DISCOVER SEQUENTIAL TRANSITIONAL PATTERNS FROM TRANSACTIONAL DATABASES

Sujatha Dandu

Aurora's Technological and Research Institute, Department of Computer Science, Hyderabad

Deekshatulu B.L.

Distinguished Fellow, IDRBT, Hyderabad

Priti Chandra

Scientist, Advanced System Laboratory, Hyderabad

ABSTRACT: Transaction databases have been extensively mined for patterns of different kinds ranging from frequent patterns to maximal patterns and emerging patterns. Recent studies have extended the research in pattern mining to include the time stamp of each transaction resulting in the discovery of transitional patterns whose frequencies change significantly over time. This paper focuses on generating interesting sequences of transitional patterns. Transaction databases are divided into lists of transactions belonging to specific sequences. A sequence could be a set of transactions belonging to a customer in a retail store or a set of page visits by a specific user. Transitional Patterns are generated on each sequence and assigned unique ids so they can be treated like items and itemsets. The sequence database thus created can be input to any standard algorithm like GSP and PrefixSpan to generate sequences of transitional patterns. We have designed a new algorithm called the sequential TP-Mine by combining the steps of transitional pattern generation and sequence generation and conducted experiments by running this algorithm on multiple standard datasets.

KEYWORDS: Frequent Patterns, Sequential Patterns, Transitional Patterns, Association Rules.

1. INTRODUCTION

Frequent Patterns are widely applied in retail, medical and many other fields for knowledge discovery and aiding decision making. Many algorithms have been introduced to mine frequent patterns such as Apriori [AS94] and FP-Growth [H+04]. The results of these algorithms were numerous frequent patterns that made useful knowledge discovery like association rule mining difficult. This led to further research in the area of pattern mining with the discovery of new patterns such as closed frequent itemsets [P+99], indirect associations [WA06, B+99] and emerging patterns [BMR02].

Closed Frequent Itemsets algorithm generates only the largest frequent itemsets that have no supersets which are of same or higher frequency. Indirect associations focus on identifying rare item pairs that are expected to be frequently found. Emerging patterns are frequent patterns whose frequency increases dramatically from one dataset to another. None of the patterns mentioned above take into account the timing of transactions even though they all deal with transactional databases. Thus they do not take into account the distribution of item frequencies in a time series. This is important because a pattern may look frequent just by counting how many times it occurs in a database it is worthwhile to see if this support is evenly distributed or there is a sudden spurt or waning of support at any point of time.

A new frequent pattern called Transitional Pattern was introduced in [WA09, WA07] to do exactly that. A Transitional Pattern is a frequent pattern whose support experiences swings at specific points of time. A Transitional Pattern could be positive if its frequency increases significantly at a point of time and it is termed negative if its frequency decreases significantly at any point of time in the span of the database.

Research on frequent patterns has also focused on interesting sequences of itemsets in a transaction database. An itemset is a set of items in a transaction. A sequence is a set of itemsets where each itemset's occurrence follows the preceding one even if it did not occur immediately after the previous itemset. A sequence is an ordered list of itemsets. The problem of mining sequential patterns is to find the maximal sequences among all sequences that have a certain user-defined support. Each such maximal sequence is a sequential pattern. Many Algorithms such as will Apriori All [AS95], FreeSpan [H+00], Pattern

Growth [P+04] and GSP [AS96] have been written to discover sequential patterns.
When transitional patterns are discovered for specific sequences of data such as retail transactions belonging to each shopper at the store or page clicks belonging to each user of a web portal sequences of transitional patterns can be built. The aim of this paper is to identify such sequences for both positive and negative transitional patterns and analyze their potential for useful knowledge discovery.

2. TERMS AND DEFINITIONS

2.1 Definition of Sequential Positive Transitional Pattern (SPTP)

A sequential positive transitional pattern is a set of transitional patterns arranged in ascending order of their significant frequency ascending milestones (SFAM) [WA09] whose SFAMs occur within a predefined window or time interval.

2.2 Definition of Sequential Negative Transitional Pattern (SNTP)

A sequential negative transitional pattern is a set of transitional patterns arranged in ascending order of their significant frequency descending milestones (SFDM) [WA09] whose SFDMs occur within a predefined window or time interval.

3. SEQUENTIAL TRANSITIONAL PATTERNS

3.1 Sample Transaction Database

Consider an example transaction database TDB shown in Table 1. It has 16 transactions belonging to a particular customer over a 14 month period. There are 8 unique items numbered P1 through P8 that have been purchased by the customer during this period. This database has been adopted from [WA09] where it was used to demonstrate transitional patterns.

3.2 Generation of transitional patterns from TDB

Transitional patterns generated from this database based on the algorithm in [WA06] are listed in Table 2 and Table 3. Each of the transitional patterns is assigned a unique ID to allow these patterns to be treated like frequent itemsets and a sequence refers to the IDs of all transitional patterns that achieve their significant frequency ascending(or descending) milestone within a predefined interval like an hour, a day or a week. The sequence database consists of such sequences of positive or negative transitional patterns.

For instance < *P*1 > and < *P*1*P*3 > in Table 4 are a sequence of transitional patterns that achieve their significant frequency ascending milestones in August 2006.

Table 1. An example transaction database TDB

TID	List of Items	Time Stamp
001	P_1,P_2,P_3,P_5	Nov.2005
002	P_1,P_2	Dec.2005
003	P_1,P_2,P_3,P_8	Jan.2006
004	P_1,P_2,P_5	Feb.2006
005	P_1,P_2,P_4	Mar.2006
006	P_1,P_2,P_4,P_5,P_6	Apr.2006
007	P_1,P_2,P_3,P_4,P_6	May.2006
008	P_1,P_4,P_6	Jun.2006
009	P_4,P_5,P_6	Jul.2006
010	P_1,P_2,P_3,P_4,P_5,P_6	Aug.2006
011	P_1,P_3,P_4,P_6	Sep.2006
012	P_1,P_3,P_5	Oct.2006
013	P_1,P_2,P_3,P_6,P_7	Nov.2006
014	P_1,P_3,P_4,P_5	Dec.2006
015	P_1,P_3,P_4	Jan.2007
016	P_1,P_2,P_3,P_5	Feb.2007

Table 2. List of positive transitional patterns for TDB

Unique ID	Positive Transitional Patterns	Milestone
001	<p1>	62.5%
002	<p1p3>	62.5%
003	<p4p6>	37.5%

Table 3. List of negative transitional patterns for TDB

Unique ID	Negative Transitional Patterns	Milestone
001	<p1>	50%
002	<p1p2>	43.75%
003	<p4p6>	62.5%

3.3 Transitional patterns from multiple sequences

The pattern < *P*1*P*3 > may occur in many sequences of PTP but it is always referred with the same Unique ID of 2 in all of these sequences. Table 4 shows a list of positive transitional patterns generated for transactions belonging to multiple customers. The customer ID is the TDB from Table 1. This table has been built as follows. In the first step transitional patterns from all sequences in 2 and 3 are sorted in ascending order of their sequences and descending order of milestones within each sequence. In the next step the sorted records are sequentially read and those transitional patterns whose milestones occur within a user defined interval are grouped together as an itemset as defined in [AS95].

$< P1 >$ *and* $< P1P3 >$ in IDs 1 and 3 of 4 form of a sequence of positive transitional patterns since their transitions happen during the same milestone of 62.5%. Such sets are found for each sequence to build Table 4.

Table 4. PTP From multiple customers

Sequence ID	PTP	TP ID	Milestone
001	<p1>	1	62.5%
	<p1p3>	2	62.5%
	<p4p6>	3	37.5%
002	<p1p3>	2	62.5%
	<p4p6>	3	37.5
003	<p1p4>	4	71%
	<p1p3>	2	62.5%
	<p1>	1	62.5%

Transitional patterns in Table 4 for Sequence IDs 2 and 3 are for illustrating the creation of a sequence database and have not been generated from any real or sample transactional database.

Table 5. Sequence database of positive transitional patterns

Sequence ID	SPTP
001	<(3)(12)>
002	<(2)(3)>
003	<(4)(12)>

When a sequence generation algorithm like GSP is run on the sequence database in Table 5 we can finally generate a list of frequent sequences based on a user defined support level. For instance our example with a desired support level of 25% leads to discovery of sequential positive transitional patterns (SPTP) $< P1 >$ *and* $< P1P3 >$.
The same procedure of generating Tables 4 and 5 can be repeated for negative transitional patterns to generate sequential negative transitional patterns (SNTP).

3.4 Sequential TP-Mine Algorithm

Algorithm 1 Sequential TP-Mine Algorithm

Require: A transaction database containing several sets of transactions over a specified period of time, each belonging to a sequence. The database should be sorted by sequence ids and by the timestamp of transactions within a sequence.
Ensure: A transaction database containing several sets of transactions over a specified period of time, each belonging to a sequence.

1. Sort all transactions on ascending order of sequence and ascending order of timestamp within each sequence.
2. HashPTP=0, HashNTP=0, SeqPTP=0, SeqNTP=0
3. while not at end of all sequences do
4. Generate Transitional Patterns for the sequence of transactions resulting in the set of pos-itive transitional patterns S(PTP) and negative transitional patterns N(PTP) using the TP-Mine Algorithm defined in [QA09]
5. for each TP in the S(PTP) do do
6. if TP not in HashPTP then
7. assign unique id to TP
8. add TP to HashPTP along with its unique id
9. end if
10. end for
11. Sort all TPs in S(PTP) in the descending order of their Milestones
12. for each TP in the S(PTP) do do
13. if (milestone(TP) - sequence Milestone) < *window* then
14. add TP to sequence
15. else
16. write sequence to SeqPTP
17. initialize new sequence, store sequence Milestone
18. add TP to new sequence
19. end if
20. end for
21. for each TP in the S(NTP) do do
22. if TP not in HashNTP then
23. assign unique id to TP
24. add TP to HashNTP along with its unique id
25. end if
26. end for
27. Sort all TPs in S(NTP) in the descending order of their Milestones
28. for each TP in the S(NTP) do do
29. if (milestone(TP) - sequence Milestone) < *window* then
30. add TP to sequence
31. else
32. write sequence to SeqNTP
33. initialize new sequence, store sequence Milestone
34. add TP to new sequence
35. end if
36. end for
37. end while
38. Generate Frequent Sequential Positive Transitional Patterns using SeqPTP as the input database to the Prefix Span Algorithm defined
39. Generate Frequent Sequential Negative Transitional Patterns using SeqNTP as the input database to the Prefix Span Algorithm defined

There are multiple phases in this algorithm. In the first phase the input transactions are sorted on their

sequences and within each sequence on the order of the timestamp of each transaction. These sets of transactions are all processed in a loop where transitional patterns area generated for each set of transactions which includes frequent pattern generation using Apriori Algorithm defined in [WA06], positive transitional patterns generation and computation of their significant frequency ascending milestones and negative transitional patterns generation and computation of their significant frequency descending milestones.

The second phase involves creation of a sequence database using the Positive transitional patterns from the previous phase. During this phase each positive TP is assigned a unique ID. Steps 5 through 17 of the algorithm define this phase. HashPTP is a hash table that contains a mapping between the positive transitional pattern and the unique ID assigned to it. All PTPs are sorted in the descending order of their milestones and sequentially read. PTPs whose milestones fall within a user defined window are grouped together as an itemset as defined in [AS95] and referred by their unique IDs. The final output set is SeqPTP that contains a sequence database of positive transitional patterns.

The third phase is the creation of a sequence database using the Negative transitional patterns from the previous phase. During this phase each negative TP is assigned a unique ID. Steps 18 through 30 of the algorithm define this phase. HashNTP is a hash table that contains a mapping between the negative transitional pattern and the unique ID assigned to it. All NTPs are sorted in the descending order of their milestones and sequentially read. NTPs whose milestones fall within a user defined window are grouped together as an itemset as defined in [AS95] and referred by their unique IDs. The final output set is SeqNTP that contains a sequence database of positive transitional patterns.

In the final phase the prefix span algorithm defined in [WA07] is used to generate Sequential Positive Transitional Patterns in Step 32 from the sequence database produced in steps 5 through 17. Similarly, in Step 33 the prefix span algorithm defined in [WA07] is used to generate Sequential Negative Transitional Patterns from the sequence database produced in steps 18 through 30.

3.5 Database scan and time complexity

As per [WA09] the time complexity of the TP-Mine algorithm is $O(/D/ + nX/T/)$ where $/D/$ is the number of transactions in D, n is the number of frequent patterns in D and $/T/$ is the maximum number of milestones in all of the transitional patterns P(k) in D. As per [H+00] the time complexity of the Prefix Span algorithm is O((N) L) where N is the number items in D and L is the maximum length of all transactions in D in terms of the number of items in a transaction.

The Sequential TP-Mine algorithm combines both TP- Mine and Prefix Span algorithms by repeating the TP cycle for many sequences of transactions and performing a Prefix Span on the resulting database of transitional patterns. In addition it also has the step of creating a sequence database where Transitional Patterns from each sequence are sorted in the descending order of their milestones and sequentially read to group them into sequences. Since Mergesort is used for this step the complexity of this step is of the order $TotTPLog(TotTP)$ where TotTP is the total number of transitional patterns generated in all sequences. Therefore its time complexity can be defined as follows - $O(/Dmax/ + nmaxX/Tmax/) + O(TotTPLog(TotTP)) + O((N)L)$ where Dmax is the maximum number of transactions in all sequences, nmax is the maximum number frequent patterns in all sequences and Tmax is the maximum number of milestones in all sequences and N is the number of transitional patterns discovered in all sequences and L is the maximum length of a transitional pattern in terms of the number of items in it.

4. EXPERIMENTAL SETUP

To evaluate the efficiency of the Sequential TP-Mine Algorithm and prove the utility of sequential transitional patterns, two sets of experiments have been performed, each using a dataset from the real world. These are retail market basket and T10I4D100K dataset [***b].

4.1 Data preparation

Both datasets mentioned did not associate transactions with specific customers, so there was no customer ID in each transaction. To overcome this constraint, the methodology outlined in [AS95] was followed where customer sequences were assumed to have a distribution, and transactions were assigned to a fixed number of customers such that some customers may get many transactions. Some may get very few while the rest would follow the distribution. To test the consistency of our model a user-defined mean number of transactions per customer was used as an input parameter to a distribution method to assign transactions to customers.

4.2 Retail market basket data

The Retail dataset was obtained from the Frequent Itemset Mining Dataset Repository [***b]. Retail Market Basket Data contains 88162 transactions with 16470 unique items in these transactions. These transactions are divided between a fixed number of

customers using a Poisson distribution. For example in one run there were about 3000 customers who were assigned these transactions at an average of 30 transactions. Outliers were introduced in the data with about 5% of customers being assigned 10 times the average number of transactions and another 5% were assigned one tenth the average number of transactions.

The parameters used to run the experiments were pattern support threshold, number of customers assigned to the transactions and the average number of transactions assigned to each customer. Pattern support threshold is kept at 05%, 10% and 15% to generate 3 series of runs that discover sequential transitional patterns. Each of these 3 series has 4 different runs generated by random distributions 2000 customers assigned an average of 45 transactions, 3000 customer assigned an average of 30 transactions, 4000 customers assigned an average of 25 transactions and 5000 customers assigned an average of 20 transactions.

The sequence support threshold was held constant at 20% for all experiments. The result in terms of sequences of transitional patterns was consistent generated by all experiments. Table 6 shows the positive transitional patterns for a few customer Ids along with their significant milestones from the run with 2000 customers for a support threshold of 5%. This is the result of the first phase of the algorithm. Similar transitional patterns were obtained for other runs even though the milestones were different because of the different distribution parameters used. Table 6 shows the sequence database for positive transitional patterns generated by this run.

Table 6. PTP by Sequence for Retail Market Basket Data

Sequence ID	TP Item ids in Sequence
001	<(1 3 30 10)(80)(6 16)>
002	<(1 296)(17 10)(18 3)(6 9 13)(16)>
004	<(3 10)(540 80 15)(16)>
006	<(3 10)(540 80 15) (16)>
008	<(3 10)(540 80 15) (16)>
113	<(10)(3 1)(30 15)(16)>
1829	<(10)(46 12 13)(136)(3)(1)(190)>
1879	<(50 17)(1)(16)(10)>
1997	<(1 2 3)(5)(4)>

Table 7 shows the mapping of each positive transitional pattern to a unique ID assigned to it by the algorithm. This ID is referenced in 6 when building the positive sequence database. Item ID 1 mapped to PTP < 48 > is found in sequences 1,113,1829,1997 while Item ID 16 mapped to < 41 > is found in sequences 4,6,8,113,1829 and Item ID 18 mapped to < 39 41 > is found in sequence 2.

Table 7. Item ids for positive transitional patterns

Item ID	PTP
001	<48>
002	<38 48>
003	<39 48>
004	<39 65>
011	<39 225>
016	<41>
018	<39 41>
031	<39 2238>
103	<48 1872>
211	<41 101>
806	<910 956>

From Table 8 it can be observed that IDs 1, 16 and 18 form a Level 2 Sequence that is frequent because they achieve their SFAM during the same milestone within an error of user-defined window of 3 days. Hence they are a Sequential Transitional Pattern (SPTP).

Table 8. Sequential positive transitional patterns - SPTP for retail market basket dataset

Level	SPTP
001	<1 3>
	<6 9>
	<6 16 18>
	<2>
002	<<1><16 18>>
	<30><16>

4.3 D10I4100K Dataset

The D10I4100K dataset was obtained from the IBM data generator of the Almaden research group. It contains 100000 transactions with 870 unique items in these trans-actions. These transactions are divided using a Poisson distribution similar to that of retail dataset. The parameters used to run experiments were also similar to that of retail dataset.

The parameters used to run the experiments were pattern support threshold, number of customers assigned to the transactions and the average number of transactions assigned to each customer. Pattern support threshold is kept at 05%, 10% and 15% to generate 3 series of runs that discover sequential transitional patterns. Each of these 3 series has 4 different runs generated by different Poisson distributions 1000 customers assigned an average of 100 transactions each, 1500 customer assigned an average of 67 transactions each, 2000 customers assigned an average of 50 transactions each and 2500 customers assigned an average of 40 transactions

each.
The sequence support threshold was held constant at 20% for all experiments. The result in terms of sequences of transitional patterns was consistent generated by all experiments. Table 9 shows the negative transitional patterns for a few customer Ids along with their significant milestones from the run with 1000 customers for a support threshold of 5%. This is the result of the first phase of the algorithm. Similar transitional patterns were obtained for other runs even though the milestones were different because of the different distribution parameters.
Table 9 shows the sequence database of Negative transitional patterns for a few sequences along with their significant milestones. The sequences 605 and 641 have 2 Transitional Patterns with IDs 28 and 34 in them. The actual patterns for these IDs can be found in the Table 10 below.

Table 9. NTP by Sequence for D10I4100K Dataset

Sequence ID	TP Item ids in Sequence
583	<(100 25)(17 1)(152 28 10)(4 60 7)(119)>
605	<(23)(27 4 34 10)(25 20)(113)(85 88)>
641	<(4)(53 106)(81)(25)(62 34 155)(11)>

Table 10. Item IDs for negative transitional patterns in D10I4100K dataset

Item ID	NTP
005	<(368)>
028	<(829)>
029	<(39)>
034	<(529)>

Table 10 shows the mapping of each negative transitional pattern to a unique ID assigned to it by the algorithm. This id is referenced in Table 9 when building the negative TP sequence database. Item ID 28 belongs to the negative transitional pattern (NTP) (829) and Item ID 34 belongs to the NTP (529).

Table 11. Sequential negative transitional patterns - SNTP for D10I4100K dataset

Level	SNTP
002	<(5)(34)>
	<(5)(28)>
	<(28)(5)>
	<(28) (34)>
	<(34) (5)>

The table 11 shows that < (28) (34) > is an interesting sequence which means NTPs < (829) (529) > is an interesting negative sequential transitional pattern where frequent patterns (829) and (529) observe a dip in their support levels at the same milestone within a user-defined error window which in our case is 3 days. Similar inferences can be drawn about the other Level 2 sequential negative transitional patterns in this table.

4.4 Evaluation of scalability

The results of different runs on each of the datasets have been plotted in terms of the running time and graphs generated. These graphs indicate linear scalability for the sequential TP-Mine algorithm for increasing number of sequences (indicated by number of customers) and decreasing support threshold. On the X-axis of these graphs is the average number of customers and the Y-axis shows the runtime of the algorithm in milliseconds. There are 3 lines plotted one each for a frequency support of 5%, 10% and 15% respectively
In the runs for both datasets it was observed that Frequency Support threshold had an inverse relationship with response time. This is because there is an increase in the number of frequent patterns to be found and tested for transitions. Both FP growth and Apriori were used to generate frequent patterns and FP growth was found to be faster so the final results shown are from FP growth.
The graph for 5% support has been separated from the other 2 lines (10% and 15%) because there is a significant increase in the runtime between the 2 sets and this needed to be highlighted.

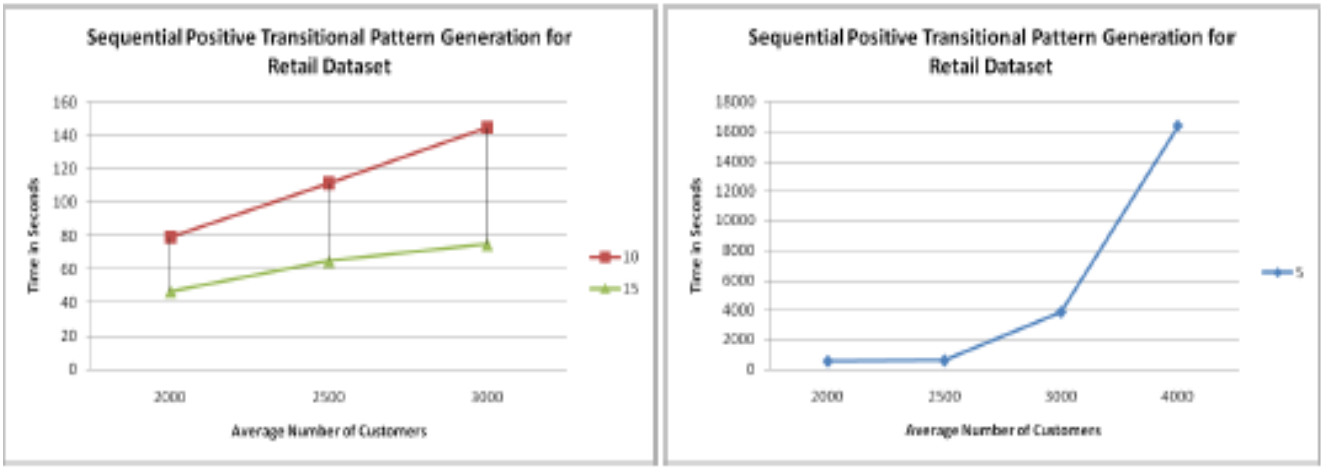

Fig. 1. Figure 1

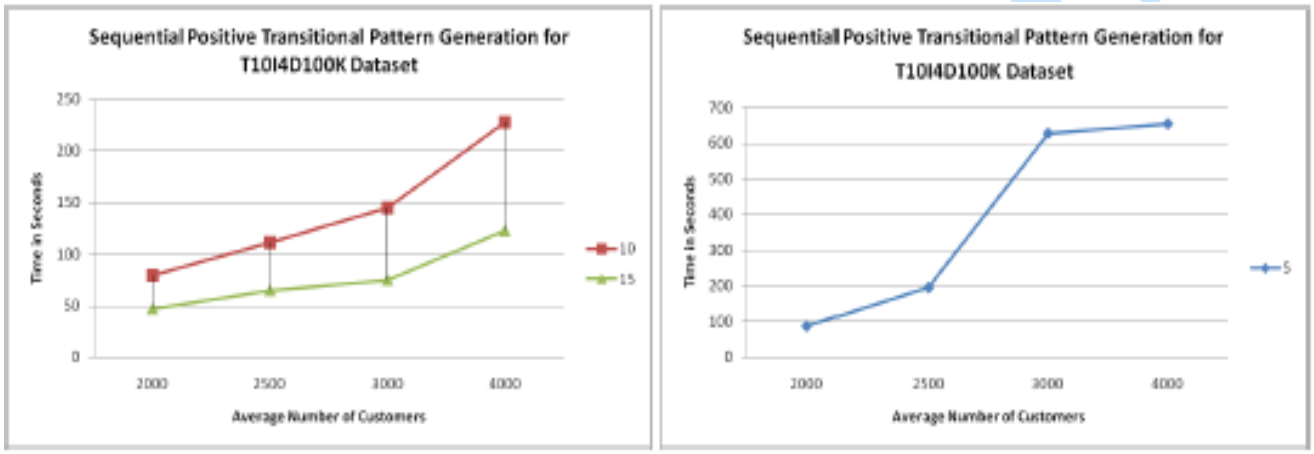

Fig. 2. Figure 2

CONCLUSIONS

One of the goals outlined in [WA09] was about research in the area of sequential transitional patterns. In this paper we undertook the research and defined two kinds of sequential-transitional patterns. SPTP or Sequential Positive transitional patterns are an ordered set of positive transitional patterns that follow their predecessor's SFAM or significant frequency ascending milestone. SNTP or Sequential Negative transitional patterns are an ordered set of negative transitional patterns that follow their predecessor's SFDM or significant frequency descending milestone. To discover these patterns we presented the Sequential TP-Mine algorithm. Our Algorithm generates transitional patterns for each sequence of transactions defined in the database in the first phase. In the next phase it divides the transitional patterns into positive and negative sets. On each of these sets unique Ids are assigned to each transitional pattern and a sequence database is built using these Ids. Finally SPTPs and NPTPs are generated from the positive and negative sequence databases using the PrefixSpan Algorithm. Our experimental results show the algorithm is reasonably scalable. In our experimental study we used two retail datasets to demonstrate the utility of the algorithm in the real world of retailing.

ACKNOWLEDGEMENTS

Some of the classes referred in the code written to conduct experiments published in this paper have been drawn from open source libraries. The authors would like to acknowledge the creators of such libraries for their invaluable contribution. All statistical classes to generate distributions used to assign transactions to customers have been taken from the Colt Project 1.Frequent Pattern generating Apriori and FP-growth algorithms and sequence generating PrefixSpan algorithms have been used from a library of open source data mining algorithms maintained by Philipe Fournier Vigner [***a]

REFERENCES

[AS94] **R. Agrawal, R. Srikant** - *Fast Algorithms for Mining Association Rules*, Proc. 1994 Intl Conf. Very Large Data Bases (VLDB 94), pp.487-499, Sept. 1994.

[AS95] **R. Agrawal, R. Srikant** - *Mining Sequential Patterns*, Proc. 11th Intl Conf. Data Eng., pp.3-14, 1995.

[AS96] **R. Agrawal, R. Srikant** – *Mining Sequential Patterns: Generalizations and Performance Improvements*, Proc. Fifth Intl Conf. Extending Database Technology (EDBT 96), 1996.

[BMR02] **T. Bailey, T. Manoukian, K. Ramamohanarao** - *Fast Algorithms for Mining Emerging Patterns*, Proc. Sixth European Conf. Principles of Data Mining and Knowledge Discovery(PKDD 02), pp. 39-50, 2002.

[B+99] **T. Brijs, G. Swinnen, K. Vanhoof, G. Wets** - *The use of association rules for product assortment decisions: a case study*, in Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, 1999.

[H+00] **J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu** - *FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining*, Proc. 2000 ACM SIGKDD Intl Conf. Knowledge Discovery in Databases (KDD 00), pp. 355-359, Aug. 2000.

[H+04] **J.-W. Han, J. Pei, Y.-W. Yin, R.-Y. Mao** - *Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*, Data Mining and Knowledge Discovery, vol.8, no. 1, pp. 53-87, 2004.

[P+04] **J. Pei, J. Han, B. Mortazavi-Asl, J.-Y. Wang, H. Pinto, Q.-M. Chen, U. Dayal, M.-C. Hsu** - *Mining Sequential Patterns by Pattern-Growth: The Prefixspan Approach*, IEEE Trans.Knowledge and Data Eng., vol. 16, no. 11, pp. 1424-1440, Nov. 2004.

[P+99] **N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal** - *Discovering Frequent Closed Itemsets for Association Rules*, Proc. Seventh Intl Conf. Database Theory (ICDT 99), pp. 398-416, 1999.

[WA06] **Q. Wan, A.-J. An** - *An Efficient Approach to Mining Indirect Associations*, J. Intelligent Information Systems, vol. 27, no. 2, pp. 135-158, 2006.

[WA07] **Q. Wan, A. An** - *Transitional Patterns and Their Significant Milestones*, Proc. Seventh IEEE Intl Conf. Data Mining, 2007.

[WA09] **Qian Wan, Aijun An** - *Discovering Transitional Patterns and their Significant Milestones in Transaction Databases*, IEEE Transactions on Knowledge and Data Engineering, pp. 1692-1707, 2009.

[***a] http://www.philippe-fournier-viger.com/spmf/
is an open-source data mining mining platform written in Java. It is distributed under the GPL v3 license

[***b] http://fimi.ua.ac.be/data/
is an open-source data mining mining repository where real and synthetic datasets are maintained. T10I4D100K is a synthetic dataset generated using the generator from the IBM Almaden Quest research group and available in this repository.