Length-Frequent Pattern Mining from Graph Traversals

Hyu Chan Park
Department of Computer Engineering, Korea Maritime and Ocean University, Korea

Summary
Data mining is to discover valuable patterns from large data set, such as item sets and graph traversals. This paper focuses on the graph traversal, which is a sequence of vertices along edges on a graph. Although there were a few works on the graph traversals, they considered mainly the frequency of patterns. This paper extends them by considering the length of patterns as well as frequency. Under such length settings, traditional mining algorithms can not be adopted directly any more. To cope with the problem, this paper proposes new algorithm by adopting the notion of support bound.

Key words:
Data mining, Graph traversal, Length-frequent pattern

1. Introduction

In the pattern mining problem, the value of pattern may be defined in several ways. Although the simplest and traditional one is the frequency of pattern, more valuable one may be defined in other ways. The mining problem this paper focus is defined as follows. Given a set of traversals on a graph, discover all the valuable patterns contained in the traversals, in which the value is measured by the combination of length and frequency. Main issue on such mining problem is how to generate candidates, from which solutions can be obtained. Another issue is how to keep the number of candidates as small as possible.
To cope with these kinds of issues, there have been some works. Chen et al. [1] defined the problem of traversal pattern mining, and then proposed algorithms with hashing and pruning techniques. However, they did not consider graph structure, on which the traversals occur. Nanopoulos et al. [2, 3] proposed the problem of mining patterns from graph traversals. They defined new criteria for the support and subpath containment, and then proposed algorithms with a trie structure. Although they considered the graph on which traversals occur, they did not consider weight. Lee and Park [4] extended the graph traversal problem to the weight settings, in which weights are attached to the vertices of graph. Such vertex weight may reflect the importance of vertex. For example, each Web page may have different importance which reflects the value of its content. The mining algorithms for these kinds of problems cannot be relied on the well-known Apriori algorithm [5] any more. Instead, the notion of support bound [6] had been adopted.

Manuscript received December 5, 2016
Manuscript revised December 20, 2016

This paper further extends the problem of graph traversal mining by considering the length of patterns. To cope with this extension, we basically adopt the previous paradigm in [4] as a whole, but revise its definitions and approaches. Although overall foundation is similar to the previous one, details are somewhat different.
This paper is organized as follows. Section 2 defines the problem of length-frequent pattern mining. In Section 3, we propose a mining algorithm based on the notion of support bound. Section 4 includes two approaches for the estimation of support bound used in the mining process. In Section 5, we experiment and analyze the approaches on synthetic data. Finally, Section 6 contains conclusion and future works.

2. Length-frequent Pattern

Definition 1. Directed graph is a finite set of vertices and edges, in which each edge joins an ordered pair of vertices. *Base graph* is a directed graph, on which traversals occur.

For example, the following base graph has 6 vertices and 8 edges.

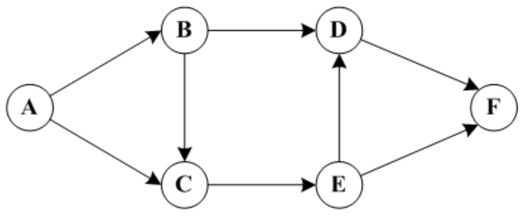

Fig. 1 Example of base graph

Definition 2. A *traversal* is a sequence of consecutive vertices along a sequence of edges on a base graph. We assume that every traversal is path, which has no repeated vertices and edges. The *length* of a traversal is the number of vertices in the traversal. A *traversal database* is a set of traversals.

We restrict any traversal to be a path, because repeated vertices or edges in a traversal may not contain useful information in many cases, such as backward movements. If a traversal has repeated vertices or edges, it can be separated into several paths, such as maximal forward references [1]. The following traversal database has totally 6 traversals, each of which has an identifier and a sequence of consecutive vertices.

Tid	Traversal
1	<A>
2	<A, B>
3	<A, C>
4	<B, C, E>
5	<B, C, E, F>
6	<A, C, E, D>

Fig. 2 Example of traversal database

Definition 3. *Subtraversal* is any subsequence of consecutive vertices in a traversal. If a pattern *P* is a subtraversal of a traversal *T*, then we say that *P* is *contained* in *T*, and vice versa *T contains P*.

For example, given a traversal of length 4, <B, C, E, F>, there are only two subtraversals of length 3, <B, C, E> and <C, E, F>. Note that non-consecutive sequences, such as <B, C, F>, are not subtraversals.

Definition 4. The *support count* of a pattern P_k with length *k*, denoted by $scount(P_k)$, is the number of traversals containing the pattern. The *support* of a pattern P_k, denoted by $support(P_k)$, is the fraction of traversals containing the pattern. Given a traversal database *D*, let $|D|$ be the number of traversals.

$$support(P_k) = \frac{scount(P_k)}{|D|} \qquad (1)$$

There is a well-known property on such support count and support as follows.

Property 1. The support count and the support of a pattern decrease monotonically as the length of the pattern increases. In other word, given a *k*-pattern P_k and any *l*-pattern containing P_k, denoted by (P_k, l), where $l > k$, then $scount(P_k) \geq scount(P_k, l)$ and $support(P_k) \geq support(P_k, l)$.

Definition 5. The *length-support* of a pattern P_k, denoted by $lsupport(P_k)$, is

$$\begin{aligned} lsupport(P_k) &= length(P_k) \times support(P_k) \\ &= k \times support(P_k) \end{aligned} \qquad (2)$$

Definition 6. A pattern P_k is said to be *length-frequent,* when the length-support is greater than or equal to a given minimum length-support (*minlsup*) threshold,

$$lsupport(P_k) \quad \geq \quad minlsup \qquad (3)$$

For example, given a base graph and traversal database of Fig. 1 and 2 with $|D|$ is 6, and *minlsup* of 0.9, then the pattern <B, C, E> is length-frequent since $3 \times 2/6 = 1.0 \geq 0.9$, but the pattern <B, C> is not since $2 \times 2/6 = 0.7 < 0.9$. From equation (1), (2) and (3), a pattern *P* is length-frequent when its support count satisfies:

$$scount(P_k) \quad \geq \quad \frac{minlsup \times |D|}{k} \qquad (4)$$

We can consider the right hand side of (4) as the lower bound of the support count for a pattern *P* to be length-frequent. Such lower bound, called *support bound*, is given by

$$sbound(P_k) = \left\lceil \frac{minlsup \times |D|}{k} \right\rceil \qquad (5)$$

We take the ceiling of the value since the function $sbound(P_k)$ is an integer. From Equation (4) and (5), we can say a pattern *P* is length-frequent when the support count is greater than or equal to the support bound.

$$scount(P_k) \quad \geq \quad sbound(P_k) \qquad (6)$$

Note that $sbound(P_k)$ can be calculated from base graph without referring traversal database. On the contrary, $scount(P_k)$ can be obtained by referring traversal database.

The problem concerned in this paper is stated as follows. Given a directed graph (base graph) and a set of path traversals on the graph (traversal database), find all length-frequent patterns.

3. Length-frequent Pattern Mining

Traditional mining algorithms have been based on Apriori algorithm [5]. The reason why Apriori algorithm works is due to the downward closure property, which says all the subsets of a large itemset must be also large. For the length setting, however, it is not necessarily true for all the subpatterns of a length-frequent pattern being length-frequent. As in the previous example, although a pattern <B, C> is a subpattern of the length-frequent pattern <B, C, E>, it is not length-frequent. Therefore, Apriori algorithm cannot be directly adopted for the mining of length-frequent patterns. Instead, we will propose new approaches by extending the notion of support bound [6].

3.1 Pruning by Support Bound

Pruning is most critical phase in mining process by reducing the number of candidates as many as possible. Such candidates that have no possibility to become length-frequent in the future can be pruned. On the contrary, we must keep such candidates that have a possibility to become length-frequent in the future. Main concern is how to decide such possibility.

Definition 7. A pattern P_k is said to be *feasible* when it has a possibility to become length-frequent in the future if

extended to longer patterns. In other words, when some future patterns containing P_k will be possibly length-frequent.
Now, the pruning problem is converted to the feasibility problem. For the decision of such feasibility, we will devise the support bound of longer patterns containing P_k. Let the maximum possible length of length-frequent patterns be u, which may be the length of longest traversal in the traversal database. Given a k-pattern P_k, suppose l-pattern containing P_k, denoted by (P_k, l), where $k < l \le u$.

We can derive the lower bound of the support count for l-pattern to be length-frequent. Such lower bound, called *l-support bound* of P_k, is given by

$$sbound(P_k,l) = \left\lceil \frac{minlsup \times |D|}{l} \right\rceil \quad (7)$$

Lemma 1. A pattern P_k is feasible if $scount(P_k) \ge sbound(P_k, l)$ for some $k < l \le u$, but not feasible if $scount(P_k) < sbound(P_k, l)$ for all $k < l \le u$.

Proof. If $scount(P_k) \ge sbound(P_k, l)$, then although $scount(P_k) \ge scount(P_k, l)$ by Property 1, there is still possibility to become $scount(P_k, l) \ge sbound(P_k, l)$. It means that (P_k, l) will possibly be length-frequent. On the contrary, if $scount(P_k) < sbound(P_k, l)$, then because $scount(P_k) \ge scount(P_k, l)$ by Property 1, there is no possibility to become $scount(P_k, l) < sbound(P_k, l)$. It means that (P_k, l) will definitely not be length-frequent.
If a pattern P_k is feasible then some l-patterns containing P_k will be possibly length-frequent. In other word, P_k has a possibility to be subpatterns of some length-frequent l-patterns. Therefore, P_k must be kept to be extended to longer patterns for possible length-frequent patterns in the coming passes. On the contrary, if a pattern P_k is not feasible, then all l-patterns containing P_k will not be length-frequent. In other word, P_k certainly has no possibility to be subpattern of any length-frequent l-patterns. Therefore, P_k must be pruned.

For example, referring to Fig. 1 and Fig. 2, given a 2-pattern <B, C>, suppose 3-pattern <B, C, *v*>. For the additional vertex '*v*', the 3-support bound of <B, C> is

$$sbound(<B,C>,3) = \left\lceil \frac{0.9 \times 6}{3} \right\rceil = 2$$

It means if the support count of <B, C> is greater than or equal to 2, some 3-patterns will be possibly length-frequent. In other word, <B, C> has a possibility to be subpatterns of some length-frequent 3-patterns. Because the support count of the pattern <B, C> is actually 2, the pattern must be extended to 3-patterns for possible length-frequent patterns.

According to Lemma 1, we can devise a pruning algorithm, called 'pruning by support bounds', as follows.

Algorithm. *Pruning by support bounds*

```
for each pattern $P_k$ in candidates set $C_k$ {
    for ($l = k+1; l \le u; l$++) {
        estimate $sbound(P_k, l)$;
        if ($scount(P_k) \ge sbound(P_k, l)$)
            break;   // $P_k$ is feasible. Keep it
    }
    if ($l > u$)
        $C_k = C_k - \{P_k\}$;   // $P_k$ is not feasible. Prune it
}
```

Fig.3 Algorithm for pruning by support bounds

3.2 Mining Algorithm

By combing the pruning algorithm as a whole, we can devise an algorithm for mining length-frequent patterns. Fig. 3 shows the algorithm proposed in this paper, which performs in a level-wise manner.

Algorithm. *Mining length-frequent patterns*

Inputs: Base graph G, Traversal database D, Minimum length-support *minlsup*

Output: List of length-frequent patterns L_k

```
{
  // 1. maximum length of length-frequent patterns
  $u$ = max(length($t$)), $t \in D$;

  // 2. initialize candidate patterns of length 1
  $C_1 = V(G)$;
  for ($k$ = 1; $k \le u$ and $C_k \ne \emptyset$; $k$++) {

    // 3. obtain support counts of candidate patterns
    for each pattern $P_k \in C_k$ {
      for each traversal $t \in D$
        if $P_k$ is contained in $t$, then $scount(P_k)$++;
    }

    // 4. determine length-frequent patterns
    $L_k = \{ P_k \mid P_k \in C_k, lsupport(P_k) \ge minlsup \}$;
        (equivalently, $scount(P_k) \ge sbound(P_k)$)

    // 5. prune candidate patterns
    $C'_k$ = Pruning($C_k$, $G$);

    // 6. generate new candidate patterns for next pass
    for each $P_k = <v_1, v_2, \ldots, v_k>$ in $C'_k$ {
      for each edge $<v_k, v>$ in $G$
        $P_k$ is extended to $P_{k+1} = <v_1, v_2, \ldots, v_k, v>$;
    }
  }
}
```

Fig.4 Algorithm for mining length-frequent patterns

In the algorithm, Step 1 is to find out the maximum possible length of length-frequent patterns, which is limited by the maximum length of traversals. Step 2 initializes candidate patterns of length 1 with the vertices of base graph. In Step 3, traversal database is scanned to obtain the support counts of candidate patterns. Step 4 is to determine support-frequent patterns if the length-support is greater than or equal to the specified minimum value. Equivalently, if the support count is greater or equal to the support bound. In Step 5, the subroutine *Pruning*(C_k, *G*) is to prune candidate patterns as described in Fig. 3. Step 6 generates new candidate patterns of length $k+1$ from the pruned candidate patterns of length k for next pass.

4. Estimations of Support Bound

Given a k-pattern P_k, l-pattern containing P_k is denoted by (P_k, l), where $k < l \le u$. The $sbound(P_k, l)$, defined by Equation (7), should be estimated as described in the pruning algorithm of Fig. 3. We propose two approaches for the estimation of such support bound.

4.1 Estimation by All Vertices

In this approach, we assume any $(l-k)$ vertices in the base graph can be chosen to extend P_k to (P_k, l). Because $sbound(P_k, l)$ decreases monotonically as l increases, we only need $sbound(P_k, u)$ to decide the feasibility of P_k, where u is the length of longest traversal in the traversal database.

For example, refer to Fig. 1 and Fig. 2, the 4-support bound for the pattern <A> is

$$sbound(<A>,4) = \left\lceil \frac{0.9 \times 6}{4} \right\rceil = 2$$

Example.
From the Fig. 1 and 2, we will show how the length-frequent patterns can be mined from the traversal database, where |*D*| is 6. Suppose the minimum length-support threshold (*minlsup*) is 0.9.

1. In the *upperLimit()* subroutine, the algorithm will scan the length of traversals, and returns the maximum length, which is 4 in this example. The maximum length is the upper limit of the length of length-frequent patterns.

2. During the initialization step, the candidate patterns of length 1 are generated with all vertices of the base graph.

C_1 = {<A>, <B>, <C>, <D>, <E>, <F>}

3. The algorithm repeats as follows.

pattern P_1	$scount(P_1)$	$sbound(P_1)$	length-frequent	$sbound(P_1,4)$	feasible
<A>	4	6		2	✓
<B>	3				✓
<C>	4				✓
<D>	1				
<E>	3				✓
<F>	1				

pattern P_2	$scount(P_2)$	$sbound(P_2)$	length-frequent	$sbound(P_2,4)$	feasible
<A, B>	1	3		2	
<A, C>	2				✓
<B, C>	2				✓
<B, D>	0				
<C, E>	3		✓		✓
<D, F>	0				
<E, D>	1				
<E, F>	1				

pattern P_3	$scount(P_3)$	$sbound(P_3)$	length-frequent	$sbound(P_3,4)$	feasible
<A, C, E>	1	2		2	
<B, C, E>	2		✓		✓
<C, E, D>	1				
<C, E, F>	1				

pattern P_4	$scount(P_4)$	$sbound(P_4)$	length-frequent
<B, C, E, D>	0	2	
<B, C, E, F>	1		

The length-frequent patterns are {<C, E>, <B, C, E>}.

4.2 Estimation by Reachable Vertices

To prune unnecessary candidates as many as possible, the support bounds need to be estimated as high as possible. The previous approach, however, has a tendency to under-estimate the support bounds. This tendency is mainly due to the non-consideration of the topology of base graph. Specifically, any vertices are chosen even though they can not be reached from the corresponding pattern. To cope with this problem, we will propose another approach which takes into account the graph topology, specifically reachable vertices.

Definition 8. Given a base graph *G*, *r-reachable vertices* from a vertex *v* is all the vertices reachable from *v* within the distance *r*.

Such *r*-reachable vertices can be regarded as the vertices within the radius *r* from *v*. Therefore, *r*-reachable vertices include all the $(r-1)$-reachable vertices.
Given a k-pattern P_k, let $R(P_k, l)$, $k < l \le u$, be the $(l-k)$-reachable vertices from the head vertex of P_k. They can be

obtained by a level wise manner. For example, from Fig. 1, *R(<A>, 2)* is {B, C}, and *R(<A>, 3)* is {B, C, D, E}.

Algorithm. *Reachable vertices: R(P$_k$, l)*

```
$S$ = {head vertex of $P_k$} for $l = k+1$,  $N_{l-1}$ for $l > k+1$;
$N_l = \varnothing$;   // new reachable vertices
for each vertex $v$ in $S$
    for each edge $<v, w>$ in $G$
        if $w$ is not in $P_k$ and $R(P_k, l-1)$ and $N_l$,
            then append $w$ to $N_l$;
$R(P_k, l) = R(P_k, l-1) \cup N_l$
```

Fig.5 Algorithm for reachable vertices

If *R(P$_k$, l)* is not empty, *(l-k)* vertices among the vertices in *R(P$_k$, l)* can be chosen to extend *P$_k$* to *(P$_k$, l)*. Therefore, we can estimate *sbound(P$_k$, l)*. For example, refer to Fig. 1 and Fig. 2, the *R(<A>, 2)* is {B, C}, therefore the 2-support bound for the pattern <A> is

$$sbound(<A>,2) = \left\lceil \frac{0.9 \times 6}{2} \right\rceil = 3$$

If *R(P$_k$, l)* is empty, *P$_k$* cannot be extended to *(P$_k$, l)*. Therefore, we cannot estimate *sbound(P$_k$, l)*. For example, the *R(<F>, 2)* is empty, therefore the 2-support bound for the pattern <F> is not applicable.

$$sbound(<F>,2) \text{ : not applicable}$$

Example.

pattern P_1	$scount(P_1)$	$sbound(P_1)$	length-frequent	$sbound(P_1,l)$ $l=2$	$l=3$	$l=4$	feasible
<A>	4	6		3	-	-	✓
<B>	3			3	-	-	✓
<C>	4			3	-	-	✓
<D>	1			3	×	×	
<E>	3			3	-	-	✓
<F>	1			×	×	×	

In the above table, '-' denotes 'no need' and '×' denotes 'not applicable'.

pattern P_2	$scount(P_2)$	$sbound(P_2)$	length-frequent	$sbound(P_2,l)$ $l=3$	$l=4$	feasible
<A, B>	1	3		2	2	
<A, C>	2			2	-	✓
<B, C>	2			2	-	✓
<B, D>	0			-	-	
<C, E>	3		✓	2	-	✓
<D, F>	0			-	-	
<E, D>	1			2	×	
<E, F>	1			×	×	

pattern P_3	$scount(P_3)$	$sbound(P_3)$	length-frequent	$sbound(P_3,l)$ $l=4$	feasible
<A, C, E>	1	2		2	
<B, C, E>	2		✓	2	✓
<C, E, D>	1			2	
<C, E, F>	1			×	

pattern P_4	$scount(P_4)$	$sbound(P_4)$	length-frequent
<B, C, E, D>	0	2	
<B, C, E, F>	1		

The length-frequent patterns are {<C, E>, <B, C, E>}.

5. Experimental Results

This section presents experimental results of the mining algorithm by comparing two support bound estimation approaches, *All vertices* and *Reachable vertices*, using synthetic dataset. During the experiment, base graph is generated synthetically according to the parameters, i.e., number of vertices and average number of edges per vertex. All the experiments use a base graph with 100 vertices and 300 edges, i.e., 3 average edges per vertex. The number of traversals is 10,000 and the minimum length-support is 0.9. We generated six sets of traversals, in each of which the maximum length of traversals varies from 5 to 10.

Fig. 6 shows the trend of the number of feasible patterns with respect to the max length of traversals. We measured the number of feasible patterns when the length of candidate patterns is (*max length of traversals – 1*). As shown in the figure, the number of feasible patterns for *Reachable vertices* is smaller than that of *All vertices*. The difference of the number of feasible patterns between two estimation approaches becomes smaller as the max length of traversals increases.

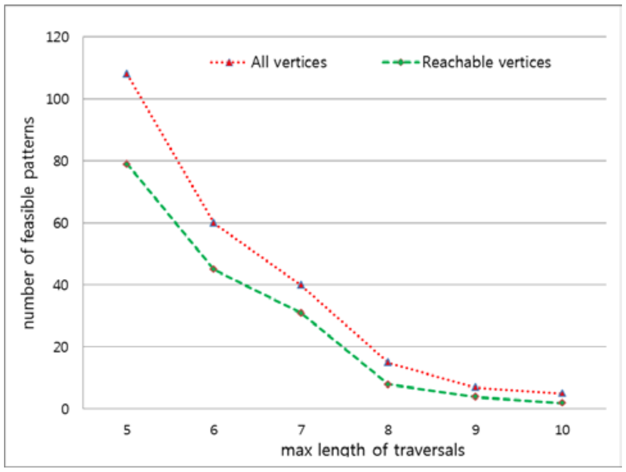

Fig.6 Number of feasible patterns w.r.t diferrent max length of traversals

6. Conclusions

This paper proposed new problem of graph traversal mining by considering length of patterns as well as frequency. With this length setting, mining algorithm should take into account the length of patterns in the measurement of support. To cope with the problem, we formalized new paradigm that is based on the notion of support bound. For the estimation of support bound, two approaches were devised, and then experimented.

References

[1] M.S. Chen, J.S. Park and P.S. Yu, “Efficient Data Mining for Path Traversal Patterns”, IEEE Trans. on Knowledge and Data Engineering, vol. 10, no. 2, pp. 209-221, Mar. 1998.
[2] A. Nanopoulos and Y. Manolopoulos, “Finding Generalized Path Patterns for Web Log Data Mining”, Proc. of East-European Conf. on Advanced Databases and Information Systems (ADBIS), Sep. 2000.
[3] A. Nanopoulos and Y. Manolopoulos, “Mining Patterns from Graph Traversals”, Data and Knowledge Engineering, vol. 37, no. 3, pp. 243-266, Jun. 2001.
[4] S.D. Lee and H.C. Park, “Mining Weighted Frequent Patterns from Path Traversals on Weighted Graph”, International Journal of Computer Science and Network Security, vol. 7, no. 7, pp. 140-148, Apr. 2007.
[5] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules”, Proc. of the 20th VLDB Conference, 1994.
[6] C.H. Cai, W.C. Ada, W.C. Fu, C.H. Cheng and W.W. Kwong, “Mining Association Rules with Weighted Items”, Proc. of International Database Engineering and Applications Symposium (IDEAS), UK, Jul. 1998.