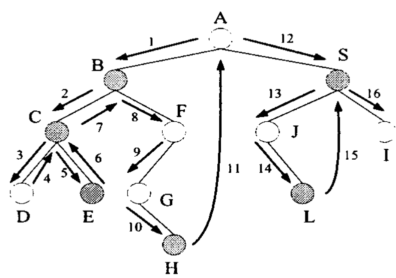
Using Pattern-Join and Purchase-Combination for Mining Web Transaction Patterns in an Electronic Commerce Environment

Ching-Huang Yun and Ming-Syan Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: chyun@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract

In this paper, we explore the data mining capability which involves mining Web transaction patterns for an electronic commerce (EC) environment. To better reflect the customer usage patterns in the EC environment, we propose a mining model that takes both the traveling patterns and purchasing behavior of customers into consideration. We devise two efficient algorithms (MTS_{PJ}, and MTS_{PC}) for determining the frequent transaction patterns, which are termed large transaction patterns in this paper. In addition, algorithm WTM devised in our prior work, is used for comparison purposes. By utilizing the path-trimming technique which is developed to exploit the relationship between traveling and purchasing behaviors, MTS_{PJ} and MTS_{PC} are able to generate the large transaction patterns very efficiently. A simulation model for the EC environment is developed and a synthetic workload is generated for performance studies.



(a)

Web Transaction	
Node	Items
A	-
B	i_1
C	-
D	-
C	-
E	i_4
C	i_2
B	-
F	-
G	-
H	i_6
A	-
S	i_7
J	-
L	i_9
S	-
I	-

(b)

Figure 1. An illustrative example for a Web transaction where nodes are marked gray if items are purchased there.

1 Introduction

With the rapid growth of the information sources available in the World Wide Web, it has become increasingly important to efficiently analyze usage patterns in various emerging Web applications [4][8]. In some existing electronic commerce environments, Web pages are usually designed as shop-windows, and customers can visit these Web pages and make Web transactions through the Web interface. One example scenario for a Web transaction is shown in Figure 1, where a customer travels along the EC system and purchases a set of items in the corresponding nodes (i.e., Web pages) of his/her traversal path. Figure 1(a) shows the traveling pattern of this customer and the Web transaction data is shown in Figure 1(b), where for example, item i_1 was purchased when the customer visits node B. Note that mining information from such an EC system can provide very valuable information on customer buying behavior and the quality of business strategies can then be improved [2][6].

Mining of databases has attracted a growing amount of attention in database communities due to its wide applicability in industry for improving marketing strategies [1][3]. Recently, data mining for Web usage has been drawing an increasing amount of attention from both research and industrial communities. Web usage mining is the process of discovering interesting patterns in Web usage data. A study on efficient mining of path traversal patterns for capturing Web user behavior was conducted in [4]. WEBMINER [5] applies Apriori-related algorithms for mining Web usage association rules and sequential patterns. WebLogMiner [8] proposes techniques for using OLAP and data mining to discover Web access patterns in Web access logs.

In this paper, we shall explore the data mining capability which involves mining *Web transaction patterns* for an EC environment where customers can seek for items of interest for possible purchases [7]. For the measurement of customer purchasing behavior association, a novel knowledge (called *Web transaction rules*) can be derived from the Web transaction patterns. Clearly, the distinctive characteristics of mining Web transaction patterns increase the difficulty of extracting information from the Web transaction data. The design and development of efficient mining algorithms for mining Web transaction patterns is taken as the objective of this paper.

Consequently, to better reflect the customer usage patterns in the EC environment, we propose a mining model that takes both the traveling patterns and purchasing behavior of customers into consideration. First, for each Web transaction, we develop *BTB (Backward-Then-Branching)* algorithm, to extract meaningful Web transaction records from the given Web transaction. After all the Web transaction records are derived from Web transactions, three algorithms are developed for determining the *large transaction patterns* from the Web transaction records, where a large transaction pattern is a transaction pattern that appeared in a sufficient number of Web transactions. In our prior work [7], algorithm *WTM (Web Transaction Mining)* is devised by directly extending the schemes on prior work in mining path traversal patterns [4] to mine Web transactions. However, without utilizing the paths of large transaction patterns, WTM may generate a lot of *unqualified candidate transaction patterns*, thus degrading the performance. In contrast, algorithm *MTS_{PJ} (Maximal-Transaction-Segment with Pattern Join)* and *MTS_{PC} (Maximal-Transaction-Segment with Purchase Combination)*, are developed based on the *path-trimming* technique to explore the fact that one can trim the generation of the candidate transaction patterns according to the paths traversed and keep, in the same maximal transaction segment, the related information which appears in the same path. A simulation model for the EC environment is developed and a synthetic workload is generated for performance studies. By utilizing the maximal transaction segment for path-trimming, MTS_{PJ} and MTS_{PC} are shown to outperform WTM in execution efficiency.

This paper is organized as follows. Preliminaries are given in Section 2. In Section 3, three algorithms, WTM, MTS_{PJ}, and MTS_{PC}, are devised for mining Web transaction patterns. Experimental studies are conducted in Section 4. This paper concludes with Section 5.

2 Preliminaries

Let N = $\{n_1, n_2, ..., n_g\}$ be a set of nodes in the EC environment and I = $\{i_1, i_2, ..., i_h\}$ be a set of items sold in the EC system. We then have the following definitions:

Definition 1 *Let $\{s_1 s_2 ... s_y\}$ be a path sequence, where $\{s_1, s_2, ..., s_y\} \subseteq N$. $\{s_1 s_2 ... s_y\}$ is said to* **path-contain** *a path $\{r_1 r_2 ... r_j\}$ as a consecutive subsequence if there exists an z such that $s_{x+z} = r_x$, for $1 \leq x \leq j$.*

Definition 2 *Let $<s_1 s_2 ... s_y : n_1\{i_1\}, n_2\{i_2\}, ..., n_x\{i_x\}>$ be a transaction pattern, where $i_m \subseteq I$ for $1 \leq m \leq x$, and $\{n_1, n_2, ..., n_x\} \subseteq \{s_1, s_2, ..., s_y\} \subseteq N$. Then, $<s_1 s_2 ... s_y : n_1\{i_1\}, n_2\{i_2\}, ..., n_x\{i_x\}>$ is said to* **pattern-contain** *a transaction pattern $<w_1 w_2 ... w_q : r_1\{t_1\}, r_2\{t_2\}, ..., r_p\{t_p\}>$ if and only if $\{s_1 s_2 ... s_y\}$ path-contains $\{w_1 w_2 ... w_q\}$ and $\{n_1\{i_1\}, n_2\{i_2\}, ..., n_x\{i_x\}\}$ contains $\{r_1\{t_1\}, r_2\{t_2\}, ..., r_p\{t_p\}\}$.*

Definition 3 *A Web transaction is said to* **pattern-contain** *$<w_1 w_2 ... w_q : r_1\{t_1\}, r_2\{t_2\}, ..., r_p\{t_p\}>$ if one of its Web transaction records pattern-contains $<w_1 w_2 ... w_q : r_1\{t_1\}, r_2\{t_2\}, ..., r_p\{t_p\}>$.*

Note that a Web transaction consists of a set of purchases along the corresponding nodes in its traversal path, such as the example shown in Figure 1. Unlike the work in the path traversal patterns [4], the mining on Web transaction patterns takes both traveling patterns and purchasing behavior into consideration. The Web transaction rule, derived from Web transaction patterns in this paper, is an implication of the form $<n_1 n_2 ... n_y : X \Longrightarrow Z>$, where X and Z both are sets of purchases, $X \cap Z = \Phi$, and $\{n_1, n_2, ..., n_y\} \subseteq N$. The rule $<n_1 n_2 ... n_y : X \Longrightarrow Z>$ has *support s* in the Web transaction database D if s% of the Web transactions in D pattern-contain $<n_1 n_2 ... n_y : X \cup Z>$. Also, the rule $<n_1 n_2 ... n_y : X \Longrightarrow Z>$ holds with *confidence c* if c% of Web transactions in D that contain X also contain Z along the path $\{n_1 n_2 ... n_y\}$.

Before devising algorithms for determining large transaction patterns in Section 3, we shall first present algorithm BTB devised for deriving meaningful Web transaction records from each Web transaction. Explicitly, given a Web transaction of a customer, BTB proceeds as follows. First, BTB traces the Web transaction to output the customer transaction records, where a customer transaction record consists of a path and the items purchased in the corresponding nodes of that path. Each customer transaction record is used as a branch for constructing the customer transaction tree. Algorithm BTB, i.e., Backward-Then-Branching, is named for the reason that it outputs the record stored in the buffer to become a branch of a customer transaction tree as long as this customer makes backward movement first and branching movement later. After the customer transaction tree is constructed, BTB will then traverse the tree in a depth-first manner to output Web transaction records. When traveling to a leaf node, BTB outputs a Web transaction record which consists of a path from the root node to that leaf node and a set of purchases made along the path. Finally, BTB stores all Web transaction records, according to the corresponding Web transaction identifiers, into the database.

3 Algorithms for Mining Web Transaction Patterns

In Section 3.1, algorithm WTM described in [7], is a procedure of mining large transaction patterns. By utilizing the maximal transaction segment for path-trimming, we devise two algorithms, algorithm MTS_{PJ} and algorithm MTS_{PC}, for efficiently determining large transaction patterns. Let C_k be a set of candidate k-transaction patterns and T_k represent the set of large k-transaction patterns. Because both MTS_{PJ} and MTS_{PC} generate T_k along with the generation of C_{k+1}, we use *round* k to refer to the procedure executed to obtain (T_k, C_{k+1}). In Section 3.2, MTS_{PJ} utilizes the pattern join scheme for the generation of candidate transaction patterns. In Section 3.3, algorithm MTS_{PC} is improved by employing the purchase combination scheme, and is able to generate fewer uncertain candidate transaction patterns than MTS_{PJ} in each round, thus reducing the computational overhead.

3.1 Algorithm WTM

Similarly to scheme FS in [4], WTM [7] joins the purchased itemsets for generating candidate transaction patterns. However, unlike FS, WTM employs a two-level hash tree, called Web transaction tree, to store candidate transaction patterns. Figure 2 is an illustrative example for mining Web transaction patterns with WTM and Figure 3 is one part of the Web transaction tree storing the C_2 in Figure 2. According to each Web transaction record of a Web transaction, the support of a candidate transaction pattern is determined by the number of Web transactions that pattern-contain this candidate transaction pattern. WTM then obtains large transaction patterns during the procedure of destructing the Web transaction tree. Each large transaction pattern is generated when its support exceeds the minimum support. For example, one can destruct the Web transaction tree in Figure 3 to determine the T_2 in Figure 2. When traversing to node E, two large 2-transaction patterns {<ABCE: B{i_1}, E{i_4}>, <ABCE: C{i_2}, E{i_4}>} are generated.

Consider the example scenario in Figure 2. In the first round, WTM constructs the Web transaction tree by hashing each Web transaction record to construct the Web transaction tree and counts the support of individual purchases. Then, WTM destructs the Web transaction tree for deriving T_1, the set of large 1-transaction patterns, and utilizes the purchased itemsets in T_1 for generating C_2, the set of candidate 2-transaction patterns to be stored in a Web transaction tree. In each subsequent round, WTM starts with candidate transaction patterns found in the previous round for the counting of supports and identifies the large transaction patterns. Then, WTM proceeds to the generation of new candidate transaction patterns and stores them to the Web transaction tree. To illustrate the operations of WTM, it can be seen from Figure 2 that both the Web transactions with WT_ID = 100 and WT_ID = 200 contain one Web transaction record <ABCE: B{i_1}, C{i_2}, E{i_4}> that pattern-contains <ABCE: B{i_1}, E{i_4}>. Also, in the Web transaction with WT_ID = 300, one Web transaction record <ABCE: B{i_1}, E{i_4}> pattern-contains <ABCE: B{i_1}, E{i_4}>. Thus, the occurrence count of <ABCE: B{i_1}, E{i_4}> is 3. In addition, consider the counting of the candidate set C_1 for example. In the Web transaction with WT_ID = 100, two Web transaction records <ABCE: B{i_1}, C{i_2}, E{i_4}> and <ABFGH: B{i_1}, H{i_6}> pattern-contain <AB: B{i_1}>. Though both pattern-containing <AB: B{i_1}>, these two Web transaction records only account for one more support count for <AB: B{i_1}> since they are from the same Web transaction 100, thus avoiding the duplicate counting in the different Web transaction records of the same Web transaction. Hence, the final support of <AB: B{i_1}> is 3.

Database

Web Transactions		
WT_ID	Path	Purchases
100	ABCE	B{i_1}, C{i_2}, E{i_4}
	ABFGH	B{i_1}, H{i_6}
	ASJL	S{i_7}, L{i_9}
200	ABCE	B{i_1}, C{i_2}, E{i_4}
	ASJLQ	S{i_7}, Q{i_{10}}
300	ABCE	B{i_1}, E{i_4}
	ABFG	B{i_1}, G{i_5}
	ASJL	S{i_7}, J{i_8}, L{i_9}
400	ABD	D{i_3}
	ABFG	G{i_5}
	ASJLQ	S{i_7}, J{i_8}, Q{i_{10}}

C1

Path	Purchase	Sup.
AB	B{i_1}	3
ABC	C{i_2}	2
ABD	D{i_3}	1
ABCE	E{i_4}	3
ABFG	G{i_5}	2
ABFGH	H{i_6}	1
AS	S{i_7}	4
ASJ	J{i_8}	2
ASJL	L{i_9}	2
ASJLQ	Q{i_{10}}	2

T1

Path	Purchase	Sup.
AB	B{i_1}	3
ABC	C{i_2}	2
ABCE	E{i_4}	3
ABFG	G{i_5}	2
AS	S{i_7}	4
ASJ	J{i_8}	2
ASJL	L{i_9}	2
ASJLQ	Q{i_{10}}	2

C2

|C2|=28

Path	Purchases	Sup.
ABC	B{i_1}, C{i_2}	2
ABCE	B{i_1}, E{i_4}	3
⋮		
AS	B{i_1}, S{i_7}	0
ASJ	B{i_1}, J{i_8}	0
⋮		
ASJLQ	J{i_8}, Q{i_{10}}	1
ASJLQ	L{i_9}, Q{i_{10}}	0

T2

Path	Purchases	Sup.
ABC	B{i_1}, C{i_2}	2
ABCE	B{i_1}, E{i_4}	3
ABCE	C{i_2}, E{i_4}	2
ASJ	S{i_7}, J{i_8}	2
ASJL	S{i_7}, L{i_9}	2
ASJLQ	S{i_7}, Q{i_{10}}	2

C3

Path	Purchases	Sup.
ABCE	B{i_1}, C{i_2}, E{i_4}	2

T3

Path	Purchases	Sup.
ABCE	B{i_1}, C{i_2}, E{i_4}	2

Figure 2. An illustrative example for mining Web transaction patterns with algorithm WTM.

After all large transaction patterns are obtained, one can derive the Web transaction rules from large transaction patterns. In this example, <ABCE : B{i_1}, C{i_2}, E{i_4}> is one large 3-transaction pattern with support = 2 and <AB : B{i_1}> is one large 1-transaction pattern with support = 3. As a result, we can derive one Web transaction rule

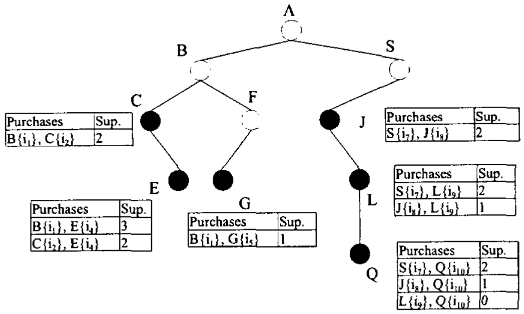


Figure 3. The Web transaction tree for storing candidate 2-transaction patterns.

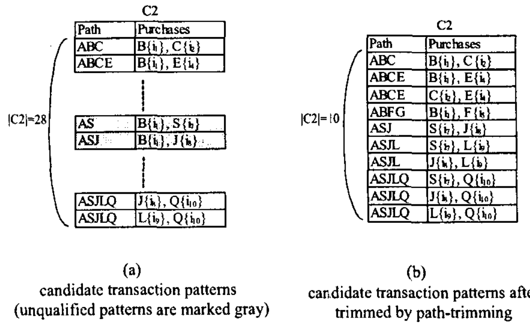


Figure 4. Path-trimming technique by using the maximal transaction segment.

<ABCE : B{i_1} $\Longrightarrow$ C{i_2}, E{i_4}> with the support equal to support(<ABCE : B{i_1}, C{i_2}, E{i_4}>) = 2 and the confidence equal to $\frac{support(<ABCE:B\{i_1\},C\{i_2\},E\{i_4\}>)}{support(<AB:B\{i_1\}>)} =$ 67%.

3.2 Algorithm MTS$_{PJ}$

As can be seen from Figure 2, without utilizing the paths of large transaction patterns, algorithm WTM generates a lot of unqualified candidate transaction patterns, thus degrading the performance. For example, one candidate 2-transaction pattern <AS: B{i_1}, S{i_7}> is generated by joining large 1-transaction patterns <AB: B{i_1}> and <AS: S{i_7}>. However, <AS: B{i_1}, S{i_7}> is never counted during the support counting because the nodes of the path AS do not even contain the node B of the purchase B{i_1}. This problem is called the *unqualified candidate transaction pattern problem.* This implies that one can trim the generation of the candidate transaction patterns according to the paths traversed. In light of this concept of path-trimming, algorithm MTS$_{PJ}$ is designed to solve the unqualified candidate transaction pattern problem. Explicitly, during the generation of large transaction patterns, by destructing the Web transaction tree, MTS$_{PJ}$ not only determines large transaction patterns but also uses a buffer to keep a segment that contains large transaction patterns and the *maximal path* so as to properly classify the patterns, where the maximal path corresponds to a path from the root node to the leaf node of the Web transaction tree. This segment is called the maximal transaction segment in that MTS$_{PJ}$ joins large transaction patterns for generating candidate transaction patterns only when the leaf node of the Web transaction tree is reached. The purpose of classifying the patterns is that the patterns, whose paths do not path-contain each other, need not be considered to generate candidate transaction patterns together.

Explicitly, MTS$_{PJ}$ applies the pattern join scheme to the large (k-1)-transaction patterns in each segment (denoted as T^S_{k-1}) for the generation of candidate k-transaction patterns in the same segment (denoted as C^S_k). Also, $C^{'}_k$ is the set of uncertain candidate k-transaction patterns generated from the pattern joins. In round k-1, we can derive $T_{k-1} = \sum T^S_{k-1}$ and $C^{'}_k = \sum C^S_k = \sum(T^S_{k-1} * T^S_{k-1})$, where $T^S_{k-1} * T^S_{k-1}$ means joining the large (k-1)-transaction patterns in each segment. C_k is derived after the (k-1)-subpattern identifications in $C^{'}_k$. Note that the candidate transaction patterns generated in the different segments may overlap, but those redundant ones will be detected and deleted when they are inserted into the Web transaction tree. For the example shown in Figure 2, WTM generates 28 candidate 2-transaction patterns shown in Figure 4(a) whereas MTS$_{PJ}$ generates 10 candidate 2-transaction patterns shown in Figure 4(b), showing a significant performance improvement of MTS$_{PJ}$ over WTM. This demonstrates the very advantage of the path-trimming technique MTS$_{PJ}$ employs.

3.3 Algorithm MTS$_{PC}$

Algorithm MTS$_{PC}$ is similar to algorithm MTS$_{PJ}$ in that it also employs the path-trimming concept of the maximal transaction segment to reduce the computational overhead, but is different from the latter in that MTS$_{PC}$, by utilizing the information in purchases, is able to reduce the number of uncertain candidate transaction patterns, thus further reducing the corresponding computational overhead and memory consumption. The method for MTS$_{PC}$ to reduce the number of uncertain candidate transaction patterns is described below. Based on the information in the pur-

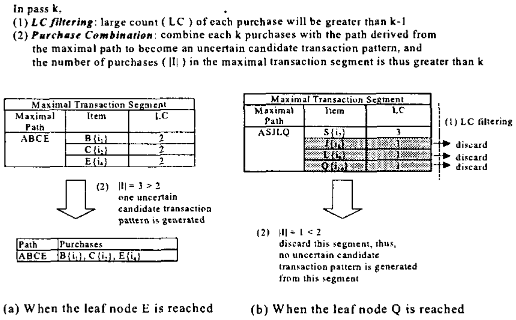


Figure 5. LC filtering and purchase combination techniques utilized by algorithm MTS$_{PC}$.

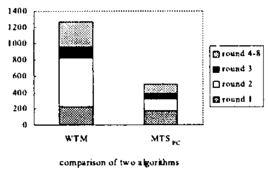
chases of large transaction patterns in the maximal transaction segment, MTS$_{PC}$ computes, for each purchase, the *large count (LC)* which refers to the number of purchases appearing in large transaction patterns, and utilizes the LC to further prune unqualified purchases (referred to as LC *filtering*). LC filtering is devised in light of the observation that for each purchase that can be qualified as a purchase in one of T_{k+1}, that purchase must appear in at least k large k-transaction patterns. In round k, when reaching the leaf node of the Web transaction tree, MTS$_{PC}$ discards those purchases whose LCs are smaller than k-1 by this technique of LC filtering. For the example shown in Figure 2, if there exists one large 3-transaction pattern <ABCE : B{i$_1$}, C{i$_2$}, E{i$_4$}>, then we know that there must exist three large 2-transaction patterns which are <ABC : B{i$_1$}, C{i$_2$}>, <ABCE : B{i$_1$}, E{i$_4$}>, and <ABCE : C{i$_2$}, E{i$_4$}>. In this example, purchases B{i$_1$}, C{i$_2$}, and E{i$_4$} in <ABCE : B{i$_1$}, C{i$_2$}, E{i$_4$}> all appear in two large 3-transaction patterns. Recall that $C_k^{'}$ is the set of uncertain candidate k-transaction patterns and $C_k^{'} = \sum C_k^S$. By utilizing the purchase combination scheme, MTS$_{PC}$ generates C_k^S by combining k purchases of those purchases satisfying the LC threshold. As such, the number of uncertain candidate transaction patterns, which will in turn lead to the subpattern identifications, can be reduced. After $C_k^{'}$ is obtained, the subpattern identification proceeds. In the subpattern identification of each uncertain candidate k-transaction pattern, its (k-1)-subpatterns are evaluated to see if they are large (k-1)-transaction patterns.

In Figure 3, when reaching the leaf node E, the scenario for MTS$_{PC}$ to utilize LC filtering and the purchase combination technique is shown in Figure 5(a). Similarly, when reaching node Q, MTS$_{PC}$ also examines the maximal transaction segment as shown in Figure 5(b). As a result, MTS$_{PC}$ is able to prune unqualified purchases by scrutinizing their LCs. In addition, MTS$_{PC}$ utilizes purchase combination to generate fewer uncertain candidate transaction patterns than the pattern join of MTS$_{PJ}$. For the same generating C_3 example, MTS$_{PJ}$ generates 4 uncertain candidate transaction patterns and three patterns are pruned in the prune step, leading to 12 subpattern identifications. In contrast, algorithm MTS$_{PC}$ only generates one uncertain candidate transaction pattern and no pattern needs to be pruned in the prune step, leading to 3 subpattern identifications. This shows the very advantage of the LC filtering and the purchase combination technique MTS$_{PC}$ employs.

4 Experimental Results

The method used by this study for generating synthetic Web transactions is similar to the one used in [4] with modification noted below. First, we construct a traversal tree and determine the items sold in the nodes of this tree to simulate the EC environment whose starting position is a root node of the tree. The traversal tree consists of two types of nodes, namely internal nodes and leaf nodes. The number of child nodes at each internal node is called *fanout* and is determined from a uniform distribution. The percentage of nodes with item selling is denoted by N_I and those nodes (i.e., selling nodes) are determined randomly among all the internal nodes. The number of items sold in a selling node is denoted by n_i and the purchasing probability of the item in that node is denoted by P_b. When a customer visits the EC system, the Web transaction completed by this customer consists of a traversal path and a set of purchases made in the corresponding nodes. A traversal path consists of nodes accessed by a user. The size of each traversal path is determined from a Poisson distribution with mean equal to $|P|$. With the root node being the entrance node, Web transactions are generated probabilistically within the traversal tree as follows. For each node, the next hop is determined according to the probability model taken in [4]. In addition, the percentage of jumping to destination nodes N_D is also modeled and assigned to 1%, and those nodes are determined randomly among all the internal nodes.

Figure 6 shows the relative performance between WTM and MTS$_{PC}$, when N_I = 80 %, the fanout is between 4 and 7, the root node has 7 child nodes, the height of the tree is 10, the numbers of internal and leaf nodes are, respectively, 16,848 and 75,632, the number of items sold in those nodes is 13,478, $|D|$ = 200,000, s = 0.5%, P_b = 0.5, and $|P|$ = 30. Figure 6(a) shows that MTS$_{PC}$ outperforms WTM in each round and Figure 6(b) contains the patterns generated. Explicitly, in round 1, MTS$_{PC}$ uses the maximal transaction segment to classify the patterns and thus generates much fewer candidate 2-transaction patterns to construct the Web transaction tree than WTM. Hence, although it spends time in classifying the patterns, MTS$_{PC}$ incurs much shorter ex-



(a)

Round		WTM	MTS_{PC}
1	T_1	172	172
	C_2'	14706	723
	C_2	14706	723
2	T_2	167	167
	C_3'	1882	383
	C_3	178	178
3	T_3	151	151
	C_4'	223	156
	C_4	135	135
4	T_4	126	126
	C_5'	165	119
	C_5	106	106
5	T_5	98	98
	C_6'	90	55
	C_6	51	51
6	T_6	50	50
	C_7'	36	15
	C_7	15	15
7	T_7	15	15
	C_8'	9	2
	C_8	2	2
8	T_8	2	2
	C_9'	0	0
	C_9	0	0

(b)

Figure 6. Performance comparison between WTM and MTS$_{PC}$ in each round.

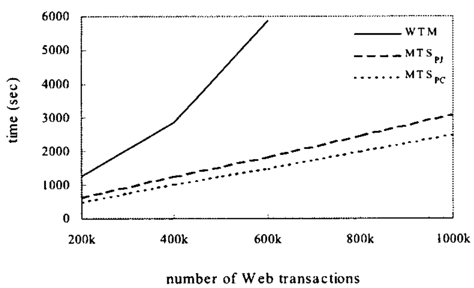


Figure 7. Execution time for WTM, MTS$_{PJ}$, and MTS$_{PC}$ when the database size increases.

ecution time than WTM. In round 2, MTS$_{PC}$ saves time by destructing a much smaller Web transaction tree constructed from round 1. By doing so, MTS$_{PC}$ uses the maximal transaction segment to classify the large 2-transaction patterns and utilizes the *LC* filtering to prune the unqualified purchases, which in turn results in much fewer uncertain candidate 3-transaction patterns generated. In the following rounds, MTS$_{PC}$ also outperforms WTM due to its advantages of path-trimming and purchase combination. To provide more insights into the maximal transaction segment devised for path-trimming, it is shown in Figure 7 that the execution times of MTS$_{PJ}$ and MTS$_{PC}$ increase linearly as the database size increases, indicating the good scale-up feature of MTS$_{PJ}$ and MTS$_{PC}$.

5 Conclusion

In this paper, we examined the issue of mining Web transaction patterns that takes both traveling patterns and purchasing behavior into consideration such that one can have a better model for an EC system and thus well capture and exploit the intrinsic relationship between these two customer behaviors. To address this issue, we developed algorithms for effectively mining Web transaction patterns. Two algorithms (MTS$_{PJ}$ and MTS$_{PC}$) are devised to determine large transaction patterns. By utilizing the path-trimming technique which is developed in light of the relationship between traveling and purchasing behaviors, MTS$_{PJ}$ and MTS$_{PC}$ are able to generate the Web transaction patterns very efficiently. A simulation model for the EC environment was developed and a synthetic workload was generated for performance studies.

Acknowledgments

The authors are supported in part by the National Science Council, Project No. NSC 89-2219-E-002-007, NSC 89-2213-E-002-032, and the Ministry of Education Project No. 89-E-FA06-2-4-7, Taiwan, Republic of China.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 478–499, September 1994.

[2] A. G. Buchner and M. Mulvenna. Discovery Internet Marketing Intelligence through Online Analytical Web Usage Mining. *ACM SIGMOD Record*, 27(4):54–61, Dec. 1998.

[3] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from a Database Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–833, 1996.

[4] M.-S. Chen, J.-S. Park, and P. S. Yu. Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, April 1998.

[5] R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.

[6] C. Schlueter and M. J. Shaw. A Strategic Framework for Developing Electronic Commerce. *IEEE Internet Computing*, 1(6):20–28, Nov./Dec., 1997.

[7] C.-H. Yun and M.-S. Chen. Mining Web Transaction Patterns in an Electronic Commerce Environment. *Proc. of the 4th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'2000)*, pages 216–219, April 2000.

[8] O. R. Zaiane, M. Xin, and J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. *Proc. Advances in Digital Libraries Conf. (ADL'98), Santa Barbara, CA*, pages 19–29, April 1998.