Mining Expressive Temporal Associations From Complex Data

Keith A. Pray[1] and Carolina Ruiz[2]

[1] BAE Systems. Burlington, MA 01803 USA
keith.pray@baesystems.com
[2] Department of Computer Science. Worcester Polytechnic Institute (WPI)
Worcester, MA 01609 USA
ruiz@cs.wpi.edu
http://www.cs.wpi.edu/∼ruiz

Abstract. We introduce an algorithm for mining expressive temporal relationships from complex data. Our algorithm, AprioriSetsAndSequences (ASAS), extends the Apriori algorithm to data sets in which a single data instance may consist of a combination of attribute values that are nominal sequences, time series, sets, and traditional relational values. Data sets of this type occur naturally in many domains including health care, financial analysis, complex system diagnostics, and domains in which multi-sensors are used. AprioriSetsAndSequences identifies predefined events of interest in the sequential data attributes. It then mines for association rules that make explicit all frequent temporal relationships among the occurrences of those events and relationships of those events and other data attributes. Our algorithm inherently handles different levels of time granularity in the same data set. We have implemented AprioriSetsAndSequences within the Weka environment [1] and have applied it to computer performance, stock market, and clinical sleep disorder data. We show that AprioriSetsAndSequences produces rules that express significant temporal relationships that describe patterns of behavior observed in the data set.

1 Introduction

This paper extends the use of association rules [2] to complex temporal relationships essential to many domains. Our association rule mining approach discovers patterns in data sets whose instances consist of any combination of standard, set-valued, and sequential attributes. These data sets occur naturally in several scientific, engineering, and business domains, and are generally richer than the transactional, the relational, and the sequence data sets to which association rule mining has been applied. To date, our mining approach has been applied to computer system performance, stock market analysis and clinical sleep disorder data [3]. Complex system diagnostics, network intrusion detection, and medical monitoring are some related domains to which this work can also be applied.

A main motivation for mining these temporal rules from complex data comes from our previous experience working on the performance analysis of a hardware and software backup/restore product. The time it takes to complete a backup can be of great importance. One of the most labor consuming tasks is to predict this time. A rule based expert system was built as a way to disseminate the performance knowledge used to predict backup time. This expert system quickly grew to include over six hundred rules. The need for automating the discovery of new rules was apparent.

Figure 1 depicts a small sample of the type of complex data set that our mining approach applies to. In this computer performance data set, each instance (row) corresponds to a single test in which a process was run until it completed a task. The attributes describe the conditions under which the test was run and state information collected during the test including standard attributes such as processor speed and physical memory size; set-valued attributes such as which algorithms the process ran; and sequential attributes such as the CPU utilization percentage, memory usage, and the total number of processes running over time. Other such numeric and nominal sequential attributes include CPU and memory usage of all other processes, I/O activity on main storage devices, memory paging, and process state (running, exit normally, exit without output). All time sequence attributes in a single data instance share the same time line.

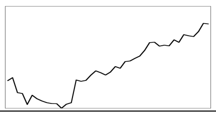
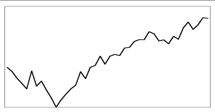
ID	**CPU %**	**CPU (MHz)**	**memory %**	**algorithms**
1		600		{neural net, back-propagation}
2	40, 52, 67, 80, ...	600	10, 26, 46, 69, 86, ...	{C4.5}
3	10, 39, 87, 96, ...	300	19, 50, 82, 80, 70 ...	{naive Bayes}
...	...	...	...	...

Fig. 1. A sample of a data set containing complex instances. Here, sequential values are represented in a graphical manner only for the first row to save space.

Events of interest in the sequential attributes in this data set include among others *increases* and *decreases* in the utilization of a resource (CPU, memory), going above/below a certain usage threshold, and whether or not a process terminates normally. A sample interesting temporal pattern in this domain is: "Java processes running a machine learning algorithm that are not given the -P option, that exhibit an increase in its memory usage, and that during this increase its memory paging also increases tend to, with likelihood 82%, exit prematurely". This temporal pattern is captured by our association rule:

process(java)-mem-usage-increase $[t_0,t_2]$ & process(java)-page-increase $[t_1,t_2]$ & flag=-P-missing $\Rightarrow$ process(java)-exit-without-output $[t_3,t_4]$, conf=82%.

here, t_0, t_1, t_2, t_3, and t_4 are *relative* time indices that are used to express the relative order in which these events are observed.

The algorithm presented here to mine these temporal relationships, *AprioriSetsAndSequences (ASAS)*, takes as input a complex temporal data set as described, a set of predefined event types, and the standard Apriori minimum support and minimum confidence parameters. An *event type* is a template of a subsequence of interest in a time sequence attribute. Our mining algorithm starts by identifying occurrences of these event types in the corresponding sequence attributes as described in Section 2. Then, it generates all the frequent relationships among the temporal occurrences of these events and among these and the other non-temporal attributes. Section 3 describes the algorithm. Since there are 13 ways of sorting just two events in time (*before, after, overlaps,* etc.) [4], and the number of possible orderings of a set of events grows exponentially with the number of events in the set, central issues in the design and implementation of our mining algorithm are the strategy used to prune unnecessary orderings from consideration, and the data structures used to effectively handle potentially frequent orderings of events. We describe these in Section 3. That section also describes our extensions of the basic notions of support and confidence needed to handle multiple occurrences of an event or a pattern in a complex data instance. Section 4 presents an evaluation of our mining algorithm in the stock market domain. Section 5 surveys related work and contrasts our approach to others. Section 6 summarizes the contributions of this paper and discusses future work.

2 Identifying Events in Sequences

Events of interests are available in multiple domains. Examples of those are *head & shoulders reversal* and *ascending triangle* in stock market analysis [5], and *increase in CPU utilization* in the performance domain. An event can be described by a Boolean condition or by a template of the event "shape". Such a template can be for example a 2–dimensional curve.

Given a collection of domain–specific events of interest, we identify occurrences of those events in the sequential attributes. Once the occurrences of an event of interest have been identified in the values of a sequential attribute, they are stored in a new event set attribute. We define an *event set attribute* as an attribute whose values are sets of an event type occurrences. As an example, consider an event attribute for the CPU time sequence attribute. The CPU time sequence attribute is the percentage of CPU usage in the overall computer system. This event attribute specifies when the CPU usage increases. Assume that in a particular data set instance I, increases in CPU usage occur from time 2 to 8, 13 to 19, and 35 to 47. Then, the I' value for the new attribute CPU-Increase is { [2,8], [13,19], [35,47] }. AprioriSetsAndSequences mines temporal associations directly from events. This keeps intact the temporal information represented in the data set while eliminating much of the work involved in scanning the actual sequences during mining. The events are akin to indexes into the sequences.

3 The ASAS Algorithm

Input ASAS takes as input a data set consisting of instances, and minimum support and minimum confidence thresholds. Each instance has a set of attributes. Attributes can be of type nominal, set, and event set. An event set is simply a set whose elements are events.

Handling Set Attributes We use the algorithm to mine association rules from set valued data described in [6], which was implemented within Weka [7].

Handling Event Attributes Since we are not interested in the occurrence of say a CPU-Increase event at absolute time [13, 18], but rather on the fact that this event occurred in a relative temporal position with respect to other events, ASAS uses a relative time line $t_0, t_1, t_2, \ldots$. There are no units implied. Each point on the relative time line corresponds to the begin times or end times of one or more events in the item set. When an event is added to an item set the item set's relative time line must be updated. As an illustration, the real time instance {Disk Increase [5,25], CPU Increase [10,40]} is represented by the relative time item set {Disk Increase [t_0,t_2], CPU Increase [t_1,t_3]}. Adding the event Memory Sustain with real times [2,35] to the item set results in {Disk Increase [t_1,t_3], CPU Increase [t_2,t_5], Memory Sustain [t_0,t_4]}. Simply sorting the real time values and numbering them starting from t_0 yields the relative times.

Level 1 Candidate Generation Our ASAS algorithm creates an item set of size one for each of the *regular* (i.e. non–event) items appearing in the data set, as Apriori does. For event items, it generates representative items. For instance, the representative of Memory Sustain [2,35] is Memory Sustain [t_0,t_1], which represents all the Memory Sustain events regarless of their real time occurrences.

Level 1 Counting Support The support of an item set is the percentage of instances in the data set that contain the item set. The weight of an item set is the number of instances that contain it. For a regular item, we count its support as the Apriori algorithm does. For an event item, say Memory Sustain [t_0,t_1], a data instance contributes to its weight (and hence to its support) if the instance contains an occurence of the item, say Memory Sustain [2,35].

Level 2 Candidate Generation As Apriori, ASAS generates candidate item sets of size two by combining each pair of frequent item sets of size one. However, for each pair of event items there exist thirteen different item sets that represent the possible temporal relationships [4] between the two event items.

Level 2 (and up) Counting Support A data instance contributes to the weight of an item set if it contains all the regular and the event items in the item set. If the item set contains only one event item, checking if the instance contains it is done as described above in Level 1 Counting Support. If the item set contains more than one event item then a mapping must exist from event items in the item set to event items in the instance. This mapping provides a match (or unification) from the relative times in the item set to the real times in the data instance that preserves the relative order of occurrence of the events. For example, a data instance that contains {Disk Increase [5,25], Disk Increase [40,55], CPU Increase [50,60]} counts towards the weight (and hence the support) of the item set {Disk Increase [t_0,t_2], CPU Increase [t_1,t_3]} with mapping $t_0 \rightarrow 40$, $t_1 \rightarrow 50$, $t_2 \rightarrow 55$, $t_3 \rightarrow 60$. Our ASAS algorithm uses a specially designed data structure to make the search for a valid such mapping very efficient.

Level 3 (and up) Candidate Generation During the combination of item sets, regular items are handled as Apriori does. Two item sets containing event items can be combined if there exists a mapping from all the event items in the first item set to all the event items in the second item set. This excludes event items that are the last item in an item set (items in an item set are sorted using an arbitrary but fixed order). As for level two candidate generation, it is possible for more than one candidate item set to be generated for each pair of frequent item sets that are combined. The algorithm for generating these possibilities is more involved than the iteration of thirteen possible temporal relationships. Consider combining the item sets A and B:

A: { Disk Increase [t_0,t_2], CPU Increase [t_1,t_3] }
B: { Disk Increase [t_1,t_2], Memory Sustain [t_0,t_3] }

These item sets can be combined since they have the same number of items, a mapping exists between the Disk Increase event in item set A and the Disk Increase event in item set B, and the event items listed last in A and B are different. The temporal relationship between the CPU Increase event in A and the Memory Sustain event in B is not known. Some of the possible relationships can be eliminated by inferring information from the fact that the Disk Increase event in both A and B is the same event. Combining the two item sets is done by adding the last item from the first item set to the second item set. All possible begin and end time pairs of the Memory Sustain event need to be determined in relation to item set A's existing relative time line. A candidate item set is generated for each pair. In these candidate item sets the relative time line is renumbered starting from t_0.

{Disk Increase [t_1,t_3], CPU Increase [t_2,t_5], Memory Sustain [t_0,t_4]}
{Disk Increase [t_1,t_3], CPU Increase [t_2,t_4], Memory Sustain [t_0,t_4]}
{Disk Increase [t_1,t_3], CPU Increase [t_2,t_4], Memory Sustain [t_0,t_5]}

Algorithm

```
 1: given a data set of instances $DS$, and minimum weight $minW$
 2: **for all** regular items $i$ in $DS$ **do**
 3:    create candidate item set $c$ of size $k = 1$
 4:    add $i$ to $c$ and add $c$ to candidate list $C$
 5: **for all** event items $e$ in data set **do**
 6:    **if** event type of $e$ not present in event type list $ET$ **then**
 7:       create candidate item set $c$ of size $k = 1$
 8:       create a new event item $ei$ with event type of $e$ and begin time = 0 and end
          time = 1
 9:       add $ei$ to $c$, add $c$ to $C$, and add $e$ to $ET$
10: **for all** instances $I$ in $DS$ **do**
11:    **for all** $c$ in $C$ **do**
12:       **if** $I$ contains the item in $c$ **then**
13:          increment weight of $c$
14: **for all** $c$ in $C$ **do**
15:    **if** weight of $c \geq minW$ **then**
16:       add $c$ to frequent item sets of size $k$ list
17: remove all from $C$
```

```
18: while frequent item set of size $k$ list is not empty do
19:   k++
20:   for all pairs of item sets $f1$ and $f2$ in the frequent item sets of size $k$-1 list do
21:     if $f1$ and $f2$ contain event items then
22:       generate 13 candidates, 1 for each possible temporal relationship between
          the event items $f1$ and $f2$ do not have in common
23:     else
24:       generate 1 candidate by combining $f1$ and $f2$
25:     add generated candidate(s) to $C$
26:   for all instances $I$ in $DS$ do
27:     for all $c$ in $C$ do
28:       if all regular items $i$ in $c$ are included in $I$ then
29:         if mapping exists between all event items $ei$ in $c$ to event items in $I$ such
            that all temporal relationships are the same then
30:           increment weight of $c$
31:   for all $c$ in $C$ do
32:     if weight of $c \geq minW$ then
33:       add $c$ to frequent item sets of size $k$ list
34:   remove all from $C$
```

Rule Construction and Confidence Calculation The construction of rules from frequent item sets is similar to that of the Apriori algorithm, with the exception of the confidence calculation. Traditionally confidence is defined for a rule A $\Rightarrow$ B as the percentage of instances that contain A that also contain B. That is, support(AB)/support(A). Consider a data set that has one time sequence: $<a,b,a,a,a,a>$. Since there is one instance in our data set and it contains the item set $\{a[t_0,t_1], b[t_2,t_3]\}$, the support of each of the item sets $\{a[t_0,t_1]\}$, $\{b[t_2,t_3]\}$, and $\{a[t_0,t_1], b[t_2,t_3]\}$ is 100%. If support were used to calculate the confidence of the rule $a[t_0,t_1] \Rightarrow b[t_2,t_3]$, it would be 100%. This implies that 100% of the time a appears, b follows. Looking at the time sequence, only 20% of the time is a followed by b. We define *the confidence of a rule containing event items as the percentage of mappings from the antecent of the rule to the data instances that can be extended to mappings from the full set of items in the rule to the same data instances.* Note that there are 5 possible mappings from the antecent $a[t_0,t_1]$ of the rule to the data instance, but only one of them can be extended to a mapping from $\{a[t_0,t_1], b[t_2,t_3]\}$ to the instance. Hence, the confidence of this rule is 1/5 or 20%.

4 Empirical Evaluation

We have applied our ASAS algorithm to different domains including computer system performance, stock market analysis, and clinical sleep disorder data. Due to space limitations we include here only some results on Stock Market data analysis. The data used [8] consists of ten years worth of closing prices from 7 technology companies from 1992 to 2002 obtained from Yahoo! Finance. Additionally, events such as new product releases, awards received, negative press releases, and expansions or mergers from each company were obtained from each respective company's web site. Each of the 24 instances in this data set represents a single quarter year. All 10 years are not represented because information on the additional events listed above were not available for all years. Before mining, the sequences of closing prices for a quarter for each company are filtered for events. The financial events detected include rounded top, selling climax, ascending triangle, broadening top, descending triangle, double bottom, double top, head & shoulders, inverse head & shoulders, panic reversal, rounded bottom, triple bottom, triple top, sustain, increase, and decrease [5].

Rules. Numerous interesting rules were found by our ASAS algorithm. Due to space limitations we show here just the pair of rules below. They have the same events in them but one has a predictive form (i.e., the events in the consequent occur later in time than the events in the antecedent) and the other has a diagnostic form (i.e., the events in the consequent occur before the events in the antecedent).

CSCO Expand Merge [t_4,t_5] & AMD Ascending Triangle [t_0,t_1]
$\Rightarrow$ SUNW Sustain [t_2,t_3] [Conf: 0.91, Sup: 0.42, Event Weight: 10]

AMD Ascending Triangle [t_0,t_1] & SUNW Sustain [t_2,t_3]
$\Rightarrow$ CSCO Expand Merge [t_4,t_5] [Conf: 1.0, Sup: 0.42, Event Weight: 11]

CSCO Expand Merge 1-7 days, AMD Ascending Triangle 6-30 days, SUNW Sustain 6-13 days

Advanced Micro Devices Inc's closing stock prices exhibits an ascending triangle pattern for 6 to 30 days. Sometime after but during the same quarter Sun Microsystems Inc's closing stock price remains fairly constant for 6 to 13 days. Sometime after in the same quarter Cisco goes through a period of expansion or merger for 1 to 7 days. The predictive form of the rule has a 100% confidence. In any quarter in the data set, every time AMD and Sun exhibit the behaviors described, Cisco expands or merges.

ASAS Performance. Figure 2 shows the seconds used to mine rules per frequent item set found and other metrics for slightly differing data sets from the stock market domain. The total time it takes to mine appears to be insensitive to the number of event attributes, the number of event occurrences, and the average length of the time line. It seems only the number of frequent item sets found in a data set greatly increases mining time. The time spent finding each frequent item set seems related to the number of event occurrences and the number of event attributes in the data set.

Figure 3 shows the results of varying the maximum number of events with the same type that can appear in a rule. This was done with a support setting of 49%. 16 rules containing 2 events of the same type were found. Beyond a maximum of 2 more time is spent per frequent item set with no additional rules found to justify the cost. The lower the percentage of new rules found by increasing the maximum number of events of the same type allowed, the more time per frequent item set will be spent during mining. Figure 4 shows results using a support of 40%. Although more rules are found due to the lower support, more time is spent per frequent item set.

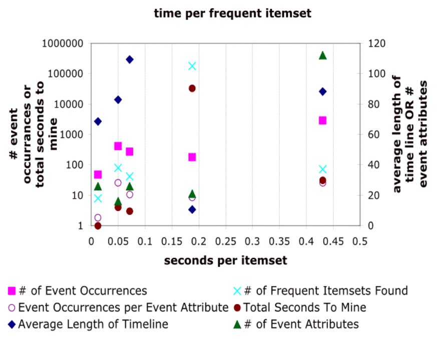


Fig. 2. Various Metrics

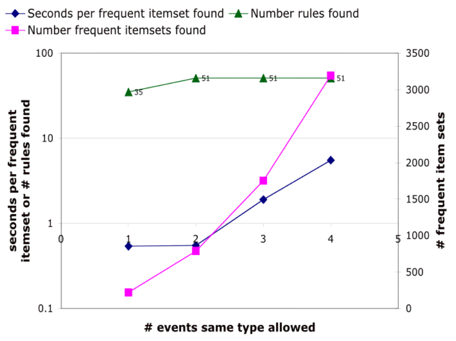


Figure 3: 49 Percent Support

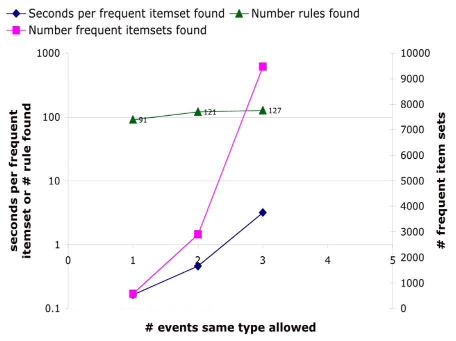


Figure 4: 40 Percent Support

5 Related Work

There has been a great deal of interest in devising approaches to mine associations from sequential data. These approaches can be roughly divided into two groups. The first group contains approaches that extend the Apriori algorithm to sequences. These approaches assume data instances that are sequences of commercial transactions. A commercial transaction is called an *event*. These approaches mine frequent patterns from those data instances. Among others, the work by Srikant and Agrawal [9] and by Zaki [10] and collaborators belong to this group. They use the notions of *time window* and *max/min gaps* to address the complexity of the mining task. Zaki [10] considers item set constraints for this same purpose. One difference between our work and the approaches in this group is that our notion of *event* is a non–trivial time interval and theirs is a point in time (instantaneous events). This has a profound impact on the expressiveness of our association rules and on the complexity of the mining process, as in our case the possible orderings of two single events is 13 while for them that number of orderings in only 3. Another important difference is that in our approach we consider data instances that are combinations of several attribute types, while their instances are sequences of transactions.

The second group of association rule approaches to sequential mining includes the work by Mannila et al. [11–13]. They consider *episodes* of events, where events are again points in time. Episodes are collections of partially ordered events that occur close to each other in time. This constraint addresses the complexity of the search in a way similar to the time window approach described above. Our work extends theirs by allowing events that are time intervals. This enhances the collection of partial orders that are applicable to a set of events and thus the expressiveness of the mined patterns.

Roddick and Spiliopoulou [14] provide an excellent survey of temporal knowledge discovery. Rainsford and Roddick [15] report efforts on extending association rules with temporal information. Their work is similar to ours in that they also consider the 13 possible ways in which two temporal events can be ordered in time. However, the expressiveness of their association rules is very restricted in comparison with ours. Bettini et al. [16] describe an approach to mine temporal associations that allows the user to define a rule template, and their algorithm finds valid instantiations of the rule template in the data set. Our approach is more general than theirs in that the user is not restricted to use just one temporal template for each mining task, as our algorithm considers all possible temporal patterns that are frequent. Also, *we can explore several time–granularities during the same mining task, just by defining an event–based attribute for each relevant time–granularity and letting them "intersect" with other events of interest.* Other approaches that employ user–defined temporal templates are those described by Han and collaborators [17, 18]. Their multidimensional intertransaction association rules are particular cases of our complex temporal association rules.

6 Conclusions and Future Work

We introduce an algorithm for mining expressive temporal relationships from complex data sets in which a single data instance may consist of a combination of attribute values that are nominal sequences, time series, sets, and traditional relational values. Our mining algorithm is close in spirit to the two-stage Apriori algorithm. Our work contributes to the investigation of prune strategies and efficient data structures to effectively handle the added data complexity and the added expressiveness of the temporal patterns. Furthermore, the work described here provides a foundation for future investigation and comparison of alternate measures of item set interestingness and alternate frequent item sets search techniques as those discussed in the association rule mining literature but in the context of complex data.

References

1. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers (2000)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc. of the ACM SIGMOD Conf. on Management of Data, Washington, D.C., ACM (1993) 207–216
3. Laxminarayan, P., Ruiz, C., Alvarez, S., Moonis, M.: Mining associations over human sleep time series. In: Proc. 18th IEEE Intl. Symposium on Computer-Based Medical Systems, Dublin, Ireland, IEEE (2005)
4. Allen, J.: Maintaining knowledge about temporal intervals. Communications of the ACM **26** (1983)
5. Little, J., Rhodes, L.: Understanding Wall Street. 3rd edn. Liberty Hall Press and McGraw-Hill Trade (1991)
6. Shoemaker, C., Ruiz, C.: Association rule mining algorithms for set-valued data. In Liu, J., Cheung, Y., Yin, H., eds.: Proc. of the Fourth Intl. Conf. on Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science. Vol. 2690. Springer-Verlag (2003) 669–676
7. Stoecker-Sylvia, Z.: Merging the association rule mining modules of the Weka and ARMiner data mining systems. Undergraduate Thesis. WPI (2002)
8. Holmes, S., Leung, C.: Exploring temporal associations in the stock market. Undergraduate Thesis. WPI (2003)
9. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Intl. Conf. on Database Engineering, IEEE (1995) 3–14
10. Zaki, M.: Sequence mining in categorical domains: Incorporating constraints. In: Proc. Intl. Conf. on Information and Knowl. Management (CIKM). (2000) 422–429
11. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovering Frequent Episodes in Sequences. In Fayyad, U.M., Uthurusamy, R., eds.: Proc. of the First Intl. Conf. on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada (1995)
12. Mannila, H., Toivonen, H.: Discovering generalized episodes using minimal occurrences. In: Proc. of the Second Intl. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, Oregon, AAAI Press (1996) 146–151
13. Das, G., Lin, K.I., Mannila, H., Renganathan, G., Smyth, P.: Rule discovery from time series. In: Proc. of the 4th Intl. Conf. on Knowledge Discovery and Data Mining, ACM (1998) 16–22
14. Roddick, J., Spiliopoulou, M.: A survey of temporal knowledge discovery paradigms and methods. IEEE Trans. on Knowledge and Data Engineering **14** (2002) 750–767
15. Rainsford, C., Roddick, J.: Adding temporal semantics to association rules. In: Proc. of the Third European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'99). (1999) 504–509
16. Bettini, C., Sean Wang, X., Jajodia, S.: Testing complex temporal relationships involving multiple granularities and its application to data mining. In: Proc. of the Fifteenth ACM Symposium on Principles of Database Systems. (1996) 68–78
17. Tung, A.K., Lu, H., Han, J., Feng, L.: Efficient mining of intertransaction association rules. IEEE Trans. on Knowledge and Data Engineering (2003) 43–56
18. Lu, H., Feng, L., Han, J.: Beyond intratransaction association analysis: mining multidimensional intertransaction association rules. ACM Trans. on Information Systems (TOIS) **18** (2000) 423–454