TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

Ph.D. Thesis

A Matter of Time: Multi-time Interval Pattern Discovery to Preserve the Temporal Information in between

Conducted for the purpose of receiving the academic title
'Doktor der technischen Wissenschaften'

Supervisor
Silvia Miksch
Institute of Software Technology & Interactive Systems (E188)

Submitted at Vienna University of Technology
Faculty of Informatics

by

ALESSIO BERTONE
0627423
Floßgasse 11
A-1020 Wien

Vienna, November 18, 2009

_____
(Signature Author)

_____
(First Supervisor)

_____
(Second Supervisor)

Abstract

Data Mining is concerned with the tasks of finding trends, patterns and relationships among patterns and many approaches have been proposed in this area. In particular, these tasks become extremely relevant when dealing with time-oriented data and information. However, especially in the field of so called sequence mining, most of the methods have a sequence of events as outcome, having neither any knowledge about the intervals between them nor about after how much time a particular pattern will reoccur. The goal of this thesis is to investigate how Temporal Data Mining can be adapted and used to analyze multivariate time-oriented data having multiple granularities in order to discover multi-time interval patterns, relations among data (previously unknown) as well as visually support this process. In detail, the approach we propose extends in particular the so called I-Apriori algorithm to non transactional databases and tries to provide a more general as well as more customizable way to find *multi-time interval patterns* in time-oriented data. The proposed approach has been applied to data coming from a shopping mall and data coming from a flight company. The results outlined the presence of interesting behaviour in the data previously unknown and were used as basis for further analysis.

Zusammenfassung

Data Mining beschäftigt sich damit, Trends, Muster und Beziehungen zwischen Mustern zu finden und es gibt eine Reihe von bestehenden Methoden in diesem Bereich. Weil Zeitreihen und zeitorientiere Daten ein besonders häufig vorkommendes und wichtiges Gebiet sind, sind Methoden von besonderer Relevanz, die sich mit zeitorientierten Daten und Informationen beschäftigen. Die im Bereich des Temporal Data Mining bekannten Methoden des sogenannten Sequence Mining liefern Abfolgen von Events als Ergebnisse, beinhalten aber weder Informationen über die zeitlichen Abstände zwischen den Events, noch darüber, in welchen Zeitabständen ein bestimmtes Muster wiederkehrt.

Das Ziel dieser Dissertation ist es, zu untersuchen, wie Temporal Data Mining Methoden für den Einsatz in der Analyse multivariater zeitorientierter Daten mit mehreren Granularitäten erweitert werden können. Dabei sollen einerseits (bisher unbekannte) Beziehungen zwischen den Daten in Form von multi-time interval patterns gefunden werden und andererseits soll dieser Prozess auch visuell unterstützt werden.

Methodisch basiert der vorgeschlagene Ansatz auf dem sogenannten I- Apriori Algorithmus, der in mehrerlei Hinsicht erweitert und weiterentwickelt wurde. Neben der Verallgemeinerung auf andere Datenstrukturen als Transaktionsdatenbanken wurde dabei eine Flexibilisierung und bessere Konfiguration bzw. Steuerbarkeit an verschiedenen Stellen des Mining Prozesses erreicht.
Um die Brauchbarkeit der entwickelten Methode zu illustrieren, wurde sie auf die zwei Anwendungszenarien angewandt: 1) Umsatz- und Personaldaten eines Einkaufszentrums sowie 2) Passagierdaten einer Fluggesellschaft. Mit Hilfe der Mining Methode konnten bisher unbekannte Muster in den Daten gefundenen und als Grundlage für weitere Analysen eingesetzt werden.

Contents

1 Introduction

1.1 Motivation

In the last decade, we took part in a tremendous expansion in our capabilities to both generate and collect data. Advances in scientific data collection, for example from remote sensors, or from space satellites have generated huge amounts of data. Advances in data storage technology such as faster, higher capacity and cheaper storage devices, better database management systems and data warehousing technology have allowed us to transform this data into "mountains" of stored data. Such volumes of data clearly overwhelm the traditional manual methods of data analysis such as spreadsheets and ad-hoc queries. These methods can create informative reports, but they cannot analyse the contents of those reports to focus on important knowledge. New methods and tools can intelligently and automatically transform data into information and furthermore, synthesize knowledge are the subject of the process of Knowledge Discovery in Database (KDD) [1]

Various terms have been used to refer to the notion of finding useful patterns in raw data. These include knowledge/data/information discovery and knowledge/data/information extraction. Some of them define Data Mining as the process of extracting previously unknown information while Knowledge Discovery is defined as the process of making sense out of the extracted information, other do not differentiate between Data Mining and Knowledge Discovery [2].

We adopted the view proposed by Fayyad et al.[1] and Frawley et al. [3], which describes KDD as the overall process of discovering useful knowledge from data while Data Mining is referred to as a step of this process, whose aim is to extract implicit, previously unknown and potentially useful information from databases.

There are many possible application fields, as for example finance, decision support, medicine. Moreover, because of its relevance, many approaches have been proposed to find patterns occurring frequently in a database. Nevertheless, when dealing with time-oriented data and information, exploring trends, patterns, and relationships among patterns are particularly complicated tasks, since in contrast to other quantitative data dimensions that are usually "flat", time has an inherent semantic structure which increases its complexity dramatically [4]. The hierarchical structure of granularities in time, as for example minutes, hours, days, weeks, months, is unlike most other quantitative dimensions. As a matter of fact, time encompasses different forms of divisions (e.g., 60 seconds resemble one minute, 12 months resemble one year, months themselves can have 28, 29, 30 or 31 days) and granularities are combined to form calendar systems (e.g., Gregorian, Julian, Business, or Academic calendars). Moreover, time contains natural cycles and re-occurrences, as for example seasons, but also social–somehow irregular–cycles, like holidays or school breaks. Therefore, time-oriented data need to be treated differently from other kinds of data and demand appropriate methods to analyze them.

Temporal Data Mining incorporates time in the Data Mining process, providing the ability to detect activities rather than just states and its methods focus on time-oriented data. However, except for exceptions like [5],[6],[90], [91],[92],[94], a sequence of events in a certain order is discovered, but neither with any knowledge about the intervals between successive events, nor about when this sequence will reoccur. This situation is clearly depicted in the following example (Figure 1). A customer decides to buy a Play Station 3 (PS3) gaming console (event A). The next day, he buys a racing game (event B). After five days, he buys a steering wheel controller (event C). Another racing game is bought after 3 months (event D). If we take into account only the sequence or the order of these events, ABCD, we cannot know after how much time the next item will be purchased. Moreover, we cannot even know after how much time a similar sequence will occur. On the contrary, if also the time intervals

are considered, we can not only profile the users according to their interests, habits and requirements, but we can also improve the selling strategies according to the timing of their shopping habits. As a matter of fact, the shopping mall can vary its offers and catalogues according to the users. For instance, it is possible to send e-mails or letters describing discounts on games for PS3 two months after the first purchase, or make special offers dedicated to those who bought a PS3 in the previous two/three months.

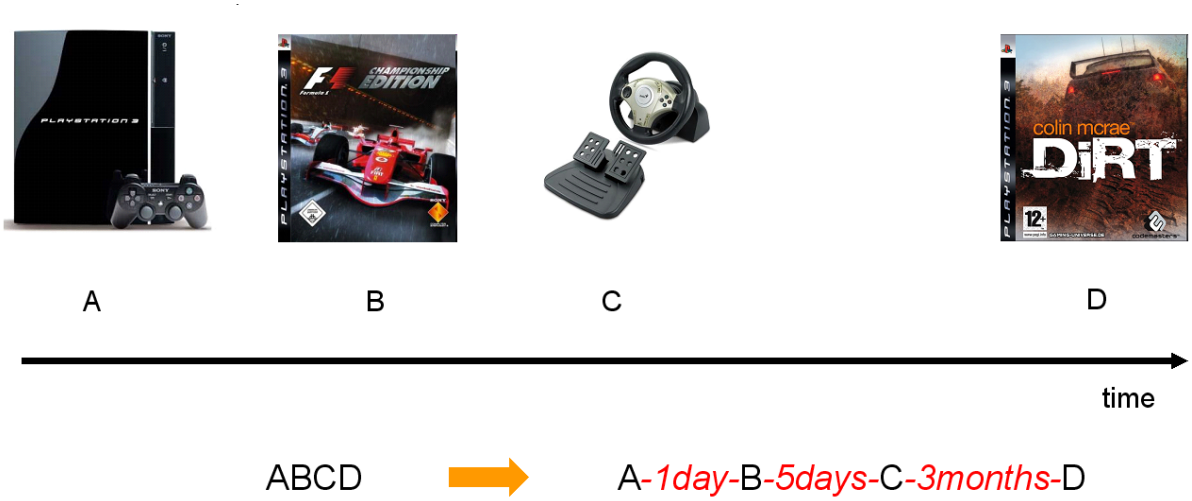

Figure 1: Different approaches to treat sequences of events. Left below: Traditional sequence pattern (order only); Right Below: Multi-time interval pattern (intervals between events).

Hence the exploitation of what can be called *multi-time interval patterns*, that is, pattern whose composing parts are divided by different time intervals, may lead to successful applications not only in retailing, but also in many other fields.

The goal of this thesis is to explore how Temporal Data Mining can be adapted and used to analyze multivariate time-oriented data having multiple granularities in order to discover multi-time interval patterns, and relations among data (previously unknown) as well as visually support this process.

Moreover, a new approach which is able to find these multi-time interval patterns is described, which focuses on the relevance of preserving temporal information between events and patterns, as this aspect is usually neglected or ignored

1.2 Research Questions

Main Questions

How can Temporal Data Mining be adapted and used to analyze multivariate time-oriented data having multiple granularities in order to discover multi-time interval patterns, relations among data (previously unknown) and visually support this process?

Sub-Questions

- How to deal with different time intervals between elements of interest in the data?

- How to deal with different time granularities of multiple time series?

- How to graphically represent not only the time series (overview), but also allow easy interaction with the proper visualizations (interaction and presentation of the results)?

1.3 Goals

When dealing with such complex topics, many are the possible directions and issues which can be faced with. However, since it is out of the scope of this work to solve any possible questions, we have to define some more specific goals, which are the following:

- automatically find elements of interest (patterns) in multivariate time-oriented data
- manage different time intervals between elements of interests
- represent the patterns properly and in a comprehensible manner
- visually support the discovery of such patterns
- support an easy interaction with all the steps of the discovery

1.4 Basic Ideas and Overview

The basic ideas for addressing the above issues are that we must firstly describe a time model which encompasses different granularities, and allows to represent time intervals. Then, based on such time model, we can define a novel approach which is able to aid in the discovery of multi-time interval patters. Finally we can identify the proper visual and interactive support in order to help users during each phase of the analysis (and discovery).
Figure 2 sketches the overview of this work and we will follow this structure throughout the thesis.

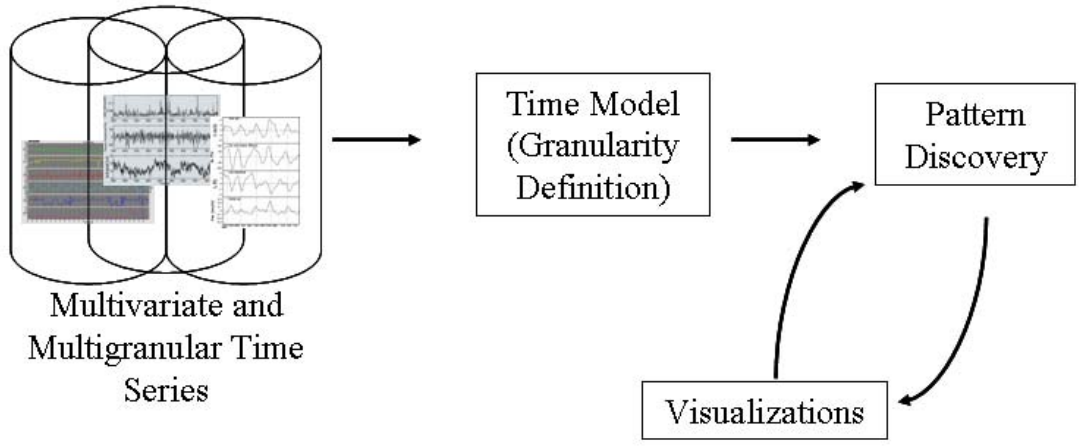

Figure 2: Abstract view of the work. Both the state of the art and the parts related to the approach are composed of three main parts, that is, Time Model (and granularity) definition, Pattern Discovery (analytical part) and Visualization.

After motivating our work and defining our goals, we will present the state of the art in research relevant for our scope, respectively regarding the Time Model (Section 2), Pattern Discovery and more in general Knowledge Discovery in Databases, Data Mining, Temporal Data Mining and Interval Mining (Section 3), and Visualization of time-oriented data (Section 4). The central part of the thesis will be devoted to the description of the approach (Section 6) and to the sketch of a possible visualization of this approach (Section 8). Moreover, we will evaluate and validate the proposed approach (Section 7) and we will then provide a short summary and concluding remarks, as well as an outlook to future directions (Section 9).

1.5 Acknowledgements

Several persons contributed to this work directly or indirectly and I would like to express my gratitude to all of them for the possibility to complete this thesis. First of all, my advisor Silvia Miksch deserves many thanks for supporting me as well as encouraging, helping, and motivating me throughout the work on this thesis, even if she doesn't like bullet points as I do. Similarly, I would like to thank my first second advisor Elpida Keravnou-Papaeliou, who, due to some administrative problems, had to drop out his role, but sent me anyway her valuable comments and input to this work, and Margit Pohl, who was very kind to accept the role of second advisor, even with short advice.

Thanks to my colleagues, in particular: Wolfgang Aigner, with whom I had long and fruitful discussions and whose hints were always well appreciated; Thomas Turic, whose role in the implementation was incredibly precious and who shows a lot of patience with my technical explanations made in German; Tim Lammarsch, whose suggestions and competences were of great help in the development phase as well as in dealing with TIS; Peter Klinka, whose technical support was always prompt, even on weekends.

Furthermore, I wish to thank Johannes Gärtner and Dieter Punzengruber from Ximes, who contributed to the discussion, especially with their experience as company, as well as the other partners from the project DisCō.

Last but definitely not the least, many thanks to my Mum and my sister, who supported me from distance, keeping in touch with me by any technical mean and sending me pesto and parmesan from Genoa. And finally to my dad, who still lives in me, and to whom this work is dedicated.

2 Time Model

What then is time? If no one asks me, I know what it is. If wish to explain it to him who asks, I do not know.

Saint Augustine

In the first part of the state of the art we will investigate how temporal information can be modelled. In particular, we will present three formal conceptual temporal data model, BCDM (*Bitemporal Conceptual Data Model*) [7], HMAP [8] and a general framework for time granularity [9],[10], as well as an example of system which is able to manage temporal data, *τZaman* [11].

As [12] clearly outlined, different research areas provide significant research efforts in the field of temporal data modeling, that is, Temporal Databases [7], [13],[14],[15], Temporal Data Modeling [16],[17],[18],[19], Temporal Reasoning [20],[21],[22], and Geographic Information Systems [23],[24]. Since temporal database research is the area that provides the most extensive body of research work (see [25],[26] for an overview) and is most relevant for our aims, it will be the focus of this part.

2.1 Bitemporal Conceptual Data Model - BCDM

The Bitemporal Conceptual Data Model (BCDM) has been created by Jensen and Snodgrass [7] with the aim to provide a semantic basis for many temporal database data models.

Based on a discrete, interval-based time model, BCDM models the underlying time domain as finite sequence of *chronons* and such sequence of chronons represents a partitioning of the real time line into equal-sized, indivisible segments. The size of single chronons is not fixed and may span from fractions of seconds to years depending on the needs of a particular domain. The time

between two chronons is called time interval. It is the only basic primitive in the BCDM and on the top of that primitives are defined as unions of time intervals.

The term "*bitemporal*" indicates that this model contains both the *valid time* perspective (to reflect the time period during which a real world fact was, is, or will be valid in the modeled reality) and the *transaction time* perspective (to reflect the time period during which a fact is/was stored in the database). This leads to the notion of bitemporal chronons which are ordered pairs of a transaction time chronon and a valid time chronon. Similarly bitemporal elements are unions of bitemporal chronons. Moreover, a so called bitemporal tuple consisting of a number of data attributes together with a bitemporal timestamp value allows to associate data (facts) with time**.** Therefore, a bitemporal conceptual relation is a set of facts where each one is timestamped with a bitemporal element. A special value of transaction time is allowed: it is called "*until changed*" or *UC* and it is used to indicate that the associated fact is current in the database until changed in the future.

Finally, some algebraic operators are provided. Similarly to the standard database operators projection and selection are provided, which results again in temporal relations. On the contrary, with no counterparts in non temporal databases, two timeslice operators, *valid-timeslice* and *transaction-timeslice*, return the tuples of a relation that were valid during a specified valid time chronon or transaction time chronon respectively.

2.2 HMAP

The temporal data model HMAP has been created by Combi and Pozzi [8] to extend the capability of defining valid times with different granularity and/or with indeterminacy.

HMAP uses a basic timeline, i.e., a point structure with precedence relation, which is a total order without right and left endpoints. The basic timeline is (partially) partitioned in non decomposable consecutive time intervals, called chronons (whereas a chronon corresponds exactly to one second). To specify a

chronon, the calendric format YY/MM/DD/HH/MI/SS (that is years/month/day/minute/second) is adopted. For example, the chronon 99/04/18/ 0/0/0 identifies the first second of April 18, 1999.

This temporal model adopts the standard set of granularities of the Gregorian calendar. Each of these granularities produces a partitioning of chronons in contiguous sets of contiguous chronons (*granules*). The produced granule is identified by the usual calendric formats, as YY/MM/DD or YY/MM. For instance, 99/04/18 identifies the granule composed by all the chronons from 99/04/18/0/0/0 to 99/04/18/23/59/59, included.

HMAP supports only a regular granularity partitioning and gaps between granules or within a granule are disallowed. Thus, for instance, mean measures are used for months and granularities such as "business-weeks" (five weekdays followed by a two-day weekend gap) are neglected.

Three basic primitives are defined: *instants*, *duration*, and *intervals*. An *instant* is a time point on the basic timeline and its position on the timeline is specified at an arbitrary granularity or with indeterminacy. In the first case, any Gregorian granularity can be used to specify an instant (e.g., YY/MM/DD or YY/MM). In the second case, an instant is specified via a lower and upper chronon that do not necessarily have to be equal with a granule, that is, an instant is a time point that may be any chronon between lower and upper bound. The *duration* is the distance between two instants and can be specified at an arbitrary granularity or with indeterminacy. The *duration* is unequivocally defined by the smallest and the greatest distance between chronons that the particular duration can coincide with (e.g., a duration of "1 minute" may be any distance of chronons between "1 minutes 0 seconds" and "1 minutes 59 seconds"). Moreover, the indeterminacy may be given explicitly by defining the lower and upper bound chronons. Besides these numerical values, the duration can assume the particular value *unknown*, if it is not possible to determine the duration itself. An *interval* is a closed set of contiguous time points, which are identified by a starting instant, an ending instant, and a duration identify. HMAP allows six different interval notations:

- the interval is a given granule of the Gregorian calendar (e.g., "year 1999"),
- starting and ending instants are given,
- starting instant and duration are given,
- ending instant and duration are given,
- intervals are defined with explicit indeterminacy by specifying an admissibility interval in form of a granule and a duration (e.g., "in April 1999 for 18 minutes"),
- starting instant, duration, and ending instant are given.

These six interval notations extend the usual way of expressing intervals having different time granularities and/or indeterminacies. Moreover, the main advantage of expressing intervals by their starting and ending instants and durations is that different granularities or indeterminacy separately for instants and duration can be used.

Some constraints are defined to guarantee the temporal data integrity. Two basic functions are defined in order to characterize such constraints: *Inf(•)* (*Infinum*) and *Sup(•)* (*Supremum*). They return the lower bound (*chronon*) of an instant or lower bound distance (*chronon count*) of a duration, and the upper bounds of instants or durations respectively.

For a variable of type interval, the following constraints are valid:

$$
\begin{aligned}
&\mathrm{Inf}(\mathrm{End}(\underline{a})) \geq \mathrm{Inf}(\mathrm{Start}(\underline{a})) \wedge \\
&\mathrm{Sup}(\mathrm{End}(\underline{a})) \geq \mathrm{Sup}(\mathrm{Start}(\underline{a})) \wedge \\
&\mathrm{Inf}(\mathrm{Dur}(\underline{a})) \geq \\
&\qquad max(0, \mathrm{Inf}(\mathrm{End}(\underline{a})) - \mathrm{Sup}(\mathrm{Start}(\underline{a})))) \wedge \\
&\mathrm{Sup}(\mathrm{Dur}(\underline{a})) \leq \mathrm{Sup}(\mathrm{End}(\underline{a})) - \mathrm{Inf}(\mathrm{Start}(\underline{a})) \wedge \\
&\mathrm{Sup}(\mathrm{Dur}(\underline{a})) \geq \mathrm{Inf}(\mathrm{End}(\underline{a})) - \mathrm{Inf}(\mathrm{Start}(\underline{a})) \wedge \\
&\mathrm{Sup}(\mathrm{Dur}(\underline{a})) \geq \mathrm{Sup}(\mathrm{End}(\underline{a})) - \mathrm{Sup}(\mathrm{Start}(\underline{a}))
\end{aligned}
$$

Figure 3: Constraints for a variable of type interval. The functions *Start()*, *End()*, and *Dur()* return the interval begin, the interval end and the interval duration respectively ([8]).

These constraints allow to avoid inconsistent situations in interval values, such as the end of an interval being before the start of the same interval.
Moreover, *temporal assertions*, that is, propositions asserted as true over a given interval (*valid interval*), allow to assign sets of non-temporal properties to a set of temporal primitives and so, to associate data with time.
Furthermore, in order to query temporal information, a set of predicates and functions (*sum of an instant and a duration*, *difference between two instants*, and *difference between an instant and a duration*) is defined in HMAP, as well as a three-valued logic (*true*, *false*, *undefined*) to support indeterminacy and an extension of Allen's basic interval relations [21] are part of the temporal model.

2.3 Time Granularity Model

A general framework for time granularity has been proposed by Bettini et al.[9],[10] in order to define a unifying model for time granularities and temporal constraints with granularities. Moreover, this model builds the basis for T_ODMG, a temporal object model supporting multi granularities management [27].
Based on *time domain*, defined as the whole set of *time instants* with a total order relationship, the authors firstly provide the notion of *granularity*: a mapping G from the integers (the so called *index set*) to subsets of the time domain such that (a) granules in a granularity do not overlap and that their index order is the same as their time domain, and (b) the subset of the index set that maps to nonempty subsets of the time domain is contiguous. For example, the usual collections *days*, *weeks*, *months*, *years* are granularities. A textual representation named *label* can then be used for each non empty *granule*. This leads to what is called *labeled granularity*, and allows, for instance, to map *day(32)* to February 1, 2001, or *month(15)* to March 2002 if we mapped *day(1)* to the subset of the time domain corresponding to January 1, 2001.
Some relationships among granularities are allowed: a granularity, e.g., G, can *group into* another granularity, e.g., H, if each granule of H is the union of

some granule of H (for instance, *days* groups into *weeks* since a week is composed of seven days). A granularity, G, can be *finer than* another granularity, H, if every granule in G is a subset of some granule in H (for example, *business-day* is finer than *day*) or similarly G can be *coarser than* H. Related to groups into, but more restrictive, a granularity G *partitions* a granularity H if G groups into and is finer than H. Moreover, a granularity G can *group periodically into* a granularity H if we have a periodic repetition of the "grouping pattern" of granules of G into granules of H. In this way we can also say that a granularity is *periodical* or *quasi periodical* (that is there are some *granularity exceptions*) with respect to another one.

The *calendar algebra* (that is the symbolic representation based on algebraic operations that capture the relationships among the granularities used in a calendar[1]) of this granularity system consists of two kinds of operations: the *grouping-oriented operations* and *granule-oriented operations*. To the former group, belong the *grouping operation* (e.g., given the granularity *day*, *week* can be generated by day(1) to day(7), where day(1) corresponds to Monday, day(2) to Tuesday, etc.), *altering-tick operation* (e.g., if a granularity represents 30-days groups and we want to add one day to every 12-th month), *shifting operation* (e.g., if we want to shift from Central European Time to Pacific Standard Time), *combining operation* (e.g., combining *business day* and *month* to obtain *business month*), *anchored grouping operation* (e.g., if we want to define each academic year as starting on the first Monday of September). Finally to the latter group belong the *subset operation* (which generates a new granularity by selecting an interval of granules from another granularity), *selecting operations* (which are binary operations and generate new granularities by selecting granule from the first operand in terms of their relationship with the granules of the second operand, and are divided into *select-down*, *select-up* and *select-by-intersect*), and *set operations* (e.g., applying set operations, that is, union, intersection, and difference to existing

[1] A *calendar* is defined as a set of granularities over a single time domain that includes a bottom granularity with respect to *group into [10].*

granularities to generate new ones, with the constraint that the granularities are “compatible” and a valid result may be obtained).

2.4 τZaman

τZaman [11] is a project directed by Professor Richard T. Snodgrass and Assistant Professor Curtis Dyreson at the Computer Science Department at the University of Arizona. It belongs to TAU[2], an umbrella project comprised of a number of inter-related and complementary projects, all with the goal of providing to users facilities to manage time-oriented data, through sophisticated user languages and APIs.

The project name is composed of the Turkish word for time, *Zaman* (pronounced *Zay-mon*), and the Greek letter, *τ* (pronounced *tau*), which denotes that it is part of the TAU project. The goal of the τZaman project is to build a native Java system providing temporal functionality for applications that need to calculate, format, parse, and/or compare times within either a single calendar or across multiple calendars (e.g., passing from Gregorian calendar to Academic or Lunar one).

More in detail, τZaman is a client/server system designed with the ability to add calendars and input-output formats on the fly, at run-time. New calendars and other user-defined information, such as natural languages or input-output formats for temporal literals, can be integrated into a multi-calendar system without recompiling or even stopping and restarting a τZaman server. Moreover, in τZaman, all calendar-related specifications are XML documents (as the Gregorian calendar specification file shown in Figure 4).

The system is based on the concepts of *calendars*, *calendric systems*, and *temporal data types* [28]: a *calendar* is a human abstraction of time; calendars are grouped into larger structures called *calendric systems*, which facilitates interaction among a group of calendars; the temporal data types are *instants*, *periods*, and *intervals*.

[2] The acronym TAU stands for “*Temporal Access for Users*” which concisely describes the project and has been chosen since the English spelling of the Greek letter "*τ*" is commonly used in scientific formalisms to denote time.

```
<calendarSpecification underlyingGranularity="second"

    <granularity name = "second">
        <irregularMapping from = "minute" relationship = "coarserToFiner" >
        <method name = "castMinuteToSecond"/>
        </irregularMapping>
    </granularity>

    <granularity name = "minute">
        <irregularMapping from = "second">
        <method name = "castSecondToMinute"/>
        </irregularMapping>
    </granularity>

    <granularity name = "hour">
        <regularMapping from = "minute" groupSize = "60"/>
    </granularity>

    <granularity name = "day" >
        <regularMapping from = "hour" groupSize = "24"/>
        <irregularMapping from = "month" relationship = "coarserToFiner">
        <method name="castMonthToDay" />
        </irregularMapping>
    </granularity>

    <granularity name = "week" >
        <regularMapping from = "day" groupSize = "7"/>
    </granularity>

    <granularity name = "month" >
        <irregularMapping from = "day">
        <method name="castDayToMonth"/>
        </irregularMapping>
    </granularity>

    <granularity name="year" >
        <regularMapping from = "month" groupSize = "12"/>
    </granularity>

</calendarSpecification>
```

Figure 4: *τ*Zaman Gregorian calendar specification expressed in XML.

τZaman allows to express complex mappings between granularities: it supports both *gap granularities* (that is, granularities which have gap(s) within individual granules, e.g., *business-months*, which not include weekend days within a month) and *holes* (which are time periods between contiguous granules, such as the two-day hole between granules representing a Friday and a Monday in the *business-days* granularity).

Another relevant aspect related to that and provided by τZaman is the possibility to perform both *regular* and *irregular mapping* between granularities. In this way, the system can manage not only for instance, the relationship between Gregorian days and Gregorian weeks which is regular since regular periods of seven days group into a week (*regular mapping*), but also for example the relationship between Gregorian days and Gregorian months (*irregular mapping*). As a matter of fact, the number of days in a

month varies from month to month, and because of leap days the same month may have a different number of days from year to year. An XML calendar specification file (as in Figure 4) stores such information and it is parsed to check its validity.

Furthermore, τZaman supports a basic set of arithmetic operations (such as division, multiplication, addition, binary subtraction) involving instances of the instant, period, and interval data types[3]. For example, one may wish to determine the arrival time of a train given its departure time and the duration of its trip by adding an interval to an instant, e.g., "April 18, 2004" + "1 day" gives the arrival instant, which is "April 19, 2004".

Finally τZaman has a complete set of temporal comparison operations, so that for example, one might discover which employees were hired during a particular year, or given two employees, who has more seniority. τZaman extends Allen's operators [21] with an analogous set of operators for the instant and interval data types, allowing temporal comparison operations such as during, before, after, and so on.

2.5 Discussion

In the first part of the state of the art, we presented two examples for temporal data models, a framework for time granularities, and an example of a system (and its related project) able to manage multiple calendars as well as operations on them. Firstly, we investigated the formal data model BCDM [7], which is often used in temporal database research for comparing different temporal data models in terms of their expressiveness and is the basis for the TSQL2 query language [162]. The main goal of BCDM is to allow a simple versioning using two temporal perspectives (valid time and transaction time).

[3] Note that not all combinations of operations are defined. For example, *instant* * *instant* is undefined since no reasonable semantics for that expression exists. Moreover, since integral indexes are adopted to represent granules, the result of any operation must be integral, either a determinate value or an indeterminate value.

However, due to its simplicity, this model allows only intervals and does not support indeterminacies and granularities, which are frequently present in real world (e.g., "Everyday it takes me one hour-one hour and half to reach my office"). Secondly, the more recent model HMAP [8] has been introduced, which has been developed to support different granularities and indeterminacy for modeling information in natural-language expressions. Among the formal models, HMAP is more expressive and flexible in terms of primitives and determinacy, but uses a fixed calendar. Moreover, it is able to model only valid time history and supports explicit indeterminacy, but only on a chronon level which limits the possibility of flexible primitive specifications.

Thirdly, a general framework for time granularity [9],[10] has been introduced. This framework has been proposed to define a unifying model for time granularities and temporal constraints with granularities. Moreover, this model builds the basis for T_ODMG, a temporal object model supporting multi granularities management [27]. It supports granularities (and labeled granularities) and algebraic calendar expressions (which can be represented as periodical set of instants).

After that, we presented τZaman [11], a project of the Computer Science Department at the University of Arizona. The goal of τZaman is to build a system providing temporal functionality for applications that need to calculate, format, parse, and/or compare times within either a single calendar or across multiple calendars. It supports calendars, granularities and gap granularities, as well as regular and irregular mappings between granularities and in general it represents an example of practical application of granularity system and calendars management.

Finally, we must outline that, on the one hand, the main goal of temporal data models in temporal database research is to represent time-oriented real world information within an information system. Our aim, on the other hand, is to be able to analyze and capture the characteristics of many different temporal data sets, determining the most frequently occurring multi-time interval patterns and ultimately visualizing them.

3 Pattern Discovery

When exploring complex data sets, the ease with which a question can be answered often determines whether it will be asked at all.

Howard Wainer

In the second part of our state of the art, the notions of Knowledge Discovery in Databases, Data Mining and Temporal Data Mining will be investigated. Moreover, a brief overview on the possible definition of the term *pattern* (and *temporal pattern*) will be provided.

3.1 Knowledge Discovery in Databases and Data Mining

The notion of finding useful patterns in data has been given traditionally a variety of names, including Data Mining, knowledge extraction, information discovery, information harvesting, data archaeology, and data pattern processing. The term Data Mining has mostly been used by statisticians, data analysts, and the management information systems communities [29]. The phrase *Knowledge Discovery in Databases* (KDD) was coined at the first KDD workshop in 1989 to emphasize that knowledge is the end product of a data-driven discovery [30]. KDD has evolved, and continues to evolve, from the intersection of research fields such as machine learning, pattern recognition, databases, statistics, artificial intelligence, knowledge acquisition for expert systems, data visualization, and high-performance computing. The unifying goal is extracting high-level knowledge from low-level data in the context of large data sets [29].

As Figure 5 illustrates, "*Knowledge Discovery in Databases is a non-trivial process of identifying patterns in data that are: valid (in the statistic sense), novel (at least to the system and to the user), potentially useful (for a given*

application) and ultimately understandable (immediately or after post processing). ([31])

The basic steps of the KDD pipeline are then the following:

- *Selection.* A target data set is created by selecting a data set or by focusing on a subset of data attributes or data samples, on which Knowledge Discovery is to be performed.

- *Preprocessing*. Basic preprocessing operations include removing noise if appropriate, accounting for time-sequence information and known changes, deciding on strategies for handling missing attribute values and collecting the necessary information to model or account for noise.

- *Transformation*. The data transformation consists of data reduction and data projection to find useful features to represent the data depending on the goal of the task. The effective number of attributes under consideration can be reduced or invariant representation for the data can be found.

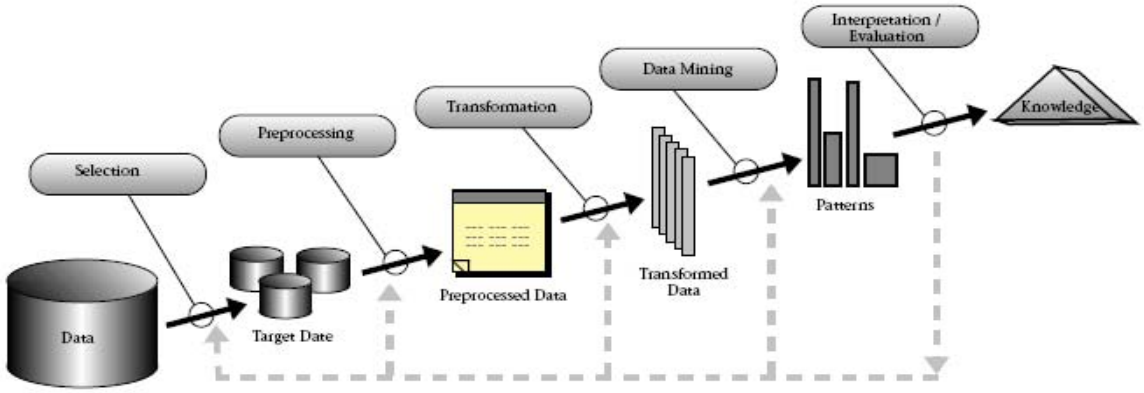

Figure 5: Overview of the KDD pipeline ([29]).

- *Data Mining.* It consists on the search for patterns of interest in a particular representational form or a set of such representations: such as classification rules and trees, clustering or association rules.

- *Interpretation/Evaluation.* The mined data are interpreted, possibly returning to any of the preceding steps for further iteration. The evaluation can also involve a visualisation of the extracted patterns and models, or a visualisation of the data given the extracted model.

Therefore, differently from e.g., pattern recognition, machine learning[4] and so on, which provides only some parts of the Data Mining step, KDD focuses on the overall process of Knowledge Discovery from data, including how the data are stored and accessed, how algorithms can be scaled to massive data sets ultimate and still run efficiently, how results can be interpreted and visualized, and how the overall man-machine interaction can usefully be modelled and supported. The KDD process can be viewed as a multidisciplinary activity that encompasses techniques beyond the scope of any one particular discipline such as machine learning [29].

3.2 Approaches to Data Mining

Berry and Linoff [32] have clearly explained the different approaches to Data Mining. They called them methodologies for Data Mining: their classification is based on the steps one would take to do mining and it is not concerned neither with the type of the outcomes nor with techniques.
We can distinguish the following approaches to carry out Data Mining [2] : top-down, bottom-up, and hybrid.

3.2.1 Top-Down Approach

In the top-down approach you have to start with some idea or a pattern or a hypothesis. Then you start querying the database and test your ideas and hypothesis: if something that does not confirm your hypothesis appears, the hypothesis has to be revised. In general, hypothesis testing is about generating ideas, developing models and then evaluating the model to determine if the

[4] Even if partially out of the focus of this state of the art, a short overview on the actual status of the research in the field of machine learning and time is given in Appendix I.

hypothesis is valid or not. Of course the main task is to develop a model: if it is not a good one, then you cannot rely on the outcome. Moreover, the model could simply be a collection of rules of the form "if a person lives in Vienna, then he owns a house worth more than 250K €"; then to evaluate the model, one needs to query the database. In the above example, one could query to select all those living in Vienna and owning houses costing less than 250K €.

3.2.2 Bottom-Up Approach

Another approach to Data Mining is called "bottom-up". In this approach there is no hypothesis to test and, of course, this is much harder as the tool has to examine the data and then come up with patterns. The bottom-up approach can be distinguished as directed or undirected. In the directed Data Mining, also referred to as *supervised learning*, you have some ideas of what you are looking for. For example, who often travels with Alex to Krems? Which item is often purchased with coffee? Like the top-down approach, models are developed and they are evaluated based on the data you analyze. On the contrary, in the undirected Data Mining, also referred as *unsupervised learning* in the machine learning literature, you have no idea of what you are looking for. Then you ask the tool to find something interesting; for instance, in image Data Mining, the Data Mining tool can find something that it appears as unusual. As before, a model has to be developed and evaluated with the data; once something interesting has been found, the directed Data Mining can be applied.

3.2.3 Hybrid Approach

The third approach is a so-called hybrid one, since it is a combination of both top-down and bottom-up mining. For instance, you can start with bottom-up mining, analyze the data and then discover a pattern. This pattern could be a hypothesis and you can now use top-down mining to test the hypothesis. As a result, you can find new patterns that become a new hypothesis; that is, the

tool can switch between top-down and bottom-up mining and again between directed and undirected mining.

3.3 Data Mining Methods

At a higher level, we can distinguish two types of Knowledge Discovery goals: *verification* (that is, verifying user's hypothesis) and *discovery* (finding new patterns automatically). The latter one can be further divided into two other sub goals: *prediction* and *description* [29]. Prediction involves using some variables or fields in the database to predict unknown or future values of other variables of interest. On the contrary, description focuses on finding human-interpretable patterns describing the data.

Before going on with the description of Data Mining tasks, we remind some concepts and terms we cope with in the following. A *model* is a structure and corresponding interpretation that summarizes or partially summarizes a set of data, for description or prediction [33]. The most of inductive algorithms generate models that can then be used as classifiers, as regressors, as patterns for human consumption, and/or as input to subsequent stages of the KDD process; a *class* is a set of objects having particular patterns or features; finally, a *data item* is a representative of data.

There are various types of Data Mining in the sense of what the outcomes will be, and many authors refer to the same method by using different substantive. Yet, the terms Data Mining outcomes, Data Mining tasks and Data Mining methods are often used as synonyms, and throughout this thesis we will use them interchangeably. Thus, for a clear explanation we first list and then we provide a short explanation, along with an example.

Some of these outcomes are:

- Classification
- Clustering
- Characterization (or Summarization)
- Regression (Prediction)
- Change and Deviation Detection (that is Trend Detection)
- Dependency Analysis and Modelling
- Association Rules
- Search & Retrieval
- Pattern Discovery

Classification is learning a function that maps (classifies) a data item into one of several predefined classes [34]. Classification is carried out by developing training sets with pre-classified examples and then building a model that fits the description of the class [2]. Examples of classification methods used as part of Knowledge Discovery applications are the classification of trends in financial markets, and the automated identification of objects of interest in large image databases [1].

For instance, a bank may wish to classify clients requesting loans into categories based on the likelihood of repayment. A rule or formula for making the classification is developed from the data in the training set (see Figure 6). Then the reliability of the rule or formula is evaluated using the test set of data and this gives an indication of how well the procedure will work on the remaining bulk of the data

Clustering is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data [35] (Figure 7). The categories can be mutually exclusive and exhaustive or consist of a richer representation, such as hierarchical or overlapping categories. Furthermore, clustering is a Data Mining outcome that is often confused with classification. While classification classifies an entity based on some predefined values of attributes, clustering groups similar records not based on some predefined values [2].

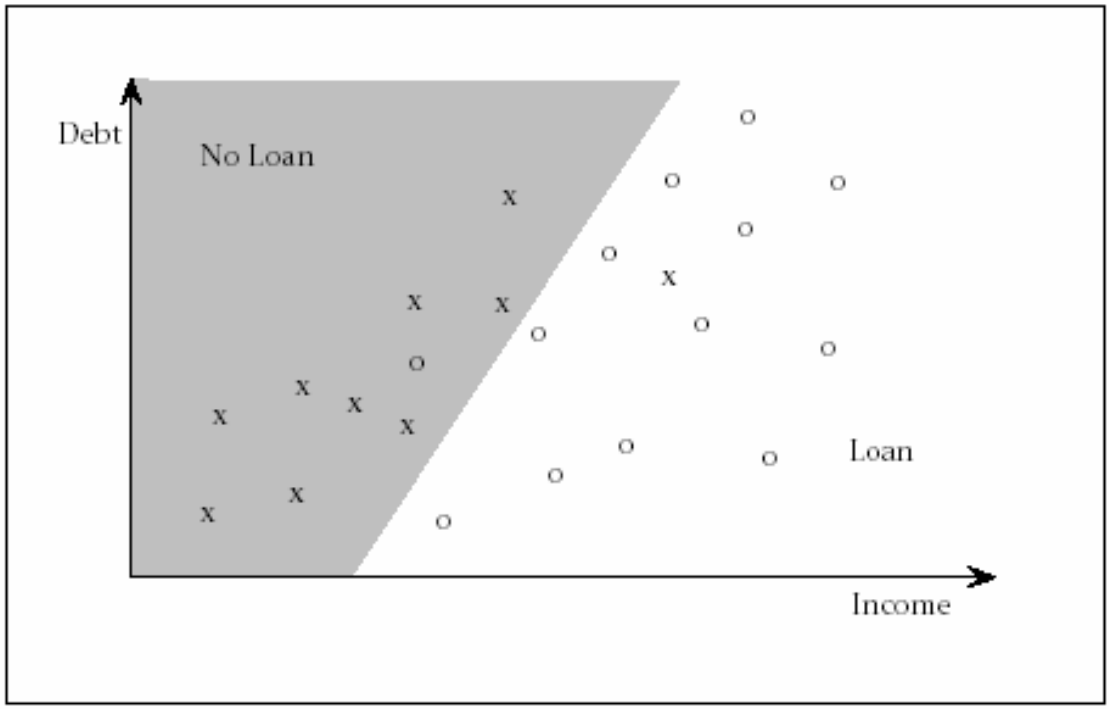

Figure 6: Classification for the loan data set consisting of 23 cases [1]. Each point on the graph represents a person who has been given a loan by a particular bank at some time in the past. The horizontal axis represents the income of the person; the vertical axis represents the total personal debt of the person (mortgage, car payments, and so on). The data have been classified into two classes: (1) "x" represents persons who have defaulted on their loans and (2) "o" represents persons whose loans are in good status with the bank (whose request will more likely be accepted).

Examples of clustering applications in a Knowledge Discovery context are discovering homogeneous subpopulations for consumers in marketing databases and identifying subcategories of spectra from infrared sky measurements [36].

Characterization (or *Summarization*) is a process that identifies a compact description for a subset of data. A simple example would be tabulating the mean and standard deviations for all fields. More sophisticated methods involve the derivation of summary rules [37], multivariate visualisation techniques, and the discovery of functional relationships between variables

[38]. Finally, summarization techniques are often applied to interactive exploratory data analysis and automated report generation.

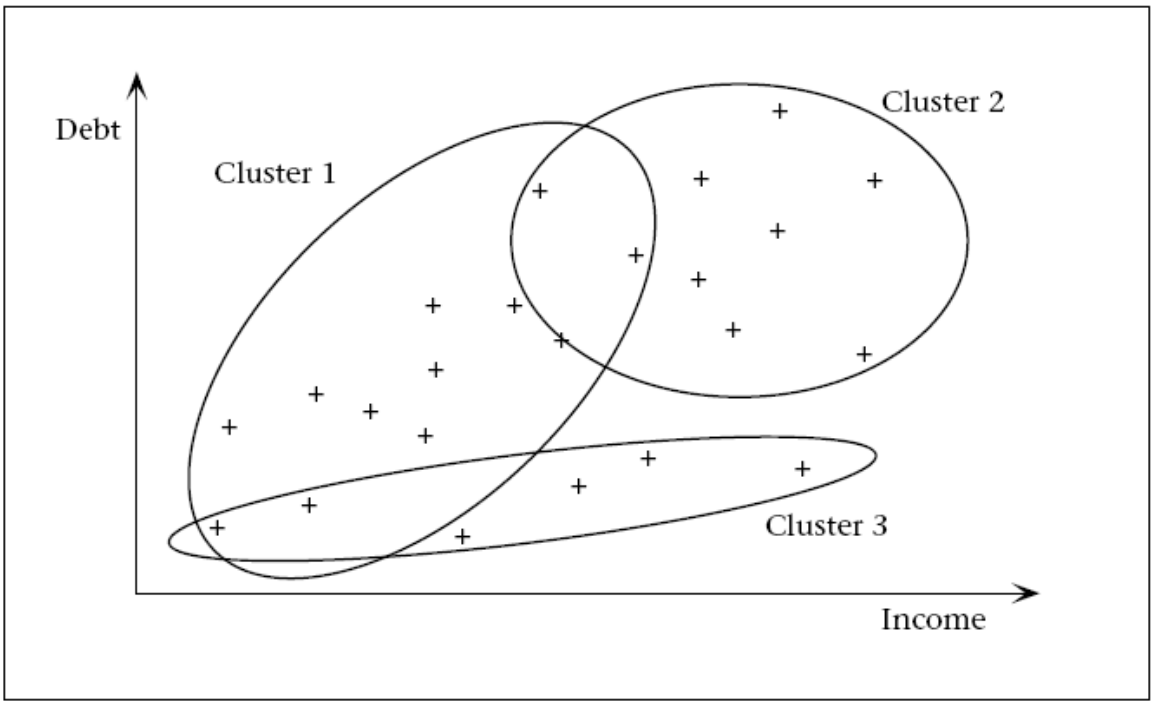

Figure 7: Clustering of the loan data set [1], whereas all the labels have been replaced by "+".

Regression is a method for discovering a function that maps a data item to a real-valued prediction variable and it is used to fit an equation to a dataset (an example is shown in Figure 8). The simplest form of regression, *linear regression*, uses the formula of a straight line *($y = mx + b$)* and determines the appropriate values for m and b to predict the value of y based upon a given value of x. Advanced techniques, such as multiple regression, allow the use of more than one input variable and allow for the fitting of more complex models, such as a quadratic equation. Several are the possible applications of regression. For example, predicting the amount of biomass present in a forest given remotely sensed microwave measurements, estimating the probability that a patient will survive given the results of a set of diagnostic tests, predicting consumer demand for a new product as a function of advertising

expenditure, and predicting time series where the input variables can be time-lagged versions of the prediction variable.

For instance, given the numbers of cars sold during the previous months, it would be possible to predict the number of cars which will be sold during the following month(s).

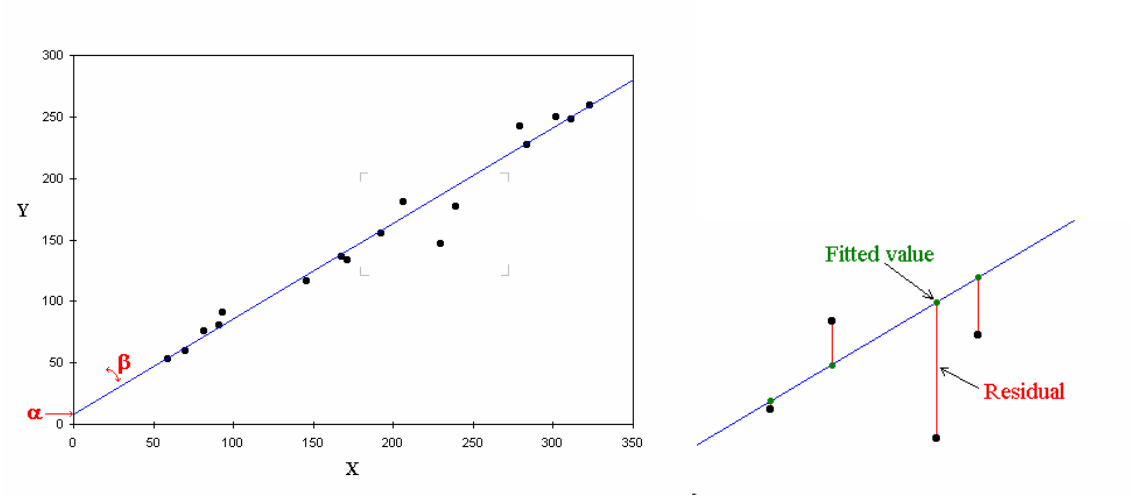

Figure 8: An example of regression, where the regression line is given by the equation "Y = α + β X + ε". In this case, α represents where the line crosses the y-axis, β the slope of the line, and ε error term describing the variation of the real data above and below the line. On the right, a zoom of the situation is shown.

Trend Detection focuses on discovering the most significant changes in the data from previously measured or normative values [39],[40],[41].

For instance, given a sequence of values representing the ice cream sold per day, it could be determined how the consumption of ice cream vary, i.e., if trends (sales grow 10% each year), cycles (summer sales are larger by 60% than winter sales) are present. Similarly, Figure 9 shows "uptrend" and "downtrend" in the currency conversion between USD and GBP.

Dependency Analysis and Modelling consists of finding a model that describes significant dependencies between variables. Dependency models exist at two levels: (1) the structural level of the model specifies (often in a graphic form)

which variables are locally dependent on each other and (2) the quantitative level of the model specifies the strengths of the dependencies using a numeric scale. For example, probabilistic dependency networks use conditional independence to specify the structural aspect of the model and probabilities or correlations to specify the strengths of the dependencies [42]. Probabilistic dependency networks are increasingly finding applications in areas as diverse as the development of probabilistic medical expert systems from databases, information retrieval, and modelling of the human genome.

Figure 9: The trend ("uptrend" and "downtrend") in the currency conversion between USD and GBP[5].

Association Rules are a well known task and often related to the previous one. An association rule is expressed in the form "if this then that" and associates events in a database (given the events A and B, the rule "if A then B" is graphically represented as "A → B"). The strength of an association rule can be measured in terms of its support and confidence [43]: *support* (or coverage) is the relative frequency or number of times a rule produced by a rule induction system occurs within the database. The higher is the support, the better is the

[5] http://www.metaquotes.net/techanalysis/trendlines/

chance of the rule capturing a statistically significant pattern. The measure of how often the collection of items in an association occurs together as a percentage of all the transactions. *Confidence* (or accuracy) of rule "B given A" is a measure of how much more likely it is that B occurs when A has occurred. It is expressed as a percentage, with 100% meaning B always occurs if A has occurred.

A typical example is the association between purchased items at a supermarket or in an on-line bookstore as explained below.

Let suppose you are collecting data at the check-out cash registers at a large book store. Each customer transaction is logged in a database, and consists of the titles of the books purchased by the respective customer, perhaps additional magazine titles and other gift items that were purchased, and so on. Hence, each record in the database will represent one customer (transaction), and may consist of a single book purchased by that customer, or it may consist of many (perhaps hundreds of) different items that were purchased, arranged in an arbitrary order depending on the order in which the different items (books, magazines, and so on) came down the conveyor belt at the cash register. The purpose of the analysis is to find associations between the items that were purchased, i.e., to derive association rules that identify the items and co-occurrences of different items that appear with the greatest (co-)frequencies. For example, you want to learn which books are likely to be purchased by a customer who you know already purchased (or is about to purchase) a particular book. This type of information could then quickly be used to suggest to the customer those additional titles. You may already be “familiar” with the results of these types of analyses, if you are a customer of various on-line (Web-based) retail businesses (see Figure 10). Many times when making a purchase on-line, the vendor will suggest similar items (to the ones purchased by you) at the time of "check-out", based on some rules such as "customers who buy book title A are also likely to purchase book title B," and so on.

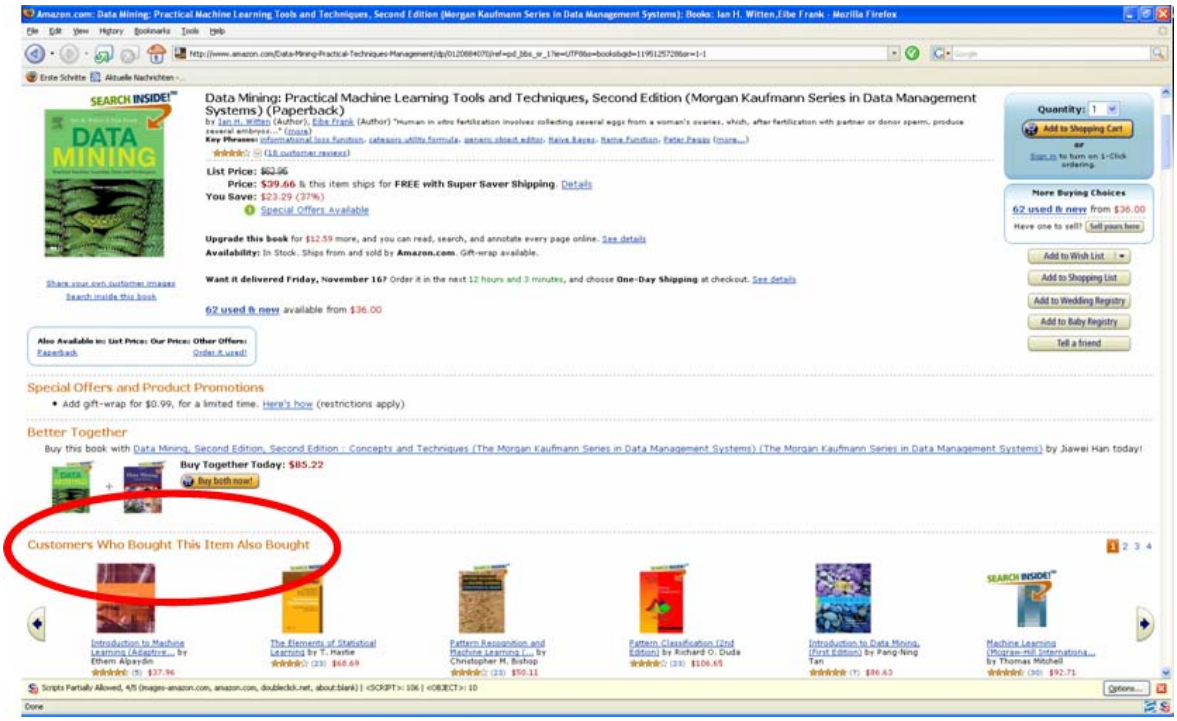

Figure 10: An application example of association rules. When selecting a certain book e.g., on Amazon.com, also items purchased along with this book by other customers are shown.

Search & retrieval encompasses searching for a priori specified queries in possibly large volumes of data and is often also referred to as "query-by-example". It can be applied to locate exact matches for an example query or approximate matches. In the latter case, similarity measures are needed that define the degree of exactness or fuzziness of the search (e.g., to find customers whose spending pattern over time are similar but not necessarily equal to a given spending profile) [44].

In contrast to the previously presented task of search & retrieval, *pattern discovery* is concerned with automatically discovering of interesting patterns (without the need for predefining queries). Often, frequently occurring patterns are of interest, like for example to analyze whether a TV commercial generally leads to an increase in sales. In contrast to that, it is often also of interest to discover patterns that occur very rarely because these occurrences

can reveal hostile behaviour or failures (e.g., mining for patterns of network security breaches) [44].

3.4 Temporal Data Mining

The adoption of time in the Data Mining tasks leads to what is known as Temporal Data Mining. To incorporate time in the Data Mining process gives us the ability to detect activities rather than just states, i.e. the ability to find behavioural aspects of communities of objects rather then just describing their states at a certain point in time.

Temporal Data Mining covers several different topics, such as the discovery of frequent, or interesting, sequences in a time series (e.g., finding customers whose spending pattern over time are similar to a given spending profile), the detection of relations between different sequences (e.g., in the stock market analysis we are interested in finding rules that describe relations between different stocks) and so on. Temporal Data Mining is used in applications from various fields, such as medicine, finance, engineering and meteorology. Applications in Temporal Data Mining often combine several different methods to reach a solution.

Even if time series analysis and Temporal Data Mining might appear similar they mainly differ for two reasons [44]: firstly, the size and nature of data sets they deal with. Temporal Data Mining methods may analyse data sets that are prohibitively large for conventional time series modelling techniques. Moreover, the sequences may be nominal-valued or symbolic (rather than being real or complex-valued), and techniques to model the time series such as autoregressive moving average (ARMA) [45] or autoregressive integrated moving average (ARIMA) [46] are inapplicable. Secondly, the information that one wants to extract from the data. As a matter of fact the scope of Temporal Data Mining overcomes the standard forecast or control applications of time series analysis. Moreover, in several (temporal) Data Mining applications,

what correlations are available and among what variables (and events) are usually unexpected, as well as hidden or unexpected trends or patterns, which are actually the ultimate goals of Temporal Data Mining.

Finally, note that sequence matching can be divided into whole matching (a query time series is matched against a database of individual time series to identify the ones similar to the query) and subsequence matching (a short query subsequence time series is matched against longer time series by sliding it along the longer sequence, looking for the best matching location). However, subsequence matching can be generalized to whole matching by dividing sequences into non-overlapping sections [47].

The discovery of relations between sequences (and sub-sequences) of events can be divided into three phases: the representation and modelling of the data sequence, the definition of similarity measures between sequences, and the application of models and representations to the actual mining problems. For this division and the terms used we referred to [44] and [48].

Representation of Temporal Sequences

The problem of how to represent temporal sequences is essential above all when dealing with time series, as it is extremely difficult to perform efficiently a direct manipulation of continuous, high-dimensional data.

The literature provides different solutions to it: firstly, it is possible to use the data with only minimal transformation, either keeping it in its original form (as in [49],[50]) or using windowing and piecewise linear approximations to obtain manageable sub-sequences ([51],[52]). Secondly, it is possible to map the data into a more manageable space, either continuous (e.g., using a Discrete Fourier Transform [53] or Discrete Wavelet Transform [54]) or discrete (e.g., using ad hoc languages plus a blurry matching [55],[56], clustering [57] or self organizing maps [58],[59]). Thirdly, it is possible to use a generative model, in which a statistical or deterministic model is obtained

from the data and can be applied to answer more complex questions ([60], [61][62],[63]). Apart from these cases, it is possible to deal with transactional databases with timing information as proposed in [64],[65].
Finally, all the methods should be able to discover and represent the sub-sequences of a sequence, e.g., using sliding window over the sequence to define a new sub-sequence, composed by the elements inside the window [66].

Similarity Measures for Sequences

The second phase in the discovery of relations between sequences (and sub-sequences) of events is the definition of a similarity measure between sequences, taking also into account outliers, data noise, amplitude differences, time axis distortion problems and so on.

The literature provides different proposals: for the time-domain continuous representations, Euclidean distances [49] and Dynamic Time Warping ([68],[69],[70]) are most commonly adopted; in case of transformation based methods, Euclidean distances [53],[54] or approaches finding relevant sub-sequences in the original data [66] are used; in case of similarity measure in discrete space, ad hoc languages plus a blurry matching [55] or algorithms for string editing to define a distance over the space of strings of discrete symbols [71],[72] can be applied. Finally for generative models, in case of deterministic models, verifying if a sequence matches a given model will often provide a choice between only two possible values (*yes/no*), in case of stochastic models, it is possible to obtain a number indicating the probability that a given sequence was generated by a given model, without using complex similarity measures between sequences.

3.4.1 Temporal Data Mining Methods

Being an extension of Data Mining, the methods previously described can be adapted and extended to their temporal counterparts. For such a reason, we will provide only a short overview of these methods.

One of the main Data Mining tasks is the discovery of association rules to capture correlations between attributes. In such cases, the conditional probability of the consequent occurring given the antecedent is referred to as confidence of the rule and a well known algorithm to discover association rules is the Apriori algorithm [73]. Other methods also consider cyclic rules [74], which are rules that occur at regular time intervals and calendrical association rules, related to different time units [75].

In the task of classification each sequence has to belong to a finite and predefined set of classes, for instance in applications such as speech recognition ([76],[77]), gesture recognition [78], and handwritten word recognition ([79],[80]).

Clustering of sequences or time series consists of grouping a collection of time series (or sequences) based on their similarity. Examples of clustering applications are web activity logs, or financial data. The fundamental problem with clustering is to find the number of clusters to represent the different sequences and initialise their parameters. However, supposing to know the number of clusters along with some Markov models [81] or adopting hierarchical clustering, like COBWEB [82],[83], to cluster temporal sequences databases, these problems can be faced with.

Finally, we would like to mention the prediction task. Time-series prediction consists of forecasting (typically) future values of the time series based on its past samples. In order to perform this task, a predictive model for the data is needed. Different predictive models can be applied assuming the time series to

be stationary ([84],[85],[86]), nonstationary (like the autoregressive integrative moving average) or locally stationary (where the series is divided into smaller frames within each of which, the stationarity condition can be assumed to hold).

3.4.2 Interval Mining and Sequence Pattern Mining

A special case of Temporal Data Mining deals in particular with the issue of time interval.

Apart from the easiest solution (that is ignoring the time intervals completely) which we do not take into account, the time-window approach is well known. For instance, Mannila et al. [61] specify the window width in order to find frequent episodes in event sequences, so that both serial episodes and parallel episodes may be found. In a similar approach, Srikant and Agrawal [87] also provide a minimum and a maximum value for the window size, but both methods cannot find the time intervals between events and may miss patterns if they exceed the window size. Villafane et al. [88] criticize the same point and propose a technique to discover temporal containment relationships, but they still miss time intervals between events.

Differently from them, Magnusson [89] firstly define so called *T-Patterns* (that is, "*characteristics of well-known behaviour patterns abstracted and combined in order to define a scale-independent, hierarchical time pattern type*") and then propose an algorithm to detect particular relationships (*critical intervals,* that is an interval that tends to contain at least one more occurrence of a pattern than would be expected by chance) between pair of events. Moreover, such approach is particularly interesting, since it is also provided with a visualization (the *Theme* program) shown in Figure 11. In detail, starting from the left, the uppermost part and the middle part show some possible tree to represent the T-Pattern ABCD. The middle-left box shows the hierarchical construction of the pattern, whereas the middle-right box shows the occurrence time points of each of its event types and the connection of points to form pattern occurrences. In the uppermost part, concurrent events appear on top of

each other, whereas in the right and bottom boxes, their connection branch becomes a simple line whereby the binary tree becomes partly invisible. The bottom box is like the uppermost part, but with no letters and with only complete pattern instances showing. All of the parts involve the same [1, N_T] observation period. Then, regarding the right part of Figure 11, a T-pattern composed of 25 event types, numbered 1 to 25, gradually detected by the bottom-up, level-by-level (breadth-first) algorithm of the Theme program. For each event type in the left box, its occurrence series, as in the behaviour record, appears immediately to its right in the right box. The first connected event types (left box)-that is, those at Level One-are (right box occurrences of) event types (1 and 2), (6 and 7), (8 and 9), (11 and 12), (14 and 15), and so forth. At Level Two, they are ((1 and 2) and 3), (5 (6 and 7)), (13 and (14 and 15)), and so forth. Time distances involved at each Critical Interval connection appear in the horizontal part of point connections, which appear, for example, clearly at the last (7th) level, where subpattern (1. . . 16) is connected to subpattern (17.... 25).

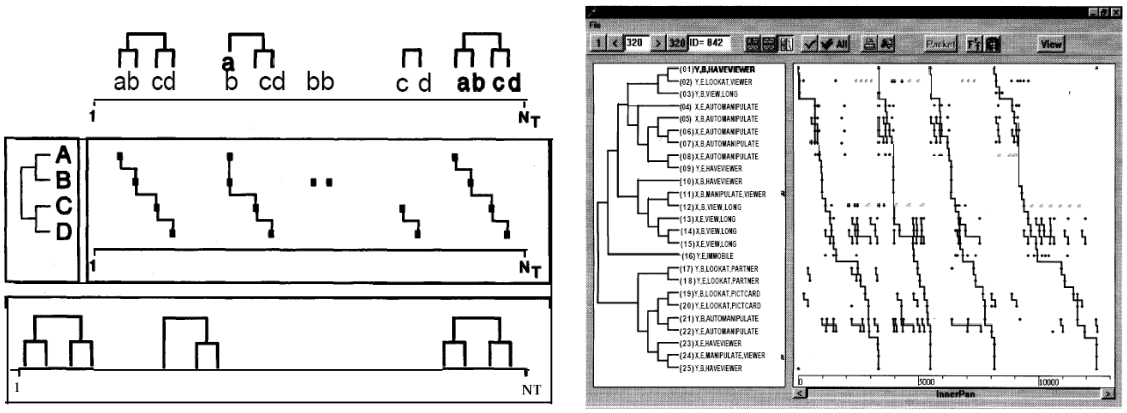

Figure 11: T-Patterns (on the left) and the implemented version of the approach proposed by Magnusson, *Theme* program [89] (on the right).

Some approaches face the problem of mining sequential patterns by annotating sequential patterns with a frequent transaction time derived from the source data. For instance, Yoshida et al [90] propose an algorithm to find so called *delta patterns*, that is, an ordered list of itemsets with the time

interval between two neighbouring itemsets that occur in a sufficient number of sequences. The proposed algorithm clusters time intervals between two neighbouring itemsets using clustering-features vectors (CF-Tree) and count the number of occurrences of each candidate pattern. An extension of delta patterns is proposed by Vautier et al. [91]. However, rather than focusing on delta patterns, which were limited to sequential constraints, they propose the notion of *chronicles*, that is, a set of temporal constraints between events which can be represented as graph. In this graph, the vertices represent the events, the edges are the temporal constraints between couples of events and as in the case of delta patterns, "*the temporal constraint takes the form of interval, bounding the difference between the connected event times*". Similarly, Giannotti et al [92] introduce a novel form of sequential pattern, called *Temporally-annotated sequences* (TAS), and based on the combination of the concepts of frequency and temporal similarity. In detail, given a candidate sequence, they search for its occurrences through all the transactions in the dataset and record the time separating an itemset from the next one. Then, for each sequence, they pass its annotations to a clustering algorithm, which gives the representative values as output, and finally they calculate the support for each representative.

Other approaches exploit the Apriori algorithm: Srikant and Agrawal [87] mine sequential patterns for point based events, Chen and Wu [93] first define temporal sequences consisting of interval-based events and then propose a so called "T-Apriori" algorithm to discover temporal patterns from them. Chen et al. [5] and Yang [6] focus on the time intervals between successive items (namely "time interval sequential patterns"): first, the time interval is partitioned into a set of fixed time-intervals, second, they are analyzed using a newly presented extension of the Apriori algorithm, called "I-Apriori". More recently, Hu et al. [93] reprise and extend these ideas also adding the possibility to deal with different time-intervals between different pairs of items, also testing the experimental results from the proposed MI-Apriori and MI-PrefixSpan algorithms. Finally must be cited some of the directions in

which sequence mining evolved, that is, to deal with various form of patterns (e.g., cyclic [95],[96],[97], similar [98],[99],[53], traversal [100],[101] or multidimensional patterns [102]), to be applied in web application as well as in the e-commerce [103], or simply to improve methods for mining sequential patterns [104],[105],[106].

For a further discussion and a more detailed comparison between the proposed approach and the methods here cited see Section 6.5.

3.5 Overview Definition of Temporal Pattern

Since Data Mining in general is considered a step of the Knowledge Discovery in Databases (KDD) process, which comprises the search for patterns of interest in a particular representational form [1], in this section we will provide a brief overview on some of the possible definitions of (temporal) pattern.

There is no fully accepted definition of pattern in the scientific community. Definitions vary depending on the field of application or research. For example, in the genetic field, Sacchi et al. [107] states that "*it is often of interest to detect the occurrence of certain temporal patterns, i.e., time intervals in which one or more time series assume behaviour of interest".*

Yet, when dealing with body motion or signals, "*a temporal pattern is defined as a segment of signals that recurs frequently in the whole temporal signal sequence. For example, the temporal signal sequences could be the movements of head, hand, and body, a piece of music, and so on. The patterns of the body movement represent the habit of a person.*" [108]

The discovery of events or sequences of events is another field where many definitions can be found: Bettini et al. [109] assert that "*The process (of discovering temporal pattern of events) usually starts with the specification by the user of an event structure which consists of a number of variables*

representing events and temporal constraints among these variables. The goal of the Data Mining is to find temporal patterns, i.e., instantiations of the variables in the structure, which frequently appear in the time sequence". On the contrary, Mannila et al. [110] state that "*one basic problem in analyzing event sequences is to find frequent episodes i.e., collections of events occurring frequently together*. Höppner [111] claims that "*a pattern thus consists of a set of intervals, their labels, and their interval relationships (like before, meets, overlaps, etc.) Sometimes, the true patterns in the data cannot be represented by a single pattern of our pattern space, for instance "B starts some time after A" may manifest in pattern "A before B"', "A meets B" or even "A overlaps B"* ". Furthermore, Povinelli et al., [112] claim that "*one of the approaches is to identify time-ordered structures (called temporal patterns) in the time series that are characteristic and predictive of the events of interest*".

Finally, using temporal abstraction in order to find interesting sequences of events, Shahar [113] states that "*an additional abstraction type is pattern which defines a temporal pattern of several other parameters*".

3.6 Discussion

In the second part of the state of the art, we firstly presented the notions of Knowledge Discovery in Databases and Data Mining. Then we extended such concepts to be able to deal with time-oriented data and ended this part, giving an overview on some possible definitions of the term temporal pattern.

In general, we would like to note that even if the definition of Data Mining provided by Fayyad [1] is well known, many groups, blogs, social networks[6] still discuss about how to update or improve it. On the one hand, giving more relevance to the fact that Data Mining has to deal not only with large amount of data, but also with few (or too few) amount of data. On the other hand, claiming that the key is that the patterns are otherwise hidden to eyeball scanning or analysis using only traditional statistical approaches and

[6] A post recently sent to the social network "Data-Mi-ning" (http://data-mi.ning.com/) suggested to amend the definition given in the header of the website and several comments followed that.

therefore Data Mining has to be considered as a collective name for theories and methods from multiple disciplines instead of a monolithic dogma; and the kinds of problems Data Mining is set to address are even broader.

Even more complicated is the situation when dealing with a definition of pattern. As a matter of fact, in the scientific community there is no fully accepted definition of (temporal) pattern, but rather several different definitions according to different research or application fields.

However, even if a contribution to the discussion about the definition of Data Mining is out of the scope of this thesis, the definition of pattern is very relevant for our goals. In fact, after describing a time model for dealing with the scenarios given in the introduction (Section 6.1), we will provide a definition of temporal pattern which appears more suitable to the aim of finding multi-time interval patterns and ultimately visualize them. Finally, as Section 3.4.2 outlined, Temporal Data Mining methods dealing with the issue of time interval do not provide satisfactory solutions to the problem. On the contrary, apart from some exceptions, the temporal information in between is often neglected. Therefore, we believe that the discovery of repeating patterns preserving the information in between is an interesting task, and this will be the focus of this work.

4 Visualization of Time-Oriented Data

Humans are becoming exponentially immersed in a new information-based reality, pouring out of supercomputers, digital instruments, and stored databases...All that comes out of the computer is an endless stream of numbers....We have to invent ideas for how to represent the numbers as visual paradigms and then turn those ideas into algorithms and programs...Visual output is the only sensible way for scientists to couple to this numerical reality [and to communicate] the results of scientific research to our colleagues in other fields and to the public at large.. The power of visualization for this purpose lies in the fact that the image is a much more universal language than the underlying mathematics in which the science is couched.

Robert Wolff and Larry Yaeger,
Visualization of Natural Phenomena,
1993

In this third part of our state of the art we will investigate which aspects are relevant when visualizing time-oriented data. A huge variety of concepts for analyzing time-oriented data are known in the literature and this makes difficult to assess the current state of the art in visualization of time-oriented data. For such reason, rather than overview a long and not organized catalogue of visualizations, we present three different possible categorization schemas, from the less recent one to the most recent one. Moreover, we conclude this part of our state of the art by presenting a brief overview of Information Visualization techniques organized as techniques leaving the discovery process mainly to humans, visual query techniques, and combined methods, that is, methods which try to jointly apply visual and computational methods to analyze time-oriented data.

4.1 Motivation

The analysis of time-oriented data is an important task in many application fields, however visualizing such data is not an easy task. Even if in recent years many approaches to this aim have been proposed, most of them aim to solve only a particular analysis problem, rather than provide a general solution. The reason is that it is difficult to consider all aspects involved when visualizing time-oriented data. For instance, the temporal relations to be taken into account when dealing with time points or time intervals are different; similarly whether to consider single or multiple variables as well as the intrinsic characteristic of data themselves lead to different choices[7].

Yet, some approaches treat time as a quantitative dimension, that is, as one quantitative variable among many others. Examples of this case are common visualization frameworks like the Xmdv-Tool [114], Visage [115], or even standard visualization techniques such as Parallel Coordinates Plot [116] (as Figure 12 clearly shows, one of the possible variable is "*Year*", that is time).

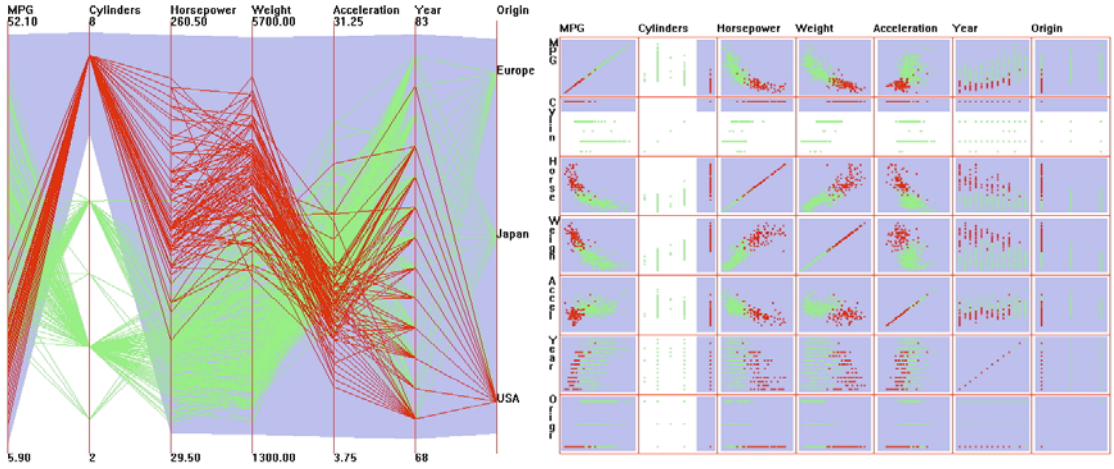

Figure 12: Parallel Coordinate Plot (on the left) and Scatterplot (on the right) taken from Xmdv Tool [114].

Finally, as outlined by [117], another relevant aspect to be taken into account is interaction and direct manipulations of data. As a matter of fact, it results

[7] As a matter of fact, even if theoretically data related to time such as financial data, medical data and so on, should be represented with the same approach, from a practical point of view, each of them requires a dedicated visualization.

crucial for the analysis process, in particular when browsing the time axis and switching between different level of temporal aggregation, e.g., from daily to weekly or monthly view.

All these aspects are relevant when developing or applying visual methods for analyzing time-oriented data, but the diversity of the involved aspects prevented from finding appropriate and more general solutions. For such reason, in the next section we propose some possible categorization schemas, that is, those proposed by Silva and Catarci [118], Müller and Schumann [119], and Aigner et al. [120],[121].

4.2 Categorization Schemas

Silva and Catarci [118] present an overview of the main visual techniques for interactive exploration of time-oriented information. In order to classify temporal data they exploit a framework defined in [17] which was originally used to compare and analyze different temporal data models. Additionally, they consider another classification criterion for time-oriented visualizations, i.e., if the visualization supports so called *snapshot* or *slice views*. This leads to a categorization schema composed of the following four categories:

- *Slice*, corresponding to a visualization of valid history, i.e., a visualization of one or more entities (and their attributes and relationships) valid at discrete (continuous) instants or intervals in a linear order
- *Periodic Slice*, corresponding to a visualization of valid history at specific discrete (continuous) patterns of time (calendar)
- *Multi-slice*[8], corresponding to a visualization of valid history at discrete (continuous) instants or intervals in a branching order
- *Snapshot*, corresponding to a visualization of event history, i.e., a visualization of one or more entities valid at a single discrete (continuous) instant or interval.

[8] Even if Multi-slice visualizations are specified in the categorization, they are not considered in the survey.

According to these categories, more than 20 visualizations are investigated and classified in subcategories (if feasible). In detail, *slice visualizations* are divided into

- *Interactive timeline* (whereas timeline is a graphical or textual display of events in chronological order displayed on a computer screen, which may be directly manipulated by the user, in order to ease the chronological navigation of large temporal data sets) such as Lifelines [122]
- *Innovative timelines* (which enrich timelines with the adoption of distortion techniques [123],[124], modeling features [125],[126],[127], visual metaphors [128],[129],[130], etc. in order to better visualize huge amounts of data on the screen and provide the user with more effective mechanisms to explore such data) such as Perspective Wall [123], Lifestreams [128] or PeopleGarden [129]
- *Visualization of Spatio-Temporal Data* (that is systems, such as geographical information systems, mobile computing, virtual environments, multimedia systems, etc, which allow the interactive visual exploration of video and spatio-temporal data), for instance MMVIS [131], VRML History [132].

Period slice visualizations are divided into

- *Visual calendars* (that is, visualizations giving emphasis to specific periodic patterns, and used e.g., in scheduling applications) such as Visual Scheduler [133], Spiral Calendar [134]and TimeScape [135].
- *Visualization of time-series data* (which explore time-series data) such as Circle Segment Technique [136] and Spiral Display [137]

Finally, no further category is given for *snapshot visualizations*, where SELES [138] is cited as example.

Müller and Schumann [119] give an overview on the visualization of multivariate time-series data and the available techniques. Similarly to the schema proposed by [139], which group visualization techniques into static and animated, they provide three different categories:

- *static representation*, that is, the visualizations does not change over time or is not time-dependent
- *dynamic representation,* that is, the visualizations changes over time or is time-dependent
- *event-based representation,* that is, the visualizations represent events, whereas an event is an occurrence in time signalling a change in data; moreover, differently from the previous two cases, based on discrete or continuous time models, this representations are event-based (i.e., time between following events is usually not constant).

According to these categories, more than 20 visualizations are investigated and classified in subcategories (if feasible). In detail, *static representations* are divided into

- *conventional visualization methods,* that is, "*those visualizations that can be understood as an application of the fundamental insights revealed by Bertin* [140] *and Mackinlay*[141] "such as Napoleon's march [142], Sequence graph [143], Time series graph [143], Point graph [143], Bar graph [143], Parallel Coordinate Plot [116]
- *visualization methods for multivariate data over time* (that is, visualizations where the data element d_i for a given time step t_i covers the data values for more than one variable) such as ThemeRiver [144], Spiral graph [145],[146], Calendar View [147], TimeWheel [148].

Finally, no further category is given for *dynamic representation* and for *event-based representations* (whereas the Metro Emergency application [149] is cited as example).

Aigner et al. [120] propose a systematic view of visualization techniques for time-oriented data, which tries to answer to three practical questions, that is, what are the characteristics of the time axis, what is represented, and how it is represented. Hence, the three corresponding proposed categories (or criteria) are: *time*, *data*, and *representation*.
These categories are then divided into subcategories. In detail, *time* is divided into two sub-criteria (according to the taxonomy proposed by [24]):

- *temporal primitives*, which describe how the time axis is composed. In this case, it can be composed of *time points* (that is, an instant in time) or *time intervals* (specified by two time points or a time point plus a duration)
- *structure of time*, which describes how time can be structured. In this case, three different structures are distinguished, that is, *linear time* (which corresponds to the natural order of time as being a totally or partially ordered collection of temporal primitives, i.e., time proceeds from the past to the future), *cyclic time* (whereas the time axis is composed of a finite set of recurring temporal primitives, e.g., the seasons of the year), and *branching time* (whereas time axes are modeled as graphs allowing alternative scenarios).

The second criterion, *data*, is subdivided into:

- *frame of reference*, that is, the context of the data taken into account. In this case one can have *abstract* data (i.e., data collected in a non spatial context or not per se connected to some spatial layout) and *spatial* data (which contain a spatial layout)
- *number of variables ,* that is, the number of time-dependent variables. In this case, one can have *univariate data* (that is, each temporal primitive is associated with a single data value) or *multivariate data* (that is, each temporal primitive is associated with a multiple data value)

- *level of abstraction*, that is, whether the whole data are shown (*data*) or they are melt down to a condensed form (*data abstraction*).

The third criterion, *representation*, is subdivided into:

- *time dependency*, that is, whether time-oriented data are represented in still images (*static* representations), or using the physical dimension time to convey the time dependency of the data themselves (*dynamic* representations)
- *dimensionality,* that is, whether the data are presented in a *2D* or *3D* space.

Finally, according to these categories, 9 visualizations are classified, such as ThemeRiver [144] (whereas the time axis is composed of *time points* in a *linear* structure of time, the frame of reference is abstract, the data are multivariate, and the representation is static and 2D), Cluster and calendar based visualization [147] (whereas the time axis is composed of *time points* and *time intervals* in a *linear* structure of time, the frame of reference is abstract, the data are univariate, and the representation is static and 2D), and Planning lines [150] (whereas the time axis is composed of t*ime intervals* in a *branching* structure of time, the frame of reference is abstract, the data are univariate, and the representation is static and 2D).

In the upcoming section we will illustrate some examples of information visualization techniques.

4.3 Examples of Visualization Techniques

This section is organized as follows: firstly, we will first present some different types of Information Visualization techniques that were developed for the tasks of revealing trends, patterns, and relationships in time-oriented data and information, so that they leave the process of discovering trends, patterns,

and relationships largely up to the user and the human perceptual system. After that, we present visual query techniques, where queries for time-oriented data and information can be specified graphically in order to retrieve data reflecting the formulated characteristics. Finally, methods which try to jointly apply visual and computational techniques to analyze time-oriented data.

4.3.1 Human-discovery-based Visualization Techniques

The *Spiral Graph* [146] is a visualization method for serial periodic data using an (animated) spiral (see Figure 13). The main intention of the technique is to detect previously unknown periodic behaviour of the data under investigation. It facilitates the detection visually by animating through different period lengths. Periodic behaviour becomes then immediately apparent by the emergence of a structure. When such a structure is spotted, the user stops the animation and a cycle length has been found. The data elements are basically ordered along a 2D spiral starting from the centre and extending to the outside whereas the data values can be mapped to line thickness and colour. Since the screen space is constrained, large datasets might not have enough space available. In this case, a 3D helix can be used for navigation and selection within the dataset (see Figure 13 right). This means that the dataset is mapped onto a 3D helix rather than a spiral and regions of interest can be selected by interactive brushing. The selected region can then be displayed on a 2D Spiral Graph for further investigation.

The visualization technique *TimeHistogram 3D* [151] has been developed in order to give an overview of complex time-varying data in the application context of computational fluid dynamics (CFD). The third spatial dimension is used to represent time. Hence, for each time step a histogram is drawn as a row of cuboids (see Figure 14). Several interactive features such as brushing, scaling, and a 2D context display that is shown on the background of the histogram are part of this technique.

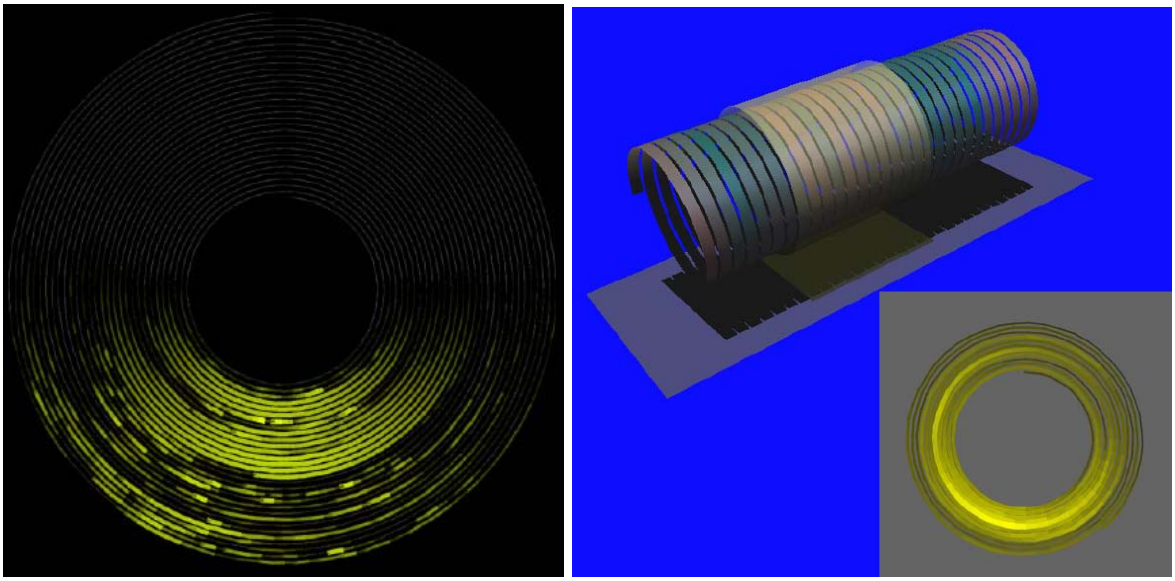

Figure 13: Spiral Graph [146]: Data values are mapped to line thickness and/or color. Animation is used for discovering cycle lengths.

A very well known technique in the Information Visualization community is the *ThemeRiver* [144] which is used to analyze thematic changes of document collections over time. It utilizes the metaphor of a "river" that flows through time. The width of the river changes to reflect thematic changes of topics that are represented as coloured "currents" (see Figure 15). A strong point of the *ThemeRiver* technique is that it allows for a macro view of thematic changes that is hard to perceive otherwise. Basically, the idea of Histograms has been extended by a horizontally centred layout and a continuous, smooth flow in contrast to discrete time steps. Hence, the main focus is directed towards establishing a picture of an easy to follow evolution over time using interpolation and approximation.

A technique for representing repetition in data sets is *Arc Diagrams* [152]. This technique is used to represent complex patterns of repetition in string data. Sequences of repeating substrings are detected and visualized by semi-

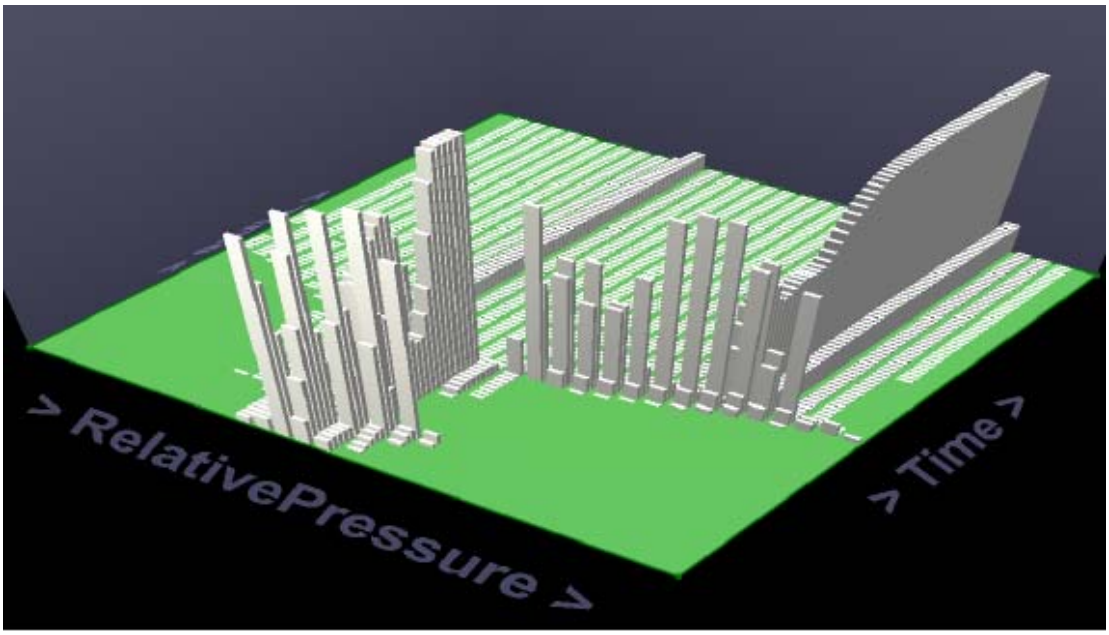

Figure 14: *TimeHistogram3D* [151] uses third dimension to represent time. For each time step a histogram is drawn as a row of cuboids. Example shows the development of pressure over time. One can see where the values change a lot and where a stable condition is reached.

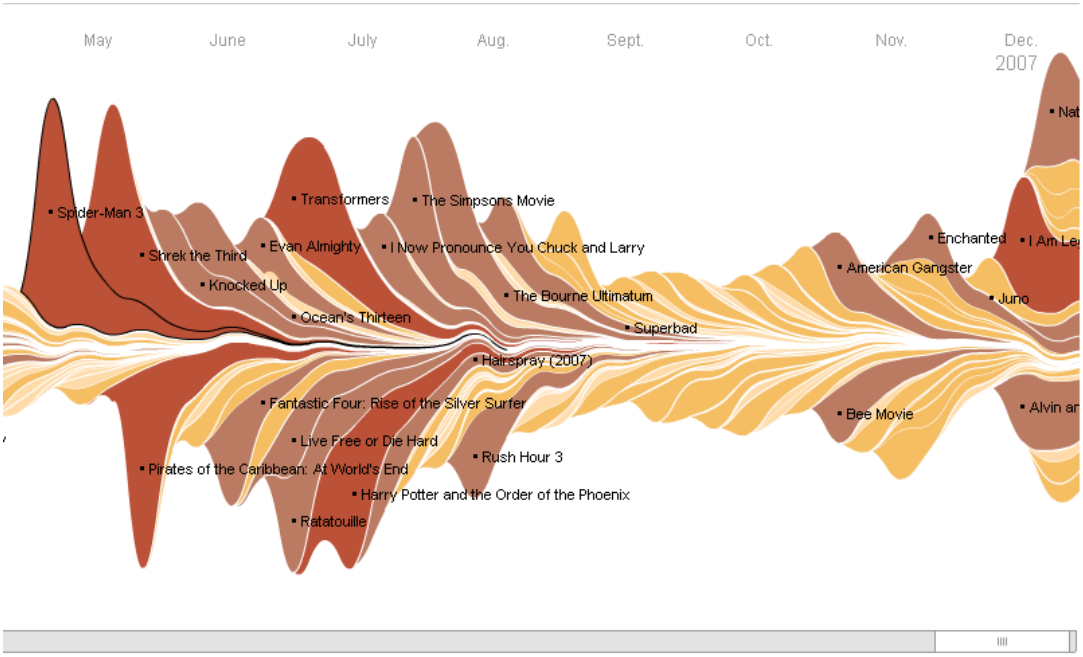

Figure 15: Box Office receipts from 1986 to 2008 visualized using *ThemeRiver* [144]. Summer blockbusters and holiday hits make up the bulk of box office revenue each year, while contenders for the Oscars tend to attract smaller audiences that build over time. Old receipts have been adjusted for inflation.

transparent arcs (see Figure 16). Hence, it is not exactly a technique for time-oriented data, but it could be easily used here as well. The technique has been applied to MIDI music files for showing the basic structure of a certain piece of music, as well as to represent DNA sequences or file structures.

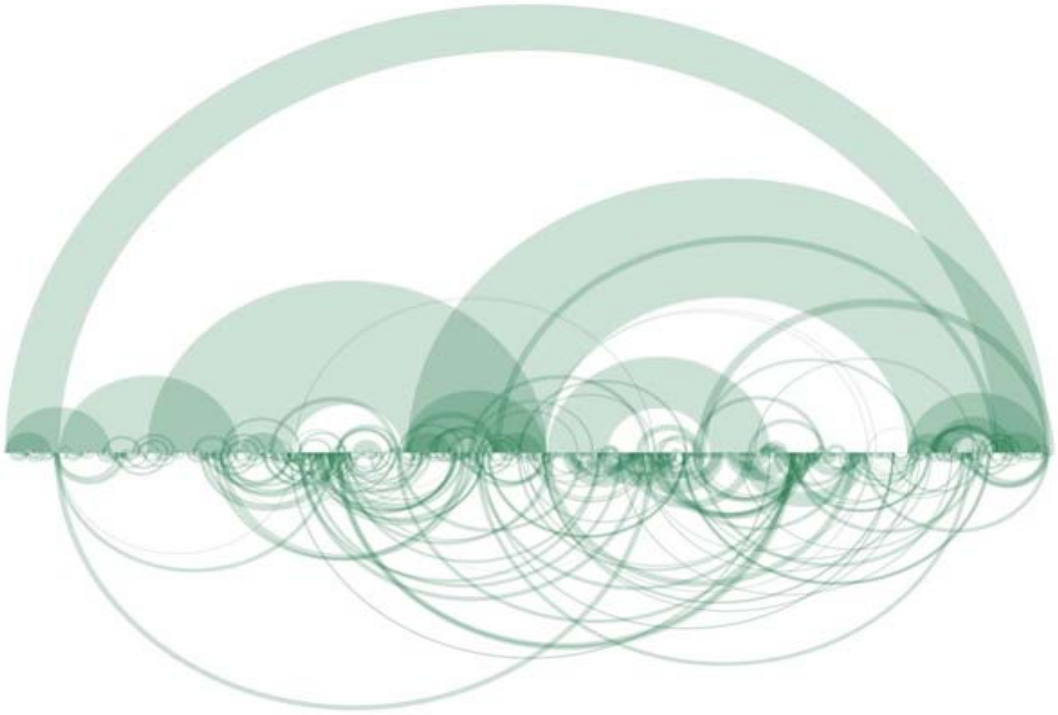

Figure 16: *Arc Diagram* of the Song "Für Elise" [152]. Sequences of repeating substrings are detected and visualized by connecting them via semi-transparent arcs.

A common characteristic of the techniques presented so far is their relatively large space consumption which limits the amount of data that can be represented at once. To overcome this, *pixel-oriented* visualization techniques (see Figure 17 left) have been developed to represent huge amounts of data on a single display in order to explore overall trends and patterns rather than specific details or single values. The basic idea of *pixel-oriented* techniques is to map each attribute value to a coloured pixel and to present the data values belonging to one attribute in separate windows. This implies that in general only one data variable can be shown in a single view. The probably best known technique is the *Recursive Pattern Technique* [153] (Figure 17 right) where

value mapping is done by colour coding and different patterns are used for the arrangement of pixels. The application of different pixel arrangements leads to the possibility of discovering patterns in the data that would be hard to detect otherwise.

Even though pixel-oriented techniques allow for the representation of large amounts of data by limiting each data value to a single pixel, the amount of data that can be depicted is still constrained by the limited resolution of current display technology. Facing the huge volumes of data to be analysed today, purely visual techniques alone are not sufficient.

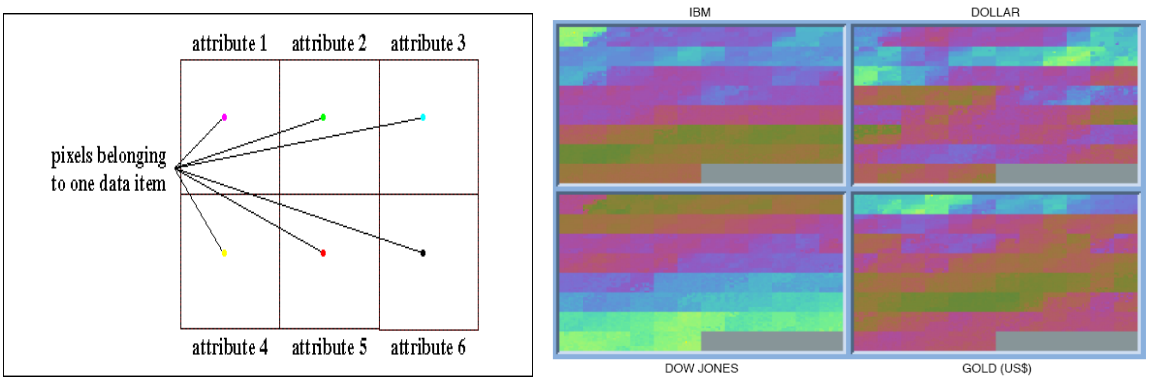

Figure 17: A representation of data with six attributes using a *pixel-oriented* technique (left). Recursive Pattern Technique [153] (right), an example showing stock price data.

4.3.2 Visual Query

Visual Query Systems are applied for querying data repositories using visual representations to formulate data characteristics of interest as well as the results of the query itself. This implies that at first, data patterns of interest are formulated visually by the user. Following this, the query is processed and the results are again presented in a visual form to the user.

TimeSearcher is a powerful example of such a system [154]. Its goal is to identify and find familiar patterns in the investigated data (see Figure 18).

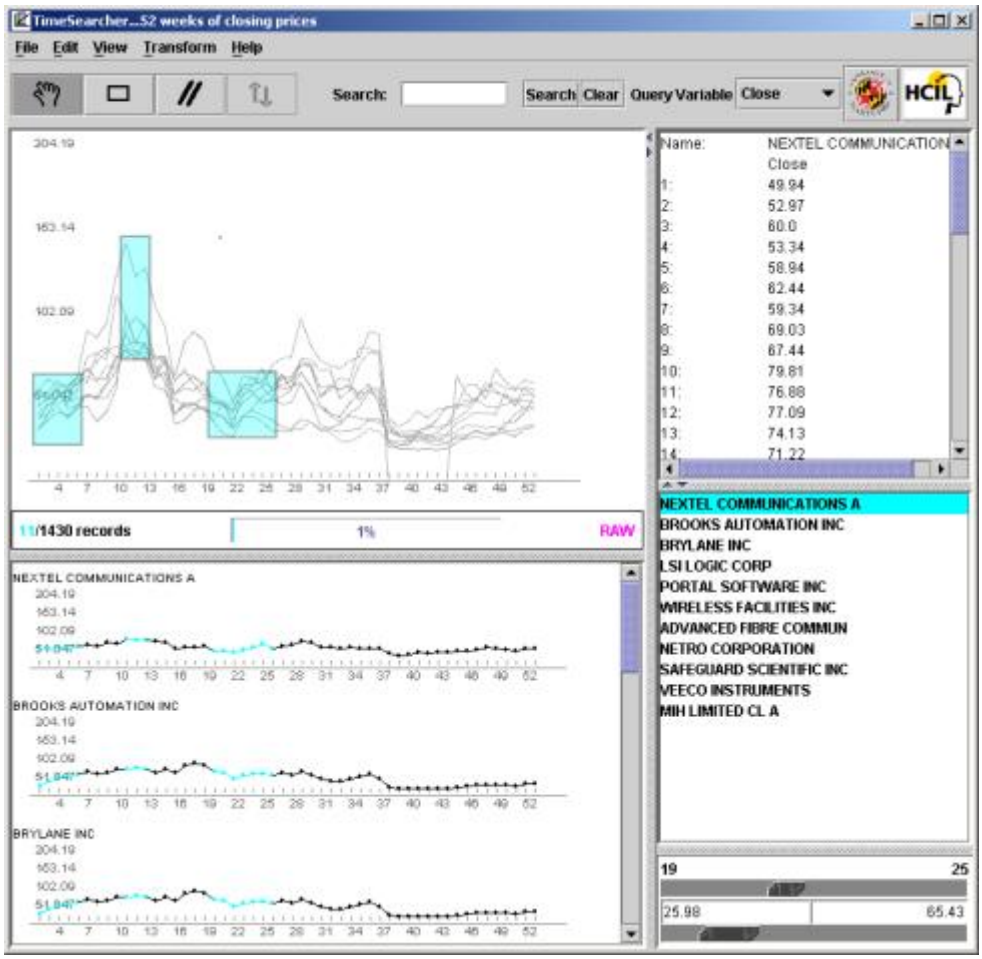

Figure 18: *TimeSearcher* [154] – Visualization and exploration tool for multiple time-series. The above example shows stock price data: individual time-series plots (bottom left), timebox queries (top left), details of selected stock (top right), list of available stocks (middle right), timesliders (bottom right).

In order to achieve that, so-called "timebox query model" allows for the specification of a rectangular query region that defines both, a time period and value range of interest. All time-series that comply with this query are shown whereas all others are filtered out. Moreover, multiple timeboxes can be combined to refine the query further. Additionally, other query functionality such as "leaders and "laggers", "angular queries", and "variable timeboxes" are part of the *TimeSearcher* technique. Stock price analysis and DNA microarray experiments are application areas where this technique has been used.

Paint Strips [155] are an example of visual query technique that supports more sophisticated queries in time including temporal uncertainties (Figure 19). They have been developed for visualizing queries on medical databases.

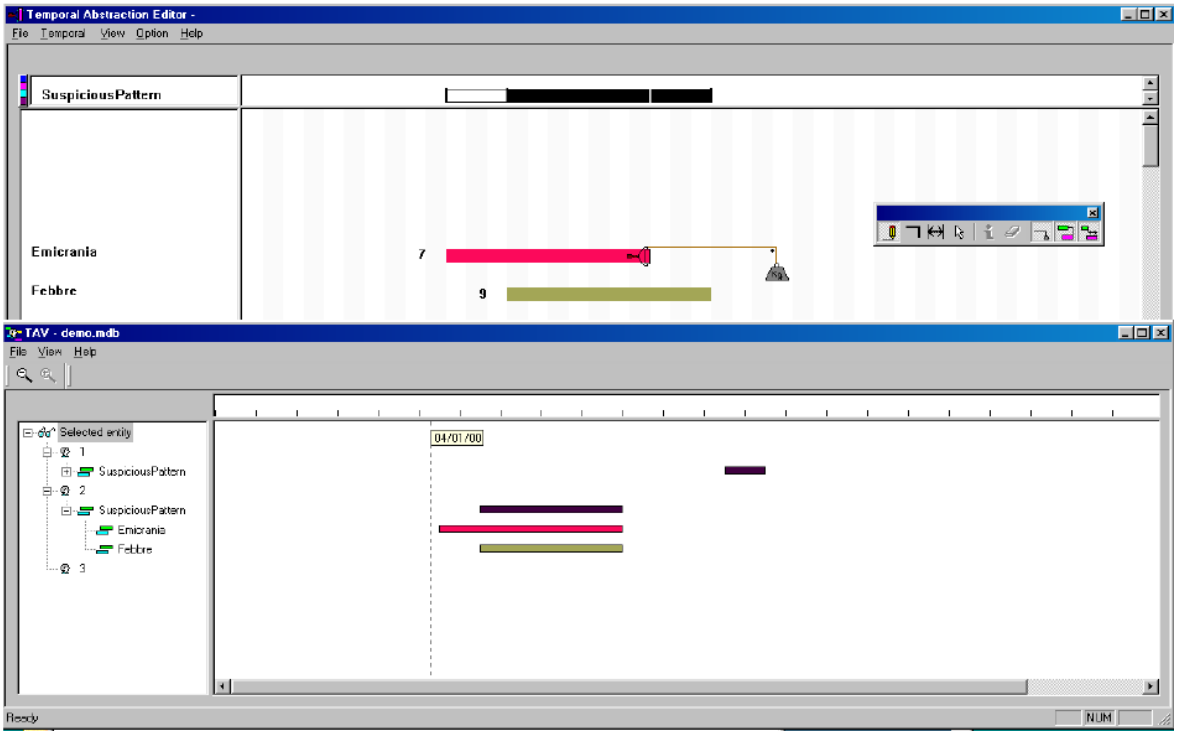

Figure 19: *Paint Strips* [155] – A paint roller at an end of a bar means that this line can expand by moving the roller until a wall is reached. The relationship of Paint Strips can be fixed, which means that if one strip moves, the other one moves in the same extent as well. This relationship is indicated graphically by connecting the involved paint rollers and attaching them to a weight at the end of a "rope" which is able to move the rollers.

The idea of simple timelines is enriched by a painting metaphor which indicates that the displayed bars are drawn by a paint roller. A paint roller at the beginning or end of a bar means that this line can expand by moving the roller until a wall is reached. This way, the maximum duration and earliest start or latest end, depending on which end of the painting strip the paint rollers are attached to, are defined and indeterminacies are shown. Another addition is the possibility to combine strips. The relationship of *Paint Strips*

can be fixed, which means that if one strip moves, the other one moves in the same extent as well. This relationship is indicated graphically by connecting the involved paint rollers and attaching them to a weight at the end of a "rope" which is able to move the rollers.

A common characteristic of visual query techniques is the fact that patterns to be searched for have to be (pre)defined by the user. This means that the user has to have quite clear hypotheses about what kind of patterns might be present in the data. This implies that the detection of new, previously unknown, or unexpected patterns is much harder and requires great effort. To tackle this issue, combined methods may be often used which support the discovery of such previously unknown patterns.

4.3.3 Combined Methods

As the application of Data Mining to time series databases is a relatively new field [156], the combination of automatic Data Mining techniques, time series and visualization, appears as even newer. Some contributions give more emphasis to time series and Data Mining ([157],[158]), others also concentrate on the visualization part (*VizTree* [159], see Figure 20, *ThemeRiver* [144], see Figure 15), or propose a combination of visual Data Mining and time series (*Parallel Bar Chart* [160], see Figure 21), or combine KDD concepts and visualizations (*Statigrafix* [161]). However, each of them has some lack in the visualization part, or leaves the mining task up to the user, or requires a strong expertise in the application field.

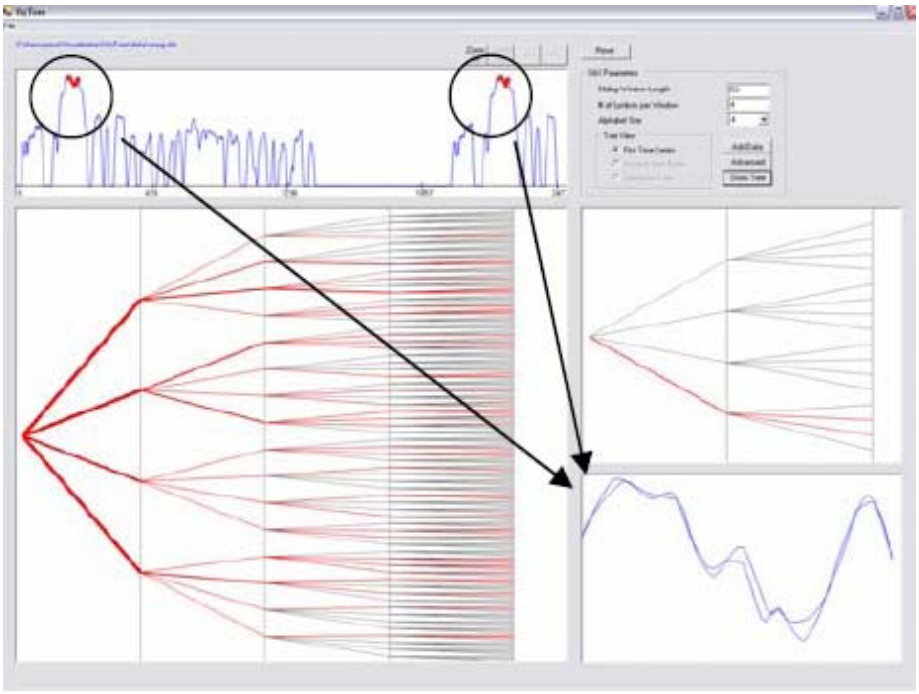

Figure 20 *Viztree* [159] – The top panel is the input time series. The bottom left panel shows the subsequence tree for the time series. On the right, the very top is the parameter setting area. Next to the subsequence tree panel, the top window shows the zoom-in of the tree, and the bottom window plots the actual subsequences when the user clicks on a branch.

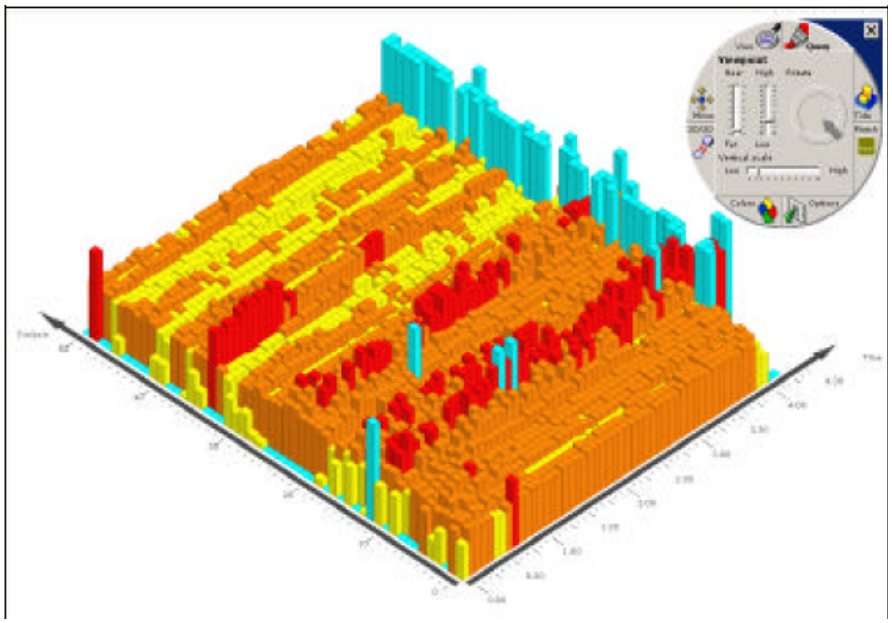

Figure 21: *Parallel Bar Chart* [160] – visually represents each time-series in a bar chart format where the X axis is associated with time (the axis on the right), the Y axis with the value (height of a bar) of the series at that time and the axis on the left identifies the different time-series, ordered by date.

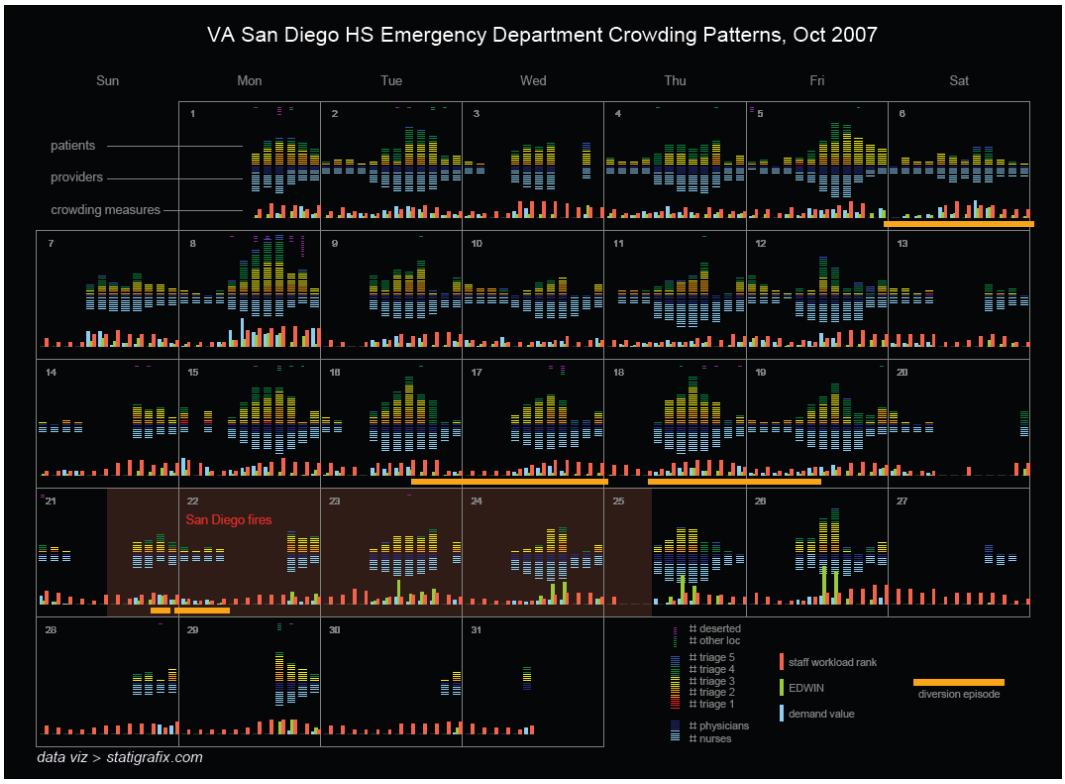

Figure 22: Calendar-template data visualization of datasets captured at the VA San Diego Health Services' Emergency Dept in Oct 2007 [161].

4.4 Discussion

Time is an exceptional data dimension due to its importance and unique influence. Thus special visual methods are needed for dealing with time in data sets in order to present, analyse and explore time-oriented information. For such a reason, we firstly motivated which aspects are relevant when visualizing time-oriented data, then we presented three different possible categorization schemas, and we concluded with some examples of Information Visualization techniques.

In general, the categorization proposed by Silva and Catarci [118] classifies temporal data by enhancing a framework defined in [17] with another classification criterion for time-oriented visualizations. It considers interaction, data, time model, but only partly task aspects are considered. The definition of time-oriented information adopted by Müller and Schumann [119] is more constrained compared to the previous one, since it only considers data

with a unique mapping from time to data element. The visualization techniques used as example are described ad-hoc and frequently mentioned very briefly. Hence, no uniform characterization of the investigated techniques is given. The most recent categorization proposed by Aigner et al. [120],[121] considers time, data, and representation as classifying categories, but differently from the previous ones, they outline the relevance of some aspects. In particular, the importance of interaction when dealing with time-oriented data and the role of the user when interacting directly with the visual representation, whereas these aspects are usually neglected or partially considered. Then, the relevance of multiple views, that is, linking several views together in order to extend the applicability and usefulness of visual methods. Furthermore, the need to include also analytical methods in the process of Knowledge Discovery, along with the visual ones.

Finally, following in particular the last aspects, in order to conclude this part of our state of the art, we illustrated a brief overview of Information Visualization techniques. We started from techniques leaving the discovery process mainly to humans, then we presented visual query techniques, and ended with methods which try to jointly apply visual and computational methods to analyze time-oriented data, and which seems the most promising way to deal with huge amount of temporal data. As a matter of fact, we are convinced that in order to fully support the Knowledge Discovery process, visual methods for analyzing time-oriented data should take Keim's Visual Analytics mantra [163] into account:

"*Analyse first, Show the Important, Zoom, filter and analyse further, Details on demand.*"

5 State of the Art: Discussion

"Discussion is just a tool. You have to aim; the final goal must be a decision."

Harri Holkeri

In the previous parts of this thesis, we analyzed the problem domain thoroughly. We showed that time is different from any other quantitative data dimension and has an inherent structure to be taken into account. Moreover, we illustrated that Temporal Data Mining methods dealing with the issue of time interval tend to neglect the temporal information between sequences of events. Besides that, visualization is an important tool for presenting, analyzing, and exploring data in general, and time-oriented information in particular. Thus we described some categorization schemas and an overview of information visualization techniques.

In detail, we started the current state of the art by investigating two formal time models from temporal database research, a framework for time granularities, and an example of a system able to manage multiple calendars as well as operations on them. The first two are used to model temporal information in databases efficiently, but they respectively do not support indeterminacies and granularities, or use a fixed calendar and limit the possibility of flexible primitive specifications. The last two focus on calendar algebra and calendars from a more theoretical and from a more practical point of view, but are not fit to represent the most frequently occurring patterns preserving the temporal information in between. Therefore, the time models at hand do not fully comply with the prerequisites needed for discovering multi-time interval patterns in social time context, visualizing them and defining as a matter of fact a visual analytic approach.

After that, we proceeded with the state of the art by investigating Temporal Data Mining methods dealing with the issue of time interval, focusing in particular on proposals extending the Apriori algorithm (Chen et al.[5], Yang [6], and Hu et al. [94]), those annotating sequential patterns (Yoshida et al. [90], Vautier et al. [91], and Giannotti et al. [92]) and the proposal from Magnusson [89]. The first group, propose an Apriori algorithm with intervals to be applied to transactional databases or extensions of such algorithm (called MI-Apriori and MI-PrefixSpan [94]). Such intervals are hidden to users and defined in order to have five equally deep intervals each of which contains roughly the same number of data. Moreover, to reduce the number of found patterns, they provide three pruning strategies, i.e., descending property, the time interval information matrix and the downward-closure property. The second group face the problem of mining sequential patterns by annotating sequential patterns with a frequent transaction time derived from the source data. Delta patterns [90], chronicles [91] and temporal annotated sequences [92] are proposed to find frequent transaction time between two itemsets and then usually the results is clustered (e.g., with CF-tree in the case of delta patterns) to find the representative elements and possibly calculate those above the support.

The last one proposes a bottom up, breadth-first detection strategy to find so called T-patterns in human behavioural components, where the interval taken into account (the so called *critical intervals*) are intervals that tend to contain at least one more occurrence of a pattern than would be expected by chance. Differently from them, we deal neither with transactional databases, nor with temporal annotations or human behavioural components, but rather we want to provide a more general as well as more customizable approach in order to find *multi-time interval patterns* in time-oriented data. Moreover, our approach gives the user the possibility to be directly involved in the analysis, by choosing and adjusting parameters, and to interact with a visual

representation of events and patterns. For such reasons, we could not apply directly such methods in this work[9].

The last part of the current state of the art was devoted to visualizations of time-oriented data. What can be discerned globally from this survey is that most of visualization techniques are very specialized and tailored to a particular model of time or application area. Moreover, many are the aspects to be taken into account when dealing with time-orient data As a matter of fact, the three categorization schemas we investigated, outlined in particular the importance of data, time, and representation, using them as classification criteria for the visualizations. Furthermore, another conclusion that can be drawn is that the role of the user and the role of interaction are particularly crucial when dealing with time-oriented data, as well as are relevant the use of multiple views and the need to include also analytical methods in the process of Knowledge Discovery, along with the visual ones.

Therefore, following these considerations, we aim at proposing a visual analytics approach involving the user in any steps of the Knowledge Discovery process, using interaction with any visual element, which is able to intertwinedly combine visual and analytical methods and, ultimately is able to discover multi-time interval patterns preserving the temporal information in between. To this aim, in the upcoming sections, we will provide a complete description of the proposed approach as well as the designed visualization.

[9] For a more detailed comparison see Section 6.5.

6 The Proposed Approach

"Happiness is the longing for repeating patterns"

Milan Kundera, The Unbearable Lightness of Being

As it was outlined in the previous parts, time is more than a flat sequence of consecutive moments. It contains not only instant (intended as start time of something, end time of something) and span (duration), but also rich complex, and partly irregular structures (e.g., day of week, time of day, year, holidays, etc.). Yet, the structural elements of time are mostly not strictly hierarchical, regular, cyclic, or are not completely defined for the whole time domain (e.g., we can have 28, 29, 30, or 31 days per month; 365 or 366 days per year; holidays are arbitrary, etc.). Moreover, individual elements within one granularity can be different from each other (e.g., the day of the week Friday often is different from Monday) and they can be also influenced by social and natural contexts (e.g., Christian Easter, which is not only related to Christian religion, but it is also a shifting date[10]) .

In order to analyze time related data many Temporal Data Mining methods exist in the literature. They usually define an event as either a certain value for a given attribute occurring at a certain time (e.g., 30 employees were at work at 4 p.m.) or a fact or sequence of facts following the time line (e.g., first fact: customer X buys a book on 18th April; second fact: customer X buys a DVD one week later).

However, except for few exceptions [5],[6],[90],[91],[92],[94],[89], a sequence of events in a certain order is discovered, but neither with any knowledge about the intervals between successive events, nor about when this sequence will

[10] Ecclesiastical rules state that Easter falls on the first Sunday following the first ecclesiastical full moon that occurs on or after the day of the vernal equinox, whereas this particular "ecclesiastical" full moon is the 14th day of a tabular lunation (new moon) and the vernal equinox is fixed as 21st March. As a result, Easter can never occur before 22nd March or later than 25th April.

reoccur. That is, given an ordered list of item sets and their transaction times, they search for the most frequently occurring patterns, but the result they provide is a sequence of items, without any information about the elapsed time between two items and above all, about the amount of time before a certain sequence will reoccur.

This situation is illustrated in Figure 23.

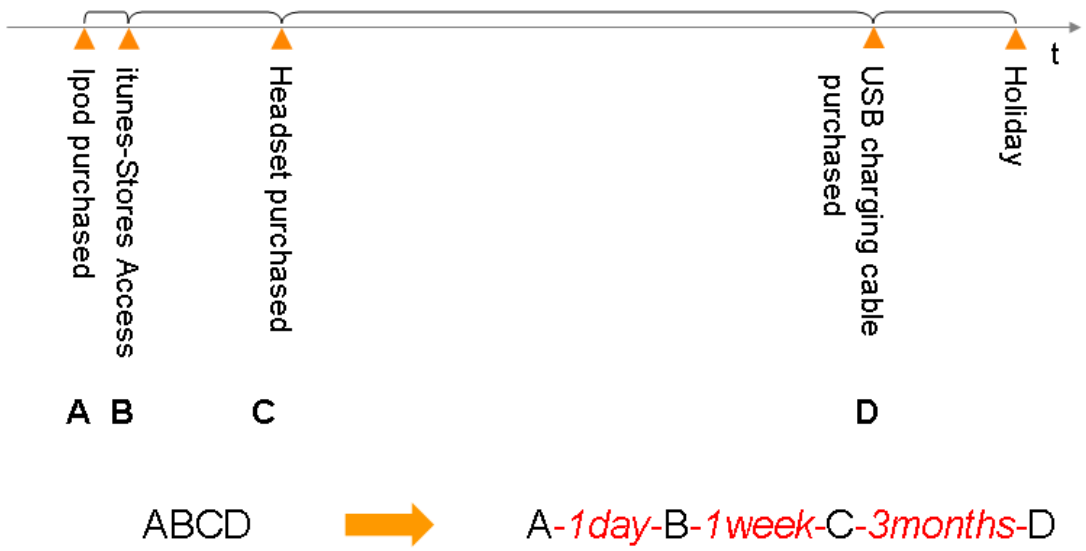

Figure 23: Left: Traditional sequence pattern (order only). Right: Multi-time interval pattern (intervals between events).

A customer buys an iPod (first event); then, within 1 day s/he registers into the iTunes Store (second event). After a week, s/he purchases a headset (third event). Finally, a week before going on holiday, s/he buys a USB charging cable (fourth event). If one takes into account only the sequence or the order of these events, that is, "iPod purchased - iTunes Store access - Headset purchase - USB charging cable purchased" (or ABCD), it is impossible to know after how much time the next item will be purchased. Moreover, we cannot even know after how much time a similar sequence will occur. On the contrary, if also the time intervals in between are considered, we can not only profile the users according to their interests, habits and requirements, but we can also improve the selling strategies according to the timing of their shopping habits. As a

matter of fact, the shopping mall can vary its offers and catalogues according to the users. For instance, it is possible to send e-mails or letters describing discounts on iPod accessories three months after the first purchase, or make special offers before summer holidays, dedicated to those who bought an iPod in the previous three/four months.

Thus, to analyze and ultimately visualize time related data correctly, we have not only to take into account the rich structural complexity of time, but also to find a way to discover temporal patterns preserving the temporal information in between (the so called "*multi-time interval patterns*"). To this aim, we will follow the same schema adopted for the state of the art (Figure 2). Firstly we describe the new time model which will be adopted in our approach and provide a formal definition of multi-time interval pattern (Section 6.1 and 6.2). Then the approach will be explained in details (Section 6.3). After that, we will provide some examples and a discussion, as well as how to customize the proposed approach and how it may be extended. Then we will present a two-folded validation for the proposed approach (Section 7). Finally Section 8.1 and 8.2 will be devoted to explain how to design a proper visualization framework, which allows not only for presenting an overview of the available data and the results, but also letting the user easily adjust the parameters as well as interact with each visual component, during each step of the approach

6.1 The New Time Model (and Granularities)

In order to consider the rich structure of time and to model it properly, on the one hand we have to identify what are the properties of time (also determined by social and natural contexts) that may be considered. Using a lattice structure, we can represent such properties and how they related with one another. Figure 24 shows a possible lattice structure of the set of time properties we took into account. A part from the most common ones, such as *hour*, *day*, *and week* also more particular properties of time are considered. For instance, *day of week* represents the day of the week during which a certain event occurs (e.g., Monday), and therefore it is a properties of time related to

day; similarly *holiday* represents a properties of time related to *day* (e.g., a day during which one releases from work). Moreover, other time properties could be added to this structure, and the model should be open to this possibility.

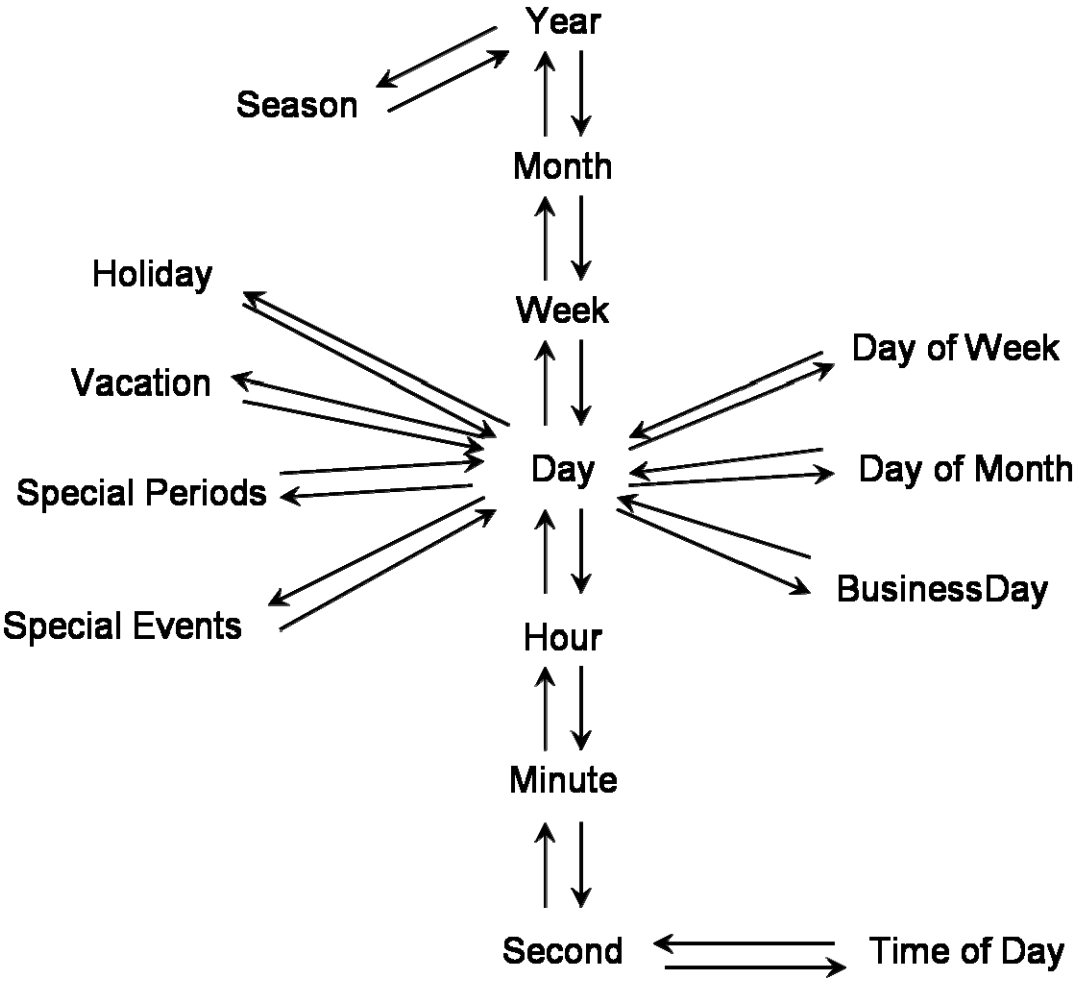

Figure 24: A possible lattice structure of the set of time properties we took into account.

On the other hand we need to model such properties so that we are able to discover temporal patterns preserving the temporal information in between. Hence, we can represent each property of time (or granularity[11]) as set of intervals, defined by a begin time point and an end time point, and one or more

[11] In the following we will use the terms property of time and granularity interchangeably.

labels to represent other properties (such as *day of week*, *day of month* and so on). Moreover, we add the constraints that (1) the intervals (to define the granularity) are not overlapping and (2) (the values of) labels do not change within the interval.

For instance, the granularity *day* is represented as follows:

Begin	**End**	**DoW**	**DoM**	**…**	**…**
01.06.2006 00:00:00	02.06.2006 00:00:00	Thu	1		
02.06.2006 00:00:00	03.06.2006 00:00:00	Fri	2		
…	…	…	…		

Table 1: The granularity *day* represented as set of intervals. The intervals are defined by "Begin and "End". Moreover, one or more labels are adopted to represent other time properties, such as *day of week* (DoW, which assumes values from Monday to Sunday), *day of month* (DoM, which assumes values from 1 to 31), or *Holiday* (which assumes values *true* or *false*).

Similarly hour and week can be represented as:

Begin	**End**	**BusinessHour**	...
01.06.2006 00:00:00	01.06.2006 01:00:00	F	
01.06.2006 01:00:00	01.06.2006 02:00:00	F	
…	…	…	

Table 2: The granularity *hour* represented as set of intervals. The intervals are defined by "Begin and "End". Moreover, one or more labels are adopted to represent other time properties, such as *BusinessHour* (which assumes values *true* or *false*).

In this way we are able to represent some of the time properties shown in the lattice structure as set of intervals (namely, the granularities *second*, *minute*,

hour, *day*, *week*, *month*, *year*), whereas others are represented as labels. Moreover, interval relations and operators are allowed. As a matter of fact, Allen's basic interval relations [21] are part of the temporal model, and operators such as *union*, *difference,* and do so on, permit for instance to define new granularities, e.g., *businessweek* as union of all *businessday* of a week (whereas *businessday* is the union of the *businesshour* of a day).

6.2 Definition of Multi-time Interval Pattern

As it has been outlined in the related state of the art, there are several definitions of the term "pattern" in literature. In order to better address our issues, we propose to define a pattern as

"a nonempty set of time intervals (with properties) or time-points, their relationship (as well as to corresponding data), meeting a certain degree of interest",

where the *degree of interest* is a measure of interestingness, which may be context or user dependent (e.g., in case a user is interested in knowing when a higher amount of employees is needed during the week, s/he will probably search for the most frequent patterns; on the contrary, for surveillance systems data, the less frequent ones will likely be of more interest).
Then, as the name suggests, a *multi-time interval pattern* can be defined as a pattern whose composing parts are divided by different time intervals, that is,

$$(1) \quad Cp_1\ I_a\ Cp_2\ I_b\ Cp_3\ \ldots$$

whereas the Cp_i represent the "composing parts" of the multi-time interval patterns and the I_j are the time intervals in between.

Given such definitions, we can apply them to several cases and to different kinds of data. In particular, our focus is on data which contain temporal and

non temporal attributes (or columns, if we represent them as a table), which may have both ordinal and nominal (categorical) values, and are expressed as collection of non overlapping sets.

Table 3 shows an example of dataset: in this case the data are about turnover (label "*Turnover*") and the number of employees (label "*#E*") present in a certain retail store. They are expressed at a daily level, that is, with a granularity *Day*. The labels "*Begin*" and "*End*" represent the two endpoints of each interval (that is, of each day), and finally the labels "*DoW*" and "*DoM*" represent "*Day of the Week*" and "*Day of the Month*" respectively. Moreover, we can have nominal attributes, such as "*Weather*", with values e.g., "*sunny*", "*rainy*", and so on.

Begin	End	DoW	DoM	#E	Turnover (€)
01.06.2006 00:00	02.06.2006 00:00	Thu	1	24,00	3.537,00
02.06.2006 00:00	03.06.2006 00:00	Fri	2	24,00	3.493,00
03.06.2006 00:00	04.06.2006 00:00	Sat	3	10,00	535,30
04.06.2006 00:00	05.06.2006 00:00	Sun	4	0,00	0,00
05.06.2006 00:00	06.06.2006 00:00	Mon	5	0,00	0,00
06.06.2006 00:00	07.06.2006 00:00	Tue	6	24,00	3.483,50
07.06.2006 00:00	08.06.2006 00:00	Wed	7	25,00	3.808,70
08.06.2006 00:00	09.06.2006 00:00	Thu	8	24,00	3.870,70
09.06.2006 00:00	10.06.2006 00:00	Fri	9	23,00	3.396,40
10.06.2006 00:00	11.06.2006 00:00	Sat	10	12,00	660,00
11.06.2006 00:00	12.06.2006 00:00	Sun	11	0,00	0,00
12.06.2006 00:00	13.06.2006 00:00	Mon	12	23,00	3.732,40
13.06.2006 00:00	14.06.2006 00:00	Tue	13	24,00	3.659,30
14.06.2006 00:00	15.06.2006 00:00	Wed	14	24,00	3.870,70
…	…	…	…	…	…

Table 3: An example of dataset: data are about turnover and number of employees in a certain shopping mall expressed day by day, with information about day of week and day of month.

6.3 Description of the Approach

As outlined in the previous sections, there have been some papers presenting a solution to find at least sequential patterns with time intervals, named "the time-interval sequential pattern". The approach we propose extends them (in particular the I-Apriori algorithm proposed in [5]) to non transactional databases and tries to provide a more general as well as more customizable approach in order to find *multi-time interval patterns* in time-oriented data.

Before starting with the description of the approach, we need to define some other concepts. In order to find interesting multi-time interval patterns in a dataset, we have to reprise the definition (1) and describe more precisely what the "composing parts" of a multi-time interval pattern are and what intervals are taken into account.
Firstly, the "composing parts" represent the occurrence of a certain value for a given attribute, or the occurrence of a set of values for a given set of attributes. In other words, each time a given attribute assumes a certain value (or a given set of attributes assume a set of values), we have an occurrence of an *event*.
We follow the definition given by Mannila et al. [110] that define the event as a pair *(event type, time of occurrence)*, where the event type can actually contain several attributes. Therefore we can state that each "composing part" of a multi-time interval pattern is an *event* and it can be defined as follows:

- *Case 1: An event is the occurrence of the value of a certain attribute.* For example, if the attribute chosen is "*#Employee*", then event e_0 is 24, event e_1 is 10, event e_2 is 0 and so on (note that on June 8th, 24 Employees were at work, this is another occurrence of event e_0).

- *Case 2: An event is the occurrence of the values of a pair of attributes.* For example, if the attributes chosen are "*#Employee*" and "*Turnover*", then

event e_0 is (#E=24 AND T=3537), event e_1 is (#E=24 AND T=3493) and so on.

- *Case 3: An event is the occurrence of the values of more than two attributes.*

Table 4 shows the situation in case of single attribute events, #E (Case 1).

Begin	**End**	**DoW**	**DoM**	**#E**	**Turnover (€)**	**…**	**…**
01.06.2006 00:00	02.06.2006 00:00	Thu	1	24	3.537,00		e_0
02.06.2006 00:00	03.06.2006 00:00	Fri	2	24	3.493,00		e_0
03.06.2006 00:00	04.06.2006 00:00	Sat	3	10	535,30		e_1
04.06.2006 00:00	05.06.2006 00:00	Sun	4	0	0,00		e_2
05.06.2006 00:00	06.06.2006 00:00	Mon	5	0	0,00		e_2
06.06.2006 00:00	07.06.2006 00:00	Tue	6	24	3.483,50		e_0
07.06.2006 00:00	08.06.2006 00:00	Wed	7	25	3.808,70		e_3
08.06.2006 00:00	09.06.2006 00:00	Thu	8	24	3.870,70		e_0
09.06.2006 00:00	10.06.2006 00:00	Fri	9	23	3.396,40		e_4
10.06.2006 00:00	11.06.2006 00:00	Sat	10	12	660,00		e_5
11.06.2006 00:00	12.06.2006 00:00	Sun	11	0	0,00		e_0
12.06.2006 00:00	13.06.2006 00:00	Mon	12	23	3.732,40		e_2
13.06.2006 00:00	14.06.2006 00:00	Tue	13	24	3.659,30		e_0
14.06.2006 00:00	15.06.2006 00:00	Wed	14	24	3.870,70		e_0
…	…	…	…	…	…	…	

Table 4: Creation of events in case of choice of a single attribute.

Secondly, in order to search for the most frequent multi-time interval patterns, we need to define the intervals within which to look for the patterns. Since it would be too time consuming to calculate all the possible intervals between

any pair of events, some intervals are given as default[12]. We have defined six different not overlapping and consecutive intervals (or ranges), taking into account the most meaningful granularities of social time, according to our experience and to the suggestions coming from experts collaborating in a common project, which will be described in Section 6.4.2 (see Figure 25 and Figure 26). The first interval spans from 0 till 1 hour, the second one from 1 hour till 1 day, the third one from 1 day to 1 week and so on. Applying these intervals, we are able to discover multi time interval patterns.

Interval	
I_o	$0 \leq T < 1$ hour
I_1	1hour $\leq T < 1$ day
I_2	1 day $\leq T < 1$ week
I_3	1 week$\leq T < 1$ month
I_4	1 month $\leq T < 1$ quarter
I_5	$T \geq 1$ quarter

Figure 25: The chosen (default) time intervals.

Thirdly, we have to define the *support*. We adopt the classical definition, that is, it represents the relative frequency or number of times a value occurs within the database (usually expressed as percentage).

Figure 26: The default time intervals as they graphically appear.

Therefore, before proceeding with the description of the approach, we can now revise the definition of *multi-time interval pattern* in a generalized form as follows:

[12] The amount of interval and their end points can be customized as it will be explained at end of this section. However for simplicity we consider the default case without any loss of validity.

(2) Be D a dataset of dimension $p \; x \; t$ with attributes $A_1,...,A_t$, and values $V_{A1,1}$, $V_{A1,2},...V_{A1,p}$, $V_{A2,1,...}$ $V_{At,p}$,E the set of possible events, having the form $A_h = V_{Ah,1} \wedge A_{h+1} = V_{Ah+1,1} \wedge ...$, $1 \leq h \leq t$, and TI the set of all the possible time intervals between any pair of events, then $M = e_0 \; I_0 \; e_1 \; I_1 \; e_z \; I_2 ... \; e_m \; I_n$ is a *multi-time interval pattern* if

- $e_j \in E$ for $1 \leq j \leq m$
- $I_k \in TI$ for $1 \leq k \leq n$
- $k \leq w \Rightarrow I_k$ (consecutive) precedes I_w
- $e_j \; I_b \; e_q \; I_c \; e_s \; \neg \Rightarrow \; I_b$ (consecutive) precedes I_c

In other words, a multi-time interval pattern can be defined as a pattern whose composing parts are events as described above, divided by non overlapping and consecutive time intervals. Moreover, any time interval in between can occur and repeat independently of its consecutive order. That is, if we compare the indexes of the intervals in between, I_2 (consecutive) precedes I_3, but the order the intervals appear does not imply the former precedes the latter (e.g., $e_1 \; I_2 \; e_2 \; I_1 \; e_2$ is a possible multi-time interval pattern and then does not imply that I_2 precedes I_1).

Figure 27 illustrates how the proposed approach works. Some input must be provided: first the support (and whether the patterns have to be greater or greater than the given support threshold); second, the attributes chosen, that is, column(s) of interest, which will be used to create the events. Moreover, as it will be explained below, if the values of one or more columns are too spread, they can be grouped (*discretized*) into a limited number of classes.

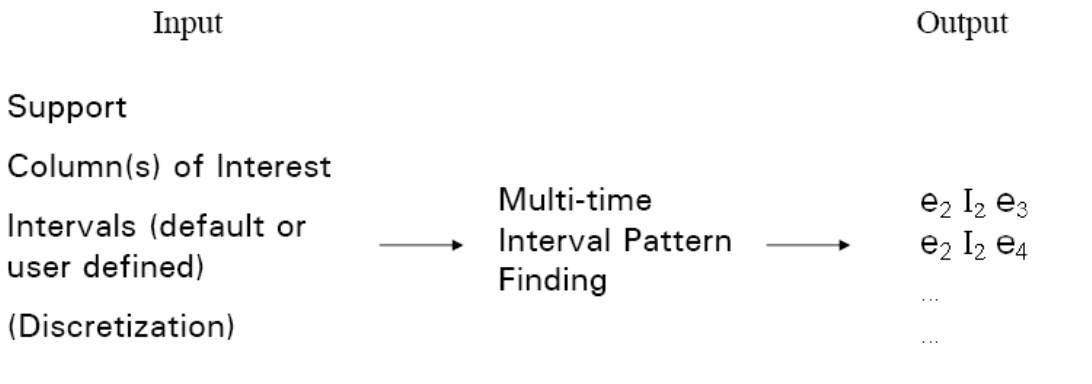

Figure 27: Input and Output of the proposed approach.

Given such inputs, and the intervals of Figure 25, we proceed as follows:

Step 0 (selection of single events):

- Select all single events whose support is > (≥) the support given by the user

Step 1 (patterns composed of two events):

- Starting from the selected single events, search for all possible patterns according to given ranges I_o, I_1, I_2…
- Count all the patterns at this step and select those patterns whose support is > (≥) the support given by the user

…

Step *i* (patterns composed of *i+1 events)*:

- Starting from the selected patterns composed of *i* events, search for all possible patterns according to given ranges I_o, I_1, I_2…
- Count all the patterns at this step and select those patterns whose support is > (≥) user's support

Stop Condition: no more new patterns are found

The output is a list of multi-time interval patterns having the form "*Event Interval Event Interval Event…*" as in definition (2). Moreover, according to the definition of *pattern* and *degree of interest* given in Section 6.2, we are able to find the most frequent patterns as well as the less frequent ones. However, due to the *anti-monotonocity property*[13], the less frequent patterns are "local" to each step and no further patterns can be found starting from them.
Beside that, as it was previously outlined, our approach allows to customize according to users' or tasks requirements, both in term of their amount and in term of their end points. As Figure 28 illustrates, different granularities are currently allowed to define the intervals: milliseconds, seconds, minutes, hours, days, weeks, months, quarters and years. In this way we can allow a more flexible use of human time and we are able to perform several kind of analysis and to discover more interesting patterns.

Finally, note that the values of a certain attribute may be too spread (as for the attribute "Turnover" in Table 3). This can lead to the creation of too many different events, or to consider two events like (#E=24 AND T=3537) and (#E=24 AND T=3493) as different, whereby for most of the analysis they could be considered the same one. Therefore, a *discretization step* (or classification step) may bring some benefits. At the moment two alternatives are provided if a classification is needed: the first one divides the range of values of the chosen attribute into five different classes of equal size. As a further option, the number of classes can be changed. The second alternative allows the user to insert not only the number of classes, but also their size, that is, to define the endpoints of each class, so that they can be easily customized.

In the following, an example illustrates how the approach works.

[13] *The anti-monotonocity property* states that if a multi-time-interval pattern is frequent, so are all of its composing patterns (sub patterns). Accordingly, if a multi-time-interval pattern is not frequent, then all the patterns starting from it (super patterns) will not be either.

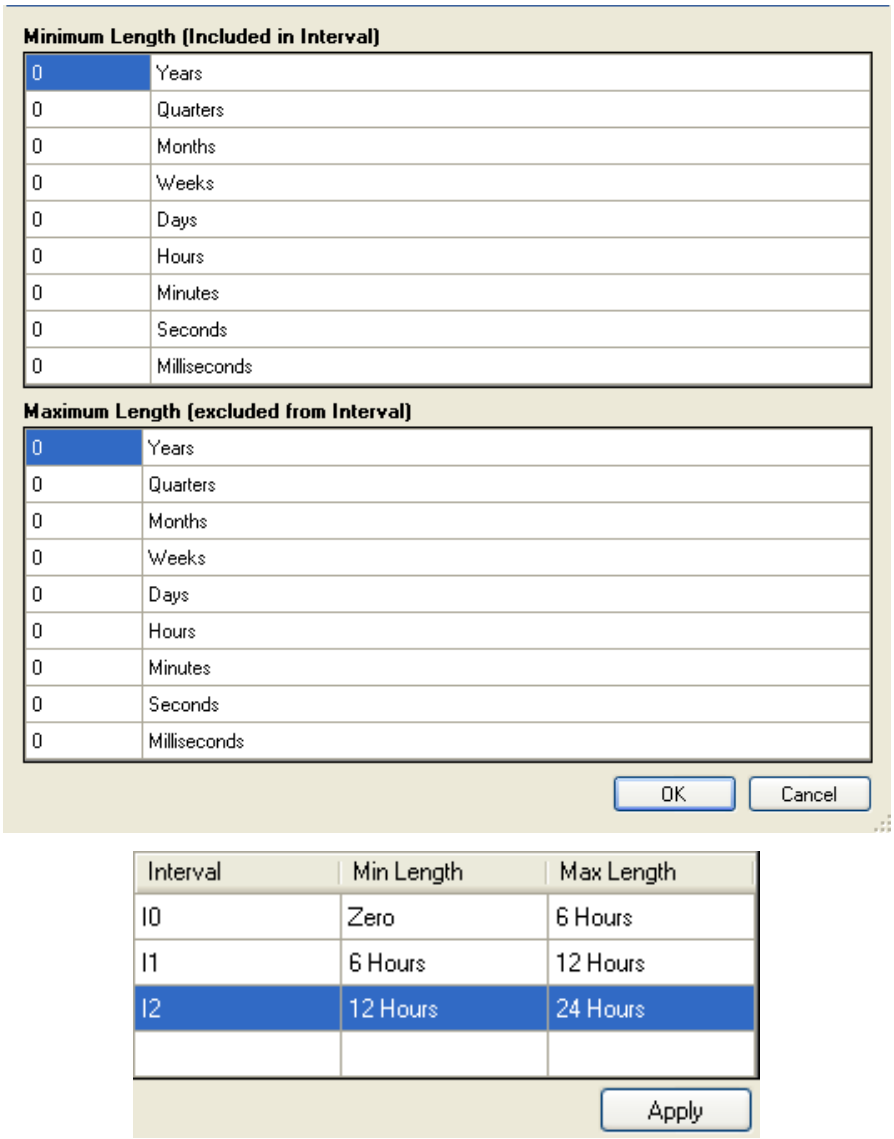

Figure 28: The interval configuration panel and the currently allowed granularities (top); an example of interval configuration (bottom).

6.4 Examples & Real World Application

In this section we first demonstrate with an easy example how the proposed approach works; after that, we illustrate how the approach has been applied to real world data within an Austrian project.

6.4.1 Examples

Let suppose that e_1, e_5, and e_8 are the most frequent single events (Step 0) obtained from the dataset partially shown in Table 4. Calculating all the patterns starting from them, we get (Step 1)

$e_1\ I_2\ e_2$

$e_1\ I_2\ e_2$

$e_1\ I_2\ e_0$

…

…

$e_1\ I_3\ e_5$

$e_1\ I_3\ e_0$

…

…

$e_1\ I_3\ e_{02.07.06}$[14]

…

This means that, starting from any occurrence of e_1 (as well as from any of e_5 and e_8) we calculate all the patterns composed by two events (or having length 1), within the range I_2 (that is, day by day, till a week). After that, we repeat within the range I_3 (that is, day by day, from one week till one month) and so on. Only those patterns above the given support are selected. Note that in this example we cannot take into account I_0 and I_1 since they have shorter durations (till one hour and till one day respectively) than the daily granularity of the dataset.

Now let's suppose that $e_1\ I_2\ e_2$ and $e_1\ I_3\ e_5$ are the patterns having a support greater than the given threshold. If we calculate all the patterns from them, we get (Step 2)

[14] $e_{02.07.06}$ is the event on the 2nd of July, that is the last event which is considered by I_3

$e_1\ I_2\ e_2\ I_2\ e_2$

$e_1\ I_2\ e_2\ I_2\ e_0$

…

and so on.

This means that, starting from any occurrence of $e_2\ I_2\ e_3$ (as well as for $e_1\ I_3\ e_5$) we calculate all the patterns composed by three events (or having length 2), within the range I_2, then I_3, and so on, and only those above the given support are selected.

Please note that according to the revised definition of multi-time interval patterns, we can also define the *length* of a multi-time interval pattern, as the number of subpatterns (i.e., patterns having form $e_x\ I_a\ e_y$) composing the whole pattern. In the above example, the patterns found during Step 1 have length 1, whereas those found during Step 2 have length 2.

6.4.2 A Real World Application: The DisCō Case Study

The approach we proposed has been applied within a project financed by the Austrian Federal Ministry of Transport, Innovation and Technology, called DisCō (Project number: 813388) and in this section and in the following one we are going to provide the first results of its application. The aim of the DisCō project is to develop novel Visual Analytics methods to visually as well as computationally analyse multivariate, time-oriented data and information to discover new and unexpected trends, patterns, and relationships. The goals of the intertwined visual and analytical methods are to ensure high usability and good control of the integrated mining techniques by applying intuitive visualizations and visual interfaces.

For what concerns the first example (Section 6.4.2.1), the developed methods are implemented within the software framework TIS[15] and the data are about employees and turnover of a certain shopping mall. On the contrary, the second example has been realized in a more general framework and the data concern a flight company.

6.4.2.1 Example 1

Let's suppose the user has to analyze the dataset which is partially shown in Table 3, and contains data spanning from the 1st of June 2006, till the 31st of December 2006. Once the "Multi-time interval pattern finding" operator is selected, a pop up window (see Figure 29) is used to define the parameters that must be inserted.

In detail, "*Columns of Interest*" represents the columns (attributes) the user selects, "*Columns to Classify*" the columns to be discretized, "*Support*" whether the support must be greater, greater than, smaller or smaller than the value given in "*Threshold*", "*Classes*" stands for the number of classes according to which one or more attributes are discretized and "*Class endpoints*" allows to define the endpoints of such classes. Finally the remaining checkbox allows the user to visualize or hide the intermediate results.

Then let's suppose a user selects the attributes "*#Employee*" and "*Turnover*", without discretization and without giving any threshold value.

Figure 30 shows the outcomes of this selection: in detail, apart from the first six columns, which have been already described in Section 6.2, the events composed by the attributes "*#Employee*" and "*Turnover*" are shown in the column "*Event*". In this case the user gets 44 times the event e_3, two times the event e_{34} and once all other events. This outcome is not particularly interesting, since for event e_3 both the value for "*#Employee*" and "*Turnover*" are null. Moreover, the values of "*Turnover*" are widespread. Therefore, the

[15] TIS (Time Intelligence Solutions ®) is a flexible Business Intelligence Solution developed by XIMES. It focuses on time data with the aim to support personnel planning questions and forecasts by analyzing temporal patterns of working time.

columns "*ResultStep1*" and "*ResultStep2*" (containing the results of Step 1 and Step2 respectively) are also not particularly relevant. For instance, the meaning of the multi-time interval pattern e_3 I_4 e_3, which occurs almost 400 times, is that the event "no employee and no turnover" occurs once, then in period between one month and a quarter (interval I_4) other 395 times (such event represents the fact that on a certain day no employees were at work and the corresponding turnover amounted to zero euro).

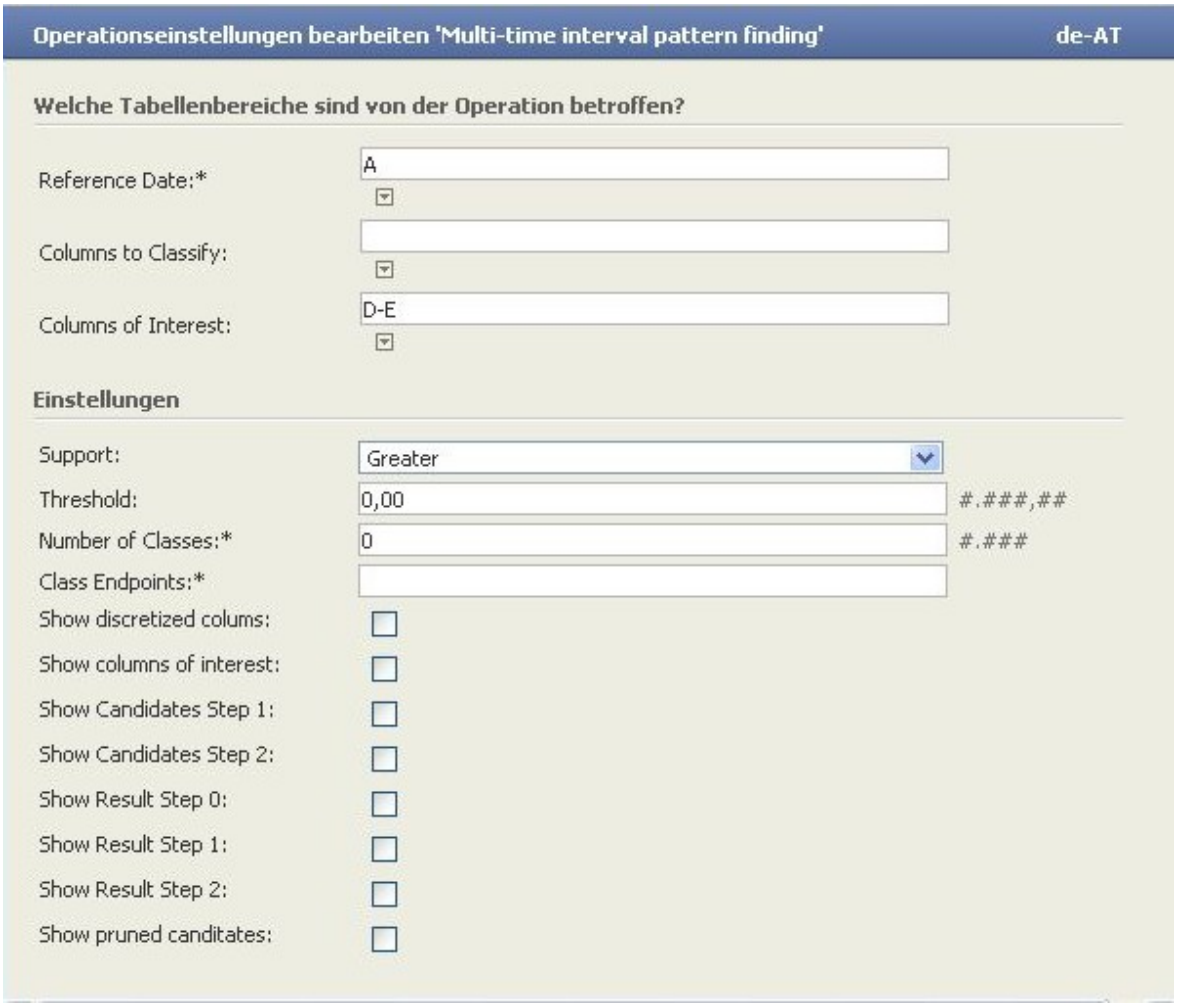

Figure 29: Input parameters. "*Columns of Interest*" represents the columns (attributes) the user selects, "*Columns to Classify*" the columns to be discretized, "*Support*" whether the support must be greater, greater than, smaller or smaller than the value given in "*Threshold*", "*Number of Classes*" stands for the number of classes according to which one or more attributes are discretized and "*Class Endpoints*"allows to define the endpoints of such classes. Finally the remaining checkbox allows the user to visualize or hide the intermediate results.

Therefore, the next step can be the discretization of the attribute "*Turnover*", for instance by defining the endpoints of the classes as 0, 1.000, 2.000, 3.000, and 4.000 and with 0.001 as support threshold. Figure 31 shows the **results** of such parameters adjustment. In this case, more candidates are found after the first step (e.g., event e_0, which means, "24 Employees and a turnover between 3.000 and 4.000 Euro", occurs 47 times) and more meaningful patterns are discovered. For instance, in a period between 1 day and 1 week (interval I_2), the event e_0 is followed 93 times by another occurrence of e_0 (e_0 I_2 e_0), 23 times by the event e_4 (that is, e_0 I_2 e_4 where e_4 means "23 Employees and a turnover between 3.000 and 4.000 Euro"), 34 times by the event e_7 (that is, e_0 I_2 e_7 where e_7 means "11 Employees and a turnover between 0 and 1.000 Euro") and so on. Similarly, in a larger range, i.e., in a period between 1 week and 1 month (interval I_3) the event e_0 is followed 312 times by another occurrence of e_0 (e_0 I_3 e_0), 116 times by e_3 (that is, e_0 I_3 e_4, where e_4 means "23 Employees and a turnover between 3.000 and 4.000 Euro"), 96 times by the event e_7 (that is, e_0 I_3 e_7 , where e_7 means "11 Employees and a turnover between 0 and 1.000 Euro") and so on.

This procedure can be repeated many times, until the results satisfy the user.

No.	A begin	B end	C kst	D #employees	E turnover	F day of week	G Event	H ResultStep0	I ResultStep1	J ResultStep2
1	01.06.2006 00:00:00	02.06.2006 00:00:00	Baho II	24,00	3.537,00	Donnerstag	e0	e0: 1	e34I4e3: 26 e34I5e3: 26	e34I4e3I3e3: 117 e34I4e3I4e3: 252 e34I4e3I5e3: 119
2	02.06.2006 00:00:00	03.06.2006 00:00:00	Baho II	24,00	3.493,00	Freitag	e1	e1: 1	e3I2e3: 44 e3I3e3: 193 e3I4e3: 395 e3I5e149: 23 e3I5e150: 23 e3I5e151: 23 e3I5e152: 23 e3I5e153: 23 e3I5e154: 24 e3I5e155: 24 e3I5e156: 24 e3I5e157: 24 e3I5e158: 24 e3I5e159: 24 e3I5e160: 25 e3I5e161: 25 e3I5e162: 25 e3I5e163: 25 e3I5e164: 25 e3I5e165: 25 e3I5e166: 26 e3I5e167: 26 e3I5e168: 26 e3I5e169: 26 e3I5e3: 314	e3I2e3I2e3: 78 e3I2e3I3e3: 143 e3I2e3I4e3: 380 e3I2e3I5e3: 318 e3I3e3I3e3: 827 e3I3e3I4e100: 71 e3I3e3I4e101: 71 e3I3e3I4e102: 66 e3I3e3I4e103: 77 e3I3e3I4e104: 77 e3I3e3I4e105: 77 e3I3e3I4e107: 77 e3I3e3I4e108: 73 e3I3e3I4e109: 83 e3I3e3I4e110: 83 e3I3e3I4e111: 83 e3I3e3I4e112: 83 e3I3e3I4e113: 83 e3I3e3I4e115: 89 e3I3e3I4e116: 89 e3I3e3I4e117: 89 e3I3e3I4e118: 89 e3I3e3I4e119: 85 e3I3e3I4e120: 85 e3I3e3I4e121: 89 e3I3e3I4e123: 89 e3I3e3I4e124: 89 e3I3e3I4e125: 85 e3I3e3I4e126: 85 e3I3e3I4e127: 89 e3I3e3I4e128: 89 e3I3e3I4e129: 89 e3I3e3I4e131: 85 e3I3e3I4e132: 85 e3I3e3I4e133: 85 e3I3e3I4e134: 85 e3I3e3I4e135: 85 e3I3e3I4e136: 85 e3I3e3I4e137: 81 e3I3e3I4e139: 85 e3I3e3I4e140: 85 e3I3e3I4e141: 85 e3I3e3I4e142: 81 e3I3e3I4e143: 75 e3I3e3I4e144: 70 e3I3e3I4e145: 69 e3I3e3I4e147: 64 e3I3e3I4e148: 64 e3I3e3I4e3: 1659 e3I3e3I4e94: 62 e3I3e3I4e95: 70 e3I3e3I4e96: 64 e3I3e3I4e97: 71 e3I3e3I4e99: 71 e3I3e3I5e143: 63 e3I3e3I5e144: 68 e3I3e3I5e145: 73 e3I3e3I5e146: 81 e3I3e3I5e147: 81 e3I3e3I5e148: 81 e3I3e3I5e150: 88 e3I3e3I5e151: 88 e3I3e3I5e152: 88 e3I3e3I5e153: 88 e3I3e3I5e154: 99 e3I3e3I5e155: 99 e3I3e3I5e156: 99 e3I3e3I5e158: 99 e3I3e3I5e159: 99 e3I3e3I5e160: 109 e3I3e3I5e161: 109 e3I3e3I5e162: 109 e3I3e3I5e163: 109 e3I3e3I5e164: 109 e3I3e3I5e166: 119 e3I3e3I5e167: 119 e3I3e3I5e168: 119 e3I3e3I5e169: 119 e3I3e3I5e3: 1181 e3I4e3I2e3: 417 e3I4e3I2e70: 62 e3I4e3I3e120: 64 e3I4e3I3e121: 64 e3I4e3I3e122: 64 e3I4e3I3e123: 64 e3I4e3I3e124: 80 e3I4e3I3e125: 80 e3I4e3I3e126: 80 e3I4e3I3e128: 80 e3I4e3I3e129: 80 e3I4e3I3e130: 80 e3I4e3I3e131: 80 e3I4e3I3e132: 80 e3I4e3I3e133: 64 e3I4e3I3e134: 64 e3I4e3I3e136: 64 e3I4e3I3e137: 80 e3I4e3I3e138: 80 e3I4e3I3e139: 64 e3I4e3I3e140: 64 e3I4e3I3e141: 64 e3I4e3I3e142: 64 e3I4e3I3e144: 63 e3I4e3I3e3: 1689 e3I4e3I3e72: 72 e3I4e3I3e73: 82 e3I4e3I3e74: 86 e3I4e3I3e75: 97 e3I4e3I3e76: 97 e3I4e3I3e78: 108 e3I4e3I3e79: 108 e3I4e3I3e80: 100 e3I4e3I3e81: 100 e3I4e3I3e82: 92 e3I4e3I3e83: 92 e3I4e3I3e84: 102 e3I4e3I3e86: 93 e3I4e3I3e87: 93 e3I4e3I3e88: 93 e3I4e3I3e89: 93 e3I4e3I3e90: 93 e3I4e3I3e91: 83 e3I4e3I3e92: 73 e3I4e3I4e100: 132 e3I4e3I4e101: 132 e3I4e3I4e102: 130 e3I4e3I4e103: 140 e3I4e3I4e104: 140 e3I4e3I4e105: 140 e3I4e3I4e106: 140 e3I4e3I4e108: 136 e3I4e3I4e109: 146 e3I4e3I4e110: 146 e3I4e3I4e111: 146 e3I4e3I4e112: 146 e3I4e3I4e113: 146 e3I4e3I4e114: 141 e3I4e3I4e116: 151 e3I4e3I4e117: 151 e3I4e3I4e118: 151 e3I4e3I4e119: 145 e3I4e3I4e120: 145 e3I4e3I4e121: 161 e3I4e3I4e122: 161 e3I4e3I4e124: 161 e3I4e3I4e125: 154 e3I4e3I4e126: 154 e3I4e3I4e127: 170 e3I4e3I4e128: 170 e3I4e3I4e129: 170 e3I4e3I4e130: 170 e3I4e3I4e132: 162 e3I4e3I4e133: 170 e3I4e3I4e134: 170 e3I4e3I4e135: 170 e3I4e3I4e136: 170 e3I4e3I4e137: 161 e3I4e3I4e138: 161 e3I4e3I4e140: 177 e3I4e3I4e141: 177 e3I4e3I4e142: 167 e3I4e3I4e143: 163 e3I4e3I4e144: 153 e3I4e3I4e145: 159 e3I4e3I4e146: 137 e3I4e3I4e148: 153 e3I4e3I4e149: 142 e3I4e3I4e150: 142 e3I4e3I4e151: 142 e3I4e3I4e152: 142 e3I4e3I4e153: 142 e3I4e3I4e154: 132 e3I4e3I4e156: 148 e3I4e3I4e157: 148 e3I4e3I4e158: 148 e3I4e3I4e159: 148 e3I4e3I4e160: 138 e3I4e3I4e161: 153 e3I4e3I4e162: 153 e3I4e3I4e164: 153 e3I4e3I4e165: 153 e3I4e3I4e166: 157 e3I4e3I4e167: 157 e3I4e3I4e168: 157 e3I4e3I4e169: 157 e3I4e3I4e3: 2898 e3I4e3I4e91: 69 e3I4e3I4e92: 79 e3I4e3I4e93: 89 e3I4e3I4e94: 99 e3I4e3I4e95: 121 e3I4e3I4e96: 121 e3I4e3I4e97: 132 e3I4e3I4e99: 132 e3I4e3I5e142: 59 e3I4e3I5e143: 79 e3I4e3I5e144: 89 e3I4e3I5e145: 99 e3I4e3I5e146: 121 e3I4e3I5e147: 121 e3I4e3I5e149: 132 e3I4e3I5e150: 132 e3I4e3I5e151: 132 e3I4e3I5e152: 132 e3I4e3I5e153: 132 e3I4e3I5e154: 142 e3I4e3I5e155: 142 e3I4e3I5e157: 142 e3I4e3I5e158: 142 e3I4e3I5e159: 142 e3I4e3I5e160: 152 e3I4e3I5e161: 152 e3I4e3I5e162: 152 e3I4e3I5e163: 152 e3I4e3I5e165: 152 e3I4e3I5e166: 162 e3I4e3I5e167: 162 e3I4e3I5e168: 162 e3I4e3I5e169: 162 e3I4e3I5e3: 1386
3	03.06.2006 00:00:00	04.06.2006 00:00:00	Baho II	10,00	535,30	Samstag	e2	e10: 1	--	--
4	04.06.2006 00:00:00	05.06.2006 00:00:00	Baho II	0,00	0,00	Sonntag	e3	e100: 1	--	--

Figure 30: Possible results of the approach (no discretization).

No.	A begin	B end	C kst	D #employees	E turnover	F day of week	G Discretized: turnover	H Event	I ResultStep0	J ResultStep1	K ResultStep2
1	01.06.2006 00:00:00	02.06.2006 00:00:00	Baho II	24,00	3.537,00	Donnerstag	d	e0	e0: 47	e0I2e0: 93 e0I2e2: 56 e0I2e3: 35 e0I2e4: 23 e0I2e7: 34 e0I3e0: 312 e0I3e11: 76 e0I3e2: 191 e0I3e3: 116 e0I3e4: 57 e0I3e6: 68 e0I3e7: 96 e0I4e0: 313 e0I4e1: 25 e0I4e10: 27 e0I4e11: 175 e0I4e13: 60 e0I4e14: 36 e0I4e2: 389 e0I4e23: 24 e0I4e3: 118 e0I4e4: 65 e0I4e6: 164 e0I4e7: 161 e0I5e0: 363 e0I5e11: 73 e0I5e13: 26 e0I5e2: 212 e0I5e3: 141 e0I5e4: 41 e0I5e7: 127	e0I3e0I3e0: 1969 e0I3e0I3e2: 1298 e0I3e0I4e0: 1383 e0I3e0I4e2: 1821 e0I3e0I5e0: 1120 e0I3e2I4e2: 1102 e0I3e2I5e0: 1087 e0I4e0I3e0: 1797 e0I4e0I3e2: 1412 e0I4e11I4e0: 2812 e0I4e11I4e2: 1692 e0I4e11I4e3: 1226 e0I4e11I5e0: 2085 e0I4e11I5e2: 1448 e0I4e2I4e0: 3582 e0I4e2I4e2: 1944 e0I4e2I4e3: 1409 e0I4e2I4e7: 1152 e0I4e2I5e0: 2107 e0I4e2I5e2: 1518 e0I4e6I4e0: 2415 e0I4e6I4e2: 1684 e0I4e6I4e3: 1292 e0I4e6I5e0: 2687 e0I4e6I5e2: 1685 e0I4e6I5e7: 1028
2	02.06.2006 00:00:00	03.06.2006 00:00:00	Baho II	24,00	3.493,00	Freitag	d	e0	e1: 3	e10I3e6: 26 e10I4e0: 48 e10I4e11: 39 e10I4e2: 56 e10I4e3: 34 e10I5e0: 100 e10I5e2: 61 e10I5e3: 30 e10I5e7: 35	e10I4e2I4e0: 1301
3	03.06.2006 00:00:00	04.06.2006 00:00:00	Baho II	10,00	535,30	Samstag	a	e1	e10: 4	e11I2e11: 28 e11I2e2: 30 e11I2e6: 28 e11I3e0: 98 e11I3e11: 78 e11I3e14: 24 e11I3e2: 116 e11I3e3: 50 e11I3e6: 67 e11I3e7: 39 e11I4e0: 403 e11I4e11: 132 e11I4e17: 26 e11I4e2: 260 e11I4e3: 167 e11I4e4: 35 e11I4e6: 31 e11I4e7: 129 e11I5e0: 282 e11I5e11: 38 e11I5e13: 40 e11I5e2: 202 e11I5e3: 95 e11I5e4: 51 e11I5e7: 119	e11I3e0I4e0: 1274 e11I3e2I4e0: 2347 e11I3e2I4e2: 1113 e11I3e3I4e0: 1035 e11I3e6I4e0: 1287 e11I3e6I5e0: 1109 e11I4e0I3e0: 3291 e11I4e0I3e2: 1677 e11I4e0I3e7: 1033 e11I4e0I4e0: 4005 e11I4e0I4e2: 2913 e11I4e0I4e3: 1239 e11I4e0I4e7: 1688 e11I4e11I3e0: 1079 e11I4e11I4e0: 2374 e11I4e2I3e0: 1649 e11I4e2I3e2: 1077 e11I4e2I4e0: 3612 e11I4e2I4e2: 1949 e11I4e2I4e3: 1098 e11I4e2I4e7: 1229 e11I4e3I3e0: 1411 e11I4e3I4e2: 1473 e11I4e7I4e0: 1440

Figure 31: Possible results of the approach (with discretization and support).

6.4.2.2 Example 2

The second example concerns data coming from a flight company. Along with the actual number of passengers of economy and business class, the data contain information about departure time, flight number, flight destination (as airport code) and the number of seats on the flight (both economy and business). Moreover, they also provide information about the number of expected passengers for any flight and day (economy and business), according to a not given forecast algorithm, whose forecast are provided about 2 months in advance and then further released 6, 4 and 2 weeks in advance. The aim of the company is to find interesting patterns in the data and to possibly improve their forecast.

The first step of our analysis was devoted to a short pre-processing phase: we added the city name to the destination, the distance between departure and destination, and a measure of how good is the forecast. To this aim, rather than the difference between the actual number of economy and business passengers (for a certain flight and date) and those provided by the forecast, this measure ("how good is the forecast") expresses the absolute error of such a difference with respect to the whole number of passengers. As a matter of fact, having on board 20 persons more or less for a Boeing 767 with about 300 available seats is completely different from the same situation for a small private jet or aircraft such as an ATR 42 (which have between 40 and 50 seats). Moreover, we divided each day in *Morning* (from 7 till 12), *Afternoon* (from 12 till 19), *Evening* (from 19 till 24) and *Night* (from 0 till 7) as well as we added the possibility to distinguish between *businessweek* (Monday to Friday) and *weekend* (Saturday and Sunday).

As shown in the "*Event Configuration*" of Figure 32 and Figure 33, for the first analysis, the destinations are classified into short distance (0-500 km) and middle-distance destinations (500-1500 km), the part of day is considered as an exact value as well as the day of the week, and the absolute error is divided into small error (from 0 to 0.3) and great error (from 0.3 to 1). Moreover, the

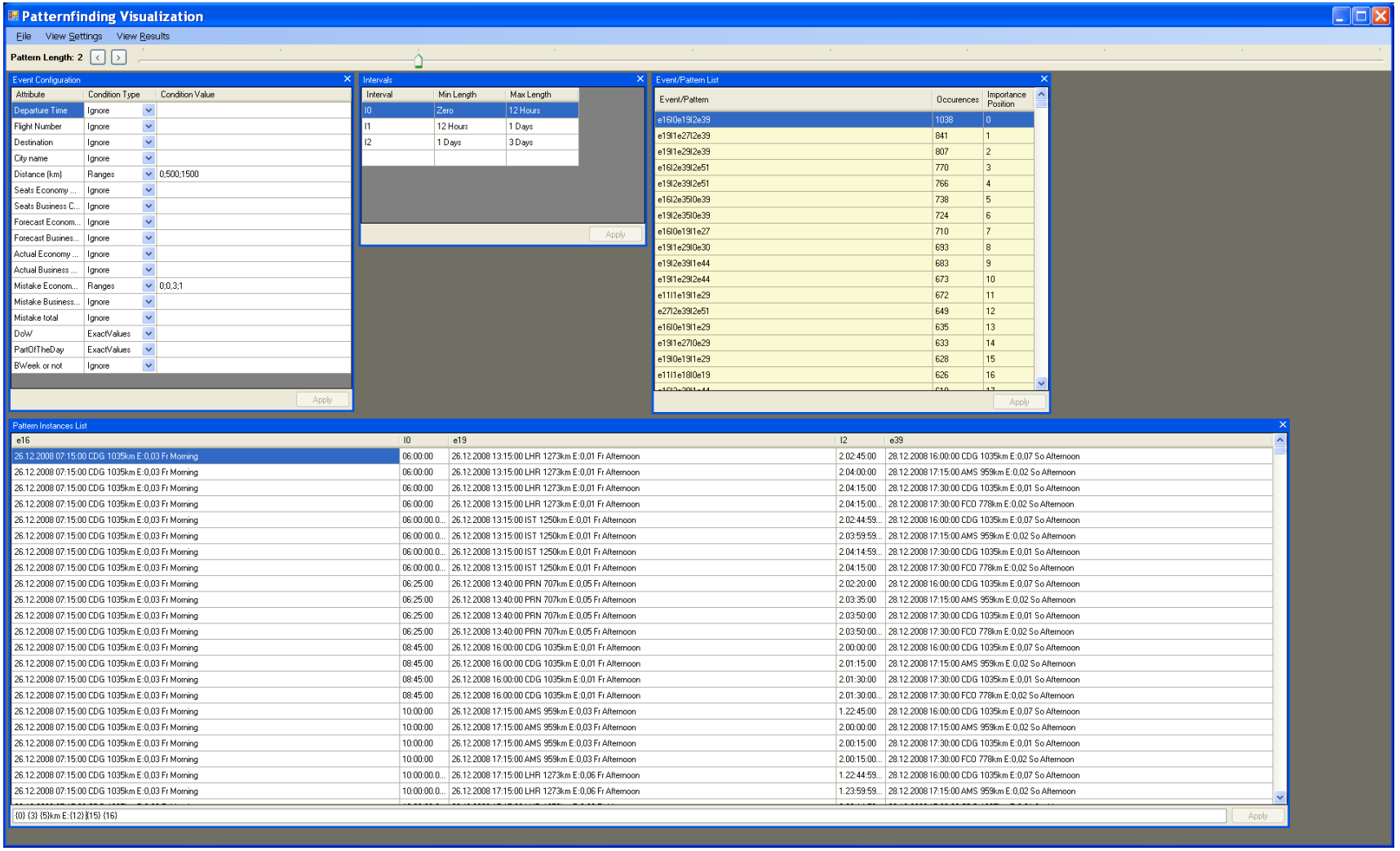

Figure 32: Results of the first analysis (economy).

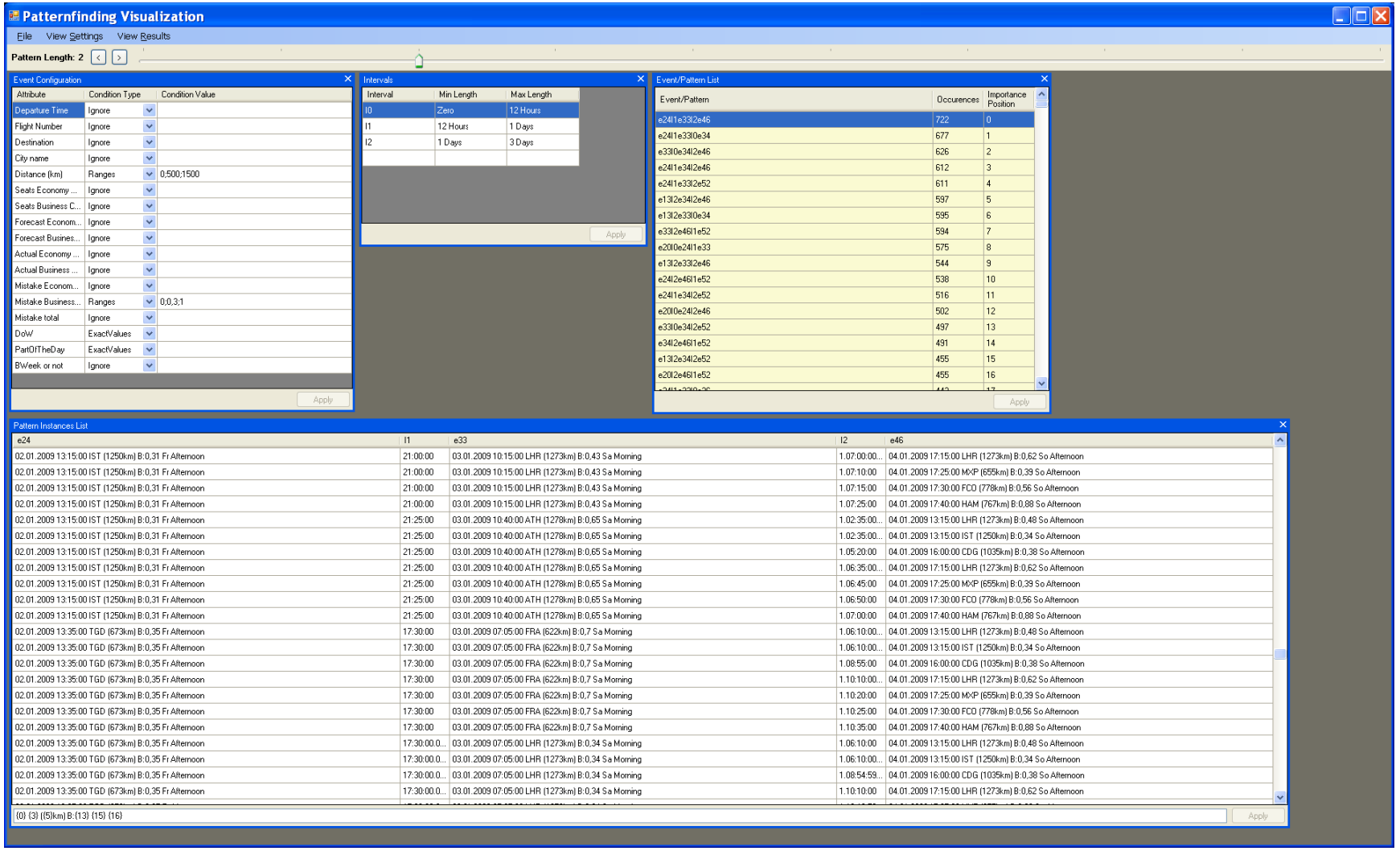

Figure 33: Results of the first analysis (business).

three intervals chosen span from 0 to 12 hours, from 12 hours to 1 day and from 1 day to 3 days respectively.

The results of the analysis shows that there are clearly some similar behaviours between passengers of economy and business class, that is, the most occurring multi-time interval pattern occurs between Friday and Sunday (then in what in the social time is usually know as "long weekend"). However, firstly they differ in the temporal occurrence of this pattern, that is, Friday morning-Friday afternoon-Sunday afternoon for the economy class and Friday afternoon-Saturday morning-Sunday afternoon for the business class. Secondly and most interestingly, they differ in the absolute error: whereas for the economy class is a small one, for the business class is greater than 0.3.

On the top of such results, a further analysis has been accomplished. Since it may be relevant to know whether the error between the forecast and the actual number of passenger is negative or positive, that is, whether there were more or less passengers than expected, we added such information to the analysis.

Then, similarly to the previous case, the error is divided into four different classes: great negative error (from -1 to -0.3), small negative error (from -0.3 to 0), small positive error (from 0 to 0.3) and great positive error (from 0.3 to 1). Moreover, we divided the day into *Business Morning* (from 6 to 9), *Business Evening* (from 16 to 24), and so called *Tourist time* (from 9 to 16), to distinguish between business trips and leisure trips.

The results of this analysis are shown in Figure 34 and Figure 35. In this case, the most occurring patterns related to the economy class occur not only during the weekend (or long weekend), but also during the business week. However, the error is always limited, assuming both negative and positive values. Concerning the most occurring patterns related to the business class, they occur both in the weekend and during the business week, but more interestingly they occur with a short time interval in between during the same day of the week (i.e. Friday). Moreover, the error is mainly focused on positive value in the small positive range.

In general, the results of these analyses outlined interesting behaviours which were previously unknown. Moreover, they suggested a possible extension to the approach which will be explained in Section 6.6.

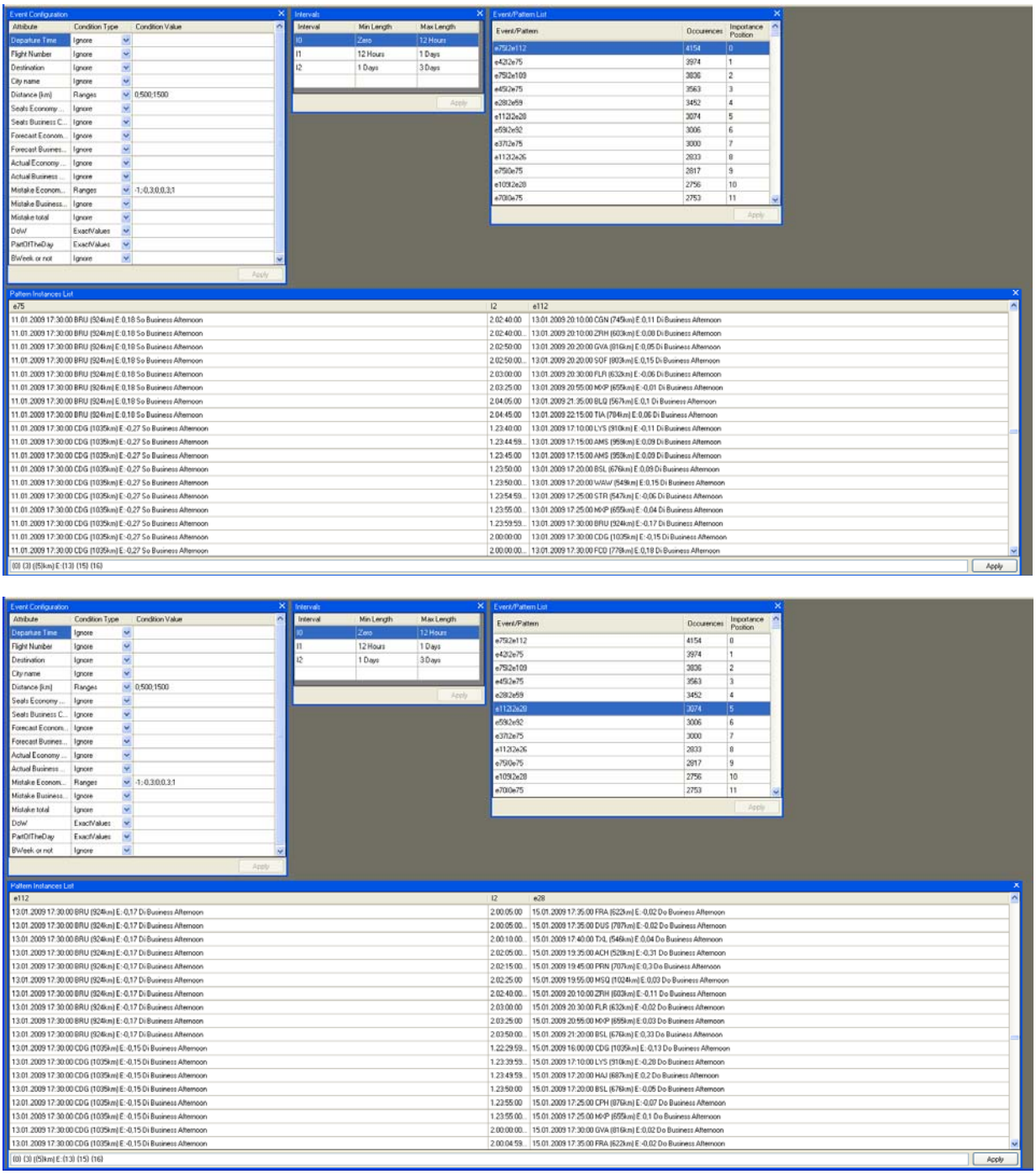

Figure 34: Improved analysis (economy).

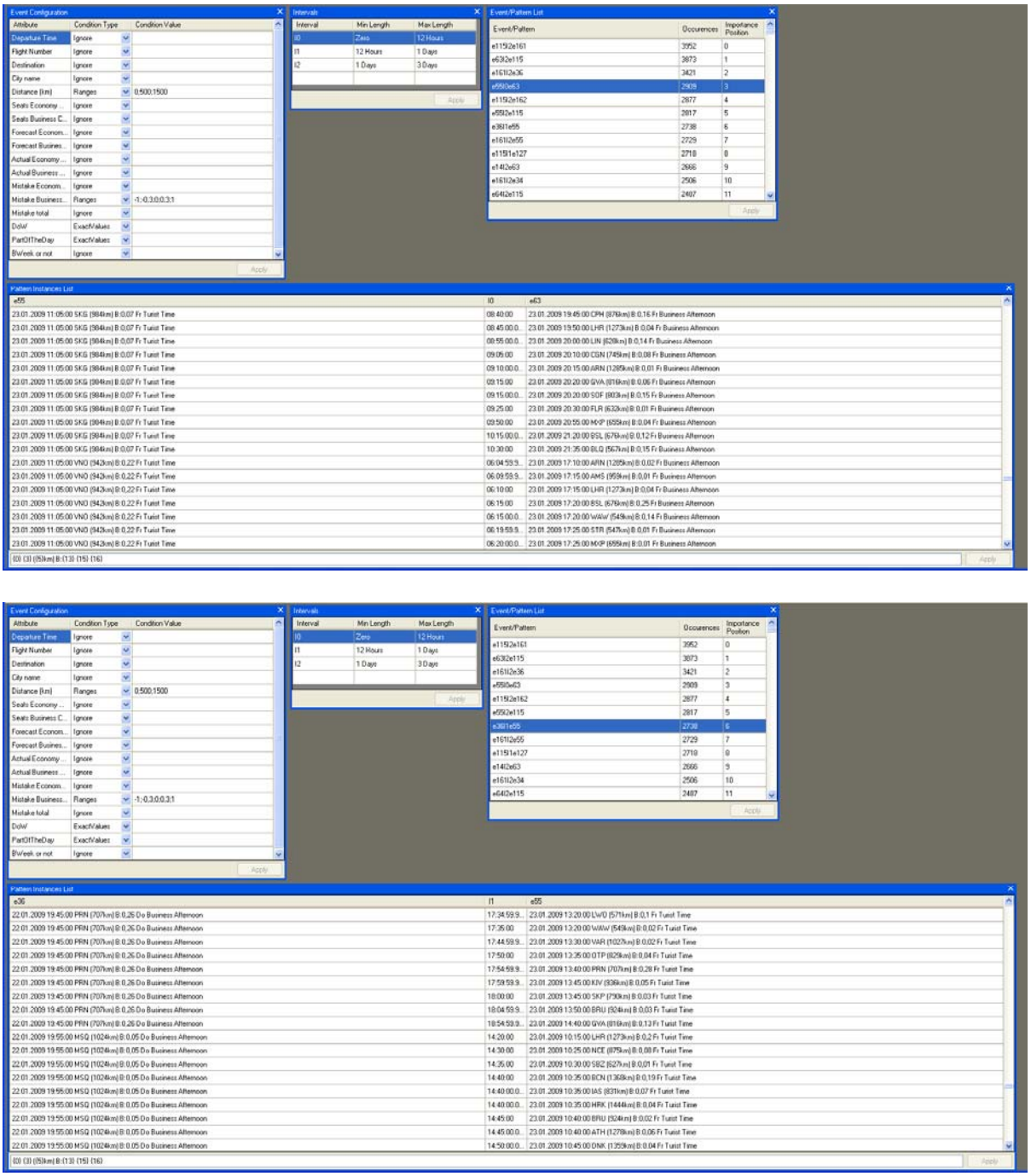

Figure 35: Improved analysis (business).

6.5 Discussion

As it was previously stated, the approach we propose extends some ideas about how to find sequential patterns with time intervals to non transactional databases and to deal explicitly with time-oriented data. However, in a

comparison with them, there are some differences. In particular we want to outline the differences with the method proposed by Chen et al. [5], Yang [6], and Hu et al. [94] and focused on "Apriori strategies", those proposed by Yoshida et al.[90], Vautier et al. [91], and Giannotti et al. [92] focused on temporal annotations and the method illustrated by Magnusson [89].

Regarding the methods focusing on Apriori strategies, the first main difference concerns the pruning strategies: apart from considerations about the necessity of pruning, which may even slow down the algorithm efficiency [164], Chen et al. [5], Yang [6] and Hu et al. [94] propose three pruning strategies, i.e., descending property, the time interval information matrix and the anti-monotonocity property. Dealing with non transactional databases and due to the fact that the datasets we considered are composed by non overlapping intervals (see Figure 25 and Figure 26), in our case we do not have to take into account the first two strategies. On the contrary, the third property holds also in our case. As a matter of fact, if $e_1\ I_2\ e_2\ I_2\ e_2$ is a frequently occurring multi-time interval pattern, its components $e_1\ I_2\ e_2$ and $e_2\ I_2\ e_2$ also occur frequently. On the other hand, if $e_1\ I_2\ e_2$ is not a frequently occurring multi-time interval pattern, any other pattern starting from that (e.g., $e_1\ I_2\ e_2\ I_3\ e_8$), will not occur frequently.

Beside that, two characteristics of our approach can be assimilated to pruning strategies: first, once an interval like I_5 in the default case[16] has been taken into account, we cannot create or search for other patterns starting from that. As a matter of fact, I_5 means the interval [1 quarter, ∞) and it does not make sense to calculate which events occur after an interval of length ∞. Therefore, any pattern such as $e_1\ I_5\ e_2\ I_2\ e_2$ is not considered. Second, according to the data, some intervals may not be considered. For example, the dataset of Table 3 is expressed at a daily level, therefore the default intervals I_0 and I_1 cannot be considered, since they span from 0 till 1 hour, and from 1 hour till 1 day

[16] This statement holds either in the case the intervals are defined as default or in the case they are user defined.

respectively. In case of datasets expressed at a level of hour or at an even finer granularity, all the intervals should be taken into account.

Moreover, concerning the definition of the intervals, Chen et al. [5] and Yang [6] and Hu et al. [94] propose a set of fixed time intervals beforehand as follows: all the intervals between successive itemsets are firstly collected to determine the whole time interval set and then these values are partitioned into five equally deep intervals, so that each of them contains roughly the same number of data. In our case, we pose our focus not only on the duration of the intervals between the events, but also on the temporal reference as well as on the social aspects of time. As a matter of fact, we defined not only six different default intervals which are relevant to the social time for several fields of applications, but we also allow users to define their own intervals, both in term of their amount and in term of their end points. In this way he user is involved in the definition of the parameters of the approach and may accomplish several tasks according to his/her needs.

Regarding the methods focusing on temporal annotations, Yoshida et al. [90] provide some heuristics to find some frequent delta patterns, but they do not consider any solution to find all of them and they are limited to serial phenomena. Moreover, concerning the time interval aspects, the interval bounds of delta patterns are always positive, and they neither take into account social time aspects, nor allow any interval choice. On the contrary, they cluster time intervals between two neighbouring itemsets using CF-Trees and count the number of occurrences of each candidate pattern, neglecting any user influence. To overcome some of these limitations, Vautier et al. [91], extends the notion of delta patterns with the use of so called chronicles: in this way, interval bounds of chronicles can be both positive and negative, and they can express both serial and parallel phenomena. However, even if they search for the most frequent intervals rather than provide them as default or as user defined input as we do, they still do not consider social time aspects and they miss any time reference, which may result very useful in several applications. Similarly, Giannotti et al. [92] introduce the notion of Temporally-annotated

sequences (TAS) based on the combination of frequency and temporal similarity: in this way, each transition in a sequential pattern is annotated with a typical transition time derived from the source data and then any annotation is used to feed a clustering algorithm in order to determine the representative values over a certain support. Therefore differently from our approach, they extract directly from the data a typical time (e.g., getting the typical time from A to B), rather than allow the user to input this parameter as we do. Beside that, they also miss any time reference and may only take a very limited advantage of social aspects of time. Moreover, to compare different TAS they adopt the τ-containment relation, which states that a TAS T1 is τ-contained into another one, T2, if the former is a subsequence of the latter and its transition times do not differ too much from those of its correspond itemsets in T2. Apart from considerations about the fuzziness of the second requirement, the time model upon which our approach is based also contain the Allen's basic interval relations, so that such operations are permitted as well as operators as union, difference, and so on.

Concerning the method proposed by Magnusson [89], the first main different is the strategy used to deal with all the possible patterns generated if all possible time windows are considered. Magnusson proposes a bottom up, level-by-level (or breadth-first) detection strategy by which simpler patterns are detected first, whereas more complex patterns are detected as patterns of simpler ones. In our case, we reduce the amount of time windows taken into account either six default different intervals or the user defined ones, which will be soon customizable. Yet, instead of a breadth-first algorithm, an a priori one along with user event definition and detection is adopted. Moreover, the temporal information in between is not given explicitly by Magnusson[17], whereas in our approach the intervals are explicitly provided.

Secondly, once the length of the patterns increases the method proposed by Magnusson shows some limitation, since repeating or similar patterns are not

[17] *Theme* (the implemented version of Magnusson's method) shows the observation period as a time line under the plot. Using such line one can indirectly obtain the temporal information in between.

easily recognizable and somehow grouped (and possibly counted). On the contrary, we group and count repeating patterns, giving also the possibility to the user to exclude from further steps some of them using a support threshold.

In general, a part from the above differences, our approach gives the user the possibility to be directly involved in the analysis, by choosing the attributes of interest, or the intervals in between, or even providing a support threshold. Furthermore, due to the methodology with which the events are created, we are able to deal not only with numerical but also with categorical attributes and both of them at the same time. For instance, given the dataset of Table 3, and another attribute, "*Weather*", with values like, "*sunny*", "*rainy*", etc., users can create an event by selecting "*Turnover*" and *"Weather"* in order to search for relations among these two attributes. In this way, it is possible for example, to test the hypothesis such as "If weather was bad for a long time and it becomes fine then it reduces shopping on Saturdays dramatically" or "After several Saturdays or weeks of fine weather levels of sales recover to a large degree".[18]

Moreover, by considering different time intervals, the approach enhances in a certain sense the idea of time window to multiple time windows: as a matter of fact, we search for repeating events and patterns using several time windows, according to the time intervals we defined.

Furthermore, note that the values of certain attributes might be too spread (as for the attribute *"Turnover"* in Table 3) and this can lead to the creation of too many events, or to consider events such as (#E=24 AND T=3537) and (#E=24 AND T=3493) as different, whereby for most of the analysis, they could be considered the same one. To address this problem, a discretization step has been defined. Currently, two alternatives are provided: an automatic classification groups the values of the chosen attribute in five different classes

[18] Both the hypotheses can be induced by the proposed approach if the dataset to be analysed contains in addition an attribute regarding the "weather conditions". Given this dataset, we can select such attribute to define the events, along with the turnover and for instance the day of the week, in order to find multi-time interval patterns which can validate these hypotheses.

of equal size, and as further option the number of classes can be changed. The second alternative allows the user to insert not only the number of classes, but also to define the endpoints of each class, so that they can be easily customized.
Lastly, as it will be clear in the Section 8.2, the possibility to visualize the results, interact with any step of the approach as well as to choose different input and to adjust them intuitively to perform several analysis is an added value of the approach itself.

6.6 Possible Extensions

As it has been outlined in the previous sections, the possibility to customize or adjust the input parameters according to user's needs is one of the great advantages of the approach we propose. In order to improve this capability and to overcome the actual limitations, we have already planned some further steps.

First of all, apart from the improvement of the possibility to define the classes for the discretization step, as well as to investigate some statistical methods (such as quantiles) to automatically calculate them, we can imagine that even the events can be user defined. Let's suppose a manager of a certain shopping mall tries to investigate the relationship between payday (set on the 15th and on the 27th of each month) and turnover. S/he wants to confirm the hypothesis that during the week after the payday, the turnover will be higher and more employees will be required. We can do this by replacing the first step of Section 6.3 with the following events

$$e_0 = \{\text{Turnover on the } 15^{th} \text{ AND Payday} = Y\}$$
$$e_1 = \{\text{Turnover on the } 27^{th} \text{ AND Payday} = Y\}$$

Considering them as the most relevant, we apply the proposed approach starting from the second step. In this way, we are able not only to find

previously unknown patterns (explorative analysis), but also to confirm some hypotheses (confirmative analysis).

Moreover, it could be useful to add a further step to the analysis which can help in the improvement of forecasting. In other words, once the most frequent occurring patterns are found, one can derive from them some summarized rules which can increase the precision of a given forecast. For example, in the case of the flight data (where the values of a forecast are already available), from the most occurring patterns we may discover that if for a certain destination on Friday morning the error for the business class is greater than 30% then in 50% of the cases there is a 10% increase of the same error in the afternoon. In this way, exploiting this information, the customer can improve his forecast analysis.

Finally, as it appears evident in Figure 30 and Figure 31, the greatest challenge for the next steps is to design a proper visualization framework, which allows not only for presenting an overview of the available data and the results, but also letting the user easily adjust the parameters as well as interact with each visual component, during each step of the approach. This issue will be discussed in the upcoming sections.

7 Validation of the Approach

No amount of experimentation can ever prove me right; a single experiment can prove me wrong.

Albert Einstein

In this part of the thesis, we provide a validation of the proposed approach. The validation is either mathematical, whereas we can demonstrate the correctness of the calculation of the events, or empirical, whereas we can show the exactness of the calculation of the multi-time interval patterns with some test datasets. In detail, we perform a two fold strategy, which will be described in section 7.1 and 7.2 respectively. The first part consists of manually calculating the events, the candidate patterns and the resulting patterns of a simple datasets and compare such results with those obtained using the implemented version of the approach. The second part represents the dual of the previous one: starting from some multi-time interval patterns, we create a synthetic dataset so that such patterns belong to the most frequently occurring ones. After that we run our implemented approach and compare the results with those expected.

Note that, as it was outlined in section 6.3, the approach is divided into a first step ("Step 0") whereas events are calculated and which differs from the rest, and some recursive steps ("Step 1", "Step 2", etc.), whereas multi-time interval patterns are calculated, which have length one, two, and so on. Thus, we demonstrate the correctness of the approach for the first step (that is, Step 0), and for the subsequent two steps ("Step 1" and "Step 2"), assuming that, cause of recursion, any further step follows the same rules, and do not require other proofs.

Furthermore, note that the whole approach is based on the events created during the first step and it works independently of the number of attributes of which they are composed (that is, one, two or more attributes). Therefore, we chose the easiest case (one attribute), but both proves still hold in case of more attributes.

7.1 Comparing Manual and Automatic Results

The first part of the validation consists of generating a small dataset, manually calculating the events, the multi-time interval patterns and their support, step by step.

The dataset (see Table 5) contains data spanning from the 1st June 2006 till the 9th June 2009 and the number of employees which are present day by day (#*E*).

Begin	End	#E
01.06.2006 00:00	02.06.2006 00:00	24
02.06.2006 00:00	03.06.2006 00:00	24
03.06.2006 00:00	04.06.2006 00:00	10
04.06.2006 00:00	05.06.2006 00:00	0
05.06.2006 00:00	06.06.2006 00:00	0
06.06.2006 00:00	07.06.2006 00:00	24
07.06.2006 00:00	08.06.2006 00:00	25
08.06.2006 00:00	09.06.2006 00:00	24
09.06.2006 00:00	10.06.2006 00:00	23

Table 5: Test dataset.

We are now ready to apply our approach as described in section 6.3.

Step 0

Associating an event to the occurrence of each unique value of the attribute #E, we have:

Begin	End	#E	Event
01.06.2006 00:00	02.06.2006 00:00	24	e_0
02.06.2006 00:00	03.06.2006 00:00	24	e_0
03.06.2006 00:00	04.06.2006 00:00	10	e_1
04.06.2006 00:00	05.06.2006 00:00	0	e_2
05.06.2006 00:00	06.06.2006 00:00	0	e_2
06.06.2006 00:00	07.06.2006 00:00	24	e_0
07.06.2006 00:00	08.06.2006 00:00	25	e_3
08.06.2006 00:00	09.06.2006 00:00	24	e_0
09.06.2006 00:00	10.06.2006 00:00	23	e_4

Table 6: Creation of the events.

After that, it is possible to count the occurrences of each event and summarize the results into Table 7.

Event	# Occurrences
e_0	4
e_1	1
e_2	2
e_3	1
e_4	1

Table 7: Events and their occurrences.

If we assume that no support threshold has been given, all the events will be selected for the second step of the approach, otherwise only those which occur more than the given support will be selected.

Let's assume that the support threshold is 10%, so only the events whose occurrence is greater than 10% are selected. In this case, all the events are selected[19].

Step 1

Starting from e0, then e_1, e_2, e_3 and e_4 we obtain all the possible candidate patterns. In detail,

from e_0 that occurs 4 times

$e_0I_2e_0$	$e_0I_2e_1$	$e_0I_2e_3$	$e_0I_2e_4$
$e_0I_2e_1$	$e_0I_2e_2$	$e_0I_2e_0$	
$e_0I_2e_2$	$e_0I_2e_2$	$e_0I_2e_4$	
$e_0I_2e_2$	$e_0I_2e_0$		
$e_0I_2e_0$	$e_0I_2e_3$		
$e_0I_2e_3$	$e_0I_2e_0$		
$e_0I_3e_0$	$e_0I_3e_4$		
$e_0I_3e_4$			

Table 8: Multi-time interval patterns derived from e_0.

from e_1 that occurs once

[19] Since the whole number of occurrences is 9, the minimal amount of occurrences is 0.9 (10 % of 9). Therefore, even the events which occur once are above the support threshold.

$e_1I_2e_2$
$e_1I_2e_2$
$e_1I_2e_0$
$e_1I_2e_3$
$e_1I_2e_0$
$e_1I_2e_4$

Table 9: Multi-time interval patterns derived from e_1.

from e_2 that occurs 2 times

$e_2I_2e_2$	$e_2I_2e_0$
$e_2I_2e_0$	$e_2I_2e_3$
$e_2I_2e_3$	$e_2I_2e_0$
$e_2I_2e_0$	$e_2I_2e_4$
$e_2I_2e_4$	

Table 10: Multi-time interval patterns derived from e_2.

from e_3 that occurs 2 times

$e_3I_2e_0$
$e_3I_2e_4$

Table 11: Multi-time interval patterns derived from e_3.

from e_4 that occurs once, we do not obtain any further patterns.

We can now summarize the results in Table 12:

Multi-time interval pattern	# Occurrences
$e_0I_2e_0$	5
$e_0I_2e_1$	2
$e_0I_2e_2$	4
$e_0I_2e_3$	3
$e_0I_2e_4$	2
$e_0I_3e_0$	1
$e_0I_3e_4$	2
$e_1I_2e_0$	2
$e_1I_2e_2$	2
$e_1I_2e_3$	1
$e_1I_2e_4$	1
$e_2I_2e_0$	4
$e_2I_2e_2$	1
$e_2I_2e_3$	2
$e_2I_2e_4$	2
$e_3I_2e_0$	1
$e_3I_2e_4$	1

Table 12: Candidate multi-time interval patterns found after Step 1 and their occurrences.

As in the previous step, if we assume that no support threshold has been given, all the multi-time interval patterns will be selected for the next step of, otherwise only those which occur more than the given support will be selected. Let's assume that the support threshold is 10%, so only the patterns whose occurrence is greater than 10% are selected. In this case, only $e_0I_2e_0$ (which occurs 5 times), $e_0I_2e_2$ and $e_2I_2e_0$ (which occur both 4 times) are selected[20].

[20] Since the whole number of occurrences is 36, the minimal amount of occurrences is 3.6 (10 % of 36). Therefore, only the multi-time interval patterns which occur more than 3.6 times are above the support threshold.

Step 2

Starting from the patterns found in the previous step, we obtain all the possible candidate patterns. In detail,

from $e_0I_2e_0$ that occurs 5 times

$e_0I_2e_0I_2e_1$	$e_0I_2e_0I_2e_3$	$e_0I_2e_0I_2e_4$	$e_0I_2e_0I_2e_3$	$e_0I_2e_0I_2e_4$
$e_0I_2e_0I_2e_2$	$e_0I_2e_0I_2e_0$		$e_0I_2e_0I_2e_0$	
$e_0I_2e_0I_2e_2$	$e_0I_2e_0I_2e_4$		$e_0I_2e_0I_2e_4$	
$e_0I_2e_0I_2e_0$				
$e_0I_2e_0I_2e_3$				
$e_0I_2e_0I_2e_0$				
$e_0I_2e_0I_3e_4$				

Table 13: Multi-time interval patterns derived from $e_0I_2e_0$.

from $e_0I_2e_2$ that occurs 4 times

$e_0I_2e_2I_2e_2$	$e_0I_2e_2I_2e_0$	$e_0I_2e_2I_2e_2$	$e_0I_2e_2I_2e_0$
$e_0I_2e_2I_2e_0$	$e_0I_2e_2I_2e_3$	$e_0I_2e_2I_2e_0$	$e_0I_2e_2I_2e_3$
$e_0I_2e_2I_2e_3$	$e_0I_2e_2I_2e_0$	$e_0I_2e_2I_2e_3$	$e_0I_2e_2I_2e_0$
$e_0I_2e_2I_2e_0$	$e_0I_2e_2I_2e_4$	$e_0I_2e_2I_2e_0$	$e_0I_2e_2I_2e_4$
$e_0I_2e_2I_2e_4$		$e_0I_2e_2I_2e_4$	

Table 14: Multi-time interval patterns derived from $e_0I_2e_2$.

from $e_2I_2e_0$ that occurs 4 times

$e_2I_2e_0I_2e_3$	$e_2I_2e_0I_2e_4$	$e_2I_2e_0I_2e_3$	$e_2I_2e_0I_2e_4$
$e_2I_2e_0I_2e_0$		$e_2I_2e_0I_2e_0$	
$e_2I_2e_0I_2e_4$		$e_2I_2e_0I_2e_4$	

Table 15: Multi-time interval patterns derived from $e_2I_2e_0$.

We can now summarize the results in Table 16, as follows:

Multi-time interval pattern	# Occurrences
$e_0I_2e_0I_2e_0$	4
$e_0I_2e_0I_2e_1$	1
$e_0I_2e_0I_2e_2$	2
$e_0I_2e_0I_2e_3$	3
$e_0I_2e_0I_2e_4$	4
$e_0I_2e_0I_3e_4$	1
$e_0I_2e_2I_2e_0$	8
$e_0I_2e_2I_2e_2$	2
$e_0I_2e_2I_2e_3$	4
$e_0I_2e_2I_2e_4$	4
$e_2I_2e_0I_2e_0$	2
$e_2I_2e_0I_2e_3$	2
$e_2I_2e_0I_2e_4$	4

Table 16: Candidate multi-time interval patterns found after Step 2 and their occurrences.

As in the previous steps, if we assume that no support threshold has been given, all the multi-time interval patterns will be selected for the next step of, otherwise only those which occur more than the given support will be selected.

Let's assume that the support threshold is 10%, so only the patterns whose occurrence is greater than 10% are selected. In this case, only $e_0I_2e_2I_2e_0$ which occurs 8 times is selected[21].

[21] Since the whole number of occurrences is 41, the minimal amount of occurrences is 4.1 (10 % of 41). Therefore, only the multi-time interval patterns which occur more than 4.1 times are above the support threshold.

As outlined above, since the steps after the Step 2 repeated recursively, we assume that no further steps need to be proved. For such reason, now we can compare these results with those from the implemented version.

Figure 36 show such results. As one can observe, the results of all the steps manually obtained and automatically calculated coincide. As a matter of fact, as result of Step 0, we have e_0 which occurs 4 times, e_2 twice, e_1, e_3 and e_4 once (as in Table 7), as result from Step 1, $e_0I_2e_0$, which occurs 5 times, $e_0I_2e_2$ and $e_2I_2e_0$, which occur both 4 times, and finally as result of Step 2 the pattern $e_0I_2e_2I_2e_0$ which occurs 8 times.

No.	A begin	B end	C kst	D #employees	E turnover	F day of week	G Event	H CanditatesStep1	I CanditatesStep2	J ResultStep0	K ResultStep1	L ResultStep2
1	01.06.2006 00:00:00	02.06.2006 00:00:00	Baho II	24,00	3.537,00	Donnerstag	e0	e0I2e0 e0I2e1 e0I2e2 e0I2e2 e0I2e0 e0I2e3 e0I3e0 e0I3e4 e0I2e1 e0I2e2 e0I2e2 e0I2e0 e0I2e3 e0I2e0 e0I3e4 e0I2e3 e0I2e0 e0I2e4 e0I2e4 e1I2e2 e1I2e2 e1I2e0 e1I2e3 e1I2e0 e1I2e4 e2I2e2 e2I2e0 e2I2e3 e2I2e0 e2I2e4 e2I2e0 e2I2e3 e2I2e0 e2I2e4 e3I2e0 e3I2e4	e0I2e0I2e1 e0I2e0I2e2 e0I2e0I2e2 e0I2e0I2e2 e0I2e0I2e0 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I3e4 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e0I2e0I2e4 e0I2e0I2e4 e0I2e0I2e2 e0I2e0I2e0 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e0I2e0I2e2 e0I2e0I2e0 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e0I2e0I2e4 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e0I2e0I2e3 e0I2e0I2e0 e0I2e0I2e4 e2I2e0I2e4 e2I2e0I2e4 e2I2e0I2e3 e2I2e0I2e0 e2I2e0I2e4 e2I2e0I2e4	e0: 4	e0I2e0: 5 e0I2e2: 4	e0I2e2I2e0: 8
2	02.06.2006 00:00:00	03.06.2006 00:00:00	Baho II	24,00	3.493,00	Freitag	e0	--	--	e1: 1	e2I2e0: 4	--
3	03.06.2006 00:00:00	04.06.2006 00:00:00	Baho II	10,00	535,30	Samstag	e1	--	--	e2: 2	--	--
4	04.06.2006 00:00:00	05.06.2006 00:00:00	Baho II	0,00	0,00	Sonntag	e2	--	--	e3: 1	--	--
5	05.06.2006 00:00:00	06.06.2006 00:00:00	Baho II	0,00	0,00	Montag	e2	--	--	e4: 1	--	--
6	06.06.2006 00:00:00	07.06.2006 00:00:00	Baho II	24,00	3.483,50	Dienstag	e0	--	--	--	--	--
7	07.06.2006 00:00:00	08.06.2006 00:00:00	Baho II	25,00	3.808,70	Mittwoch	e3	--	--	--	--	--
8	08.06.2006 00:00:00	09.06.2006 00:00:00	Baho II	24,00	3.870,70	Donnerstag	e0	--	--	--	--	--
9	09.06.2006 00:00:00	10.06.2006 00:00:00	Baho II	23,00	3.396,40	Freitag	e4	--	--	--	--	--

« Zurück Vor » Gehe zu Seite 1 insg. 1 Anzahl Zeilen: 9 | Spalten: 12

Figure 36: Results of the test dataset. The columns "*CandidateStep1*" and "*CandidateStep2*" show all the possible candidates of Step 1 and Step 2 respectively. The columns *"ResultStep0"*, *"ResultStep1* and *"ResultStep2"* show the result of Step 0 (events) and the results of Step 1 and Step2, that is, multi-time interval patterns (of length 1 and 2) which are above the given support threshold.

7.2 Finding Expected Patterns

The second part of the validation consists of generating two events, some multi-time interval patterns and create a synthetic dataset, in which those patterns are the most frequently occurring. After that we compare the expected results with those obtained from the implemented version of the approach.

To this aim, we choose as events the values 18 and 74, which can be identified as e_x and e_y respectively. We decide that e_x occurs 10 times and e_y 8 times. Then, starting from them we choose to discover the following multi-time interval patterns:

$e_x\ I_2\ e_x$

$e_x\ I_3\ e_y$

$e_x\ I_2\ e_y\ I_3\ e_y$

Given such constraints, we create a synthetic dataset which contains those events and patterns as some of the most frequent ones (Table 17).

Begin	**End**	**Value**
01.01.2008 00:00	02.01.2008 00:00	1
02.01.2008 00:00	03.01.2008 00:00	2
03.01.2008 00:00	04.01.2008 00:00	18
04.01.2008 00:00	05.01.2008 00:00	3
05.01.2008 00:00	06.01.2008 00:00	18
06.01.2008 00:00	07.01.2008 00:00	4
07.01.2008 00:00	08.01.2008 00:00	5
08.01.2008 00:00	09.01.2008 00:00	6
09.01.2008 00:00	10.01.2008 00:00	7
10.01.2008 00:00	11.01.2008 00:00	8
11.01.2008 00:00	12.01.2008 00:00	9
12.01.2008 00:00	13.01.2008 00:00	74
13.01.2008 00:00	14.01.2008 00:00	10
14.01.2008 00:00	15.01.2008 00:00	11
15.01.2008 00:00	16.01.2008 00:00	12
16.01.2008 00:00	17.01.2008 00:00	13
17.01.2008 00:00	18.01.2008 00:00	14

18.01.2008 00:00	19.01.2008 00:00	15
19.01.2008 00:00	20.01.2008 00:00	74
20.01.2008 00:00	21.01.2008 00:00	17
21.01.2008 00:00	22.01.2008 00:00	19
22.01.2008 00:00	23.01.2008 00:00	20
23.01.2008 00:00	24.01.2008 00:00	21
24.01.2008 00:00	25.01.2008 00:00	22
25.01.2008 00:00	26.01.2008 00:00	23
26.01.2008 00:00	27.01.2008 00:00	24
27.01.2008 00:00	28.01.2008 00:00	25
28.01.2008 00:00	29.01.2008 00:00	26
29.01.2008 00:00	30.01.2008 00:00	27
30.01.2008 00:00	31.01.2008 00:00	28
31.01.2008 00:00	01.02.2008 00:00	29
01.02.2008 00:00	02.02.2008 00:00	30
02.02.2008 00:00	03.02.2008 00:00	31
03.02.2008 00:00	04.02.2008 00:00	32
04.02.2008 00:00	05.02.2008 00:00	33
05.02.2008 00:00	06.02.2008 00:00	34
06.02.2008 00:00	07.02.2008 00:00	35
07.02.2008 00:00	08.02.2008 00:00	18
08.02.2008 00:00	09.02.2008 00:00	18
09.02.2008 00:00	10.02.2008 00:00	36
10.02.2008 00:00	11.02.2008 00:00	37
11.02.2008 00:00	12.02.2008 00:00	38
12.02.2008 00:00	13.02.2008 00:00	39
13.02.2008 00:00	14.02.2008 00:00	18
14.02.2008 00:00	15.02.2008 00:00	41
15.02.2008 00:00	16.02.2008 00:00	74
16.02.2008 00:00	17.02.2008 00:00	42

17.02.2008 00:00	18.02.2008 00:00	43
18.02.2008 00:00	19.02.2008 00:00	44
19.02.2008 00:00	20.02.2008 00:00	45
20.02.2008 00:00	21.02.2008 00:00	46
21.02.2008 00:00	22.02.2008 00:00	47
22.02.2008 00:00	23.02.2008 00:00	48
23.02.2008 00:00	24.02.2008 00:00	74
24.02.2008 00:00	25.02.2008 00:00	50
25.02.2008 00:00	26.02.2008 00:00	51
26.02.2008 00:00	27.02.2008 00:00	52
27.02.2008 00:00	28.02.2008 00:00	53
28.02.2008 00:00	29.02.2008 00:00	54
29.02.2008 00:00	01.03.2008 00:00	55
01.03.2008 00:00	02.03.2008 00:00	56
02.03.2008 00:00	03.03.2008 00:00	57
03.03.2008 00:00	04.03.2008 00:00	58
04.03.2008 00:00	05.03.2008 00:00	59
05.03.2008 00:00	06.03.2008 00:00	60
06.03.2008 00:00	07.03.2008 00:00	61
07.03.2008 00:00	08.03.2008 00:00	62
08.03.2008 00:00	09.03.2008 00:00	63
09.03.2008 00:00	10.03.2008 00:00	18
10.03.2008 00:00	11.03.2008 00:00	64
11.03.2008 00:00	12.03.2008 00:00	18
12.03.2008 00:00	13.03.2008 00:00	65
13.03.2008 00:00	14.03.2008 00:00	18
14.03.2008 00:00	15.03.2008 00:00	67
15.03.2008 00:00	16.03.2008 00:00	68
16.03.2008 00:00	17.03.2008 00:00	69
17.03.2008 00:00	18.03.2008 00:00	70

18.03.2008 00:00	19.03.2008 00:00	71
19.03.2008 00:00	20.03.2008 00:00	74
20.03.2008 00:00	21.03.2008 00:00	72
21.03.2008 00:00	22.03.2008 00:00	73
22.03.2008 00:00	23.03.2008 00:00	74
23.03.2008 00:00	24.03.2008 00:00	76
24.03.2008 00:00	25.03.2008 00:00	77
25.03.2008 00:00	26.03.2008 00:00	78
26.03.2008 00:00	27.03.2008 00:00	79
27.03.2008 00:00	28.03.2008 00:00	80
28.03.2008 00:00	29.03.2008 00:00	81
29.03.2008 00:00	30.03.2008 00:00	82
30.03.2008 00:00	31.03.2008 00:00	83
31.03.2008 00:00	01.04.2008 00:00	84
01.04.2008 00:00	02.04.2008 00:00	85
02.04.2008 00:00	03.04.2008 00:00	86
03.04.2008 00:00	04.04.2008 00:00	87
04.04.2008 00:00	05.04.2008 00:00	88
05.04.2008 00:00	06.04.2008 00:00	89
06.04.2008 00:00	07.04.2008 00:00	90
07.04.2008 00:00	08.04.2008 00:00	91
08.04.2008 00:00	09.04.2008 00:00	92
09.04.2008 00:00	10.04.2008 00:00	93
10.04.2008 00:00	11.04.2008 00:00	94
11.04.2008 00:00	12.04.2008 00:00	95
12.04.2008 00:00	13.04.2008 00:00	96
13.04.2008 00:00	14.04.2008 00:00	97
14.04.2008 00:00	15.04.2008 00:00	18
15.04.2008 00:00	16.04.2008 00:00	98
16.04.2008 00:00	17.04.2008 00:00	99

17.04.2008 00:00	18.04.2008 00:00	100
18.04.2008 00:00	19.04.2008 00:00	18
19.04.2008 00:00	20.04.2008 00:00	101
20.04.2008 00:00	21.04.2008 00:00	102
21.04.2008 00:00	22.04.2008 00:00	103
22.04.2008 00:00	23.04.2008 00:00	104
23.04.2008 00:00	24.04.2008 00:00	105
24.04.2008 00:00	25.04.2008 00:00	106
25.04.2008 00:00	26.04.2008 00:00	107
26.04.2008 00:00	27.04.2008 00:00	108
27.04.2008 00:00	28.04.2008 00:00	109
28.04.2008 00:00	29.04.2008 00:00	110
29.04.2008 00:00	30.04.2008 00:00	74
30.04.2008 00:00	01.05.2008 00:00	74

Table 17: A synthetic dataset containing the expected events and patterns as some of the most frequent ones.

After that we run our implemented version using this dataset and compare the results[22]. As Figure 37 shows, the e_x (e_2 in the figure) occurs 10 times and the patterns $e_x\ I_2\ e_x$ and $e_x\ I_3\ e_y$ are some of the most expected: as a matter of fact, $e_2I_2e_2$ and $e_2I_3e_{10}$ occur 8 and 18 times respectively.

Figure 38 shows the results of Step 2, that is the multi-time interval patterns of length 2. Among them, as expected appears also $e_x\ I_2\ e_y\ I_3\ e_y$ ($e_2I_2e_2I_3e_{10}$ in the figure, which occurs 12 times).

As in the previous section, since the steps after the Step 2 repeated recursively, we assume that no further steps need to be proved. For such

[22] In order to reduce the number of possible candidates we set the support threshold at a very small value (0.0009).

reason, we stop the computation after Step 2 and compare these results with those expected.

No.	A Begin	B End	C Value	D Event	E ResultStep0	F ResultStep1
1	01.01.2008 00:00:00	02.01.2008 00:00:00	1	e0	e0: 1	e1OI3e2: 15 e1OI4e10: 17 e1OI4e2: 13
2	02.01.2008 00:00:00	03.01.2008 00:00:00	2	e1	e1: 1	e2I2e2: 8 e2I3e10: 18 e2I4e10: 26 e2I4e2: 31
3	03.01.2008 00:00:00	04.01.2008 00:00:00	18	e2	e10: 8	--
4	04.01.2008 00:00:00	05.01.2008 00:00:00	3	e3	e100: 1	--
5	05.01.2008 00:00:00	06.01.2008 00:00:00	18	e2	e101: 1	--
6	06.01.2008 00:00:00	07.01.2008 00:00:00	4	e4	e102: 1	--
7	07.01.2008 00:00:00	08.01.2008 00:00:00	5	e5	e103: 1	--
8	08.01.2008 00:00:00	09.01.2008 00:00:00	6	e6	e104: 1	--
9	09.01.2008 00:00:00	10.01.2008 00:00:00	7	e7	e11: 1	--
10	10.01.2008 00:00:00	11.01.2008 00:00:00	8	e8	e12: 1	--
11	11.01.2008 00:00:00	12.01.2008 00:00:00	9	e9	e13: 1	--
12	12.01.2008 00:00:00	13.01.2008 00:00:00	74	e10	e14: 1	--
13	13.01.2008 00:00:00	14.01.2008 00:00:00	10	e11	e15: 1	--
14	14.01.2008 00:00:00	15.01.2008 00:00:00	11	e12	e16: 1	--
15	15.01.2008 00:00:00	16.01.2008 00:00:00	12	e13	e17: 1	--
16	16.01.2008 00:00:00	17.01.2008 00:00:00	13	e14	e18: 1	--
17	17.01.2008 00:00:00	18.01.2008 00:00:00	14	e15	e19: 1	--
18	18.01.2008 00:00:00	19.01.2008 00:00:00	15	e16	e2: 10	--
19	19.01.2008 00:00:00	20.01.2008 00:00:00	74	e10	e20: 1	--
20	20.01.2008 00:00:00	21.01.2008 00:00:00	17	e17	e21: 1	--

« Zurück Vor » Gehe zu Seite 1 insg. 7 Anzahl Zeilen: 121 | Spalten: 6

Figure 37: First part of the results of the synthetic dataset, where the occurrences of the events (Step 0) and multi-time interval patterns of length 1 are shown.

No.	A Begin	B End	C Value	D Event	E ResultStep2
1	01.01.2008 00:00:00	02.01.2008 00:00:00	1	e0	e1OI3e2I2e2: 14 e1OI3e2I2e63: 6 e1OI3e2I2e64: 6 e1OI3e2I3e10: 27 e1OI3e2I3e44: 5 e1OI3e2I3e45: 5 e1OI3e2I3e46: 5 e1OI3e2I3e48: 5 e1OI3e2I3e49: 5 e1OI3e2I3e50: 5 e1OI3e2I3e51: 5 e1OI3e2I3e52: 5 e1OI3e2I3e53: 5 e1OI3e2I3e54: 5 e1OI3e2I3e56: 5 e1OI3e2I3e57: 5 e1OI3e2I3e58: 5 e1OI3e2I3e68: 6 e1OI3e2I3e69: 6 e1OI3e2I3e70: 6 e1OI3e2I3e71: 6 e1OI3e2I3e73: 6 e1OI3e2I3e74: 6 e1OI3e2I3e75: 6 e1OI3e2I3e76: 6 e1OI3e2I3e77: 6 e1OI3e2I3e78: 6 e1OI3e2I3e79: 6 e1OI3e2I3e81: 6 e1OI3e2I3e82: 6 e1OI3e2I3e83: 6 e1OI3e2I3e84: 6 e1OI3e2I3e85: 6 e1OI3e2I3e86: 6 e1OI3e2I4e10: 32 e1OI3e2I4e101: 11 e1OI3e2I4e102: 11 e1OI3e2I4e103: 11 e1OI3e2I4e104: 11 e1OI3e2I4e2: 35 e1OI3e2I4e63: 5 e1OI3e2I4e64: 5 e1OI3e2I4e66: 5 e1OI3e2I4e67: 5 e1OI3e2I4e68: 5 e1OI3e2I4e69: 5 e1OI3e2I4e70: 5 e1OI3e2I4e71: 5 e1OI3e2I4e72: 5 e1OI3e2I4e74: 5 e1OI3e2I4e75: 5 e1OI3e2I4e76: 5 e1OI3e2I4e77: 5 e1OI3e2I4e78: 5 e1OI3e2I4e79: 5 e1OI3e2I4e80: 5 e1OI3e2I4e82: 5 e1OI3e2I4e83: 5 e1OI3e2I4e84: 5 e1OI3e2I4e85: 5 e1OI3e2I4e86: 5 e1OI3e2I4e87: 7 e1OI3e2I4e88: 7 e1OI3e2I4e90: 9 e1OI3e2I4e91: 11 e1OI3e2I4e92: 11 e1OI3e2I4e93: 11 e1OI3e2I4e94: 11 e1OI3e2I4e95: 11 e1OI3e2I4e96: 11 e1OI3e2I4e98: 11 e1OI3e2I4e99: 11 e1OI4e10I2e10: 7 e1OI4e10I2e70: 6 e1OI4e10I2e71: 6 e1OI4e10I2e72: 6 e1OI4e1OI3e10: 5 e1OI4e1OI3e76: 6 e1OI4e1OI3e77: 6 e1OI4e1OI3e78: 6 e1OI4e1OI3e79: 6 e1OI4e1OI3e80: 6 e1OI4e1OI3e81: 6 e1OI4e1OI3e82: 6 e1OI4e1OI3e84: 6 e1OI4e1OI3e85: 6 e1OI4e1OI3e86: 6 e1OI4e1OI3e87: 6 e1OI4e1OI3e88: 6 e1OI4e1OI3e89: 6 e1OI4e1OI3e90: 6 e1OI4e1OI3e92: 6 e1OI4e1OI3e93: 6 e1OI4e1OI3e94: 6 e1OI4e1OI4e10: 20 e1OI4e1OI4e100: 9 e1OI4e1OI4e101: 9 e1OI4e1OI4e102: 9 e1OI4e1OI4e104: 9 e1OI4e1OI4e2: 6 e1OI4e1OI4e95: 6 e1OI4e1OI4e96: 6 e1OI4e1OI4e97: 6 e1OI4e1OI4e98: 9 e1OI4e1OI4e99: 9 e1OI4e2I2e63: 6 e1OI4e2I2e64: 6 e1OI4e2I2e95: 6 e1OI4e2I2e96: 6 e1OI4e2I3e10: 23 e1OI4e2I3e101: 6 e1OI4e2I3e102: 6 e1OI4e2I3e104: 6 e1OI4e2I3e68: 6 e1OI4e2I3e69: 6 e1OI4e2I3e70: 6 e1OI4e2I3e71: 6 e1OI4e2I3e72: 6 e1OI4e2I3e73: 6 e1OI4e2I3e75: 6 e1OI4e2I3e76: 6 e1OI4e2I3e77: 6 e1OI4e2I3e78: 6 e1OI4e2I3e79: 6 e1OI4e2I3e80: 6 e1OI4e2I3e81: 6 e1OI4e2I3e83: 6 e1OI4e2I3e84: 6 e1OI4e2I3e85: 6 e1OI4e2I3e86: 6 e1OI4e2I4e10: 16 e1OI4e2I4e100: 7 e1OI4e2I4e101: 7 e1OI4e2I4e103: 7 e1OI4e2I4e104: 7 e1OI4e2I4e2: 15 e1OI4e2I4e89: 5 e1OI4e2I4e90: 5 e1OI4e2I4e91: 7 e1OI4e2I4e92: 7 e1OI4e2I4e94: 7 e1OI4e2I4e95: 7 e1OI4e2I4e96: 7 e1OI4e2I4e97: 7 e1OI4e2I4e98: 7 e1OI4e2I4e99: 7
2	02.01.2008 00:00:00	03.01.2008 00:00:00	2	e1	e2I2e2I3e10: 12 e2I2e2I4e100: 6 e2I2e2I4e101: 6 e2I2e2I4e102: 6 e2I2e2I4e103: 6 e2I2e2I4e104: 6 e2I2e2I4e2: 23 e2I2e2I4e91: 6 e2I2e2I4e93: 6 e2I2e2I4e94: 6 e2I2e2I4e95: 6 e2I2e2I4e96: 6 e2I2e2I4e97: 6 e2I2e2I4e98: 6 e2I2e2I4e99: 6 e2I3e1OI2e71: 5 e2I3e1OI2e72: 5 e2I3e1OI3e10: 12 e2I3e1OI3e2: 35 e2I3e1OI3e53: 5 e2I3e1OI3e54: 5 e2I3e1OI3e55: 5 e2I3e1OI3e57: 5 e2I3e1OI3e58: 5 e2I3e1OI3e59: 5 e2I3e1OI3e60: 5 e2I3e1OI3e61: 5 e2I3e1OI3e62: 5 e2I3e1OI3e63: 5 e2I3e1OI3e77: 5 e2I3e1OI3e78: 5 e2I3e1OI3e79: 5 e2I3e1OI3e80: 5 e2I3e1OI3e81: 5 e2I3e1OI3e82: 5 e2I3e1OI3e83: 5 e2I3e1OI3e85: 5 e2I3e1OI3e86: 5 e2I3e1OI3e87: 5 e2I3e1OI3e88: 5 e2I3e1OI3e89: 5 e2I3e1OI3e90: 5 e2I3e1OI3e91: 5 e2I3e1OI3e93: 5 e2I3e1OI3e94: 5 e2I3e1OI4e10: 38 e2I3e1OI4e100: 10 e2I3e1OI4e101: 10 e2I3e1OI4e102: 10 e2I3e1OI4e103: 10 e2I3e1OI4e2: 28 e2I3e1OI4e64: 6 e2I3e1OI4e65: 6 e2I3e1OI4e66: 6 e2I3e1OI4e67: 6 e2I3e1OI4e68: 6 e2I3e1OI4e69: 6 e2I3e1OI4e71: 9 e2I3e1OI4e72: 9 e2I3e1OI4e73: 9 e2I3e1OI4e74: 9 e2I3e1OI4e75: 9 e2I3e1OI4e76: 9 e2I3e1OI4e77: 9 e2I3e1OI4e79: 9 e2I3e1OI4e80: 9 e2I3e1OI4e81: 9 e2I3e1OI4e82: 9 e2I3e1OI4e83: 9 e2I3e1OI4e84: 9 e2I3e1OI4e85: 9 e2I3e1OI4e87: 9 e2I3e1OI4e88: 9 e2I3e1OI4e89: 9 e2I3e1OI4e90: 7 e2I3e1OI4e91: 7 e2I3e1OI4e92: 7 e2I3e1OI4e93: 7 e2I3e1OI4e95: 7 e2I3e1OI4e96: 7 e2I3e1OI4e97: 7 e2I3e1OI4e98: 10 e2I3e1OI4e99: 10 e2I3e1OI5e10: 8 e2I4e1OI2e10: 11 e2I4e1OI2e69: 5 e2I4e1OI2e70: 10 e2I4e1OI2e71: 10 e2I4e1OI2e72: 10 e2I4e1OI2e73: 5 e2I4e1OI2e74: 5 e2I4e1OI2e75: 5 e2I4e1OI3e2: 32 e2I4e1OI3e73: 5 e2I4e1OI3e74: 5 e2I4e1OI3e75: 5 e2I4e1OI3e76: 10 e2I4e1OI3e77: 10 e2I4e1OI3e78: 10 e2I4e1OI3e80: 10 e2I4e1OI3e81: 10 e2I4e1OI3e82: 10 e2I4e1OI3e83: 10 e2I4e1OI3e84: 10 e2I4e1OI3e85: 10 e2I4e1OI3e86: 10 e2I4e1OI3e88: 10 e2I4e1OI3e89: 10 e2I4e1OI3e90: 10 e2I4e1OI3e91: 10 e2I4e1OI3e92: 10 e2I4e1OI3e93: 10 e2I4e1OI3e94: 10 e2I4e1OI3e96: 5 e2I4e1OI3e97: 5 e2I4e1OI4e10: 32 e2I4e1OI4e100: 14 e2I4e1OI4e101: 14 e2I4e1OI4e102: 14 e2I4e1OI4e103: 14 e2I4e1OI4e2: 8 e2I4e1OI4e95: 9 e2I4e1OI4e96: 9 e2I4e1OI4e97: 9 e2I4e1OI4e98: 14 e2I4e1OI4e99: 14 e2I4e2I2e10: 7 e2I4e2I2e2: 24 e2I4e2I2e62: 8 e2I4e2I2e63: 13 e2I4e2I2e64: 13 e2I4e2I2e65: 9 e2I4e2I2e66: 9 e2I4e2I2e67: 5 e2I4e2I2e93: 6 e2I4e2I2e94: 6 e2I4e2I2e95: 12 e2I4e2I2e96: 12 e2I4e2I2e97: 6 e2I4e2I2e98: 6 e2I4e2I2e99: 6 e2I4e2I3e100: 6 e2I4e2I3e101: 12 e2I4e2I3e102: 12 e2I4e2I3e103: 12 e2I4e2I3e104: 12 e2I4e2I3e44: 6 e2I4e2I3e45: 6 e2I4e2I3e47: 6 e2I4e2I3e48: 6 e2I4e2I3e49: 6 e2I4e2I3e50: 6 e2I4e2I3e51: 6 e2I4e2I3e52: 6 e2I4e2I3e53: 6 e2I4e2I3e55: 6 e2I4e2I3e56: 6 e2I4e2I3e57: 6 e2I4e2I3e58: 6 e2I4e2I3e67: 8 e2I4e2I3e68: 13 e2I4e2I3e69: 13 e2I4e2I3e71: 13 e2I4e2I3e72: 13 e2I4e2I3e73: 13 e2I4e2I3e74: 13 e2I4e2I3e75: 13 e2I4e2I3e76: 13 e2I4e2I3e77: 13 e2I4e2I3e79: 13 e2I4e2I3e80: 13 e2I4e2I3e81: 13 e2I4e2I3e82: 13 e2I4e2I3e83: 13 e2I4e2I3e84: 13 e2I4e2I3e85: 13 e2I4e2I3e87: 9 e2I4e2I3e88: 9 e2I4e2I3e89: 5 e2I4e2I3e90: 5 e2I4e2I3e97: 6 e2I4e2I3e98: 6 e2I4e2I3e99: 6 e2I4e2I4e100: 19 e2I4e2I4e101: 19 e2I4e2I4e102: 19 e2I4e2I4e103: 19 e2I4e2I4e104: 19 e2I4e2I4e2: 52 e2I4e2I4e63: 6 e2I4e2I4e65: 6 e2I4e2I4e66: 6 e2I4e2I4e67: 6 e2I4e2I4e68: 6 e2I4e2I4e69: 6 e2I4e2I4e70: 6 e2I4e2I4e71: 6 e2I4e2I4e73: 6 e2I4e2I4e74: 6 e2I4e2I4e75: 6 e2I4e2I4e76: 6 e2I4e2I4e77: 6 e2I4e2I4e78: 6 e2I4e2I4e79: 6 e2I4e2I4e81: 6 e2I4e2I4e82: 6 e2I4e2I4e83: 6 e2I4e2I4e84: 6 e2I4e2I4e85: 6 e2I4e2I4e86: 6 e2I4e2I4e87: 10 e2I4e2I4e89: 14 e2I4e2I4e90: 14 e2I4e2I4e91: 19 e2I4e2I4e92: 19 e2I4e2I4e93: 19 e2I4e2I4e94: 19 e2I4e2I4e95: 19 e2I4e2I4e97: 19 e2I4e2I4e98: 19 e2I4e2I4e99: 19
3	03.01.2008 00:00:00	04.01.2008 00:00:00	18	e2	--
4	04.01.2008 00:00:00	05.01.2008 00:00:00	3	e3	--

Figure 38: Second part of the results of the synthetic dataset, where the occurrences multi-time interval patterns of length 2 are shown.

8 Visualization of the Approach

The sexy job in the next ten years will be statisticians…The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it—that's going to be a hugely important skill.

Hal Varian, Google's Chief Economist

As it appears evident from Figure 30 and Figure 31, one of the greatest challenge is to design a proper visualization framework (or visual analytics approach), which allows not only for presenting an overview of the available data and the results, but also lets the user easily adjust the parameters as well as interact with each visual component, during each step of the approach.

In general, one can assume to adopt some already existing visualizations and jointly apply them. As a matter of fact, some simple visualizations such as histograms or bar charts can show the quantitative aspects of the found patterns (that is how many times a certain pattern occurs), whereas more complex ones the qualitative aspects of the found patterns (that is when a certain pattern occurs) such as Arc Diagrams [152] (see Figure 16).

However, as it was outline in the related state of the art, many are the aspects to be taken into account when dealing with time-oriented data. To this aim, and to support the user during every step of the discovery process, as well as during each step of the approach, we designed a new visualization which is able to encompass in the discovery process both the analytical and the visual methods. This topic will be debated in Section 8.1 along with a discussion about what should be visualized step by step, which interaction should be allowed, and providing some elementary sketches. After that, a first prototypical implementation of such visualization will be provided in Section 8.2.

8.1 Visualization Design

In order to design a proper visual analytics approach which is able to aid the user in interacting with data, results and visual elements, we need to identify what elements must be represented. Following the approach given in Section 6.3 and its abstract representation (Figure 27), firstly we have to provide an easy interface for the input, that is, the support threshold, the selected attribute(s) or column(s) of interest and if one or more of them are discretized as well as the intervals to be taken into account (default or user defined). After that, we have to represent step by step the occurrence of events (and their amount), the occurrence of multi-time interval patterns (and their amount), and the intervals in between. Moreover, during each step an intuitive interaction has to be provide in order to access information not only about single events and multi-time interval patterns, but also about the values the events assume and about their occurrence in a social time context (that is, whether they occur on a Monday rather than on a Tuesday or on the 5th day of a month rather than on another day).

To accomplish these requirements, let's start from the input. A simple interface with some text fields and/or check box should fulfil this task. This interface will remain visible during the whole discovery process, allowing the user to adjust the parameters and to proceed step by step and back and forth. After that, we have to represent events and multi-time interval patterns. Since both of them are related to time interval information, we have firstly to display such temporal information and then the rest.

A possible solution is to display all the available intervals as concentric circles as Figure 39 shows. Along with that, a legend will provide with the information about the meaning of each interval (similarly to Figure 25).

After that we are ready to visualize the results of Step 0. Each event is represented as a bubble in the centre of the concentric circles, whose size is

proportional to the number of occurrences of the event itself. Thus, the higher number of times an event occurs, the bigger is the bubble representing it.

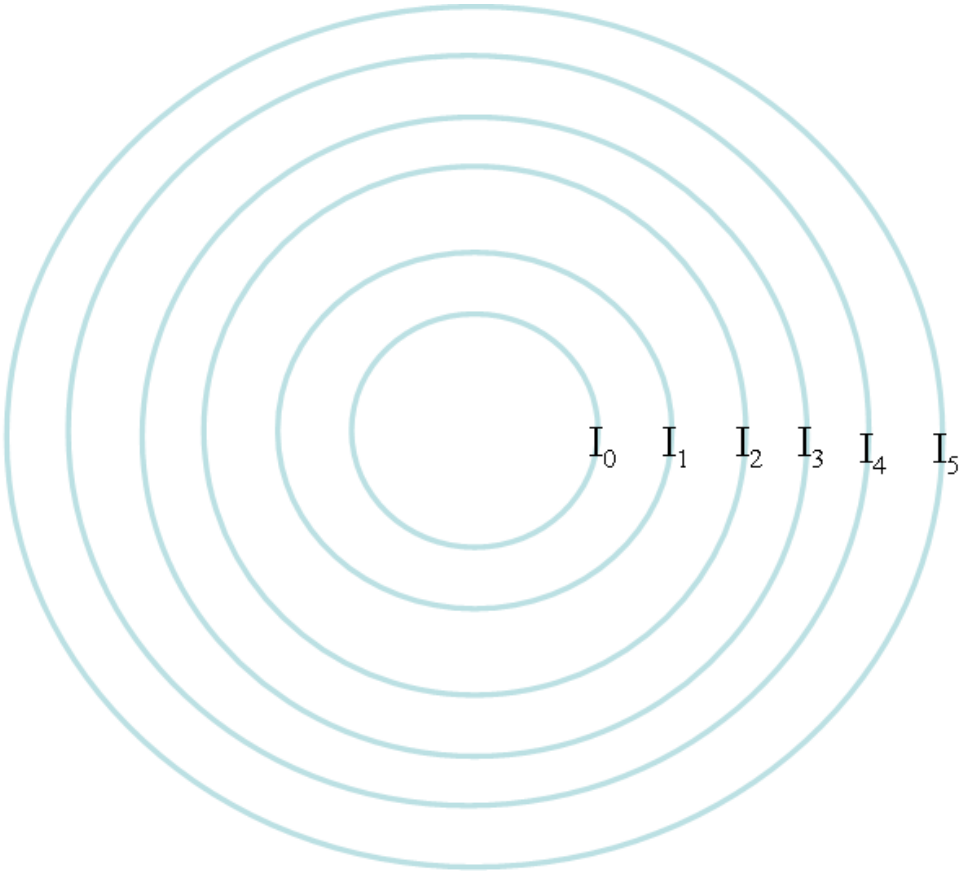

Figure 39: Concentric circles layout to represent all the available intervals.

Supposing that only the event e_3 and e_{34} are above a given threshold, we have the situation shown in Figure 40. In this way the user immediately has information about what events are above the threshold, how many times they occur, and about the data themselves. As a matter of fact, according to the granularity they are expressed, some intervals might be out of interest, and then disappear. In the case shown in Figure 40, we assume to analyze data at granularity day, therefore intervals I_0 and I_1 disappears.

To visualize the results of Step 1, we adopt a strategy similar to the previous step. Since during Step 1 we visualize multi-time interval patterns of length 1, that is, patterns composed of two events, we represent them using a segment connecting the events in the centre of the circles (as in Step 0) and events located on the interval/circle after that they occur.

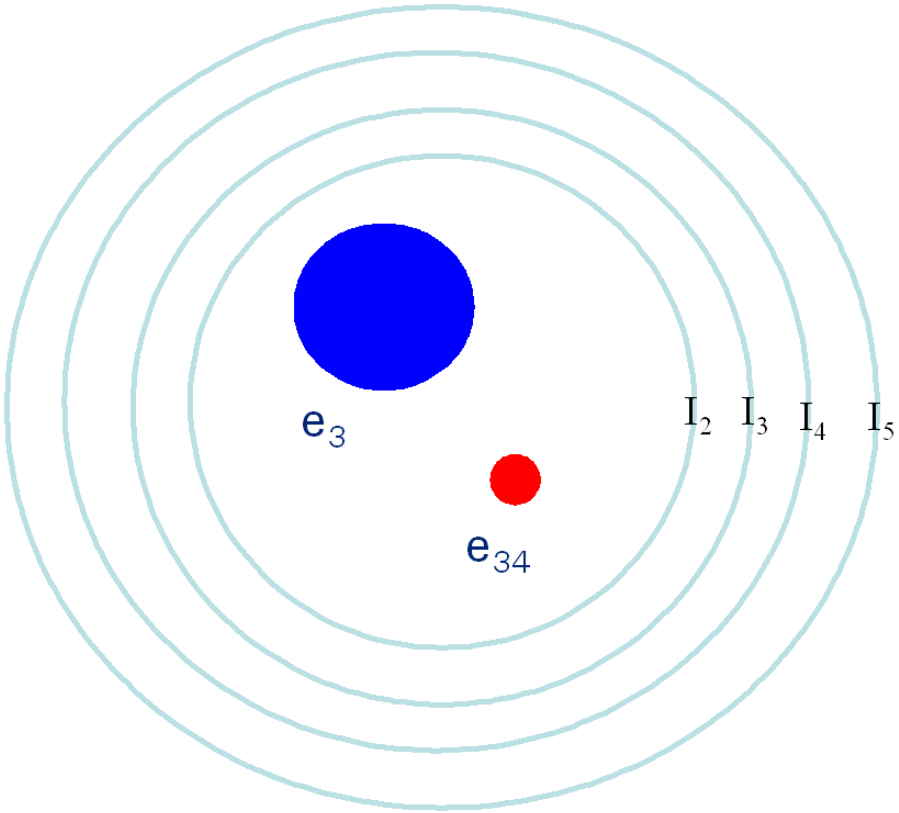

Figure 40: Graphical representation of Step 0. The user gets immediately insight about the events above the threshold (e_3 and e_{34}), how many times they occur (size of the bubble), and the granularity of the data, that is day (I_0 and I_1 disappear).

For example, the multi-time interval patterns $e_3I_3e_2$ and $e_{34}I_4e_4$ are represented as in Figure 41. Moreover, the thickness of the connecting segment corresponds to the number of occurrences of the pattern itself. Hence, the thicker the segment is, the higher number of times the pattern occurs.

Similarly to Step 1, we visualize the results of Step 2 using connecting segments between pairs of events. In this case, each multi-time interval pattern has length 2. Even in this case, the thickness of the connecting segment corresponds to the number of occurrences of the pattern itself. For example, the multi-time interval patterns $e_3I_3e_2I_5e_7$ and $e_{34}I_4e_4I_3e_9$ are represented as in Figure 42.

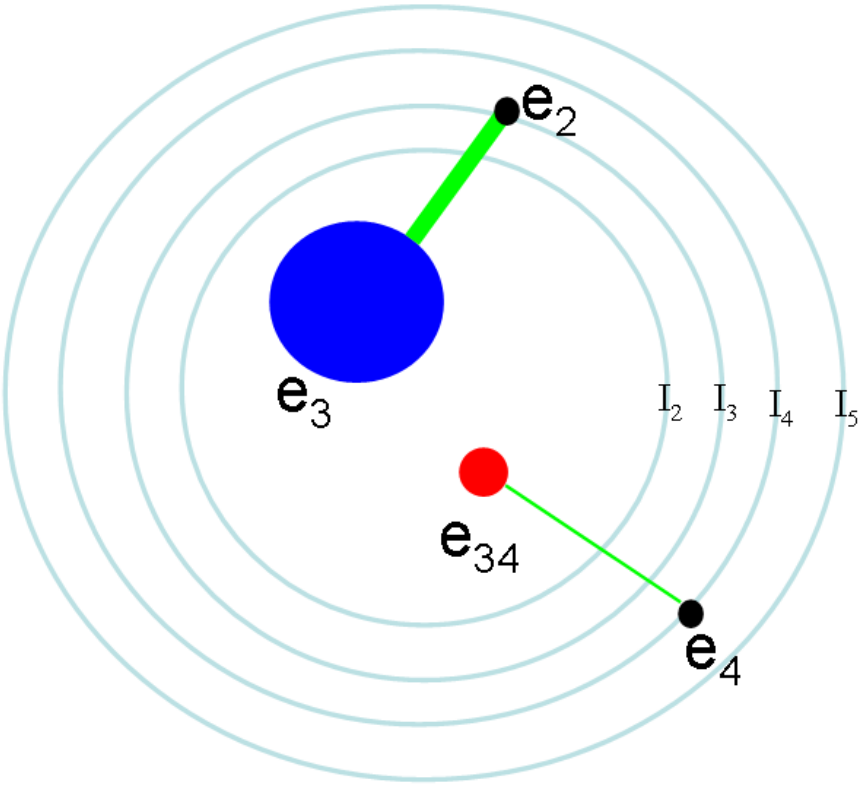

Figure 41: Graphical representation of Step 1: only multi-time patterns of length 1 are shown, such as $e_3I_3e_2$ and $e_{34}I_4e_4$. The thickness of the segment connecting the events (e.g., e_{34} and e_4) is proportional to the occurrences of the multi-time interval pattern (e.g., $e_{34}I_4e_4$).

After that, any further step follows this strategy, therefore we will not describe them in detail.

As it was stated at the beginning of this section, to perform a detailed visual analysis we need to provide intuitive interaction with each visual element. To this aim, different types of interaction will be provided. For instance, tooltips will appear when moving the mouse over an event, providing in this way information about the values of the attributes which compose it; another kind of interaction will allow to select one or more patterns and to focus on them.

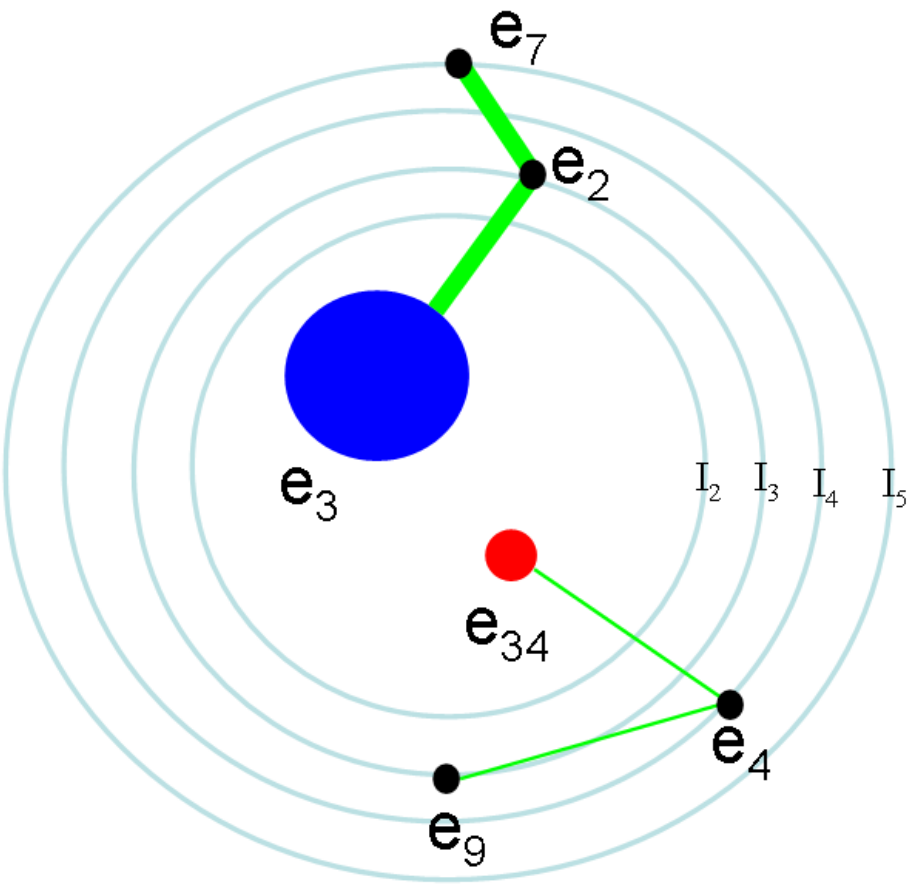

Figure 42: Graphical representation of Step 2: only multi-time patterns of length 2 are shown, such as $e_3I_3e_2I_5e_7$ and $e_{34}I_4e_4$ I_3e_9. The thickness of the segment connecting the events (e.g., e_{34} - e_4, and e_4 - e_9) is proportional to the occurrences of the multi-time interval pattern (e.g., $e_{34}I_4e_4I_3e_9$).

However, one of the most interesting one is related to social time aspects. As Figure 43 shows, two other views are provided: the first one is related to the days of week, the second one to the days of the month. Both of them are connected with the concentric circle visualization and with one another (via brushing and linking), so that when selecting an event or a multi-time interval pattern, the related temporal information about day of the week and day of the month will be highlighted. Vice versa, when selecting one or more day of the week (or day of the month), only the related events will be highlighted. Moreover, the area of each circular sector is proportional to the number of

occurrences of the related events, therefore this visually suggests how relevant a certain day of the week (day of the month) is.

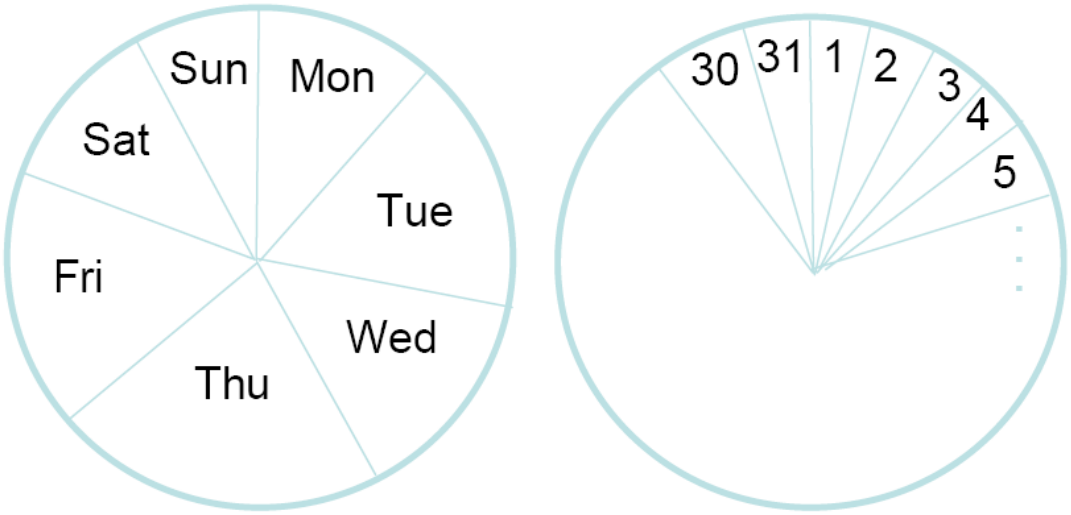

Figure 43: Graphical representation of Day of the Week (on the left) and Day of the Month (on the right). The area of each circular sector represents the amount of e.g., Thursdays or the 5th of the month during the period of time covered by the analyzed data. The bigger the area is, the more frequent is the corresponding Day of the Week (in this case Thursday) or Day of the Month. Moreover, due to fact that all the active visualizations are interconnected to provide continually a coherent view of the data, different kinds of interaction provide several insights during the whole discovery process. For instance, a click on a single event will highlight the corresponding Days of the Week or Days of the Month involved. Similarly a click on one or more Days of the Week (Days of the Month) will highlight the corresponding events in the active visualizations.

Another aspect which has been investigated in the design task is related to the positioning of the events along the circles. As a matter of fact the order in which the events are located can encompass other relevant information. For such reason, we decided to provide a clockwise numerical order as default (that is, e_0, e_1, e_2, etc.). Beside that, the user is allowed to order the events according to one or more attributes they are composed of. In detail, given for instance,

the data shown in Table 18, they can be ordered according to the index of the events, i.e., in a numerical order (which is the default option and so they appear in the table), to the number of Employee, *#E*, (that is e_2, e_6, e_9, e_{34} and e_5 – see Figure 43 left), to the *Turnover* (that is, e_2, e_6, e_{34}, e_5 and e_9 – see Figure 43 right), or according to both of them. In the last case, we generate a global score to take into account all relative orders (that is, the orders related to each attribute[23]). In the example, given 5 events, we assign 5 points to the first place in the order related to *#E*, 4 points to the second place, and so on (see column *Sc(#E)* in Table 19); then we apply the same principle for the order related to *Turnover* (column *Sc(Turn)* in Table 19), and after that we sum the two partial scores (column *Gen. Score* in Table 19), obtaining the final order (column *Order* in Table 19)[24].

Event	#E	Turnover
e_2	10	550
e_5	25	3750
e_6	12	660
e_9	14	3800
e_{34}	24	2900

Table 18: A Dataset ordered according the index of the events (numerical order).

Finally, being a concentric circle visualization to discover multi-time interval patterns, we name it "*2C*", which comes from the two C of "Concentric Circle" and sounds like the verb "to see" (to outline the fact we are able to visualize such patterns).

In the next section we will provide some screenshots of a prototypal version of *2C*.

[23] In case of categorical attributes, a lexicographical order is adopted.
[24] In case of two or more events with the same score, the order of the event index prevails.

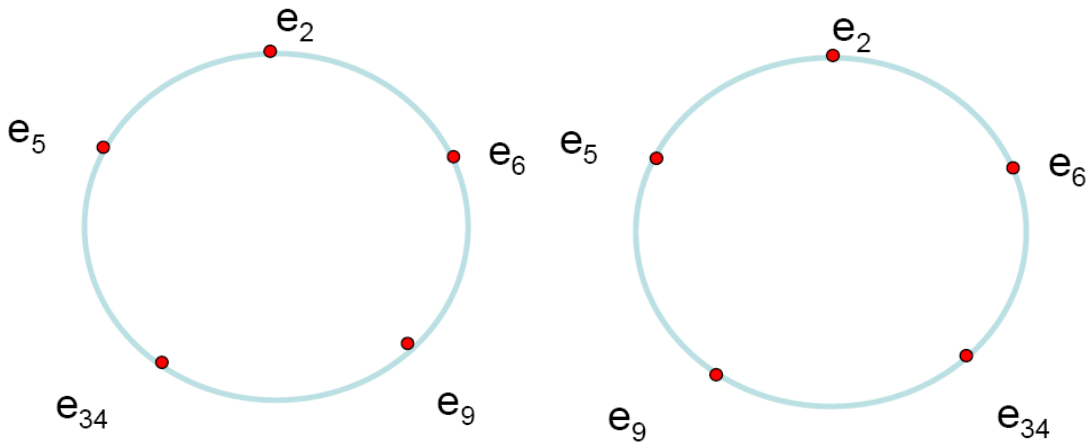

Figure 44: On the left: ordering of the event according to the number of employee; on the right: ordering of the event according to the turnover.

Event	#E	Turnover	Sc(#E)	Sc(Turn)	Gen. Score	Order
e_2	10	550	5	5	10	1st
e_5	25	3750	1	2	3	5th
e_6	12	660	4	4	8	2nd
e_9	14	3800	3	1	4	4th
e_{34}	24	2900	2	3	5	3rd

Table 19: A partial score (*Sc(#E)* and *Sc(Turn)*) is assigned to each placement in the relative orders to obtain a general score (*Gen. Score*) and from that an order which takes into account all the attributes.

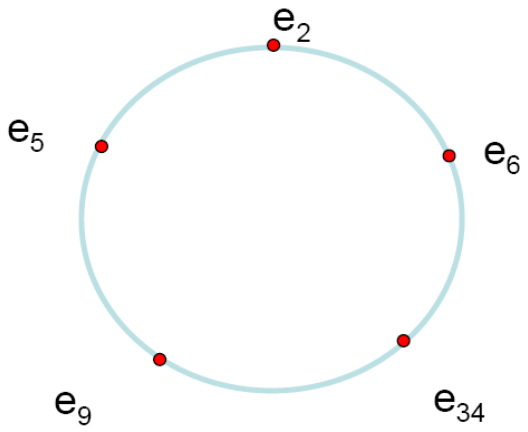

Figure 45: Ordering of events which takes into account all the attributes, based on Table 19.

8.2 2C: From Design to Prototype

In this section we show some screenshot of the first prototypical implementation of *2C*.

Figure 46 shows the user interface and the uploaded data. For simplicity we adopted the data already used for the validation in the previous chapter (see Table 5). Once the data have been uploaded, the user has to provide as input:

- the attribute(s) of interest (in this case, the attribute *Value*) and if one or more of them must be discretized providing the range of the classes (both in the proper window called "Event Configuration")
- the value of the support threshold: in this case for each step the value 10% has been inserted ("Filter Criteria")
- the time intervals to be taken into account, which can be configured as explained in Section 6.3 (in the proper window called "Intervals").

After that the user can visualize the uploaded data and the associated event (in the window called "Data/Event List") and the list of events or patterns according to the current step (in the window called "Event/Pattern List").

Each time an input parameter is modified, both of the windows are updated in order to provide a coherent view of the actual step, in this case Step 0. This fact is clearly outlined by the slide bar in the upper part of Figure 46: moving it back and forth or clicking on the back and forth buttons (indicated with a links and a right arrow respectively), the user can intuitively pass from a step of the algorithm to the following or the previous one. Once the configuration phase satisfies the user, the visualization can be activated. This situation is shown in Figure 47: all the events are above the given threshold, and the size of the bubbles represents the number of occurrence of each event. In particular, a tooltip will show the exact number of occurrence (e.g., e_0 occurs 4 times). Similarly, tooltips indicate what interval is considered (for instance, I_2 if such circle is selected), and so on, so that each visual element provides usual information to user. Moreover, the selection of an event, e.g., e_0 , highlights the graphical representation of Day of the Week and Day of the Month. In this case, e_0 occurs on Tuesday, Thursday and Friday and on the 1st, 2nd, 6th, and 8th of the month. This fact is confirmed by the proper windows showing the uploaded data in Figure 46.

The user can now proceed to the second step, Step 1, as depicted in Figure 48. According to the results of the previous chapter (Table 12), the right window shows the possible candidates. However, only the three of them are above the given threshold. Therefore we decided to outline this fact using a light yellow background which allows then to show both all possible candidates and those over the threshold. On the contrary, to better represent the results, the visualization shows only the multi-time interval patterns above the threshold. In particular, Figure 48 shows the selection of the multi-time interval pattern $e_0I_2e_0$ and $e_0I_2e_2$. As it happens also in the previous step, the thickness of the line connecting the events indicates the number of occurrences: in this case, $e_0I_2e_0$ occurs 5 times and $e_0I_2e_2$ 4 times.

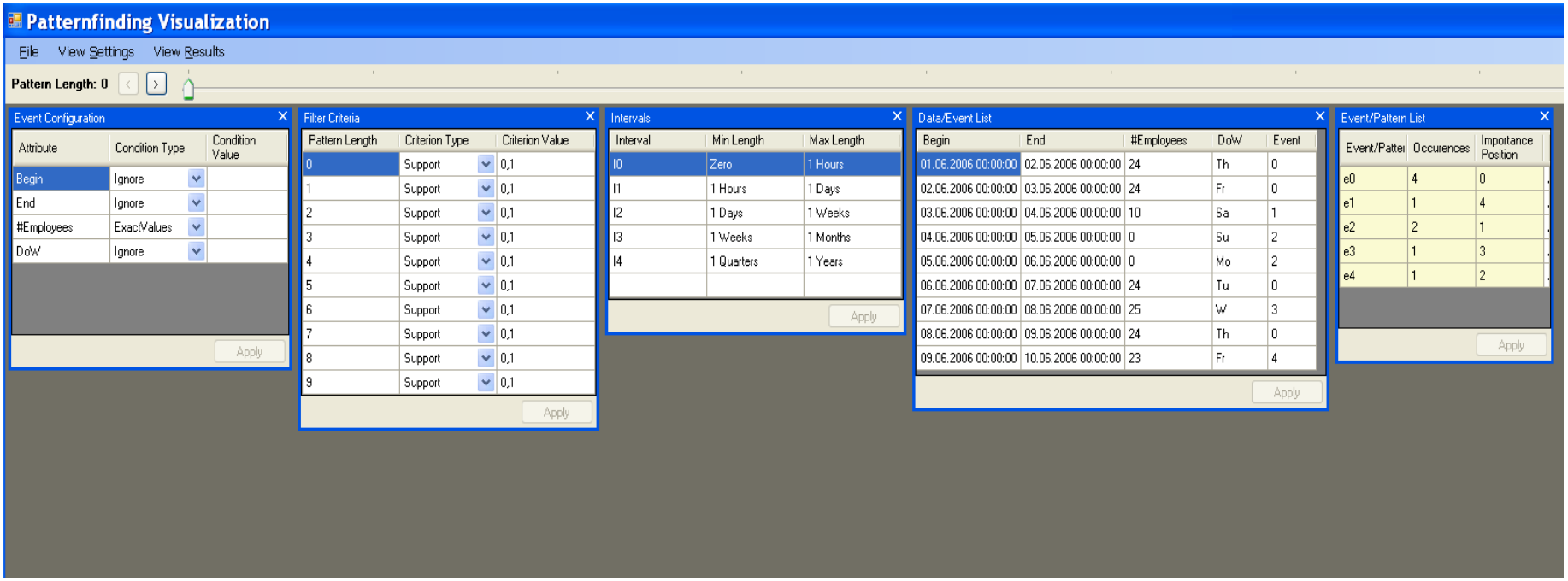

Figure 46: 2C user configuration interface and uploaded data: (from left to right) the user has the possibility to choose the attribute(s) of interest ("Event Configuration"), provide the support for each step ("Filter Criteria"), the time interval to be taken into account ("Intervals"). The remaining two windows show the actual data and their associated event ("Data/Event List"), and the list of events or patterns ("Event/Pattern List") respectively.

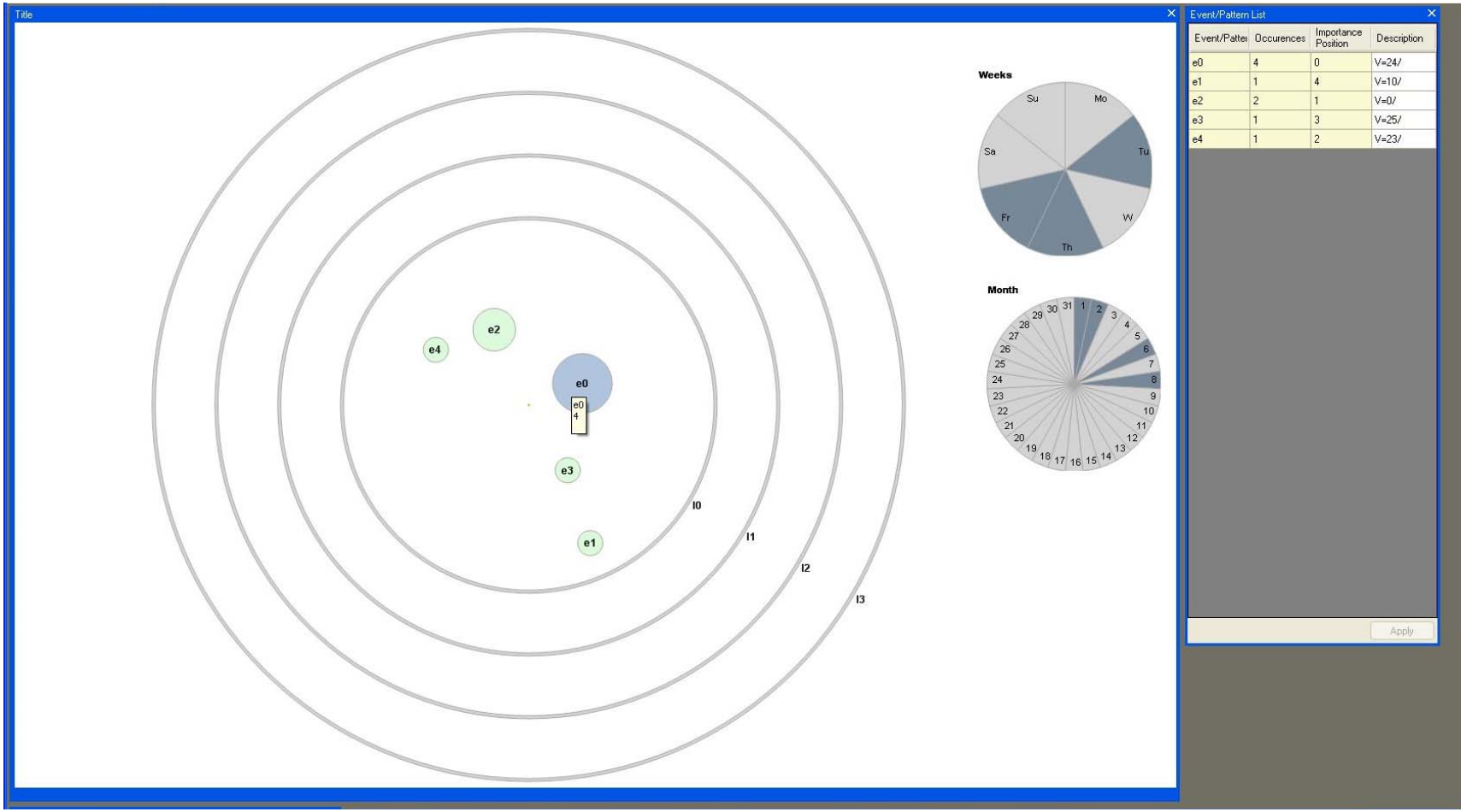

Figure 47: Events above the given threshold are shown (Step 0). The size of the bubbles represents the number of occurrence of each event. The selection of an event, e.g., e_0 , highlights the graphical representation of Day of the Week and Day of the Month. In this case, e0 occurs on Tuesday, Thursday and Friday and on the 1st , 2nd , 6th , and 8th of the month.

Similarly, the user can proceed to Step 2, whose results are shown in Figure 49. According to the results of Table 16, the right window shows the possible candidates: only a pattern ($e_0I_2e_2I_2e_0$) is above the given threshold, then it has a light yellow background and it is also represented in the proper visualization.

By repeating this process with different parameters, the user is then able to explore the available data, taking advantage of the jointly combination of both analytical and visual methods, which represents an added value of this work.

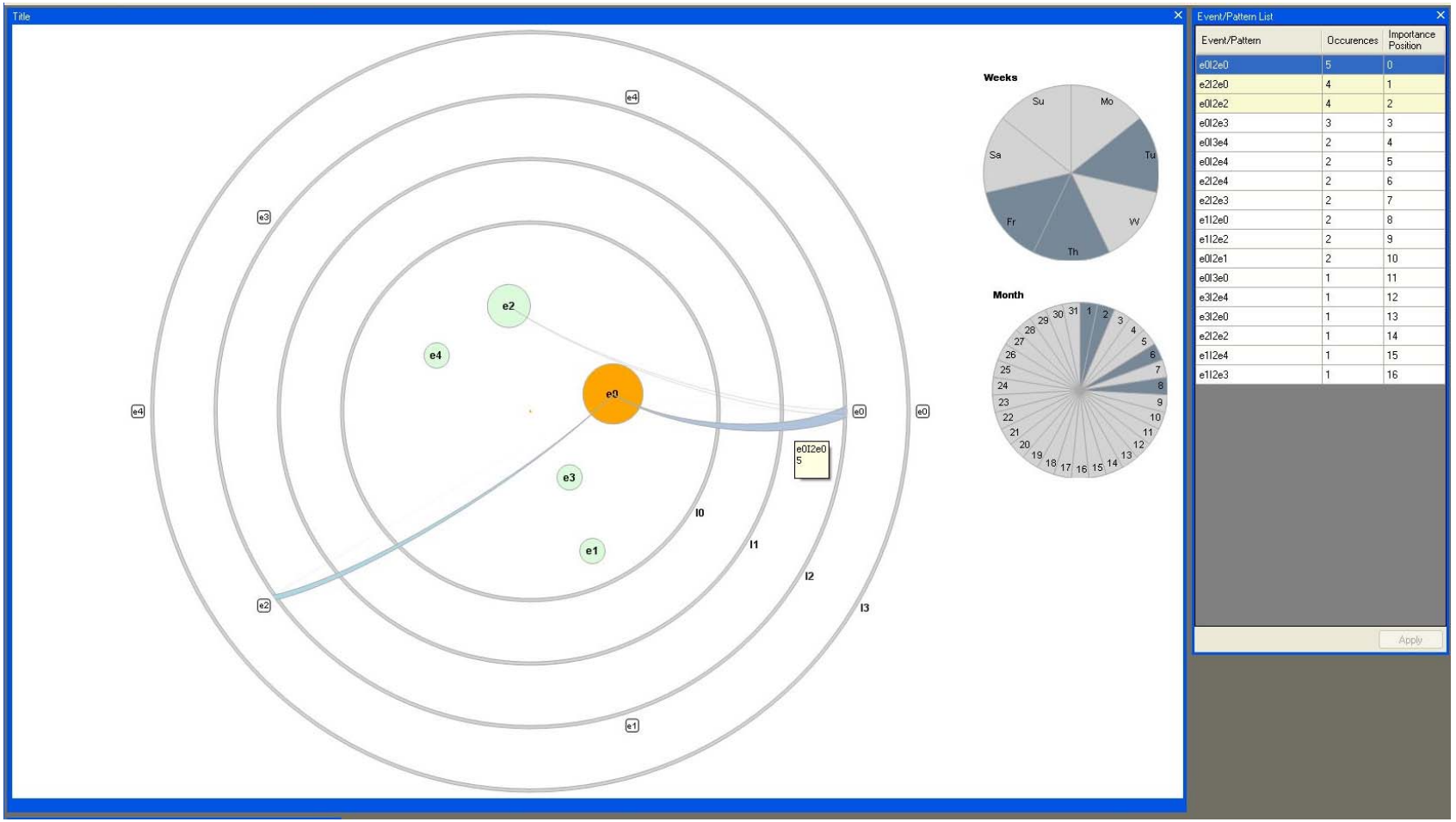

Figure 48: Results of Step 1: events and their occurrences are shown.

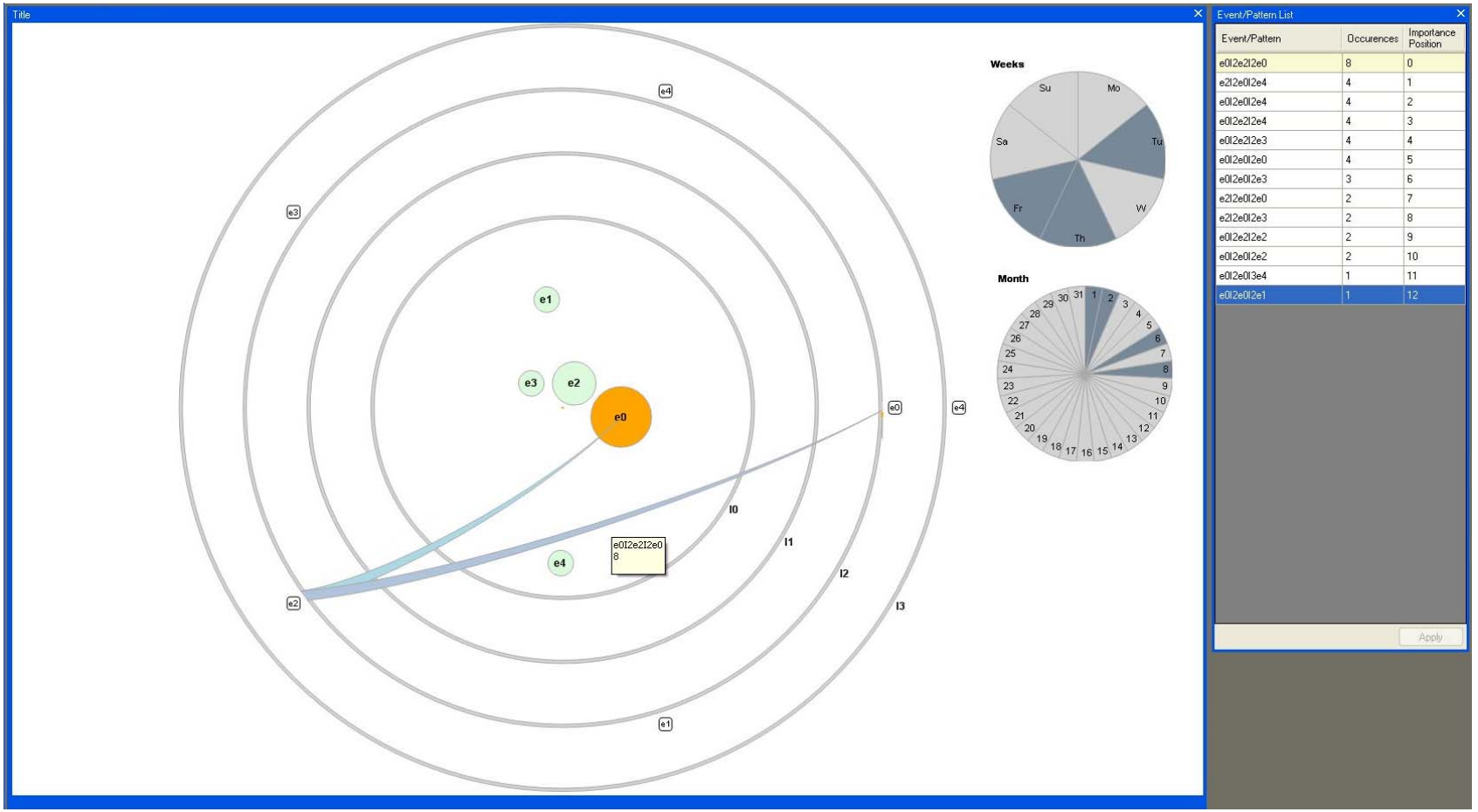

Figure 49: Results of Step 2: events and their occurrences are shown.

9 Conclusion and Future Directions

Finance, medicine, meteorology are only some of the fields where people have to deal everyday with huge amount of temporal data and with the need to analyze them. Time itself is another data dimension which is crucial especially when searching for patterns and relationships in the data. However, apart from few exceptions [5],[6],[90],[91],[92],[94] most of the Temporal Data Mining methods tend to discover a sequence of events in a certain order, but neither with any knowledge about the intervals between successive events, nor about when this sequence will reoccur. Moreover, they usually provide a sort of black box where the user sets the input and obtain some output and they rarely offer a visualization to support the whole discovery process. To cover this gap we proposed an approach which extends in particular the idea of I-Apriori algorithm proposed in [5] to non transactional databases, and try to provide a more general as well as more customizable approach in order to find *multi-time interval patterns* in time-oriented data, so that we are able to preserve the temporal information in between.

In particular, to accomplish this task we analyzed the state of the art in related research fields. Therefore, we investigated how temporal information can be modelled, by presenting three formal conceptual temporal data model and an example of system able to manage temporal data. We outlined that differently from temporal data models in temporal database research whose aim is to represent time-oriented real world information within an information system, we aim at analyzing and capturing the characteristics of many different temporal data sets, determining the most frequently occurring multi-time interval patterns and ultimately visualizing them.

Then we explored the field of Knowledge Discovery in Databases, focusing on the concepts of Data Mining, Temporal Data Mining and temporal pattern. In particular, we outlined that Temporal Data Mining methods dealing with the issue of time interval do not provide satisfactory solutions to the problem. On

the contrary, apart from few exceptions, the temporal information in between is often neglected.

After that, we examined which aspects are relevant when visualizing time-oriented data. To this aim, we presented three different categorization schemas, and a brief overview of Information Visualization techniques. We concluded that the role of the user when interacting directly with the visual representation is relevant as well as the role of interaction when dealing with time-oriented data, whereas these aspects are usually neglected or partially considered. Similarly, the adoption of multiple views and the need to include also analytical methods along with the visual ones result crucial in the process of Knowledge Discovery.

Following these considerations, we defined a time model and proposed a new definition of temporal pattern, the so called multi-time interval pattern, in order to take into account the rich structural complexity of time and to find a way to discover temporal patterns preserving the temporal information in between. Then, we illustrated a new approach which, on the one hand, extends the idea the idea of I-Apriori algorithm [5] to non transactional databases, allows to find *multi-time interval patterns* in time-oriented data, preserving the temporal information in between, and provide a more general as well as more customizable solution to the problem. On the other hand, the approach enhances in a certain sense the idea of time window to multiple time windows, by considering different time intervals.

Moreover, we provided a validation of the proposed approach. The validation is two folded, that is, either mathematical, whereas it is possible to demonstrate the correctness of the calculation of the events, or empirical, whereas it is possible to show the exactness of the calculation of the multi-time interval patterns with some test datasets. Furthermore, we applied the proposed approach in an Austrian project (DisCō) and showed the results with two different case studies.

After that, we designed a new visualization, called *2C*, which is able to encompass in the discovery process both the analytical and the visual methods,

and supports the user during every step of the discovery process, as well as during each step of the approach. Even if it is in quite advanced prototyping phase, it showed already the advantages coming from a visual exploration and interaction with time-oriented data.

In the introductory section of this thesis, we presented a number of research questions that guided our work. In the following, we will give specific answers to these questions.

9.1 Answers to Research Questions

How can Temporal Data Mining be adapted and used to analyze multivariate time-oriented data having multiple granularities in order to discover multi-time interval patterns, relations among data (previously unknown) and visually support this process? The main prerequisite is to consider the rich structural complexity of time and to find a way to discover temporal patterns preserving the temporal information in between. A suitable solution for this has been presented throughout the thesis. It allows the user to discover multi-time interval patterns and relations among data. Moreover, it permits to visually support the whole process, via intuitive interaction step by step.

How to deal with different time intervals between elements of interest in the data? This question has been explored within the proposed approach. In order to search for the most frequent multi-time interval patterns, we needed to define the intervals within which to look for the patterns. But, since it would have been too time consuming to calculate all the possible intervals between any pair of events, we decided both to give them as default (whereas we defined six not overlapping and consecutive intervals and taking into account the most meaningful granularities of social time, according to our

experience and to the suggestions coming from experts collaborating in a common project) and to give the users the possibility to configure their amount and duration. Therefore, applying these intervals, we are able to discover multi time interval patterns.

How to deal with different time granularities of multiple time series? This question has been discussed within the proposed time model. Using a lattice structure, we represented the granularities (properties of time) and how they relate with one another. After that, we modelled each property of time as set of intervals, defined by a begin time point and an end time point, and one or more labels to represent other properties. Moreover, we added the constraints that the intervals (to define the granularity) are not overlapping and (the values of) labels do not change within the interval. This seems to be a suitable solution to this question.

How to graphically represent not only the time series (overview), but also allow easy interaction with the proper visualizations (interaction and presentation of the results)? The main prerequisite is to take into account the considerations derived from the state of the art in the field of visualisation of time-oriented data. The proper solution has been designed and it is in its quite advanced stage of implementation. Following such considerations, it encompasses the importance of the user when interacting directly with the visual representation, the role of interaction itself when dealing with time-oriented data, the relevance of multiple views, and the need to include also analytical methods along with the visual ones in the process of Knowledge Discovery.

After giving answers to our research questions, we will sum up by presenting the main contributions of this thesis.

9.2 Main Contributions

To recapitulate, we have identified four main contributions to Temporal Data Mining and partially to Information Visualization research of this thesis work.

Time model: we defined a new time model, so that we are able both to consider the rich structure of time, also taking into account the social and natural contexts, and to discover temporal patterns preserving the temporal information in between.

Temporal pattern and multi-time interval pattern definition: To overcome the lack of a unique definition of pattern and to better address our issues, we proposed a new definition of temporal pattern. Based on that, we were then able to define a multi-time interval pattern and to revise this definition throughout the explanation of the approach, according to the concepts defined step by step.

The approach: Based on the given time model and pattern definition, we proposed a new approach which allows the user to discover multi-time interval patterns. The approach is highly customizable and is open to further extension.

A new visualization: We designed a new visualization (called *2C*) which allows to jointly apply analytical and visual methods in order to discover multi-time interval pattern and in general to analyze time-oriented data. The role of the user, the use of interaction, the adoption of multiple views and the

contemporary use of visual and analytical methods are the foundation of this new visual analytical approach.

After looking back by giving a summary and conclusion of our work, we will now turn around and look forward to future directions for further research.

9.3 Future Directions

Since the proposed approach involves (mainly) Temporal Data Mining aspects and (some) information visualization aspects, which are composite research areas, several future work directions are in range of our current position. We will now outline what further steps are needed and would be of interest to improve the current state of research.

2C further implementation and evaluation: The first important issue is to complete the implementation of the presented visualization. Currently we are in a quite advanced stage of prototyping, but once a stable version providing a certain amount of features is finished, we will then perform usability studies in order to obtain real users' responses and starting from them improve both the visualization and the approach.

Definition of events: Similarly to the configuration of time intervals, we can imagine that events can be user defined. Let's suppose a manager of a certain shopping mall tries to investigate the relationship between payday (set on the 15^{th} and on the 27^{th} of each month) and turnover. S/he wants to confirm the hypothesis that during the week after the payday, the turnover will be higher and more employees will be required. We can do this by replacing the first step of Section 6.3 with the following events

e_0 = {Turnover on the 15th AND Payday = Y}

e_1 = {Turnover on the 27th AND Payday = Y}

Considering them as the most relevant, we apply the proposed approach starting from the second step. In this way, we are able not only to find previously unknown patterns (explorative analysis), but also to confirm some hypotheses (confirmative analysis).

Forecast improvement: A further step to the analysis can be added, which can help in the improvement of forecasting. Once the most frequent occurring patterns are found, one can derive from them some summarized "rules" which can increase the precision of an already given forecast. For example, in the case of the flight data, from the most occurring patterns we may discover that if for a certain destination on Friday morning the error for the business class was greater than 30% then in 50% of the cases there is a 10% increase of the same error in the afternoon. In this way, exploiting this information, the customer can improve his/her forecast analysis.

Length of events: Till now we have presented examples of the approach involving events at granularity day or anyhow at a single granularity level. However, the approach and in particular the part related to the creation of the event, is influenced only by the chosen attributes and not by the length of the events themselves. Therefore, it would be of interest to investigate how to enhance the definition of events also concerning their length. For instance, an event could be one day long, another one two days and the third one one week. Whether this could result useful for some tasks is still open, but from a theoretical point view is not forbidden.

Granularity configuration and definition: Given the described time model, it could be of interest to provide the user with the possibility to define new granularities either deriving them from those already existing using some operators, or define new ones tout court. A visual support to this task would then of great help to understand how are structured the relations between granularities and would add an arrow in Figure 2 (representing the structure of the thesis) between *time model* and *visualization*. In addition, the adoption of ontology of time could be advantageous (for instance [165],[166]).

Visual analytics approach enhancement and whole development: Throughout the thesis we only sketched the structure of a visual analytic approach. Therefore, it would be necessary to better investigate any step of this approach, formalize it, and provide an abstract and more general solution to this issue.

Use of domain ontology to order and to analyze categorical data: At the moment the events shown in *2C* are ordered according to mathematical rules or using some simple methods to weight the importance the user gives to each attribute composing the events themselves. However, regarding the categorical attributes, they are treated as numerical values even if they represent more than values, they rather have a meaning. This solution encompasses a great drawback since being ordered in a lexicographical order, this can lead to misleading results. For instance, if we order lexicographically the values for the attribute *Weather*, that is, e.g., *cloudy*, *foggy*, *partially cloudy*, *rainy*, *sunny*, and we visually represent them adopting such order, one could conclude that *rainy* and *sunny* (one close to the other) are more similar than *cloudy* and *partially cloudy*. To overcome this problem, we could make use of domain ontology in order to calculate the similarity of these values and

represent their distances correctly. After that, it could even possible to analyze the data starting from such task, performing a complete explorative analysis.

Ontology visualization of categorical data: Starting from the considerations in the previous point, we could investigate how to visualize the similarity of categorical data, and include this capability in the visual analytic approach. As a matter of fact, the problem of visualizing properly the similarity of categorical data is an issue still far from being addressed.

10 Appendix I: Machine Learning and Time

As well known to the scientific community, machine learning is the research area within artificial intelligence that studies algorithms inducing models from a set of data. Even if a huge amount of work has been devoted to the description of this research area, most of them describe the data attribute to be taken into account as "static". This clearly clashes against real world applications, where usually the attributes are "dynamic", since they can vary over time and therefore time appears as really relevant.

Traditionally there have been different types of Machine Learning [167]: unsupervised learning, supervised learning and semi-supervised learning. The goal of unsupervised learning is to find interesting structure (patterns) in a given set of examples; given a set of pairs, the goal of supervised learning is to learn a mapping between the two elements of a pair (using algorithms known as generative or discriminative). Finally semi-supervised learning (SSL) represents a hybrid combination between unsupervised and supervised learning: in addition to unlabeled data, some supervised information are provided, leading to an exploitation of both labelled and unlabeled data.

As a matter of fact, since labelled instances require the efforts of experienced human annotators, they are often difficult, expensive, or time consuming to obtain. On the contrary unlabeled data may be relatively easy to collect: SSL use large amount of unlabeled data along with the labelled data in order to build better classifiers. Thus it achieves higher accuracy with less human effort and currently represents a challenge for new approaches dealing with time and machine learning.

In the following some references dealing with temporal data and the three above classification are given.

10.1 Unsupervised and Supervised Learning

Kadous [168] presents a technique for temporal classification using machine learning: it consists of extraction and parameterization of sub-events from the training instances, which allows feature construction for a subsequent learning process.

Gonzales et al. [169] present a supervised classification method for temporal series, which extends inductive logic programming systems with predicates to deal with time series classification tasks.

Geurts [170] proposes to extend classifiers in order to allow them to detect local shift invariant properties or patterns in time series and combine the binary test classification into decision trees by using piecewise constant modelling to increase the efficiency of search for candidate patterns.

Rüping et al. [171] outline that the use of support vector machines for different temporal learning tasks is particularly suited for high-dimensional data.

Manganaris et al. [172], [173] suggest using Bayesian network to classify time series according to their features, starting from pre classified examples.

Hsu et al. [174] describe a system for learning heterogeneous time series using artificial neural networks and Bayesian networks in order to alleviate the prediction task during critical events ("crisis monitoring").

10.2 Semi-supervised Learning

As previously said, Semi-Supervised Learning exploits both labelled and unlabeled data. There are a plenty of methods which have been proposed and can be divided into five classes according to their assumptions: SSL with generative models, low density separation, graph-based models, co-training methods and self-training methods (a complete description can be found in [167], [175]).

In the following some references about SSL and different models are proposed.

Nigam [176] describes generative model to deal with text classification tasks providing high accuracy in the results.

Bruce [177] presents a Bayesian network approach to semi-supervised learning, so that the parameters of a probability model are estimated using Bayesian techniques and then used to perform the classification task in the field of word-sense disambiguation.

Amini et al. [178] propose discriminating algorithms using both labelled and unlabeled data for text classification and text summarization tasks. Similarly Vittaut et al. [179] extends these algorithms from text classification to email spam detection. Finally, Wei et al. [180] apply a nearest neighbour with Euclidean distance classifier (and a stopping criterion) for building time series classifiers to deal with text classifications tasks or yoga motions.

11 Appendix II: Terms and Definitions

In the following, we provide a summary of terms and their definitions used in this work.

Data Mining: It consists on the search for patterns of interest in a particular representational form or a set of such representations: such as classification rules and trees, clustering or association rules [1].

Data Mining Outcome: The various types of Data Mining in the sense of what the outcomes will be [29]. Usually the term Data Mining Task is used as synonym for Data Mining tasks and Data Mining methods. Examples of Data Mining Outcomes are classification, clustering, and so on.

Data Mining Task: see Data Mining Outcome.

Event: It represents the occurrence of a certain value given attribute (or when a given set of attributes assume a set of values).

Granularity: In the time model we proposed, the term granularity is used as synonym for property of time. It is represented as set of intervals, defined by a begin time point and an end time point, and one or more labels to represent other properties (such as *day of week*, *day of month* and so on). Moreover, for each granularity hold the constraints that (1) the intervals (to define the granularity) are not overlapping and (2) (the values of) labels do not change within the interval.

Knowledge Discovery in Databases: Knowledge Discovery in Databases is a non-trivial process of identifying patterns in data that are: valid (in the statistic sense), novel (at least to the system and to the user), potentially

useful (for a given application) and ultimately understandable (immediately or after post processing) [31].

Length of a multi-time interval pattern: It is the number of subpatterns (having form $e_x\ I_a\ e_y$) composing the whole pattern.

Multi-time interval pattern: Be D a dataset of dimension $p\ x\ t$ with attributes $A_1,...,A_t$, and values $V_{A1,1}$, $V_{A1,2,}...V_{A1,p,}$ $V_{A2,1,...}$ $V_{At,p}$,E the set of possible events, having the form $A_h = V_{Ah,1} \wedge A_{h+1} = V_{Ah+1,1} \wedge ...$, $1 \leq h \leq t$, and TI the set of all the possible time intervals between any pair of events, then $M = e_0\ I_0\ e_1\ I_1\ e_z\ I_2 \ldots\ e_m\ I_n$ is a *multi-time interval pattern* if

- $e_j \in E$ for $1 \leq j \leq m$
- $I_k \in TI$ for $1 \leq k \leq n$
- $k \leq w \Rightarrow I_k$ (consecutive) precedes I_w
- $e_j\ I_b\ e_q\ I_c\ e_s\ \neg \Rightarrow\ I_b$ (consecutive) precedes I_c

Pattern: "a nonempty set of time intervals (with properties) or time-points, their relationship (as well as to corresponding data), meeting a certain degree of interest", where the *degree of interest* is a measure of interestingness, which may be context or user dependent.

Support: It represents the relative frequency or number of times a value occurs within the database (usually expressed as percentage).

Bibliography

[1] U. Fayyad, P. Smyth, G. Piatetetsky-Shapiro and R. Uthurusamy, Advances in knowledge discovery and data mining, AAAI Press/The MIT Press, 1996

[2] B.Thuraisingham, Data Maning: technologies, techniques, tools, and Trends, CRC Press, 1999

[3] W.J.Frawley, G. Piatetsky-Shapiro and C.J. Matheus, Knowledge Discovery in Databases: An Overview, AAAI/MIT press, 1991

[4] W. Aigner, A. Bertone, S. Miksch, H. Schumann and C. Tominski, Towards a Conceptual Framework for Visual Analytics of Time and Time-Oriented Data, Henderson, S. G.; Biller, B.; Hsieh, M.; Shortle, J.; Tew, J. D. & Barton, R. R. (eds.), Proceedings of the 2007 Winter Simulation Conference, pp. 721-729, Washington, D.C., invited Paper, December, 2007

[5] Y. L. Chen, M. C. Chiang and M. T. Kao, Discovering time-interval sequential patterns in sequence databases, Expert Systems with Applications, 25 (3), pp.343-354, 2003

[6] H.R. Yang, Discovering multi-time-interval sequential patterns in sequence database, http://thesis.lib.ncu.edu.tw/ETD-db/ETD-search-c/view_etd?URN=91423007 (accessed on 02nd February 2009)

[7] C. S. Jensen, and R.T.Snodgrass, Semantics of Time-Varying Information. Information Systems, 21(4), pp.311–352, 1996

[8] Combi, C. and Pozzi, G,. HMAP - A Temporal Data Model Managing Intervals with Different Granularities and Indeterminacy from Natural Language Sentences. The VLDB Journal, 9(4), pp. 294–311, 2001

[9] C. Bettini , X. S. Wang , S. Jajodia, A general framework for time granularity and its application to temporal reasoning, Annals of Mathematics and Artificial Intelligence, v.22 n.1-2, pp. 29-58, 1998

[10] C. Bettini, S. Jajodia, X.S. Wang, Time Granularities in Databases, Data Mining, and Temporal Reasoning, Springer-Verlag, July 2000

[11] B. Urgun, C. E. Dyreson, N. Kline, J. K. Miller, R. T. Snodgrass, M. D. Soo, and C. S. Jensen, Integrating Multiple Calendars using τZaman, Software—Practice&Experience, Volume 37 , Issue 3, March 2007

[12] W. Aigner, Visualization of Time and Time-Oriented Information: Challenges and Conceptual Design, Vienna University of Technology, Institute of Software Technology and Interactive Systems, PhD Thesis (2006)

[13] C.S. Jensen, and R. T. Snodgrass, Temporal Data Management. IEEE Transactions on Knowledge and Data Engineering, 11(1), pp. 36–44, 1999.

[14] H. Gregersen, and C. S. Jensen, Temporal Entity-Relationship Models - A Survey. Knowledge and Data Engineering, 11(3), pp. 464–497, 1999

[15] T. Myrach, Temporale Datenbanken in betrieblichen Informationssystemen. Teubner, 1. edition, 2005

[16] I. A. Goralwalla, Y. Leontiev, M. T. Özsu, and D. Szafron, Modeling Temporal Primitives: Back to Basics. In CIKM, pp. 24–31, 1997

[17] I. A. Goralwalla, M. T. Özsu, and D. Szafron, An Object-Oriented Framework for Temporal Data Models, pages 1–35. Springer, 1998

[18] A. Kaiser, Die Modellierung zeitbezogener Daten. Peter Lang Verlag, 2000.

[19] J. Drucker, J. B. Nowviskie, Temporal Modelling: Conceptualization of Temporal Relations for Humanities Scholarship. Technical report, Institute for Advanced Technology in Humanities, University of Virginia, 2003

[20] A. U. Frank, Qualitative Temporal Reasoning in GIS - Ordered Time Scales. In 6th International Symposium on Spatial Data Handling, volume 1, pp. 410–430, Edinburgh, Scotland, 1994

[21] J. F. Allen, Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11), pp. 832–843, 1983

[22] J. F. Allen, Time and Time Again: The Many Ways to Represent Time. International Journal of Intelligent Systems, 6, pp.341–355, 1991

[23] D. Peuquet, Making Space for Time: Issues in Space-Time Data Representation, GeoInformatica, 5(1), pp. 11–32, 2001

[24] A. U. Frank, Spatial and Temporal Reasoning in Geographic Information Systems, chapter Different Types of "Times" in GIS, pages 40–62. Oxford University Press, New York, 1998

[25] Y. Wu, S. Jajodia, and X. S. Wang, Temporal Database Bibliography Update, In Temporal Databases, Dagstuhl, pp. 338–366, 1997

[26] J.C. Augusto, Temporal Reasoning for Decision Support in Medicine. Artificial Intelligence in Medicine, 33(1), pp. 1–24, 2005

[27] I. Merlo, Extending the ODMG Object Model with Temporal and Active Capabilities, PhD Thesis, Università di Genova, 2001

[28] C. S. Jensen and C. E. Dyreson (editors). A Consensus Glossary of Temporal Database Concepts – February 1998 Version. In Temporal Databases: Research and Practice, Lecture Notes in Computer Science 1399, pp. 367-405. Springer-Verlag, 1998

[29] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth: From Data Mining to Knowledge Discovery in Databases. AI Magazine 17(3), pp. 37-54 (1996)

[30] G. Piatetsky-Shapiro, Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop. AI Magazine 11(5), pp. 68–70, 1991

[31] M. Ankerst, Visual Data Mining, *PhD Thesis*, Ludwig-Maximilians-Universität, München, 2001

[32] M. J. A. Berry, G. Linoff, *Data Mining Techniques*, New York, John Wiley & Sons, 1997

[33] Glossary of Terms, Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Machine Learning, 30, pp. 271-274, 1998

[34] S. I. Weiss, C. Kulikowski, Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Networks, Machine Learning, and Expert Systems, San Francisco, Calif., Morgan Kaufmann Publisher, 1991

[35] A. K. Jain, R. C. Dubes, Algorithms for Clustering Data, Englewood Cliffs, N.J., Prentice-Hall, 1988

[36] P. Cheeseman, J. Stutz, Bayesian Classification (AUTOCLASS): Theory and Results, Advances in Knowledge Discovery and Data Mining, eds.,1996

[37] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, I. Verkamo, Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, pp. 307-328. Menlo Park, Calif., AAAI Press, 1996

[38] R. Zembowicz, J. Zytkow, From Contingency Tables to Various Forms of Knowledge in Databases, *Advances in Knowledge Discovery and Data Mining*, eds. U. Fayyad, G. Piatetsky-Shapiro, P.Smyth, and R. Uthurusamy, pp. 329-351, Menlo Park, Calif., AAAI Press, 1996

[39] W. Kloesgen, A Multipattern and Multistrategy Discovery Assistant, Advances in Knowledge Discovery and Data Mining, eds., U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R.Uthurusamy, pp. 249-271. Menlo Park, Calif., AAAI Press, 1996

[40] O. Guyon, N. Matic, N. Vapnik, Discovering Informative Patterns and Data Cleaning, Advances in Knowledge Discovery and Data Mining, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, pp. 495–516, Menlo Park, Calif., AAAI Press, 1996

[41] D. Berndt, J. Clifford, Finding Patterns in Time Series: A Dynamic Programming Approach, Advances in Knowledge Discovery and Data Mining, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, pp. 229-248, Menlo Park, Calif., AAAI Press, 1996

[42] D. Heckerman, Bayesian Networks for Knowledge Discovery, *Advances in Knowledge Discovery and Data Mining*, eds. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, pp. 273-306, Menlo Park, Calif., AAAI Press,1996

[43] A Data Mining Glossary http://www.thearling.com/glossary.htm (accessed on, 18th April 2009)

[44] S. Laxman, and P. Sastry. 2006. A Survey of Temporal Data Mining. Sadhana 31, pp. 173–198

[45] G.Box, G.M. Jenkins, and G. C. Reinsel. Time Series Analysis: Forecasting and Control, third edition. Prentice-Hall, 1994

[46] T.C. Mills, Time Series Techniques for Economists. Cambridge University Press, 1990

[47] J. Lin, E. Keogh, S. Lonardi, J.P. Lankford, D.M. Nystrom, Visually Mining and Monitoring Massive Time Series. In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, WA, Aug 22-25, 2004

[48] C. M. Antunes and A. L. Oliveira. Temporal data mining: An overview. In KDD Workshop on Temporal Data Mining, pp. 1-13, San Francisco, CA, 26 August 2001

[49] R. Agrawal, K. Lin, H. Sawhney, K., Shim, Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. VLDB, pp.490-501, 1995

[50] L. Lin, T. Risch, Querying Continuous Time Sequences. VLDB, pp.170-181, 1998

[51] V. Guralnik. , and J. Srivastava: Event Detection from Time Series Data. KDD, pp.33-42, 1999

[52] G. Das, D. Gonopulos, H. Mannila, Finding Similar Time Series. PKDD, pp.88-100, 1997

[53] R. Agrawal, C. Faloutsos, A. Swami, A.: Efficient similarity search in sequence databases. FODO - Foundations of Data Organization and Algorithms Proceedings, pp.69-84, 1993

[54] K. Chan, W. Fu, Efficient Time Series Matching by Wavelets. ICDE, pp.126-133, 1999

[55] R. Agrawal., P. Giuseppe, W.L. Edward, M., Zait, Querying Shapes of Histories. VLDB pp.502-514, 1995

[56] J. Roddick, M. Spiliopoulou, A Survey of Temporal Knowledge Discovery Paradigms and Methods. In IEEE Transactions of Knowledge and Data Engineering, vol. 13, 2001

[57] G. Das, H. Mannila, P. Smyth, Rule Discovery from Time Series. KDD, pp.16-22, 1998

[58] C. Giles, S. Lawrence, A.C. Tsoi, Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference. Machine Learning, vol. 44, pp.161-184, 2001

[59] G. Guimarães: The Induction of Temporal Grammatical Rules from Multivariate Time Series. ICGI, pp.127-140, 2000

[60] X. Ge, P. Smyth, Deformable Markov Model Templates for Time Series Pattern Matching. KDD, pp.81-90, 2000

[61] H. Mannila, H.Toivonen and A.Inkeri Verkamo, Discovering frequent episodes in sequences. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD '95). Montreal, Canada, pp. 210–215, 1995

[62] C. Bettini, S.X. Wang, S. Jagodia, J.L. Lin, Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences. IEEE Transactions on Knowledge and Data Engineering, vol. 10, nº 2, pp. 222-237, 1998

[63] V. Guralnik, D. Wijesakera, J. Srivastava, Pattern Directed Mining of Sequence Data. KDD, pp.51-57, 1998

[64] R. Agrawal, R. Srikant, Mining sequential patterns. ICDE, pp.3-14, 1995

[65] J. Han, J. Pei,,Y. Yin, Mining Frequent Patterns without Candidate Generation. ACM SIGMOD Int. Conf. on Management of Data, pp.1-12, 2000.

[66] Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast Subsequence Matching in Time-Series Databases. ACM SIGMOD Int. Conf. on Management of Data, pp.419-429, 1994

[67] R. Agrawal, K. Lin, H. Sawhney, K., Shim, Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. VLDB, pp.490-501, 1995

[68] D. Berndt, J. Clifford, Finding Patterns in Time Series: a Dynamic Programming Approach. In: Fayyad, U., Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI Press. pp.229-248, 1996

[69] E. Keogh, M. Pazzani, Scaling up Dynamic Time Warping to Massive Datasets. Proc. Principles and Practice of Knowledge Discovery in Databases, pp.1-11, 1999

[70] B. Yi, H. Jagadish, C. Faloutsos, Efficient Retrieval of Similar Time Sequences Under Time Warping. ICDE, pp.201-208, 1998

[71] Y. Huang, P. Yu,Adaptive Query Processing for Time Series Data. KDD, pp. 282-286, 1999

[72] H. Mannila, P. Ronkainen, Similarity of Event Sequences. In TIME, pp.136-139, 1997

[73] R. Agrawal, R. Srikant Fast algorithms for mining association rules in large databases. In Proc.20th Int. Conf. on Very Large Data Bases, pp 487–499, 1994

[74] B. Özden, S. Ramaswamy, A. Silberschatz, Cyclic association rules. ICDE, pp.412-421, 1998

[75] S. Ramaswamy, S. Mahajan, A. Silberschatz, On the Discovery of Interesting Patterns in Association Rules. VLDB, pp.368-379, 1998

[76] B.H. Juang, L. Rabiner, Fundamentals of speech recognition, Englewood Cliffs, NJ: Prentice Hall, 1993

[77] D. O'Shaughnessy, Speech communications: Human and machine, Piscataway, NJ: IEEE Press, 2003

[78] T. Darrell , A. Pentland A Space-time gestures. In Proc. 1993 IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition (CVPR'93), pp 335–340, 1993

[79] A. Kundu, Y. He, P. Bahl, Word recognition and word hypothesis generation for handwritten script: A Hidden Markov Model based approach. In Proc. 1988 IEEE Comput. Soc. Conf. on ComputerVision and Pattern Recognition (CVPR'88), pp 457–462, 1998

[80] C.C. Tappert ,C.Y. Suen, T. Wakahara, The state of the art in on-line handwriting recognition. IEEE Trans. Pattern Anal. Machine Intell. 12, pp.787–808, 1990

[81] P. Smyth, Probabilistic Model-Based Clustering of Multivariate and Sequential Data. Artificial Intelligence and Statistics, pp.299-304, 1999

[82] A. Ketterlin, Clustering Sequences of Complex Objects. KDD, pp.215-218, 1997

[83] D. Fisher, D. Knowledge Acquisition Via Incremental Conceptual Clustering. Machine Learning vol. 2, pp.139-172, 1987

[84] Box, G. E. P., Jenkins, G.M., Reinsel, G. C. Time Series Analysis: Forecasting and control, Singapore,Pearson Education Inc.Chatfield 1996

[85] C. Chatfield C. The Analysis of Time Series. 5th edn., New York, Chapman and Hall, 1996

[86] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: Data mining, inference and prediction, New York, Springer-Verlag, 2001

[87] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT), Avignon, France, 1996

[88] R.Villafane, K.A. Hua, D. Tran, and B. Maulik, Mining Interval Time Series, LNCS Vol. 1676, Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery, pp.: 318 - 330, 1999

[89] M.S. Magnusson, ,Discovering Hidden Time Patterns in Behavior: T-Patterns and their Detection. Behavior Research Methods, Instruments and Computers, 32(1): pp. 93-110, 2000

[90] M. Yoshida, T. Iizuka, H. Shiohara,and M. Ishiguro, Mining sequential patterns including time intervals, In "Data Mining and Knowledge Discovery: Theory, Tools and Technology II (SPIE Conference)", 2000

[91] A. Vautier, M.-O. Cordier, and R. Quiniou. An inductive database for mining temporal patterns in event sequences. In Workshop on Mining Spatial and Temporal Data, 2000

[92] F. Giannotti , M. Nanni , D. Pedreschi , F. Pinelli, Mining sequences with temporal annotations, Proceedings of the 2006 ACM symposium on Applied computing, , Dijon, France, April 23-27, 2006

[93] Y.L.Chen and S. Wu: Mining Temporal Patterns from Sequence Database of Interval-Based Events. FSKD06, pp. 586-595, 2006

[94] Y.-H. Hu, T.C. Huang, H.-R. Yang, Y.-L. Chen, On mining multi-time-interval sequential patterns, Data & Knowledge Engineering, Vol. 68, No. 10, (23 October 2009), pp. 1112-1127, 2009

[95] S. Ma, J.L. Hellerstein, Mining partially periodic event patterns with unknown periods, in: Proceedings of the 17th International Conference on Data Engineering, pp. 205–214, 2001

[96] J. Han, W. Gong, Y. Yin, Mining segment-wise periodic patterns in time-related databases, in: Proceedings of the 1998 International Conference on Knowledge Discovery and Data Mining, pp. 214–218, 1998

[97] J. Han, G. Dong, Y. Yin, Efficient mining of partial periodic patterns in time series database, in: Proceedings of the 1999 International Conference on Data Engineering, pp. 106–115, 1999

[98] Z. Stejic, Y. Takamn, K. Hirota, Genetic algorithm-based relevance feedback for image retrieval using local similarity patterns, Information Processing and Management 39 (1), pp. 1–23, 2003

[99] C. Li, P.S. Yu, V. Castelli, Hierarchyscan: a hierarchical similarity search algorithm for databases of long sequences, in: Proceedings of the 12th International Conference on Data Engineering, pp. 546–553, 1996

[100] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu, Mining Access Patterns Efficiently from Web Logs, Proc. 2000 Pacific-Asia Conf. Knowledge Discovery and Data Mining, pp. 396–407, 2000

[101] M.S. Chen, J.S. Park, P.S. Yu, Efficient data mining for path traversal patterns, IEEE Transactions on Knowledge and Data Engineering 10 (2), pp. 209–221, 1998

[102] C.C. Yu, Y.L. Chen, Mining sequential patterns from multi-dimensional sequence data, IEEE Transactions on Knowledge and Data Engineering 17 (1), pp. 136–140, 2005

[103] G. Song, G. Salvendy, A framework for reuse of user experience in web browsing, Behaviour and Information Technology 22 (2), pp.79-90, 2003

[104] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, Freespan: frequent pattern-projected sequential pattern mining, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 355–359, 2000

[105] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, Mining sequential patterns by pattern-growth: the prefixspan approach,IEEE Transaction on Knowledge and Data Engineering 16 (11), pp. 1424–1440, 2004

[106] M.J. Zaki, SPADE: an efficient algorithm for mining frequent sequences, Machine Learning Journal 42 (1/2), pp. 31–60, 2001

[107] L. Sacchi, C. Larizza, C. Combi, and R. Bellazzi, Data mining with Temporal Abstractions: learning rules from time series, Data Mining and Knowledge Discovery archive, Volume 15 , Issue 2 (October 2007), pp. 217 – 247, 2007

[108] P. Hong and T. S. Huang, Automatic Temporal Pattern Extraction and Association, ICPR 2002

[109] C. Bettini, X.S. Wang, S. Jajodia. Testing Complex Temporal Relationships Involving Multiple Granularities and Its Application to Data Mining. Proc. of ACM PODS, pp. 68-78, Montreal, 1996.

[110] H. Mannila, H. Toivonen, A.I. Verkamo, Discovery of Frequent Episodes in Event Sequences, Data Mining and Knowledge Discovery, Volume 1, Issue 3, pp. 259 – 289, 1997

[111] F. Höppner: Discovery of core episodes from sequences -- using generalization for defragmentation of rule sets. In Pattern Detection and Discovery in Data Mining, LNAI 2447, pp. 199-213. London, England, Sept 2002.

[112] R.J. Povinelli and Xin Feng, A New Temporal Pattern Identification Method for Characterization and Prediction of Complex Time Series Events (2003), IEEE Transactions on Knowledge and Data Engineering, Volume 15 , Issue 2, pp. 339 - 352, 2003

[113] Y. Shahar. Knowledge-Based Temporal Interpolation. Journal of Experimental and Theoretical Artificial Intelligence, pp. 11:123-144. 1996

[114] M. O. Ward, XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data, in: Proceedings of IEEE Visualization, Washington, USA, 1994

[115] J. Kolojejchick, S. F. Roth, P. Lucas, Information Appliances and Tools in Visage, IEEE Computer Graphics and Applications 17 (4), pp. 32–41, 1997

[116] A. Inselberg, A Survey of Parallel Coordinates, in: H.-C. Hege, K. Polthier (Eds.), Mathematical Visualization, Springer, Berlin, Germany, Ch. 3, pp. 167–179, 1998

[117] B. Shneiderman, Direct Manipulation: A Step Beyond Programming Languages,IEEE Computer 16 (8), pp. 57–69, 1983

[118] S. F. Silva, T. Catarci, Visualization of Linear Time-Oriented Data: A Survey, in: Proceedings of International Conference on Web Information Systems Engineering, Hong Kong, China, 2000

[119] W. Müller, H. Schumann, Visualization Methods for Time-dependent Data - An Overview, in: Proceedings of Winter Simulation Conference,New Orleans, USA, 2003

[120] W. Aigner, S. Miksch, W. Müller,H. Schumann, and C. Tominski, Visualizing Time-Oriented Data - A Systematic View, Computers & Graphics, Vol. 31, No. 3, p. 401-409, Elsevier, June, 2007

[121] W. Aigner, S. Miksch, W. Müller,H. Schumann, and C. Tominski, Visual Methods for Analyzing Time-Oriented Data, Transactions on Visualization and Computer Graphics, Vol. 14, No. 1, pp. 47-60, IEEE Computer Society Press, 2008

[122] C.Plaisant, B.Milash, A.Rose, S.Widoff, and B.Shneiderman, Lifelines: Visualizing Personal Histories. In Proc. of ACM CHI'96, Vancouver, Canada, April, pp. 221-227, 1996

[123] J.D.Mackinlay, G.G.Robertson, and S.K.Card. The Perspective Wall: Detail and Context Smoothly Integrated. In Proc. of ACM CHI'91, New York, pp. 173-179, 1991

[124] T.Carpendale, M. Sheelagh, A.Fall, D.J.Cowperthwaite, J.Fall, and F.D.Fracchia Visual Access for Landscape Event based Temporal Data. In Proc. of Visualization '96, p. 425-428, 1996

[125] G.M.Karam. Visualization Using Timelines. In Proc. of Intl Symposium on Software Testing and Analysis (ISSTA), 1994, also in SIGSOFT, ACM Software Engineering Notes, 1994

[126] C.Plaisant, B. Shneiderman, and R. Muhslin. An Information Architecture to Support the Visualization of Personal Histories. Information Processing & Management, Vol. 34, N. 5, pp. 581-597, 1998

[127] V.Kumar, R.Furuta, and R.B.Allen. Metadata Visualization for Digital Libraries: Interactive Timeline Editing and Review, In Proc. of ACM Digital Libraries, Pittsburgh, USA, pp. 126-133, 1998

[128] E.Freeman, and D.Gelernter. Lifestreams: A Storage Model for Personal Data. ACM Sigmod Bulletin, March 1996

[129] R.Xiong, and J.Donath. PeopleGarden: Creating Data Portraits for Users. In Proc of ACM UIST'99, Asheville, NC, pp. 37-44, 1999

[130] E.H.Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S.K. Card, Visualizing the Evolution of Web Ecologies, In Proc. of ACM CHI'98, Los Angeles CA USA, pp. 400-407, 1998

[131] S.Hibino, and E.A.Rundensteiner. MMVIS: Design and Implementation of a Multimedia Visual Information Seeking Environment. In Proc. of ACM Multimedia 96, Boston MA, USA, pp. 75-86, 1996

[132] H.Luttermann, and M. Grauer. VRML History: Storing and Browsing Temporal 3D-Worlds. In Proc. of ACM VRML'99, Paderborn Germany, pp. 153-160, 1999

[133] D. Beard, M. Palaniappan, A. Humm, D. Banks, A. Nair and Y.P. Shan. A Visual Calendar for Scheduling Group Meetings. In Proc. of ACM CSCW'90, pp 279-290, 1990

[134] J.D.Mackinlay, G.G.Robertson, R.DeLine. Developing Calendar Visualizers for the Information Visualizer. In Proc. of ACM UIST'94, pp. 109-118, 1994

[135] J.Rekimoto. TimeScape: A Time-Machine for the Desktop Environment. In Proc. of ACM CHI'99 Extended Abstracts, pp. 180-181, 1999

[136] D.Keim. Visual Techniques for Exploring Databases, Invited Tutorial. In Proc. Int. Conf. on Knowledge Discovery in Databases (KDD'97), Newport Beach, CA, 1997

[137] J.V.Carlis, J.A.Konstan. Interactive Visualization of Serial Periodic Data. In Proc. of ACM UIST'98, San Francisco, CA, pp. 29-38, 1998

[138] J.Fall, A.Fall Seles: A Spatially Explicit Landscape Event Simulator. In Proc. GIS/Environmental Modeling Conference, pp. 104-112, 1996

[139] T. T. Elvins, VisFiles – Presentation Techniques for Time-series Data. Computer Graphics, 31(2): 14-16, 1997

[140] J. Bertin, Semiology of Graphics: Diagrams, Networks, Maps. University of Wisconsin Press, USA, 1983

[141] J. Mackinlay, Automating the Design of Graphical Presentations of Relational Information. ACM Transactions on Graphics. 5 (2), pp. 110-141, 1986

[142] C. J. Minard, Des Tableaux Graphiques et des Cartes Figuratives. E. Thunot et Cie, Paris. ENPC: 3386/C161, BNF: Tolbiac, V-16168; 8 p. and plate(s), 1861

[143] R. L. Harris, Information Graphics – A Comprehensive Illustrated Reference. Management Graphics, Atlanta, Georgia, USA, 1996

[144] S. Havre, B. Hetzler, and L. Nowell, ThemeRiver: Visualizing Theme Changes over Time. In Proc. IEEE Symposium on Information Visualization 2000 (InfoVis '00). IEEE Computer Society, Los Alamitos, USA, pp. 115-123, 2000

[145] J. V. Carlis, and J.A. Konstan, Interactive visualization of serial periodic data. Proc. 11th annual ACM symposium on User interface software and technology, San Francisco, California, USA, pp. 29-38, 1998

[146] M. Weber, M. Alexa, and W. Müller, Visualizing Time-Series on Spirals. In Proc. IEEE Symposium on Information Visualization 2001 (InfoVis '01), San Diego, USA, pp. 7-13, 2001

[147] J. J. van Wijk, and E. van Selow, Cluster and Calendar-based Visualization of Time Series Data. In Proc. IEEE Symposium on Information Visualization (InfoVis '99). (ed) G. Wills, D. Keim. IEEE Computer Society, pp. 4-9, 1999

[148] C. Tominski, J. Abello, and H. Schumann, "Axes-Based Visualizations with Radial Layouts," in Proc. of ACM Symp. on Applied Computing. ACM Press, pp. 1242–1247, 2004

[149] R. Dörner, P. Grimm, and Ch. Seiler, Agents and Virtual Environments for Communication and Decision Training for Emergencies. (ed) C. Sierra et al., Proc. of the Fourth International Conference on Autonomous Agents. ACM Press, New York, pp. 50-51, 2000

[150] W. Aigner, S. Miksch, B. Thurnher, S. Biffl, PlanningLines: Novel Glyphs for Representing Temporal Uncertainties and their Evaluation, in: Proceedings of International Conference on Information Visualisation, London, UK, 2005

[151] R. Kosara, F. Bendix, and H. Hauser, TimeHistograms for Large, Time-Dependent Data, in Proceedings of Joint Eurographics – IEEE TCVG Symposium on Visualization (VisSym04), O. Deussen, C. Hansen, D. Keim, and D. Saupe, Eds., pp. 45–54, 2004

[152] M. Wattenberg, Arc Diagrams: Visualizing Structure in Strings, In Proceedings of the IEEE Symposium on Information Visualization (InfoVis02), pp.110–116, IEEE, 2002

[153] M. Ankerst, D. A. Keim, H.-p. Kriegel: Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data, Proc. Visualization '95, Atlanta, GA, December, 1995, pp. 279-286

[154] H. Hochheiser, Interactive Querying of Time Series Data, in CHI '02, extended abstracts on Human Factors in Computing Systems. ACM Press, pp. 552–553, 2002

[155] L. Chittaro, C. Combi, Visualizing Queries on Databases of Temporal Histories: New Metaphors and their Evaluation, Data and Knowledge Engineering 44 (2), pp. 239–264, 2003

[156] E. Keogh, S. Lonardi, and W. Chiu, Finding Surprising Patterns in a Time Series Database In Linear Time and Space, In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. July 23 - 26, 2002. Edmonton, Alberta, Canada, pp.550-556, 2002

[157] F Guil, JM Juarez, R Marin, Mining Possibilistic Temporal Constraint Networks: A Case Study in Diagnostic Evolution at Intensive Care Units, Proceedings of IDAMAP Workshops, Verona (Italy), August 25-26, 2006

[158] S Badaloni, M Falda, Mining Temporal Characterization of Ill-known Diseases, Proceedings of IDAMAP Workshops, Verona (Italy), August 25-26, 2006

[159] J. Lin, E. Keogh, S. Lonardi, J.P. Lankford, D.M. Nystrom, Visually Mining and Monitoring Massive Time Series. In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, WA, Aug 22-25, 2004

[160] Chittaro L., Combi C., Trapasso G., Data Mining on Temporal Data: a visual Approach and its Clinical Application to Hemodialysis, Journal of Visual Languages and Computing, vol.14, no.6, pp.591-620, December 2003

[161] http://statigrafix.com/ (accessed on 18th April 2009)

[162] R.T Snodgrass, The TSQL2 Temporal Query Language. Kluwer, 1995

[163] D.A. Keim, F. Mansmann, J. Schneidewind and H. Ziegler, Challenges in Visual Data Analysis, Information Visualization (IV 2006) ,Invited Paper, July 5-7, London, United Kingdom, Ieee Press, 2006

[164] F. Bodon and L. Schmidt-Thieme. The relation of closed itemset mining, complete pruning strategies and item ordering in apriori-based fim algorithms. In Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05), Porto, Portugal, 2005

[165] F. Pan and J.R. Hobbs, Time in OWL-S. In Proceedings of the AAAI Spring Symposium on Semantic Web Services, Stanford University, CA, pp. 29-36, 2004

[166] J.R. Hobbs and F. Pan., An Ontology of Time for the Semantic Web. ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing, Vol. 3, No. 1, pp. 66-85,March 2004

[167] O. Chapelle, A. Zien, and B. Schölkopf, Semi-supervised learning.MIT Press, 2006

[168] M. W. Kadous, Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series. PhD thesis, School of Computer Science & Engineering, University of New South Wales, 2002

[169] C. A. González and J.J. Rodríguez Diez, Time Series Classification by Boosting Interval Based Literals. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 11: pp. 2-11, 2000

[170] P. Geurts, Pattern extraction for time series classification. Lecture Notes in Computer Science 2168, pp. 115–127, 2001

[171] S. Rüping and K. Morik, Support Vector Machines and Learning about Time, in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2003

[172] S. Manganaris and D. Fisher, Learning Time Series for Intelligent Monitoring, Third Intl. Symp. on AI, Robotics, and Automation for Space, pp. 71-74, 1994

[173] S. Manganaris, Supervised classification with temporal data, PhD Thesis, Vanderbilt University, 1997

[174] W. H. Hsu, N. D. Gettings, V. E. Lease, Y. Pan, and D. C. Wilkins, Crisis Monitoring: Methods for Heterogeneous Time Series Learning, In Proceedings of the International Workshop on Multistrategy Learning (MSL-98), Milan, Italy, June 1998

[175] X. Zhou, Semi-supervised learning literature survey http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf (accessed on 18th April 2009)

[176] K. Nigam. Using Unlabeled Data to Improve Text Classification. Doctoral Dissertation, Computer Science Department, Carnegie Mellon University. Technical Report CMU-CS-01-126, 2001

[177] R. Bruce, Semi-supervised learning using prior probabilities and EM, Presented at the International Joint Conference of AI Workshop on Text Learning: Beyond Supervision, Seattle, Washington, 2001

[178] M.R. Amini and Patrick Gallinari, The use of unlabeled data to improve supervised learning for text summarization, SIGIR 2002, pp.105-112, 2002

[179] J.N. Vittaut, M.R. Amini and Patrick Gallinari, Learning Classification with Both Labeled and Unlabeled Data, ECML 2002, pp. 468-479, 2002

[180] L. Wei and E. Keogh, Semi-Supervised Time Series Classification, In Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), pp. 748 - 753, Philadelphia, PA, U.S.A., August 20-23, 2006

The end

(but the end is the beginning of something else)

Palombo, perde l'equilibrio... rischia il dribbling, perde palla, arriva Milanetto..... Attenzione ! ..... due contro..... ZERO.... avanza Palladino ! .... davanti ha Castellazzi... porta vuota !... Milito !.. Milito !.. RETE !, RETE !, RETE ! GAME OVER SUL DERBY ! IMPAZZISCE MARASSI !

Maurizio Compagnoni, Comment on Genoa-Sampdoria 3-1, Serie A 2008-2009, 3rd goal Milito (hattrick).

Curriculum Vitae

Address

ALESSIO BERTONE

Floßgasse 11/13

A-1020 Wien

E-mail: alessio.bertone@gmail.com

Date of Birth

April 18th ,1974

Genoa, Italy

Education

October 2006 to December 2009

Ph.D. studies in Computer Science

at Vienna University of Technology, Austria

Finished in November 2009

Ph.D. Thesis: A Matter of Time: Multi-time Interval Pattern Discovery to Preserve the Temporal Information in between

Supervisors: Silvia Miksch, Margit Pohl

April 23rd ,2001

Laurea in Computer Science, DISI, Dipartimento di Informatica e Scienze dell'Informazione (Università degli Studi di Genova, Genova), (MSc equipollent)

Publications

Please see

http://www.donau-uni.ac.at/alessio.bertone

Job Experience

Since June 2006

Scientific researcher at the Department of Information & Knowledge Engineering (ike), Danube University Krems

December 2001 to December 2005

Fellowship by IMATI-CNR, Dep. of Genoa

May 2001 to August 2001

Collaboration with Jackson Libri for the translation (English-Italian) of the book "Modern operating systems 2nd Edition" (A.S. Tanenbaum)

July 1998 to October 1998

Collaboration (150 hours) with Istituto di Storia Medioevale (Via Lomellini, Genova) in order to update the database of the library and solve hardware and software problems of the existing computers