PSMSP: A Parallelized Sampling-Based Approach for Mining Top-*k* Sequential Patterns in Database Graphs

Mingtao Lei[1], Xi Zhang[1(✉)], Jincui Yang[1], and Binxing Fang[1,2]

[1] Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China
{leimingtao,zhangx,jincuiyang,fangbx}@bupt.edu.cn

[2] Institute of Electronic and Information Engineering of UESTC in Guangdong, Dongguan, China

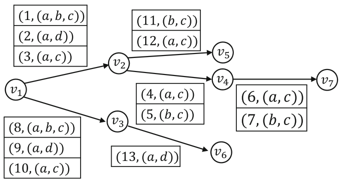
Abstract. We study to improve the efficiency of finding top-k sequential patterns in database graphs, where each edge (or vertex) is associated with multiple transactions and a transaction consists of a set of items. This task is to discover the subsequences of transaction sequences that frequently appear in many paths. We propose **PSMSP**, a Parallelized Sampling-based Approach For Mining Top-k Sequential Patterns, which involves: (a) a parallelized unbiased sequence sampling approach, and (b) a novel PSP-Tree structure to efficiently mine the patterns based on the anti-monotonicity properties. We validate our approach via extensive experiments with real-world datasets.

Keywords: Database graph · Sequential pattern mining · Parallelized sampling

1 Introduction

The problem of sequential pattern mining, which discovers frequent subsequences from a sequence database, has been extensively studied in [3–6]. Motivated by the rich knowledge in the database graph, a related but different problem is proposed [2], which is to perform sequential pattern mining in database graphs, aiming to discover the subsequences that frequently appear in many paths of the graph. It is natural to ask whether we can extend the existing sequential pattern mining methods for this new task. Unfortunately, there is no straightforward way as most of existing methods don't account for a database graph as input. Moreover, the number of transaction sequences induced by all possible

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



(a) G

$\langle(a,b,c)(a,c)(a,c)\rangle$	$\langle(a,c)(a,c)(a,c)\rangle$
$\langle(a,b,c)(a,c)(b,c)\rangle$	$\langle(a,c)(a,c)(b,c)\rangle$
$\langle(a,b,c)(b,c)(a,c)\rangle$	$\langle(a,c)(b,c)(a,c)\rangle$
$\langle(a,b,c)(b,c)(b,c)\rangle$	$\langle(a,c)(b,c)(b,c)\rangle$
$\langle(a,d)(a,c)(a,c)\rangle$	$\langle(a,d)(b,c)(a,c)\rangle$
$\langle(a,d)(a,c)(b,c)\rangle$	$\langle(a,d)(b,c)(b,c)\rangle$

(b) D_3

Fig. 1. An example of a graph G, where each edge is associated with a transaction database. D_3 represents the set of length-3 transaction sequences of the graph G. The first sequence of D_3 is the sequence of transactions with $tid = 1, 4, 6$.

random walks of a database graph is extremely huge, demanding more efficient approaches. Despite the fact that a recent study [2] has proposed to approximate the sequential patterns in a database graph, it still falls short in efficiency.

To fill this gap, we propose to improve the efficiency of finding top-k sequential patterns in database graphs through two-stage optimizations. First, we develop a parallelized sampling method as well as a weighting mechanism for the sampled transaction sequences. Second, we develop a novel Tree of Trees (ToT) structure, PSP-Tree, to mine the approximate top-k patterns by exploring the anti-monotonicity properties.

2 Problem Definition

Let α be an item, I be a set of *items* and an *itemset* X be a subset of I. We assume that items follow an arbitrary fixed order in each itemset. A *transaction* is a tuple $\tau = (tid, X)$, where tid is a unique transaction-id and X is the corresponding itemset.

A *database graph* $G = (V, E, \mathbb{T})$ is directed, where V is a finite set of vertices, $E \subseteq V \times V$ is a set of directed edges and $\mathbb{T} = \{T_e | e \in E\}$ is a set of transaction databases that each transaction database T_e is associated with the edge e.

A *(directed) path* p in G is a sequence of edges $p = \langle e_1, \ldots, e_h \rangle$, where e_i and e_{i+1} $(1 \leq i < h)$ are incident, and the *length* of the path is $len(p) = h$. A *transaction sequence* supported by p, denoted by $ts = \langle \tau_1, \ldots, \tau_h \rangle$, is a sequence of transactions such that there exists a path $p = \langle e_1, \ldots, e_h \rangle$ and $\tau_i \in T_{e_i}$ for $1 \leq i \leq h$. The length of ts is $len(ts) = h$. A *sequential pattern* $s = \langle X_1, \ldots, X_h \rangle$ is a series of itemsets, and it is said to be contained by a transaction sequence $ts = \langle \tau_1, \ldots, \tau_h \rangle$ if $X_i \subseteq \tau_i$ for $1 \leq i \leq h$; i.e., X_i is an itemset of the transaction τ_i, and it is also called the i-th itemset of ts.

Denote D_l as the set of all length-l transaction sequences supported by paths in G. The *support* and *frequency* of a pattern s in D_l, denoted by $sup(s)$ and $f(s)$ respectively, are the number and the proportion of unique transaction sequences in D_l that contain s.

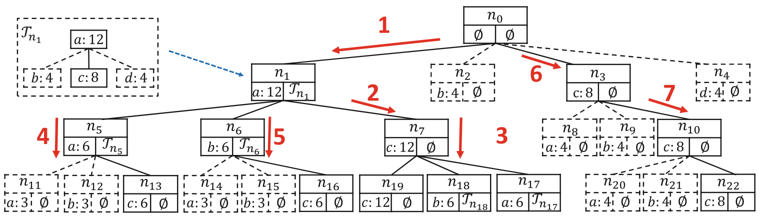


Fig. 2. The PSP-Tree for top-6 length-3 frequent patterns in Fig. 1. $\mathcal{T}_{n_1}$ is a FP-Tree of the node n_1. The numbers in red indicate the mining order and the nodes with dashed lines are pruned during the mining process. (Color figure online)

In this paper, given a database 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.

For example, given G in Fig. 1, the top-6 length-3 patterns when $k = 6$ and $l = 3$ are $\langle(a)(c)(c)\rangle$, $\langle(a,c)(c)(c)\rangle$, $\langle(a)(a)(c)\rangle$, $\langle(a)(b)(c)\rangle$, $\langle(a)(c)(b)\rangle$, and $\langle(a)(c)(a)\rangle$.

3 The Method

In this section, we introduce our PSMSP method, consisting of two stages, the parallelized sampling stage and the PSP-Tree mining stage.

The Parallelized Sampling Stage. We assume that the database graph G has been partitioned into m disjoint groups $\{E_1, \ldots, E_m\}$. Given a partition $\mathcal{G}_i(V_i, E_i)$, which is induced by E_i, and the path length l, denote E^+ as the set of additional edges beyond $\mathcal{G}_i$ that can be reached from an edge $e \in E_i$ through a path whose length is at most l. P_l^+ is the set of all the length-l paths with the starting edges in E_i and the end edges in E^+. $\mathcal{G}_i{}^+(V_{\mathcal{G}_i{}^+}, E_{\mathcal{G}_i{}^+})$ is defined as the l-enhanced partition of $\mathcal{G}_i$, where $V_{\mathcal{G}_i{}^+} = V_i \cup \{v | v \in p' \wedge p' \in P_l^+\}$ and $E_{\mathcal{G}_i{}^+} = E_i \cup \{e | e \in p' \wedge p' \in P_l^+\}$. For p and ts supported by p, we can define their weights as $\varphi(p) = |P_l^i|$ and $\varphi(ts) = \varphi(p)Z$, where $Z = \prod_{q=1}^{l} |T_{e_q}|$ is the number of all transaction sequences supported by p.

The parallelized sampling process is shown as follows. Given the enhanced partition set $\mathcal{G}^+ = \{\mathcal{G}_1^+, \ldots, \mathcal{G}_m^+\}$, we draw μ samples from each partition and the total sample size is thus $m\mu$. For each partition $\mathcal{G}_i^+$, in its h-th iteration, we first sample a path p_h uniformly and calculate its weight $\varphi(p_h)$. We then sample a transaction sequence ts_h from p_h and calculate its weight $\varphi(ts_h)$. ts_h and $\varphi(ts_h)$ are then added into the set S_l. At last, we output S_l, which consists of all the sampled transaction sequences and the corresponding weights.

The PSP-Tree Mining Stage. To mine the top-k sequential patterns efficiently, we propose a novel Tree of Trees (ToT) model, PSP-Tree, denoted by $\mathcal{T}$,

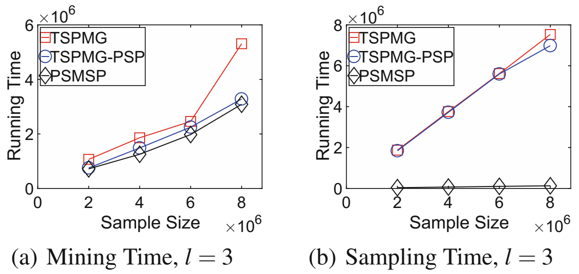


Fig. 3. The mining time and sampling time (in milliseconds). The partition number is 20.

where each node of the main (pattern) tree itself may contain another (itemset-specific) tree. The i-th node of $\mathcal{T}$ is denoted by n_i, which contains an item β_i with its occurrence, and a FP-Tree [4] $\mathcal{T}_{n_i}$. $\mathcal{T}_{n_i}$ is used to represent the itemsets containing β_i with their occurrences, and its root is β_i. The *occurrence* of the q-th itemset X_q is the frequency of transaction sequences containing X_q. With the pre-computed occurrence of β_i, we can determine whether to build n_i in the main tree by comparing this occurrence to the k-th largest frequency of patterns. For clarity, $\mathcal{T}$ is called a main tree and $\mathcal{T}_{n_i}$ is called a itemset-specific tree. When we generate the PSP-Tree, we first generate all qualified nodes for the main tree and then build corresponding itemset-specific trees. This enables us prune unqualified patterns earlier.

The itemsets of each node n_i can be easily extracted from $\mathcal{T}_{n_i}$. For the example in Fig. 2, the main tree has 12 nodes shown in the solid lines. n_1 contains an item a and $\mathcal{T}_{n_1}$, and the occurrence of a is 12. The itemsets that are derived from n_1 are $\{(a), (a, c)\}$. n_5 is a child of n_1 and n_{13} is a child of n_5. The patterns can be obtained by traversing the PSP-Tree in a top-down manner. For example, traversing n_1, n_5 and n_{13} sequentially can obtain the patterns $\langle(a)(a)(c)\rangle$ and $\langle(a, c)(a)(c)\rangle$.

However, the number of remaining patterns in the main tree may be still larger than k. Thus, for all the remaining patterns in the tree, we compute their supports and select top-k patterns that have the largest supports.

4 Experiments

We evaluate the performance of **PSMSP** (using parallelized sampling and PSP-Tree), **TSPMG** [2] (using serial sampling and PrefixSpan) and **TSPMG-PSP** (using serial sampling and PSP-Tree) on the collaboration network [1]. We set $l = 3$ and $k = 100$.

Figure 3(a) shows TSPMG-PSP and PSMSP have similar mining time, as they all adopt the PSP-Tree in the mining stage. In addition, they all run much faster than TSPMG as TSPMG uses the slow PrefixSpan rather than PSP-Tree.

Figure 3(b) shows the sampling time cost increases when the sample size increases due to the larger time cost in sampling processing. TSPMG-PSP is faster than TSPMG due to the adoption of the efficient PSP-Tree. In summary, compared to other sampling baselines, our method PSMSP can further improve the efficiency.

References

1. DBLP. http://dblp.uni-trier.de/
2. Lei, M., Chu, L., Wang, Z.: Mining top-k sequential patterns in database graphs: a new challenging problem and a sampling-based approach. arXiv preprint arXiv:1805.03320 (2018)
3. Li, H., Yi, W., Dong, Z., Ming, Z., Edward, Y.C.: PFP: parallel FP-growth for query recommendation. In: RecSys, pp. 107–114 (2008)
4. Pei, J., et al.: PrefixSpan: mining sequential patterns by prefix-projected growth. In: ICDE, pp. 215–224 (2001)
5. Riondato, M., Upfal, E.: Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. In: ECML PKDD, pp. 25–41 (2012)
6. Riondato, M., Upfal, E.: Mining frequent itemsets through progressive sampling with rademacher averages. In: KDD, pp. 1005–1014 (2015)