EuroVis Workshop on Visual Analytics (2013)
M. Pohl and H. Schumann (Editors)

Mind the Time: Unleashing the Temporal Aspects in Pattern Discovery

T. Lammarsch,[1] W. Aigner,[1] A. Bertone,[2] S. Miksch,[1] & A. Rind[1]

[1]Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria
[2]Institute of Cartography, Dresden University of Technology, Germany

Abstract

Temporal Data Mining is a core concept of Knowledge Discovery in Databases handling time-oriented data. State-of-the-art methods are capable of preserving the temporal order of events as well as the information in between. The temporal nature of the events themselves, however, can likely be misinterpreted by current algorithms. We present a new definition of the temporal aspects of events and extend related work for pattern finding not only by making use of intervals between events but also by utilizing temporal relations like meets, starts, or during. The result is a new algorithm for Temporal Data Mining that preserves and mines additional time-oriented information.

Categories and Subject Descriptors (according to ACM CCS): H.2.8 [Information Systems]: Database Applications—Data Mining

1. Introduction

Data Mining is a central part of Knowledge Discovery in Databases (KDD). A very important data type is time, and one of the most successful approaches for Temporal Data Mining is the search for temporal patterns. Methods for temporal pattern finding include clustering, classification, and association rules. Their main goal is disclosing local structures of interest [LS06]. State-of-the-art methods are capable of preserving the temporal order of events as well as the information in between [BLT*10b]. Pattern finding usually requires complex parametrization and it results in large amounts of patterns that need to be explored. Visual Analytics (VA) can support this by intertwining the algorithm with interactive visual interfaces [BLT*10a]. Since earlier work in pattern finding [AIS93], the temporal aspect of patterns has been increasingly focused on in research [BLT*10b]. A weakness of current methods is that they mostly ignore temporal relations among events. Even the most advanced ones focus on time intervals between events only, while the time aspects of the events themselves are taken directly from the source data. For example, if a fastfood business in a shopping street is open the whole week while the shops are closed during the weekends, turnover might be significantly lower on Saturday and Sunday (Table 1). If the turnover dataset has one value for each day, then each event will also have a length of one day. Based on events of day-length, algorithms

Table 1: *A dataset of turnover classification according to previous approaches (like MuTIny [BLT*10b]) compared to our novel approach (Section 3).*

Day	Turnover	Previous Appr.	Our Approach
Monday	40	e_{high}	e_{high}
Tuesday	43	e_{high}	
Wednesday	39	e_{high}	
Thursday	41	e_{high}	
Friday	45	e_{high}	
Saturday	18	e_{low}	e_{low}
Sunday	24	e_{low}	

tend to find patterns like the business week: For five consequent days, there is high turnover, and for two consequent days there is low turnover. These sequences alternate. Such patterns are interesting, but by talking to domain experts, we found out that they are already well-known. As a result, such patterns clutter the results when searching for new and unknown patterns: They make the list of results so big that it is hard to compute and even harder to visualize. The basic idea for our approach is to reduce the number of events in a way that makes patterns much simpler while retaining a reasonable amount of information. Over the data value dimensions, methods for simplification help to keep the complexity low [KCPM01b, KCPM01a, LKLC03, LKWL07]. According to the temporal aspects, these methods can only simplify time by rasterization, which requires a fine raster size to prevent information occlusion and does not take into account the structure of time and temporal relations. Aggregating according to domain knowledge is better, but if important (albeit known) cycles overlap, the usability of aggregation declines further, so we aim for a different approach. Our main contribution is a new definition of events (Section 3). We also show a new definition of patterns that not only considers intervals between events, but also other relations (Section 3). The result is a new algorithm for Temporal Data Mining that preserves and mines additional time-oriented information which we show in the context of VA (Section 4).

2. Related Work

Dealing with data locally is a newer method than developing a global model (overview [LS06], first publication [Yul27]). Most work in finding patterns goes back to Agrawal et al. [AIS93] who only consider patterns of events happening together in a set. They also consider time, but only as a method of separating different pattern candidates. However, they already perform planning towards further steps with more complex patterns that can overstretch time steps [SA96]. Agrawal, Srikant, and others, also introduced the concept of "support". The support of a pattern is the frequency of its appearance. and is the main method of determining which patterns are important for many of the algorithms we present below. Support is a good method to find important patterns, but in conjunction with temporal patterns that result from social events, it results in finding a dominance of patterns that are important yet well-known (compare Section 1). Mannila et al. [MTIV97] focus further on sequences of events at different points in time. They also explicitly consider the time step at which one event occurs. Magnusson [Mag00] is among the first to consider the time intervals between events, which is an important step to increase the consideration of time. His T-patterns are tree-shaped and, therefore, differ from the patterns in most other publications that are sequences. Chen et al. [CCK03] introduce the I-Apriori algorithm which extends the Apriori algorithm for pattern finding [LS06] by the consideration of intervals between events. Hu et al. [HHYC09] provide a similar approach where the focus is on patterns with events that do not need to be consecutive, as long as the time intervals are kept. Bertone et al. [BLT*10a, BLT*10b] provide a similar approach, called MuTIny, with user configurable time intervals that can have variable length and consider calendar aspects. They also deal with interactive visualizations that allow users to engage into the process. They advance the concept of pure support by allowing user manipulation. Still, challenging patterns, like mentioned in Section 1, appear and have to be removed manually. In this approach, events are defined by users. They can map value ranges among all variables in the dataset to event classes. We adopt this approach for our algorithm described in Section 3. Furthermore, users have to define intervals. These intervals can have any length, but users can apply different calendar granularities, like months or days, in their definition. After defining the events and intervals, an extended I-Apriori algorithm is applied in multiple iterations. In each iteration, each existing pattern (starting with patterns of length zero that are identical to events) is combined with each interval and each possible followup event that falls into that interval. Therefore, with each iteration, the patterns grow by one interval and one event. Between those iterations, the less frequent patterns are filtered out. This is done by users providing a threshold for the support.

3. Our Novel Temporal Pattern Finding Approach

Our work is primarily related to the MuTIny algorithm [BLT*10b]. However, we introduce a different definition of events and also extend the intervals to event relations.

Defining Events Our event definition allows events of any length and is applicable for rastered data (equal time steps with one data element for each one) as well as unrastered data, which can have any temporal aspect in each data element. We also allow temporal conditions for event definition. There are many different ways to define events. We have abstracted a definition from the related work above which can also be seen as a simplified variant of the philosophical approach by Kim [Kim76]:

Definition 1 An event e is an interval $[t_{begin}, t_{end}]$ during which given conditions $\{\chi_0, \ldots, \chi_n\}$ are fulfilled.

The conditions that are fulfilled can also be temporal, e.g., an interval is during another interval, e.g., a certain day of week, or an interval has a certain length. Our goal is a computability, so we have to further define these conditions. Good definitions for comparing and finding certain values are given in the task framework by Andrienko and Andrienko [AA06]. For elementary tasks that consider a finite number of data elements, they define lookup tasks, comparison tasks, and relation-seeking tasks. Without loss of generality for those definitions, in our algorithm we always use time as reference domain and everything else as data values. We also assume that time is being discretized in collecting the data and that the data elements are sorted by the infimum (the first discrete instant) of each reference. Based on these definitions and assumptions, we can define a condition:

Definition 2 A condition χ_i is fulfilled for a data element d if it is a possible result for an elementary task.

As elementary tasks may look for the reference (time), it is possible to define conditions based on time as well as on data values. According to the definition above we can modify our event definition and define event classes:

Definition 3 An event e is a set of contiguous data elements $\{d_0, \ldots, d_m\}$ sorted by the time references t_d that all fulfill the same conditions $\{\chi_0, \ldots, \chi_n\}$.

Definition 4 An event class E is a collection of events $\{e_0, \ldots, e_h\}$ that fulfill the same conditions $\{\chi_0, \ldots, \chi_n\}$.

Event Relations and Patterns To combine events to patterns, the I-Apriori algorithm searches for possible followup events for existing patterns that fall into given time intervals. We extend this by replacing the intervals with temporal relations, based on the ones presented by Allen et al. [All83]: *before*, *equal*, *meets*, *overlaps*, *during*, *starts*, *finishes*. Intervals between events are an indeterminate variant of *before*.

Definition 5 A pattern p^k of length k is a tuple $\{p^{k-1}, r, e\}$, where p^{k-1} is a pattern of length $k-1$, r is a time relation, and e is an event.

Definition 6 A pattern p^0 of length 0 is an event e.

Definition 7 A pattern class P^k of length k is a collection of patterns $\{p_0^k, \ldots, p_j^k\}$ that consist of events of the same class in the same order and the same relations in the same order.

The Algorithm The algorithm consists of two steps. First, the events have to be found. Conditions that, e.g., limit the minimum length of an event can lead to the situation that a number of data elements does not fulfill the conditions, but with another data element, they do. Therefore, we need to make two subsidiary definitions:

Definition 8 Loose conditions are a set of conditions where one or more conditions have been removed.

Definition 9 An event candidate is a set of data elements sorted by the time reference that fulfill loose conditions. It has to be saved together with those loose conditions as well as that loose conditions' source conditions.

The event finding is explained in Algorithm 1. The resulting events are also patterns of length 0. For longer patterns, the pattern finding in Algorithm 2 can be performed iteratively k times to find patterns of length k. We have already implemented them based on a time-oriented Prefuse [HCL05] extension, only the visual interface is still in development.

4. Applicability in Visual Analytics

Our approach, like MuTIny [BLT*10b] and other pattern finding methods, relies on a number of parametrization issues: (1) Choosing and parametrizing a method for classifying data elements, which might be multi-variate, into event classes, i.e. categoric data. (2) Parametrizing the conditions for events forming a pattern. (3) Choosing the most important patterns between iterations in order to perform the next iteration with a filtered set. These choices require domain experts and have to be made for each analysis individually. To gain insight from the results, sensible visualizations for exploring the results are needed. This can improve the parametrization choices of domain experts in an iterative analysis process. Bertone et al. propose VA as a solution [BLT*10a]. For parametrizing event classes, we already have designed an interactive visual interface [LAB*13]. It needs to be combined with interface elements

Algorithm 1: Event Finding: A number of conditions has to be provided for each event class the user wants to define. Those are subsets of a collective set of conditions.

```
input : dataSet, conditionsSet
output: events
eventCandidates, closedEventCandidates, looseConditionsSet,
  events ← empty key/value-tables;
foreach conditions in conditionsSet do
    looseConditions ← conditions \ conditions that would
    need events to end later or be longer than they actually are;
    looseConditionsSet: add (looseConditions/conditions);
end
foreach dataElement in dataSet do
    foreach (eventCandidate/looseConditions) in
    eventCandidates do
        combined ← eventCandidate ⋂ dataElement;
        if combined fulfills looseConditions then
            eventCandidate ← combined;
        else
            closedEventCandidates: add
              (eventCandidate/looseConditions);
            eventCandidates: remove
              (eventCandidate/looseConditions);
        end
    end
    foreach (looseConditions/conditions) in
    looseConditionsSet do
        if dataElement fulfills looseConditions then
            eventCandidates: add
              (dataElement/looseConditions);
        end
    end
end

foreach (eventCandidate/looseConditions) in
closedEventCandidates do
    if eventCandidate fulfills conditions of looseConditions
    from looseConditionsSet then
        events: add (eventCandidate/conditions);
    end
end
```

Algorithm 2: Pattern Finding: Source patterns of length $k-1$ are transformed to patterns of length k (each consisting of $k+1$ events and k relations).

```
input : sourcePatterns, events, relations, threshold
output: patterns
filteredPatterns ← sourcePatterns \ patterns from classes that
are less frequent than threshold;
foreach filteredPattern in filteredPatterns do
    foreach relation in relations do
        foreach event in events do
            if relation (filteredPattern,event) is fulfilled then
                patterns: add (filteredPattern,relation,event);
            end
        end
    end
end
```

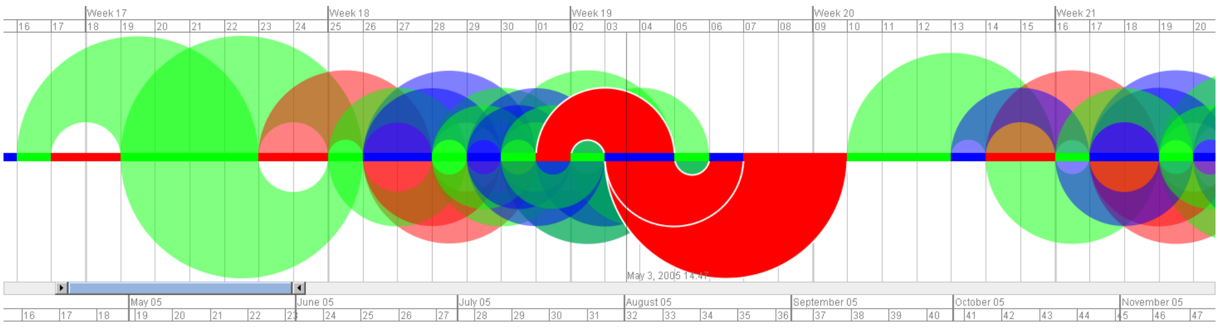

Figure 1: *An ArcDiagram [Wat02] example implementation showing resulting patterns from our algorithm. The patterns visualized here are based on non-overlapping events that fall into one of three classes (red, green, and blue). The opaque bars in the middle show the temporal sequence of events. Patterns are of length 2 and represented by transparent arcs that are colored according to their first event. Upper arcs represent the first two events and their temporal relation (in this case, the first event must be one or two days before the second one), lower arcs represent the last two events and their temporal relation. If the mouse hovers an arc, all arcs belonging to related patterns become opaque with a white border (like seen for the red ones).*

for parametrizing relations. We have also started developing interactive visual interfaces for exploring results of the pattern finding algorithm. Figure 1 shows an ArcDiagram [Wat02] implementation using the free Dodgers Loop Sensor dataset [Hut06]. By directly interacting with events and patterns, we also intend to help users to change the original parameters later in the process, recalculating the results.

5. Preliminary Results

We used two datasets to compare our algorithm to MuTIny [BLT*10b]: (1) A synthetic dataset with 91 random elements; (2) Six months of the Dodgers dataset [Hut06] aggregated into 175 data elements of daily mean values. For both datasets and both approaches, we defined three event classes on traffic value ranges and intervals of one or two days as two different relations. The first observations are: Our approach results in 30% less patterns of length 2 for dataset 1 and 75% less patterns of length 2 for dataset 2. We found out that in our approach, several data elements are part of single events (and therefore patterns). It seems that a comparable amount of data elements is involved in both approaches. The stronger decrease in pattern occurrences for the real-world dataset could be caused by actual information in the day granularity that might be interesting: Certain conditions (i.e., traffic value ranges) exist over certain intervals and form single events. For MuTIny, several pattern variants stemming from those, altogether, describe the same information. The results from our approach have several advantages: Less patterns have to be stored and computed in further analysis steps, improving memory consumption and speed while simplifying visualization and understanding by users. The potential information in the day granularity are expressed on the event level by our approach. This happens at a complexity no worse than linear, while MuTIny (run on a day granularity) requires patterns of the length of the number of involved days, leading to a higher complexity that has to be reduced by heavy threshold filtering. Since our implementation also saves patterns as a forest, it saves further memory compared to the original MuTIny implementation. While we cannot give an absolute statement about the scalability yet, it scales much better than its closest related work.

6. Conclusion and Future Work

First, we presented a novel pattern finding approach for time-oriented data that follows and improves the I-Apriori [CCK03] and MuTIny [BLT*10b] approaches. Second, we described and implemented an algorithm that finds patterns according to our approach. Third, we discussed interactive, visual, iterative parametrization and exploration of the results using VA. Forth, we presented a visualization and preliminary results. Our definition of events results in rather different constructs from other definitions that base events on rastered datasets and do not allow for variable event lengths. Considering our goal to simplify temporal structures, this is intended. In Table 1, for example, our approach does model business weeks and weekends (which is no new information), but instead of using four classes of two-event-patterns it needs only two which contain the same information. Currently, we argue that our approach depicts the same information in less patterns by real-world examples and several control samples from our tests. To provide more solid evidence, the first step of our future work will be to compare the mapping of data elements to events and patterns between state-of-the-art approaches and our approach with many different datasets and tasks. For this, we intend to develop automated as well as visual methods. We also intend to develop interactive visual interfaces for all parametrization issues mentioned in Section 4 as well as exploration of results and intermediate results. All those interfaces are to be combined in a tool allowing for iterative and interactive analysis by domain experts which will then be tested in user studies.

This work was supported by the Austrian Science Fund (#P22883).

References

[AA06] ANDRIENKO N., ANDRIENKO G.: *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Verlag, 2006. 2

[AIS93] AGRAWAL R., IMIELIŃSKI T., SWAMI A.: Mining association rules between sets of items in large databases. *ACM SIGMOD Record 22*, 2 (1993), 207–216. doi:10.1145/170036.170072. 1, 2

[All83] ALLEN J.: Maintaining knowledge about temporal intervals. *Communications of the ACM 26*, 11 (1983), 832–843. doi:10.1145/182.358434. 3

[BLT*10a] BERTONE A., LAMMARSCH T., TURIC T., AIGNER W., MIKSCH S.: Does Jason Bourne need Visual Analytics to catch the Jackal? In *Proceedings of the First International Symposium on Visual Analytics Science and Technology held in Europe, EuroVAST* (2010), Kohlhammer J., Keim D., (Eds.), Eurographics, pp. 61–67. doi:10.2312/PE/EuroVAST/EuroVAST10/061-067. 1, 2, 3

[BLT*10b] BERTONE A., LAMMARSCH T., TURIC T., AIGNER W., MIKSCH S., GAERTNER J.: MuTIny: a multi-time interval pattern discovery approach to preserve the temporal information in between. In *Proceedings of the IADIS European Conference on Data Mining, ECDM* (2010), pp. 101–106. 1, 2, 3, 4

[CCK03] CHEN Y., CHIANG M., KO M.: Discovering time-interval sequential patterns in sequence databases. *Expert Systems with Applications 25*, 3 (2003), 343–354. doi:10.1016/S0957-4174(03)00075-7. 2, 4

[HCL05] HEER J., CARD S., LANDAY J.: Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), ACM, pp. 421–430. doi:10.1145/1054972.1055031. 3

[HHYC09] HU Y., HUANG T., YANG H., CHEN Y.: On mining multi-time-interval sequential patterns. *Data & Knowledge Engineering 68*, 10 (2009), 1112–1127. doi:10.1016/j.datak.2009.05.003. 2

[Hut06] HUTCHINS J.: Dodgers loop sensor data set. UCI Machine Learning Repository, 2006. Accessed on March 2nd, 2013. URL: http://archive.ics.uci.edu/ml/datasets/Dodgers+Loop+Sensor. 4

[KCPM01a] KEOGH E., CHAKRABARTI K., PAZZANI M., MEHROTRA S.: Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems 3*, 3 (2001), 263–286. doi:10.1007/PL00011669. 1

[KCPM01b] KEOGH E., CHAKRABARTI K., PAZZANI M., MEHROTRA S.: Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record 30*, 2 (2001), 151–162. doi:10.1145/376284.375680. 1

[Kim76] KIM J.: Events as property exemplifications. In *Action Theory, Proceedings of the Winnipeg Conference on Human Action* (1976), Brand M., Walton D., (Eds.), D. Reidel Publishing, pp. 159–177. 2

[LAB*13] LAMMARSCH T., AIGNER W., BERTONE A., BÖGL M., GSCHWANDTNER T., MIKSCH S., RIND A.: Interactive visual transformation for symbolic representation of time-oriented data. In *Proceedings of the International Conference on Human Factors in Computing & Informatics, SouthCHI* (2013), Holzinger A., Ziefle M., Glavinic V., (Eds.). forthcoming. 3

[LKLC03] LIN J., KEOGH E., LONARDI S., CHIU B.: A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (2003), pp. 2–11. doi:10.1145/882082.882086. 1

[LKWL07] LIN J., KEOGH E., WEI L., LONARDI S.: Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery 15*, 2 (2007), 107–144. doi:10.1007/s10618-007-0064-z. 1

[LS06] LAXMAN S., SASTRY P.: A survey of temporal data mining. *Sadhana 31*, 2 (2006), 173–198. doi:10.1007/BF02719780. 1, 2

[Mag00] MAGNUSSON M.: Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods 32*, 1 (2000), 93–110. doi:10.3758/BF03200792. 2

[MTIV97] MANNILA H., TOIVONEN H., INKERI VERKAMO A.: Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery 1*, 3 (1997), 259–289. doi:10.1023/A:1009748302351. 2

[SA96] SRIKANT R., AGRAWAL R.: Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology – Proceedings of the 5th International Conference on Extending Database Technology, EDBT* (1996), Apers P., Bouzeghoub M., Gardarin G., (Eds.), LNCS 1057, Springer, pp. 1–17. doi:10.1007/BFb0014140. 2

[Wat02] WATTENBERG M.: Arc diagrams: Visualizing structure in strings. In *Proceedings of the IEEE Symposium on Information Visualization, InfoVis* (2002), pp. 110–116. doi:10.1109/INFVIS.2002.1173155. 4

[Yul27] YULE G.: On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 226* (1927), 267–298. 2