Mining Significant Patterns from Graph Traversals by Considering Frequency and Average Weight

Hyu Chan Park

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

Summary
Graph traversal is a sequence of vertices along edges on a graph, by which a lot of real world problems can be modeled. Mining patterns from such traversals has been found useful in several applications such as Web mining. However, previous works considered only frequency or summed weight of patterns. This paper extends them by considering average weight of patterns. Under such weight settings, traditional mining algorithms can not be adopted directly any more. To cope with the problem, this paper proposes new methodology by considering average weight along with frequency.

Key words:
Data mining, Graph traversal, Average weight

1. Introduction

Graph traversal is widely used to model several classes of real world problems. For example, user navigations on the Web site can be modeled as graph traversals. Once such graph traversals are given, valuable information can be mined. Most common and simplest form of the information may be frequent patterns. Chen et al. [1] proposed 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. They considered the graph, on which traversals occur. For these mining problems, the well-known Apriori algorithm can be used [4]. Reasoning on these approaches is that frequent patterns are valuable.

More valuable patterns can be mined by considering weights in the mining process. For such weighted mining, there have some works [5, 6, 7, 8]. However, they are related to the mining of association rules and itemsets, but not traversals. Lee and Park [9] had combined the mining problem of traversal pattern and weighted mining. This approach considers weights 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 weight of a pattern is calculated by the summation of weights of vertices in the pattern. With these weight settings, the mining algorithm cannot be relied on the well-known Apriori paradigm any more. Instead, the notion of support bound can be adopted [5]. This paper extends previous works by considering average weight of patterns, but not summed weight. Although overall foundation is similar to the previous, but details may be different. To cope with this difference, we will propose new definitions and approaches.

This paper is organized as follows. Section 2 formalizes traversal pattern mining problem by considering frequency and weight. In Section 3, we propose an algorithm for the discovery of average-weighted frequent patterns from traversals on weighted graph. Section 4 includes two methods for the estimation of weight and support bound used in this mining process. In Section 5, we experiment and analyze the algorithm on synthetic data. Finally, Section 6 contains the conclusion and future works.

2. Average-weighted Frequent Patterns

Definition 1. A weighted directed graph is a finite set of vertices and edges, in which each edge joins one ordered pair of vertices, and each vertex is associated with a weight value. A base graph is a weighted directed graph, on which traversals occur.

For example, the following base graph has 6 vertices and 8 edges, in which each vertex is associated with a weight.

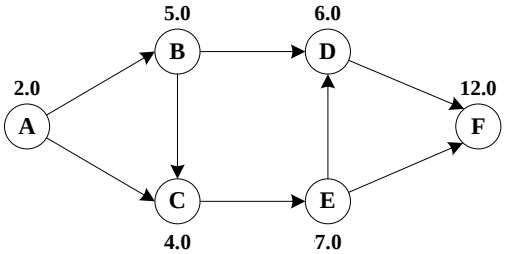

Fig. 1 Example of base graph

Definition 2. A traversal is a sequence of consecutive vertices along a sequence of edges on a base graph. We

Manuscript received November 5, 2015
Manuscript revised November 20, 2015

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. The weight of a traversal is the sum of vertex weights 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. A 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.

There is a well known property on such subtraversal [2, 3] as follows.

Property 1. 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, <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.

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

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

Property 2. 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) ≥ scount(P_k, l) and support(P_k) ≥ support(P_k, l).

Given a base graph with a set of vertices V = {v_1, v_2, …, v_n}, in which each vertex v_j is assigned with a weight w_j ≥ 0, we will define the average-weighted support of a pattern.

Definition 5. The *average-weighted support* of a pattern P_k, denoted by *wsupport(P_k),* is

$$wsupport(P_k) = \frac{\sum_{vj \in Pk} w_j}{k} support(P_k) \quad (2)$$

Definition 6. A pattern P_k is said to be *average-weighted frequent*, i.e. *significant*, when the average-weighted support is greater than or equal to a given minimum weighted support (*minwsup*) threshold,

$$wsupport(P_k) \geq minwsup \quad (3)$$

For example, given a base graph and traversal database of Fig. 1 and 2 with |*D*| is 6, and *minwsup* of 1.5, then the pattern <B, C, E> is average-weighted frequent since (5.0 + 4.0 + 7.0)/3 × 2/6 = 1.8 ≥ 1.5, but the pattern <B, C> is not since (5.0 + 4.0)/3 × 2/6 = 1.0 < 1.5.

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

$$scount(P_k) \geq \frac{k}{\sum_{vj \in Pk} w_j} minwsup \times |D| \quad (4)$$

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

$$sbound(P_k) = \left\lceil \frac{k}{\sum_{vj \in Pk} w_j} minwsup \times |D| \right\rceil \quad (5)$$

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 average-weighted frequent when the support count is greater than or equal to the support bound.

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

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 weighted directed graph (base graph) and a set of path traversals on the graph (traversal database), find all average-weighted frequent patterns.

3. Mining Average-weighted Frequent Patterns

We propose a methodology for the mining of average-weighted frequent patterns. An efficient algorithm for mining large itemsets has been Apriori algorithm. 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 weighted setting, however, it is not necessarily true for all the subpatterns of a average-weighted frequent pattern being average-weighted frequent. For example, although a pattern <B, C> is a subpattern of the average-weighted frequent pattern <B, C, E>, it is not average-weighted frequent. Therefore, we can not directly adopt Apriori algorithm. Instead, we will extend the notion of support bound, which can be applied to the pruning and candidate generation.

3.1 Pruning by Support Bound

One of the cornerstones 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 average-weighted frequent in the future. On the contrary, we must keep such candidates that have a possibility to become average-weighted 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 average-weighted frequent in the future if extended to longer patterns. In other words, when some future patterns containing P_k will be possibly average-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 average-weighted 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 $k < l \le u$. For the additional $(l - k)$ vertices, if we can estimate upper bounds of the weights as $w_{r1}, w_{r2}, \ldots, w_{rl-k}$, then the upper bound of the weight of the l-pattern is given by

$$wbound(P_k, l) = \frac{\sum_{v_j \in Pk} w_j + \sum_{j=1}^{l-k} w_{rj}}{l} \quad (7)$$

The first sum is the sum of the weights for the k-pattern P_k. The second one is the sum of the $(l - k)$ estimated weights, which can be estimated in several ways. We will propose two estimation methods in the following section.

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

$$sbound(P_k, l) = \left\lceil \frac{l}{\sum_{v_j \in Pk} w_j + \sum_{j=1}^{l-k} w_{rj}} minwsup \times |D| \right\rceil \quad (8)$$

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. Let l_i be anyone out of l. If $scount(P_k) \ge sbound(P_k, l_i)$, then because $scount(P_k) \ge scount(P_k, l_i)$ by Property 2, there is a possibility to be $scount(P_k, l_i) \ge sbound(P_k, l_i)$. It means that (P_k, l_i) will possibly be average-weighted frequent. On the contrary, if $scount(P_k) < sbound(P_k, l_i)$, then because $scount(P_k) \ge scount(P_k, l_i)$ by Property 2, $scount(P_k, l_i) < sbound(P_k, l_i)$. It means that (P_k, l_i) will definitely not be average-weighted frequent.

If a pattern P_k is feasible then some l-patterns containing P_k will be possibly average-weighted frequent. In other word, P_k has a possibility to be subpatterns of some average-weighted frequent l-patterns. Therefore, P_k must be kept to be extended to longer patterns for possible average-weighted 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 average-weighted frequent. In other word, P_k certainly has no possibility to be subpattern of any average-weighted 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, –>. 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

$$sbound(<B,C>,3) = \left\lceil \frac{3}{(5.0 + 4.0) + (12.0)} 1.5 \times 6 \right\rceil = 2$$

It means if the support count of <B, C> is greater than or equal to 2, some 3-patterns will be possibly average-weighted frequent. In other word, <B, C> has a possibility to be subpatterns of some average-weighted 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 average-weighted 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 in candidates set Ck {
    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)
        Ck = Ck – {P};    // P 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 average-weighted frequent patterns. Fig. 3 shows the algorithm proposed in this paper, which performs in a level-wise manner.

Algorithm. *Mining average-weighted frequent patterns*

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

Output: List of average-weighted frequent patterns L_k

```
{
  // 1. maximum length of average-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 average-weighted frequent patterns
    Lk = {p | p ∈ Ck, p.averageWsupport ≥ 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.4 Algorithm for mining average-weighted frequent 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(Ck, 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. Estimations of Support Bound

We propose two methods for the estimation of weight and support bound.

4.1 Estimation by All Vertices

Given a *k*-pattern P_k, suppose *l*-pattern containing P_k, where $k < l \leq u$. Let *V* be the set of all vertices in the base graph. Among the remaining vertices $(V - P_k)$, let the vertices with the $(l - k)$ greatest weights be $v_{r1}, v_{r2}, \ldots, v_{rl-k}$. Then, the *l*-weight bound, $wbound(P_k, l)$, and the *l*-support bound, $sbound(P_k, l)$, 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

$$sbound(<A>,3) = \left\lceil \frac{3}{(2.0)+(12.0+7.0)} 1.5 \times 6 \right\rceil = 2$$

Example.
From the Fig. 1 and 2, we will show how the average-weighted frequent patterns are generated from the traversal database, where $|D|$ is 6. Suppose the minimum weighted support threshold (*minwsup*) is 1.5.

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 average-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_1	$scount(P_1)$	$sbound(P_1)$ $(wbound(P_1))$	average-weighted frequent	$sbound(P_1,l)$ $(wbound(P_1,l))$			feasible
				$l = 2$	$l = 3$	$l = 4$	
<A>	4	5(2.0)		2(7.0)	-	-	✓
<B>	3	2(5.0)	✓	2(8.5)	-	-	✓
<C>	4	3(4.0)	✓	2(8.0)	-	-	✓
<D>	1	2(6.0)		1(9.0)	-	-	✓
<E>	3	2(7.0)	✓	1(9.5)	-	-	✓
<F>	1	1(12.0)	✓	1(9.5)	-	-	✓

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

pattern P_2	$scount(P_2)$	$sbound(P_2)$ $(wbound(P_2))$	average-weighted frequent	$sbound(P_2,l)$ $(wbound(P_2,l))$		feasible
				$l = 3$	$l = 4$	
<A, B>	1	3(3.5)		2(6.3)	2(6.5)	
<A, C>	2	3(3.0)		2(6.0)	-	✓
<B, C>	2	2(4.5)	✓	2(7.0)	-	✓
<B, D>	0	-		-	-	
<C, E>	3	2(5.5)	✓	2(7.7)	-	✓
<D, F>	0	-		-	-	
<E, D>	1	2(6.5)		2(8.3)	2(7.5)	
<E, F>	1	1(9.5)	✓	2(8.3)	2(7.5)	

pattern P_3	$scount(P_3)$	$sbound(P_3)$ $(wbound(P_3))$	average-weighted frequent	$sbound(P_3,l)$ $(wbound(P_3,l))$	feasible
				$l = 4$	
<A, C, E>	1	3(4.3)		2(6.3)	
<B, C, E>	2	2(5.3)	✓	2(7.0)	✓
<C, E, D>	1	2(5.7)		2(7.3)	
<C, E, F>	1	2(7.7)		2(7.3)	

pattern P_4	$scount(P_4)$	$sbound(P_4)$ $(wbound(P_4))$	average-weighted frequent
<B, C, E, D>	0	-	
<B, C, E, F>	1	2(7.0)	

The average-weighted-frequent patterns are {<B>, <C>, <E>, <F>, <B, C>, <C, E>, <E, F>, <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. 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 can not 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 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, but not in P_k and not through the vertices in 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 $l > k+1$;
$N_l = \varnothing$;
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

Among the vertices in $R(P_k, l)$, let the vertices with the $(l - k)$ greatest weights be $v_{r1}, v_{r2}, \ldots, v_{rl-k}$. Then, the l-weight bound, $wbound(P_k, l)$, and the l-support bound, $sbound(P_k, l)$, of P_k 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

$$sbound(<A>,3) = \left\lceil \frac{3}{(2.0)+(7.0+6.0)} 1.5 \times 6 \right\rceil = 2$$

Example.

pattern P_1	$scount(P_1)$	$sbound(P_1)$ $(wbound(P_1))$	average-weighted frequent	$sbound(P_1,l)$ $(wbound(P_1,l))$			feasible
				$l = 2$	$l = 3$	$l = 4$	
<A>	4	5(2.0)		3(3.5)	-	-	✓
<B>	3	2(5.0)	✓	2(5.5)	-	-	✓
<C>	4	3(4.0)	✓	2(5.5)	-	-	✓
<D>	1	2(6.0)		1(9.0)	-	-	✓
<E>	3	2(7.0)	✓	1(9.5)	-	-	✓
<F>	1	1(12.0)	✓	×	×	×	

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

pattern P_2	$scount(P_2)$	$sbound(P_2)$ $(wbound(P_2))$	average-weighted frequent	$sbound(P_2,l)$ $(wbound(P_2,l))$		feasible
				$l = 3$	$l = 4$	
<A, B>	1	3(3.5)		3(4.3)	2(6.3)	
<A, C>	2	3(3.0)		3(4.3)	2(6.3)	✓
<B, C>	2	2(4.5)	✓	2(5.3)	-	✓
<B, D>	0	-		-	-	
<C, E>	3	2(5.5)	✓	2(7.7)	-	✓
<D, F>	0	-		-	-	
<E, D>	1	2(6.5)		2(8.3)	×	
<E, F>	1	1(9.5)	✓	×	×	

pattern P_3	$scount(P_3)$	$sbound(P_3)$ $(wbound(P_3))$	average-weighted frequent	$sbound(P_3,l)$ $(wbound(P_3,l))$ $l = 4$	feasible
<A, C, E>	1	3(4.3)		2(6.3)	
<B, C, E>	2	2(5.3)	✓	2(7.0)	✓
<C, E, D>	1	2(5.7)		2(7.3)	
<C, E, F>	1	2(7.7)		×	

pattern P_4	$scount(P_4)$	$sbound(P_4)$ $(wbound(P_4))$	average-weighted frequent
<B, C, E, D>	0	-	
<B, C, E, F>	1	2(7.0)	

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

5. Experimental Results

This section presents experimental results of the mining algorithm, and compares two estimation algorithms, *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. 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 1.5. 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 algorithms becomes smaller as the max length of traversals increases.

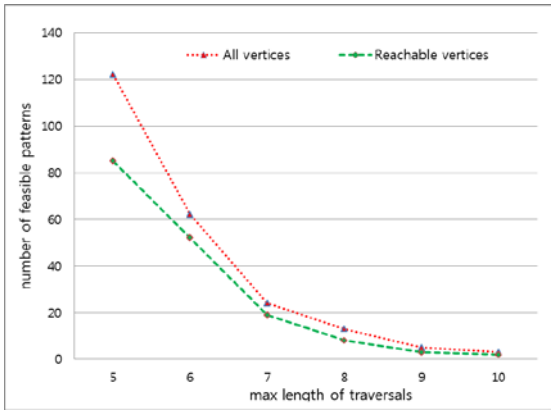

Fig.6 Number of feasible patterns w.r.t diferrent max length of traversals

6. Conclusions

This paper proposed new formalization and algorithms for the mining of traversal patterns by considering weight as well as frequency. In the formalization, vertices of graph are attached with weights which reflect their importance. With this weight setting, we presented new mining algorithm which takes into account average-weights in the measurement of support. This algorithm is based on the notion of support bound. We also proposed two methods for the estimation of support bound, and then experimented on them.

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] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", Proc. of the 20th VLDB Conference, 1994.
[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.
[8] U. Yun and J.J. Leggett, "WLPMiner: Weighted Frequent Pattern Mining with Length-Decreasing Support Constraints", Proc. of Pacific-Asia International Conference on Knowledge Discovery and Data Mining (PAKDD), Vietnam, May 2005.
[9] 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.