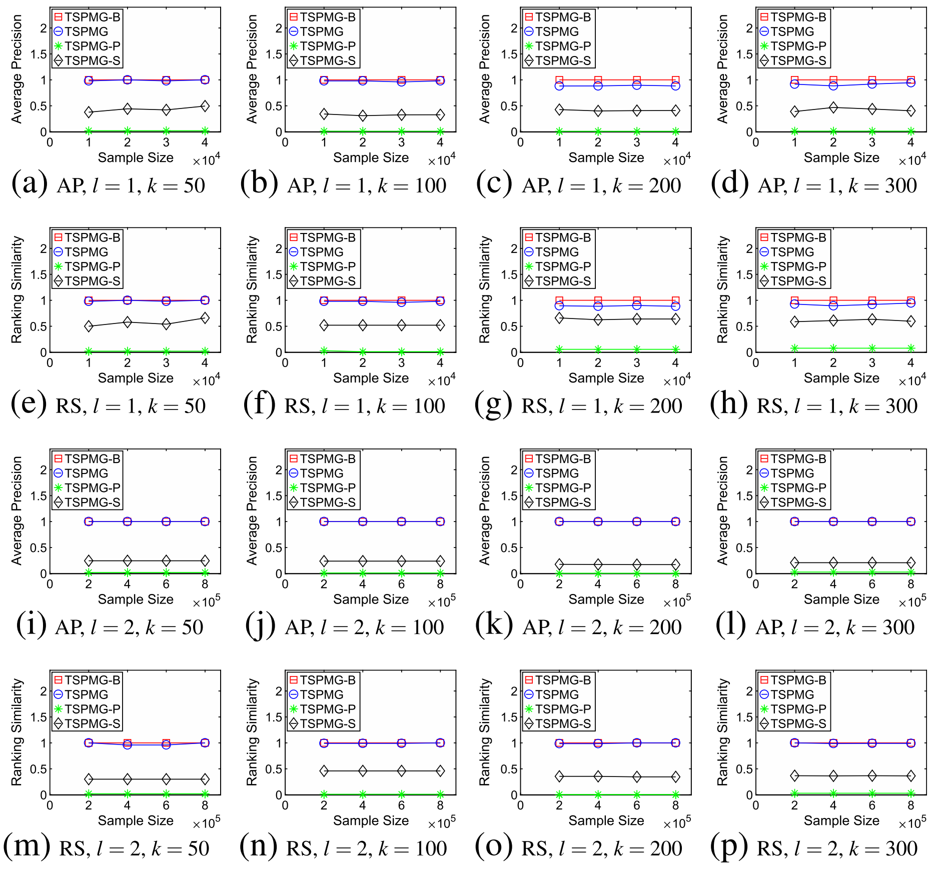
In this section, we introduce the experimental results on the real-world datasets. We focus on analyzing how AP, RS, time cost and space cost of TSPMG-B and TSPMG change when the sample size $|S_l|$ and the values of k increase. Since TSPMG-B is computationally expensive, we use the small real-world dataset Flight when $l = 1, 2$ to demonstrate the performances of TSPMG-B and TSPMG, and test the scalability of our method on Flight, CN1 and CN2. To test the stability of TSPMG, for CN1 and CN2, we use the ranked list generated by TSPMG using a large sample size $|S_l| = 10 \times 10^5$ as the pseudo ground truth to evaluate AP and RS of TSPMG using different sample sizes.

5.5.1 The effectiveness on real-world datasets

Figure 7a-p show AP and RS of TSPMG, TSPMG-B, TSPMG-P and TSPMG-S on Flight. For different path lengths and values of k, TSPMG has higher AP and RS than both TSPMG-P and TSPMG-S. This shows that, TSPMG can achieve a more accurate approximated ranked list of top-k sequential patterns than TSPMG-P and TSPMG-S.

Figure 8a-h show AP and RS of TSPMG on CN1 and CN2. For both $l = 1$ and $l = 2$, when the sample size increases, AP and RS approach 1 quickly. This demonstrates that the estimated ranked list of patterns becomes more accurate when the sample size increases. Please note that, when k increases, AP and RS may not increase. The reason is that, the patterns that locate at the tail of the ranked list commonly have similar frequencies and the differences between the frequencies of these patterns may be small. According to Theorem 8, given a specific sample size, the error bound ε may be larger than the differences. This may result in a problem that some patterns are incorrectly put into the top-k ranked list, making AP and RS decrease.

We can also see that, for $l = 1$ and $l = 2$, when the sample size increases, AP and RS of TSPMG converge faster on CN1 than that on CN2. The reason is that, when the size of the dataset increases, the newly added paths may not support a specific pattern s. Thus, although the total number of transaction sequences in the graph increases, the number of sequences containing s may not increase. This makes the frequency of s decrease and the average frequency of top-k patterns in CN2 is smaller than that in CN1. Consequently, according to Theorem 8, to achieve the same quality of results, the error bound ε for CN2 should be smaller than that for CN1. We thus need a larger sample size in CN2 than that in CN1.



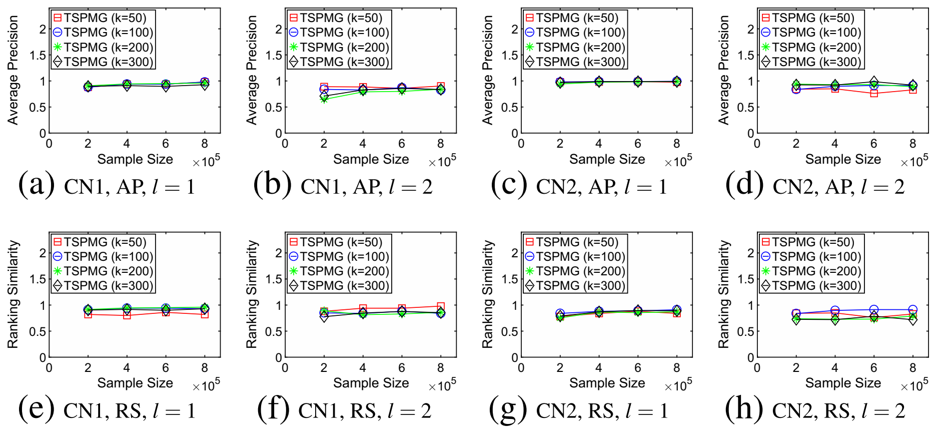
(a) AP, $l = 1$, $k = 50$ (b) AP, $l = 1$, $k = 100$ (c) AP, $l = 1$, $k = 200$ (d) AP, $l = 1$, $k = 300$

(e) RS, $l = 1$, $k = 50$ (f) RS, $l = 1$, $k = 100$ (g) RS, $l = 1$, $k = 200$ (h) RS, $l = 1$, $k = 300$

(i) AP, $l = 2$, $k = 50$ (j) AP, $l = 2$, $k = 100$ (k) AP, $l = 2$, $k = 200$ (l) AP, $l = 2$, $k = 300$

(m) RS, $l = 2$, $k = 50$ (n) RS, $l = 2$, $k = 100$ (o) RS, $l = 2$, $k = 200$ (p) RS, $l = 2$, $k = 300$

Figure 7 The AP and RS on the Flight dataset for $l = 1$ and $l = 2$



(a) CN1, AP, $l = 1$ (b) CN1, AP, $l = 2$ (c) CN2, AP, $l = 1$ (d) CN2, AP, $l = 2$

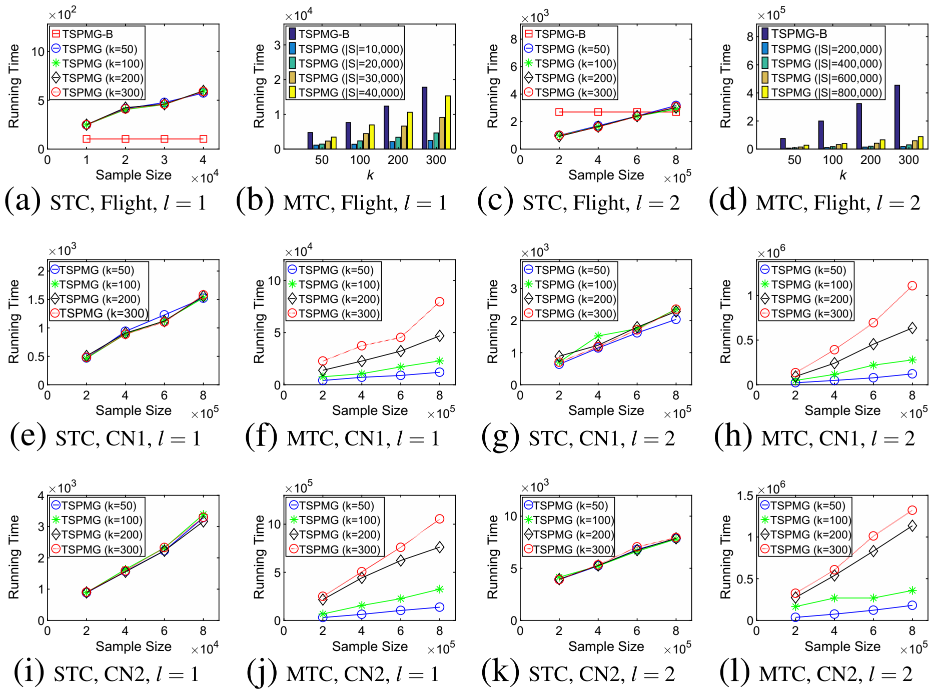
(e) CN1, RS, $l = 1$ (f) CN1, RS, $l = 2$ (g) CN2, RS, $l = 1$ (h) CN2, RS, $l = 2$

Figure 8 The AP and RS on the CN1 and CN2 datasets for $l = 1$ and $l = 2$

5.5.2 The efficiency on real-world datasets

Figure 9a-l show the time cost of sampling and mining process on Flight, CN1 and CN2 for $l = 1$ and $l = 2$ respectively. As TSPMG has achieved much better AP and RS than TSPMG-P and TSPMG-S, we only report the time cost of TSPMG. For both $l = 1$ and $l = 2$, the sampling time increases almost linearly when the sample size increases. This demonstrates the superior scalability of the proposed sampling-based method. Note that, given the same path length and sample size, the time cost is almost the same for different values of k. This shows that, the sampling time is independent of the values of k. As shown in Figure 9a and c, the sampling time may be larger than the time of enumerating all transaction sequences (i.e., TSPMG-B). This is because, the time complexity of transaction sequence enumeration is $\mathcal{O}(|V|^{(l+1)}C^{(l+1)})$, while that of sampling is $\mathcal{O}(2|S_l|l)$. Thus, due to the small size of Flight, the values of $|V|$ and C are also small but $|S_l|$ is relatively large, making the sampling-based method cost more time than transaction sequence enumeration.

As shown in Figure 9, the mining time increases when the sample size increases. Note that, the mining time of TSPMG-B is larger than TSPMG, as TSPMG-B uses all the transaction sequences in the graph, which is always larger than the sample size. This shows that, the mining time is positively correlated to the number of transaction sequences. The mining time also increases when the value of k increases. The reason is that, when k increases, it needs more time to prune unqualified patterns and reserve the top-k ones. When l increases, the sampling time and the mining time increase. We now give an example to illustrate that, given a small input k, how many times Algorithm 4 can achieve a speedup. As shown in Figure 9d, h and l, given $l = 2$ and $|S_2| = 8 \times 10^5$, the mining time on Flight, CN1 and

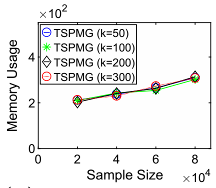


(a) STC, Flight, $l = 1$ (b) MTC, Flight, $l = 1$ (c) STC, Flight, $l = 2$ (d) MTC, Flight, $l = 2$

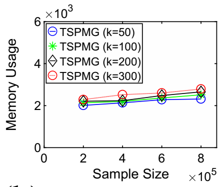
(e) STC, CN1, $l = 1$ (f) MTC, CN1, $l = 1$ (g) STC, CN1, $l = 2$ (h) MTC, CN1, $l = 2$

(i) STC, CN2, $l = 1$ (j) MTC, CN2, $l = 1$ (k) STC, CN2, $l = 2$ (l) MTC, CN2, $l = 2$

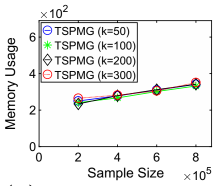
Figure 9 The STC and MTC on CN1 and CN2 datasets for $l = 1$ and $l = 2$. Note that we omit the results of TSPMG-B on CN1 and CN2 as it cannot accomplish the task within a feasible time limit



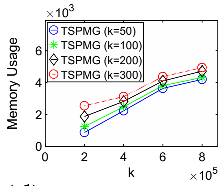
(a) SSC, CN1, $l = 1$



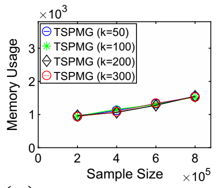
(b) MSC, CN1, $l = 1$



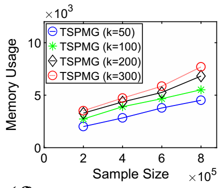
(c) SSC, CN1, $l = 2$



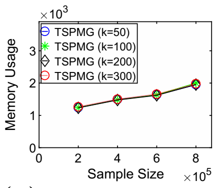
(d) MSC, CN1, $l = 2$



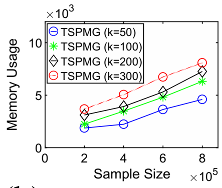
(e) SSC, CN2, $l = 1$



(f) MSC, CN2, $l = 1$



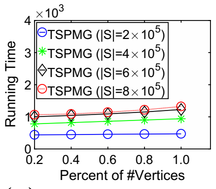
(g) SSC, CN2, $l = 2$



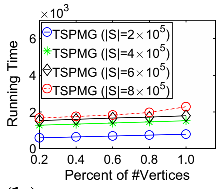
(h) MSC, CN2, $l = 2$

Figure 10 The SSC and MSC on the CN1 and CN2 datasets for $l = 1$ and $l = 2$

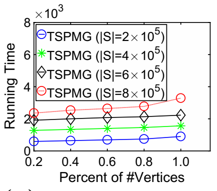
CN2 when $k = 50$ is 4.82, 9.11 and 7.32 times faster than that when $k = 300$. Moreover, on Flight, when $k = 50$ and $|S_2| = 8 \times 10^5$, the mining time of TSPMG is 11.1 times faster that of TSPMG-B.



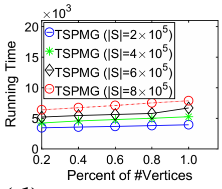
(a) STC, CN1, $l = 1$



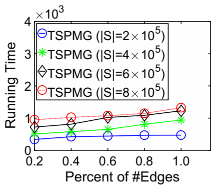
(b) STC, CN1, $l = 2$



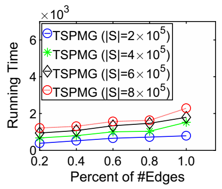
(c) STC, CN2, $l = 1$



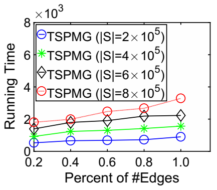
(d) STC, CN2, $l = 2$



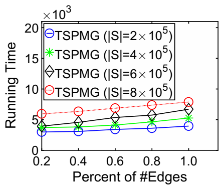
(e) STC, CN1, $l = 1$



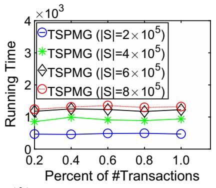
(f) STC, CN1, $l = 2$



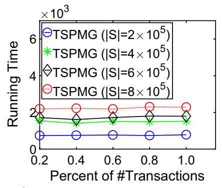
(g) STC, CN2, $l = 1$



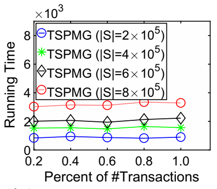
(h) STC, CN2, $l = 2$



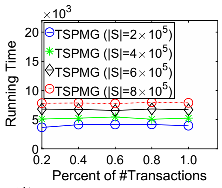
(i) STC, CN1, $l = 1$



(j) STC, CN1, $l = 2$



(k) STC, CN2, $l = 1$



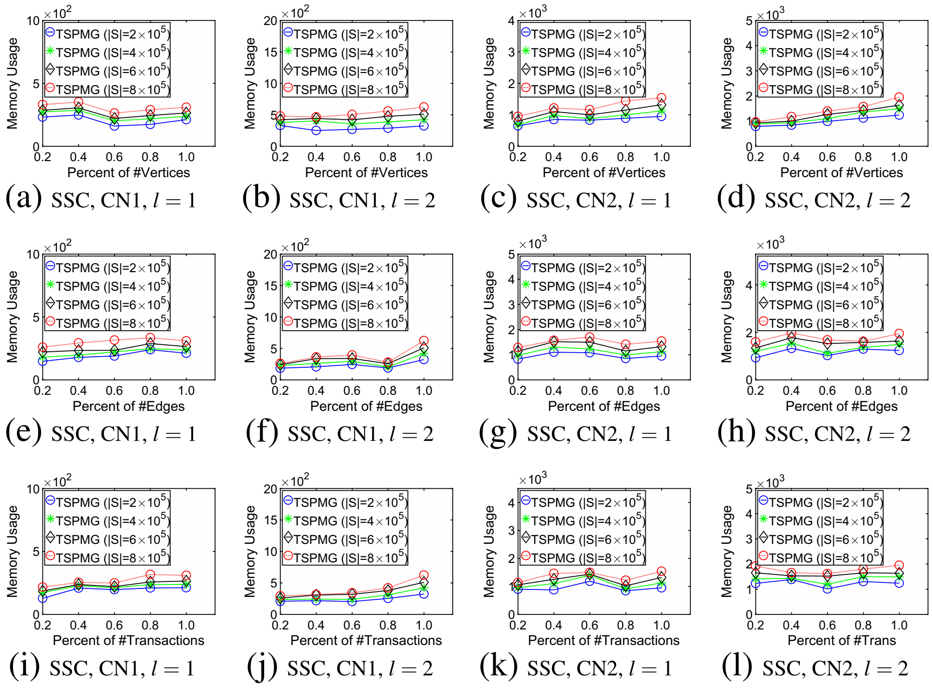
(l) STC, CN2, $l = 2$

Figure 11 The STC with different percents of total numbers of vertices, edges and transactions of each vertex on the CN1 and CN2 datasets for $l = 1$ and $l = 2$, where #Vertices, #Edges, #Transactions denote the total number of vertices, edges and transactions of each vertex respectively. $|S|$ denotes the sample size

Figure 10a-h show the space cost of sampling and mining process on CN1 and CN2 for $l = 1$ and $l = 2$ respectively. As TSPMG has achieved much better AP and RS than TSPMG-P and TSPMG-S, we only report the space cost of TSPMG. For both $l = 1$ and $l = 2$, the space cost of the sampling process increases almost linearly when the sample size increases. The reason is that, TSPMG needs larger main memory to store the sampled transaction sequences. We also observe that, given the same path length, for different values of k, the space cost is almost the same as the sampling process is independent of the values of k. The space cost of mining process increases when the sample size increases, and it also increases when the value of k increases. This is because, when k increases, it needs more space to store the candidate patterns. When l increases, the space cost increases as the lengths of transaction sequences increases.

Figure 11 shows the time cost of the sampling process (i.e., Algorithm 4) with different numbers of vertices, edges and transactions of each vertex on the CN1 and CN2 datasets when $l = 1$ and $l = 2$. Given the same percent of vertices (similarly, edges or transactions), the time cost increases when the sample size increases. This is because a larger sample size leads to longer time to obtain and store the sampled transaction sequences. Given the same sample size, when the number of vertices or edges increases, the time cost increases as it needs more time to compute the probability according to (2). However, the time cost with different numbers of transactions on each vertex is almost the same. The reason is that, according to the time complexity analysis of Algorithm 4, $|E|$, m and l are independent of the number of transactions of each vertex.

Figure 12 shows the space cost of the sampling process (i.e., Algorithm 4) with different numbers of vertices, edges and transactions of each vertex on the CN1 and CN2 datasets



(a) SSC, CN1, $l = 1$ (b) SSC, CN1, $l = 2$ (c) SSC, CN2, $l = 1$ (d) SSC, CN2, $l = 2$

(e) SSC, CN1, $l = 1$ (f) SSC, CN1, $l = 2$ (g) SSC, CN2, $l = 1$ (h) SSC, CN2, $l = 2$

(i) SSC, CN1, $l = 1$ (j) SSC, CN1, $l = 2$ (k) SSC, CN2, $l = 1$ (l) SSC, CN2, $l = 2$

Figure 12 The SSC with different percents of total numbers of vertices, edges and transactions of each vertex on the CN1 and CN2 datasets for $l = 1$ and $l = 2$, where #Vertices, #Edges, #Transactions denote the total number of vertices, edges and transactions on each vertex respectively. $|S|$ denotes the sample size

for $l = 1$ and $l = 2$. Given the same percent of vertices (similarly, edges or transactions), the space cost increases when the sample size increases, as a larger sample size leads to larger space cost to obtain and store the sampled transaction sequences. However, given the same sample size, when the number of vertices, edges and transactions of each vertex increases, the space cost may not increase. This is because, the space complexity is related to the sample size m, the path length l and the average number of items for each transaction $|X|_{ave}$. As the sample size and the path length are the same, the space cost mainly depends on $|X|_{ave}$. However, each sampling process may obtain different values of $|X|_{ave}$, and larger $|X|_{ave}$ leads to more space cost.

In summary, TSPMG can obtain accurate top-k sequential patterns when the sample size is not too small, and is much more scalable than the exact method in mining frequent sequential patterns from large graphs.

5.6 A case study

In this section, we report a case study on the CN1 dataset. Mining frequent sequential patterns in CN1 finds interesting patterns of topic relations in paper citations.

Tables 5 and 6 show some interesting patterns obtained by the sampling-based method with $l = 1$ and $l = 2$, respectively. Those frequent patterns show significant trends of cross-field citations, which reveal active interdisciplinary research hotspots among different research fields. The activeness of those research hotspots is also confirmed by *the number of related publications* (#RP) obtained by searching *Google Scholar* (*scholar.google.com*) using the keywords in the mined sequential patterns. Specifically, the patterns in Tables 5 and 6 are sorted by the estimated pattern frequencies obtained by our sampling-based method. At the same time, the ranked lists are highly consistent with descending order in #RP value. This consistency demonstrates that the results produced by the sampling-based method matches the real world patterns, and verifies the effectiveness of mining sequential patterns in transaction database graphs.

We also look into the specific patterns and find they are meaningful. Take ⟨(deep learning), (reinforcement learning)⟩ as an example. By extracting from Google Scholar all paper citations related to this pattern, we find that many deep learning papers cite reinforcement learning papers in recent years. As we know, deep reinforcement learning has been one of the hottest research trends in the field of deep learning.

Table 5 Some interesting patterns in CN1 ($l = 1$)

Sequential Patterns	#RP
⟨(machine learning), (social network)⟩	21,200
⟨(social network), (random walk)⟩	12,900
⟨(deep learning), (clustering)⟩	12,600
⟨(graph embedding), (classification)⟩	5,390
⟨(social network), (anomaly detection)⟩	4,920
⟨(deep learning), (reinforcement learning)⟩	3,870
⟨(graph partitioning), (community detection)⟩	3,500
⟨(dynamic network), (game theory)⟩	3,340
⟨(deep learning), (anomaly detection)⟩	1,430
⟨(network evolution), (game theory)⟩	1,170

Table 6 Some interesting patterns in CN1 ($l = 2$)

Sequential Patterns	#RP
⟨(data mining), (machine learning), (information retrieval)⟩	37,700
⟨(machine learning), (social network), (approximation algorithm)⟩	10,900
⟨(data mining), (machine learning, social network), (web search)⟩	7,450
⟨(deep learning), (clustering), (prediction)⟩	7,430
⟨(dynamic network), (game theory), (clustering)⟩	4,410
⟨(social network), (random walk), (recommendation)⟩	4,180
⟨(social network), (anomaly detection), (classification)⟩	3,240
⟨(graph embedding), (classification), (clustering)⟩	2,980
⟨(deep learning), (reinforcement learning), (classification)⟩	2,660
⟨(social network), (anomaly detection), (pattern mining)⟩	518

The relationship between machine learning and social network, captured by some patterns found, is also interesting. In Table 5, the top pattern ⟨(machine learning), (social network)⟩ reveals the hot research trend of applying machine learning algorithms to solve social network problems. However, it is an open problem that classic machine learning algorithms are not scalable enough to handle large scale social networks, thus many researchers use approximation techniques to improve the scalability of machine learning algorithms on social networks. This trend is captured by pattern ⟨(machine learning), (social network), (approximation algorithm)⟩ in Table 6.

6 Conclusion

In this paper, we tackled the novel problem of finding top-k sequential patterns in transaction database graphs. We designed a fast sampling-based top-k sequential pattern mining algorithm. Our experiments on both synthetic data sets and real data sets showed the superior effectiveness and efficiency of the proposed sampling method in finding meaningful sequential patterns. As future work, we will further improve the scalability and extend the sampling method to handle dynamic graphs.

Acknowledgements This work was supported in part by the National Key Research and Development Program of China (No. 2017YFB0803301), the Natural Science Foundation of China (No. U1836215), DongGuan Innovative Research Team Program (No.201636000100038), and the 111 Project (No. B18008).

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the 11th International Conference on Data Engineering, ICDE'95, pp. 3–14 (1995)
2. Bartlett, P.L., Boucheron, S., Lugosi, G.: Model selection and error estimation. Mach. Learn. **48**(1-3), 85–113 (2002)
3. Bonferroni, C.E.: Teoria statistica delle classi e calcolo delle probabilita. Libreria internazionale Seeber (1936)
4. Calders, T., Garboni, C., Goethals, B.: Efficient pattern mining of uncertain data with sampling. In: Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD'10, pp. 480–487 (2010)

5. Cherkassky, V.: The nature of statistical learning theory. IEEE Trans. Neural Netw. **8**(6), 1564 (1997)
6. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. Math. Stat. **23**(4), 493–507 (1952)
7. Cochran, W.G. Sampling techniques, 3rd. Wiley, New York (1977)
8. Dong, G., Pei, J.: Sequence data mining. Springer, Berlin (2007)
9. Dutta, S., Nayek, P., Bhattacharya, A.: Neighbor-aware search for approximate labeled graph matching using the chi-square statistics. In: Proceedings of the 26th International Conference on World Wide Web, WWW'17, pp. 1281–1290 (2017)
10. Fournier-Viger, P., Gomariz, A., Gueniche, T., Mwamikazi, E.T.: Tks: efficient mining of top-k sequential patterns. In: Proceedings of the 9th International Conference on Advanced Data Mining and Applications, ADMA'13, pp. 109–120 (2013)
11. Ge, J., Xia, Y.: Distributed sequential pattern mining in large scale uncertain databases. In: Proceedings of the 20th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD'16, pp. 17–29 (2016)
12. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: Freespan: Frequent pattern-projected sequential pattern mining. In: Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'00, pp. 355–359 (2000)
13. Huang, D., Xu, K., Pei, J.: Malicious url detection by dynamically mining patterns without pre-defined elements. World Wide Web Journal **17**(6), 1375–1394 (2014)
14. Kimura, M., Saito, K.: Tractable models for information diffusion in social networks. In: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD'06, pp. 259–271 (2006)
15. Leskovec, J., Faloutsos, C.: Neighbor-aware search for approximate labeled graph matching using the chi-square statistics. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'06, pp. 631–636 (2006)
16. Liu, C., Zhang, K., Xiong, H., Jiang, G., Yang, Q.: Temporal skeletonization on sequential data: Patterns, categorization, and visualization. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'14, pp. 1336–1345 (2014)
17. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: mining sequential patterns by prefix-projected growth. In: Proceedings of the 17th International Conference on Data Engineering, ICDE'01, pp. 215–224 (2001)
18. Pfeiffer, J.J., Moreno, S., Fond, T.L., Neville, J., Gallagher, B.: Attributed graph models: modeling network structure with correlated attributes. In: Proceedings of the 23rd International Conference on World Wide Web, WWW'14, pp. 831–842 (2014)
19. Pietracaprina, A., Riondato, M., Upfal, E., Vandin, F.: Mining top-k frequent itemsets through progressive sampling. Data Min. Knowl. Disc. **21**(2), 310–326 (2010)
20. Raïssi, C., Poncelet, P.: Sampling for sequential pattern mining: From static databases to data streams. In: Proceedings of the 7th IEEE International Conference on Data Mining, ICDM'07, pp. 631–636 (2007)
21. Ribeiro, B.F., Wang, P., Murai, F., Towsley, D.: Sampling directed graphs with random walks. In: Proceedings of the IEEE International Conference on Computer Communications, INFOCOM'12, pp. 1692–1700 (2012)
22. Riondato, M., Upfal, E.: Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD'12, pp. 25–41 (2012)
23. Riondato, M., Upfal, E.: Mining frequent itemsets through progressive sampling with rademacher averages. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'15, pp. 1005–1014 (2015)
24. Shang, J., Peng, J., Han, J.: Macfp: maximal approximate consecutive frequent pattern mining under edit distance. In: Proceedings of the 2016 SIAM International Conference on Data Mining, SDM'16, pp. 558–566 (2016)
25. Singhal, A.: Modern information retrieval: a brief overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **24**(4), 35–43 (2001)
26. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the 5th International Conference on Extending Database Technology, EDBT'96, pp. 3–17 (1996)
27. Tang, J., Zhang, J., Yao, L., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'08, pp. 990–998 (2008)
28. Thompson, S.K. Sampling, 3rd. Wiley, New York (2012)

29. Toivonen, H.: Sampling large databases for association rules. Proceedings of the Vldb Endowment **96**, 134–145 (1996)
30. Tong, H., Faloutsos, C., Gallagher, B., Eliassi-Rad, T.: Fast best-effort pattern matching in large attributed graphs. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'07, pp. 737–746 (2007)
31. Tzvetkov, P., Yan, X., Han, J.: Tsp: mining top-k closed sequential patterns. Knowl. Inf. Syst. **7**(4), 438–457 (2005)
32. Wang, X., Lin, J., Senin, P., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S.: Rpm: representative pattern mining for efficient time series classification. In: Proceedings of the 19th International Conference on Extending Database Technology, EDBT'16, pp. 185–196 (2016)
33. Ye, W., Zhou, L., Mautz, D., Plant, C., Böhm, C.: Learning from labeled and unlabeled vertices in networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'17, pp. 1265–1274 (2017)
34. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1/2), 31–60 (2001)
35. Zhang, J., Tang, J., Ma, C., Tong, H., Jing, Y., Li, J.: Panther: fast top-k similarity search on large networks. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'15, pp. 1445–1454 (2015)
36. Zheng, Z., Wei, W., Liu, C., Cao, W., Cao, L., Bhatia, M.: An effective contrast sequential pattern mining approach to taxpayer behavior analysis. In: Proceedings of the 25th International Conference on World Wide Web, WWW'16, pp. 633–651 (2016)

Affiliations

Mingtao Lei[1] · Lingyang Chu[2] · Zhefeng Wang[3] · Jian Pei[2] · Caifeng He[4] · Xi Zhang[1] · Binxing Fang[1]

Mingtao Lei
leimingtao@bupt.edu.cn

Lingyang Chu
lca117@sfu.ca

Zhefeng Wang
zhefwang@mail.ustc.edu.cn

Jian Pei
jpei@cs.sfu.ca

Caifeng He
hecaifeng@huawei.com

Binxing Fang
fangbx@bupt.edu.cn

[1] Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China

[2] Simon Fraser University, Burnaby, Canada

[3] University of Science and Technology of China, Hefei, China

[4] Noah Ark's Laboratory, Huawei Technologies, Shenzhen, China