Mining Frequent Patterns from Weighted Traversals on Graph using Confidence Interval and Pattern Priority

Seong Dae Lee,[†] *and Hyu Chan Park* [††]

[†, ††] Department of Computer Engineering, Korea Maritime University, Busan, Korea

[††] School of Information Technology & Electrical Engineering, University of Queensland, QLD, Australia

Summary
A lot of real world problems can be modeled as traversals on graph. Mining from such traversals has been found useful in several applications. However, previous works considered only unweighted traversals. This paper generalizes this to the case where traversals are given weights to reflect their importance. A new algorithm is proposed to discover frequent patterns from the weighted traversals. The algorithm adopts the notion of confidence interval to distinguish between confident traversals and outliers. By excluding the outliers, more reliable frequent patterns can be obtained. In addition, they are further ranked according to their priority. The algorithm can be applied to various applications, such as Web mining.

Key words:
Data mining, Frequent pattern, Weighted traversal, Graph, Confidence interval

1. Introduction

Data mining refers to the process of semi-automatically analyzing large databases to discover useful patterns [1]. Several data structures and algorithms have been proposed and successfully applied to many applications [2]. Recently, data mining on graph becomes a center of interest. Graph is widely used to model several classes of data in the real world. The structure of Web site can be modeled as a graph, for example, in which vertices are for Web pages, and edges represent hyperlinks between the pages. User navigations on the Web site can be modeled as traversals on top of the graph. Each traversal can be represented as a sequence of vertices, or equivalently a sequence of edges. Once the graph and its traversals are given, valuable information can be discovered. Most common form of the information may be frequent patterns, i.e., the sub-traversals that contained in a large ratio of traversals. In the previous works, but, traversals are treated uniformly without considering their importance [3, 4, 5].

In this paper, traversals are assigned with weights to reflect their importance. For example, when users navigate Web site, they may have different interest in each page, and therefore stay for different times. Each edge, which represents a transition between Web pages, can be assigned with a weight standing for the user stay time. This paper generalizes the mining problem to the case where traversals are given such weights reflecting their importance. The weights are taken into account in the measurement of support, the ratio of traversals which contains a candidate pattern. If a traversal has some edges with extremely smaller or larger weight, then it is treated as an outlier, and can not contribute to the support. For example, when users navigate Web site, they may pass through a page very quickly to transfer to another page, or take their eyes off a page for a long time while do another work. This kind of page-view is abnormal and treated as an outlier because the page is not effectively read by the user.

We adopt the notion of confidence interval to classify the weights into confident ones and outliers. If a weight lies within the confidence interval, then it is considered as a confident one, but if it lies outside the confidence interval, then it is considered as an outlier. The confidence interval is defined statistically according to the distribution of values. On top of the notion, we propose a level-wise algorithm for the discovery of frequent patterns. In each pass, candidate patterns are tested on the traversals to count their supports, and then evaluated with respect to the supports to become frequent patterns. The frequent patterns are joined together to generate one-step larger candidates. It proceeds until no more candidates are generated. The frequent patterns are further ranked according to their priority. It reflects other aspects of the patterns, such as the connectivity and vertex weights besides the support.

This paper is organized as follows. In Section 2, we review previous works related with the traversal pattern mining and weighted mining. Section 3 proposes an algorithm for the discovery of frequent patterns from weighted traversals on graph. In Section 4, we experiment and analyze the algorithm on synthetic data. Finally, Section 5 contains the conclusion and future works.

Manuscript received May 5, 2006.
Manuscript revised May 25, 2006.

2. Related Works

The main stream of data mining, which is related to our work, can be divided into two categories, i.e. the traversal pattern mining and the weighted mining. For the traversal pattern mining, there have been few works. Chen et al. [3] 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. [4, 5] 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. Although the above works dealt with the mining of traversal patterns, to the best of our knowledge, there is no work which considers the notion of weight as our one.

For the weighted mining, most of previous works are related to the mining of association rules and its sub-problem, the discovery of frequent itemsets. Cai et al. [6] 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. [7] extended the problem by allowing weights to be associated with items in each transaction. Their approach ignores the weights when finding frequent itemsets, but considers during the association rule generation. Tao et al. [8] proposed an improved model of weighted support measurement and the weighted downward closure property. Yun et al. [9] also considered weighted items in the process of frequent itemsets, and the length-decreasing support constraints for a new measurement of support. Although the above works take the notion of weight into account as examined in this paper, they can not be adapted directly to our work because they only concerned on the mining from items, but not from traversals.

3. Mining Frequent Patterns

The algorithm proposed in this paper is mainly composed of three phases. The graph augmentation phase is a pre-processing phase, in which each edge of the base graph is augmented with average weight and standard deviation. The frequent patterns discovery phase is the main phase, in which frequent patterns are discovered from the augmented graph and traversal database. The pattern priority phase is a post-processing phase, in which the frequent patterns are ranked according to their importance to users. We first define some related definitions and the problem statement, and then propose algorithms for the phases.

3.1 Definitions and Problem Statement

Definition 1. A *simple directed graph* is a finite set of vertices and edges, in which each edge joins one ordered pair of vertices. The graph contains no self loop which joins a vertex with itself. A *base graph* is a simple directed graph, on which traversals occur.

Definition 2. A *traversal* is a sequence of consecutive edges of a base graph. It can be represented with a sequence of the connecting vertices of each edge, thus a traversal $t = \langle v_1, v_2, \ldots, v_n \rangle$. A weighted traversal is a traversal, in which each edge has an associated weight. Thus a traversal t with associated weights w is represented as $(t, w) = (\langle v_1, v_2, \ldots, v_n \rangle, \langle w_1, w_2, \ldots, w_{n-1} \rangle)$, where w_i means the weight of edge $\langle v_i, v_{i+1} \rangle$. A *traversal database* is a set of weighted traversals.

Definition 3. A *subtraversal* is any subsequence of consecutive vertices in a traversal. If $t = \langle v_1, v_2, \ldots, v_n \rangle$ is a traversal, then $s = \langle s_1, s_2, \ldots, s_m \rangle$ is a subtraversal of t when there exists a $k \geq 0$ such that $t_{j+k} = s_j$ for all $1 \leq j \leq m$. If an arbitrary *pattern* is a subtraversal of a traversal, then we said that the pattern is contained in the traversal, and vice versa, the traversal contains the pattern.

Definition 4. Let $G = (V, E)$ be a base graph, and T be a traversal database, then an *augmented graph* G_w is defined as follows. Each node $v_i \in V$ is assigned with a weight w_i. Each edge, $\langle v_i, v_j \rangle \in E$, is labeled with a pair of average weight and standard deviation, (μ_{ij}, σ_{ij}), which are obtained from the weights of the corresponding edges of traversals in T.

Definition 5. A *confidence interval* is an interval between two numbers, within which a random variable X lies with a *confidence level.* In our problem, if a weight lies within the confidence interval, then it is considered as a confident one, but if it lies outside the confidence interval, then it is considered as an outlier.

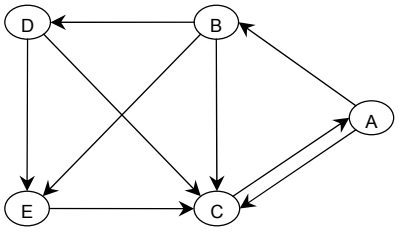
The problem concerned in this paper is stated as follows. Given a base graph and weighted traversals on the graph, find all patterns contained in the traversals in a ratio of larger than *minSup*. This ratio is called *support*, and a pattern with the support larger than *minSup* is called *frequent*. When counting the support, the weights of traversals should lie within a specified *confidence interval*. In addition, determine the priority of frequent patterns according to their importance criteria besides the support.

3.2 Augmentation of Base Graph

When a base graph and weighted traversals are given, first phase of the algorithm is to augment the base graph with

supplementary information to be used for the subsequent phases. The supplementary information includes the average weight and standard deviation for each edge, and the weight for each vertex.

Fig. 1 depicts an example of base graph and traversal database. On the base graph, all the traversals traverse the vertices through the edges. The traversal #1, for example, traverses consecutively the vertices *A, B* and *C* through the edge <*A B*> with the weight 2.2, and <*B C*> with 2.0.



(a) Base graph

ID	Traversal	Weight
1	<A B C>	<2.2 2.0>
2	<B D E C A>	<3.0 4.3 3.5 3.1>
3	<C A B D>	<2.9 2.0 4.0>
4	<D C A>	<4.0 3.0>
5	<B C A>	<2.2 2.9>
6	<A B E C>	<2.1 3.4 3.2>
7	<A B D E C>	<1.4 3.9 4.4 3.2>
8	<B E C>	<2.3 3.4>
9	<B D C>	<3.8 3.1>
10	<C A B D>	<2.5 2.2 4.1>

(b) Traversal database

Fig. 1 Example of base graph and traversal database

Given the base graph and traversal database, the base graph can be augmented as follows. For each edge of the base graph, we can collect the corresponding weights of the edge from the traversal database, and then calculate the average and standard deviation. For the edge <*A B*> as an example, we can collect the weight values, 2.2, 2.0, 2.1, 1.4 and 2.2, and then calculate the average 2.0 and the standard deviation 0.3. Fig. 2 shows the augmented graph as the result. Each vertex is also assigned with an arbitrary weight, which may reflect the importance of the vertex.

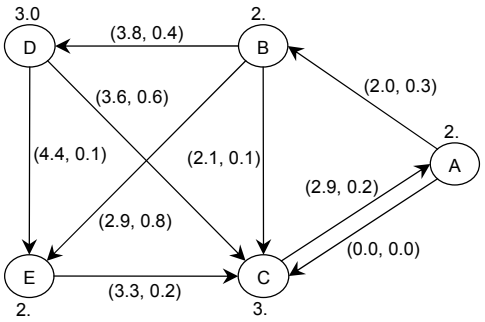


Fig. 2 Example of augmented graph

3.3 Discovery of Frequent Patterns

Main phase of the algorithm is to find frequent patterns from the traversal database and augmented graph. To derive the algorithm, we first investigate an important property of patterns. Let the length of a pattern be the number of vertices contained in it. On the augmented graph, any pattern $p = \langle p_1, p_2, \ldots, p_k \rangle$ of length k has exactly two subpatterns of length $k-1$, i.e., $\langle p_1, p_2, \ldots, p_{k-1} \rangle$ and $\langle p_2, p_3, \ldots, p_k \rangle$. For example, a pattern <*A B D E C*> in Fig. 2 has two subpatterns, <*A B D E*> and <*B D E C*>. Therefore, a pattern of length k is frequent only if its two subpatterns of length $k-1$ are also frequent. Such downward closure property allows us to develop a level-wise algorithm like the Apriori algorithm [1].

Fig. 3 shows the algorithm proposed in this paper, which performs in a level-wise manner. It initializes the candidate patterns of length 1 with all vertices of the augmented graph. In each pass of the algorithm, the traversal database is scanned to count the supports of all candidates. The supports are then adjusted according to the specified confidence interval. Next, frequent patterns are determined from the candidates whose supports are larger than the specified minimum support. Finally, new candidates are obtained from the frequent patterns for next pass. It repeats until no more candidates are generated.

Input: augmented graph G_w, traversal database T,
minimum support *minSup*, confidence interval *confInv*
Output: frequent patterns L_k

// initialize candidate patterns of length 1
$C_1 \leftarrow$ set of all vertices
$k = 1$

while ($|C_k| > 1$) { *// while candidates exist*

// count supports for candidate patterns
for each traversal $t \in T$ {
$P = \{p \mid p \in C_k, p$ is a subtraversal of $t\}$
$\forall p \in P$ $p.count$++

```
    }

    // prune candidate patterns w.r.t confInv
    if (k ≥ 2) C_k ← pruneCandidates(C_k, G_w, confInv)

    // generate frequent patterns
    L_k = {p | p ∈ C_k, p.count ≥ minSup}

    // generate candidate patterns for next pass
    C_{k+1} ← genCandidates(L_k, G_w)
    k++
}
```

Fig. 3 Algorithm for frequent patterns

In the algorithm, *pruneCandidate*() adjusts the supports of candidate patterns as follows. Given a pattern $p = <p_1, p_2, ..., p_k>$ is a subtraversal of a weighted traversal $(t, w) = (<v_1, v_2, ..., v_n>, <w_1, w_2, ..., w_{n-1}>)$. If there is an edge $<v_i, v_j>$ in the part of the traversal coincided with the pattern, whose weight w_i lies outside the confidence interval, then the traversal can not contribute for the support of the pattern. For example, even though a pattern $<A\ B\ D>$ is contained in the traversal #7 $(<A\ B\ D\ E\ C>, <1.4\ 2.3\ 4.4\ 3.2>)$ of Fig.1 (b), the traversal can not contribute for the support because its edge $<A\ B>$ has the weight 1.4 which lies outside the confidence interval 1.41~2.59. For the determination of confidence interval for each edge of the augmented graph, we assume that the distribution of weight values follows the normal distribution. As in almost applied practices, if the confidence interval is stated at the 95% confidence level, then $P(\mu - 1.96\sigma \le X \le \mu + 1.96\sigma) = 0.95$, where μ is the average and σ is the standard deviation. In other words, 95% of weight values are considered to exist within the confidence interval, $(\mu - 1.96\sigma) \sim (\mu + 1.96\sigma)$, but remaining 5% exist outside. For example, the edge $<A\ B>$ in Fig. 2 has the confidence interval, $(2.0 - 1.96 \times 0.3) \sim (2.0 + 1.96 \times 0.3) \equiv 1.41 \sim 2.59$. If a weight value lies outside this interval, then it can be considered as an outlier. Therefore, traversals whose edges have such weight values can not contribute for the support of patterns.

genCandidates() generates new candidate patterns for next pass. By the downward closure property, new candidates of length $k+1$ can be obtained by joining the frequent patterns of length k. If there are two frequent patterns of length k, $<p_1, p_2, ..., p_k>$ and $<p_2, p_3, ..., p_{k+1}>$, a new candidate pattern of length $k+1$, $<p_1, p_2, ..., p_{k+1}>$ is obtained. For example, $<A\ B\ C>$ and $<B\ C\ D>$ result in $<A\ B\ C\ D>$. Note that $<A\ B\ C>$ and $<C\ D\ E>$, but, can not be joined to make $<A\ B\ C\ D\ E>$.

An example of the algorithm is shown in Fig. 4, which is derived from the traversal database in Fig. 1 (b), and the augmented graph in Fig. 2. We assume the minimum support as 2, and the confidence level as 95%. The algorithm initializes the candidates C_1 of length 1 with all the vertices. By scanning the database, the support of each candidate is determined as shown in C_1. The candidates, whose support is larger than 2, become the frequent patterns of length 1 as in L_1. By joining the frequent patterns, new candidates of length 2 are obtained as in C_2, after deleting non-existing edges in the augmented graph. The database is again scanned to count the support of the candidates. The supports are then adjusted by using the confidence interval. For example, the support of the pattern $<A\ B>$ is 5 initially, but must be decreased to 4. This is because the weighted traversal #7, $(<A\ B\ D\ E\ C>, <1.4\ 2.3\ 4.4\ 3.2>)$ can not contribute for the support because the weight 1.4 of the edge $<A\ B>$ lies outside the confidence interval 1.41~2.59. From the adjusted candidates, the frequent patterns L_2 are obtained. Again, the candidates of length 3, C_3, are obtained by joining the L_2. For example, $<A\ B>$ and $<B\ C>$ result in $<A\ B\ C>$. The algorithm proceeds similarly up to the L_3, and then terminates because there is no candidate of length 4 by joining the L_3.

C_1

Candidate Pattern	Pruned Support
<A>	8
<B>	9
<C>	10
<D>	6
<E>	4

L_1

Frequent Pattern	Pruned Support
<A>	8
<B>	9
<C>	10
<D>	6
<E>	4

C_2

Candidate Pattern	Pruned Support
<A B>	4
<A C>	0
<B C>	2
<B D>	4
<B E>	2
<C A>	4
<D C>	2
<D E>	2
<E C>	4

L_2

Frequent Pattern	Pruned Support
<A B>	4
<B C>	2
<B D>	4
<B E>	2
<C A>	4
<D C>	2
<D E>	2
<E C>	4

C_3

Candidate Pattern	Pruned Support
<A B C>	1
<A B D>	2
<A B E>	1
<B C A>	1
<B D C>	1
<B D E>	1
<B E C>	2
<C A B>	1
<D C A>	1
<D E C>	2
<E C A>	1

L_3

Frequent Pattern	Pruned Support
<A B D>	2
<B E C>	2
<D E C>	2

Fig. 4 Example of frequent patterns

3.4 Priority of Patterns

The algorithm determined the importance of frequent patterns, as in the previous works, according to the number of their occurrences in the traversals. Although such support is concerned as the primary criterion for most problems, variety of supplementary information can be adopted as secondary criteria. This paper proposes one possible criterion as follows.

Priority of a pattern p =
support of p
+ number of edges incident into p / total edges
+ sum of edge weights of p / total edge weights
+ sum of vertex weights of p / total vertex weights

The priority of patterns are determined by the combination of support, ratio of incident edges, ratio of edge weights, and ratio of vertex weights. The reasoning behind the combination is that a pattern becomes more important as it occurs more times, more referred from other vertices, and contains edges and vertices with higher weights. Fig. 5 shows the pattern priority of the frequent patterns. Although the three patterns have the same support, they can be further ranked according to their priority.

Frequent Pattern	Pruned Support	Pattern Priority	Rank
<A B D>	2	2.93	3
<B E C>	2	3.28	2
<D E C>	2	3.42	1

Fig. 5 Example of pattern priority

4. Experiments

We conducted several experiments on the algorithm, specifically to evaluate the effect of confidence interval. The experiment adopts Windows 2000 Professional as the operating system, Microsoft Visual C++ 6.0 for the programming language, and Microsoft SQL Server 2000 as the database. During the experiment, base graphs are generated synthetically according to the parameters, i.e., number of vertices and average number of edges per vertex. We then generate traversals, each of which traverses on the base graph. During the generation, weights are assigned to the edges, and have the normal distribution.

Fig. 6 shows the effect of confidence interval on the length of frequent patterns. This experiment uses a base graph with 100 vertices and 20,000 edges, i.e., 20 average edges per vertex. The number of traversals varies from 10,000 to 50,000, and the maximum length of traversals is 51. The minimum support is 5%, which means that a pattern can be frequent only if it is subtraversals of more than 5% of the traversals. The confidence level is 95%, which means that roughly 95% of edge weights are confident, and remaining 5% are outliers. As shown in the figure, the length of frequent patterns is much lower when considering the confidence interval. This means that the exclusion of outliers by the confidence interval allows us to discover more reliable frequent patterns.

Fig. 7 shows the trend of the length of frequent patterns according to the confidence level. The experiment is conducted on 100,000 traversals. As shown in the figure, the length of frequent patterns becomes larger as the confidence level increases. The length changes from 15 up to 51 as the confidence level increases from 80% to 100%. The change is very steep, so we need to select the confidence level with intention.

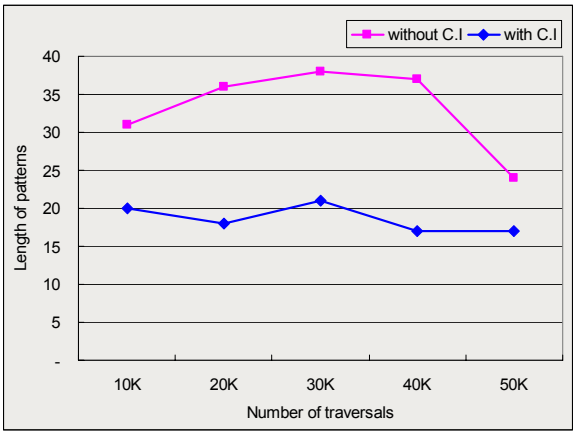


Fig. 6 Length of frequent patterns w.r.t the number of traversals

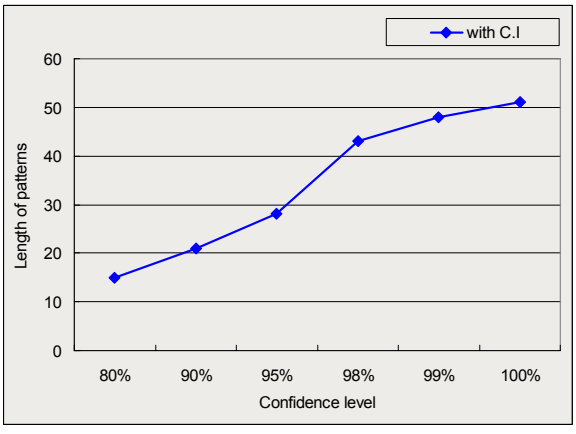


Fig. 7 Length of frequent patterns w.r.t confidence level

5. Conclusions

This paper examined the problem of discovering frequent patterns from weighted traversals on graph. Differently from previous approaches, traversals have weights that reflect their importance. We presented a level-wise algorithm which takes the weights into account in the measurement of support. The traversals, which have weights outside the confidence interval, are treated as outliers, and do not contribute to the support. Through his approach, we can discover more reliable frequent patterns. The patterns are further ranked according to their priority, which reflects several criteria of patterns beside the support. We are currently working on new support criteria such as weighted support, and applications such as Web mining.

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] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2001.

[3] 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.

[4] 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.

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

[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.

[7] 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.

[8] 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.

[9] 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.

.