Mining Weighted-Frequent Traversal Patterns using Graph Topology

Hyu Chan Park

Department of Computer Engineering, Korea Maritime and Ocean University, Korea

Summary
Mining problem is to discover valuable patterns from large data set, such as item sets and graph traversals. This paper extends such mining problems to the case where vertices of graph are attached with weights to reflect their importance. Under such weight settings, traditional mining algorithms can not be adopted directly any more. To cope with the problem, this paper proposes new algorithms to discover weighted-frequent patterns from the traversals. Specifically, we devise support bound paradigms for candidate generation and pruning during the mining process.

Key words:
Data mining, Graph, Traversal, Weighted-frequent pattern.

1. Introduction

Data mining is to find significant patterns from huge amount of data sets, i.e., large databases. The significance of patterns may be defined in several ways. The simplest and traditional one is the frequency of patterns, which has been adopted for the mining of un-weighted settings. More valuable one may be weighted-frequency of patterns, which can be adopted for the mining of weighted settings. The mining problem this paper focus is the later and defined as follows. Given a set of traversals on a graph with weighted vertices, discover all patterns contained in the traversals weighted-frequently. 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 issues, we will devise new paradigms to prune unnecessary candidates as many as possible.

The overall structure of our mining algorithm is similar to the Apriori algorithm [1], but the detailed steps contain some differences. To begin with, the algorithm runs in a level-wise manner with increasing sizes. However, since the subpattern of a weighted-frequent pattern may not be weighted-frequent, we cannot generate candidate *(k+1)*-patterns simply from the *k*-patterns as in the Apriori algorithm. Instead, we should find a way to keep the *k*-patterns which may possibly become some weighted-frequent *l*-patterns, for $l > k$, in the coming passes. In order to find such *k*-patterns, we will propose support bound paradigms. During the operation, support count and *l*-support bounds are estimated for each candidate *k*-pattern. If the support count is less than all the *l*-support bounds, we can say the *k*-pattern cannot be any weighted-frequent patterns in the coming passes, therefore it can be pruned. Otherwise, the *k*-pattern has a possibility to be weighted-frequent patterns in the coming passes, therefore it should be kept. The performance of this mining process is heavily relied on the performance of the support bound estimations. To cope with this problem, three estimation methods will be proposed, two of them take into account the graph topology intensively.

This paper is organized as follows. Section 2 reviews previous works related with traversal pattern mining and weight settings. In Section 3, we formulate weighted-frequent mining problem of graph traversals, and then propose new algorithm. Section 4 includes three methods for the estimation of support bound. In Section 5, we experiment and analyze the methods on synthetic data. Finally, Section 6 contains the conclusion and future works.

2. Related Works

Our work is related to the mining problem of traversal patterns with weight settings. For the traversal pattern mining, Chen et al. [2] proposed mining algorithms with hashing and pruning techniques. However, they did not consider graph structure, on which the traversals occur. Nanopoulos et al. [3,4] defined new criteria for the support and subpath containment, and then proposed algorithms with a trie structure. However the above works did not take into account any weight on the traversals.

For the mining with weight settings, most of previous works are related to the mining of association rules and its sub-problem, the discovery of frequent itemsets. Cai et al. [5] generalized the discovery of frequent itemsets to the case where each item is given an associated weight. They introduced new criteria to handle the weights in the process of finding frequent itemsets, such as the weighted support for the measurement of support, and the support bound for the pruning of candidates. Wang et al. [6] extended the problem by allowing weights to be associated with items in each transaction. Their approach ignores the weights when

Manuscript received October 5, 2014
Manuscript revised October 20, 2014

finding frequent itemsets, but considers during the association rule generation. Tao et al. [7] proposed an improved model of weighted support measurement and the weighted downward closure property. Although the above works take into account the notion of weight, they can not be applied directly to the mining problem of traversal patterns.

3. Mining Weighted-Frequent Traversal Patterns

This section describes definitions to be used, and then formulates the mining problem of weighted-frequent traversal patterns. According to the formulations, mining algorithm will be proposed.

A *base graph* is a weighted directed graph, on which traversals occur. For example, the following base graph has 6 vertices and 8 directed edges, in which each vertex is attached with a weight value.

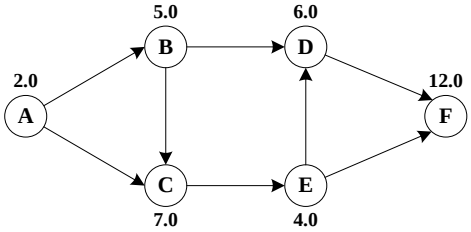

Fig. 1 Base graph

A *traversal* is a sequence of consecutive vertices along a sequence of edges on the base graph. A *traversal database* is a set of traversals. The following traversal database has totally 6 traversals, each of which has an identifier and a sequence of consecutive vertices.

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

Fig. 2 Traversal database

A *subtraversal* is any subsequence of consecutive vertices in a traversal. If a traversal pattern *P* is a subtraversal of a traversal *T*, then we say that *P* is *contained* in *T*, and vice versa *T contains P*. Given a traversal of length *k*, there are only two subtraversals of length *k-1*. For example, given a traversal of length 4, <B, C, E, F>, there are only two subtraversals of length 3, that is <B, C, E> and <C, E, F>. Note that non-consecutive sequences, such as <B, C, F>, are not subtraversals.

The *support count* of a pattern *P*, *scount(P)*, is the number of traversals containing the pattern. The *support* of a pattern *P*, *support(P)*, is the fraction of traversals containing the pattern. Given a traversal database *D*, let $|D|$ be the number of traversals.

$$support(P) = \frac{scount(P)}{|D|} \tag{1}$$

Given a base graph with a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$, in which each vertex v_j is assigned with a weight $w_j \geq 0$, the *weighted support* of a pattern *P*, *wsupport(P)*, is

$$wsupport(P) = \left(\sum_{v_j \in P} w_j \right) \left(support(P) \right) \tag{2}$$

A pattern *P* is said to be *weighted-frequent* when the weighted support is greater than or equal to a given minimum weighted support threshold, *minwsup*.

$$wsupport(P) \geq minwsup \tag{3}$$

For example, given a base graph and traversal database of Fig. 1 and 2, and *minwsup* of 5.0, then the pattern <B, C, E> is weighted-frequent since $(5.0 + 7.0 + 4.0) \times 2/6 = 5.3 \geq 5.0$, but the pattern <B, C> is not since $(5.0 + 7.0) \times 2/6 = 4.0 < 5.0$.

From equation (1), (2) and (3), a pattern *P* is weighted-frequent when its support count satisfies

$$scount(P) \geq \frac{minwsup \times |D|}{\sum_{v_j \in P} w_j} \tag{4}$$

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

$$sbound(P) = \left\lceil \frac{minwsup \times |D|}{\sum_{v_j \in P} w_j} \right\rceil \tag{5}$$

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

$$scount(P) \geq sbound(P) \quad (6)$$

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

To cope with the problem, we propose a mining algorithm based on the Apriori algorithm. The reason why Apriori algorithm works is due to the downward closure property, which says all the subpatterns of a frequent pattern are also frequent. With weight settings, however, it is not necessarily true for all the subpatterns of a weighted-frequent pattern being also weighted-frequent. For example, although <B, C> is a subpattern of weighted-frequent pattern <B, C, E>, it is not weighted-frequent as shown in the previous. Therefore, we can not directly adopt Apriori algorithm. Instead, we will extend Apriori algorithm with the notion of support bound. Fig. 3 shows the extended algorithm, which performs in a level-wise manner.

Algorithm. *Mining weighted-frequent traversal patterns*

Inputs: Base graph G, Traversal database D, Minimum weighted support *minwsup*

Output: List of weighted-frequent patterns L_k

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

    // 2. initialize candidate patterns of length 1
  C1 = V(G);

  for (k = 1; k ≤ u and Ck ≠ ∅; k++) {

      // 3. obtain support counts of candidate patterns
    for each pattern p ∈ Ck {
      for each traversal t ∈ D
        if p is contained in t, then p.scount++;
    }

      // 4. determine weighted-frequent patterns
    Lk = {p | p ∈ Ck, p.wsupport ≥ minwsup};
                (equivalently, p.scount ≥ p.sbound)

      // 5. prune candidate patterns
    C'k = pruneCandidates(Ck, G);

      // 6. generate new candidate patterns for next pass
    for each P = <p1, p2, …, pk> in C'k {
      for each edge <pk, v> in G
        P is extended to <p1, p2, …, pk, v>;
    }
  }
}
```

Fig.3 Algorithm for mining weighted-frequent traversal patterns

In the algorithm, each step is outlined as follows. Step 1 is to find out the maximum possible length of weighted-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 weighted-frequent patterns if the weighted 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 *pruneCandidates*(C_k, G) is to prune candidate patterns, which will be described in the next section. Step 6 generates new candidate patterns of length $k+1$ from the pruned candidate patterns of length k for next pass.

4. Pruning by Support Bound

The cornerstone to improve the mining performance is to devise a pruning method which can reduce the number of candidates as many as possible. We must prune such candidates that have no possibility to become weighted frequent in the future. On the contrary, we must keep such candidates that have a possibility to become weighted-frequent in the future. Main concern is how to decide such possibility.

Definition 1. A pattern P is said to be *feasible* if it has a possibility to become weighted-frequent in the future when extended to longer patterns. In other words, if some future patterns containing P will be possibly weighted-frequent.

Now, the pruning problem is converted to the feasibility problem. For the decision of such feasibility, we will first devise the weight bound of a pattern. Let the maximum possible length of weighted-frequent patterns be u, which may be the length of longest traversal in the traversal database. Given a k-pattern P, suppose l-pattern containing P, denoted by (P, l), where $k < l \leq u$. For the additional $(l - k)$ vertices, if we can estimate upper bounds of the weights as $w_{r_1}, w_{r_2}, \ldots, w_{r_{l-k}}$, then the upper bound of the weight of the l-pattern (P, l) is given by

$$wbound(P,l) = \sum_{v_j \in P} w_j + \sum_{j=1}^{l-k} w_{r_j} \quad (7)$$

We call this upper bound as *l-weight bound* of *P*. The first sum is the sum of the weights for the *k*-pattern *P*. The second one is the sum of the *(l − k)* estimated weights, which can be estimated in several ways. We will propose three estimation methods in the following subsections.

From (5) and (7), we can derive the lower bound of the support count for *l*-pattern containing *P* to be weighted-frequent. Such lower bound, called *l-support bound* of *P*, is given by

$$sbound(P,l) = \left\lceil \frac{minwsup \times |D|}{wbound(P,l)} \right\rceil \quad (8)$$

With these formulations, we can say a pattern *P* is feasible if *scount(P)* ≥ *sbound(P, l)* for some $k < l \le u$, but not feasible if *scount(P)* < *sbound(P, l)* for all $k < l \le u$. If a pattern *P* is feasible then some *l*-patterns containing *P* will be possibly weighted-frequent. In other word, *P* has a possibility to be subpatterns of some weighted-frequent *l*-patterns. Therefore, *P* must be kept to be extended to longer patterns for possible weighted-frequent patterns in the coming passes. On the contrary, if a pattern *P* is not feasible, then all *l*-patterns containing *P* will not be weighted-frequent. In other word, *P* certainly has no possibility to be subpattern of any weighted-frequent *l*-patterns. Therefore, *P* must be pruned.

For example, referring to Fig. 1 and Fig. 2, given a 2-pattern <B, C>, suppose 3-pattern <B, C, –>. For the additional vertex '–', we can estimate a possible upper bound of the weight as 12.0, which is the greatest weight among the remaining vertices besides B and C. Therefore, the 3-support bound of <B, C> is

$$sbound(< B,C >,3) = \left\lceil \frac{5.0 \times 6}{(5.0+7.0)+(12.0)} \right\rceil = 2$$

It means if the support count of <B, C> is greater than or equal to 2, some 3-patterns will be possibly weighted-frequent. In other word, <B, C> has a possibility to be subpatterns of some weighted-frequent 3-patterns. Because the support count of the pattern <B, C> is actually 2, the pattern must be kept and extended to 3-patterns for possible weighted-frequent patterns. According to the formulations, we can devise a pruning algorithm as follows.

Algorithm. *Pruning by support bounds*

```
for each pattern P in candidates set C_k {
    for each l from k+1 to u {
        estimate sbound(P, l);
        if (scount(P) ≥ sbound(P, l))
            break;    // P is feasible. Keep it
    }
    if (l > u)
        C_k = C_k – {P};    // P is not feasible. Prune it
}
```

Fig.4 Algorithm for pruning by support bounds

In the algorithm, the key problem is how to estimate the support bound. We propose three methods in the followings.

4.1 Support Bound by All Vertices

Given a *k*-pattern *P*, suppose *l*-pattern containing *P*, where $k < l \le u$. Let *V* be the set of all vertices in the base graph. Among the remaining vertices *(V − P)*, let the vertices with the *(l − k)* greatest weights be $v_{r_1}, v_{r_2}, \ldots, v_{r_{l-k}}$. Then, the *l*-weight bound, *wbound(P, l)*, and the *l*-support bound, *sbound(P, l)*, of *P* are defined same as Equation (7) and (8), respectively.

For example, refer to Fig. 1 and Fig. 2, the 3-support bound for the pattern <A> is

$$sbound(< A >,3) = \left\lceil \frac{5.0 \times 6}{(2.0)+(12.0+7.0)} \right\rceil = 2$$

Example.

From Fig. 1 and 2, we will show how the weighted-frequent patterns are generated from the traversal database. Suppose the minimum weighted support threshold, *minwsup*, is 5.0.

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 weighted-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*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*	feasible

				$l = 2$	$l = 3$	$l = 4$	
<A>	4	15(2)		3(14)	-		✓
<B>	3	6(5)		2(17)	-	-	✓
<C>	4	5(7)		2(19)	-	-	✓
<D>	1	5(6)		2(18)	2(25)	1(30)	✓
<E>	3	8(4)		2(16)	-	-	✓
<F>	1	3(12)		2(19)	2(25)	1(30)	✓

In the above table, '-' denotes 'no need'.

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*		feasible
				$l = 3$	$l = 4$	
<A, B>	1	5(7)		2(19)	2(26)	
<A, C>	2	4(9)		2(21)	-	✓
<B, C>	2	3(12)		2(24)	-	✓
<B, D>	0	-		-	-	
<C, E>	3	3(11)	✓	-	-	✓
<D, F>	0	-		-	-	
<E, D>	1	3(10)		2(22)	2(29)	
<E, F>	1	2(16)		2(23)	2(29)	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*	feasible
				$l = 4$	
<A, C, E>	1	3(13)		2(25)	
<B, C, E>	2	2(16)	✓	-	✓
<C, E, D>	1	2(17)		2(29)	
<C, E, F>	1	2(23)		2(29)	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent
<B, C, E, D>	0	-	
<B, C, E, F>	1	2(28)	

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

4.2 Support Bound by Reachable Vertices

To prune unnecessary candidates as many as possible, the support bounds need to be estimated as high as possible. It means that we must estimate the weight bounds as low as possible. The previous method, however, has a tendency to over-estimate the weight bounds. This tendency is mainly due to the non-consideration of the topology of base graph. Specifically, the vertices with greatest weights are chosen one after one, even though they cannot be reached from the corresponding pattern. To cope with this limitation, we will propose another method which takes into account the graph topology, specifically reachable vertices.

Definition 2. 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.

With this definition, we can find reachable vertices of a pattern as follows: Given a *k*-pattern *P*, let *R(P, l)*, $k < l \leq u$, be the *(l-k)*-reachable vertices from the head vertex of *P*. 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, l)*

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

Fig.5 Algorithm for reachable vertices

Among the vertices in *R(P, l)*, let the vertices with the *(l − k)* greatest weights be $v_{r_1}, v_{r_2}, \ldots, v_{r_{l-k}}$. Then, the *l*-weight bound, *wbound(P, l)*, and the *l*-support bound, *sbound(P, l)*, of *P* are obtained by Equation (7) and (8), respectively.

For example, refer to Fig. 1 and Fig. 2, the 3-support bound for the pattern <A> is

$$sbound(<A>,3) = \left\lceil \frac{5.0 \times 6}{(2.0) + (7.0 + 6.0)} \right\rceil = 2$$

Example.

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*			feasible
				$l = 2$	$l = 3$	$l = 4$	
<A>	4	15(2)		4(9)	-	-	✓
<B>	3	6(5)		3(12)	-	-	✓
<C>	4	5(7)		3(11)	-	-	✓
<D>	1	5(6)		2(18)	×	×	
<E>	3	8(4)		2(16)	-	-	✓
<F>	1	3(12)		×	×	×	

In the above table, '×' denotes 'not applicable'.

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*		feasible
				$l = 3$	$l = 4$	
<A, B>	1	5(7)		3(14)	2(26)	
<A, C>	2	4(9)		3(13)	2(27)	✓
<B, C>	2	3(12)		2(16)	-	✓
<B, D>	0	-		-	-	
<C, E>	3	3(11)	✓	-	-	✓
<E, D>	1	3(10)		2(22)	×	
<E, F>	1	2(16)		×	×	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*	feasible
				l = 4	
<A, C, E>	1	3(13)		2(25)	
<B, C, E>	2	2(16)	✓	-	✓
<C, E, D>	1	2(17)		2(29)	
<C, E, F>	1	2(23)		×	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent
<B, C, E, D>	0	-	
<B, C, E, F>	1	2(28)	

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

4.3 Support Bound by Reachable Paths

The previous method, however, still has a tendency to over-estimate the weight bounds. This tendency is mainly due to the non-consideration of reachable paths in the base graph. Specifically, the vertices with greatest weights are chosen one after one, even though they are not on the same path. To cope with this limitation, we will propose yet another method which takes more into account the graph topology, specifically reachable paths.

Definition 3. Given a base graph *G*, *r-reachable paths* from a vertex *v* are all the paths of length *r* from *v* to other vertices. The *weight* of a *r*-reachable path from *v* is the sum of weights of the vertices in the path excluding *v* itself.

With this definition, we can find reachable paths of a pattern as follows: Given a *k*-pattern *P*, let *F(P, l)*, $k < l \leq u$, be the *(l-k)*-reachable paths from the head vertex of *P*. They can be obtained by a level wise manner. For example, from Fig. 1, *F(<A>, 2)* is {<A, B>, <A, C>}, and *F(<A>, 3)* is {<A, B, C>, <A, B, D>, <A, C, E>}.

Algorithm. *Reachable paths: F(P, l)*

S = {<head vertex of *P*>} for *l* = *k+1*,
  F(P, l-1) for *l* > *k+1*;
for each path *t* = < …, *v*> in *S*
  for each edge <*v*, *w*> in *G*
    if *w* is not in *P* and *t*, then append <…, *v*, *w*>
      to *F(P, l);*

Fig.6 Algorithm for reachable paths

Among the paths in *F(P, l)*, let the vertices of the greatest weight path be $v_{r1}, v_{r2}, \ldots, v_{rl-k}$, exclusive of the head vertex of *P*. Then, the *l*-weight bound, *wbound(P, l)*, and the *l*-support bound, *sbound(P, l)*, of *P* are defined same as Equation (7) and (8), respectively.

For example, refer to Fig. 1 and Fig. 2, the 3-support bound for the pattern <A> is

$$sbound(< A >,3) = \left\lceil \frac{5.0 \times 6}{(2.0) + (5.0 + 7.0)} \right\rceil = 3$$

Example.

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*			feasible
				l = 2	*l* = 3	*l* = 4	
<A>	4	15(2)		4(9)	-	-	✓
<B>	3	6(5)		3(12)	-	-	✓
<C>	4	5(7)		3(11)	-	-	✓
<D>	1	5(6)		2(18)	×	×	
<E>	3	8(4)		2(16)	-	-	✓
<F>	1	3(12)		×	×	×	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*		feasible
				l = 3	*l* = 4	
<A, B>	1	5(7)		3(14)	2(25)	
<A, C>	2	4(9)		3(13)	2(25)	✓
<B, C>	2	3(12)		2(16)	-	✓
<B, D>	0	-		-	-	
<C, E>	3	3(11)	✓	-	-	✓
<E, D>	1	3(10)		2(22)	×	
<E, F>	1	2(16)		×	×	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent	*sbound(P,l)* *(wbound(P,l))*	feasible
				l = 4	
<A, C, E>	1	3(13)		2(25)	
<B, C, E>	2	2(16)	✓	-	✓
<C, E, D>	1	2(17)		2(29)	
<C, E, F>	1	2(23)		×	

pattern *P*	*scount(P)*	*sbound(P)* *(wbound(P))*	weighted-frequent
<B, C, E, D>	0	-	
<B, C, E, F>	1	2(28)	

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

5. Experimental Results

This section presents experimental results of the mining algorithm, and compares three support bound estimation methods, *All vertices*, *Reachable vertices*, and *Reachable paths* 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. And then, we assigned distinctive weight to each vertex of the base graph. 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 weighted support is 2.0. We generated six

sets of traversals, in each of which the maximum length of traversals varies from 5 to 10.

Fig. 7 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 their length is (max length of traversals – 1). As shown in the figure, the number of feasible patterns for *Reachable paths* is smaller than those of *All vertices* and *Reachable vertices*. The difference of the number of feasible patterns between three methods becomes smaller as the max length of traversals increases.

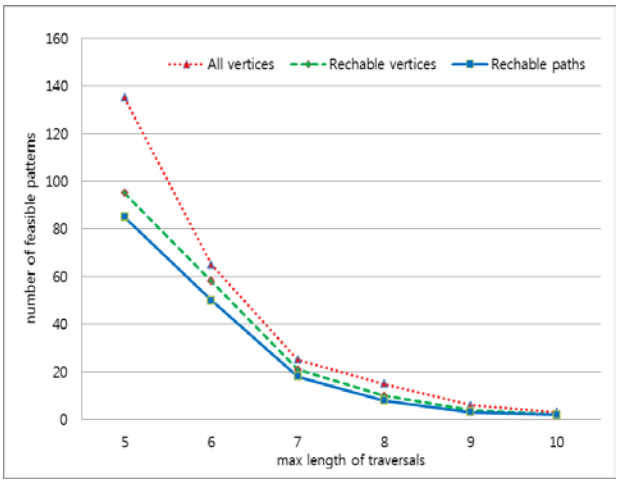

Fig. 7 Number of feasible patterns w.r.t diferrent max length of traversals

6. Conclusions

This paper extended mining problem to the discovering of weighted-frequent traversal patterns on graph. Differently from previous approaches, vertices of graph are attached with weights which reflect their importance. With these weight settings, we presented the mining algorithm which takes into account the weights in the measurement of support. The algorithm is based on the notion of support bound. For the estimation of support bound, three methods were also proposed, and then experimented. The method using graph topology resulted in better performance. Future works may include other weight settings, such as edge weights and traversal weights.

References

[1] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules”, Proc. of International Conference on Very Large Databases (VLDB), Chile, Sep. 1994.

[2] 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.

[3] 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.

[4] A. Nanopoulos and Y. Manolopoulos, “Mining Patterns from Graph Traversals”, Data and Knowledge Engineering, vol. 37, no. 3, pp. 243-266, Jun. 2001.

[5] 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.

[6] W. Wang, J. Yang and P.S. Yu, “Efficient Mining of Weighted Association Rules (WAR)”, Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), USA, Aug. 2000.

[7] F. Tao, F. Murtagh and M. Farid, “Weighted Association Rule Mining using Weighted Support and Significance Framework”, Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), USA, Aug. 2003.