Discovering complex quantitative dependencies between interval-based state streams

Amine El Ouassouli

N° d'ordre NNT : 2020LYSEI061

Thèse de doctorat de l'Université de Lyon

opérée au sein de

L'INSA de Lyon

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité / discipline de doctorat : Informatique

Soutenue publiquement le 24/07/2020 par

Amine El Ouassouli

Discovering Complex Quantitative Dependencies between Interval-based State Streams

Devant le jury composé de:

Pr. Farouk Toumani	Université Blaise Pascal	Rapporteur
Dr. Reza Akbarinia	INRIA, LIRMM	Rapporteur
Pr. Fréderique Laforest	INSA-Lyon	Examinatrice
Pr. Karine Zeitouni	UVSQ, Université Paris-Saclay	Examinatrice
Dr. Lionel Robinault	Foxstream	Co-directeur de thèse
Dr. Vasile-Marian Scuturici	INSA-Lyon	Directeur de thèse

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** http://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr INSA : R. GOURDON	**M. Stéphane DANIELE** Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX directeur@edchimie-lyon.fr
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** http://edeea.ec-lyon.fr Sec. : M.C. HAVGOUDOUKIAN ecole-doctorale.eea@ec-lyon.fr	**M. Gérard SCORLETTI** École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 gerard.scorletti@ec-lyon.fr
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** http://e2m2.universite-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES secretariat.e2m2@univ-lyon1.fr	**M. Philippe NORMAND** UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX philippe.normand@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** http://www.ediss-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Curien - 3ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tel : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tel : 04.72.44.83.69 hamamache.kheddouci@univ-lyon1.fr
Matériaux	**MATÉRIAUX DE LYON** http://ed34.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction ed.materiaux@insa-lyon.fr	**M. Jean-Yves BUFFIÈRE** INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** http://edmega.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX jocelyn.bonjour@insa-lyon.fr
ScSo	**ScSo*** http://ed483.univ-lyon2.fr Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 veronique.cervantes@univ-lyon2.fr	**M. Christian MONTES** Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 christian.montes@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Abstract

The increasing utilization of sensor devices in addition to human-given data make it possible to capture real world systems complexity through rich temporal descriptions. More precisely, the usage of a multitude of data sources types (devices, humans) allows to monitor an environment by describing the evolution in time of several of its dimensions through data streams. One core characteristic of such configurations is heterogeneity that appears at different levels of the data generation process: data sources, time models and data models. In such context, one challenging task for monitoring systems is to discover non-trivial temporal knowledge that is directly actionable and suitable for human interpretation. In this thesis, we firstly propose to use a Temporal Abstraction (TA) approach to express information given by heterogeneous raw data streams with a unified interval-based representation, called state streams. A state reports on a high level environment configuration that is of interest for an application domain. It is defined as a predicate involving data from one or several data sources. Such approach solves problems introduced by heterogeneity, provides a high level pattern vocabulary and also permits also to integrate expert(s) knowledge into the discovery process. Second, we introduced the Complex Temporal Dependencies (CTD) that is a quantitative interval-based pattern model. It is defined similarly to a conjunctive normal form and allows to express complex temporal relations between states. Contrary to the majority of existing pattern models, a CTD is evaluated with automatic statistical assessment of streams intersection avoiding the use of any significance user-given parameter. Third, we proposed CTD-Miner a first efficient CTD mining framework. CTD-Miner performs an incremental dependency construction. CTD-Miner benefits from pruning techniques based on a statistical correspondence relationship that aims to accelerate the exploration search space by reducing redundant information and to provide a more usable result set. Finally, as discovering pairwise significant time lag dependencies is a core operation in the CTD-Miner process, we proposed the Interval Time Lag Discovery (ITLD) algorithm. ITLD is based on a confidence variation heuristic that permits to reduce the complexity of the discovery process from quadratic to linear w.r.t a temporal constraint Δ on time lags. To evaluate our approach, we implemented a motion simulation tool permitting to build data sets corresponding to a wide range of configurations. We also gathered data from a real world experiment using video cameras and real time video processing methods to build a real motion data set. Experiments showed that ITLD provides efficiently more accurate results in comparison with existing approaches. Hence, ITLD enhances significantly the accuracy, performances and scalability of CTD-Miner. The encouraging results given by CTD-Miner on our real world motion data set suggests that it is possible to integrate insights given by real time video processing approaches in a knowledge discovery process opening interesting perspectives for monitoring smart environments.

Resumé

Les avancées significatives qu'ont connu les technologies de capteurs, leur utilisation croissante ainsi que leur intégration dans les systèmes d'information permettent d'obtenir des descriptions temporelles riches d'environnements réels. L'information générée par de telles sources de données peut être qualifiée d'hétérogène sur plusieurs plans: types de mesures physiques, domaines et primitives temporelles, modèles de données etc. Dans ce contexte, l'application de méthodes de fouille de motifs constitue une opportunité pour la découverte de relations temporelles non-triviales, directement utilisables et facilement interprétables décrivant des phénomènes complexes. Nous proposons d'utiliser un ensemble d'abstraction temporelles pour construire une représentation unifiée, sous forme des flux d'intervalles (ou états), de l'information générée par un système hétérogène. Cette approche permet d'obtenir une description temporelle de l'environnent étudié à travers des attributs (ou états), dits de haut niveau, pouvant être utilisés dans la construction des motifs temporelles. A partir de cette représentation, nous nous intéressons à la découverte de dépendances temporelles quantitatives (avec information de délais) entre plusieurs flux d'intervalles. Nous introduisons le modèle de dépendances Complex Temporal Dependency (CTD) défini de manière similaire à une forme normale conjonctive. Ce modèle permets d'exprimer un ensemble riche de relations temporelles complexes. Pour ce modèle de dépendances nous proposons des algorithmes efficaces de découverte : CTD-Miner et ITLD - Interval Time Lag Discovery. Finalement, nous évaluons les performances de notre proposition ainsi que la qualité des résultats obtenus à travers des données issues de simulations ainsi que des données réelles collectées à partir de caméras et d'analyse vidéo.

"À force de sacrifier l'essentiel pour l'urgence, on finit par oublier l'urgence de l'essentiel."

Edgard Morin - La Méthode

Contents

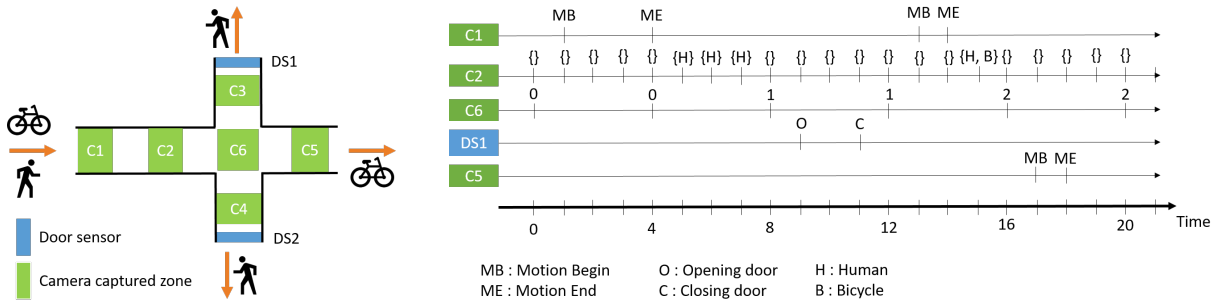
1 Introduction

1.1 Heterogeneous data-driven perception of real world

The remarkable advances in sensors, network technologies, and low storage costs had lead to the massive and rapidly growing use of sensor devices. This is demonstrated by the spectacular growth of sensors and the"smart object" industry in the last decade. Current numbers and future projections of used sensor devices are estimated with dozens of billions [113] showing the great interest arousing around sensing technologies. Their dissemination and democratisation had touched public and private areas at very different scales (e.g., human body, transportation engines, buildings, cities) and concerned a broad range of application areas (e.g., healthcare, manufacturing, energy).

This technological background makes it possible to acquire vast amounts of diverse temporal data that describe environments through a wide range of attributes. Traditional sensors, more sophisticated devices (e.g., mobile phones, video cameras), and humans (through manual reporting) can be seen as potential data sources that may provide a data-driven perception of their environment. Moreover, advances in information retrieval techniques make it possible to obtain high-level information from complex data with improving accuracy that approaches some specific human cognitive capabilities. For instance, recent computer vision technologies with image and video processing allow to visually "sense" an environment. The availability of a wide variety of data sources offers the opportunity to build informationally rich temporal descriptions of environments through a set of data streams that we label as heterogeneous. **Heterogeneity is an opportunity.** This claim is based upon a simple observation: the more qualitatively diverse is a dataset, the more it may include enough information to capture the potential complexity of temporal phenomena/behaviours occurring in a given environment. To illustrate this, let us consider the following toy example.

We describe in Figure 1.1.1.a a set of corridors equipped with a sensor system composed of door sensors and video cameras. "Real-time" video processing algorithms are applied to perform motion detection, object counts and object recognition: C_1, C_3, C_4 and C_5 runs a motion detection algorithm; C_2 provides object recognition; C_6 counts moving objects. A sample of data streams obtained by this sensor configuration is described in Fig-



(a) Corridors equipped with door sensors and video cameras

(b) Some attribute data streams gathered from the sensor system depicted in Fig.3.1.1a

Figure 1.1.1: A sensed environment example and corresponding attribute streams

ure 1.1.1.b. Using motion events permits to describe 3 main qualitative trajectories: *"C1 then C3"*, *"C1 then C4"* and *"C1 then C5"*. The inclusion of door sensors data permits provides a piece of additional information. Only trajectories *"C1 then C3"* and *"C1 then C4"* lead to a door openings:*"C1 then C3" then "DS1 open"*, *"C1 then C4" then "DS2 open"*, *"C1 then C5"*. A More insightful description of this environment is possible with the inclusion of object recognition data from camera 2: *"C1 then C3" then "pedestrian in C2" then "DS1 open"*, *"C1 then C4" then "pedestrian in C2" then "DS2 open"*, *"C1 then "bicycle in C2" then C5"*. The description provided by the latter three temporal patterns permits to understand that door openings are correlated with the object type "pedestrian" and that passage through C_5 is observed exclusively by a bicycle. This simple example permits to emphasise that the inclusion of the object recognition information in the description of typical trajectories in this environment permits to acknowledge different behaviours types. This information type can then be used to predict the outcome of a trajectory ("bicycle in C_2" is always followed by motion in C_5) and trigger an action (display a helmet advertisement in a screen near C_5 in 1 second).

We believe that the use of multiple dimensions enhances descriptions expressivity and allows a better understanding, interpretability of patterns describing temporal phenomena occurring in a given sensed environment. The effort of our thesis work leans towards contributing to solving the following general problem:

Problem 1

How to discover non-trivial and directly actionable temporal knowledge from a set of heterogeneous data streams?

Before tackling this challenging problem, let us first discuss the scope and definitions of the research field to which this dissertation belongs: Data Mining and Knowledge Discovery.

1.2 Data mining and knowledge discovery: an overview

Frawley et al. [51] defined Knowledge Discovery (KD) as follows:

> *"The non-trivial extraction of implicit, previously unknown, and potentially useful information from data."*

In this perspective, *data* refers to a set of facts F (e.g a database) and *information* to a pattern: a statement S in a predefined language L describing relationships among a subset of F, assessed with a certainty measure C. Certainty is defined as the ability of patterns to fit non-analysed data. A *pattern* is considered as discovered knowledge if it is enough certain and interesting according to the user. This definition permits to highlight several characteristics of a Knowledge Discovery process. First, a knowledge discovery is *non-trivial* in that it has to *"do more than blindly*

compute" basic "*statistics*" [51] (e.g average) but involves more sophisticated relationships. The second underlying and recurrent characteristic of Knowledge discovery is that of *novelty* and *non-triviality* of discovered insights ([40, 41, 51]). Indeed, the main aim of a knowledge discovery task to provide previously unknown insights from data. While sharing similar ideas, Fayyad et al. [40] provided a slightly different definition:

> "*The non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*".

In that work, the notion of *pattern* can also describe a model applicable to a subset of data. The interesting about this pattern definition is that a KD task is to discover underlying structures that may rule data sets which is a more complex task than simple descriptions. Hence, this definition is general enough to include a large range of insight models (e.g., motifs, sequential patterns, clusters, association rules, dependencies). KD is also referred to as an *interactive, and iterative process* including numerous steps like understanding application domain and result interpretation as users' task or data sampling and data mining as computational tasks. What is suggested here is that every KD process involves users input at different stages. We emphasise here the distinction made between Knowledge Discovery and Data Mining. In the point of view of Fayyad et al. [40] data mining is:

> "*[...] a step in the KDD process consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce enumeration of patterns of the data*"

Dunham [37] also shared this idea while defining data mining algorithms task as determining "*a model that is closest to the data*". Another definition given by Hand et al. [75] and Zaki and Meira [166] emphasised the *often large* or *massive* characteristic of data sets to be analysed by a data mining approach in that it creates specific problems and challenges in comparison with small and human manageable data. However, they did not share the clear boundaries of data mining in knowledge process as defined by Fayyad et al. [40] because"*for example, to many people data transformation is an intrinsic part of data mining*". For example, Aggarwal [2] defined data mining as "*[...] the study of collecting, cleaning, processing, analysing, and gaining useful insights from data.*". From our point of view, a more balanced and complete definition of Knowledge Discovery and Data Mining are the ones given by Mörchen [105]:

> "***Data Mining*** *is the process of finding hidden information and/or structure in a data collection. This includes extraction, selection, preprocessing, and transformation of features describing different aspects of the data*"

> "***Knowledge Discovery*** *is Data Mining with the goal of finding knowledge, i.e., novel, useful, interesting, understandable and automatically interpretable patterns.*"

The underlying about these definitions is to stress the centrality of the *human in the loop* characteristic of Knowledge Discovery. Indeed, KD assumes the output to be patterns or structures that are novel, useful, interesting and understandable. All these characteristic involve somewhat prior human knowledge, intuition and subjectivity. Hence, data mining appears to be the tool and knowledge discovery the aim.

Knowledge discovery and data mining are intrinsically related to the database community. Hence, this research area was often referred to as Knowledge Discovery in Databases (e.g. [40, 41, 51]). However, while made from a database community perspective (expect [105]), the mentioned definitions may apply to a more general context that is regardless of storage technique. For example, with the Internet of Things or monitoring applications, data amounts to be processed can be large but, in some cases, considered as *volatile*: the information can be processed

but not necessarily kept in (de) centralised storage facilities. Keeping that in mind we propose the following definitions for the rest of this thesis:

> *"**Data Mining** is the process of finding real-world hidden information or structure from data-driven observations."*

> *"**Knowledge Discovery** is Data Mining with the goal of finding knowledge, i.e., novel, useful, interesting, understandable and automatically interpretable insights."*

This research area gained interest as information systems, and storage capabilities made it possible to obtain large amounts of data that can not be efficiently ingested by human analysts. Indeed, data overload (from a database perspective [40]) or complexity of data structures (e.g. graphs, spatio-temporal data) motivate the need to design efficient approaches that may assist human users to extract knowledge, or useful insights, from such amounts of data. Without claiming exhaustivity, we provide hereafter some examples of application domains.

Experimental Sciences. An example of data mining application in scientific research is that of bio-informatics where data mining is involved in various experimental studies [87]. One of the most famous application can be that of genetics studies [153]. Data mining is used extensively is astronomy that is known for its large data sets [21] [144]. Another example is that of medical research where, for example, the aim is to discover new association rules between symptoms, diseases and treatments [164] [147]. Data mining has also a growing interest in social sciences [64] [70], history [146] economics [38] and more generally in humanities.

Healthcare and hospital management. As highlighted by Koh and Tan [86], data mining can be used in healthcare management to design more appropriate and efficient interventions and medical protocols and reduce numbers of hospital admissions.

Telecommunication. The huge amounts of data generated by telecommunications technologies (e.g. network events, phone calls logs) were also of interest for data mining approaches in the aim of detecting frauds or identifying/predicting network faults [157]. Telecommunication data has also been motivational for some notable data mining contributions as the well-known episode mining of Mannila and Toivonen [101].

Retail and marketing. One famous example of data mining usage in retail and marketing is the well-know market basket problem introduced by Agrawal and Srikant [4]. Data mining techniques are also used for recommender systems [130] (e.g. e-commerce).

Banking and Finance. One common application of data mining techniques in the banking sector is that of fraud detection [126] that is devised to detect fraudulent behaviours in financial transactions.

Manufacturing and industry. The increasing use of sensors technologies in manufacturing and the rise of the so-called "Industry 4.0", large amounts of data can be generated by industrial processes. In this context, data mining techniques can be used for tasks as predictive maintenance or machines health predictions [92].

Internet of Things (IoT). The usage of connected devices grows at a fast pace [114] making it possible to generate huge amounts of data that can be processed by data mining techniques to extract hidden information, making predictions or identifying recent trends [103]. Such context is relevant for a wide variety of context. One example is "smart cities" [115] with application in traffic prediction or grid management. Another relevant context is "smart homes" where data mining was applied, for example, to discover human activities in residences for health care applications [129] [163].

To conclude this section, we emphasise that methods and techniques included in the Data Mining and Knowledge Discovery scope are relevant in any context where the need to generate value from large amounts of data is necessary. In a world where sensing technologies are democratised, these approaches can be used to extract novel knowledge and descriptions, permitting a better understanding of characteristics and phenomena that rules these environments. In this thesis, we are interested in temporal data that, thanks to the temporal dimensions, permits to describe attributes evolution in time.

1.3 The high-level interval-based state representation approach

Temporal data describe the state of attributes of a given subject or environment and situate this information in the time dimension. As we will discuss in Chapter 2, data describing temporal information are modelled w.r.t several characteristics of the data generation process that defines their semantics and interpretation.

Time domains. Generally speaking, data generation processes use a discrete representation of time: no realisable data source can claim infinite temporal precision. Discrete time domains are principally characterised by the adoption of a *time granularity* defining the level of temporal precision. The choice of *temporal granularities* depends on several parameters among which semantics of temporal facts or hardware capabilities.

Temporal primitives. Temporal facts can use a duration concept. *Time point data* permits to provide a snapshot of an attribute at a given point in time (e.g. the train arrived at 8:00 p.m.). On the other hand, *time intervals* express states lasting in time (e.g. temperature was above 20 degrees between 10:00 a.m. and 1:00 p.m.).

Data models. Following temporal information semantics, data can use a wide variety of data models. For example, numerical values are used for time series (e.g. temperature sensor), symbolic values for telecommunication alarms (e.g. connection events), item sets for market baskets, pixel matrices for videos and images (e.g. outdoor cameras).

Datasets temporal formats. We distinguish in this work between two main categories of temporal datasets: *subject-centred* datasets associate each temporal fact to a subject (e.g client 1 purchased item *a* in February 12^{th}) and *attribute-centred* datasets that discribes temporal facts without assigning them to any subject (e.g item *a* was bought in February 12^{th}).

In a heterogeneous data source system, several data models with various characteristics may co-exist if different attributes are generated by different types of data sources or express semantically different concepts. In order to be able to extract temporal knowledge from data gathered from heterogeneous data sources sensing a particular environment, the first question one may pose is:

Problem 2

How to take into account information given by heterogeneous data sources in the same knowledge discovery process?

We think that two main directions can be investigated to solve this problem. The first is to use or design efficient data mining techniques permitting to handle different data models. As we will discuss in Chapter 4, the majority

of temporal pattern mining algorithms assume homogeneity in data models and their temporal aspects. The approach we adopt in this thesis is that of representing information given by heterogeneous sources with a single data model. This approach is called Temporal Abstraction and was defined by Moskovitch and Shahar [108] as:

> *"Temporal abstraction (TA) is the segmentation of a series of raw time-stamped, multivariate data into a symbolic time interval series representation, often at a higher level of abstraction [...] suitable for human inspection or for data mining."*

We use TA to represent information as a set of symbolic high-level interval-based data streams. We will discuss briefly in Chapter 3 how the information provided by heterogeneous data sources systems may be represented as a set of state streams. A *state* corresponds to a data configuration of interest for an application domain. It is defined as a predicate on data produced by a data source system: it can use data from one or multiple data sources. A state stream is an interval-based sequence composed of non-overlapping intervals $[t_i, t_{i+1})$ where a state is said to be valid (i.e its corresponding predicate is verified during $[t_i, t_{i+1})$. The main difficulty of the TA approach is the definition of states. This task can be accomplished using common sense knowledge (e.g a video-monitored area is in state *In Motion* between events *motion begin* and *motion end*), domain expert knowledge (e.g a patient is in state *Sick* if its corporal temperature is above 37°C degrees). Also, users may use data model specific discretisation approaches to automatise the definition of states predicates (e.g. time series discretisation, clustering). We believe that this approach may permit to obtain insightful temporal knowledge from mainly two reasons:

- The set of defined states provides a high-level *vocabulary* for temporal pattern languages to be used in a knowledge discovery process. Knowledge expressed with such high-level vocabulary is more easily interpretable than with *raw* labels insights.
- The set of defined states can use knowledge from multiple domains. Therefore, the state stream representation of raw information can include various interpretations or perceptions of the same set of temporal facts. This can constitute an opportunity to discover multi-domain knowledge.

In this work, we analyse different aspects of this data model unification approach and provide a general framework without investigating in detail its practical and implementation aspects. We will, indeed, focus on the discovery of temporal knowledge between interval-based state streams.

1.4 Temporal pattern languages and assessment

The problem of mining temporal knowledge from interval-based state streams do not differ from any data mining task. Mannila and Toivonen [102] provides a general problem formalization of data mining from an inductive database perspective:

Problem 3 (Pattern mining [101])
Let r be a dataset, $\mathcal{L}$ a language for expressing properties of the data and a constraint $\mathcal{C}$. The task of data mining is to find a theory $\mathcal{T}$ such that:

$$\mathcal{T}(\mathcal{L}, r, \mathcal{C}) = (\varphi \in \mathcal{L} \mid \mathcal{C}(r, \varphi) \text{ is true})$$

Data mining in this definition consists of exploring a search space composed of all sentences of a language $\mathcal{L}$ to find elements $\varphi \in \mathcal{L}$ that fulfils a given constraint in a dataset r. The data mining algorithm computes the theory $\mathcal{T}$.

In our context, r is a set of interval-based state streams $r = \{A_0, A_1, ...A_n\}$. In order to extract temporal knowledge from r, we need to define a temporal pattern language $\mathcal{L}$, a set of constraints $\mathcal{C}$ defining the interest of an element in $\mathcal{L}$ and design algorithm devised to compute the theory $\mathcal{T}$. Therefore, to define our problem, we need to respond to the following two questions.

A. What temporal knowledge do we want to extract?

In this dissertation, we aim to find "interesting" temporal relationships, or in other terms, "interesting" co-occurrences and successions of temporal facts. This task is known as temporal pattern mining that we will survey in Chapter 4. We distinguish contributions in this research field according to two main criteria that permit to guide the choice of a consistent pattern language: time primitives and qualitative/quantitative temporal information.

Two main temporal primitives are used for temporal pattern mining: temporal information are either modelled as *time points* or *time intervals*. *Time points* is the time primitive that received the largest research effort. Three temporal qualitative temporal relationships exists between time point data: *before*, *after*, *co-occurrence*. These temporal relations define relatively simple pattern language that consists mostly of event sequence description (e.g. A after B after C ...). The co-occurrence relationship can be expressed using an event set sequence (e.g. A after $\{B, C\}$ after D). As argued by Allen [11], *time intervals* allow to express more complex relationships. The temporal algebra that was formalised by the former authors is composed of 13 pairwise qualitative relationships that make the definition of non-ambiguous and clear pattern languages more difficult and harder to compute. Indeed, as reported in [82], the majority of interval-based pattern languages suffers from expressivity problems (e.g. the same pattern may describe different temporal configurations).

Temporal patterns can also use quantitative information about time lags between successive temporal facts. The very few contributions that include temporal information into pattern languages represents time lags as ranges (e.g. A occurs 1 to 3 time units after B) or as typical time lags (A occurs 2 time units after B). The quantitative information can provide very insightful information as can be demonstrated by the following example where specifics actions can be triggered according to the time lag characterisation of quantitatively different and qualitatively equivalent behaviours.

> **Example 1 (Energy efficient advertising billboards)**
> *Say that an advertising agency possesses a video billboard on a road where pedestrians, bicycles and cars move along. To reduce the energy consumption of the billboard, the agency equips the road with four motion sensors (two at each side of the road) and uses qualitative pattern detection to turn on or off sides of the billboard: if an object moves toward the billboard, the later is turned on. The agency stores information given by motion sensors and runs a sequential pattern mining algorithm on the gathered data. In this case, the extraction of qualitative patterns seems to be useless. Qualitative successions of events are already known (and used) by the domain expert. On the other hand, quantitative patterns permit to highlight and characterise actor types based on time lags: short time lags for cars, medium for bicycles and large for pedestrians. This quantitative characterisation of time lags is a novel insight that can be used by the advertising agency to personalise ads displayed in the billboard screens w.r.t actors types based on their speed (e.g. display discounts on wheels for cars, helmets ads for bicycles and nearby restaurants menus for pedestrians).*

Quantitative information with interval-based pattern languages also permits to overcome some expressivity problems posed by purely qualitative approaches. To the best of our knowledge, only two existent approaches can be applied to interval-based patterns from interval streams [78, 127]. These two works represent the temporal information between successive intervals as a pair (α, β). A relationship between intervals A and B with a (α, β)

time lag denotes that the first endpoint of B often follows the first endpoint of A after α time units and the second endpoint of B often follows the second endpoint of A after β time units. We use this temporal information model in this work. However, Hassani et al. [78] and Plantevit et al. [127] adopted different *interestingness* measures and different constraints. This leads to the second main question.

B. What is an interesting temporal relationship?

Depending on the type of data sets (i.e. sensory-like single sequences or database of short sequences), different constraints and assessment approaches can be defined for temporal patterns. Numerous types of interestingness criteria can be considered in a pattern mining approach [60]. In this section, we will focus on two main aspects permitting to initially define the general problem of this thesis: an interestingness measure and an assessment approach.

To the best of our knowledge, the majority of existing temporal pattern mining contributions considered occurrence counting as interestingness measures. Without going into details, a pattern is more interesting if more of its occurrences appear in a data set. Depending on the structure of data set (e.g database of sequences, streams), it can be calculated as an absolute number (i.e pattern p appears n times), as an absolute frequency (e.g pattern p appears in 30% of the transactions) or as relative frequency, or confidence (e.g item B appear after 20% of occurrences of A). Compared with other assessment approaches, using confidence permits to have more predictive power as it define interest w.r.t the specific items. For instance, a pattern A *before* B may have an absolute frequency of 50% but a confidence of 100%: A *before* B appear in 50% of the analysed sample and A is always followed by B. Occurrence counting interestingness measures were largely used for *time point* and *time interval* data (e.g [78]). The inherent duration concept of intervals allows to consider a different measure based on duration of co-occurrence or intervals intersection as initially proposed by Plantevit et al. [127]. They proposed to model a pairwise temporal relationship between states A and B as a dependency $A \rightarrow B$ denoting that valid state A often occur with state valid B in terms of duration. This dependency is assessed via an intersection-based confidence value:

$$\mathbf{conf}(A \rightarrow B) = \frac{\mathbf{len}(A \cap B)}{\mathbf{len}(A)}$$

where $\mathbf{len}(A \cap B)$ is the cumulative intervals' duration of the temporal intersection between intervals of A and B. In order obtain time lagged dependencies, the conclusion streams, here B, can be shifted with an (α, β)-transformation. The resulting stream, noted $B^{(\alpha,\beta)}$ is composed by intervals such that $\forall [t_i, t_{i+1}) \in B, \exists [t_j, t_{j+1}) \in B^{(\alpha,\beta)} \mid t_j = t_i - \alpha$ and $t_{j+1} = t_{i+1} - \beta$. The confidence of the time lagged dependency $A \rightarrow B^{(\alpha,\beta)}$ is similarly obtained via:

$$\mathbf{conf}(A \rightarrow B^{(\alpha,\beta)}) = \frac{\mathbf{len}(A \cap B^{(\alpha,\beta)})}{\mathbf{len}(A)}$$

The intersection-based interestingness measure allows in some cases, particularly with dense sensory-like data, to provide a better assessment of temporal relationships. We will provide several arguments in favour of this claim in Chapter 3.

The second main criteria related to interestingness we discuss in this section is that of assessment. Generally speaking, existing temporal pattern mining algorithms use user-defined thresholds for interestingness measures. For instance, with occurrence counting measures, existing algorithms use minimum support thresholds: a pattern is interesting if its support/frequency/confidence exceeds the minimum support. While this approach offers the practical advantage of permitting to users to define levels of interestingness according to their *à priori* knowledge, thresholds can be challenging to define mainly when no prior knowledge is available before the mining process

or when discovery algorithm must be adapted automatically to changing contexts (e.g. fluctuating data density).

In [127], authors proposed to assess the intersection-based confidence measure w.r.t to a Pearson χ^2 test of independence. The null hypothesis of this test states that the validity duration of A and B are statistically independent. This test permits to automatically set interestingness thresholds for the mining process that is computed without additional cost for each couple of interval-based streams.

The third element of a knowledge discovery process is that of the mining algorithm. It is devised to enumerate all patterns satisfying the problem constraints in a dataset. In [127], pairwise intersection based dependencies are computed by the TEDDY algorithm devised to explore the quadratic search space of (α, β) time lags where α and β values are included in a user-give temporal range $\Delta = [min, max]$. It uses a level-wise breadth-first strategy that starts from the most general time lag (min, max) down to singletons of the form (α, α). To accelerate the mining process, it uses pruning criteria to avoid computing and assessing unpromising time lag candidates and refine them while controlling the loss in the before mentioned confidence value.

The objective of this thesis is to extend the pairwise intersection-based dependency model proposed by Plantevit et al. [127] in the aim of expressing multi-state dependencies. More precisely, we aim to build an expressive pattern language based on intersection-based dependencies, assessed automatically via a statistical test of independence, that is general enough to express complex temporal phenomena. The second objective of this work is to design efficient algorithms permitting to extract such patterns from a set of interval-based state streams.

1.5 Contributions

This dissertation tackles the problem of discovering quantitative temporal knowledge from a set of interval-based state streams. In the following chapters, we will propose a new quantitative multi-state dependency model and design algorithms for its efficient discovery. We can summarise the contributions presented in this thesis with the following three main points.

The Complex Temporal Dependency model (CTD) . We propose a pattern model and language permitting to express complex temporal relationships in a normal conjunctive form-like manner. In addition to the inference $(\rightarrow)$ that defines the confidence relationship parameters, CTD includes conjunctive $(\wedge)$ and disjunctive $(\vee)$ operators allowing to express respectively conditional dependencies and reduce pattern redundancy. Complex temporal dependencies can be used to build conditional temporal state graphs where each transition is temporally quantified (information about time lag is provided) and have a confidence measure that takes into account prior state validity information.

Discovering Conditional temporal relationships. We design an algorithm, named **CTD-Miner**, devised to efficiently extract the subset of statistically significant Complex Temporal Dependencies that are necessary to build a conditional temporal model. This algorithm integrates pruning criteria based on both Apriori-like assumption and a closed-like property of Complex Temporal Dependencies. These are used to reduce the computational cost of the exploration of the CTDs search space as well as reducing information redundancies in the result set.

Mining specific time lags between pairs of interval-based streams. One crucial step of CTD-Miner is that of discovering pairwise temporal dependencies. This step is the core operation in our mining process as its efficiency (computational time cost, precision of results) determines the performance of CTD-Miner. As the available algorithm, named TEDDY, is quadratic (at the worst case) w.r.t the temporal constraint Δ, we propose an alternative approach, named *Interval Time Lag Discovery*, that is linear w.r.t Δ. It uses an exploration based on the statistical assessment of minimal confidence variations heuristic, i.e. confidence gains and losses while adding or

subtracting a unit from a time lag value. The evaluation of ITLD compared to the existing approaches, intersection based TEDDY and occurrence counting-based PIVOTMiner [78], suggests that ITLD permits to efficiently more precise results. Hence, it permits to enhance performances of CTD-Miner.

1.6 Thesis outline

The reminder of this thesis is structured as follows. Chapter 2 discusses different aspects of temporal data modelling. Chapter 3 motivates the temporal abstraction approach and the choice of the interval-based model as a unique high-level symbolic representation of temporal information. Chapter 4 presents the state of the art of temporal pattern discovery and categorizes available patterns languages and mining approaches w.r.t time primitives and temporal information nature. In Chapter 5, we introduce and motivate the Complex Temporal Dependency model. We also show that CTDs can be used to build conditional models of temporal phenomena in a state stream data set. Chapter 6 presents the *CTD-Miner* algorithm and its pruning criteria. Chapter 7 proposes ITLD as a method for time lag discovery between interval-based streams. In Chapter 8, we evaluate the approach presented in this dissertation using synthetic datasets as well as real data gathered from an experiment using motion detection from outdoor cameras. Finally, Chapter 9 concludes this work and discusses future directions and open questions.

Part I

TEMPORAL DATA AND INTERVAL-BASED ABSTRACTION

2

Modeling Temporal Data

Contents

Temporal data differs from static data by its temporal dimension. In addition to providing information about a subject/attribute, temporal data situate this piece of information in time. Indeed, temporality is core information when it comes to describing a dynamic world where features and behaviours are permanently changing in time. In this work, we are interested in discovering knowledge depicting quantitatively temporal phenomena from a set of data streams. In order to have a general understanding of our problematic, we first discuss in this chapter how real-world temporal phenomena can be modelled through a brief survey on several aspects of temporal data.

2.1 Time domains

The first question one may pose when modelling temporal data concerns the time domain or the timeline representation used to express temporal data. Three main time domain types can be distinguished: *ordinal*,*continuous* and *discrete* .

The *ordinal time domain* does not properly contain elements but is defined as a relative order between temporal assertions. This time domain is used to express relative temporal relations between facts without any concerns about situating them on a quantitative timeline [7]. This model is often used for data mining tasks (e.g. sequential pattern mining) when only the total order is of interest to the application domain. For instance, a customer purchases in a market can be expressed as a sequence describing the ordering of bought products baskets:

$$\langle \{milk, diapers\}, \{tea, bread, milk\}, ..., \{coffee, diapers\} \rangle$$

In this example, we can notice that no information is given about the duration between the customer purchases nor their dates (i.e. timestamps).

The *discrete* and *continuous* time domains can be characterized by their ability to situate pieces of data in a timeline, or the temporal dimension. They are constituted by an infinite set of totally ordered elements t, called time anchors, that are to be interpreted as "landmarks" in the temporal dimension. One common analogy permitting to illustrate the significant difference between these time domains is that of real and natural number. On the one hand, the *discrete* time domain can be mapped into the set of integers: any element t of a discrete time domain have a unique direct predecessor and unique direct successor. On the other hand, the *continuous* time domain is to be related to the set of real numbers: the gap between two distinct elements contains an infinite set of elements.

In a *continuous* time domain, the basic temporal elements are absolute *points* in time that can be associated with a real number. It can be straightforwardly defined as follows.

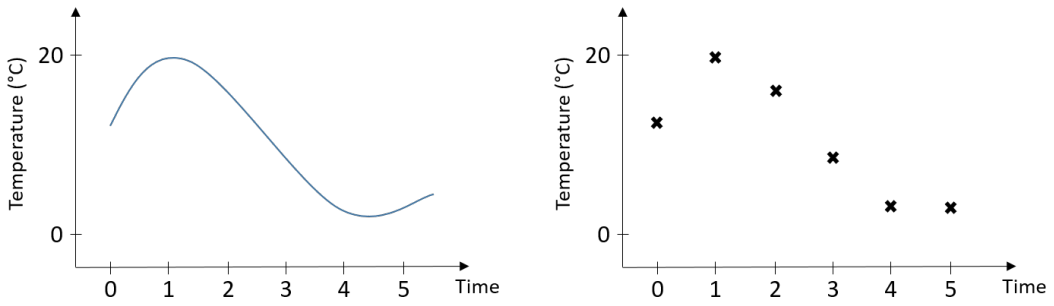
Definition 1

The continuous time domain $\mathcal{T}_{cont}$ is an infinite time line representing the time dimension defined as a infinite sequence of totally ordered time anchors:

$$\mathcal{T}_{cont} = \{t_i\}$$

where $i \in \mathbb{N}$, $t_i < t_{i+1}$ and $\forall i, \exists t, t_i < t < t_{i+1}$.

The inherent assumption made by this time domain is that of infinite precision of the temporal information [68]. This domain provides a handy abstraction tool to deal with temporal data and is the most used temporal domain in data mining and machine learning fields. Frank [50] also argues that the *continuous* time domain is *"preferred in [...] all sciences that build models similar to physics, using differential equations, etc"*. However, from a data-driven practical point of view, the straightforward usage of such the time domain may suffer from limitations



(a) Temperature evolution corresponding to a real phenomenon.

(b) Data produced by a temperature sensor as a time series.

Figure 2.1.1: evolution of temperature and its time series representation.

when it comes to reflecting real-world data generation processes (hardware specifications), real-time processes or temporality in natural language [136].

Let us consider the example of time series. Figure 2.1.1 provides an example of a real-world continuous temperature evolution over time and its time series description that is given by a sensor with a 1 Hz sample rate (one given measure per second). Let us assume the continuous time domain and that a user wants temperature information at the exact time point $t = 1.5$. The naive answer to this query is *unknown*: the sensor provided no data value for this exact "point" in time. However, the practical usage of such time series to describe the physical measure of temperature relies on the assumption of its "smooth" evolution: a measure given at t is considered to be *valid*, or a good enough approximation of the real value for any time point between t and $t+1$. Segev and Shoshani [137] had a similar intuition while defining the "continuous" time sequence type that involves, at the semantics level, an interpolation function that assigns data to undefined data points in the continuous domain. This can also be remarked from a natural language perceptive [140]. For example, the assertion *the train arrived at 9 o'clock* does not usually refer to the exact absolute point in time *9 o'clock* but to a portion of time that is small enough to provide sufficient precision about this event (e.g. 1 minute for train arrivals). What is to emphasize here is that the representation of temporal data must include abstract notions (e.g. precision, validity, interpolation) to properly reflect the temporal approximation and assumptions that are inherent to the discrete data generation process. Indeed, there is no realizable data source (humans nor devices) that can produce data fitting the continuous time domain. Such data source should be capable of producing an infinity of values for any portion of time, which requires an infinite storage capability. Snodgrass [140] exposed several practical arguments in that same direction including imprecision of time measurement (an event can at best be situated within a small duration) and *necessity [...] to have some discrete encoding for time* for any implementation. Goralwalla et al. [68] tackled the notion of *determinacy* to address the uncertainty problem of temporal information at the conceptual level. A temporal assertion is *determinate*, i.e. it is valid during an entire given duration, or *indeterminate*, i.e. it is verified during at least a portion of the given duration. Another simplification is the use of an ε uncertainty amount for each data value as stated by Frank [50] but may lead to contradictory temporal relations (total order transitivity is not guaranteed) which makes it complicated and non-practical to express temporal relations and order.

The *discrete* time domain $\mathcal{T}_{disc}$ is principally characterized by a temporal granularity corresponding to a non-null duration defining the degree of precision, or resolution level, of the temporal dimension [67]. Several definitions and interpretations were given to this amount of time. Ariav [15] named it *chronon* and provided an application-oriented definition: *"the shortest meaningful unit of time in the specific application"*. A more abstract and

philosophical definition was given by Allen and Hayes [13] for whom a *moment* indicates a non-decomposable period (i.e amount of time) during which *"nothing could change [...]"*. In the sense of Allen and Hayes [13], a natural example of a *moment* is *"the time taken for a flash of lightning [...] where the world appears to be frozen"*. The term *time granularity*, in the sense used here, can be also referred to as *bottom granularity* [50] to differentiate it from useful temporal mappings permitting to simplify temporal assertion (e.g for data summarization or visualization [7]). In this work, we will refer to a temporal granularity as defined in the following.

Definition 2 (Temporal Granularity (TG))
A temporal granularity, TG, is a positive non-zero value in $\mathbb{R}^{*+}$ corresponding to a fixed amount of time. It refers to the minimum validity duration that a temporal fact can have within a temporal description $\mathcal{D}$.

Time granularities define the elements of a discrete time domain: indivisible time intervals. They are used to structure the temporal dimension, or the timeline, into chunks of equal length. Follows the definition of a time unit.

Definition 3 (Time unit (TU))
Let TG be the temporal granularity. The time unit TU defined by TG is the smallest time interval, of duration TG, that can be used to situate data in time.

For Goralwalla et al. [68], a time granularity determinates a mapping G between an element of the discrete time domain $\mathcal{T}_{disc}$ and its interval counterpart in the continuous time domain $\mathcal{T}_{cont}$. This mapping is defined as follows:

$$G : t_{disc} \rightarrow [t_{cont}, t_{cont} + TG)$$

where $t_{disc} \in \mathcal{T}_{disc}$, $t_{cont} \in \mathcal{T}_{cont}$ and t_{cont} and t_{disc} are considered to be equal in $\mathbb{R}$. Thus, every temporal fact expressed with a discrete time domain corresponds to a time interval in the continuous time domain. Notice that intervals used in the before mentioned mapping are closed at their left endpoint and open at their right endpoint to avoid paradoxical information. For example, say that an attribute A have different values in intervals $I_1 = [a, b]$ (value v_1) and $I_2 = [b, c]$ (value v_2). This configuration includes a paradoxical information: at the absolute time point b the attribute A has two values v_1 and v_2. In the rest of this document, we will assume that temporal assertions are always temporally *determinate* in the sense of [68]. Hence, a piece of data is considered to be valid at least during a duration corresponding to TG. Follows the formal definition of a discrete time domain we use in this thesis.

Definition 4 (Discrete time domain)
A discrete time domain $\mathcal{T}_{disc}$ of temporal granularity TG is a semi-infinite set of totally ordered *time units*:

$$\mathcal{T}_{disc} =< t_i >$$

where $i \in \mathbb{N}$, $t_i < t_{i+1}$ and $\nexists j, t_i < t_j < t_{i+1}$. Each time unit t_i in $\mathcal{T}_{disc}$ is defined as:

$$t_i = [t_i^c, t_i^c + TG)$$

where $t_i^c \in \mathbb{R}$ such that $t_i^c = i * TG$. $t_0 = [0, 0 + TG)$ is the origin of T_{disc}.

This definition, based on mapping proposed in [68], support only a linear complete order between time units composing the discrete time domain. Other temporal ordering can also be used for other purposes than data

representation as branching time (e.g. for model checking) or circular view [42] but are out of the scope of this work.

A piece of temporal information is situated in the discrete time domain by a discrete temporal anchor.

Definition 5 (Discrete Temporal Anchor)
Let $\mathcal{T}_{disc} =< t_i >$ a discrete time domain of temporal granularity TG. A temporal anchor t in $\mathcal{T}_{disc}$ is a sequence of contiguous time units in $\mathcal{T}_{disc}$:

$$t =< t_j, t_{j+1}, t_{j+2} ... >$$

The size of a temporal anchor, noted $|t|$, is the number of its time units.

The use of discrete temporal anchors permits to situate pieces of data, or temporal data values, in the discrete time domain. Segev and Shoshani [137] gave a general definition of temporal data values.

Definition 6 (Temporal Data value [137])
Let $\mathcal{T}_{disc}$ a discrete time domain. A data value over $\mathcal{T}_{disc}$ is a triplet (a, v, t) where a is an attribute, v is a value and t a temporal anchor.

An example of data values can be extracted from the example depicted in Figure 2.1.1b: $(Temperature, 20, < 1 >)$ that can be interpreted as Temperature has a value of 20 at time anchor 1. It is to notice that this definition does not include a specified data model for values to include all possible data structures. Other works, especially in the database community, considers that a data value, or a temporal database, can be bi-temporal if it supports *valid time* (time anchor when a fact is valid) and *transaction time* (time anchor when a fact is recorded) [140]. This dimension (transaction time) will not be treated in this work, and we focus on valid times.

2.2 Temporal primitives

Any representation of temporal data assumes an underlying temporal semantic that guides the choice of temporal primitives. Indeed, information that is perceived as instantaneous at a specific time granularity or lasting in time supports different primitives: respectively, time points and intervals. The usage of a temporal primitive defines the set of possible temporal relations that can stand between data values. As a consequence, any temporal data mining problem is, partially, defined w.r.t to temporal primitives used in its input. Hereafter, we introduce the most commonly used temporal primitives: time points and intervals. For each, we will discuss its temporal semantics, provide a definition and list the possible pairwise temporal relations.

2.2.1 Point-based temporal data

In a discrete time model, point-based data refer to situations or facts that are perceived as instantaneous, i.e. with a "zero duration". The following are some examples.

1. The train arrived at the station at 10:32 am
2. Temperature in the hall was 20 degrees at 12:31:20 pm
3. Last Monday's weather was "sunny."

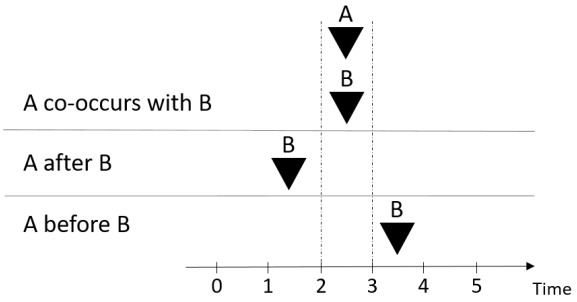


Figure 2.2.1: Relations between two point-based data values A and B.

These examples show that the notion of "instantaneity" depends on the semantics of an application domain or data source's hardware capabilities. It is, indeed, strongly related to used temporal granularity. Point-based data can be thought to as "snapshots" of an attribute at a certain degree of precision. It does not involve any duration concept but describes the state of a data source or an attribute at a *temporal anchor* of size 1, i.e. composed by a single time unit.

Definition 7 (Point-based data value)
Let $\mathcal{T}$ a discrete time domain. A point-based data value over $\mathcal{T}$ is defined as follows:

$$d_{point} = (a, v, t)$$

where a is an attribute, v a value and t a temporal anchor such that $|t| = 1$.

Following the discussion of Section 2.1, a point-based data value is said to be valid during its temporal anchor.

Point-based data support 3 basic binary temporal operators that are defined w.r.t temporal total order: *before*, *after* and *co-occurrence* . Let $A = (a_A, v_A, t_A)$ and $B = (a_B, v_B, t_B)$ two point based data values. The possible qualitative temporal operators between A and B are:

- A occurs *before* B if $t_A < t_B$
- A occurs *after* B, if $t_A > t_B$
- A *co-occurs* with B if $t_A = t_B$

In some contributions, these basic operators can use quantitative informations, A is before n time units before B, or numerical thresholds (cf Chapter 4). An example can be, A is at most 3 time units *before* B. This type of temporal relations can be encountered in temporal data mining approaches using temporal constraints as maximal or minimal gaps. Another form of quantitative threshold informs that a relation stands within the duration of a time window [101]. The basic temporal operators can also be associated with qualitative time lag semantics as *much before* or *closely after* as stated by Dubois and Prade [36] while proposing a fuzzy temporal reasoning. Notice that the latter operators provide additional information about time lags between data values.

Many extensively used temporal data models consider the point-based primitive among which time series and event sequences. Time series can be considered as a sequence of anchored numerical values, often provided periodically with a period p (e.g. a temperature sensor). Usually, this period p is considered to be the temporal granularity of the time series. An event (numerical or symbolic) sequence is a non-periodic set of point-based

data values. This temporal primitive is also the most commonly used in many temporal data mining algorithms (cf Chapter 4).

2.2.2 Interval-based data

Interval-based data notify on temporal information that involves a duration concept explicitly. They are often used to express the *state* of an attribute, an item or an environment. The following are some examples.

1. A door was open between 12:30:10 and 12:30:20
2. Temperature was high for 10 hours starting from 8:00 am
3. The number of employees in the office building was 100 between 8 am and 11 pm.

An interval-based data value associates an attribute value to a time range. Even in a discrete time domain, this representation includes somewhat a continuous aspect suggested by its semantics. For example, in the second example mentioned higher, the statement *high temperature* is considered as valid at any absolute time point ranging between time anchor 8:00 am to 8:00 am +10 hours. Therefore, an interval of time can be considered as a continuous time range whose *endpoints* are defined in a discrete time domain. Following the general definition 2.1 of a temporal data value, one can define, straightforwardly, an interval-based data value as a temporal data value whose temporal anchor have a size greater or equal to 1 time unit corresponding to the property duration. However, this temporal representation with simple temporal anchors (i.e. a set of contiguous time units), is not practical in terms of interpretation and storage and does not permit to express explicitly the continuous aspect of interval-based data. Indeed, interval-based data anchors are often represented as a pair of point-based endpoints. Follows the property of equivalence between a temporal anchor (set of time units) and an anchored interval.

Property 1 (Anchored Interval)
Let $t =< t_1, t_2, \ldots, t_n >$ be a temporal anchor over a discrete time domain $\mathcal{T}$ where $\forall t_i \in t, t_i = [t_i^b, t_i^e)$. An anchored interval t_{int} equivalent to t in $\mathcal{T}$ is the union of time units of t:

$$t \Leftrightarrow t_{int} = [t_0^b, t_n^e)$$

The duration of t_{int}, noted $|t_{int}|$, equals to $|t| * TG$

It is to notice that intervals anchors are right-closed to avoid information paradoxes. Indeed, we consider that an absolute point in time cannot support two different values for the same attribute. This representation is often practical as it permits to ensure a validity continuity and is sufficient to the present work. However, this representation is not a consensus. While defining its interval logic, Allen and Hayes [13] argue that endpoints do not belong to intervals which comes, somewhat, to consider open intervals. An interval in Allen' logic is not a set of points defined over a continuous time domain. Time points, in their perspective, are not temporal entities where a fact can or cannot be true but defines transition times between intervals. However, an interval can include a time point. In their axiomatization of temporal logic, they use the MEET relation over time moments: two intervals a and b meeting at a time point t infer that there is no possible interval meeting both a and b. In this work, we refer to interval-based data values as defined hereafter.

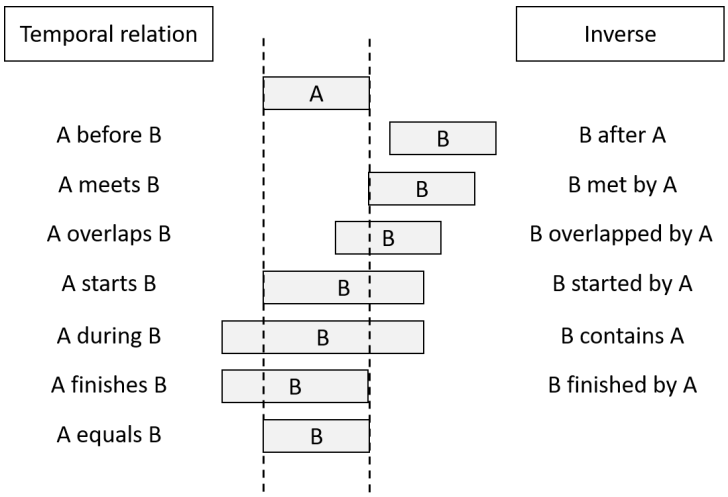


Figure 2.2.2: Graphical representation of the 13 intervals binary relations defined by Allen [10]. Each row depicts a relation and its inverse. "Equals" is its own inverse.

Definition 8 (Interval-based data value)

Let $\mathcal{T}$ be a discrete time domain of time granularity TG. A interval-based data value over $\mathcal{T}$ is defined as follows:

$$d_{interval} = (a, v, t)$$

where a is an attribute, v a value and t a interval anchor such that $|t| = n * TG$ with $n \in \mathbb{N}^*$. t is noted $[t_b, t_e)$ for readability. The duration of $d_{interval}$ is $|t|$. v is said to be valid during interval $[t_b, t_e)$.

Time intervals permits to express more complex and richer temporal relations than point-based temporal data. For example, Mörchen [105] argues that interval-based temporal reasoning, in contrast with point-based data, can express the *coincidence* concept i.e when two facts are partially valid together. As a consequence, interval-based data can support more complex binary temporal relations. The most known interval-based temporal logic was proposed by Allen and Hayes [13] [9] [10] [11] [12] who defined the 13 possible qualitative binary relations: *before, meets, overlaps, during, finishes, after, met by, overlapped by, started by, contains, finished by* and *equals*. These relations are depicted in Figure 2.2.2. Authors also proposed transitivity tables permitting to infer Allen' relations between intervals from previously known relations. For instance, if an interval i is *before* j and k is *during* j, then i is before k.

Allen' algebra was extended by Freksa [52] for semi-intervals reasoning. A semi-interval is defined as an interval endpoint. Other extensions were also proposed to include fuzzy reasoning, e.g. [18, 36], and the relation between uncertain intervals, i.e. where endpoints are modelled as ranges, e.g. [54, 61, 111, 116, 133, 135]. The latter type of temporal reasoning is very relevant in contexts such as historical data. We will not develop more on this aspect as it is out of the scope of this work.

2.3 Data models

In addition to time domains and temporal primitives, temporal data can use a wide variety of data models. Without claiming exhaustivity, this section aims to provide an overview of the diversity of temporal data models that can be encountered in the literature and industrial applications. Figure 2.3.1 illustrate one possible categorization

of temporal data and provides several examples. One first common categorization of data models is structural. Generally, we can distinguish 3 data models types: *Structured*, *Semi-structured* and *Unstructured* data [55].

As stated by Gandomi and Haider [55] *Structured data* usually refers to *tabular data found in spreadsheets or relational databases*. More generally, one can consider that data is structured if they are defined over a known and unambiguous data model. This model can consist of a database schema for object/relational databases, nodes and edges attributes for graph databases or simply data type (e.g. float, integer) of time series values. In other words, the common characteristic of structured data is that records, or observations, can be straightforwardly interpreted, processed and queried without any additional computation or ontology. Structured data can use several data models as simple numeric or symbolic values, symbolic item sets or more complex structures such as graphs and spatial data. Any of these data models can support a temporal dimension to build sequences.

Unstructured data are all data that do not follow any structural model. Data types that stand in this category are often referred to as *raw* data that needs specific analysis techniques to extract information. One example is that of plain text with the extensively studied but yet challenging fields of Natural Language Processing (NLP) and text mining [3]. Another example of unstructured data is image and video whose analysis extensively gained interest with the promising and spectacular results given by deep learning field techniques [91]. Notice that video data is temporal by nature and is not to be confused with image sequences (i.e. images can have multiple sources).

Semi-structured data are all data that are neither unstructured nor fully-structured [1]. Buneman [22] stated that in semi-structured data, information that is usually associated with schemes or data models in structured data is contained within the data and can be qualified sometimes as "self-describing". Unstructured data are encountered mainly in the web and have received many research interests in, for example, the semantic web community. Typical example of semi-structured data are XML files (e.g RDF) or JSON objects.

2.4 Subject-centred vs. Attribute-centred data sets

In addition to the characterization of temporal data nuggets w.r.t time domains, primitives and data models, one key aspect to consider is the nature of temporal data sets. In this work, we distinguish between two main classes of temporal data sets based on "activity separations": *subject-centred* (transaction-like) and *attribute-centred* (sensory-like) data sets. In addition to defining different knowledge discovery problems (cf Chapter 4), insights that can be discovered from these data sets types have subtle semantics and interpretation differences that are worth noticing. To illustrate the main differences between the proposed classes, let us consider the following example.

> **Example 2**
> *Let us consider the set of temporal facts depicted in Figure 2.4.1 describing timestamped events/actions associated with an identified subject. This set of facts can describe the behaviour of human actors a, b, c and d through a set of actions A, B, C and D in a building. An action can describe behaviours such as opening a door, passing through a corridor or turning on the light. The building manager is interested in discovering frequent sequences of actions describing typical behaviours.*

Let us consider two data generation scenarios for the former example:

- **Case 1.** The building manager collects data manually. For every single actor, the building manager identifies the performed actions. He collects information about time, gives a unique ID to each individual and labels each of its actions.

Figure 2.3.1: Several data types categories and examples

Time	SubjectID	Event
0	a	A
0	b	A
1	c	C
2	a	A
2	b	B
2	c	D
2	d	A
3	a	B
3	b	C

Time	SubjectID	Event
3	c	B
4	b	D
4	d	D
6	d	B
8	c	A
9	d	C
11	a	D
11	c	B
12	c	C

Time	SubjectID	Event
13	c	A
14	c	D
14	d	B
15	a	C
16	d	A
17	d	A
18	a	D
19	d	D
20	a	A

Figure 2.4.1: A set of temporal facts. At time stamp 0, the fact A is associated with object a

SubjectID	Sequence
a	(A, 0) (A, 2) (B, 3) (D, 11) (C, 15) (D, 18) (A, 20)
b	(A, 0) (B, 2) (C, 3) (D, 4)
c	(C, 1) (D, 2) (B, 3) (A, 8) (B, 11) (C, 12) (A, 13) (D, 14)
d	(A, 2) (4, D) (6, B) (9, C) (B, 14) (A, 16) (A, 17) (D, 19)

Figure 2.4.2: The sequence database corresponding to temporal fact set in Figure 2.4.1 and Case 1 data generation process.

- **Case 2.** The builder manager uses sensors to keep track of activity in the building. For example, he equips door with opening/closing sensors. The sensors monitor actions and time data but do not associate them with a particular individual.

In the first case, the gathered data is commonly formatted as a sequence database as depicted in Figure 2.4.2 where a sequence of items (here actions) is associated to a unique subject ID (see Figure 2.4.4a for a graphical representation). Notice that this database contains all information contained in the set of facts in Figure 2.4.1. The ID attribute permits to make a clear separation between individuals behaviours. Rashidi and Cook [128] called this dataset configuration *transaction data*. We call it in this work *subject-centred* to stress the ability to describe the set of facts associated with particular subjects. The well-known market basket data belongs to this category [4]. This type of configuration is also used for medical records [108] or GPS tracking data. Applying a pattern mining approach, for the above example, can answer the following question:

*What sequences of activities do **individuals** frequently perform in the building?*

In the second case, sensory data is an unbounded flow of events/activities without a clear separation nor link between individuals actions: the individuals' ID is unknown for the generation process. This type of data is often represented as a single sequence of events (e.g telecommunication data in [101]) or multiple attributes sequences (e.g sensor data in [127] [128]). Figure 2.4.3 illustrates these representations. Rather than monitoring individuals by describing their activities, this type of data monitors the activities themselves. In other words, the data describe the actors' activity effects on the environment through physical measures. One typical example is that of temperature sensor that monitors temperature over time without explicitly associating its variation to a particular phenomenon or an identified cause. For this reason, we call this data configuration *attribute-centred data*. One consequence of attribute-centred generation processes is that the data may not quantitatively reflect the importance, i.e. number of involved actors or occurrences, of a given fact. Let us illustrate this by the following example.

Example 3

Let us assume the fact set of Example 2. Say that event A corresponds to "passage through entrance door". In Figure 2.4.1, one can notice that two individuals, with ids 1 and 2 passes simultaneously through the entrance door at timestamp 0. With subject-centred data depicted in Figure 2.4.2, one can notice that these two occurrences are reported in the sequence database: one for each individual. Indeed, the manual generation process permits to distinguish two simultaneous events. On the other hand, the building manager uses a motion sensor on top of the door to report on passages. When the two individuals pass simultaneously through the door, a motion will be detected, and a unique event A is recorded as shown be Figure 2.4.3. Indeed, the motion sensor is capable of providing a binary information (i.e. "motion" and "not motion") without any quantification of numbers or nature of actors in motion (e.g. a dog may also enter the building).

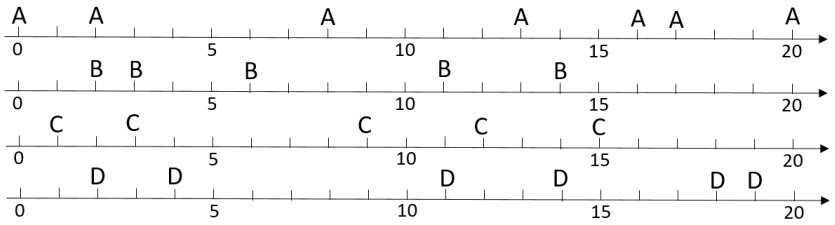
This slight but significant nuance in the generation process induces different discovery problems in comparison with subject-centred data. For our example, the problem that can be solved is the following:

*What are the significant/frequent/usual successions of **activities** occurring in the building?*

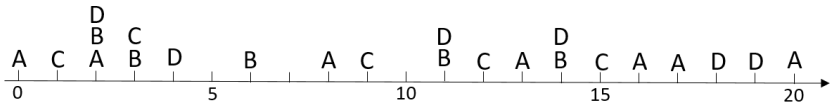
It is to notice that the former problem is different from the one posed earlier for subject-centred data. As a consequence succession of events (patterns) that can be extracted can be different and has to be interpreted according to the data configuration. To illustrate this statement, Figure 2.4.4 depicts several occurrences of pattern *ABCD* corresponding to the set of facts of Example 2 given by a frequent pattern mining approaches with a temporal gap constraint between successive events. The two red patterns correspond to successions extracted from both the sequence database (subject-centred) and single sequence (attribute-centred). What is to emphasize here is that the neat interpretation of these two patterns is different:

- Subject-centred: half individuals in the database perform activity sequence *ABCD*
- Attribute-centred: there are two occurrences of the sequence *ABCD*.

Moreover, with the single sequence representation two more occurrences of the *ABCD* pattern, depicted in green, are observed. One remark is that, for this particular example, the subject-focused result set is a subset of the attribute-centred result set if the interpretation is not taken into account. This may make sense as individuals event sequences are, to some extent, a particular partition of the general event sequence. What is to stress here is that the problem of mining attribute-centred data seems to be a more general problem than mining subject-centred data. While transaction data can be straightforwardly transformed into a single sequence framework, for instance by the concatenation of sequences in databases with specific conditions (e.g. distance between sequences), the inverse seems to be more difficult. Indeed, this supposes having information permitting to partition the single sequence such that each sub-sequence corresponds to facts related to a single subject. Techniques performing this kind of partitions, as the well-known sliding window, does not allow to maintain the semantic meaning that is underlying to subject-centred data. This aspect will be discussed in the next sections with more details.



(a) Attribute sequences (streams) representations

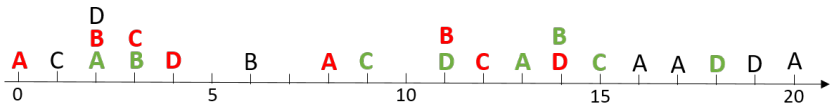


(b) Single sequence representation

Figure 2.4.3: Common representations of attribute-centered (non transaction-like) temporal data sets

ID	Sequence
a	A A B D C D A
b	A B C D
c	C D B A B C A D
d	A D B C B A A D

(a) Subject-centered sequence database



(b) Attribute-centered single sequence

Figure 2.4.4: A graphical representation of subject-centred (a) and attribute-centred (b) datasets corresponding to Example 2. Coloured activity labels belong to occurrences of sequence pattern ABCD. Green labels correspond to patterns found from a subject-centred data set. Red labels are included in patterns discovered only from the attribute-centred database

3

State-based model unification of heterogeneous data

Contents

3.1 Introduction

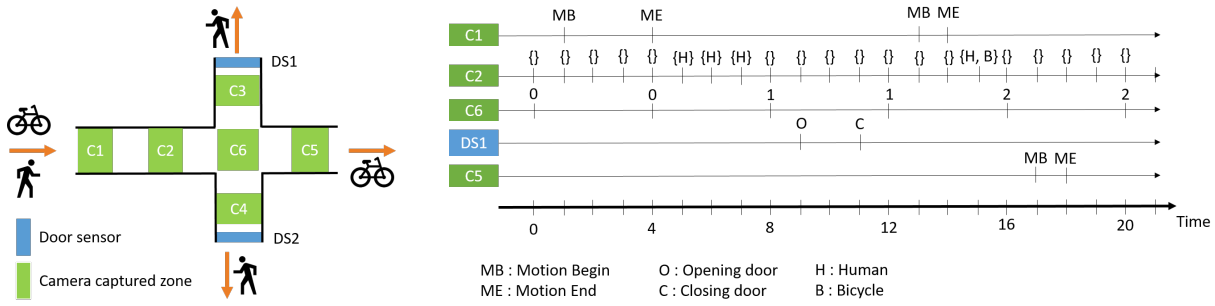
Remarkable advances in sensors, network technologies and low storage costs had lead to the massive and rapidly growing use of sensor devices in many application domains impacting both private, public and industrial areas and at different scales (human body, buildings, industrial factories, cities). This is demonstrated by the spectacular growth of sensors and "smart objects" in the last decades. Current numbers and projections of used sensor devices are estimated at dozens of billions [113] of devices. This shows the great interest arousing around sensing technologies and explain their dissemination. Maybe the more illustrative example of the former claim is the emergence of the Internet Of Things (IoT) paradigm where sensor devices, or "smart objects", are integrated into the internet global network [158]. This makes it possible for monitoring application to "sense" environments, "contextualise", "understand" and interact back. Indeed, classical sensors, more sophisticated devices (mobile phones, video cameras) as well as humans (through manual reporting) can be seen as data sources providing a perception of their environment.

Another traditional field impacted by sensors technologies development is that of Industry. Indeed, manufacturing systems (e.g. production lines, industrial engineering) were traditionally using sensors to monitor, in real-time, the state of machinery as production lines (temperature, pressure, counting pieces). However, the increasing affordability of sensors as well as storage capabilities, makes it possible to acquire vast amounts of reliable data describing manufacturing processes and enterprise operations. Besides, the use of such diverse data sources comes with a significant heterogeneity: physical measures from traditional sensors, videos from cameras, textual reports from human operators, images.

One common fact about the after-mentioned application domains is the growing heterogeneity of the generated and available data. This heterogeneity is to be seen as a great opportunity and poses various challenging research problems.

Heterogeneity is an opportunity. This claim is based upon the following observation. *The more attributes describe a given environment, the more they allow a better accurate expression of possibly complex activities/behaviours/phenomena*. Besides, advances of information retrieval techniques for unstructured data permits to obtain, with improving precision, complex information that tries to emulate the human cognitive capabilities. For instance, recent computer vision technologies in image and video processing (e.g. object classification) offers the opportunity to recognise and visually "sense" an environment with great accuracy. Another example is Natural Language Processing and associated fields that permit to retrieve information and provide context from human speech and written text. This huge availability of diverse data permits to describe, for the same environment or subject of study, a multitude of variables over time that captures the possible complexity of temporal phenomena.

In this context, Knowledge discovery (cf Chapter 4) may permit to extract patterns or structure in the data describing hidden temporal phenomena. Besides, patterns heterogeneity may provide more explainable insights. Let us illustrate this with a simple toy example. We describe in Figure 3.1.1 a set of corridors equipped with a sensor system composed of door sensors and video cameras capturing parts of the corridors. For data produced by video cameras, advances in image and video processing make it possible to obtain useful insights: motion detection, objects counts, object recognition. In this example (in Figure 3.1.1, C_1, C_3, C_4 and C_5 provides motion detection, C_2 provides object recognition and C6 counts moving objects. The analysis of motion information would permit to obtain 3 main trajectories *"C_1 then C_3"*, *"C_1 then C_4"* and *"C_1 then C_5"*. In this environment, C_3 and C_4 can only be reached by pedestrians and C_5 by cyclists. More valuable insight can then be obtained while using different attribute types to form a heterogeneous temporal relation. For example, a pattern *"C_1 then <Bicycle> then C_5"* provides more insight than a simple trajectory pattern *"C_1 then C_5"*. This information can then



(a) Corridors equipped with door sensors and video cameras

(b) Some attribute data streams gathered from the sensor system depicted in Fig.3.1.1a

Figure 3.1.1: A sensed environment example and corresponding attribute streams

be used to perform automatically specific tasks in the environment with regards to the object type information.

From a very general perspective, one can distinguish two main types of data sources providing perception: devices (e.g. sensors) and human reports. Although insights from these data types can have different characteristics and level of complexity, we will consider them as equivalent in this thesis to provide a very general point of view. We will refer as *data sources* any device or human providing data. Follows the definition of a data source and a data source system.

Definition 9 (Data Source)

A data source is an entity that can react and report on the state and evolution of a single or various attributes of an environment. Formally, a data source d is defined as $d = (id, \langle (a_i, S_i) \rangle)$ with id is its unique identifier and a_i an environment attribute monitored by d and S_i a temporally anchored observation sequence of attribute a_i produced by d.

For example, in Figure 3.1.1, $C1$ is a data source defined as $(C_1, \langle (motion, ((MB, 1), (ME, 4), (MB, 13), (MB, 14)) \rangle)$.

Definition 10 (Data source system)

Let $\mathcal{E}$ be an environment where a set $\mathcal{A}$ of activities/behaviours/phenomena occur. A data source system associated with $\mathcal{E}$, noted DS is a set of data sources $DS = \{d_i\}$ sensing the evolution of $\mathcal{E}$ under the effect of $\mathcal{A}$. A data source system provides a temporal description of $\mathcal{E}$.

A data source system can be composed of sensors of different types as well as inputs from human users (e.g. via mobile phones). It composes a perception layer providing "real-time" monitoring of activities/behaviours/phenomena occurring in a single environment. We can say that the state of the perception layer is changed by the environment temporal evolution. This perception layer provides, then, a set of data streams reporting each on a physical measure describing one of the aspects of the environment. Figure 3.1.2 provides an illustration of such configuration.

In this context, one general challenging problem posed to monitoring applications and knowledge discovery techniques is the following:

> Let $\mathcal{E}$ an environment equipped with a heterogeneous data source system DS. How to extract non-trivial, novel and directly actionable temporal knowledge of $\mathcal{E}$ from data streams given by DS?

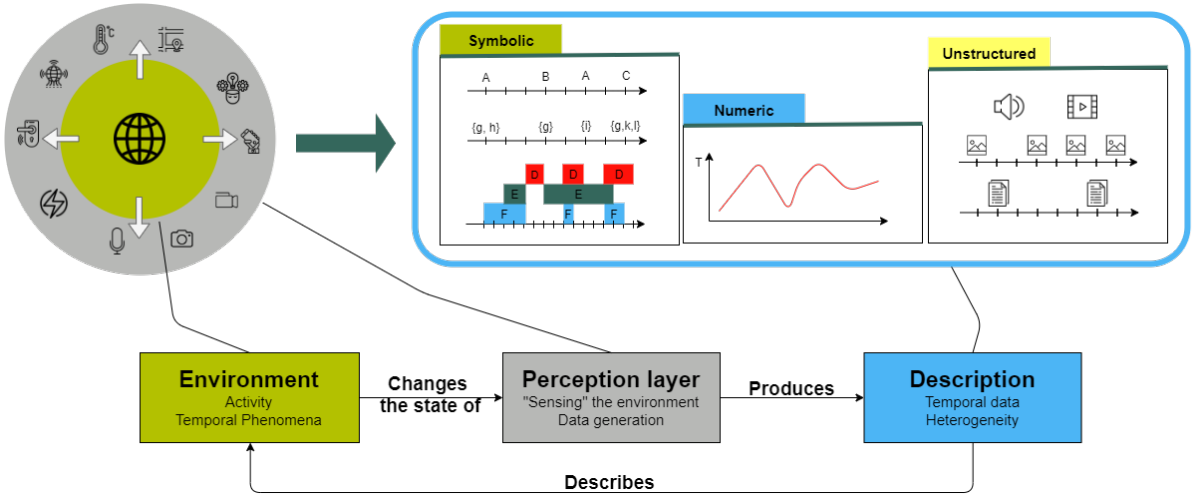


Figure 3.1.2: From environment evolution to temporal data. A simple data generation process overview

Temporal data provided by a heterogeneous data system can be highly heterogeneous at different levels. We discussed in Section 2.1 different aspect of temporal data modelling including time domains, temporal primitives, data models and data formatting. Each of these aspects may pose problems when it comes to performing temporal knowledge extraction:

- **Time domains heterogeneity**. Assuming that data generation processes always use a discrete time domain, each data source d_i in a data source system uses a time granularity TG_i corresponding to its data semantics or sources capabilities (human perception, devices specifications).
- **Temporal primitives heterogeneity**. As discussed in Section 2.2, temporal data can use mainly two temporal primitives, time points or time intervals, depending on data semantics. Indeed, temporal information can include or not a duration concept. Semi-intervals can also be useful to express unbounded information. Besides, fuzzy temporal information (e.g. an event occurred between t_1 and t_2) can also add complexity and heterogeneity to the set of temporal primitives used in a data source system.
- **Data models heterogeneity**. The qualitative diversification of data sources comes with great heterogeneity in temporal data representations. For instance, structured data (e.g. timestamped events, time series), semi-structured data (e.g. JSON files, XML files) and unstructured data (e.g. images, videos, sound) can all be generated by a data source system monitoring one single environment and providing a detailed temporal description of it.
- **Data format heterogeneity**. A data source system can also produce temporal data that are attribute-centred (e.g. sensory data) and subject-centred (e.g. market basket data).

The approach described in this chapter consists of representing the information given by a heterogeneous data source system with a unique temporal information model. This process of data transformation can be referred to as Temporal Abstraction (TA). Moskovitch and Shahar [109] defined TA as follows.

Temporal abstraction (TA) is the segmentation and/or aggregation of a series of raw, timestamped, multivariate data into a symbolic time interval series representation, often at a higher level of abstraction [...] suitable for human inspection or for data mining.

The main idea is to transform available heterogeneous data into a unique interval-based data model maintaining sufficient temporal information contained in the original raw data for comprehension and analysis. This representation relies on (1) specific information retrieval algorithms and (2) a set of predefined environment states permitting to build a state stream description of an environment and provide a high level "vocabulary" to compose temporal patterns. The remainder of this section provides (1) the set of general hypothesises we adopt in this work, (2) the motivations behind the interval-based temporal abstraction process, (3) a more detailed description of our overall data transformation process, and finally (4) a discussion.

3.2 General Hypotheses

The first hypothesis we consider in this work is the temporal synchronisation of data sources.

Hypothesis 1 (Synchronisation of time domains)
Let $\mathcal{E}$ be an environment and $DS = \{d_i\}$ a data source system with $d_i = (id, \langle (a_i, S_i) \rangle)$ a data source. Data streams or sequences S_i generated by DS are temporally synchronized with respect to an absolute time domain.

In other words, this hypothesis states that origin time points (say $t = 0$) of all data sources are aligned. From a more technical point of view, this hypothesis assumes that internal clocks of all devices generating data streams are synchronised. The problem of independent clock synchronisation is not the object of this work.

Secondly, as we discussed in Section 2.1, we consider that all data sources generate data in a discrete way. Therefore, data are always expressed in a discrete time domain. In this work, we make the hypothesis that every data source has a known and fixed temporal granularity.

Hypothesis 2
Let $\mathcal{E}$ be an environment and $DS = \{d_i\}$ a data source system with $d_i = (id, \langle (a_i, S_i) \rangle)$ a data source. Every data source d_i provides a data stream expressed in a discrete time domain of fixed granularity TG_i.

Temporal granularities can be directly retrieved from the temporal information precision that is used to express temporal anchors. For instance, if a data value has timestamps expressed in seconds, then we assume that its granularity is 1 second. We also discussed in Section 2.1, validity assumptions that are inherent to the discrete data generation process. Indeed, a data source temporal granularity refers also to the minimal duration in which a data value is considered as valid. This is our third hypothesis.

Hypothesis 3
Data provided by a data source system are determinate (in the sense of [68])

This hypothesis states that any data value is valid during, at least, a duration corresponding to the temporal granularity of its data source. The interesting problem of indeterminate data information fuzziness will not be treated in this work.

3.3 Motivations

We can distinguish several main motivations behinds a temporal abstraction approach aiming to represent temporal information given by a heterogeneous data source system in the form of a set of state streams.

3.3.1 Time domain and primitives heterogeneity

As discussed in Section 2.1, the first aspect one should consider while dealing with temporal data is what kind of time domain to use. We argued that any real data generation process is inherently discrete - i.e. must be considered with a discrete time domain - as do not exists any data source (device nor human) that can pretend to have infinite precision. On the other hand, to the best of our knowledge, temporal pattern mining approaches consider, in the analysis process, "continuous" time domains as they assume homogeneous temporal granularity. Indeed, if one considers a set of homogeneous temporal granularities, the inherent data "validity" and "interpolation functions" assumptions are homogeneous in the entire data set. As a consequence, it is possible to interpolate the discrete data into the continuous time domain without loss of qualitative nor quantitative relational information. For instance, two co-occurring events generated with the same temporal granularity are also co-occurring if one considers them as absolute time points in the continuous time domain. The same can be stated for the duration between time points. In this work, we investigate the ability to perform data analysis on heterogeneous temporal data sets that may contain data modelled over different temporal granularities. For instance, a data source system can contain a temperature sensor providing data with a granularity of a millisecond, a door sensor monitoring door opening and closing with a granularity of a second, and human-given events about the weather using a granularity of 1 day. It is to notice that all after-mentioned events are time points as they do not consider any duration concept at the semantic level. The use of such heterogeneous time granularities for the same environment description raises the following problem for a data model unification:

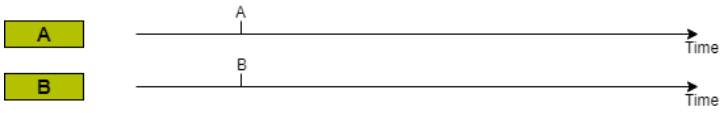
> Let Σ be a set of data streams having heterogeneous temporal granularities. How to represent temporal data of Σ in a "continuous" time domain suitable for knowledge discovery in a way that maintains possibly all temporal information given by Σ and consistent temporal relations between data?

To illustrate this problem, let us consider the toy example depicted in Figure 3.3.1a. Streams A and B have a temporal granularity of two time units and one time unit respectively. Considering the fact that both A and B contains time point events, the naive interpolation into the continuous time domain is to consider events in A and B as simple time points. For a raw event having a timestamp t in raw discrete A or B, the naive interpolation into the continuous time domain that considers absolute timestamps is depicted in Figure 3.3.1b. What is to notice here is that this naive interpolation comes with information loss at the relational level. For instance, absolute time points in A and B are co-occurring in Figure 3.3.1b. This does not reflect correctly the available information provided by raw streams A and B which can be expressed with Allen algebra with *"B starts A"*. As a consequence, the unification of time domains for data analysis purposes can be done through interpolation of discrete temporal data into a "continuous" time domain that maintains information about temporal granularities that are specific to each data generation process. This permits to reflect better temporal information given be several data sources, with several degrees of precision, while preserving relational aspects that can better describe the actual temporal ordering of data.

The use of the term "continuous" in the last paragraphs to qualify time domains is, in reality, a simplification that necessitates a precision. Indeed, the use of the continuous time domain "abstraction" through the time granularity-based mapping do not produce data that are really in the continuous time domain. The unification implicitly assumes the least common multiple of all temporal granularities as the granularity of the unified representation.

As temporal granularities are defined as the smallest duration of time where a fact is valid, the unification of time domain w.r.t a time granularity TG also allows considering time intervals as the unique time primitive

(a) Discrete raw data streams A and B. The time granularity of A equals to twice that of B.



(b) A naive and straightforward interpolation into the continuous time domain

(c) Interpolation of time points events of streams A and B considering their temporal granularity

Figure 3.3.1: Two time point streams having different granularities (described in time axis). The naive interpolation of time point data of streams A and B into the continuous time domain, in figure (b), induces a loss of information. Their inherent validity assumption is not taken into account causing a deformation of temporal relationship. Considering time granularities permits to maintain information given data validity assumptions.

for the knowledge discovery process. Indeed, time points will be considered as intervals having as duration their temporal granularity, and time intervals remain time intervals with consideration of temporal granularity for their endpoints.

3.3.2 Data format heterogeneity

We discussed in Section 2.4 the two different temporal data formats that we called subject-centred data, e.g. sequence databases, and attribute centred-data, e.g. sensory data. We argued that the main difference between these two data generation processes is the segmentation of data using activities separation. For subject-centred data, temporal data are segmented to reflect attributes evolution that is related to single objects. On the other hand, attribute-centred data describe the evolution of attributes without any information about their subject (i.e. what object in an environment is responsible for the attribute evolution). The difference between these classes of temporal data has consequences on data mining tasks as different problems can be posed and different interestingness measures can be used (e.g. semantics behind support are different for the two classes). The questioning posed by data formats heterogeneity is the following.

At which extent it is possible to benefit from the information given by both attribute-centred and subject-centred temporal data in a discovery process aiming to extract knowledge of an environment?

Several approaches can be considered to solve this problem. One direction is to design algorithms that are capable, with sufficient guarantees and corresponding interestingness measures, to process both attribute and subject centred data. In this work, we consider a different approach to data unification. That means that we aim to build a homogeneous description of a given environment that can be processed with traditional and existing mining approaches.

There are mainly two ways to unify these two data formats. The first is to convert attribute-centred data to subject-centred data. This consists of the segmentation of attribute centred into a set of data sets, each containing data generated by a single subject in the environment, which is a difficult task. Indeed, it requires additional knowledge on the monitored temporal phenomena that can be hard to obtain or extract from the data with

enough consistency guarantees. Sensory data are typical examples of the former claim. Follows an illustrative example.

> **Example 4**
> *Let us consider the data given by a traffic ultrasonic sensor set. Each device composing the system detects whether a vehicle is present on the road in a given sensing range and provides a data stream to a monitoring application. The obtained data set is composed of a set of attribute-centred data streams (one data stream per device) describing the traffic of a given road (here the road is the environment). In the post-generation processing stage, transforming this data set into subject-centred data consists of partitioning it such that each partition contains events generated by a single car. This cannot be done straightforwardly with enough consistency guarantees without additional information.*

The second approach is to extract an attribute-centred description from subject-centred data. In other words, we aim to discover typical temporal relations and correlations between environment attributes. If subject partitioning (where each piece of data is only related to other pieces of data that are related to the same subject) is considered, the knowledge discovery process will provide insight about typical individual behaviours. As discussed briefly in Section 2.4, interesting relations and patterns may arise between data generated by multiple subjects. The main idea is to extract environment description from attributes that are related to particular subjects evolving in this environment. For example, if one considers a set of door access control devices in a building providing timestamped data about access related to users identification. This information if subject-centred as each event is associated with a unique user. However, if the ID attribute of each event is not considered, the obtained data streams can be seen as an attribute-centred data set describing the set of doors in the environment (where the attribute is "door access"). Another example can be that of GPS data. The attribute-centred version of coordinates (x, y) locating a particular mobile device is: *the* (x, y) *location is occupied*. At the technical stage, this process requires simply getting rid (or the non-consideration) of the unique identification attribute.

3.3.3 Data models heterogeneity

As described in this section's introduction, data sources systems can be composed of heterogeneous data sources. Therefore, obtained temporal data sets can be composed of data expressed in a multitude of data models. We described briefly in Section 2.3 several data models that can be encountered simultaneously in data sources systems. The main remark here is that a single environment can be temporally described by both structured, semi-structured and non-structured data. Our goal is to be able to benefit from the information given by a multitude of data sources, regardless of their data models, to build and discover complex temporal knowledge.

We attempt to unify the representation of the information given by heterogeneous data models. One naive approach is to consider the most basic information representation for each attribute. for example, by using streams containing binary data. More precisely, each data source provides values in a given domain with particular bounds. For instance, a time series domain can be $\mathbb{N}$ or $\mathbb{R}$ with given bounds and precision, each pixel of an image or a video can have a value in $\mathbb{N}^3$ in the RGB space, and each tag in a stream of RDF/XML based documents can have values in limited/unlimited string space. The basic idea here is to consider a binary stream describing the validity of facts "data sources X has value y". In the following, we provide several examples of this process.

Let us consider the simple example of a symbolic event streams S. Events types in S belong to a set of symbols $I = \{a, b, c\}$. Information given by the event stream S can be represented with three attribute sequences A, B and C containing each interval where resp. facts a, b and c are valid.

Another example is that of a time series TS containing values in $\mathbb{N}$ that are bounded in the range δ of size n (e.g. for a temperature sensor δ can be $[-20, 50]$ in Celsius degrees). Information given by TS can be represented with n streams each one containing intervals of time where TS has a particular value v.

A third and more complex example is that of video data that is composed of n pixels having values in the RGB colour space (including 256^3 possible values). The straightforward application of the after-mentioned principle induces the constitution of $n * 256^3$ different values streams.

While this approach can be reasonable for symbolic data, the last extreme example shows that the attribute value streams representation is nor do-able, the size of attribute values space can be huge, nor interpretable in a straightforward manner for many data models. Indeed, what is of interest in video data is the visual information that needs interpretation and vision capabilities. Video analysis, image processing and computer vision algorithms aim to detect static or dynamic pixel configurations that are considered to be of interest or that emulate the human cognitive capabilities. This can be seen as a data segmentation or discretisation process aiming to detect high-level visual concepts that can easily be interpreted and understood. As a consequence, video processing algorithms can be used to generate high-level categorical data streams: one binary stream per possible high-level category. For example, motion detection in videos can provide a stream "Motion in Video 1" containing intervals of time where a motion is detected. The same general approach can also be applied to numerical data with time series discretisation, to raw text and audio files with, for example, speech analytics, and so on. The main idea here is to use data model specific analysis process for information extraction to build high level attributes value streams. This process has mainly two main advantages:

- It provides a more comprehensible and interpretable description of a given environment. This kind of description permits to solve the interpretation problem of complex data (mainly with unstructured and numerical data) encoding information into the streams label.
- Provides raw data with structure that permits a more simple joint use for knowledge discovery.

In addition to specific information retrieval algorithms, knowledge-based temporal abstraction, that was firstly formalised by Shahar [139], can be used to build an interval stream-based representation that reports on high-level states of a given environment. This task can be seen as the generic interpretation of raw data aiming to provide a domain-specific perception of states and trends in the data corresponding to a given goal [139]. This goal in our work is temporal knowledge discovery.

In addition to the unification of information representations, the description of an environment in terms of high-level interpretation can be interesting in two main aspects. As for automatic specific information retrieval processes, domain knowledge can provide useful ways to discretise and aggregate data. Indeed, the high-level concepts introduced by domain experts can describe complex situations of interest involving one or various data sources. This data "wrapping" may reduce data amounts to be processed in the knowledge discovery process when complex domain knowledge is available. The second aspect enhanced by temporal abstraction is that of interpretability. Indeed, domain concepts (states and trends) can be seen as a set of high-level symbols forming a description "vocabulary" for both environment description and temporal knowledge representations.

We sum up the motivation behind the interval-based data unification approach in the following:

- The interval-based interpolation of raw discrete data into the "continuous" domain using time granularities-based temporal mapping permits to handle better time granularity heterogeneity. It allows to properly reflect data validity assumptions that are inherent to the discrete data generation process.

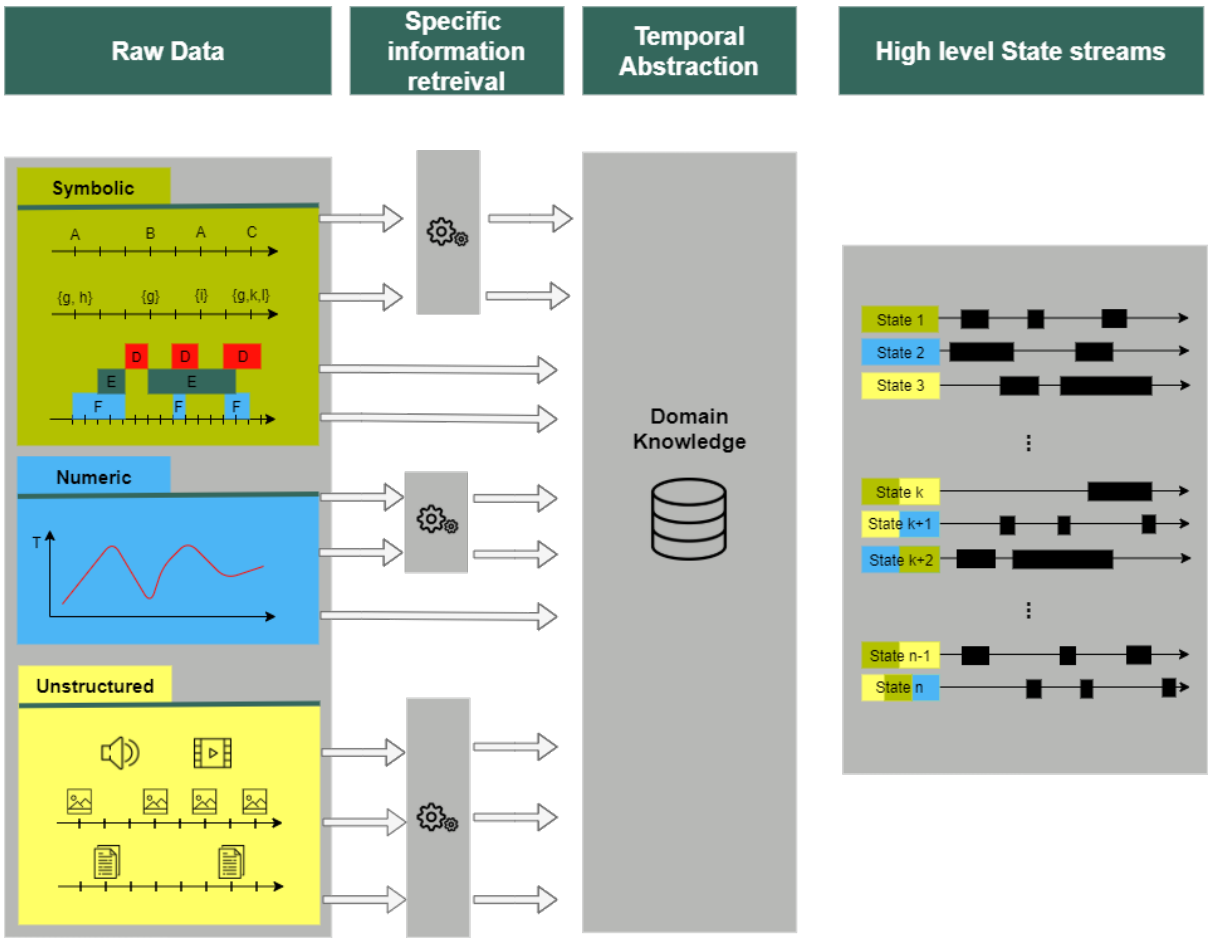


Figure 3.4.1: State streams construction process

- Subject-centred data can be converted into attribute-centred data describing the evolution of the environment through its subjects attributes. Existing knowledge discovery approaches can be straightforwardly be applied (with single sequences approaches and associated interestingness measures).
- Attribute values streams and data discretisation based on specific automatic processes and knowledge-based temporal abstraction permits to express information given by a set of heterogeneous data models with a unique interval-based, high level, easily interpretable representation.

3.4 The state streams construction framework

In the last section, we motivated our approach devised to obtain a unified interval-based representation of heterogeneous data. Hereafter, we describe the overall framework and technical details of this process that aims to produce state streams. It is composed of two main tasks: specific information retrieval and knowledge-based temporal abstraction. Figure 3.4.1 provides an overview of this framework.

One core characteristic of this framework is that it can be used online to obtain the state stream representation of the environments in a near real-time capability. The reasons for this functionality follows. First, we aim to provide the possibility to run temporal data analytics on the run. In other words, the knowledge discovery process must be capable of building temporal insights and models incrementally. Secondly, this data unification framework can be used to perform forecasting based on the discovered temporal knowledge.

3.4.1 Specific information retrieval

The first task, named *specific information retrieval* in Figure 3.4.1, aims to extract information from raw data using algorithms that are specific to each data model. At this stage, it is mandatory to run information retrieval, at least, on unstructured data to obtain a structured data representation that is at a higher level. It can be whether numeric or symbolic.

The output of information retrieval must be modelled as an attribute data stream containing values in a specific symbol set. For example, motion information provided from a video analysis will contain time point event in $\mathcal{T}_{motion} = \{MB, ME\}$ where MB and ME denotes respectively *motion begin* and *motion end*. For semi-structured data, a straightforward attribute stream approach can be considered. However, if a data source system uses heterogeneous sources providing data using, for example, different XML tags (e.g. different tags for the same physical attribute) or different JSON structures, it can be interesting to unify representations using data integration techniques. With structured data, it is also possible to use specific information detection approaches, especially for numerical values. For example, automatic discretization approaches (e.g SAX [94]) can be used for time series. Other approaches as online outlier analysis [71] can also be integrated at this stage of the process.

The result of the information retrieval stage is a set symbolic or numeric structured streams that can be formally expressed with:

$$(a, TG, S)$$

with a an attribute label, TG a temporal granularity and $S = \langle (v_i, t_i) \rangle$ a sequence of data values v_i (numeric or symbolic) anchored at time t_i (that can be a time point or an interval anchor) in the discrete-time domain defined by TG_i. It is to notice that data streams given by specific information retrieval algorithms may have heterogeneous time granularities depending on the actual raw data sampling rates and algorithmic capabilities.

3.4.2 Knowledge-based temporal abstraction

The second stage of data unification consists of building a high-level interval-based stream representation of the information given by the set structured symbolic and numeric attribute streams. Temporal abstraction in our context is devised to build a representation of the environment through a set of domain-specific temporal concepts called states. A state of an environment $\mathcal{E}$ is a particular temporal data configuration referring to a non-trivial situation of interest for the application domain. This situation can be defined with respect to data from one or several attribute streams provided by one or several data sources. Follows the formal definition.

Definition 11 (State)
Let $\mathcal{E}$ be an environment monitored by data source system DS and $AT = \{(a_i, TG_i, S_i)\}$ a set of attribute streams extracted from DS. Formally, a state A_s is defined as a predicate:

$$A_s : \mathcal{T}, AT^n \rightarrow \mathbb{B}$$

where $\mathcal{T}$ is the "continuous" time domain.

The result of state A_s predicate is a boolean value stating that A_s is active or inactive at a given temporal anchor $t \in \mathcal{T}$ given a subset of AT. It is to notice here that a state definition can rely on streams having heterogeneous temporal granularities that has to be taken into account while defining a state predicate. As stated before, the "continuous" time domain is, in reality, a discrete one defined by a temporal granularity TG_s defined as the least

common multiple granularities of the considered subset of AT. As a consequence, a state predicate on time anchor $t \in \mathcal{T}$, defines the validity of the predicate on a time interval $[t, t + TG_s)$.

Temporal abstraction [139] provides a broad theoretical background for knowledge-based temporal abstraction involving multiple sub-tasks (temporal context restriction, vertical temporal inference, horizontal temporal inference, temporal interpolation and temporal pattern matching) and multiple domain knowledge types (structural, classification, temporal semantic, temporal dynamic). For this work, we adopted a simplified view of TA based on simple basic operators. State predicates are defined using two types of operators: logical and temporal. Logical operators, conjunctions $\vee$ and disjunctions $\wedge$, permits to express conditions on data values. A temporal operator permits to situate data values conditions in time with respect to the temporal attribute of the state predicate. Two principal temporal operators types can be useful for predicates definition. Relative temporal operators refer to a data value that is temporally situated relatively to a temporal anchor. It infers conditions on ordering of data in S_i. Follows two relative temporal operators examples:

- **next**$(t, (a_i, TG_i, S_i))$ retrieve data value in sequence S_i that directly follows the absolute time point t in the discrete time domain defined by TG_i
- **previous**$(t, (a_i, TG_i, S_i))$ retrieve data value in sequence S_i that directly precedes absolute time point t in time the discrete time domain defined by TG_i

The other type of temporal operators is qualified as absolute. It permits to retrieve data at a particular temporal anchor that can be a time point or an interval. The output can be then a single value or a sub-sequence.

- **valueat**$(t, (a_i, TG_i, S_i))$. Provides data value of S_i having temporal anchor containing t in the discrete time domain defined by TG_i.
- **subseqat**$(t_b, t_e, (a_i, TG_i, S_i))$. Gets sub-sequence s of S_i containing values having temporal anchors t such that $t_b \leq t \leq t_e$ where t_b and t_e are absolute time points.

Some examples of states predicates that can be defined for this environment depicted in Figure 3.1.1b:

- **motionInC1**$_s(t, \{C1\}) ::= previous(t, C1) = MB$
- **increasingOccupancy**$_s(t, \{C6\}) ::= valueat(t, C6) - valueat(t-1, C6) > 0$
- **congestion**$_s(t, \{C1, C6\}) ::= (previous(t, C1) = MB) \wedge (valueat(t, C6) - valueat(t-1, C6) = 0)$

Predicates can be simple conditions on a single raw data sequence values as $motionInC1_s$ that reports on simple motion activity (the last event produced by C1 at the timestamp t is MB=Motion Begin). States can also report on trends as $increasingOccupency_s$, or can also integrate data from various sensors as for $congestion_s$.

The set of defined state predicates encoding domain knowledge are stored in a knowledge base and used to build state streams.

Definition 12 (State stream)

A state stream A corresponding to state A_s is defined as

$$A = (A_s, \langle [t_{b_i}, t_{e_i}) \rangle)$$

such that $\forall t_{b_i}, t_{e_i} \in \mathcal{T}, t_{b_i} < t_{e_i} < t_{b_{i+1}}$ and $\forall t \in [t_{b_i}, t_{e_i}) \mid A_s(t, \lambda) = True$ with $\lambda \subseteq AT$ with AT an

attribute stream set.

The size of a state stream A noted $\#A$ corresponds to the number of its intervals. The length of a state stream is the sum of its active intervals duration:

$$\mathbf{len}(A) = \sum_{[t_b,t_e)\in A} (t_e - t_b)$$

For example, the state stream corresponding to $motionInC1_s$ from Fig. 3.1.1b is $motionInC1 = (motionInC1_s, \langle[1,4),[13,14)\rangle$, with $\#motionInC1 = 2$ and $len(motionInC1) = 4$.

3.4.3 Operations on states streams

We define also three operations on state streams: intersections, unions and (α,β)-temporal transformation.

Definition 13 (Intersection of state streams)
The intersection of two state streams A and B, noted $A \cap B$, is a state stream containing intervals where both A and B are active.

$$A \cap B = (A_s \wedge B_s, \langle[t_{b_i}, t_{e_i})\rangle$$

such that $\forall t \in [t_{b_i}, t_{e_i}), \exists [t_{b_j}, t_{e_j}) \in A, [t_{b_k}, t_{e_k}) \in B$ such that $t \in [t_{b_j}, t_{e_j})$ and $t \in [t_{b_k}, t_{e_k})$. Computing the intersection is done in $\Theta(Max(\#A, \#B))$

Definition 14 (Union of state streams)
The union of two state streams A and B, noted $A \cup B$, produces a new state stream containing the intervals where A or B are active.

$$A \cup B = \left(a \vee b, \langle[t_{b_i}, t_{e_i})\rangle\right)$$

such that $\forall t \in [t_{b_i}, t_{e_i}), \exists [t_{b_j}, t_{e_j}) \in A, [t_{b_k}, t_{e_k}) \in B$ such that $t \in [t_{b_j}, t_{e_j})$ or $t \in [t_{b_k}, t_{e_k})$. Computing the union of two state streams can be done in $\Theta(Max(\#A, \#B))$.

It is to notice that intersections and unions of state streams can be built within the temporal abstraction process using the conjunction and disjunctions of predicates A_s and B_s respectively.

Definition 15 ((α,β)-temporal transformation)
Let $A = (A_s, \langle[t_{b_i}, t_{e_i})\rangle)$ be a state stream. An (α,β)-temporal transformation is a linear operation on a state stream's intervals endpoints. The (α,β)-temporal transformation of A provides a state stream, noted $A^{(\alpha,\beta)} = \left(A_s, \langle[t_{b_j}, t_{e_j})\rangle\right)$ and defined with: $\forall [t_{b_j}, t_{e_j}) \in A^{(\alpha,\beta)}, \forall t \in [t_{b_j}, t_{e_i}), \exists [t_{b_i}, t_{e_i}) \in A$ such that $t \in [t_{b_i} - \alpha, t_{e_i} - \beta)$. α is called intervals expansions. β is called intervals reductions. The (α,β)-temporal transformation of a state stream A is done in $\Theta(\#A)$.

The interpretation of an interval $[t_b, t_e)$ of an (α,β) transformed state stream $A^{(\alpha,\beta)}$ is the following:

- State A will be activated after a duration α from t_b and will be deactivated after a duration β from t_e.

Intersections, unions and (α,β)-temporal abstractions can be composed to build more complex state streams with specific semantic interpretation corresponding to a wide range of situations. Follows two basic examples using intersection or union with (α,β)-temporal transformation:

- An interval of $A \cap B^{(\alpha,\beta)}$ denotes that A is active and B will be activated after a duration α from the beginning of A and deactivated after a duration β from the end of A.
- An interval $[t_b, t_e)$ of $A \cup B^{(\alpha,\beta)}$ describes that whether A is active or B will be activated after a duration α from t_b and deactivated after a duration β from t_e.

Here, we want to stress that the composition of state streams with intersections, unions and (α, β)-temporal transformations denotes an interval of times where specific situations occur or will occur. From a high-level point of view, composed state streams are temporal "narratives" composed by situations of interest that are defined by the user (the environment states) the are maintaining quantitative information through (α, β)-temporal transformations. Our objective in this work is to extract temporal knowledge permitting to describe typical relations between these kinds of quantitative temporal narratives.

3.5 Discussion

In this section, we proposed and described the use of a unified interval-based representation of temporal information given by a heterogeneous data sources system. The output of the described framework is a set of interval-based streams containing each portion of time where a high-level state is active. This state-based approach provides an interpretable temporal description of a given environment. It also defines a high-level vocabulary for to-be-discovered temporal knowledge that integrates domain-specific concepts permitting more straightforward interpretability of novel insights.

The use of temporal abstraction to integrate domain insights into the process of building a high-level description of a given environment can be seen as an interpretation of temporal phenomena and environment characteristic through the concepts of a given domain. One opportunity that is given by this approach that could be investigated is the use of multiple domain knowledge to interpret the same set of temporal facts given by a data source system. The intuition here is that significant temporal relations and correlations may exist between concepts belonging to different domains. Indeed, the use of multiple domain knowledge may permit to enlight each of single domains perception "blind spots" and produce a more detailed temporal description from which novel insights can be extracted.

Nevertheless, it remains several open issues that may impact the knowledge discovery process. The first one is:

Are all information provided in the heterogeneous data sources system contained in the set of state streams?

While it is hard to provide a clear answer for unstructured data (e.g. are all visual pieces of information in a video encoded in the state stream representation?), this question remains valid for structured, semi-structured or even with the attribute stream representation used in our framework. Indeed, with our approach, there is no guarantee that all raw data values (or attribute values) that are provided by the data source system are represented in the obtained state stream set. In other terms, the question here is the existence of a bijection between raw-data and their state stream representation. One direction to obtain insight about the information use issue is to design "data usage" indicators that permit to quantify information in a certain high-level state-based description.

The second issue is somewhat the last question opposite.

Are the information described by a state stream S contained in another state stream S?

The problem here is that of information representation redundancy that could affect the knowledge discovery process. Indeed, if the same piece of data is involved in the definition of two state streams, to what extent the information redundancy will produce trivial temporal knowledge? One may obtain states correlations due to the influence of a single attribute that is involved in the predicate definition of both states. In this work, we make the hypothesis that state streams are an abstraction of mutually exclusive raw data.

Hypothesis 4

In a state-based description of an environment, all state streams contain mutually exclusive information.

Part II

DISCOVERING QUANTITATIVE TEMPORAL DEPENDENCIES

4

Temporal Patterns Discovery: an overview

Contents

4.1 Introduction

Mining temporal data has received much interest in the last decades. Indeed, the need to obtain insight about the evolution of processes is a necessity in various application domain as healthcare, retail, telecommunications. As for static data (i.e. without a temporal dimension), temporal data can support multiple data mining task including clustering, classification ... We refer the reader to [88, 131] for a more global overview of temporal knowledge discovery and data mining.

In this section, we focus on Temporal Pattern Mining (TPM), i.e. algorithms devised to extract interesting temporal relations between events/items from explicitly temporally ordered data. We propose to categorize approaches of this field following the main criteria of temporal primitives: time points and time intervals. Each temporal primitive defines a different set of possible pairwise temporal relations between data values and, by extension, pattern languages. In addition to the primitive categorization, and in order to provide a comprehensible overview of this research field, this section will use different categorization criteria. We introduce them in the following.

Input data formats. As discussed in Section 2.4, temporal data mining problems definitions also depend on how temporal data sets are formatted. Indeed, while using the same pattern model, two approaches can tackle different mining problems and theoretical issues. Categories for this criteria are the following: *Sequence databases* and *single sequences*.

Temporal information. Temporal relations between data values can whether be *qualitative* or *quantitative*. While the former reports on qualitative relations (e.g *before, after, during*, etc.), the later maintains information about time lags and duration (e.g *A follows B after n time units*). We consider that quantitative relations are more expressive and provide more insights about a given temporal relation as the maintained quantitative information can be a discriminant factor. Follows two examples.

> **Example 5**
>
> *In a medical context, let us consider a sequence database of symptoms and disease records. A research team is interested in finding patterns describing temporal relations between symptoms and diseases in order to provide more accurate medication and treatments. In the available database, the pattern SYM_A SYM_B (symptom A precedes symptom B) is interesting (according to a given significance criteria). Let us assume that half of the occurrences of succession SYM_A SYM_B are followed by disease DIS_C and the other half by disease DIS_D. In this example, DIS_C follows SYM_A SYM_B if the duration between SYM_A and SYM_B is around d and DIS_D follows SYM_A SYM_B if the duration between SYM_A and SYM_B is approximately d' with $d' = 2 * d$. Using qualitative pattern permits to extract temporal relation between the succession of the two symptoms with two distinct diseases which are already useful. However, it does not permit to predict which of these two diseases are more likely to follow this succession of symptoms. In the other hand, maintaining temporal relations provides insight permitting to predict which disease is more likely to occur given the mined time information: if SYM_B follows SYM_A after d time units, it is more likely that patient will have disease DIS_C, if the time lag is $2 * d$ then the medical team can expect DIS_D and adapt the treatments.*

> **Example 6**
>
> *Say that an advertising agency possesses a video billboard on a road where pedestrians, bicycles and cars move along. To reduce the billboard energy consumption, the agency equips the road with four motion sensors (two at each side of the road) and uses qualitative pattern detection to turn on or off sides of the billboard: if an*

> *object moves toward the billboard, the later is turned on. The agency stores information given by motion sensors and runs a sequential pattern mining algorithm on the gathered data. In this case, the extraction of qualitative patterns seems to be useless. Qualitative successions of events are already known (and used) by the domain expert. On the other hand, quantitative patterns permit to highlight and characterize actor types based on time lags: short time lags for cars, medium for bicycles and large for pedestrians. This quantitative characterization of time lags is a novel insight that can be used by the advertising agency to personalize ads displayed in the billboard screens w.r.t actors types based on their speed (e.g. display discounts on wheels for cars, helmets ads for bicycles and nearby restaurants menus for pedestrians).*

Interestingness. As reported by Geng and Hamilton [60] interestingness measures in data mining are devised to select and rank discovered patterns according to their interest for the user. In their work, they proposed nine interestingness criteria: *Conciseness, Generality/Coverage, Reliability, Peculiarity, Diversity, Novelty, Surpisingness, Utility, Actionability/Applicability*. In the rest of this section, Geng and Hamilton [60] classification of interestingness measures will be used. For the sake of readability, definitions of useful criteria and related measures will be provided while being relevant.

The remainder of this section, we will provide an overview of both point-based and interval-based temporal pattern mining approaches. For each primitive, we will introduce the major mining problems treated in the literature, describe the most used pattern models and provide an overview of the main algorithmic approaches.

4.2 Point-based patterns

4.2.1 Patterns from Sequential Databases

Qualitative sequential patterns

In this section, we introduce the problem of mining sequential patterns from a sequence database and its extensions. Temporal patterns from point-based sequential databases have received much interest in the last decades with a considerable number of contributions. We will try to provide a comprehensive overview of this field with significant works. We refer the reader to the extensive surveys of Fournier-Viger et al. [49] for further details. This section will also introduce several key concepts to the temporal pattern mining domain useful for the remainder of this thesis.

The sequential pattern mining problem was introduced by Agrawal and Srikant [4] to tackle the problem of finding frequent sequences in a database of customer transactions. An example of such databases is given by Figure 4.2.1a. Each record associates a set of items to a customer ID and a transaction time that is considered as a time point. Sequential patterns are extracted from a costumer sequence version of this database, as shown in Figure 4.2.1b where records associate a temporally ordered list of item sets to a customer ID.

Sequential pattern mining as introduced by Agrawal and Srikant [4] can be formalized as follows. Let $I = \{i_1, i_2, \ldots i_n\}$ a set of items or symbols. An itemset X is a subset of I. The size of X, denoted $|X|$ corresponds to its items number. A itemset sequence $S = \langle X_1, X_2, \ldots X_k \rangle$ is an ordered list of itemsets. The length n of the sequence is the sum of its itemsets sizes. A sequence of length n is referred to as a n-sequence. A sequence $S_b = \langle X'_1, X'_2, \ldots X'_n \rangle$ is said to be a sub-sequence of $S_a = \langle X_1, X_2, \ldots X_m \rangle$ if there exists integers $i_1 < i_2 \ldots < i_n$ such that $X'_1 \subset X_{i_1}, X'_2 \subset X_{i_2} \ldots X'_n \subset X_{i_n}$. A sequence database $SDB = \{S_1, S_2 \ldots S_p\}$ is a set of sequences, called *customer sequence*, having each a unique ID $(1, 2, \ldots, p)$. A costumer *supports* an itemset sequence S if S is contained in the customer sequence. The support of a sequence s is defined as the portion of costumers supporting s. It is a

Custumer ID	Transaction Time	Items Bought
1	June 25	30
1	June 30	90
2	June 10	10,20
2	June 15	30
2	June 20	40, 60, 70
3	June 25	30, 50, 70
4	June 25	30
4	June 30	40, 70
4	July 25	90
5	June 12	90

(a) Customers transactions database sorted by Customer Id and transaction time

Customer ID	Customer Sequence
1	$\langle(30)\ (90)\rangle$
2	$\langle(10, 20)\ (30)\ (40, 60, 70)\rangle$
3	$\langle(30, 50, 70)\rangle$
4	$\langle(30)\ (40, 70)\ (90)\rangle$
5	$\langle(90)\rangle$

(b) Customer-Sequence Version of the database

Figure 4.2.1: A database of costumers transactions [4]

Generality/Coverage interestingness measure.

> **Definition 16 (Generality/Coverage interestingness measure [60])**
> A pattern is general if it covers a relatively large subset of data set. [It] measures the comprehensiveness of a pattern, that is, the fraction of all records in the data set that matches the pattern.

In a set of frequent sequences Σ, a *maximal* sequence is a frequent sequence that is not a sub-sequence of other sequence in Σ. Follows an example.

> **Example 7**
> *[4] Let us consider the sequence database in Figure 4.2.1b. For a user-given minimum support of 25%, $\langle(30), (90)\rangle$ is a sequential pattern as it satisfies the minimum support (appears for customer 1 and 4) and it is not a sub-sequence of an other frequent sequence. $\langle(30)(40, 70)\rangle$ is an example of sequence that not satisfies the minimum support. $\langle(30), (40)\rangle$ is frequent but not maximal as $\langle(30)(40, 70)\rangle$ is frequent.*

Sequence *Maximality* can be seen as a *Conciseness* interestingness measure.

> **Definition 17 (Conciseness interestingness measure [60])**
> A pattern is concise if it contains relatively few attribute-value pairs while a set of patterns is concise if it contains relatively few patterns [...].

Indeed, a set of sequence meeting the support threshold can contain redundant information if the user is interested in discovering the largest frequent sequences. This can lead to a massive number of sequences with redundant information (w.r.t maximal interestingness criteria): for a frequent n-sequence, up to $2^n - 1$ non-empty

frequent sub-sequences can exist. The potentially huge amount of patterns may make the results hard to understand, interpret and remember for the end-user. This is a general problem for the data mining domain called pattern flooding.

The problem of sequential pattern mining is the following:

> **Problem 4 (Sequential pattern mining [4])**
> Given a sequence database *SDB* and a user-given minimum sequence support *min-sup*, find all maximal sequences having a support greater than the user given minimum support *min-sup*.

Sequential pattern mining is an enumeration problem devised to explore the search space of all possible sub-sequences in a database and find all frequent sequential patterns. Two main exploration strategies are used for this exploration process: *breadth-first* or *depth-first* search. The search space to be explored can be represented as a lattice where specifications use two kinds of sequence extensions: sequential extensions (adding an itemset at the end of a sequence) and items sets extensions (adding an item to a sequence itemset).

Breadth-first approaches consider first all 1-sequences (i.e sequences of length 1). These sequences are extended using both sequential and itemset extension to generate and process 2-sequences. This process is repeated (i.e generate k-sequences from (k-1)-sequences) until reaching the length of the largest sequence in the database. For example, let *SDB* be a sequential database and $I = \{a, b, c\}$ the symbols used in *SDB*. Let us assume that the largest sequence in *SDB* have a length of 4. A breadth-first algorithm using sequential extensions will first consider sequences $\langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle$. Then, it will process 2-sequences $\langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{a, b\}\rangle, \langle\{a, c\}\rangle, \langle\{b\}\{a\}\rangle, \langle\{b\}\{b\}\rangle, \langle\{b\}\{c\}\rangle, \langle\{b, c\}\rangle, \langle\{c\}\{a\}\rangle, \langle\{c\}\{b\}\rangle, \langle\{c\}\{c\}\rangle$. After that, 3-sequences and 4-sequences will be considered successively. The exploration stops at this stage as no sequence in *SDB* has a length greater than 4.

Depth-first algorithms assume a total order (e.g lexicographical order) for the set of symbols I in *SDB*. For example, with $I = \{a, b, c\}$, a possible order to be used is $c \succ b \succ a$. A depth-first strategy will start with 1-sequences and then perform recursively itemset extensions and sequence extensions with respect to the lexicographical order. When a pattern can not be extended, the algorithm cancels the last extension to generate other sequences using the next item w.r.t the order. Assuming the former example configuration, the sequence processing order for a *depth-first strategy* using itemset extensions before sequence extension, with a maximal sequence length of 2 will process patterns in the following order: $\langle\{a\}\rangle, \langle\{a, b\}\rangle, \langle\{a, c\}\rangle, \langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{b\}\rangle, \langle\{b, c\}\rangle, \langle\{b\}\{a\}\rangle, \langle\{b\}\{b\}\rangle, \langle\{b\}\{c\}\rangle, \langle\{c\}\rangle, \langle\{c\}\{a\}\rangle, \langle\{c\}\{b\}\rangle, \langle\{c\}\{c\}\rangle$.

This exploration problem is considered as hard since it defines huge search spaces, even for small databases containing a small set of sequences. One can convince himself of the last statement noticing that a sequence having n items can contain up to $2^n - 1$ distinct non-empty sub-sequences. As a consequence, the naive approach consisting of computing the support of all possible sub-sequences in a database and provide only those meeting the coverage threshold is not efficient nor realistic for most of the real-life databases. Efficient sequential pattern mining algorithm must be designed in order to avoid exploring the entire sub-sequences search space. The basic mechanism used to prune this search space is based on the *Apriori property*. It states that if s_b is a sub-sequence of s_a then s_a have an equal or lesser support than s_b. As a consequence, if a sequence s is found to be not frequent, then all its extensions (or super-sequences) can be pruned.

The first sequential pattern mining algorithm was proposed by Agrawal and Srikant [4], *AprioriAll*. It uses a breadth-first search strategy with candidate generation. First, the algorithm makes a pass over the sequence database to count support of all 1-sequences and keep only the frequent ones in memory. Then, the algorithm uses frequent k-sequences to generate (k+1)-sequences candidates by combining pairs of k-sequences sharing all

$\{a\}$	
ID	**Timestamps**
1	10
2	30
3	20,40,50
4	10

$\{b\}$	
ID	**Timestamps**
1	20
2	30
3	30,50
4	10,30

$\{a\}$ before $\{b\}$	
ID	**Timestamps**
1	20
2	
3	30,50
4	30

$\{ab\}$	
ID	**Timestamps**
1	
2	30
3	50
4	10

Figure 4.2.2: An IDList example: a vertical representation of a sequential database

but one item and, then, dropping sequences containing non-frequent sub-sequences using the Apriori property. After that, a pass over the database is made to compute the support of each of the generated candidates and only frequent sequences are kept in memory for the next iteration. Candidate generation and support count are repeated until no frequent candidate is found. Maximal patterns are computed in a post-processing phase on the set of frequent sequences. In order to improve the efficiency of *ApioriAll*, the authors proposed the well known *GSP* algorithm [143] that uses a different candidate generation process and an improved algorithm for support counting (claiming an up to 20% improvement rate). This first type of sequential pattern mining can be labelled as *breadth-first candidate generation* algorithms. This kind of algorithms suffers from several limitations including multiple database scans that come with high computational cost with large databases, the generation of non-existent candidates (i.e. sub-sequences that are not occurring in the database) that increases the search space or the necessity to keep all k-frequent patterns in memory for candidate generation which can consume a tremendous amount of memory.

Another sequential pattern mining approach family is that of the SPADE algorithm proposed by Zaki [168] inspired by ECLAT [167] an algorithm devised for association rules mining. The main idea of this type of algorithms is to use a vertical representation of sequential databases. For each i in the itemset of symbols I, the vertical representation associates a sequence identifier *sid* to the list of occurrence times of i, called *IDList*. We describe in Figure 4.2.2 *IDLists* example for itemsets $\{a\}$ and $\{b\}$ from sequence database depicted in Figure 4.2.1b. IDLists are used to compute straightforwardly (i.e. without a database pass) the support of sequential patterns using temporal joins. For example, in Figure 4.2.2 the support of $\{a\}$ before $\{b\}$ and $\{ab\}$ can be obtained directly from IDLists of $\{a\}$ and b. Using this representation, the algorithm requires, at most, three database scans which minimize the I/O cost considerably compared to algorithms as GSP or AprioriAll. The SPADE algorithm can use both *breadth-first* or a *depth-first* strategies. The first permits a more efficient pruning as all IDLists of a level are available while the second is more memory efficient as it needs to keep track of only intermediary IDLists. We refer the reader to the original paper [168] for further details on this approach. Many extensions were proposed including those using optimization based on a bitmap representation of IDLists (e.g. [16, 17, 69, 134, 141]) permitting to use bitwise operations rather than temporal joins. Another vertical representation based contribution was proposed by Fournier-Viger et al. [48], called CM-Spade, that use co-occurrence maps to store frequent 2-sequences occurrence. This permit to prune (i.e. not compute the temporal join) on sequences having a non-frequent suffix.

In addition to the former types of sequential pattern mining, another category stands for *pattern growth* algorithms. The most popular algorithm belonging to this category is probably PrefixSpan that was proposed by [120] (and its former version FreeSpan [74]). This algorithm was inspired by FPGrowth [74] in that it uses the database projection structure. These algorithms use a *depth-wise* search strategy according to a total order (e.g. lexicographic order). It proceeds as follows. First, the algorithm scans the database to find frequent items

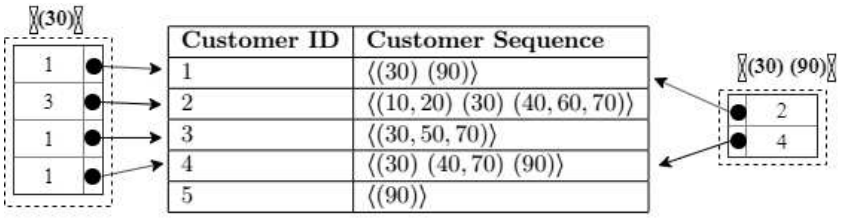
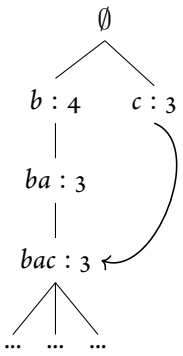


Figure 4.2.3: Index set of pattern $\langle(30)\rangle$ and $\langle(30)(90)\rangle$ used by the MEMSIP algorithm [95]

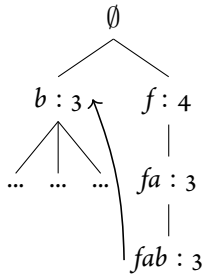
or 1-sequences. During the *depth-first* exploration, for a frequent k-sequence *s*, PrefixSpan builds the projected database of *s*. This projection contains the set of sequences containing the sub-sequence *s* from which all itemsets preceding *s* are removed. Then PrefixSpan scans this projected database to compute the support of all (k+1)-sequences and selects the frequent ones. The depth-first strategy continues with building the projected database of the first frequent (k+1)-sequence (w.r.t total order) starting from the projected database of *s*. These operations are recursively repeated in a depth-first manner until all sequential patterns are found. It is to notice that the larger is a sequence, the lesser is the cost of computing the projected database. *Pattern-growth* algorithms have the advantage of computing only sequences that appear in the database. This is a considerable gain in search space size compared to approaches using candidate generation as GSP or even vertical approaches. The main limitation of *pattern-growth* algorithm is the cost of building the projected database that can use a considerable amount of memory. Some contributions proposed an optimization based on the usage of pointers to build projected databases rather than making hard copies [121].

Lin and Lee [95] proposed the MEMory Indexing for Sequential Pattern mining (MEMISP) approach that uses an all-in memory approach needing a single database scan and avoiding candidate generation. As its name suggests, this algorithm uses an index data structure to store information about the frequent pattern. Figure 4.2.3 shows the indexes for patterns $\langle(30)\rangle$ and $\langle(30)(90)\rangle$. It is composed by the pattern occurrence position in the sequence and a pointer the this later. Notice that the index structure does not refer to all sequences in the original database. The MEMISP proceeds as follows. First, it scans the database once to maintain the entire database in memory, to find frequent items and build their indexes. Then, the algorithm performs the following steps recursively for each frequent (k)-pattern associated with an index: (1) counts the support for all existing (k+1)-patterns extending the (k)-pattern by one item (2) for each frequent (k+1)-pattern, the algorithm computes its index (from the index of its (k)-pattern prefix). These operations are repeated until no frequent patterns are found. Notice that the Apriori property is used to prune non-frequent patterns. The main limitation of this approach is that it has the maintain in memory the entire database in addition to patterns' indexes. To be able to process large database, the authors proposed to divide the original database into several chunks that can stick in memory, run the MEMISP algorithm and then scan the original database to compute the actual support. Experiments showed that the MEMISP algorithm outperforms both *PrefixSpan* [120] and *GSP* [143]. However, this approach is hardly relevant to huge databases as it needs greater memory consumption.

The former sequential pattern mining approaches aim to find all sequential patterns based on algorithmic improvements and novel databases representations. In order to accelerate the mining process, reduce the pattern flooding problem and make long patterns discovery feasible (from a memory consumption perspective), many contributions have proposed to discover subsets of frequent sequences directly. They use constraints meeting the end-user interest and the conciseness interestingness. Indeed, the algorithms described above may integrate a post-processing phase to compute maximal patterns (or other conciseness criteria), but all of them are devised



(a) Backward sub-pattern



(b) Backward super-pattern

Figure 4.2.4: Backward sub-sequence and backward super-pattern based pruning. Tree's nodes are represented as *sequence* : *support*.

to mine all frequent sequences. The integration of constraints permits to provide pruning criteria permitting to accelerate the mining process by several orders of magnitude and make the result set more easily interpretative as containing less and more accurate (w.r.t the user) patterns.

Among the used constraints, several works proposed to mine directly concise representations of frequent sequential patterns. These contributions aim to discover straightforwardly a subset of frequent sequential patterns that is considered to be *enough* informative for the user's task. One kind of concise representation is that of *maximal patterns* (that we already defined higher). Many contributions have tackled the problem of mining straightforwardly maximal patterns, e.g. [6, 39, 45, 48, 57, 98, 99]. One limitation of maximal patterns is that they loose information. Indeed, all sequential patterns can be extracted from the set of maximal patterns, but their support needs a database scan. Another subset of frequent sequential patterns is that of *generator patterns* [46, 56, 79, 96, 125]. A sequence s is said to be *generator* if there is no sub-sequence $s_{sub} \sqsubseteq s$ having the same support as s. Another common definition, uses the concept of *equivalence class* introduced by Pasquier et al. [118] for frequent itemsets mining. An *equivalence class* contains all patterns supported by the same set of sequences. Generator patterns are the minimal members of an *equivalence class*. The maximal members of an equivalence class are called *closed patterns* that are the third representation that we treat. A closed sequential pattern s in a set of patterns S is a pattern such none of its super-sequences in S have the same support.

The subset of closed patterns is interesting as it provides a *lossless* representation of the set of frequent sequences. That means that the support of every frequent sequence can be derived from the set of closed frequent sequences. This is not the case for maximal patterns. The first work dealing with the problem of closed frequent sequence mining is CloSpan of Yan et al. [162] that extends the PrefixSpan. The integrated closure checking permits to perform efficient search space pruning based on *backward sub-patterns* and *backward super-patterns*.

In Figure 4.2.4 the support of sequence $\langle\{b\}\{a\}\{c\}\rangle$ is 3. The depth-first strategy using the lexicographical order explores this pattern (and its super-sequences) before $\langle\{c\}\rangle$ that have the same support than $\langle\{b\}\{a\}\{c\}\rangle$. This means that every occurrence of c in the database is involved in pattern $\langle\{b\},\{a\}\{c\}\rangle$. Thus any super-sequence having c as a prefix can not be closed: it will be included in a sequence previously obtained by *extending* the $\langle\{b\},\{a\}\{c\}\rangle$ sequence. Moreover, they will have the same support. As a consequence, the algorithm can avoid computing any extension of sequence $\langle\{c\}$. $\langle\{c\}\rangle$ is said to be the backward sub-pattern of $\langle\{b\},\{a\}\{c\}\rangle$. The same reasoning can be applied to backward super-pattern of Figure 4.2.4. Among further contributions, some extends also the PrefixSpan algorithm as[154, 155] and others uses the vertical database representation as

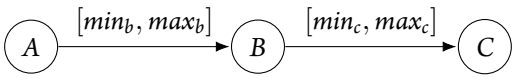


Figure 4.2.5: Graphical representation of a Delta pattern [165]

[53, 65, 90, 148].

Another type of constraints that can be used in sequential pattern mining is that of *contextual* constraints. They differ from concise representation constraints by the fact that the end-user defines them following its prior knowledge. They permit to obtain a sub-set of frequent sequences fitting previously known properties. The exploration process can then benefit from these constraints to perform search-space pruning (rather than applying them in a post-processing stage) to gain in efficiency. Pei et al. [122] studied the incorporation of several constraints in the frequent sequence mining process such as *Maximum/Minimum length constraint*, *Item constraints* (selection of items to appear in a pattern) or *Aggregate constraints*. Other works (e.g) [8, 58, 59, 66] utilized user-given regular expressions. Another form of constraints that are more relevant to the present work is that of temporal constraints. They are user-defined conditions on temporal relations between itemsets in a sequence. The most known temporal constraints are *min-gap*, *max-gap* and *duration* (i.e time lag between the first and last item must be less than a specified threshold). Algorithms such as the early GSP [143], or more recent works, e.g. [20], had used such constraints.

Quantitative sequential patterns

It is to notice that sequential pattern as defined by Agrawal and Srikant [4] are purely quantitative patterns. All works use the pattern definition cited previously. An exception can be made for algorithms using temporal constraints that are, somewhat, temporal information indicating sequential temporality. However, this information is not discovered but is user-given, yet, it is not to be considered as novel knowledge.

Few contributions are interested in mining quantitative frequent sequences. The problem of mining quantitative patterns can be called *Temporal Sequence Mining* and is to be considered as an extension of Sequential Pattern Mining. To the best of our knowledge, Yoshida et al. [165] were the firsts to model quantitative patterns and to propose a method to find *frequent time lags*. They proposed the *Delta pattern* model where duration between neighbouring itemsets in a pattern is modelled as a range. They are defined as follows. Let I be a set of symbols and SDB a timestamped itemsets sequence database over I. A *delta pattern* is a sequential pattern:

$$DP = \langle (X_i, \Delta_i) \rangle$$

where $X_i \subset I$ and $\Delta_i = [min_i, max_i]$ such that time lag δ_t between X_i and X_{i-1} is between min_i and max_i. A delta pattern supports a sequence $S = \langle (Y_i, t_i) \rangle$ with $Y_i \subset I$ and $t_i \in \mathbb{R}^+$ if $\exists p_1 < p_2 < p_3 < \ldots < p_n$ such that $X_1 \subset Y_{p_1}, X_2 \subset Y_{p_2} \ldots X_n \subset Y_{p_n}$ and $min_2 \leq t_{p_2} - t_{p_1} \leq max_2 \ldots min_n \leq t_{p_n} - t_{p_{n-1}}$. Figure 4.2.5 provides a graphical representation of a Delta Pattern.

Yoshida et al. [165] proposed a *breadth-first candidate generation* algorithm similar to GSP that maintains lag information and integrate temporal constraints. In order to find frequent *Delta patterns*, the authors proposed to perform clustering, based on a CF-Tree [173], on time lag between successive itemsets. The CF-Tree is built during the scan of the database. As all *breadth-first* exploration, the proposed algorithm suffers from multiple database scans that can slow significantly the mining process.

Another early work is that of Chen et al. [26] where quantitative information is represented as single value reporting on an exact time lag. They proposed both an Apriori candidate generation, I-Apriori, and a pattern-growth, I-PrefixSpan, approaches. The main limitation of this work is considering the exact time lags. Indeed, temporal successions of events may come with slight temporal variations that cannot be captured with this approach. For example, say that a pattern $\langle\{a\}\{b\}\rangle$ occurs with time gaps that slightly vary from 2 to 4 time units. The algorithm will count support for 2, 3 and 4 separately. The support of each of these time-lagged occurrences may not fit the coverage criteria leading to an underestimation of the "importance" this item succession. To solve this problem, Chen and Huang [25] proposed to use fuzzy logic to use a set of *linguistic terms* such as *Short*, *Middle* and *Long* defined with membership functions. For example, the term *Short* can be defined with the following function:

$$\mu_{Short}(t) = \begin{cases} 1 & t \leq 2 \\ \dfrac{15-t}{13} & 2 < t < 15 \\ 0 & t \geq 15 \end{cases}$$

These membership functions are used in order to determine the degree at which a sequence supports a particular pattern. Using an adequate fuzzy-support, the authors proposed an Apriori algorithm, FTI-Apriori, devised to discover fuzzy sequential patterns. An example of such patterns can be: $(a, Short, b, Long, e)$ denoting that a is shortly followed by b and e follows after a *long* interval. It is to notice that patterns given by this approach cannot be qualified entirely as quantitative as they do not provide numerical values for temporal information. Moreover, user given membership functions for temporal semantics may be difficult to set.

Following the same logic, Hirate and Yamana [80] proposed to use a user-defined time granularity Δ to group patterns occurrence that fit into intervals of interest. For example, say that the user wants to assess a pattern a then b with a Δ of 1 day, all occurrences of b following a within 1 day will be counted together, occurrences of b following a between 1 and days are considered together, and so on. They proposed an extension of PrefixSpan, including the time granularity parameter. Again, this approach does not discover *typical* time lags but assesses the support the frequent sequences w.r.t to a user-given granularity.

Giannotti et al. [62] [63] proposed the *Temporally Annotated Sequence* (TAS) pattern model where quantitative information refer to a typical time lags. They are defined as follows.

> Given a set of items I, a temporally annotated sequence of length n [...] is a couple $T = (\bar{s}, \bar{a})$ where $\bar{s} = \{s_1, s_1, ...s_n\}$ is a ordered list of itemsets $\forall i \in [1, n], s_i \subset I$ and $\bar{a} = \{a_2, a_3, ...a_n\}$ is an ordered list of temporal annotation such that a_i is the time lag between s_{i-1} and s_i.

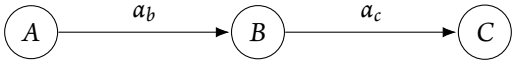


Figure 4.2.6: Graphical representation of a Temporally Annotated Sequence [62]

Figure 4.2.6 is the representation of a 3-TAS between A, B and C. The containment relation between temporally annotated sequences is defined w.r.t a time threshold τ. A TAS $T_1 = (\bar{s}_1, \bar{a}_1)$ is contained in an other $T_2 = (\bar{s}_2, \bar{a}_2)$ if $\bar{(s)}_1$ is a sub-sequence of $\bar{(s)}_2$ and its itemsets time lags correspond to their corresponding itemsets in T_2 with a variation of at most τ. Let us consider $T_1 = (\langle\{a\}, \{b\}, \{c\}\rangle, \langle 5, 9\rangle)$ and $T_2 = (\langle\{a\}, \{b, d\}, \{f\}, \{c, g\}\rangle, \langle 3, 3, 6\rangle)$ represented as follows:

$$T_1 : \{a\} \xrightarrow{5} \{b\} \xrightarrow{10} \{c\}$$

$$T_2 : \{a\} \xrightarrow{3} \{b, d\} \xrightarrow{3} \{f\} \xrightarrow{6} \{c, g\}$$

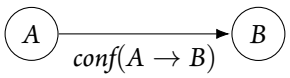
In this example, $\overline{s_1}$ is a sub-sequence of $\overline{s_2}$. T_1 is not contained in T_2 for $\tau = 0$: time lag between $\{a\}$ and $\{b\}$ is 5 for T_1 and 3 for T_2. T_1 is contained in T_2 for $\tau = 2$. Notice that the time lag between $\{b\}$ and $\{c\}$ in T_2 is the sum of time lags of consecutive itemsets $3 + 6 = 9$. It is to notice that a frequent qualitative sequence (without annotations) may not lead to a frequent TAS if its occurrences in the sequence database have sparse time lag not allowing annotations $\overline{a}$ to be close enough from each other in $\mathcal{R}^n_+$ for a sequence with n itemsets. In this case, the frequent sequence has no typical temporality. The process of mining TASs is devised to group sequences having similar annotations. Thus, a frequent sequence whose annotations are dispersed will not lead to any frequent *TAS*. The authors proposed to see this task for a fixed-length sequence as a density estimation problem. For a given n-sequence T, each of its occurrences annotations can be associated with a single point in R^n_+ where each temporal annotation corresponds to a dimension. Considering the inclusion property of parameter τ, if there exists a typical annotation (or temporality), sequences points in R^n_+ will be included in a dense hypercube with sides of length 2τ. This main idea was integrated into a PrefixSpan extension, called *MiSTA*. It performs the same steps as *PrefixSpan* for the depth-first exploration (including projected databases) and perform density estimation to find specific annotations. The algorithm also prunes infrequent annotations to speed up the exploration process.

Sequential rule mining

One interesting extension of the sequential pattern mining problem is that of *sequential rules* that is an extension of the well studied *Association Rule Mining*. A sequential rule between a premise A and a conclusion B is noted $A \rightarrow B$ and reports on a sequential relation: A is followed by B. A sequential rule is generally associated with a traditional support in addition to a confidence measure defined as:

$$conf(A \rightarrow B) = \frac{supp(\langle A, B \rangle)}{supp(\langle A \rangle)}$$

The confidence measures the conditional probability of obtaining B *after* the occurrence of A.



Confidence can be considered as a *Reliability* interestingness measure according to Geng and Hamilton [60] taxonomy.

Definition 18 (Reliability[60])
A pattern is reliable if the relationship described by the pattern occurs in a high percentage of applicable cases.

Sequential rules address the main limitation of sequential patterns. Indeed, even if a sequence may frequently appear in a sequence database, it may have a low confidence measure. As a consequence, frequent sequences may have low predictive power. The confidence assessment of sequential rules, in addition to the support, provide information about the reliability of succession of items which is a key insight for prediction and decision making. A sequential rule is, then, considered to be interesting if its support and confidence are respectively greater than user given minimum support and minimum confidence. Several variations of the sequential rule mining from sequences databases problem exist.

The first category of sequential rules is that considering the premise and the conclusion to be sequences: $A \rightarrow B$ such that $A = \langle\{X_1, X_2 \ldots X_n\}\rangle$ and $A = \langle\{Y_1, Y_2 \ldots Y_m\}\rangle$ where $\forall X_i, \forall Y_j, X_i \subset I, Y_i \subset I$. The first work addressing this problem was proposed by Spiliopoulou [142], whose idea is to extract sequential rules in post-processing after mining the full set of frequent sequences. Some redundant rules are pruned in a post-mining phase. Later, Lo et al. [97] investigates the extraction of a compressed and non-redundant set of sequential rules considering smaller representations of frequent sequences as the set of generator patterns or closed patterns. Authors claimed a huge improvement on scalability and a significant reduction of the number of found rules (avoiding pattern flooding) in comparison with extracting the full set of rules from the full set of frequent sequences. However, the proposed algorithm also performs rule extraction as a post-processing phase. Van et al. [149] proposed two algorithms as improvements for mining the full set of sequential rules. The first approach *MSR-ImpFull* uses an ascending sort of frequent sequences w.r.t to their size so that sequences having X as a prefix are located after X. For a sequence s, the algorithm passes over the following sequences and tries to form rules having s as premises. The second algorithm *MSR-PreTree* uses a pattern-growth algorithm to build a prefix tree of frequent sequences. It uses a depth-first exploration to compute sequential rules. An extension of this algorithm, *IMSR-PreTree*, was proposed by the same team [150]. It mainly includes pruning techniques in the prefix tree exploration based on the following theorem

Theorem 1 ([150])

Given three nodes n_1, n_2 and n_3 in a prefix tree such that n_1 is the parent of n_2 and n_2 is the parent of n_3. If $\frac{sup(n_2)}{sup(n_1)} < minconf$ then $\frac{sup(n_3)}{sup(n_1)} < minconf$.

This theorem states that if Y is a child node of X and $\frac{sup(Y)}{sup(X)} < minconf$ then all child nodes of Y, Z_i, cannot form a rule with X. This theorem is used for pruning, potentially large, portions of the prefix-tree which accelerate the exploration process.

The second category of sequential rules $A \rightarrow B$ considers sequence premises $A = \langle X_1 \ldots X_n\rangle$ with $\forall i, X_i \subset I$ and an itemset as a conclusion $B \subset I$ as a conclusion [30]. Chen and Lee [30] proposed a projected database approach. For each k-sequence projected database, the algorithm builds (k+1)-sequences projected databases and computes frequent itemsets. A sequential rule is generated with the k-sequence s if a frequent itemset i is found in its projected database of the form $s \rightarrow i$ and is considered as a result if its confidence is greater than the minimum user-given threshold.

Another category of algorithms is that of sequential rules between itemsets: $A \rightarrow B$ such that $A \subset I$ and $B \subset I$ [43, 44, 47].The items in A and B are not ordered. Let us illustrate this with the following simple sequence.

$$\langle\{a\}\{b\}\{c\}\{d\}\rangle$$

Examples of sequential rules supporting this sequence are: $\{a\} \rightarrow \{d\}$, $\{a, b\} \rightarrow \{c\}$ and $\{a, b\} \rightarrow \{c, d\}$ To the best of our knowledge, the first work to address this problem was Fournier-Viger et al. [43] with CM-Rules. Their main idea is to consider a sequence database (with temporal information) as a transaction database that ignores temporal information (i.e. a sequence is considered as an itemset), find association rules with an algorithm such as Apriori of Agrawal et al. [5] and finally scan the original temporal sequences database to compute sequential support and confidence to eliminate rules that not correspond to sequential relations. The main limitation of this algorithm relies on the number of intermediate discovered association rules. The larger is this number, the more extracting sequential rule is executed with a greater cost. Moreover, this approach may gener-

ate a large number of association rules corresponding to no frequent sequence in the database. Another approach proposed by the same team proposed RuleGrowth [44] that is a PrefixSpan extension and ERMiner [47] using the equivalence class concept discussed higher.

4.2.2 Patterns from a single sequence

The second main category of temporal pattern mining problems concerns mining patterns from a single sequence. They either temporally describe a single subject attributes/events over time or attribute/events generated by multiple subjects without any separation between them. While sharing similar modelling concerns with pattern discovery from sequential databases (e.g. pattern models), extracting patterns from a single sequence poses several different theoretical issues. In the remainder of this section, we refer to a single sequence as a temporally ordered list of items belonging to a set of symbols I:

$$S = \langle (A_1, t_1), (A_2, t_2) \ldots (A_n, t_n) \rangle$$

where $\forall i, A_i \in I, t_i \in \mathbb{R}^+ \leq t_{i+1}$. T_b and T_e will denote the starting time and the ending time of a sequence, respectively. As a consequence, $\forall i, T_e \leq t_i \leq T_b$.

One of the early, and famous, contribution in this field was proposed by Mannila and Toivonen [101] with the well known Episode Mining problem. Episodes are defined as follows. An episode α is a triple $(V, \leq, g)$ where V is a set of nodes, $\leq$ is a partial order on V and $g : V \rightarrow I$ is a mapping associating each node of V to a symbol in I. This definition comes with a straightforward inclusion definition. Informally, an episode A is a sub-episode of B if it includes all nodes of B respecting the order $\leq$. Notice that serial episode (i.e. with $\leq$ is a total order) can be associated with sequential patterns, while parallel episodes (where $\leq$ is trivial) to itemsets.

Generally speaking, with single sequences, the frequency cannot be assessed straightforwardly with the traditional sequential pattern mining support. As a consequence, there is a need to partition this sequence to obtain a coverage measure. This is a common problem for all discovery algorithms for single sequences. To solve this problem, the authors proposed a time window-based coverage interestingness measure. A window (w, t_b, t_e) of size w in S can be formally defined as a sub-sequence of S containing the pairs (A_i, t_i) such that $T_b \leq t_b \leq t_i \leq t_e \leq T_e$. Notice that the size of the set $\mathcal{W}(w, S)$ of all windows in a sequence S is given by $|\mathcal{W}(w, S)| = T_e - T_b + w + 1$. Semantically, a user-defined time window size denotes how close items in a sequence has to be in order to be relevant for the user. It can also be considered as a temporal constraint. The *frequency* of an episode α is then defined w.r.t to the fraction of windows in which the episode occurs:

$$f(\alpha, S, w) = \frac{|(w, t_b, t_e) \in \mathcal{W}(S, w) \text{ such that } \alpha \text{ occurs in } (w, t_b, t_e)|}{|\mathcal{W}(w, S)|}$$

An episode is, then, frequent if its frequency is larger than a user-given minimum frequency *minfr*. Notice that the support and frequency do not have the same semantics meaning. The support reports on the number of occurrences in a given sequence sample (in a sequence database) while the frequency can be interpreted as the probability that a randomly chosen window contains an episode. The following example highlights the difference. Let $A \rightarrow B$ be an episode. Assume that the analyzed sequence contains an occurrence of A, followed by B with a time lag of 2 time units. Say that the user defines a time window of 10 time units. Hence, this occurrence of $A \rightarrow B$ will contribute at least 5 times to the frequency measure: one for each successive window of length 10. On the other hand, it contributes by one unit to the occurrence counting support. Notice also that different frequency

scores as defined by Mannila and Toivonen [101] are obtained for different window sizes: an occurrence of $A \rightarrow B$ contribute with one unit to the frequency with a window size of 3.

Mannila and Toivonen [101] proposed a first algorithm, *WINEPI*, using this frequency measure. It is a *breadth-first candidate generation* algorithm devised to discover the full set of frequent episodes using the Apriori property for pruning. An interesting thing to notice is that WINEPI reduces the problem of mining a single sequence to a sequential pattern mining problem. Indeed, if one builds a sequence database where each sequence corresponds to a time window in a single sequence, finding frequent episodes (in the sense of Mannila and Toivonen [101]) comes to find frequent sequential patterns (in the sense of [4]). One drawback of WINEPI is that it has to scan the entire sequence at each iteration to compute the frequency of episode candidates. This operation is very costly if the number of candidates is large. We refer the reader to the original work for further algorithmic details as it is very similar to former algorithms description.

In the same work [101], the authors also proposed to compute *episode rules* from the set of frequent episodes. An episode rule is an expression $\alpha \Rightarrow \beta$ such that α is a sub-episode of β having a confidence given by:

$$conf(\alpha \Rightarrow \beta) = \frac{f(\beta, S, w)}{f(\alpha, S, w)}$$

The episode rule generation is done in a post-processing phase with the complexity $\Theta(n^2)$ where n is the number of frequent episodes. It consists simply of checking for each pair of frequent episodes, α and β if α is a sub-episode of β and compute its confidence measure. If this confidence is greater than a user-given *minimum confidence* threshold, it is outputted as a result.

In this work, authors also proposed an alternative approach, *MINEPI*, to assess episodes' interestingness with a support using the minimal occurrences concept. Given an episode α and a sequence S, a minimal occurrence of α is a time interval $[t_b, t_e)$ in which α occurs such that there is no proper sub-interval of $[t_b, t_e)$ containing α. The MINEPI algorithm is also an Apriori breadth-first algorithm that stores minimal occurrences of each episode. More precisely, it scans the sequences once to store minimal occurrences of episodes of size 1. After that, pairs of frequent episodes of length k are used to build episodes of length k+1. Minimal occurrences of the (k+1)-candidates are computed using a temporal join of minimal occurrences of the two frequent k-episodes (for serial episodes). This process may remind the reader of vertical database representation for sequential pattern mining. The main drawback of MINEPI is its memory consumption that can make discovering large episodes infeasible. Indeed, the storage of minimal occurrences can even be larger, in terms of memory cost, than the original sequence, especially within the first iterations. Sequential rules derived from episodes given the minimal occurrences approach can, contrary to *WINEPI* results, contain temporal information. For example, if α is a frequent episode with known minimal occurrences $\langle [t_{bi}, t_{ei}) \rangle$ and β a frequent sub-episode of α with minimal occurrences $\langle [t_{bj}, t_{ej}) \rangle$, it is possible to build episode rules of the form:

$$\beta[w_1] \Rightarrow \alpha[w_2]$$

The interpretation of this rule is the following: if β has a minimal occurrence in an interval $[t_b, t_e)$ smaller than w_1 then α occurs at interval $[t_b, t'_e)$ with $t'_e - t_b \leq w_2$ with a confidence *conf*. Computing the confidence of this rule can be done in one pass through minimal occurrences of α and β. For each minimal occurrence of β, $[t_{bj}, t_{ej})$ with $t_{ej} - t_{bj} < win_1$ the algorithm locates the first minimum occurrence of α, $[t_{bi}, t_{ei})$ such that $t_{ej} \leq t_{ei}$ and $t_{ei} - t_{bj} \leq win_2$.

One main limitation of window-based approaches is that they are not suitable to discover large patterns. In-

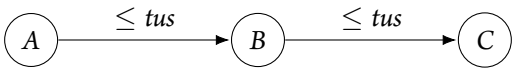


Figure 4.2.7: Episodes obtained using the time unit separation constraint [23]

deed, for both WINEPI and MINEPI, the window size is a temporal bound on the overall episode, which is limiting if an interesting succession of events stands in a single sequence. This can be a limiting aspect for applications such as trajectory discovery. To tackle this issue, Casas-Garriga [23] proposed a new frequency measure using a temporal constraint, called *time unit separation* and noted *tus*, instead of window size. This constraint is the same as the maximum gap constraint for sequential pattern mining. It defines the maximum time lag between successive events (for serial episodes). This way, episodes can be extended with no overall time limit. A graphical representation is depicted in Figure 4.2.7. In this example, *tus* specifies the maximum time lag between A, B and B, C. One can infer straightforwardly that the maximum time lag between A and C is $2 * tus$. Authors proposed an algorithm *EpiBF* that implements a Best-first strategy [19]. It proceeds as follows. First, the algorithm computes the set of all episode candidates of size 2 and the set of potential candidates of size 3. Then, the algorithm scans repeatedly the sequences to compute the support of candidate episodes until this set is empty. In this process, when an episode of size k becomes frequent w.r.t the user-given minimum support, it 'notifies' its super-episodes of size $k+1$ in the set of potential candidates. If a potential candidate of size $k+1$ has two frequent sub-episodes of size k (obtained by dropping either the first item or the last item), it is added to the set of candidates, and its support begins to be counted. A candidate is not considered any more if the continuous sequence scan comes to the point where the counting started for this episode.

Another extension of episode mining was proposed by Laxman et al. [89] that proposed another frequency measure based on non-overlapping episode occurrences.

Definition 19 (Non-overlapping episode occurrences [89])
Two occurrences of an episode are said to be non-overlapping if no event associated with one appear in between the events associated with the other. The frequency of an episode is defined as the maximum number of non-overlapping episode occurrences in the event sequence.

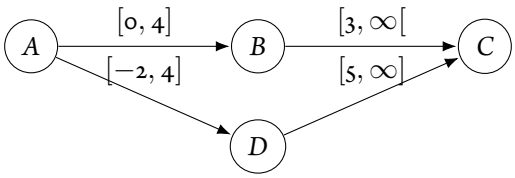
Given this definition, one can remark that non-overlapping frequent episodes are a subset of frequent episodes as defined by Mannila and Toivonen [101] or Casas-Garriga [23]. One primary motivation behind this definition is that an event must belong to at most one episode occurrence and that episode occurrences cannot be overlapping. Indeed, authors in that work proposed to formally use episode and episode rules to build a particular class of Hidden Markov Chains (HMM) called *Episode Generating HMMs*. Authors claimed in their work that HMMs generated from the most frequent episodes are much likely to generate a given sequence. The algorithm proposed by the authors implements an automata-based counting approach. While scanning an event sequence, the algorithm maintains an automaton per episode candidate whose states corresponds to episode nodes. Whenever an event A is found in the sequence, all automatons waiting for event A are incremented (passes to the next state). When the final state is reached, the episode count is incremented by a unit, and the corresponding automaton is reinitialized.

Episode mining can also support conciseness measures. Harms et al. [77] proposed *Gen-FCE* and *Gen-REAR* to discover respectively frequent closed episodes and generate rules (called *Representative Association Rule*) from the set of frequent closed episodes. The authors used a Formal Concept Analysis approach associated with a sequence database built w.r.t a window size. This approach has similarities with *WINEPI*: each time window in

the event sequence is considered as a database entry.

Harms and Deogun [76] proposed to extend episode rules as given by MINEPI with a time lag between the the premise and the conclusion sequences. These time lagged episodes are noted $\alpha[w_1] \Rightarrow_{lag} \beta[w_2]$ where α and β are constrained episodes (i.e α occurs within a time window w_1) and *lag* is the delay between the occurrence of the premise α and the conclusion β. It is to notice that *lag* can be either a fixed time lag or a maximum time lag constraint. The proposed algorithm, *MOWCATL*, is mostly inspired by the *MINEPI* algorithm and uses minimal occurrences storage. The rule generation phase is done via the precedent algorithm *Gen-REAR*. To the best of our knowledge, this contribution is the first to propose a, somewhat, quantitative episode rule. Indeed, it includes, in the case of the fixed time lag constraint, a time delay information. However, this approach may be non-efficient to extract strong quantitative relationships for two main reasons. First, Premises and conclusions in episode rules are only temporally constrained by a time window: it does not provide any information about time lags between intra-episodes events. Second, considering the exact time lag is somewhat limiting as temporal phenomena often occur with slight temporal variations. As a consequence, this approach may miss interesting temporal relations. This aspect is captured using the maximal time lag constraints but is general and comes with significant loss of temporal information (is there one or various *typical* time delays? what is the time delays distribution?).

One of the first contributions dealing with mining quantitative patterns from a single point-based sequence was proposed by Dousson and Duong [34]. In that work, authors used the chronicles formalism [35] that is defined as a pair (S, T) where S is a set of labels and T is a constraint graph on instances of S where nodes corresponds to events in S and edges correspond to temporal constraints taking the form of an interval $[l_{min}, l_{max}]$ where $l_{min}, l_{max} \in \mathbb{Z}$ and $l_{min} < l_{max}$. Follows a graphical example:



It is to notice that this pattern model is very similar to *Delta Patterns* that we discussed earlier. The main difference relies on endpoints of temporal constraints that have to be positive for *Delta Patterns* while *Chronicles* support negative and infinite bounds. As a consequence, *Delta Patterns* are limited to sequential pattern while *Chronicles* are not. Chronicles can be seen as episodes (containing both serial and parallel relations) that are extended with time lag information. In that preliminary work, the authors proposed the FACE algorithm (breadth-first candidate generation). However, this algorithm is not complete. As for a set of event types, the algorithm will discover a unique chronicle (the most general one) while several can exist. Several algorithms using the chronicle formalism have also been proposed among which *HCDA* [31] that is devised to obtain the complete set of frequent chronicles or [151] [32] for the discovery of discriminant chronicles, i.e. frequent patterns are used to discriminate between two sequences groups: the chronicle appears in the positive group and do not appear in the negative one. (Chronicles was also used for sequential databases mining [14] [83] [138])

Ma and Hellerstein [100] proposed another form of quantitative patterns. The authors designed an algorithm to discover partially periodic associations, called *p-patterns*. A *p-pattern* describes a set of symbols, noted *IS*, occurring frequently (w.r.t a minimum support) and periodically with a period p and a time tolerance δ. The support of a *p-pattern* is computed w.r.t to successive occurrences, called inter-arrivals. The authors proposed to discover significant periods based on a Chi-squared statistical test of independence. This test compares the support of an inter-arrival time with the number that would be expected from a random sequence of inter-arrivals.

Two algorithms were proposed to discover *p-patterns* based on two main steps: (1) find temporal associations, and (2) find statistically significant periods. Temporal associations are found thanks to an Apriori candidate generation algorithm that is largely similar to the window-based episode mining approach. Significant periods are discovered as follows. For a given itemset the sequences are scanned to count occurrences of all inter-arrival values (this operation is done in $\Theta(n)$ with n number of events in the sequence). The support of each inter-arrival time is adjusted w.r.t δ to group similar time lags. Then, the statistical threshold *thresh* is computed and compared to the empirical support *sup*. If $sup \geq thresh$, then the *p-pattern* is considered as statistically significant. The authors proposed two main approaches using the former algorithm steps. The first, called *Period-First*, finds significant periods for each item and then computes temporal associations (*p-patterns* of itemsets). The second algorithm, *Association-first*, uses the inverse strategy. The authors argue that *Period-First* is more efficient as it can use early pruning on time periods. On the other hand, *Association-first* is more robust to sequence noise. Indeed, one drawback of using inter-arrival-based support is that any noise item placed in between the occurrence of a periodic pattern generates two 'false' inter-arrival times and avoid the one 'true' occurrence. As a consequence, mining association with a window-based algorithm avoid early period-based pruning.

Li and Ma [93] extended the former work to discover pairwise event dependencies with statistically significant time lags. They proposed a two-stage approach. The first one consists of a statistical preprocessing stage that eliminates unpromising time lags to reduce the search space. For a relation A before B and a time lag r it estimates the time lag distribution of B from a randomly chosen time points in the sequence. It is compared to an estimated conditional distribution of time lags between events A and A. If A and B are not temporally correlated, the former two distribution must be equal. If so, time lag r is pruned and considered as non-promising. The second stage consists of identifying statistically valid time lags using the approach introduced in [100] for periodic patterns (with the use of the chi-squared test). This approach is also based on inter-arrivals that can be inefficient with noisy data or with overlapping occurrences.

This former problem was addressed by Tang et al. [145] who designed *STScan* for the discovery of pairwise dependencies between events where quantitative information is given as a range. This algorithm is based, as for the work of [100] and [95], on the support statistical assessment. More precisely, a dependency $A \rightarrow_r B$ where $r = [t_{min}, t_{max}]$ is the time lag range, is statistically significant if the number of the successions AB with a time lag within r, noted n_r, exceeds the minimum required number according to the χ^2 test of independence. This test measures the degree of independence comparing the observed n_r and the expected one under the independence assumption. A time lag is said to be *qualified* (or significant) for a temporal relation, if its the support exceeds a minimum support threshold *minsup* and its χ^2 statistics is higher than a minimum threshold χ^2_c. As the authors aim is to discover time-lagged dependencies without a direct succession constraint (inter-arrival), a brute force algorithm must test every possible time lag range r which is not affordable for large sequences To improve the discovery process, STScan utilizes a sorted table, built in one sequence pass, that maintains all possible time lags and their corresponding occurrences. STScan also reduces the search space considering a higher bound on the size of r that is computed thanks to the minimal threshold χ^2_c. This minimal interval length can be seen as a temporal constraint.

More recently, contributions as [156, 170, 171] tackled the problem of discovering pairwise dependencies with multiple and fluctuating time lags. Indeed, in many real cases, (1) multiple time lags between two events types can exist and (2) the typical time lag of a phenomenon can fluctuate due to multiple factors (e.g., inherent phenomena temporal variability, missing values, acquisition noise). In these works, the time lag between two events is modelled as a normal distribution $\mathcal{N}(\mu, \sigma^2)$ where μ is the actual time lag and σ the variance representing

the fluctuations. These contributions utilize an expectation maximization-based (EM) approach to find time lags with maximum likelihood. Notice than with this framework does not provide any interestingness criteria that may indicate the strength of a temporal dependency.

4.3 Interval-based patterns

Time intervals report on temporal events/attributes lasting in time. Temporal pattern mining from interval-based data aims to discover temporal relations between several interval types/labels from interval-based temporal data.

One main difficulty posed by the use of intervals is the induced pattern language. Indeed, contrary to point-based data supporting simple relations (*before, after, co-occurrence*), 13 possible relations can stand between a pair of intervals. As a consequence, temporal relations between multiple intervals are more challenging to model efficiently with enough expressive power. We refer the reader to Höppner and Peter [82] work for a detailed interval-based expressiveness study. Besides, regardless of exploration strategies, using Allen' relations makes the search space much larger than for point-based sequential pattern mining [11]. Instead of maintaining one relation support (before), interval-based approaches must count occurrences of 7 possible temporal relations.

In the following section, we will consider several expressivity criteria, that can be encountered in the literature (e.g. [82] [78]) to assess the quality of pattern languages.

- **Completeness**. A pattern language is said to be complete if it permits to infer the 13 possible temporal relations between intervals. Notice also that the degree of completeness defines the search space size for a pattern mining problem.

- **Ambiguity**. A pattern language is qualified as ambiguous if a pattern expressed in that language can support different interpretations. This criterion may be referred to also as **Concurrency**. We distinguish two forms of ambiguity. The first is the qualitative ambiguity referring to cases where a pattern can support several qualitative interpretations. The second is the quantitative ambiguity referring to patterns that can support different interpretation in terms of time gaps and interval duration.

- **Inelasticity**. One main drawback of Allen' qualitative relations is that they are susceptible to small temporal variations. For example, a *during* relationship can be transformed to *begins with* with only a variation of 1 time unit. Inelasticity can be a serious limitation for, say, a frequent pattern mining approach as very similar successions of labelled intervals (describing the same temporal phenomenon) can lead to different patterns that are counted separately. As a consequence, a temporal phenomenon may be missed by the algorithm if neither of its temporal variations meets to minimum support threshold. This criterion is also called **Robustness**

- **Pattern size**. We evaluate the size of a pattern involving k intervals as the minimum number of *nodes* and *edges* that would be necessary to represent a pattern as a directed graph. In these graphs, a node denotes a symbol (intervals label), and an edge denotes the temporal relations between them.

In the following section, we will describe the main interval-based patterns languages and their associated mining approaches. As for point-based temporal pattern mining, we will principally categorize approaches w.r.t their input data format (i.e. sequence databases or single sequence). In addition to this, we will distinguish between quantitative and qualitative pattern models and discuss their expressive power.

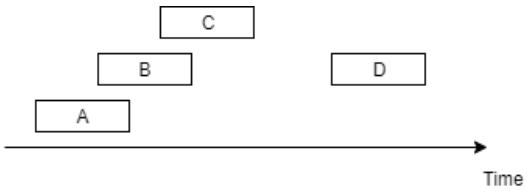


Figure 4.3.1: A sequence of intervals

We begin with a rapid common formal definition of interval sequences that will be used for the rest of this section. Let $I = \{i_1, i_2, i_3, ..., i_n\}$ be a set of symbols denoting interval types. An interval E_k is pair $E_k = (i_k, [t_{b_k}, t_{e_k}))$ where $i_k \in I$ is the label of E and $[t_{b_k}, t_{e_k})$ its interval temporal anchor. In some works, intervals are considered to be closed $[t_b, t_e]$. However, we will use the right-opened intervals as an unified definition (for reasons described in Section 2.1) as these differences do not impact the comprehension and comparison between different pattern models. For the sake of readability, we will also denote $E_k.t_s$ as the first endpoint of E_k and $E_k.t_e$ as its second endpoint. An interval sequences is an ordered list of intervals $S_{int} = \langle E_1, E_2, E_3, ..., E_n \rangle$. The ordering of intervals may differ from a contribution to another. Some works considers total order of intervals using starting points i.e $\forall k, t_{b_k} \leq t_{b_{k+1}}$, while other considers the total order of ending points i.e $\forall k, t_{e_k} \leq t_{e_{k+1}}$

4.3.1 Patterns from sequences databases

Qualitative Patterns

Most of the existing temporal pattern mining from interval-based data are interested in processing sequences databases. This problem is very similar to the one posed earlier for point-based temporal data. These approaches have as input a database of interval-based sequences $\mathcal{D} = \{(id_1, S_1), (id_2, S_2), ..., (id_n, S_n)\}$ where $\forall k$, S_k is an ordered sequence of intervals and id_k its identifier. The task of mining sequential patterns in interval-based sequences databases aims to discover patterns (defined w.r.t a pattern language) frequently occurring in a sequence database.

To the best of our knowledge, Kam and Fu [85] was the first contribution to be interested to this problem. They proposed a pattern language that the temporal arrangement of intervals as a composition of pairwise seed relationships $(X\, rel\, Y)$ where $X, Y \in I$ and rel is an Allen' relationship. For example, one can describe the sequence depicted in Figure 4.3.1 with the following formalism: "(A overlaps B) overlaps (C before D)" or "A overlaps (B overlaps (C before D)". As the number of such representations is exponential with respect to the number of intervals, the authors proposed to discover a sub-set of these representation, called *A1-pattern*, taking the form $((...(A_1\, rel_1\, A_2)\, rel_2\, A_3)...\, rel_{n-1}\, A_k)$. For example, the sequence depicted in Figure 4.3.1 corresponds to the following *A1-pattern*:

(((A overlaps B) overlaps C) before D)

This representation is rather useful to mine long patterns as it permits to explore the search space in a level-wise manner. An *A1-pattern*, say $X\, rel\, Y$ can be considered as a virtual interval with endpoints $[t_b, t_e)$ such that $t_b = min(X.t_s, Y.t_s)$ and $t_e = Y.t_e$. As a consequence, the Allen' relation between an *A1-pattern* and an interval is computed similarly to the the relationship between two intervals. It is to notice that only 7 Allen' relations are used: reverse relations do not provide any additional information considering the proposed framework. Authors proposed an Apriori-like candidate generation algorithm. The candidate generation is basically the same

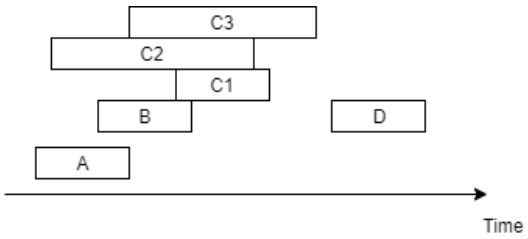


Figure 4.3.2: Ambiguity of A1-patterns. C1, C2 and C3 are cases of instances of interval C leading to (((A overlaps B) overlaps C) before D)

as for sequential pattern mining and the Apriori property is used for unfrequent candidates pruning. The main difference is that the algorithm evaluate which of the Allen' relationships holds between an *A1-pattern* and the extension candidate on the basis of intervals endpoints. This approach permit to emphasize the fact that interval-based sequential mining is a more difficult task than its point-based version: a candidate pattern composed by a *A1-pattern* of length k and a interval may generate, in the worst case, 7 frequent *A1-patterns* of length $k+1$. In that same work, authors proposed also a second pattern model, called *A2-pattern*, and defined recursively as follows: (1) a temporal pattern of size 2 is an *A2-pattern* (2) if X is an *A2-pattern* and Y is a temporal pattern of size 2, then X *ref* Y is an *A2-pattern*. As a consequence, *A2-patterns* are always composed by an even number of intervals. An *A2-pattern* corresponding to sequence in Figure 4.3.1 is:

(A overlaps B) overlaps (C before D)

A2-patterns are discovered with a similar approach as for *A1-patterns* with the difference that candidate generation uses seeds patterns of size 2 to extend frequent patterns of size $2k$. One can notice that *A2-patterns* are a subset of *A1-patterns*.

The pattern models proposed by Kam and Fu [85] can be considered as **complete** as they use the entire set of Allen' relations. They are also **ambiguous**. An *A1-patterns* or *A2-patterns* between k intervals needs the specification of $k-1$ relations. We show in Figure 4.3.2 different interpretations of (((A overlaps B) overlaps C) before D). The ambiguity problem may lead to a misinterpretation of the resulting patterns but can also impact the mining process as different arrangement of intervals can be counted together for an extension leading to potential support overestimation. Later, ito et al. [84], Patel [119] extended this pattern model to reduce the ambiguity problem. They proposed to augmente with an information about relations count to differentiate between different interval arrangement. For instance, in Figure 4.3.2, $C1$ has 1 *overlap* (with B) and 0 *meet*, $C2$ has 2 overlaps (with A and B) and 0 *meet* and $C3$ has 1 *overlap* (with B) and 1 meet. More precisely, they integrate the count of 5 Allen relations to the *A1-pattern* model: contain (c), finish by (f), meet (m), overlap (o), start (s). Counts are stored in an ordered vector $[c, f, m, o, s]$ of length 1 where the first value contains the count for *contain*, the second for *finish by* and so on. The former examples can be then represented as follows:

$$((A \text{ overlaps } [0,0,0,1,0]\ B) \text{ overlaps } [0,0,0,1,0]\ C1)$$
$$((A \text{ overlaps } [0,0,0,1,0]\ B) \text{ overlaps } [0,0,0,2,0]\ C2)$$
$$((A \text{ overlaps } [0,0,0,1,0]\ B) \text{ overlaps } [0,0,1,1,0]\ C3)$$

To discover this kind of patterns, authors proposed *IEMiner* [119]. It uses a breadth-first strategy with an optimized candidate generation for interval mining. For each enumeration level k, the algorithm extracts the set

of frequent 2-patterns from frequent k-patterns. Candidates of length $k+1$ are generated combining two frequent k-patterns having the same prefix. Frequent 2-patterns are used to prune non-frequent candidates before the support checking that is performed in one database scan.

The ambiguity problem was also addressed by Wu and Chen [160] who proposed to consider interval sequences as a sequence of point-based endpoints. An interval $E = (i, [t_b, t_e))$ is represented as two endpoints $e^+ = (i^+, t_b)$ and $e^- = (i^-, t_e)$ where i^+ and i^- respectively denotes beginning and ending of event or state i. As a consequence, interval sequences, and by extension interval-based patterns, can be represented as a series of endpoints with relations *before/after* (<) or *co-occurrence* (=). An example of such representation corresponding to sequence in Figure 4.3.1:

$$(A^+, t_1)(B^+, t_2)(A^-, t_3)(C^+, t_4)(B^-, t_5)(C^-, t_6)(D^+, t_7)(D^-, t_8)$$

with $\forall t_i, t_i \leq t_{i+1}$. If two intervals of the same type occur in the same sequence, an additional index is appended to interval endpoints. For instance, if two intervals A occur in a sequence, the endpoint representation will contain endpoints A_1^+, A_1^-, A_2^+ and A_2^-. This representation permits, somehow, to turn the problem of interval-based patterns to a classical point-based sequential mining problem and permits to utilize existing approaches. The authors proposed the *TPrefixSpan* algorithm that extends *PrefixSpan* to handle interval sequences. The main difference stands in the candidate generation process where intervals, rather than endpoints, are used to extend patterns and ensure non-ambiguity. This representation can also reduce the inelasticity problem partially. Indeed, a small variation in intervals endpoints is less likely to change the relations between the two endpoints. Chen et al. [27] also used this pattern model for the *CEMiner* algorithm that is devised to discover closed patterns from endpoint sequences. It uses a depth-first exploration strategy extending the BIDE [154] algorithm.

The endpoint representation approach was extended by Mörchen and Fradkin [106] who proposed the *Semi-Interval Sequential Pattern* (SISP). As its name may suggest, authors proposed to mine interval-based pattern based on the semi-interval formalism. The main difference between this work compared with the contribution of Wu and Chen [160] is that a SISP is designed to express temporal relations between intervals, semi-interval (i.e. intervals endpoints) and time point events. Authors utilize a closed sequential pattern mining algorithm (e.g. BIDE) instead of mining the entire set of frequent sequential patterns. Follows some example of SISPs that can be extracted from the sequence in Figure 4.3.1

$$\begin{aligned} p_1 &= A^+ B^+ A^- B^+ \\ p_2 &= B^+ C^- \\ p_3 &= C^- D^+ D^- \end{aligned}$$

p_1 is a pattern between intervals, p_2 and pattern between two semi-intervals and p_3 a pattern between an interval and a semi-interval. However, using such generalization of the interval-based sequential pattern comes with ambiguity. As endpoints are considered independently, SISP loses somewhat the inherent link between intervals endpoints (contrary to in [160]). Figure 4.3.3 shows two interpretations of a pattern $A^+B^+C^+B^-C^-$. Authors also proposed the *Semi-Interval Partial Order pattern* (SIPO) model, that is a relaxed version of SISP in that do not require the definition of all pairwise temporal relations between endpoints. Indeed, SIPOs define a partial order between endpoints and can be defined as a directed acyclic graph. As a consequence, they may better handle the inelasticity problem. SIPOs can be considered as ambiguous (but one can argue that they are designed to be so). It is to notice the SISPs or SIPOs can express a larger set of patterns than Allen' relation-based patterns.

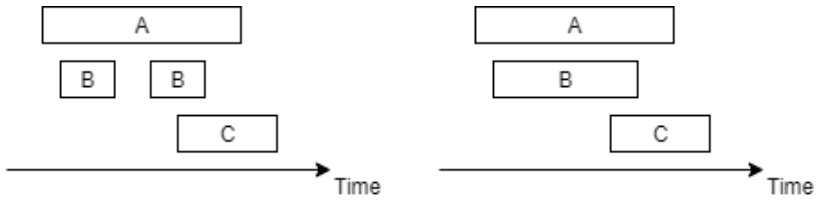


Figure 4.3.3: Ambiguity of the SISP pattern model. These two sequences support pattern $A^{+}B^{+}C^{+}B^{-}C^{-}$

	A	B	C	D
A	=	o	b	b
B		=	o	b
C			=	b
D				=

Figure 4.3.4: Relationship matrix corresponding to sequence in Figure 4.3.7. o: overlaps, b: before. The relationship between B and A is the inverse Allen relationship of the relation between A and B.

Similar to SISP, Wu and Chen [161] proposed the *Hybrid Temporal Pattern* (HTP) supporting both interval-based and point-based data. As in the previous works, the authors proposed to transform interval sequences into a sequence of endpoints. The main difference is that multiple occurrences of the same interval types are distinguished using additional annotations. An interval of type A is denoted $\langle A^{1+}A^{1-}\rangle$ in the interval sequence permitting to distinguish endpoints belonging to different intervals, e.g. (A^{1+} and A^{2-}). As a consequence, the ambiguity problem depicted in Figure 4.3.3 is resolved.

Winarko and Roddick [159] proposed the ARMADA algorithm that is devised to mine non-ambiguous patterns. They adopted the Relationship Matrix (RM) pattern model that was introduced by Höppner and Klawonn [81] in their work dealing with single sequences. RM is known to be the first **non-ambiguous** interval based pattern model. Its main idea is to specify the qualitative relationship between all pairs of intervals in a pattern. Thus, the size of this representation for a pattern containing k intervals needs $\frac{n(n-1)}{2}$ relation specifications. An example of a relationship matrix corresponding to the interval sequence of Figure 4.3.1 is depicted in Figure 4.3.4. The algorithmic aspect of the work of Höppner and Klawonn [81] will be treated in the Section 4.3.2 as it is out of the scope of the current section. The ARMADA algorithm is an extension of MEMISP [95] descried in Section 4.2.1. ARMADA uses an extension of the MEMISP index that contains intervals instances that are necessary to compute Allen' relations while extending patterns. ARMADA uses also the same partitioning approach to handle very large databases. In that work, authors also proposed to compute sequential rules from the set of all frequent patterns. As for episodes [81], rules are depicted as $X \Rightarrow Y$ where X is a sub-pattern of Y. For each frequent pattern $Y = \langle s_1, s_2, s_3, ..., s_n\rangle$, there is at most $n-1$ sub-pattern X of Y such that $X = \langle s_1, ..., s_i\rangle$ with $i \in [1, n-1]$. For each sub-pattern X the confidence of the rule $X \Rightarrow Y$ is computed and considered as a result if $conf \geq minconf$. The algorithm processes sub-patterns X from $i = n-1$ to $i = 1$ in order to perform pruning: if a rule $X_i \Rightarrow Y$ have a confidence lower than *minconf* then rules $X_j \Rightarrow Y$ with $j < i$ are not tested as they will have a lower confidence. ARMADA also include a maximum gap constraint. The gap between two intervals $[t_{b_i}, t_{e_i})$ and $[t_{b_j}, t_{e_j})$ with $t_{b_i} \leq t_{b_j}$ is defined with $t_{b_j} - t_{e_i}$. Notice that the gap between two intervals can be negative (e.g an overlap relationship). If the user-given gap constraints is a positive integer, it affects only intervals with a *before* relationship.

Another contribution using a similar pattern representation was proposed by Papapetrou et al. [117]. In that

work, the authors proposed to use a more flexible version of Allen' relations to reduce the inelasticity problem. It consists of the use of an ε parameter to relax the temporal relations between endpoints (this approach was also used in [107] that we discuss later).

Definition 20 (ε-temporal relations [117] [107])
Let t_1 and t_2 be two time points and $\varepsilon \in \mathbb{N}$ a relaxation threshold. The temporal relations between t_1 and t_2 considering ε are defined as follows:

- t_1 co-occur with t_2, noted $t_1 =^{\varepsilon} t_2$ if $|t_1 - t_2| \leq \varepsilon$
- t_1 is before t_2, noted $t_1 \leq^{\varepsilon} t_2$ if $t_2 - t_1 > \varepsilon$.

We can argue that the use of the ε parameter is equivalent to consider a greater time granularity for the mining process. That idea permits to reduce the inelasticity problem but comes with a loss of qualitative information depending on the choice of ε, which needs domain knowledge. Notice also that a large ε parameter may reduce time intervals to time points if $|t_b - t_e| \leq \varepsilon$ which can be considered as information loss. Papapetrou et al. [117] proposed *H-DFS* that explores an enumeration tree where each node is associated with the set of possible arrangements (or RM). For example, a node $\{A, B\}$ will contain all seven possible Allen relation between A and B. HDF-S is inspired by the SPAM algorithm of Ayres et al. [17]. It uses both depth-first and breadth-first strategies are used. It also uses the IDList vertical representation approach (cf. Figure 4.2.2). More precisely, HDF-S uses a depth-first strategy for the first two levels of the enumeration tree to obtain frequent 2-patterns and their respective IDLists. Then, H-DFS performs a depth-first strategy. It uses the Apriori property to avoid testing every sub-pattern of a found frequent one. Maintaining IDLists of frequent 2-patterns (from the early breadth-first exploration of levels 1 and 2 of the enumeration tree) permits to extend every k-pattern to a $(k+1)$-pattern without scanning the entire database to find pairwise relations. As the RM representation is a conjunction of pairwise relations, the support computation and IDList build for a k-pattern can be done with temporal joins between IDList of the (k-1)-pattern to be extended and IDList of frequent 2-patterns. This way, H-DFS benefits from the advantages of both depth-first and breadth-first strategies. Authors also proposed to integrate temporal constraints (maximum gap, interval duration) and structural constraints (overlapping and containment percentage) into the mining process. One main drawback of the former approach is the RM candidates generation that considers all seven possible Allen' relations, which slows down the discovery process.

This problem was addressed by Moskovitch and Shahar [107, 110] that proposed the *KarmaLego* algorithm. It also uses the flexible Allen' relations based on the ε parameter with the maximum gap constraints. KarmaLego algorithm is composed of two steps *Karma* and *Lego*. Karma scans each sequence in the database to count the occurrences of each existing pairwise interval relations (w.r.t ε) and keeps as results those meeting the coverage criteria ($support \geq minsupport$). This approach is mainly the same as ARMADA as it uses an index of all existing patterns. Frequent pairwise patterns found by *Karma* are extended recursively by *Lego* using a *depth-first* strategy. The authors proposed to speed up this exploration process using temporal transitivity to reduce the number of generated extension candidates. Indeed, for each k-pattern extension, a naive approach may generate seven extension candidates (corresponding to Allen' relations) for each of the k intervals. That leads to generate up to 7^k possible candidates to be tested. However, the set of all possible extensions can include several contradictory relations. For example, in Figure 4.3.1, we suppose that Relationship Matrix for pattern $\langle A, B, C\rangle$ is known. If the relation between C and D is known (*before*), one can infer the relation of D with A and B using their relations with C. For instance, A can not *overlap* D as A is *before* C and C is before D. Transitivity permits to infer this relation straightforwardly: A before D. This notion was formalized as a transition table by Allen [10] and later by Freksa

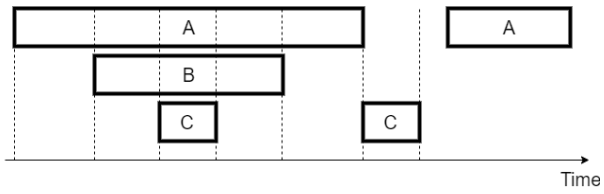


Figure 4.3.5: An interval sequence

[52]. Using transition tables, KarmaLego generates candidates recursively. More precisely, it generates possible candidate relations between the last item of the *to-be-extended* pattern and frequent single intervals. At this stage, the process is similar to the naive candidate generation. After that, for each possible candidate relation, the algorithm computes recursively possible candidates w.r.t the transition table that returns only temporally consistent candidates rather than the 7 Allen relations. This process permits to reduce the number of candidates for each extension considerably potentially. The authors also proposed to perform an interval pre-clustering w.r.t intervals duration using an algorithm as *k-means* to append duration information to intervals labels. This process leads to an increase in intervals symbols (each duration cluster is considered as a symbol) and by extension increases the search space. Authors claimed that KarmaLego outperforms *ARMADA*, *H-DFS* and *IEMiner*. Later, Moskovitch and Shahar [108, 109] used the KarmaLego interval-based pattern algorithm to perform sequence classification.

Chen et al. [28] proposed another pattern model, called *Coincidence Representation* (CR). A CR can be considered as an extension of both TSKR and SISP supporting only total order. The TSKR model uses the coincidence concept, i.e two interval coincides if they overlap in time, to transform multiple interval sequences into a sequence of coincidences, i.e itemset denoting co-occurring intervals. As this approach was proposed for single sequences, we provide further details in Section 4.3.2. As TSKR, CR utilizes the coincidence relationship (*chords* in TSKR) to model simultaneity. It uses interval labelling with sub-intervals, called slices, rather than endpoints in SISP. More precisely, given an interval $A = [t_b, t_e)$, three slices can be encountered:

- A^+ is the first sub-interval of A. $A^+ = [t_b, t_{e^+})$ with $t_b < t_{e^+} < t_e$
- A^- is the last sub-interval of A. $A^+ = [t_{b^-}, t_e)$ with $t_b < t_{b^-} < t_e$
- A^* is neither the starting nor the ending sub-interval of A. $A^* = [t_{b^\#}, t_{e^*})$ with $t_b < t_{b^*} < t_{e^*} < t_e$

In addition to interval labelling, the CR model includes an implicit specification of the meet relation noted @ in order to distinguish joint and disjoint intervals. The coincidence representation corresponding to interval sequence in Figure 4.3.5 is the following:

$$\langle (A^+)(A^*B^+)(A^*B^*C)(A^*B^-)@CA \rangle$$

Coincidence Representation is non-ambiguous and complete. In that work, authors proposed CTMiner to mine the set of all coincidence representations frequent patterns. It is mostly based on PrefixSpan using projected databases with a depth-first strategy. The same team extends this approach with CCMiner [29] to mine closed CR frequent patterns.

Quantitative Patterns

While discovering qualitative patterns in interval-based data had received much interest, very few relatively recent approaches were designed to discover quantitative relations in interval-based data.

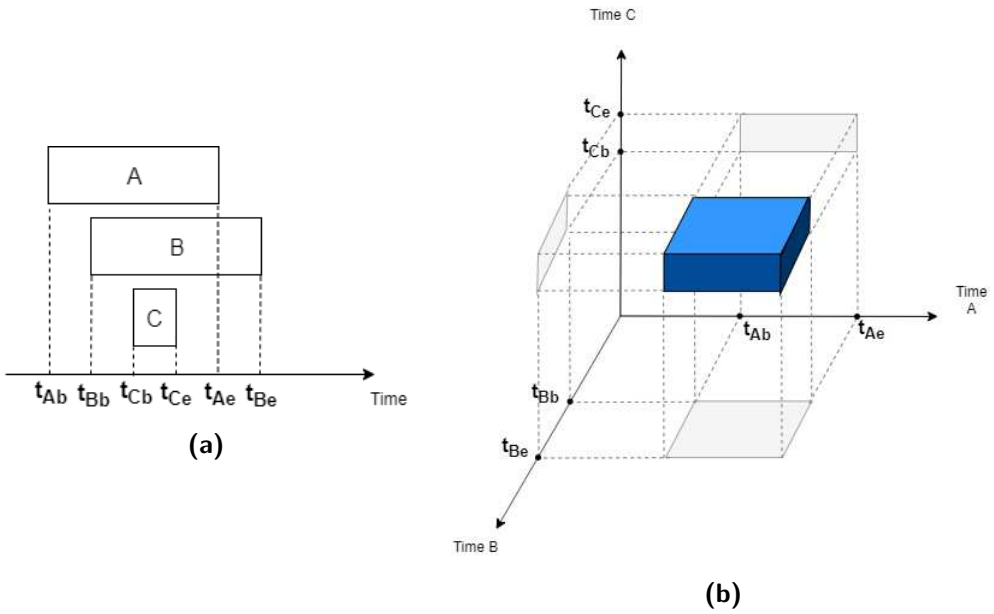


Figure 4.3.6: A 3 intervals sequence (a) and its hyper-cube representation in $\mathbb{R}^3$ (b)

To the best of our knowledge, Guyet and Quiniou [72] was the first to address the problem of mining frequent quantitative patterns from interval sequences databases. Their approach is based on a hypercube representation of sequences in the database. More precisely, an interval sequence of n intervals can be represented as a geometric volume in $\mathbb{R}^n$ whose orthogonal projection over each axis in $\mathbb{R}^n$ corresponds to the endpoint of an interval. Figure 4.3.6 provides an illustration for a sequence with 3 intervals. Using this representation, it is then possible to obtain representative temporal bounds of a given frequent pattern using a density-based multi-dimensional clustering as EM [33]. More precisely, if a set of intervals frequently appears in the database, their occurrences hyper-cube representations are used to run the clustering algorithm to find "dense" representative and frequent hyper-cubes that are considered to be the quantitative patterns taking the form of a set of labelled intervals $\{(i_k, [t_{b_k}, t_{e_k}))\}$. This process is integrated into an Apriori candidate generation algorithm called *QTempIntMiner*. Later, in [73] extended this approach with *QTIPrefixSpan* that used a depth-first strategy based on *PrefixSpan* and used another multi-dimensional clustering approach (K-Means) to speed up the exploration process. The main drawback of this approach is that it is endpoint sensitive: the same pattern occurring at $t = 0$ and $t = \Delta$ may not overlap using the hypercube representation. This fact leads to underestimating of some useful temporal relations potentially.

Another approach based on endpoint sequences was given by Nakagaito et al. [112]. In that work, the authors proposed two main strategies to model quantitative interval-based patterns. The first consists of considering a single endpoint representative per interval to define the total order within the pattern. It can be whether the first, the last or the middle endpoint of intervals. An interval-based quantitative pattern is denoted:

$$\left\langle i_1 \overset{(d_1,d_1'),(t_2,t_2')}{>} i_2 \ \ldots \overset{(d_2,d_2'),(t_n,t_n')}{>} i_n \right\rangle$$

where i_k denotes an interval symbol, (d_k, d_k') a duration in the range $[d_k, d_k']$ of interval of type i_k and (t_k, t_k') a time lag in the range $[t_k, t_k']$ between intervals of types i_k and i_{k+1}. The algorithm, *QPrefixSpan* is a direct extension of *PrefixSpan*. Within the depth-first search space exploration, for each discovered representative endpoint frequent pattern, the algorithm performs a numerical clustering to find and append quantitative duration and time lag

annotations. The main drawback of this approach is its ambiguity that comes from the fact of considering a single endpoint per interval. This issue was tackled in the same work by considering the endpoint representation that we have discussed earlier. It leads to patterns of the form:

$$\langle {}^{(d_1,d'_1)}_{ep_1} \; {}^{(t_2,t'_2)}_{>} \; {}^{(d_2,d'_2)}_{ep_2} \; \dots \; {}^{(t_n,t'_n)}_{>} \; {}^{(d_n,d'_n)}_{ep_n} \rangle$$

where ${}^{(d_k,d'_k)}_{ep_k}$ is an endpoint of an interval whose duration is in $[d_k, d'_k]$. The proposed algorithm *QTPrefixSpan* follows the same approach than *QPrefixSpan* but with endpoints.

Another class of quantitative approaches utilizes Reich' plots as a geometric interval space. It consists of representing intervals in a 2-dimensional space: one for each temporal attribute: endpoints and/or duration. This 2D space can be whether formed by axes *first endpoint & second endpoint* or *first endpoint & duration*.

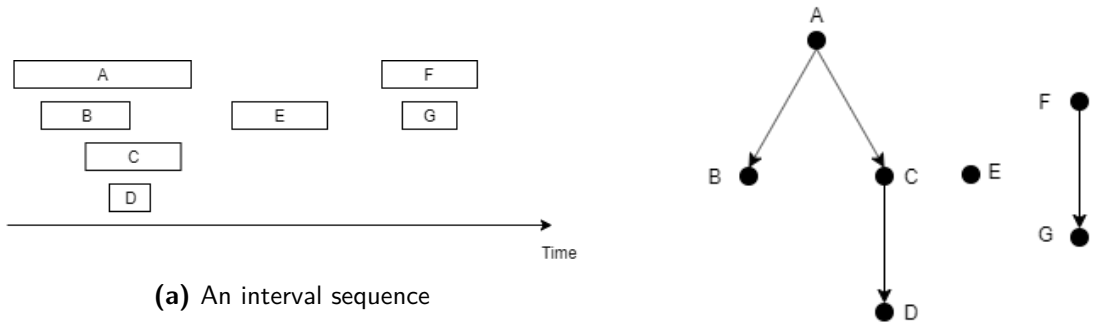
This approach was first used by Ruan et al. [132], who proposed *PESMiner*. The algorithm first plots intervals contained in the sequence database on a 2D-space formed by axes *first endpoint* and *duration*. This representation is used by the user to choose an appropriate clustering technique (setting as well their parameters) or to define, based on domain knowledge, centroids of intervals "groups" of the same type. The second step of the algorithm consists of running the clustering technique to define a set of clusters' centroids that will be used as seed pattern (1-frequent patterns) for the sequential pattern mining procedure. The user can adjust the clustering results based on his domain knowledge. The third step is the execution of an Apriori-like candidate generation algorithm to find frequent arrangements of patterns (or prototypes as called in the paper). The pattern matching (i.e. either a sequence supports a pattern candidate) is calculated in a fuzzy fashion based on the intersection duration of intervals of the same type, even if they do not appear in the same order to avoid the inelasticity problem. Redundant patterns are removed in the last step of *PESMiner*. This approach is also endpoint sensitive as it assumes that patterns are temporally aligned in sequences in the database.

The endpoint sensitivity is solved by *PIVOTMiner* that was proposed by Hassani et al. [78]. In this work, authors focused on discovering quantitative information between pairwise interval types. Their pattern model, called Base Pattern, consists of a tuple $\{O, T, \Delta, S\}$ where O is an origin label, T a target label, Δ the average relative timing $(s_T - s_O, e_T - e_O)$ between endpoints of O and T, and finally, S the support. *PIVOTMiner* uses also the Reich' plot but with different space axes *first endpoint & second endpoint*. As for *PESMiner*, intervals in the database are plotted in the 2D space. In order to mine base pattern involving A and B, the algorithm performs an origin transformation consisting of translating all (geometrical) vectors having A as an origin and B as a target such that vectors origins coincide with the space origin. A geometrical clustering technique (DBSCAN in the original work) is performed on instances of B to obtain the relative timing between A and B consisting of the coordinates of the clusters' centroids. This process is executed for each pair (O, T) once to extract base patterns. This approach is endpoint-sensitive as temporal information is based on the relative time lags between intervals rather than their exact ordering in a sequence.

4.3.2 Patterns from single sequence data

Qualitative patterns

To the best of our knowledge, Villafane et al. [152] were the firsts to address sequential interval-based patterns for a single sequence, often called interval series. They proposed the containement pattern language that is based on the *during* relation proposed by Allen. The main idea of the authors it to transform an interval sequence to containment graphs $G = (V, E)$ where each node in V corresponds to a unique interval and each edge in E



(a) An interval sequence

(b) Containment graph

Figure 4.3.7: An interval sequence and its corresponding containment graph. For the sake of readability all edges are not depicted in the containment graph (e.g A contains D)

links two nodes with a containment relationship. An example is depicted in 4.3.7. It is to notice that with that model, the relationship between, for example, interval A and E (i.e before) is not taken into account. As a consequence, containment patterns are **not complete**. However, this model can be sufficient to model and discover hierarchical temporal relations. Containment patterns can be considered as **non-ambiguous**. Indeed, the *during* relation is specified between all pairs of nodes. The approach proposed by the authors is to transform the interval sequence to a containment graph and enumerate and count all possible containment relationships with a depth-first strategy (graph based algorithm).

Zhang et al. [172] proposed an approach devised to mine similar patterns. The proposed pattern model, called *During Temporal Pattern* (DTP), is also based uniquely on Allen' *during* relationship. In that work, the authors proposed to divide the single sequence into multiple small databases containing intervals having the same label. Thus, when exploring the search space of DTPs, computing the support of each *during* relationship is done via a join of these databases. This can remind the reader of the vertical representation discussed for sequence databases (cf. Figure 4.2.2 without sequence IDs). The authors proposed a level-wise approach using the former sequence partition and joins to compute pattern support.

Another pattern model, named *Relationship Matrix* (RM), was proposed by Höppner and Klawonn [81]. The authors proposed a mining approach inspired by episode mining [101] that uses a window-based frequency measure. They used an Apriori candidate generation algorithm. Besides, rules are derived from the set frequent RM in the same formalism for episode rule. A rule is denoted $X \Rightarrow Y$ where X is the premise pattern and Y the rule pattern such that X is a sub-pattern of Y. They also interestingly discussed the use of window-based frequency to compute pattern rules. They argued that pattern extension with fixed-width sliding windows could cause a counter-intuitive drop in confidence measure as the conclusion part of the rule may not fit entirely for all time windows where the premise appear. They proposed a new rule semantics defined as follows.

> *Given a randomly selected sliding window that contains an instance of the premise pattern, then with probability [confidence] p this window overlaps a sliding window that contains the rule pattern.*

This new confidence definition increases the confidence value as more sliding windows can fit it in comparison with the traditional frequency confidence. Notice that this issue can be avoided using a traditional support measure based for example on minimal occurrences [101]. However, authors chose not to use this latter arguing that it induces a less robust discovery on noisy data as it can miss temporal relations between "non-noise" intervals if, for example, noise intervals of the conclusion's type occur in the gap between the original intervals. In addition

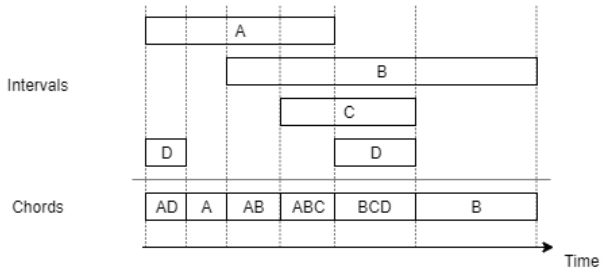


Figure 4.3.8: Chords sequence corresponding to an interval sequence

to the confidence measure, authors also proposed to use a J-Measure to rank rule patterns by their information content.

Another pattern model, named Time Series Knowledge Representation (TSKR), was proposed by Mörchen [104, 105]. The originality of this work compared with previously cited contributions is the use of coincidence as the core relationships for interval-based patterns. The relation between two intervals is assessed w.r.t to their overlapping duration rather than the count of this relation occurrence count. Two pattern models (contained in the TSKR representation, that is to be seen more as a framework) were proposed.

First, *chords* patterns express simultaneity between intervals. It can be defined as a subset IS of the symbol set I. It describes time intervals where all intervals of types $i_k \in IS$ coincides, i.e. occur simultaneously. The coverage interestingness measure of a chord, called support in that work, is defined w.r.t total duration in the sequence where all intervals $i_k \in IS$ are active simultaneously. To avoid ambiguity of terms, we will refer to this coverage measure as duration support. Authors also defined margin-closure for chord patterns.

Definition 21 (Margin-closed chord)

A *chord* c_i is said to be margin-closed w.r.t to a threshold $0 < \alpha < 1$ if there are no *super-chord* c_j that have almost the same support:

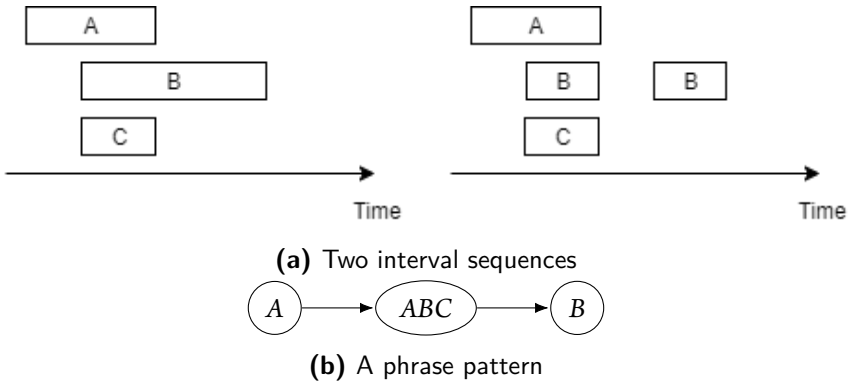
$$\frac{supp(c_i)}{supp(c_j)} < 1 - \alpha$$

The margin-closed chord is a pattern of several intervals occurring mostly together. It is to remark that margin-closedness is a generalization of closed patterns. Mining "frequent" chords consist of finding all margin-closed subset of symbols occurring together with a duration support exceeding the user given minimum threshold. This task is very similar to frequent itemset mining. Thus, the authors proposed to use the enumeration approach of CHARM, proposed by Zaki and Hsiao [169], that is a frequent itemset mining algorithm using a depth-first exploration strategy. *Chords* patterns are non-ambiguous but clearly non-complete. Indeed, they only permit to express simultaneous relations (i.e. a relaxed version of Allen' *equals* relation).

The second pattern model, that is more of interest for this work, are *phrase* patterns. A *phrase* is defined similarly to episodes [101] but for interval-based data. A phrase is a partial order of $k > 0$ chords. Phrase pattern are **ambiguous** as can be shown by Figure 4.3.9.

For this pattern model, a traditional occurrence counting support is used. The margin-closedeness is defined as for chord pattern with respect to a parameter α. Authors proposed an algorithm for mining margin-closed *phrase* patterns that is inspired by the work of [24] that addressed mining closed partial order from itemset sequence databases.

The proposed algorithm is detailed in the following. First, it converts a chord sequence into a series of interval

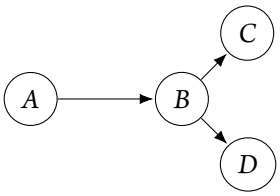


(a) Two interval sequences

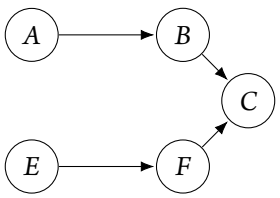
(b) A phrase pattern

Figure 4.3.9: Two interval based sequences (a) supporting the same phrase pattern (b)

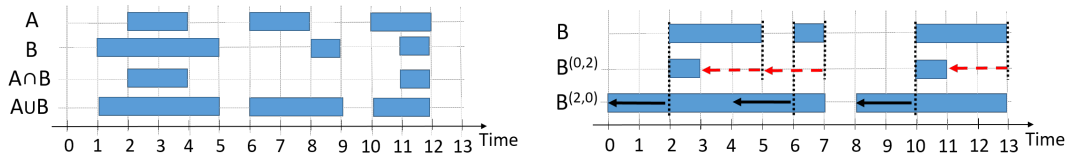
itemset considering a minimum duration parameter ε: an item is created for each chord lasting at least ε time units. The obtained itemset sequence is converted into a database of sequences using a non-overlapping windowing (in contrast with a sliding window approach) in order to be able to apply a closed frequent pattern algorithm (e.g. CLOSPAN or BIDE). Mörchen [104] argued that the non-overlapping approach can be suggested by the data (e.g. if significant gaps are between groups of intervals) or can be given *à priori* (w.r.t to the data generation process). Other techniques can also be used (e.g. minimal occurrences). After that, the algorithm uses a sequential pattern mining algorithm (qualitative)s to obtain closed frequent sequences w.r.t the user-given minimum support. The output of the algorithm must keep track of transactions T, where a closed sequential pattern s occurs. From the obtained pairs (s, T) the algorithm builds groups (S, T) where S is a set of all closed patterns occurring in all transactions in T. Frequent patterns in S are the patterns which are margin-closed with respect to a and that S is maximal, i.e. there is no other pattern occurring in all transaction in T and no additional transaction $t \notin T$ where patterns in S occur. The last step of the algorithm builds partial order, *phrase*, for each margin-closed set of sequential patterns. The partial order must represent each sequential pattern in S, and the partial order can represent no other pattern. For instance, from a set $S = \{\langle A, B, C\rangle\langle A, B, D\rangle\}$ where A, B, C, and D are set of intervals symbols, the partial order that is built is represented as follows.



Another example with a set $S = \{\langle A, B, C\rangle\langle E, F, C\rangle\}$ follows.



We refer to the original work [105] and the earlier contribution of Casas-Garriga [23] for more algorithmic details.



(a) Intersection and union **(b)** (α, β)-transformation

Figure 4.3.10: Examples of intersection and union and (α, β)-transformation

Quantitative patterns

A first work that addressed mining quantitative patterns from single sequences was proposed by Peter et al. [123, 124] who introduced the *Pattern graphs* (PG) model. PG is very similar to TSKR and CR in that it is defined as a partial order between several nodes. Additionally, Pattern Graphs may specify the absence of specific intervals in a given node and can include "do not care" nodes permitting to model gaps between successive nodes. The quantitative information is specified for interval duration and defined as a range that is initialized with $[1, \infty]$ and refined by the mining algorithm. In [123], the authors proposed an algorithm for learning pattern graphs that is not based on frequent pattern mining but used a heuristic search (beam search).

Plantevit et al. [127] designed an algorithm, *TEDDY*, devised to discover quantitative pairwise dependencies between interval streams. Contrary to the majority of mentioned contributions, authors proposed to use exclusively the intersection of state streams (a stream contains all intervals of a given type) as a coverage interestingness measure. Other contributions using coincidence, as TSKR with chords, used coincidence for discovering simultaneous relations but the partial order is mined w.r.t to minimum support. Besides, these algorithms require thresholds as the minimum support or frequency (except [123]) while TEDDY utilizes statistical tests to set significance thresholds automatically. It also focuses on reliability rather than coverage interestingness measures. Follows more details algorithm TEDDY.

A pairwise temporal dependency between interval-based streams A and B, noted $A \rightarrow B$, notifies that A *occur simultaneously with* B. We call hereafter A the premise and B the conclusion. $A \rightarrow B$ is assessed with the intersection length of A and B. The intersection of two state streams A and B, noted $A \cap B$ is a state stream containing intervals where both A and B are active (Fig.4.3.10). Formally, $A \cap B = (a \wedge b, \langle [t_{b_i}, t_{e_i})\rangle)$ such that $\forall t \in [t_{b_i}, t_{e_i}), \exists [t_{b_j}, t_{e_j}) \in A, \exists [t_{b_k}, t_{e_k}) \in B$ such that $t \in [t_{b_j}, t_{e_j})$ and $t \in [t_{b_k}, t_{e_k})$. This operation is computed in $\Theta(Max(\#A, \#B))$ where $\#A$ and $\#B$ denotes respectively the number of intervals in A and B.

The intersection length is used to define the following confidence measure.

$$conf(A \rightarrow B) = \frac{len(A \cap B)}{len(A)}$$

Notice that confidence is maximal (=1) if all active intervals of A are included in active intervals of B: A occurs *always* with B. For example, in Fig. 4.3.10 $conf(A \rightarrow B) = \frac{3}{6}$.

In order to find relations of interest, and to avoid the utilization of a user-given threshold, this confidence measure is statistically assessed via a Pearson χ^2 test of independence. The independence hypothesis states that A and B are statistically independent within a duration of $\mathcal{T}$. It relies on the following assumption: if the active length of B is uniformly distributed in $\mathcal{T}$, there is no significant correlation between A and B. The given validity

threshold on confidence is noted $th(len(A), len(B))$ and is given by:

$$th(lp, lc) = \frac{lp * lc + \sqrt{\frac{3.84}{T} lp * lc(T_{obs} - lc)(T_{obs} - lp)}}{T_{obs} * lp}$$

for a significance level of 0.05. This thesis extends these results and more details about this statistical test will be given in further chapters.

An important remark is that simultaneity do not permit to express dependencies when it happens that B is time-delayed regarding A. The conclusion stream can be temporally shifted with an (a, β)-transformation to obtain a relation of type $A \rightarrow B^{(a,\beta)}$. This operation results on a state stream $B^{(a,\beta)} = \langle [t_{b_i} - a, t_{e_i} - \beta) \mid [t_{b_i}, t_{e_i}) \in B \rangle$. Hereafter, a is called *expansion* and β *reduction*. We describe in Fig. 4.3.10 two examples: a $(0, 2)$-reduction $B^{(0,2)}$ and a $(2, 0)$-expansion $B^{(2,0)}$. This temporal transformation is done in $\Theta(\#B)$. Therefore, the associated confidence measure for a delayed temporal relation is given by:

$$conf(A \rightarrow B^{(a,\beta)}) = \frac{len(A \cap B^{(a,\beta)})}{len(A)}$$

A dependency $A \rightarrow B^{(a,\beta)}$ means that: B starts at most a time units after A and B finishes at least β time units after A. For simplicity purposes, we refer hereafter to this relation with: if A *is active then* $B^{(a,\beta)}$ *is active.*

The authors propose an algorithm, TEDDY, devised to discover such dependencies given a temporal constraint Δ on temporal transformation and by extension time lags. It explores the search space as a semi-lattice defined by temporal transformations inclusion (i.e if an (a, β)-transformation sequence contains all intervals of (a', β')-transformation then (a, β) includes (a', β')) using a breadth-first strategy. As the used confidence measure is monotonic, multiple temporal dependencies can express the same temporal relation. Therefore, TEDDY integrates pruning criteria based on a dominance relationship permitting to select the most specific dependencies by refining temporal transformations while controlling the loss in the intersection-based confidence measure. In [127], the authors used pairwise dependencies to build a dependency graph. Further discussions on TEDDY will be provided in next chapters.

The use of intersection length permits to obtain dependencies with a different meaning for frequent pattern mining. The use of intersection as an interestingness measure for streaming data can, in some cases, provide a better assessment of the strength of a temporal relation than the use of support. This will be discussed in the next chapter.

Another approach that can be applied to single sequences is PIVOTMiner [78] that we already discussed. While not being designed for single sequences, this approach can be adapted easily to this data format as it is not endpoint-sensitive. Indeed, it considers the relative temporal relations between intervals rather than their absolute positions or the other quantitative interval-based pattern mining approaches for sequences databases.

5

Complex Temporal Dependencies

Contents

5.1 Introduction

In this section, we describe and motivate the knowledge representation that we aim to extract from a set of state streams. The proposed pattern or dependency model, named Complex Temporal Dependencies (CTD) aims to generalize the pairwise dependency model using an intersection-based assessment proposed by Plantevit et al. [127] to dependencies involving multiple states.

In the rest of this chapter, we will first discuss the main motivations behind an intersection-based assessment of dependencies. This approach will be compared to the widely used counting support assessment for particular examples. A detailed description of the automatic statistical intersection-based assessment will also be provided. Secondly, we introduce the CTD dependency model that permits to intersection-based dependencies between multiple states. Finally, we will discuss how CTDs can be used to build models of typical temporal phenomena.

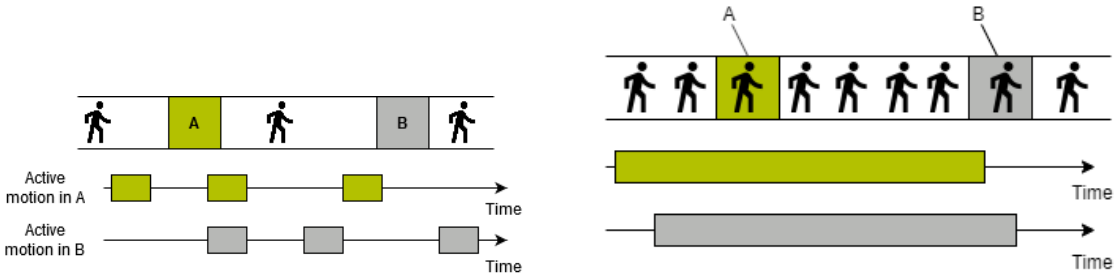
5.2 Background and motivations

5.2.1 Intersection-based assessment

The majority of approaches devised to extract temporal patterns from interval-based data use an occurrence counting support as a coverage interestingness measure. As discussed in Chapter 4, the occurrence counting support may have different underlying semantics depending on the temporal data format and pattern constraints (e.g. proportion of sequences, number of occurrences, the proportion of time windows).

The common point of these support and frequency measures is their relation assessment, or how relationships between intervals contribute to the interestingness measure. Indeed, they are counted on a qualitative basis without regard to intervals durations as it can be done for time point data. This qualitative counting approach can be sufficient for a wide range of cases, particularly when intervals duration is not of interest for an application domain, with sparse data or with homogeneous intervals durations. However, we argue in this work that this assessment approach can be misleading with interval-based streams in that the interestingness of a given temporal relation can be underestimated, particularly with attribute-centred data as sensory data. Indeed, the useful monotonic property of an interestingness measure (the more occurrences of a given pattern, the greater is the measure) is not always guaranteed. We believe that the intersection-based assessment, as proposed by [127], provides better insights. Temporal relation between two state streams A and B is assessed based on the length of their intersection $A \cap B$. The evaluation of delayed relations is obtained via temporal transformations. For instance, $A \cap B^{(t,t)}$ is used to assess the relation *A occurs before B with a time delay of t*. To illustrate the differences between the before mentioned assessment approaches, let us consider the following simple example using sensory data.

> **Example 8**
>
> *Figure 5.2.1 describes a corridor equipped with two video cameras monitories the areas A and B. A motion analysis algorithm is applied to videos given by A and B and produces two interval-based streams describing the validity of states Motion in area X (in each area A and B). In this environment, the typical temporal phenomenon is a uni-directional trajectory described by the qualitative relation "A before B": pedestrians pass through A then B after a time delay t. For the sake of simplicity, let us assume that intervals of A and B have a unique duration d. Let us consider two activity scenarios. In Figure 5.2.1a, few pedestrians pass through the corridor. The produced state streams can be described as "sparse" as they contain a number of intervals that is equivalent to the number of pedestrians involved in the motion phenomena. The second case is that of dense*



(a) Sparse activity **(b)** Dense activity

Figure 5.2.1: A corridor equipped by two video cameras providing motion state streams for two areas A and B

> *interval-based streams produced when activity in the corridor is intense: pedestrians follow each other in the motion analysis area without any temporal separation, as shown in Figure 5.2.1b. In that case, state streams A and B contain one long interval each that reports on data produced by the overall set of pedestrians (the state Motion is continuously activated in each monitored area).*

Let us now assess the temporal relation between A and B with occurrence counting and intersection-based assessments. Considering the first scenario with spare activity (3 trajectory occurrences), the assessment of the temporal relation between A and B is the following:

- Occurrence counting: A *before* B has 3 occurrences
- Intersection-based: $\mathbf{len}(A \cap B^{(t,t)}) = 3 * d$

In this case, both occurrence counting and intersection provide a good strength assessment of relation A *before* B. It is proportional to the number of real-world occurrences: trajectory A then B was performed three times. In the case of dense data (with nine pedestrians walking successively):

- Occurrence counting assessment: A *overlaps* B has one occurrence.
- Intersection-based assessment: $\mathbf{len}(A \cap B^{(t,t)}) = d'$ with $3 * d < d' \leq 9 * d$

Two things are worth noticing with the occurrence counting assessment. First, the occurrence counting score is smaller with dense data describing more trajectories occurrences (a score of 3 is counted for sparse data describing three trajectories occurrences while a score of 1 is obtained for nine trajectories occurrences.). This assessment can be misleading and counter-intuitive as it does not respect the expected monotonicity: the more a phenomenon is performed, the more its score is higher. Also, this can problematic in a pattern discovery process as the interest of patterns (here occurrences of a trajectory) is no longer correlated with the assessment value. Secondly, the qualitative Allen relation that appears with dense data, A *overlaps* B, does not describe properly the physical phenomenon (a trajectory described as with a *before* relationship). On the other hand, the intersection-based assessment value is monotonic with the activity described in the interval streams: the more frequent are the occurrences of the given trajectory, the greater is the intersection length. For the sake of clarity, we stress the fact that the intersection of streams is not necessarily strictly proportional to the number of temporal relations occurrences. This is due to the attribute-centred nature of the data generation process as in the former example: intervals reporting on the same trajectory are overlapping. We can express the intersection length value $A \cap B^{(t,t)}$ as:

$$d' = 9 * d - \sum \mathbf{len}(overlaps)$$

where $9 * d$ is the proportional value of intersection length (theoretical subject-centred maxima) and $\sum$ *overlaps* is the sum of valid time intervals' length that describes the motion of at least two pedestrians. Nevertheless, the intersection length remains monotonic with respect to the trajectories occurrences. As a consequence, we consider that an intersection based temporal relation provides more insights and is more easily interpretable. Second, the quantitative temporal relation obtained with dense data is the same as for "sparse" data and assessed with $A \cap B^{(t,t)}$. The core information given by this assessment approach is quantitative. The qualitative relation between interval streams can be extracted from such quantitative information.

5.2.2 Pairwise Temporal dependencies, Confidence and statistical assessment

In Chapter 4, we have seen that the straightforward use of intersection-based assessment for interval sequences was used in [127] for state streams pairwise dependencies. We recall that a dependency between a state stream A and state stream B, noted $A \rightarrow B$, is assessed via an intersection-based confidence measure:

$$\mathbf{conf}(A \rightarrow B) = \frac{\mathbf{len}(A \cap B)}{\mathbf{len}(A)}$$

It is a reliability measure that reports on the proportion of validity duration of A that is simultaneous with B. For instance, a dependency $A \rightarrow B$ having a confidence of 1 means that *A is always valid with B*. What is assessed here is the temporal dependency or "correlation" between the validity of two states. Delayed temporal dependencies use temporal transformations and is assessed in the same manner. For instance, the confidence of dependency between A and $B^{(\alpha,\beta)}$, noted $A \rightarrow B^{(\alpha,\beta)}$, is assessed with:

$$\mathbf{conf}(A \rightarrow B^{(\alpha,\beta)}) = \frac{\mathbf{len}(A \cap B^{(\alpha,\beta)})}{\mathbf{len}(A)}$$

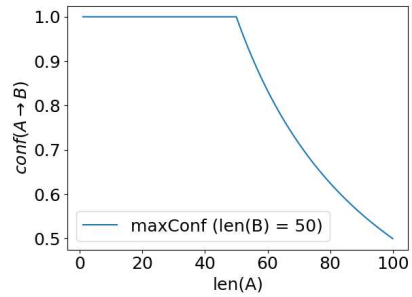
In the rest of this manuscript, we will use notations without temporal transformations when these later are not relevant or not necessary for comprehension (transformed streams are also intervals streams).

This confidence value is higher-bounded by the following quantity:

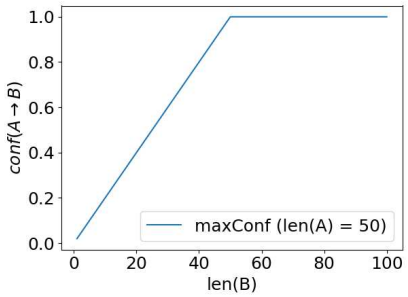
$$maxConf(A \rightarrow B) = \frac{\mathbf{min}\big(\mathbf{len}(A), \mathbf{len}(B)\big)}{\mathbf{len}(A)}$$

This higher-bound can be easily proven as $\mathbf{len}(A \cap B) \leq \mathbf{min}\big(\mathbf{len}(A), \mathbf{len}(B)\big)$. Figure 5.2.2 shows the maximum confidence with respect to premise and conclusion length. What can be noticed is that the maximum confidence is monotonic with respect to both premise and conclusion length.

The interest of a given dependency is usually expressed via a user-given threshold (e.g. support threshold) that defines the minimum value of an interestingness measure of a relation of interest for the user. The definition of such thresholds may require prior knowledge on datasets to be processed by knowledge discovery techniques. As a consequence, in many cases, including when prior insights about a data are not available or if the knowledge discovery process needs to be fully automatized), setting a user-defined interest threshold may be a difficult and costly task. This is particularly true with large datasets and for pattern languages defining large search spaces (cf. Chapter 4). The contribution of [127] aims to discover pairs of interval-based streams that are statistically dependent. Two state streams are said to be in a statistically valid temporal dependence if their intersection is sufficiently large to not be due to chance. More specifically, they used a Pearson χ^2 test of independence on



(a) Maximum confidence w.r.t premise length



(b) Maximum confidence w.r.t conclusion length

Figure 5.2.2: Maximum confidence of a dependency $A \rightarrow B$ with respect to premise and conclusion length

intersection length to determine a statistical significance threshold on confidence value. We describe hereafter the details of this statistical test.

The Person χ^2 test of independence is performed to determine whether two state streams A and B are statistically independent over an observation duration $T_{obs} = [t_{begin}, t_{end})$ with:

$$t_{begin} = min(\min_{\forall [t_i, t_{i+1}) \in A} t_i, \min_{\forall [t_j, t_{j+1}) \in B} t_j) \text{ and } t_{end} = max(\max_{[t_i, t_{i+1}) \in A} t_{i+1}, \max_{[t_j, t_{j+1}) \in B} t_{j+1})$$

A time point (or elementary time interval in the discrete time domain) of T_{obs} can belong or not to an interval of a state stream A. Given that observation, it is possible to build a contingency table O, Table 5.2.1, which partitions the duration of T_{obs} into four possible observed outcomes expressed in terms states A and B validity.

Table 5.2.1: Observed contingency table O

	B	$\overline{B}$
A	$\mathbf{len}(A \cap B)$	$\mathbf{len}(A) - \mathbf{len}(A \cap B)$
$\overline{A}$	$\mathbf{len}(B) - \mathbf{len}(A \cap B)$	$\mathbf{len}(T_{obs}) - \mathbf{len}(A) - \mathbf{len}(B) + \mathbf{len}(A \cap B)$

The statistical test of independence aims to compare these observed outcomes to the expected outcomes under the independence hypothesis. The null hypothesis of this statistical test of independence states that the validity of states A and B is statistically independent. In order to build an expected contingency table under the independence assumption, we make the following hypothesis.

Hypothesis 5 (Independence under the uniform distribution)
Let A and B two non-empty state streams over an observation duration T_{obs} . If A or B occurs uniformly over T_{obs}, then states streams A and B are independent.

Following the later hypothesis, if one supposes that the validity of A is uniformly distributed over T_{obs}, the probability that an elementary interval of B (a valid sub-interval of B of length 1) co-occurs with a valid state A is $\frac{\mathbf{len}(A)}{\mathbf{len}(T_{obs})}$. As a consequence, as B has valid during a duration of $\mathbf{len}(B)$, the co-occurrence duration of A and B (i.e the length of their intersection) under the null hypothesis is given by:

$$\frac{\mathbf{len}(A) * \mathbf{len}(B)}{\mathbf{len}(T_{obs})}$$

The three other outcomes are computed following the same reasoning. The expected contingency table is given by Table 5.2.2.

Table 5.2.2: Expected contingency table E under the null hypothesis (independence)

	B	$\overline{B}$
A	$\frac{\mathbf{len}(A) * \mathbf{len}(B)}{\mathbf{len}(T_{obs})}$	$\frac{\mathbf{len}(A) * \big(\mathbf{len}(T) - \mathbf{len}(B)\big)}{\mathbf{len}(T_{obs})}$
$\overline{A}$	$\frac{\big(\mathbf{len}(T) - \mathbf{len}(A)\big) * \mathbf{len}(B)}{\mathbf{len}(T_{obs})}$	$\frac{\big(\mathbf{len}(T) - \mathbf{len}(A)\big) * \big(\mathbf{len}(T) - \mathbf{len}(B)\big)}{\mathbf{len}(T_{obs})}$

Given the observed and expected contingency tables, the value of the test-statistic is given by:

$$\begin{aligned} X^2 &= \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \\ &= \frac{\mathbf{len}(T)\big(\mathbf{len}(T)\mathbf{len}(A \cap B) - \mathbf{len}(A)\mathbf{len}(B)\big)^2}{\mathbf{len}(A)\mathbf{len}(B)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(B)\big)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(A)\big)} \end{aligned} \tag{5.1}$$

The null distribution of the statistic value is approximated by the χ^2 distribution with 1 degree of freedom ($df = (2-1) * (2-1) = 1$) for a traditional significance level of 0.05. The corresponding critical value is $\chi^2_{0.05} = 3.84$. As a consequence, the test-statistic value X has to be greater than $\chi^2_{0.05}$ to consider that the co-occurrence duration is very likely to not to be due to chance. The following quadratic inequation in $\mathbf{len}(A \cap B)$ is derived from equation 5.1:

$$\begin{aligned} X^2 &\geq \chi^2_{0.05} \\ \frac{\mathbf{len}(T)\big(\mathbf{len}(T)\mathbf{len}(A \cap B) - \mathbf{len}(A)\mathbf{len}(B)\big)^2}{\mathbf{len}(A)\mathbf{len}(B)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(B)\big)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(A)\big)} &\geq 3.84 \\ \big(\mathbf{len}(T)\mathbf{len}(A \cap B) - \mathbf{len}(A)\mathbf{len}(B)\big)^2 &\geq \frac{3.84}{\mathbf{len}(T)}\mathbf{len}(A)\mathbf{len}(B) * \\ &\quad \big(\mathbf{len}(T_{obs}) - \mathbf{len}(B)\big)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(A)\big) \end{aligned}$$

The roots of this equation, noted $\cap_1$ and $\cap_2$, are then given by:

$$\cap_1 = \frac{\mathbf{len}(A)\mathbf{len}(B) - \sqrt{\frac{3.84}{\mathbf{len}(T_{obs})}\mathbf{len}(A)\mathbf{len}(B)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(B)\big)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(A)\big)}}{\mathbf{len}(T_{obs})}$$

$$\cap_2 = \frac{\mathbf{len}(A)\mathbf{len}(B) + \sqrt{\frac{3.84}{\mathbf{len}(T_{obs})}\mathbf{len}(A)\mathbf{len}(B)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(B)\big)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(A)\big)}}{\mathbf{len}(T_{obs})}$$

As a consequence inequation 5.2 is verified for $0 \leq \mathbf{len}(A \cap B) \leq \cap_1$ (case 1) and $\cap_2 \leq \mathbf{len}(A \cap B) \leq \mathbf{len}(T_{obs})$ (case 2). Both these two cases denotes situations permitting to reject the null hypothesis of independence but have different interpretations:

- Case 1: the intersection length is lower than expected under the independence hypothesis.
- Case 2: the intersection length is higher than expected under the independence hypothesis. Having such intersection length means that a dependency between A and B exists and may permit to explain partly the validity duration distribution of both A and B.

In this work, we focus on Case 2 intersection lengths as we aim to find significant dependencies between state streams expressing *typical* temporal relations. Nevertheless, the case 1 intersection lengths (anomalies or anti-dependencies) are worth to be studied in order to provide proper interpretation or infer anti-dependencies: Can we infer dependencies of the form $A \rightarrow \overline{B}$ and $B \rightarrow \overline{A}$ from such intersection length cases?

In order to provide a significance threshold on confidence value, $\cap_2$ is normalized by $\mathbf{len}(A)$:

$$\begin{aligned} \mathbf{len}(A \cap B) &\geq \cap_2 \\ \frac{\mathbf{len}(A \cap B)}{\mathbf{len}(A)} &\geq \frac{\cap_2}{\mathbf{len}(A)} \\ \mathbf{conf}(A \rightarrow B) &\geq \frac{\cap_2}{\mathbf{len}(A)} \end{aligned} \tag{5.2}$$

Then, a dependency $A \rightarrow B$ is said to be valid, iff. $conf(A \rightarrow B) \geq \frac{\cap_2}{\mathbf{len}(A)}$. For conciseness, the statistical threshold of $A \rightarrow B$ will be noted $th(A \rightarrow B)$ in the rest of this thesis. Nevertheless, it is known that the χ^2 needs large sample sizes to be powerful enough and provide acceptable results. To guarantee this fact, we use, as in [127], a conventional rule of thumbs that enforces the expected values in Table 5.2.2 to be greater than 5. Otherwise, the dependency is not considered. As a consequence, very sparse state streams and extremely dense ones are not considered: E_{11} and E_{22} can be respectively very small. In such cases, one may use other statistical independence tests that are more accurate with small sample sizes as *Fisher' exact test of independence*. We do not investigate this direction in this work and focus on dependencies inducing large expected outcomes.

Statistical threshold function study

One main difference between the complex temporal dependencies and other pattern models is the assessment approach. The assessment of CTDs is based on dynamic thresholds obtained from statistical tests of independence rather than a constant (user-given) interest threshold which is monotonic by definition. In sequential pattern mining approaches, this property is useful at different levels: defining closure, Apriori pruning ... In this subsection, we study the dynamic statistical threshold behaviour.

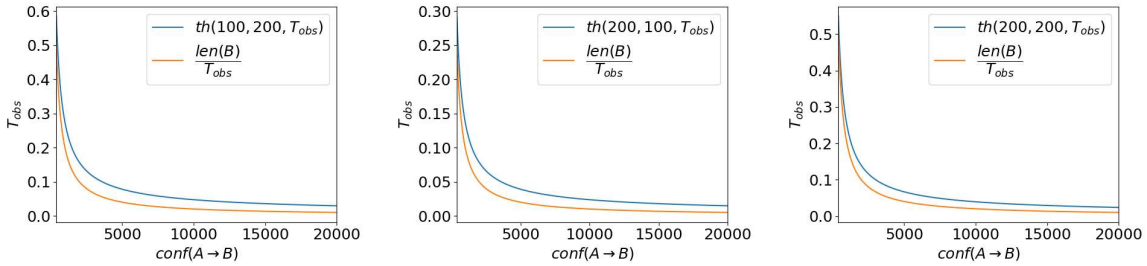
The statistical threshold defined above can be seen as a function of 3 parameters:

$$th(A \rightarrow B) = th(\mathbf{len}(A), \mathbf{len}(B), T_{obs})$$

Hereafter, we study the behaviour of this function w.r.t to its three variables.

Behaviour of $th(A \rightarrow B)$ with respect to T_{obs}.

Firstly, let us study the behaviour of the statistical thresholds when lengths of the premise and the conclusion are constant and the observation duration is increasing. We describe in Figure 5.2.3 the behaviour of *th* in



(a) $\mathbf{len}(A) < \mathbf{len}(B)$ **(b)** $\mathbf{len}(A) > \mathbf{len}(B)$ **(c)** $\mathbf{len}(A) = \mathbf{len}(B)$

Figure 5.2.3: Evolution of *th* for a dependency $A \rightarrow B$ with respect to T_{obs}. $T_{obs} \in [400, 20000]$

different configurations $\mathbf{len}(premise) < \mathbf{len}(conclusion)$, $\mathbf{len}(premise) > \mathbf{len}(conclusion)$ and $\mathbf{len}(premise) = \mathbf{len}(conclusion)$.

In these figures, one can notice that for a fixed premise and conclusion length, the statistical threshold is decreasing monotonically with T_{obs} and tends towards 0. This is a direct result from the statistical threshold formula.

Proof. Let $len(A)$ and $len(B)$ be constant positive values:

$$
\begin{aligned}
th(T_{obs}) &= \frac{\mathbf{len}(A)\mathbf{len}(B) + \sqrt{\frac{3.84}{\mathbf{len}(T_{obs})}\mathbf{len}(A)\mathbf{len}(B)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(B)\big)\big(\mathbf{len}(T_{obs}) - \mathbf{len}(A)\big)}}{\mathbf{len}(T_{obs})len(A)} \\
&= \frac{A + \sqrt{\frac{B * len(T_{obs})^2 + C * len(T_{obs}) + D}{len(T_{obs})}}}{E * len(T_{obs})} \\
&= \frac{A}{E * len(T_{obs})} + \frac{\sqrt{B * len(T_{obs})^2 + C * len(T_{obs}) + D}}{E * len(T_{obs})^{1.5}}
\end{aligned}
$$

with A, B, C, E positive values. $\frac{A}{E * T_{obs}}$ is strictly decreasing with $T_{obs} \geq 1$. Let $f(x) = \frac{\sqrt{B * x^2 + C * x + D}}{E * x^{1.5}}$. $f(x+1) - f(x)$ can be expressed as follows.

$$
\begin{aligned}
f(x+1) - f(x) = &\frac{\sqrt{B(x^5 + 2x^4 + x^3) + C(x^4 + x^3) + Dx^3}}{E\big(x(x+1)\big)^{1.5}} \\
&- \frac{\sqrt{B(x^5 + 3x^4 + 3x^3 + x^2) + C(x^4 + 3x^2 + 3x + x) + D(x+1)^3}}{E\big(x(x+1)\big)^{1.5}}
\end{aligned}
$$

For $x \geq 1$, it is easy to proof that $f(x+1) - f(x) < 0$. Then; for $len(T_{obs}) \geq 1$, $\frac{\sqrt{B * len(T_{obs})^2 + C * len(T_{obs}) + D}}{E * len(T_{obs})^{1.5}}$ is strictly decreasing with T_{obs}. Then, we conclude that statistical threshold is strictly decreasing with T_{obs} if A and B have constant lengths.

As $\lim_{T_{obs} \rightarrow \infty} \frac{A}{E * T_{obs}} = 0$ and $\lim_{T_{obs} \rightarrow \infty} \frac{\sqrt{B * T_{obs}^2 + C * T_{obs} + D}}{E * T_{obs}^{1.5}} = 0$, we can conclude that the statistical thresholds approaches 0 if T_{obs} approaches infinity □

Several observations can be made straightforwardly from these properties:

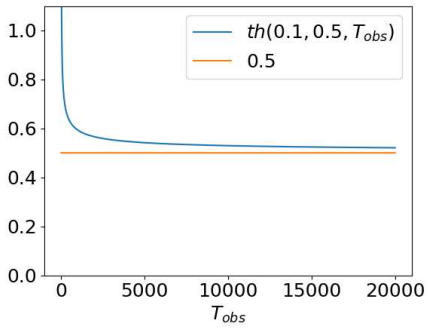


Figure 5.2.4: Statistical threshold w.r.t to observation duration T_{obs} with premise and conclusion length in constant proportion of T_{obs}

- Let a dependency $A \rightarrow B$ be statistically valid for a given T_{obs}. If no additional intervals are added to A or B within $[T_{obs}, +\infty[$, then $A \rightarrow B$ is statistically valid for $[0, T'[$ where $T' \in [T_{obs}, \infty[$.
- If $conf(A \rightarrow B) = \varepsilon > 0$, then exists a $T_{obs} \in [0, +\infty]$ such that $A \rightarrow B$ is statistically valid for all $T \geq T_{obs}$.

The first observation can be useful in a streaming context where a system aims to maintain a set of valid dependencies between streams. More precisely, if a dependency is found to be valid for a given T_{obs}, it is possible to estimate the maximum loss in confidence value in the worst case (no intersection between new intervals of A and B are observed) and run new dependency analysis when this quantity reaches the minimum threshold. This aspect will not be considered in this work but can be explored in further work. The second observation is problematic since every non-null confidence value may lead to statistically valid dependency. This problem is tackled by the use of the rule of thumbs used in the statistical threshold computation that states that any value in the expected outcomes contingency table has to be greater than 5.

From another perspective, we study the behaviour of the statistical threshold when lengths of the premise and the conclusion have a constant proportion of T_{obs}. This can be seen as a simplification of situations when the data generation process is constant over time: it generate an amount of data that is proportional to the observation duration. Let $x = \dfrac{\mathbf{len}(A)}{T_{obs}}$ and $y = \dfrac{\mathbf{len}(B)}{T_{obs}}$ be respectively the observation time length proportion of the premise (A) and the conclusion (B). The statistical threshold of a dependency $A \rightarrow B$ can be expressed as follows:

$$th(A \rightarrow B) = \frac{xy + \sqrt{\dfrac{3.84}{T_{obs}} xy(1-x)(1-y)}}{x}$$

If one considers that x and y are constant values, one can remark that the statistical threshold approaches y when T_{obs} approaches infinity. This can be remarked in Figure 5.2.4 for $x = 0.1$ and $y = 0.5$. This observation indicates that if a dependency has a confidence that is strictly higher than the lower bound $\dfrac{len(B)}{T_{obs}}$ for all T_{obs}, then exists a T such that this dependency is statistically valid for all $T_{obs} \in [T, \infty[$. This suggests that if a temporal phenomena between A and B is occurring uniformly over time, i.e if the data generation process is constant, a dependency between A and B is to be valid if enough observation duration is available. For instance, in Figure 5.2.4, let us consider that a dependency $A \rightarrow B$ has a dependency with confidence 0.55 for every T_{obs}. In this case, $A \rightarrow B$ is non statistically valid for a duration of $T_{obs} = 1000$ while it is for all $T_{obs} > 10000$.

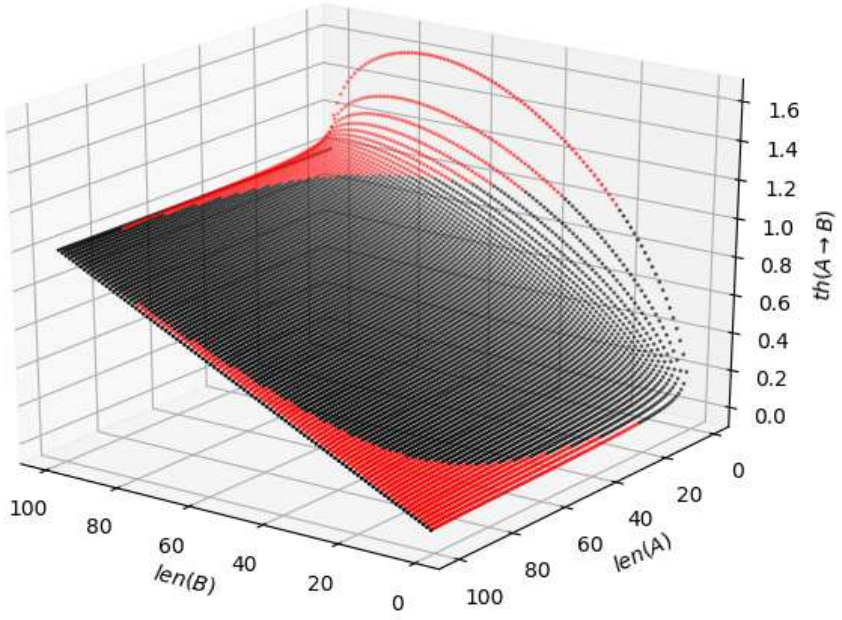


Figure 5.2.5: Statistical threshold w.r.t premise and conclusion length. $T_{obs} = 100$, $len(A) \in [1, 100]$ and $len(B) \in [1, 100]$. Red points are thresholds that are strictly greater than the confidence higher bound.

From a probabilist point of view, y can be seen as the probability of obtaining a valid elementary interval $[t, t+1)$ in a duration T_{obs} in the conclusion stream, noted $P(conclusion)$, and the confidence value interpreted as the conditional probability of obtaining a valid conclusion knowing a valid premise, noted $P(conclusion|premise)$. The statistical threshold is lower bounded by $P(conclusion)$ as if $P(conclusion) = P(conclusion|premise)$ then the premise and the conclusion are independent. From the precedent paragraph, one can say the minimum conditional probability $P(conclusion|premise)$ to assert that a dependency is valid approaches $P(conclusion)$ with T_{obs}. The straightforward interpretation of the former statement can be trivial: more data produce more accurate dependencies as less significant dependencies are left. However, this result can be used in order the quantify the degree of "accuracy" of a given model. Indeed, for a given x and y, one can specify a threshold on quantity $\varepsilon = th(x, y, T_{obs}) - y$, that correspond to minimum observation duration to consider that a statistical analysis provides satisfactory outcomes. This can be formalized in future work.

Behaviour of $th(A \rightarrow B)$ with respect to $len(A)$ and $len(B)$.

We describe in Figure 5.2.5 the thresholds values w.r.t premise and conclusion length for a fixed observation duration. To obtain this figure, we computed all values of $th(len(A), len(B), T_{obs})$ for $len(A) \in [0, 100]$ and $len(B) \in [0, 100]$ and $T_{obs} = 100$. In this figure, red points correspond to configurations where the statistical thresholds are strictly greater than the confidence higher bound. This means that such configurations cannot lead to statistically valid dependencies. Hereafter, we will discuss these cases more precisely while describing the behaviour of the statistical threshold with respect to the premise and the conclusion lengths separately.

Let us first consider the evolution of statistical threshold w.r.t premise length expressed in terms of portions of T_{obs}. We describe in Figure 5.2.6 the statistical threshold and the confidence higher bound for different constant

conclusion lengths. One can notice that the statistical threshold function decreases monotonically with premise length. Also, the statistical threshold can be higher as the confidence upper bound. More precisely, one can remark that it is the case for great premise lengths when the conclusion length is small (e.g. conclusion length equal to 1,5,10 in Figure 5.2.6) or for small premise lengths when conclusion lengths are longer (e.g. conclusion length of 80 and 90). The same observations can be done for statistical thresholds with respect to conclusion lengths as described in Figure 5.2.7 for different premise configurations. The statistical threshold is increasing and then decreasing in $[0, T_{obs}]$. It is however monotonic in $[0, x]$ such that $x < T_{obs}$ and $th(len(c), x, T_{obs}) = 1$.

Extreme configurations where premise or conclusion lengths equal to the observation duration 5.2.6 and 5.2.7 are interesting to remark. Indeed, following the statistical validity definition, every dependency having a premise or a conclusion length that is equal to observation duration will be valid as confidence value is always equal to 1 and so does the statistical threshold. We believe that this kind of dependencies is not of great interest and can be considered as trivial for mainly the following reason: a fully valid stream (i.e. having T_{obs} as length) is "correlated" with every non-zero length interval stream.

5.3 Complex Temporal Dependencies: a normal conjunctive form-like representation of temporal relations

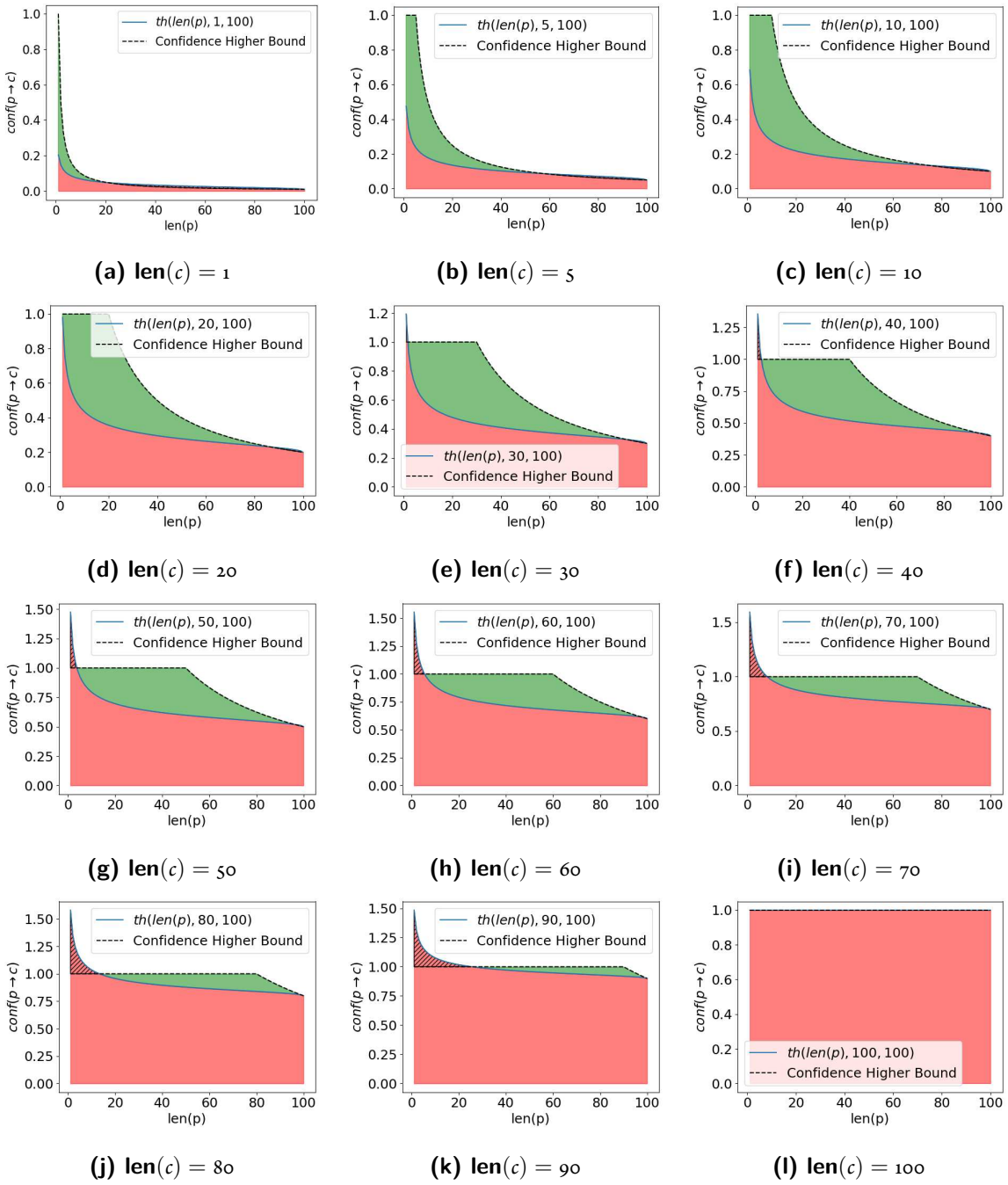
Starting from the pairwise temporal dependencies proposed in [127], we aim to model an intersection-based dependency model permitting to express temporal relations between multiple states. Our objective is to be able to describe potentially complex temporal phenomena within a particular environment. In this section, we introduce the Complex Temporal Dependencies (CTD) model. It is based on a normal conjunctive form-like representation of temporal relations. Hereafter, we first discuss the main motivations behind this modelling approach before providing a detailed description of the different operators used in our model and their semantic interpretation.

This temporal dependencies model must permit to express a large set of temporal relations with appropriate and meaningful confidence measures. In other words, a multi-state dependency model, or our pattern language, must:

- Express exclusively and without ambiguity temporal relations between several states. That is to say that the interpretation of the representation of a temporal relation must lead only to existing temporal phenomena.
- Provide confidence values that are consistent with the actual succession of states. More precisely, we want to interpret a confidence value as an "exact" probabilities estimation rather than a lower or upper bounds.
- Permit conciseness to enhance interpretability of results.

In order to make our reasoning easy to follow, we use examples of different practical configurations of motion sensors from which the user aims to extracts significant trajectories. A trajectory can be seen as a succession of valid motion states in several areas of an environment.

One first and straightforward approach is to use an aggregation of pairwise dependencies modelled as a dependency graph to extract multi-state dependencies. It is a directed attributed graph $G = (V, E)$ where $V \subset \mathcal{I}_{states}$ with $\mathcal{I}_{states}$ the set of states and $E = \{((A, B)|(A, B) \in V^2 \wedge A \neq B, (\alpha, \beta), \mathbf{conf}(A \rightarrow B^{(\alpha,\beta)}))\}$ the set of edges between element of V. An edge $\big((A, B), (\alpha, \beta), c\big)$ denotes that it exists a dependency between state A and



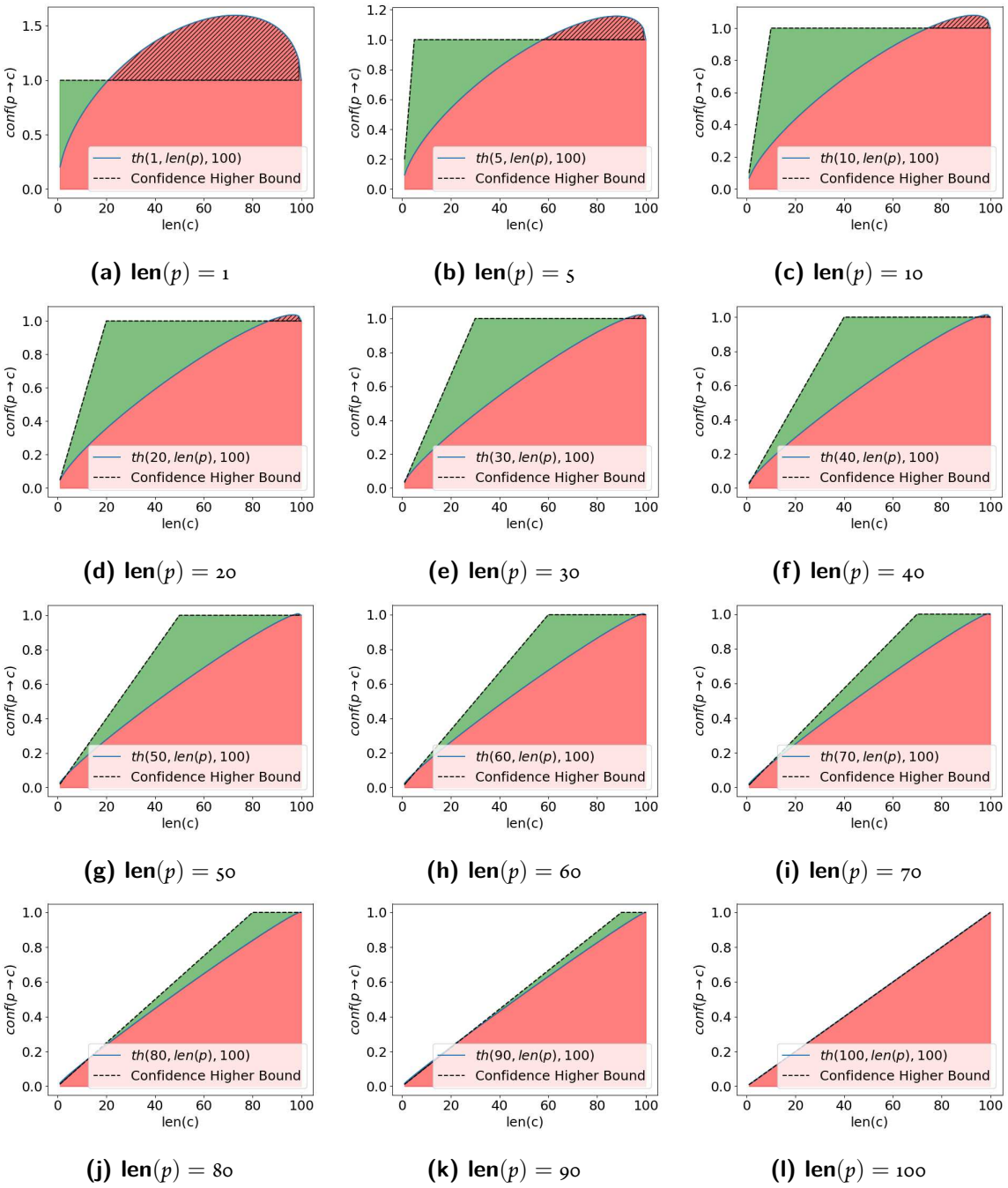
(a) $\mathbf{len}(c) = 1$ **(b)** $\mathbf{len}(c) = 5$ **(c)** $\mathbf{len}(c) = 10$

(d) $\mathbf{len}(c) = 20$ **(e)** $\mathbf{len}(c) = 30$ **(f)** $\mathbf{len}(c) = 40$

(g) $\mathbf{len}(c) = 50$ **(h)** $\mathbf{len}(c) = 60$ **(i)** $\mathbf{len}(c) = 70$

(j) $\mathbf{len}(c) = 80$ **(k)** $\mathbf{len}(c) = 90$ **(l)** $\mathbf{len}(c) = 100$

Figure 5.2.6: Statistical thresholds with respect to premise (p) length for a fixed conclusion (p) . $T_{obs} = 100$. Green areas corresponds to valid confidence values. Red areas corresponds to non-valid confidence values. Hashed areas are theoretically impossible non-valid confidence values.



(a) $\mathbf{len}(p) = 1$ **(b)** $\mathbf{len}(p) = 5$ **(c)** $\mathbf{len}(p) = 10$

(d) $\mathbf{len}(p) = 20$ **(e)** $\mathbf{len}(p) = 30$ **(f)** $\mathbf{len}(p) = 40$

(g) $\mathbf{len}(p) = 50$ **(h)** $\mathbf{len}(p) = 60$ **(i)** $\mathbf{len}(p) = 70$

(j) $\mathbf{len}(p) = 80$ **(k)** $\mathbf{len}(p) = 90$ **(l)** $\mathbf{len}(p) = 100$

Figure 5.2.7: Statistical thresholds with respect to conclusion (c) length for different premise (p) lengths. $T_{obs} = 100$. Green areas corresponds to valid confidence values. Red areas corresponds to non-valid confidence values. Hashed areas are theoretically impossible non-valid confidence values.

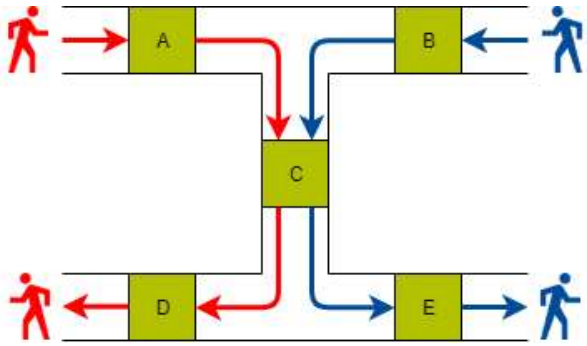
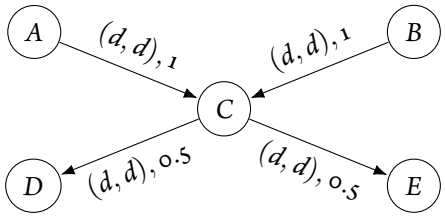


Figure 5.3.1: A set of corridors equipped with motion sensors. Red and blue arrows depict the two pedestrian trajectories occurring in this environment. Also, there is an equal number of blue and red pedestrians.

(α, β)-transformed state B, $A \rightarrow B^{(\alpha,\beta)}$ with confidence c. In this graph, one may consider all possible paths as multi-state temporal dependencies.For example, the pairwise dependency graph corresponding the configuration depicted in Figure 5.3.1 is:



It is composed by a subset of all possible dependencies that can be extracted from the former environment:

- $A \rightarrow C^{(d,d)}$ with a confidence of 1. All red pedestrians activate state A followed by state C after a delay of d time units.
- $B \rightarrow C^{(d,d)}$ with confidence of 1 with the same interpretation as above.
- $C \rightarrow D^{(d,d)}$ with a confidence of 0.5. Both red and blue pedestrians activate motion in C and only half of them (red pedestrians) activate motion in D after a delay of d.
- $C \rightarrow E^{(d,d)}$ with a confidence of 0.5 with the same interpretation as above.

This example permits to highlight two insightful observations. First, all existent paths in that graph do not necessarily refer to an existent multi-state temporal relation: e.g there is no succession *A then C then E*. The set of existent multi-state qualitative temporal relations can be seen as a subset of all paths in the dependency graph. As a consequence, modelling and discovering multiple state can not be achieved by a simple consideration of pairwise dependencies: the amount of false dependencies that can be constructed can be significantly huge. What can be also be observed in that example is that the expression of relation between more than 2 states is tied with a conditionality notion. For example, temporal relation between active motion in C and E is strictly *conditioned* by a prior valid motion in B. The conclusion of this first observation is that **modelling multiple state dependencies must permit to express conditional relation**.

The second observation that can be made relates to the confidence values of existent multi-state temporal relations. In the former example, all (red) pedestrians who activate successively states A then C, activate state D. The translation of this fact into a multi-state dependency language must be assessed with a confidence value that is consistent with the actual conditional temporal relation:

```
IF      motion in A
        AND motion in C after a duration d
THEN    motion in D after a duration d
WITH    CONFIDENCE of 1
```

In the pairwise dependency graph, one may notice that the confidence of dependency linking C to D is 0.5: only half pedestrians activating C also activate D. What is to stress here is that most insightful confidence value (i.e. 1) must take into account the historicity induced by the conditional temporal relations. This is due to the fact that temporal phenomena described as temporal relations between environment states are not necessarily Markov-like processes.

Definition 22 (Markov process)

A stochastic process whose evolution after a given time t does not depend on the evolution before t, given that the value of the process at t is fixed.[a]

[a] Markov process - Encyclopedia of Mathematics - http://www.encyclopediaofmath.org/index.php?title=Markov_process&oldid=37905 - Accessed 12/30/2019

Indeed, if one considers confidences as transition probabilities, one may want to model an environment through its states as a structure similar to a Markov chain (i.e. discrete Markov process) considering pairwise dependencies. The need for conditional temporal relations and, by extension, conditional confidences makes the Markov property irrelevant for multiple states temporal relation model. In a graph representing existent multi-states temporal relations in a given environment, each edge must have a confidence value that takes into account states validity history. As a consequence, in a *conditional* dependency graph, multiple edges may stand between two states: one for each significant states interesting multiple correlations. We will discuss later in this chapter, how to use temporal dependencies between multiple states to build such *complex & conditional* models.

The last expressivity criteria we deal within this section is that of conciseness of multi-state temporal relations representation. Let us consider the example in Figure 5.3.2 that describes a set of directed corridors. To make this example easy to follow, let us assume that there is no conditionality in this environment trajectories (i.e. entering the environment by A do not lead to a particular exit). The objective is to describe pedestrians significant trajectories in this environment concisely.

With a conditional temporal relations approach, trajectories in this environment can be described through 9 multi-state dependencies of the form:

```
IF          motion in A
AND         motion in D after a duration d
THEN        motion in E after a duration d'
WITH CONF.  0.33
```

Indeed, from each entrance (A or B or C) three exits are equivalently possible (E or F or G) and all trajectories must pass through D. What is to be remarked is that this representation approach induces possibly information redundancy. For instance, trajectories ADE, ADG and ADF have a common temporal relation prefix AD: this is given in three conditional temporal relations. The insight that can be expressed from this situation is: *If a motion in A is followed by B then a motion in E or F or G is to be observed*:

```
IF          motion in A
```

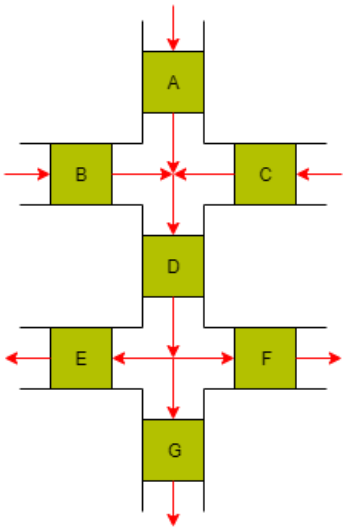


Figure 5.3.2: A set of corridors equipped with motion sensors. Red arrows indicate movement directions.

```
AND         motion in D after a duration d
THEN        (motion in E
            OR motion in G
            OR motion in F) after a duration d'
WITH CONF. 1
```

Following the same reasoning can be applied to trajectories *ADF*, *BDF* and *CDF*. As a consequence, all states describing the corridor's activity can be included in a unique temporal dependency:

```
IF          (motion in A
            OR motion in B
            OR motion in C)
AND         motion in D after a duration d
THEN        (motion in E
            OR motion in G
            OR motion in F) after a duration d'
WITH CONF. 1
```

We believe that this type of temporal dependencies is more expressive for mainly two reasons. First, information redundancy is reduced. In the former example, one unique dependency permits to express nine simple conditional temporal relations. We consider that this is useful to provide a rapidly comprehensible insight on complex temporal phenomena that sums up several behaviours (here trajectories) in a unique dependency. Secondly, the aggregation of several conditional temporal relations can provide valuable information. In the former example, each simple succession of states *ADE*, *ADG* and *ADF* tells simply that the succession of states *AD* leads respectively to *E*, *G* and *F* with each a confidence value of 0.33. The disjunctive relation *AD*(*E* or *G* or *F*) with confidence of 1 tells another story: if a pedestrian passes through *A* then *D*, he will necessarily leave the corridor area from *E*, *G* or *F*. This insight is not guaranteed if one takes into account the three conditional relations mentioned above.

From a more discovery process perspective, it is to notice that sometimes a temporal relation linking a disjunction of states to another state can be *interesting* while each of its *equivalent* conjunctive dependencies are not.

One can convince himself of the latter statement considering a counting support interestingness measure. Let a sequence AB have a support of 0.1 and AC a support of 0.15. The execution of a frequent sequential pattern mining algorithm with a minimum support of 0.2 will not provide AB nor AC as results. However, a sequence $A(B \text{ or } C)$ is of interest as its support equals 0.25. The last example can be transposed to the intersection-based confidence assessment. We believe that disjunctive relationships can provide additional temporal insights that cannot be discovered with simple conjunctions of states.

To conclude this section, we recall the three main specifications (or constraints) of our multi-state dependency model:

- The multi-state dependency model must be able to express conditional temporal relations.
- Confidence values must be consistent with the conditionality of temporal relations.
- The multi-state dependency model must be able to express disjunctive temporal relations.

In the remainder of this chapter, we will introduce the proposed Complex Temporal Dependencies (CTD) model that implements the former conditions. In the next subsection, we describe the three operators used by our model to express temporal relationships between states.

5.3.1 CTD's Operators

Inference

As pairwise dependencies, the Complex Temporal Dependencies uses an inference operator, noted $\rightarrow$. A dependency between two state streams A and B is noted $A \rightarrow B$. The left part of a Complex Temporal Dependency is also called *premise* and its right part the *conclusion*. The inference operator's main role is to associate a dependency to the intersection-based confidence value. We recall hereafter the formula:

$$\mathbf{conf}(A \rightarrow B) = \frac{\mathbf{len}(A \cap B)}{\mathbf{len}(A)}$$

With a slightly different point of view, one can observe that the inference operator induces a stream's intersection operation that defines the confidence parameters: it is the intersection between the dependency premise and its conclusion. As discussed in Section 3.4.3, the intersection of two state streams is a state stream whose intervals have specific semantics: an interval from $A \cap B$ denotes that both A and B are active. In other words, the resulting state stream contains all portion of time where a dependency is verified: it is the *representative* stream of a dependency.

Definition 23 (Temporal Dependency representative stream)
Let A and B be two state streams and $A \rightarrow B$ a temporal dependency. A state stream representative of $A \rightarrow B$, noted $(A \rightarrow B)_r$ is a state stream obtained via the intersection of its premise and conclusion:

$$(A \rightarrow B)_r = A \cap B$$

Considering state streams representatives permits to consider temporal dependencies between state streams as state streams themselves. The usefulness of this concept will be discussed later.

We recall that, semantically, the inference operator expresses a simultaneity concept as it induces a stream intersection. $A \rightarrow B$ with a confidence c means that *A occurs with B with confidence c*. The delayed temporal relations are expressed via (α, β)-temporal transformations.

Conjunctions

As discussed in the motivations section, a multi-state temporal dependency must express conditional temporal relationships. From a practical point of view, temporal conditionality can be seen as a selection process or a temporal join. For instance, let us consider the following conditional temporal dependency:

```
IF      motion in A
        AND motion in B after a duration d
THEN    motion in C after a duration d
WITH    CONFIDENCE of c
```

This relation express the following relationships: C is correlated to intervals of A that are followed by B after a duration d with a confidence c. Therefore, the premise of this temporal dependency is composed by portions of time where intervals of stream A are co-occurring with intervals of the stream $B^{(d,d)}$. The premise is then computed via a stream intersection describing a conjunction of states A and temporally transformed B. A conjunctive relation in a multi-state temporal dependency is noted $\wedge$. A temporal dependency can be then be defined as a correlation between conjunctions of state streams:

$$A_1 \wedge A_2 \wedge \ldots \wedge A_k \rightarrow A_{k+1} \wedge A_{k+2} \wedge \ldots \wedge A_n$$

where both premises and conclusion can be conjunctions of streams. The quantitative temporal information in such dependencies must be expressed w.r.t a reference stream in order to make it easier to interpret. In the reminder of this work, we will consider that at least one state in a dependency premise is not temporally transformed in order to constitute a temporal reference. In addition to that, conjunction of streams will be noted following the temporal order as a writing convention to facilitate dependencies reading. Thus, a conjunctive temporal dependency can the formalized as:

$$R = A_1^{(o,o)} \wedge A_2^{(a_2,\beta_2)} \wedge \ldots \wedge A_k^{(a_k,\beta_k)} \rightarrow A_{k+1}^{(a_{k+1},\beta_{k+1})} \wedge A_{k+2}^{(a_{k+2},\beta_{k+2})} \wedge \ldots \wedge A_n^{(a_n,\beta_n)}$$

with $\forall a_i, a_i \leq a_{i+1}$ and $\forall \beta_i, \beta_i \leq \beta_{i+1}$. A_1 is the temporal reference of this dependency.

Since state streams conjunctions are state streams, the confidence value of conjunctive dependencies can be simply obtained after the computation of the intersection stream of the premise and the conclusion:

$$\mathbf{conf}(R) = \frac{\mathbf{len}(P \cap C)}{\mathbf{len}(P)}$$

with $P = A_1^{(o,o)} \cap A_2^{(a_2,\beta_2)} \cap \ldots \cap A_k^{(a_k,\beta_k)}$ and $C = A_{k+1}^{(a_{k+1},\beta_{k+1})} \cap A_{k+2}^{(a_{k+2},\beta_{k+2})} \cap \ldots \cap A_n^{(a_n,\beta_n)}$.

It is to notice that both the inference and the conjunction operators express simultaneity (or succession in the case of using temporal transformation). The main difference between these two operators is that the inference operator indicate the set of states belonging to the premise and the set of states belonging to the conclusion. As a consequence, the obtained confidence value is determined by the position of the inference operator. Indeed, the value of $\mathbf{len}(P \cap C)$ is the same wherever the inference operator is placed. For example, if we consider:

$$R' = A_1^{(o,o)} \rightarrow A_2^{(a_2,\beta_2)} \wedge \ldots \wedge A_n^{(a_n,\beta_n)}$$

The corresponding confidence value can be obtained with:

$$\mathbf{conf}(R') = \frac{\mathbf{len}(P \cap C)}{\mathbf{len}(A_1^{(o,o)})}$$

since $\mathbf{len}(P \cap C) = \mathbf{len}(P' \cap C')$ with $P' = A_1^{(o,o)}$ and $C' = A_2^{(\alpha_2,\beta_2)} \cap ... \cap A_n^{(\alpha_n,\beta_n)}$. These different confidence values show also that the interpretation of these dependencies is specified by the inference operator position. For instance, the interpretations of R and R' are the following:

- R: If A_1 is *followed* by A_2 after a delay given by (α_2, β_2) and by A_3 after a delay given by (α_3, β_3) ... and by A_k after a delay given by (α_k, β_k) **then** follow A_{k+1} after a delay given by $(\alpha_{k+1}, \beta_{k+1})$ ... and A_n after a delay given by $(\alpha_{k+1}, \beta_{k+1})$ **with confidence conf**(R).
- R': If A_1 is active **then** follow A_2 after a delay given by (α_2, β_2) ... and A_n after a delay given by (α_n, β_n) **with confidence conf**(R').

Disjunctions

With conjunctive and inference operators, it is possible to express conditional dependencies between premises and conclusions formed by a conjunction of delayed states. As discussed is the motivation section, this model can use disjunctive relationships to enhance the interpretation of temporal dependencies through conciseness by performing factorization of similar relationships. This also permits to build summaries providing general overviews of complex temporal phenomena.

From a general perspective, our expressivity constraint can be translated into a conjunction of disjunctive relationships. Indeed, what is to be achieved by a disjunctive relation is the ability to group conjunctive dependencies that share partially one or various quantitative states. Such temporal dependency can be noted:

$$D_o \wedge ... \wedge D_k \rightarrow D_{k+1} \wedge ... \wedge D_n$$

with D_i a disjunction of states of $\mathcal{S} = \{A_o, A_1, ...A_n\}$. A disjunction of two states A and B is noted $A \vee B$. The corresponding interval stream is composed by all portion of time where either state A or B are active. Therefore, it is obtained via the union $A \cup B$. At this stage a temporal dependency can be formalized as a correlation between two conjunctive normal forms:

$$\bigwedge_i \bigvee_j A_{ij}^{(\alpha_{ij},\beta_{ij})} \rightarrow \bigwedge_x \bigvee_y A_{xy}^{(\alpha_{xy},\beta_{xy})}$$

with $A_{ij} \in \mathcal{S}, A_{xy} \in \mathcal{S}$ and $\exists i, j$ such that $(\alpha_{ij}, \beta_{ij}) = (o, o)$. This definition on temporal dependencies permits to express temporal relations as in Figure 5.3.2, where there are disjunctive relationships between simple streams. However, we consider that this is not sufficient to express more complex temporal relationships. Let us illustrate this problem with a simple corridor example depicted in Figure 5.3.3. In this environment, all pedestrians perform a trajectory going from A to E in a similar manner (delay between A and E is always the same). Two equivalent groups (in terms of numbers and behaviours) are to be distinguished: the first group chose to pass through the bottom corridor that is sensed by one motion sensor B, the second group choose the top corridor sensed by two motion sensors C and D. Motion activity in that environment can represented with the following:

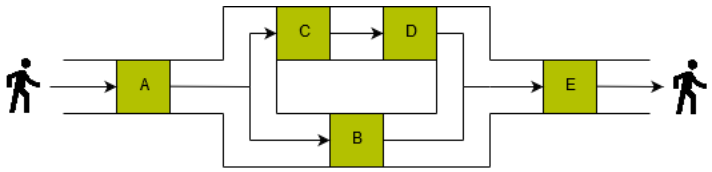
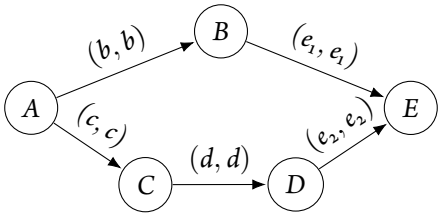


Figure 5.3.3: Pedestrians pass through the top or the bottom corridors. All pedestrians have the same speed and take the same time to go from A to E regardless of the chosen trajectory.

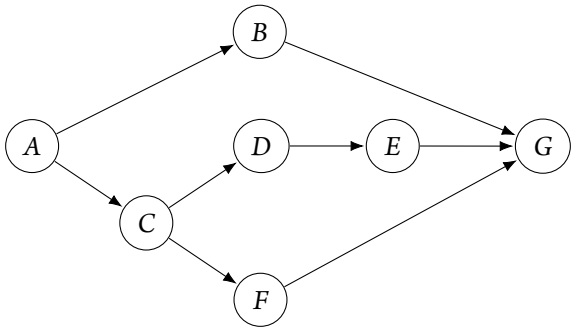


Expressing trajectories occurring in this environment with temporal dependencies using simple disjunctive relations does not permit to capture the overall activity. Indeed, if one considers such disjunctive relations, the obtained dependencies are $A \wedge (B^{(b,b)} \vee C^{(c,c)}) \rightarrow E^{(b+e_1,b+e_1)}$ and $A \wedge \left(B^{(b,b)} \vee D^{(c+d,c+d)}\right) \rightarrow E^{(b+e_1,b+e_1)}$. We consider that a dependency of the form $A \wedge (B^{(b,b)} \vee (C^{(c,c)} \wedge D^{(c+d,c+d)}) \rightarrow E^{(b+e_1,b+e_1)}$ is more insightful. Indeed, semantically it reports that paths ABE and $ACDE$ are equivalent in quantitative terms $(b+e_1 = c+d+e_2)$. This permit to induce that the bottom and the above trajectories occur in the same way while considering simple disjunctive relations does not provide such information. From a formalization point of view, the former example complexity needs temporal dependencies defined as normal conjunctive forms where atomic formulas are conjunctive normal forms of states in $\mathcal{S}$. Formally, a dependency able to express this kind of temporal relation can be defined as:

$$\bigwedge_i \bigvee_j \bigwedge_k \bigvee_l A_{ijkl}^{(a_{ijkl},\beta_{ijkl})} \rightarrow \bigwedge_x \bigvee_y \bigwedge_z \bigvee_t A_{xyzt}^{(a_{xyzt},\beta_{xyzt})}$$

.

This reasoning about disjunctive relations can be extended to higher levels of complexity with more disjunctives levels. For example, the following temporal configuration is considered more complex than the former example.



The expression of this temporal configuration needs three levels of normal conjunctive forms. It can be written as the following (without temporal transformation for more clarity):

$$A \wedge \Big(B \vee \big(C \wedge \big(F \vee (D \wedge E)\big)\big)\Big) \rightarrow G$$

We define the level of dependency complexity as the number of levels of normal conjunctive forms that are

necessary to express a complex temporal relation fully. For instance, the temporal pattern corresponding to Figure 5.3.2 can be expressed with the first level of complexity, temporal patterns corresponding to Figure 5.3.3 can be expressed with a second complexity level dependency and the last example with a third level complexity dependency.

Complex Temporal Dependencies definition

Considering the three mentioned operators in this section and our discussion about temporal relations complexity, we provide hereafter the formal definition of Complex Temporal Dependencies.

Definition 24 (Complex Temporal Dependency)

Let $\mathcal{S} = \{A_1, A_2, ..., A_n\}$ be a set of state streams. A Complex Temporal Dependency over $\mathcal{S}$ is defined with:

$$\bigwedge_{i_1}^{l_o} \bigvee_{i_2}^{l_{i_1}} \dots \bigwedge_{i_k}^{l_{i_{k-1}}} \bigvee_{i_{k+1}}^{l_{i_k}} S_{i_1 \dots i_{k+1}}^{(a_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}})} \rightarrow \bigwedge_{j_1}^{l_1} \bigvee_{j_2}^{l_{j_1}} \dots \bigwedge_{j_n}^{l_{j_{n-1}}} \bigvee_{j_{n+1}}^{l_{j_n}} S_{j_1 \dots j_{n+1}}^{(a_{j_1 \dots j_{n+1}}, \beta_{j_1 \dots j_{n+1}})}$$

where $\forall k, S_k \in \mathcal{S}$ and $\exists i_1 \dots i_{k+1}$ such that $(a_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}}) = (0, 0)$

The confidence value of a Complex Temporal Dependency R is given by:

$$conf(R) = \frac{\mathbf{len}(P \cap C)}{\mathbf{len}(P)}$$

where P and C are respectively the stream resulting of the computation of the "recursive" normal conjunctive forms of the dependency's premise and conclusion. This operation is done with the computation of streams corresponding to normal conjunctive forms from the highest level of complexity to the lower. The stream corresponding to normal conjunctive forms are obtained by (1) computing temporal transformations (2) computing disjunctions via interval streams unions, then (3) computing conjunctions via interval streams intersection. Let us consider the following dependency for a running example:

$$A \wedge \Big(B \vee \big(C \wedge \big(F \vee (D \wedge E) \big) \big) \Big) \rightarrow G$$

If we consider the states in this dependency are transformed streams (temporal transformations are already computed), the confidence of this dependency is given by computing successively:

- Intersection $S_{DE} = D \cap E$
- Union $S_{DEF} = S_{DE} \cup F$
- Intersection $S_{DEFC} = S_{DEF} \cap C$
- Union $S_{DEFCB} = S_{DEFC} \cup B$
- Intersection $P = S_{DEFCB} \cap A$. The result at this stage corresponds to the premise stream. The conclusion in this example is constituted by a unique stream G
- Confidence: $\frac{\mathbf{len}(P \cap G)}{\mathbf{len}(P)}$

From a complexity point of view, all necessary stream operations for a confidence calculation are linear w.r.t number of intervals in the streams.

Sub-dependencies

A sub-dependency of a dependency R is a dependency involving a subset of states of R such that it preserves the conjunctive and disjunctive relations between these states. For instance, if A and B are linked with a conjunctive relation in R, a sub-dependency of R involving A and B must contain a conjunctive relation between A and B.

Definition 25 (Complex Temporal Sub-Dependencies)

Let $\mathcal{S} = \{A_1, A_2, ..., A_n\}$ be a set of state streams and R a Complex Temporal Dependency over $\mathcal{S}$:

$$R = \bigwedge_{i_1}^{l_0} \bigvee_{i_2}^{l_{i_1}} \dots \bigwedge_{i_k}^{l_{i_{k-1}}} \bigvee_{i_{k+1}}^{l_{i_k}} S_{i_1 \dots i_{k+1}}^{(a_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}})} \rightarrow \bigwedge_{j_1}^{l_1} \bigvee_{j_2}^{l_{j_1}} \dots \bigwedge_{j_n}^{l_{j_{n-1}}} \bigvee_{j_{n+1}}^{l_{j_n}} S_{j_1 \dots j_{n+1}}^{(a_{j_1 \dots j_{n+1}}, \beta_{j_1 \dots j_{n+1}})}$$

A sub-dependency R' of R is a complex temporal dependency defined as:

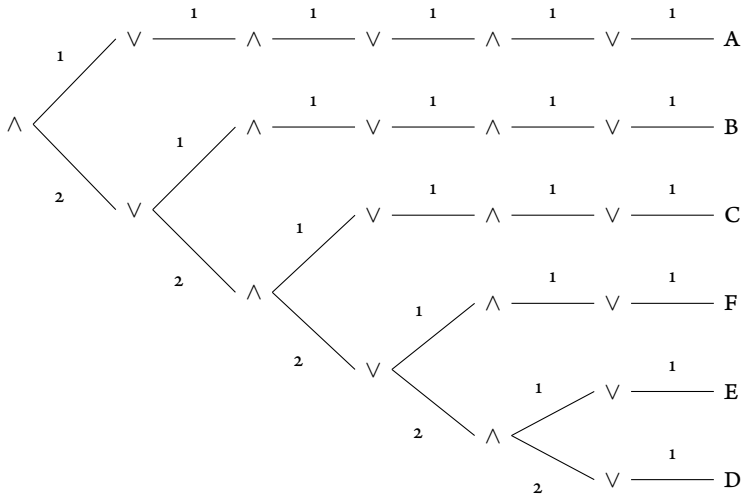
$$R' = \bigwedge_{i_1}^{L_0} \bigvee_{i_2}^{L_{i_1}} \dots \bigwedge_{i_k}^{L_{i_{k-1}}} \bigvee_{i_{k+1}}^{L_{i_k}} S_{i_1 \dots i_{k+1}}^{(a_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}})} \rightarrow \bigwedge_{i_1}^{L_1} \bigvee_{j_2}^{L_{j_1}} \dots \bigwedge_{j_n}^{L_{j_{n-1}}} \bigvee_{j_{n+1}}^{L_{j_n}} S_{j_1 \dots j_{n+1}}^{(a_{j_1 \dots j_{n+1}}, \beta_{j_1 \dots j_{n+1}})}$$

where $\forall i, j\, i \geq 1$ and $j \geq 1$; $\forall x = y$ and $S_x \in R, S_y \in R', S_x = S_y$; and L_x is a subset of $\big(\{x_1, x_2, ...\}, \leq\big)$ the set of totally ordered naturals such that $\forall x \in [0, l_x], x \in \{x_1, x_2, ...\}$.

In order to make this definition clearer, let us consider the following example.

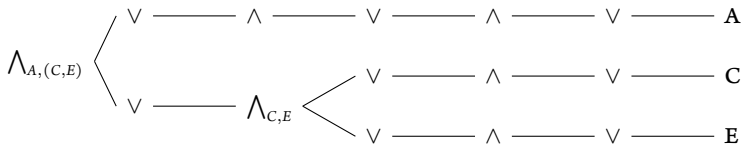
$$R = A \wedge \Big(B \vee \big(C \wedge \big(F \vee (D \wedge E)\big)\big)\Big) \rightarrow G$$

The logical relations within the premise of this dependencies can be represented as a logical tree where depth levels correspond to conjunctive or disjunctive operators. As the CTD model is defined as a recursive conjunctive normal form, the root of the logical tree is a conjunction and children of a conjunctive node are disjunctions and vice versa. The logical tree corresponding to our example is the following.



Edges' labels in this tree corresponds to the used indexes in the recursive conjunctive normal form formula. For instance, state A corresponds to S_{111111} and D to S_{111112}. This representation is useful to easily specify the logical relation between two states in a complex recursive conjunctive normal form. It is given by the first common

ancestor of two states. For example, A and B are related with a conjunction, B and F with a disjunctive relation and C and E with a conjunction. Following the same reasoning, one can also specify logical expression between multiple states. For example, the logical relationship between A, C and D can be obtained with the following tree portion that is built preserving the path to the root of each state:



This tree portion is to be interpreted in terms of recursive normal conjunctive forms with:

$$A \wedge (C \wedge E)$$

as the common ancestor of C and E is a conjunction, and the common ancestor of A and $C \wedge E$ is also a conjunction. This relation can be called a logical sub-expression of the premise of R. In the complex temporal sub-dependency, the use of a subset of totally ordered naturals as indexes for conjunctive and disjunctive permits to maintain logical "paths" and, thus, extract the corresponding tree portion linking A, C and E.

A sub-dependency of a R is, then, defined as a complex temporal dependency between a logical sub-expression of the premise of R and a logical sub-expression of its conclusion. For instance, $A \wedge (C \wedge E) \rightarrow G$ is a sub-dependency of R.

Confidence monotonicity

One useful characteristic of interestingness measures is that of monotonicity. For instance, the downward monotonicity of the counting support measure, stating that any sub-sequence of a frequent sequential pattern is frequent, permits to use the Apriori pruning mechanism in sequential pattern mining. In this subsection, we study the monotonicity of the intersection-based confidence measure w.r.t premise and conclusion extension.

A complex temporal dependency extension consists of adding a conjunctive or disjunctive logical relation to either the premise or the conclusion. For instance, $A \wedge B \rightarrow C$ is a premise conjunction extension of $A \wedge C$ and $A \rightarrow B \vee C$ is a conclusion disjunctive extension of $A \rightarrow B$. Follows, the confidence (non)-monotonicity properties w.r.t the premise and conclusion extensions.

Property 2 (Premise conjunctive extension effect on confidence)
Let R and R' be CTDs such that R' is obtained by a premise conjunctive extension of R. The confidence of R' can be higher, lower or equal to the confidence of R.

Proof. The confidence of $A \rightarrow C$ and $A \wedge B \rightarrow C$ are given by

$$\mathbf{conf}(A \rightarrow C) = \frac{\mathbf{len}(A \cap C)}{\mathbf{len}(A)}$$
$$\mathbf{conf}(A \wedge B \rightarrow C) = \frac{\mathbf{len}(A \cap C \cap B)}{\mathbf{len}(A \cap B)}$$

Since

$$\mathbf{len}(A \cap C) \geq \mathbf{len}(A \cap C \cap B)$$
$$\mathbf{len}(A) \geq \mathbf{len}(A \cap B)$$

Then, $\mathbf{conf}(A \rightarrow C)$ can be higher, lower or equal to $\mathbf{conf}(A \wedge B \rightarrow C)$. □

Property 3 (Premise disjunctive extension effect on confidence)
Let R and R' be CTDs such that R' is obtained by a premise disjunctive extension of R. The confidence of R' can be higher, lower or equal to the confidence of R.

Proof. The confidence of $A \rightarrow C$ and $A \wedge B \rightarrow C$ are given by

$$\mathbf{conf}(A \rightarrow C) = \frac{\mathbf{len}(A \cap C)}{\mathbf{len}(A)}$$
$$\mathbf{conf}(A \vee B \rightarrow C) = \frac{\mathbf{len}\big((A \cup B) \cap C\big)}{\mathbf{len}(A \cup B)}$$

Since

$$\mathbf{len}(A \cap C) \leq \mathbf{len}\big((A \cup B) \cap C\big)$$
$$\mathbf{len}(A) \leq \mathbf{len}(A \cup B)$$

Then, $\mathbf{conf}(A \rightarrow C)$ can be higher, lower or equal to $\mathbf{conf}(A \wedge B \rightarrow C)$. □

Property 4 (Conclusion conjunctive extension effect on confidence)
Let R and R' be CTDs such that R' is obtained by a conclusion conjunctive extension of R. The confidence of R' is lower or equal to the confidence of R.

Proof. The confidence of $A \rightarrow C$ and $A \rightarrow B \wedge C$ are given by

$$\mathbf{conf}(A \rightarrow C) = \frac{\mathbf{len}(A \cap C)}{\mathbf{len}(A)}$$
$$\mathbf{conf}(A \rightarrow B \wedge C) = \frac{\mathbf{len}\big(A \cap (B \cap C)\big)}{\mathbf{len}(A)}$$

Since

$$\mathbf{len}(A \cap C) \geq \mathbf{len}\big(A \cap (B \cap C)\big)$$

then,

$$\mathbf{conf}(A \rightarrow C) \geq \mathbf{conf}(A \rightarrow B \wedge C)$$

□

Property 5 (Conclusion disjunctive extension effect on confidence)
Let R and R' be CTDs such that R' is obtained by a conclusion disjunctive extension of R. The confidence of R' is greater or equal to the confidence of R.

Proof. The confidence of $A \rightarrow C$ and $A \rightarrow B \vee C$ are given by

$$\mathbf{conf}(A \rightarrow C) = \frac{\mathbf{len}(A \cap C)}{\mathbf{len}(A)}$$
$$\mathbf{conf}(A \rightarrow B \wedge C) = \frac{\mathbf{len}\big(A \cap (B \cup C)\big)}{\mathbf{len}(A)}$$

Since

$$\mathbf{len}(A \cap C) \leq \mathbf{len}\big(A \cap (B \cup C)\big)$$

then,

$$\mathbf{conf}(A \rightarrow C) \leq \mathbf{conf}(A \rightarrow B \wedge C)$$

□

These four properties show that the intersection-based confidence is monotonic w.r.t conclusion extension and non-monotonic for premise extensions. These properties can be extended to temporal transformations and proofed with similar reasoning.

Conclusion

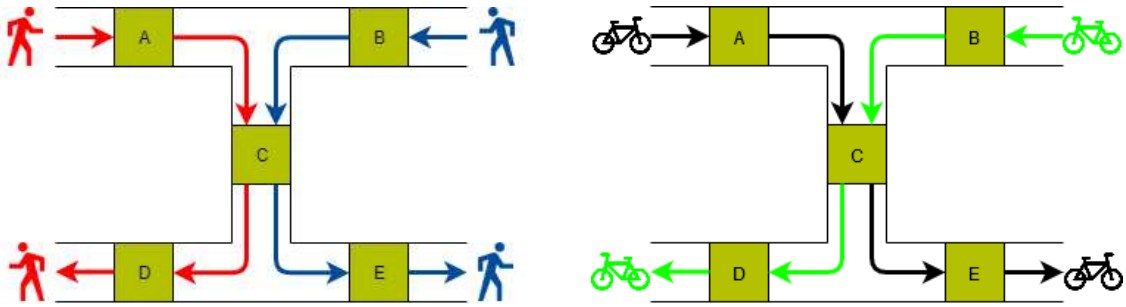
In this section, we motivated and defined the Complex Temporal Dependency model. It permits to express and build complex dependencies, assessed with an intersection-based confidence measure. It is based on the inclusion of conjunctive and disjunctive relations and defined as an inference relation between recursive normal conjunctive forms. In the next section, we will discuss how Complex Temporal Dependencies can be used to build a temporal model describing temporal phenomena occurring in a set of interval streams.

5.4 Modelling environment activities through CTDs

In Section 5.3, we argued that modelling complex temporal relationships between states can not be straightforwardly be achieved with simple pairwise dependency graph as the Markov property cannot always be verified. Indeed, the extraction of complex temporal dependencies from a dependency graph needs additional computations to ensure the existence of correlations and calculate the appropriate confidence measures.

In this section, we discuss how to build predictive models for complex temporal dependencies that are similar to Markov chains, i.e. taking the form of a graph structure. Indeed, this can be useful to model complex temporal phenomena occurring in an environment with a useful model that can be used straightforwardly for several application as:

- Perform temporal predictions: with conditional relationships that take into account the historicity and the corresponding confidence measures, it is possible to predict the expected outcomes with great confidence precision.
- Monitoring an environment through a conditional graph may allow performing concept drift analysis to detect behavioural changes in an environment and trigger a new temporal dependency discovery process.
- Perform precise, realistic simulations based on conditional temporal relations.



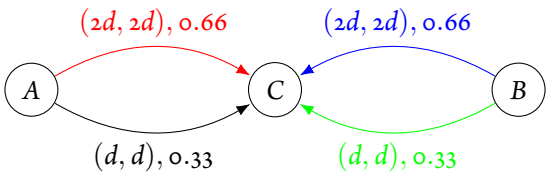
(a) Pedestrian trajectories **(b)** Bicycles trajectories

Figure 5.4.1: A set of corridors where pedestrians and bicycles have different typical trajectories

In order to illustrate this modelling problem, we consider the corridor environment we used in Figure 5.3.1 with a slightly more complex temporal configuration where both pedestrians and bicycles behave differently as depicted in Figure 5.4.1. For simplicity purposes, say that our system follows the following rules:

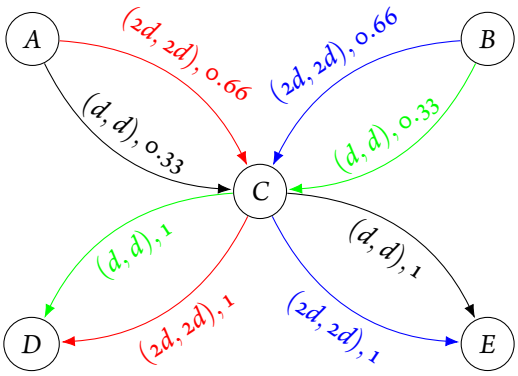
- During the observation duration, the same number of pedestrians and bicycles crossed the environment.
- Bicycles had a speed twice higher than pedestrians. As a consequence, intervals induced by bicycles have a duration that is twice lesser than those generated by pedestrians.

We want to model such an environment through its motion states in a way reflecting the quantitative and qualitative variety of trajectories occurring in it. Let us consider firstly, the pairs of states corresponding to the starting of each trajectory. The following graph is obtained:



This graph is obtained with the following pairwise dependencies: $A \rightarrow C^{(d,d)}$, $A \rightarrow C^{(2d,2d)}$, $B \rightarrow C^{(d,d)}$ and $A \rightarrow B^{(2d,2d)}$. The red and blue arrow correspond to pedestrian behaviours while the green and the black one to bicycles'. It is to notice that time delays induced by pedestrians are twice greater than those generated by bicycles. Both can be considered as significant and must appear in the description graph as they constitute two different behaviours. The second thing to notice is the confidence difference. Pedestrians generate valid intervals having twice the duration of those generated by bicycles. The direct consequence of this is that the intersection length of, for example, $A \cap C^{(d,d)}$ equals twice the intersection length of $A \cap C^{(2d,2d)}$. As a consequence $2 * conf(A \rightarrow C^{(d,d)}) = conf(A \rightarrow C^{(2d,2d)})$ even if the number of pedestrians and bicycles is equal. This example permits to highlight the fact that confidence values are not meant to be interpreted in terms of occurrence probabilities, i.e. with an actor counting reasoning, but in terms of the proportion of active premise length.

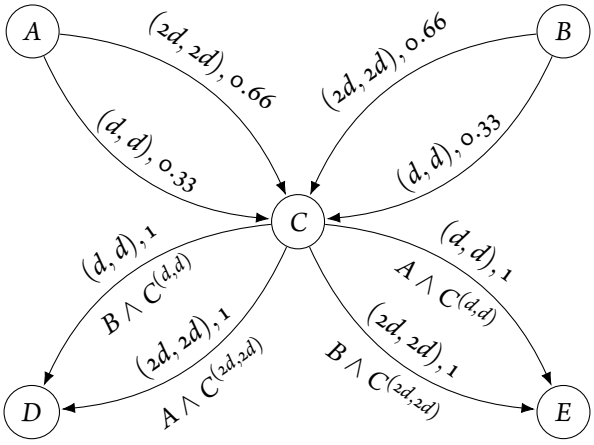
From the precedent partial graph, the objective is to extend it to obtain a quantitative description of each trajectory with corresponding confidence measures. The resulting graph must be the following:



For instance, the red graph path depicting the red pedestrian trajectory is to be interpreted in the following way.

- State A is correlated with state C with temporal transformation $(2d, 2d)$ and confidence 0.66.
- State C is correlated with state D with temporal transformation $(2d, 2d)$ and confidence 1 if the first temporal transformation is verified.

We propose to materialize the conditional relationship (represented using colours in the former graph) using conjunctive expressions labels on edges. With the former example, the red edge linking C to D will have an additional attribute $A \wedge C^{(2d,2d)}$ denoting that the red edge is valid if and only if the interval of C is "involved" in a valid temporal relation $A \rightarrow C^{(2d,2d)}$. Following the same logic, the obtained temporal dependency graph will be:



This type of conditional temporal models can be useful to provide answers to queries such as:

- What statistically valid temporal relations can be expected if the current state is C?
- What statistically valid temporal relations can be expected if the current state is C, given a previous occurrence of state A?
- What statistically valid temporal relations can be expected if the current state is C, given an occurrence of a state A, $2d$ times units before?

For instance, the first query answer is: a state C can be followed by state D after a duration (d, d) with confidence 1 if exists an occurrence of state B before a (d, d) duration **OR** by state D after a duration $(2d, 2d)$ with confidence 1 if exists an occurrence of state C before a duration $(2d, 2d)$ **OR** ...

We believe that this type of insights provide explainable predictive information as it takes into account the context. For instance, a state C will be followed by a state D (and not a state E) after a given duration because there was a valid state B before. In the next chapter, we will propose an algorithm devised to discover statistically significant Complex Temporal Dependencies permitting to build such conditional models.

5.5 Discussion and conclusion

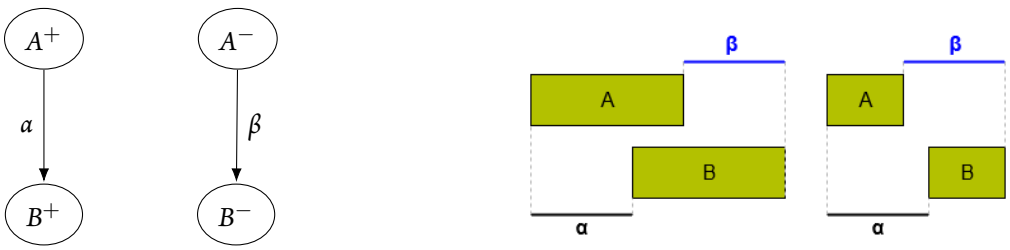
In this chapter, we introduced the Complex Temporal Dependency model. It permits to express a broad set of complex temporal relations and configurations between multiple environment states.

Nevertheless, we believe that this model can be improved on its temporal aspect to unsure a complete non-ambiguity of temporal descriptions. Indeed, modelling time lags and delays with an (α, β)-temporal transformation does not provide a theoretical guarantee for non-ambiguity. Indeed, as discussed in Chapter 4, an interval, without regards to its semantics, is defined with two parameters (i.e. two endpoints, or an endpoint and a duration). The temporal relation between two intervals A and B can be represented graphically and relationally as follows:

where A^+, B^+ denote the first endpoints and A^-, B^- the second endpoints. What can be noticed is that any variation in a temporal parameter induces the variation of one other which insures non-ambiguity. This can be formalized by the following simple equation:

$$\alpha + \theta = \gamma + \beta$$

Thus, the complete characterization of a temporal relation between two intervals can be achieved if three of the four variables of the former equation can be discovered. With (α, β)-temporal transformation, the available temporal information is α and β in the former illustration without any information about intervals durations (γ and θ). The obtained temporal patterns can be represented as follows:



As it can be noticed in the right illustration, the specification of α and β does not permit to obtain fully non-ambiguous temporal pattern permitting to associate it with a unique Allen temporal qualitative relation. The extension of the CTD model to take into account this temporal information would permit to obtain precise temporal quantitative models. Unfortunately, to the best of our knowledge (cf Chapter 4), there is no available pattern

discovery algorithm permitting to fully specify necessary temporal insights to build non-ambiguous quantitative temporal patterns (this problem share similarities with Temporal Constraints Satisfaction Problems TCSP [82]). In chapter 7, we propose an approach that can be extended in that direction. In this work, we will consider the proposed Complex Temporal Dependency as sufficient and propose algorithms for their discovery.

6

Discovering Complex Temporal Dependencies

Contents

In this chapter, we introduce the CTD-Miner algorithm. It is devised to discover statistically significant complex temporal dependencies in order to build a conditional temporal model. The complex temporal dependencies model defines an infinite search space. Its exploration with a brute force approach is nor efficient nor feasible for even small state streams set. Therefore, in the first section, we discuss and motivate several constraints and assumptions we made to restraint our search and result space. In the second section, we discuss the applicability of two properties, Apriori and patterns closure, to our complex temporal dependencies model. These two properties will be used as conciseness interestingness constraints devised to reduce information redundancy in the result set provided by our approach as well as enhancing the efficiency of the discovery process. In the third section, we will describe CTD-Miner before concluding this chapter.

6.1 Problem definition

The complex temporal dependency model (CTD) introduced in the last chapter defines an infinite search space. One can convince himself of this statement noticing that a dependency $A \rightarrow A$ is statistically valid (with a confidence of 1). By extension, all complex temporal dependencies taking the form of:

$$A \wedge A \wedge .... \wedge A \rightarrow A \wedge A .... \wedge A$$

are likely to be significant. The set of such dependencies is infinite (the premise and conjunction streams can be composed of an infinity of conjunctive and disjunctive relations). Besides, we consider that this type of complex temporal dependencies is not of interest as they do not provide any non-trivial information.

Another dimension that is not bounded in our dependency model is the temporal dimension. A temporal transformation can have both α and β values in $\mathbb{N}$. This can be nuanced considering a finite observation duration T_{obs}. Indeed, one may consider that α and β can be bounded in $[0, T_{obs}]$. The temporal transformations search is then in $\Theta(|T_{obs}|^2)$ for each stream involved in a temporal dependency.

Hereafter, we first introduce and motivate a set of hypothesises, assumptions and constraints permitting to define a finite search sub-space corresponding to a useful form of dependencies permitting to build conditional temporal models as described in Chapter 5. Secondly, we formalise our problem with regards to this set of constraints and assumptions and conclude the section with a search space study.

6.1.1 Constraints on Complex Temporal Dependencies

Temporal Constraint

In order to propose an efficient CTD discovery algorithm, the first problem one may address concerns the temporal search space. As described before, considering a finite observation time T_{obs}, each state involved in a dependency may support a number of temporal transformation in $\Theta(|T_{obs}|^2)$ since every (α, β) (with $0 \leq \alpha \leq T_{obs}$ $0 \leq \beta \leq T_{obs}$) combination must be tested. As discussed in Chapter 4, temporal patterns discovery algorithm often use temporal constraints to reduce search spaces and enhance exploration efficiency. The commonly used assumption behind the use of such constraints is the following:

> In order to be interesting, two temporal facts must occur close enough in time. [101]

This assumption was used, for example, in the work of Mannila and Toivonen [101] about episode mining to motivate the use of temporal windows. When it comes to interval streams dependency mining, authors in [127]

used a range $\Delta = [min, max]$ as bounds for temporal transformation values for pairwise dependency. In this work, we will consider this same temporal constraint.

The question that is raised for multi-state temporal dependencies is how to apply this temporal constraint. Generally speaking, there are two main ways to consider a temporal constraint when multiple items are considered (cf Chapter 4):

- Window-like constraint. The temporal constraint is applied to the overall temporal pattern.
- Maximum gap-like constraint. The temporal constraint is considered for successive temporal facts.

For our discovery algorithm, we chose to apply the temporal constraint in a maximum gap way for mainly two reasons and limitations introduced by the window-like temporal constraint: a window like a constraint limits the overall duration of temporal patterns to be discovered and discovering temporally large pattern needs the definition of large temporal search spaces.

To illustrate these two aspects, let us consider a temporal phenomenon P described by ten facts. It occurs typically in a period of 200 seconds with time lags between successive facts ranging between 0 and 30. Considering a window-like constraint, one must specify a temporal constraint that is large enough (e.g. $\Delta = [0, 300]$) to catch the correlation between the first and the last temporal fact. Otherwise, the overall temporal correlation between the ten facts will not be discovered. Hence, if the user does not have precise knowledge about temporal phenomena described in a data set (particularly with streaming data that have no activity separation), it is likely to "miss" large interesting temporal phenomena. We consider the large successions of temporal facts provide more insight about phenomena. For instance, a trajectory of 100 steps is more interesting than numerous sub-trajectories of 10 steps. The maximum gap constraint permits to obtain such large temporal relations as it does not bound patterns temporally to be discovered but constrains the temporal gap between the closest facts.

The second argument in favour of gap-like constraint is the temporal search space size. With our former example, discovering the overall temporal phenomenon needs a minimal window-like constraint of $\Delta = [0, 200]$. This constraint defines a search space of 200^2 temporal transformations. If one considers an Apriori-like algorithm, performing an incremental extension of temporal patterns taking into account conditional temporal relations (as in our model), each extension of the temporal pattern (performed nine times for ten steps) needs the exploration of 200^2 possible temporal transformations. With a maximum gap constraint, the same results can be achieved with a $\Delta = [0, 30]$ performing nine explorations of a temporal search space of size 30^2. The temporal search space cost with the gap constraint is smaller with the gap constraint by one order of magnitude in comparison with even one search space exploration with the window-like temporal constraint approach. Following this discussion, we defined temporal constrained Complex Temporal Dependencies.

Definition 26 (Temporally constrained CTD)

Let $\mathcal{S} = \{A_1, A_2, ..., A_n\}$ be a set of state streams and $\Delta = [min, max]$ a constraint on temporal transformations. Let R be a Complex Temporal Dependency over $\mathcal{S}$:

$$R = \bigwedge_{i_1} \bigvee_{i_2} \dots \bigwedge_{i_k} \bigvee_{i_{k+1}} S_{i_1 \dots i_{k+1}}^{(\alpha_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}})} \rightarrow \bigwedge_{i_{k+2}} \bigvee_{i_{k+3}} \dots \bigwedge_{i_{n-1}} \bigvee_{i_n} S_{i_{k+2} \dots i_n}^{(\alpha_{i_{k+2} \dots i_n}, \beta_{i_{k+2} \dots i_n})}$$

where $\forall i, S_i \in \mathcal{S}$ and $\exists i_1 \dots i_{k+1}$ such that $(\alpha_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}}) = (0, 0)$. R is said to verify the Δ constraint if $\forall (\alpha_i, \beta_i), \exists (\alpha_j, \beta_j)$ such that $|\alpha_i - \alpha_j| \in \Delta$ and $|\beta_i - \beta_j| \in \Delta$.

For instance, let $\Delta = [0, 10]$ be a temporal constraint. The complex temporal dependency is a temporally constrained Δ:

$$A^{(0,0)} \wedge B^{(5,4)} \rightarrow C^{(15,6)} \wedge D^{(15,16)}$$

One can remark that C and D have temporal transformations that are not in Δ but are temporally distant from another state in the dependency by a time lag in Δ. For instance, the temporal transformation of C has an expansion value, 15, that is temporally distant by $10 \in \Delta$ from the expansion of B: the time lag between state B and state C respects the temporal constraint.

Non-cyclic constraint

The Δ temporal constraint makes the temporal search space finite, but the overall search space (combinations of states and operators) remains infinite. In order to propose the first algorithm using the Complex Temporal Dependencies model, we chose to constraint the search space to non-cyclic dependencies. A non-cyclic dependency is a dependency where a state can appear at most once. Follows the formal definition.

Definition 27 (Non-cyclic CTD)
Let $\mathcal{S} = \{A_1, A_2, ..., A_n\}$ be a set of state streams. Let R be a Complex Temporal Dependency over $\mathcal{S}$:

$$R = \bigwedge_{i_1} \bigvee_{i_2} ... \bigwedge_{i_k} \bigvee_{i_{k+1}} S_{i_1 ... i_{k+1}}^{(a_{i_1 ... i_{k+1}}, \beta_{i_1 ... i_{k+1}})} \rightarrow \bigwedge_{i_{k+2}} \bigvee_{i_{k+3}} ... \bigwedge_{i_{n-1}} \bigvee_{i_n} S_{i_{k+2} ... i_n}^{(a_{i_{k+2} ... i_n}, \beta_{i_{k+2} ... i_n})}$$

where $\forall i, S_i \in \mathcal{S}$ and $\exists i_1 ... i_{k+1}$ such that $(a_{i_1 ... i_{k+1}}, \beta_{i_1 ... i_{k+1}}) = (0, 0)$. R is said to be non-cyclic if $\forall S_i \in R, \nexists S_j \in R, i \neq j$ such that $S_i = S_j$.

For example, $A^{(0,0)} \wedge B^{(10,12)} \rightarrow A^{(10,0)}$ do not respect the non-cyclic constraint. This type of dependencies can be enough to describe temporal phenomena where states are not likely to appear more than once. A typical example is that of straight trajectories. The non-cyclic constraint is introduced for simplicity allowing the design a first algorithm devised to validate our approach and study its efficiency. A straightforward direction towards allowing cyclic temporal relationships is to duplicate a given stream and name it differently. However, this approach causes information duplication since every interval-based stream is correlated with itself (a confidence of 1 is given for a temporal transformation $(0, 0)$). Hence, even if no "interesting" dependency exists in a dataset of n duplicated streams, at least $\frac{n}{2}$ dependency will be given. We will discuss one direction to get rid of the non-cyclic approach at the end of this chapter.

Non-conjunctive conclusions

In Section 5.4, we discussed how complex temporal dependencies could be used to build temporal models of environments activities. What can be concluded from that section is that such models can be built using complex temporal dependencies with simple conclusions and dependencies representatives. Such dependencies can be called non-conjunctive conclusion dependencies and defined as follows.

Definition 28 (Non-conjunctive conclusion CTD)
Let $\mathcal{S} = \{A_1, A_2, ..., A_n\}$ be a set of state streams. A non-conjunctive conclusion Complex Temporal

Dependency over $\mathcal{S}$ is defined as:

$$R = \bigwedge_{i_1} \bigvee_{i_2} \dots \bigwedge_{i_k} \bigvee_{i_{k+1}} S_{i_1 \dots i_{k+1}}^{(a_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}})} \rightarrow S_j^{(a_j, \beta_j)}$$

where $\forall i, S_i \in \mathcal{S}$ and $\exists i_1 \dots i_{k+1}$ such that $(a_{i_1 \dots i_{k+1}}, \beta_{i_1 \dots i_{k+1}}) = (0, 0)$.

For instance, $A \rightarrow B^{(a,\beta)} \wedge C^{(a',\beta')}$ is not a non-conjunctive conclusion CTD since its conclusion is composed by a conjunction of states of the input stream set.

6.1.2 Problem formalization

In this chapter, we are interested on the following problem:

Problem 5

Let $\mathcal{S} = \{A_1, A_2, A_3, \dots A_n\}$ a set of state streams and $\Delta = [min, max]$ a temporal constraint on temporal transformations. Our goal is to discover statistically valid, temporally constraint constrained by Δ, non-cyclic with non-conjunctive conclusion complex temporal dependencies.

6.1.3 Search space study

In this subsection, we study the search space defined by the constrained temporal dependencies pattern language. Let us consider a set of n state streams $\mathcal{S} = \{A_1, \dots, A_n\}$ and a temporal constraint $\Delta = [min, max] \in \mathbb{N}$. To structure the study of the search space defined by our problem, we propose to consider its two dimensions: temporal and logical.

The temporal search space for a single state is constituted with all its temporally transformed variations. As discussed earlier in this section, the number of theoretically possible temporal transformation for a single state with a $\Delta = [min, max]$ temporal constraint is $|\Delta|^2 = |max - min|^2$ if one considers all pairs (a, β). As a consequence, for each dependency of k states, exists $(|\Delta|^2)^{(k-1)}$ temporal variation: each state can use $|max - min|^2$ temporal variations except the reference state (with $(0, 0)$ transformation).

The logical search space contains all possible combinations of states. Since, we are interested in single conclusions dependencies with the non-cyclic constraint, the number of possible conjunctive dependencies (without disjunctions) is given by:

$$\sum_{k=1}^{n-1} \binom{n}{k} (n - k)$$

For a given index k of the sum, the first term $\binom{n}{k}$ corresponds to the number of possible premises given by the k-combination from the set of states $\mathcal{S}$. The second term $(n - k)$ is the number of possible conclusions given the non-cyclic constraint: for each k states premise, there is $(n-k)$ possible conclusion. Thus, the product $\binom{n}{k}(n-k)$ provides the number of dependencies of the form $S_1 \wedge S_2 \dots \wedge S_k \rightarrow S_{k+1}$ where $S_i \in \mathcal{S}$. This sum can be expressed in a more simplified way with:

$$(2^{n-1} - 1) * n$$

Proof.

$$\begin{aligned}
\sum_{k=1}^{n-1} \binom{n}{k}(n-k) &= \sum_{k=1}^{n-1} \frac{n!}{k!(n-k)!}(n-k) \\
&= n * \sum_{k=1}^{n-1} \frac{(n-1)!}{k!(n-k-1)!} \\
&= n * (\sum_{k=0}^{n-1} \frac{(n-1)!}{k!(n-k-1)!} - 1) \\
&= n * \big((\sum_{k=0}^{n-1} \binom{n-1}{k}) - 1\big) \\
&= n * \big(2^{n-1} - 1\big)
\end{aligned}$$

□

The overall number of dependencies using only the conjunctive relations with a simple conclusion is given by:

$$n \sum_{k=1}^{n-1} \frac{(n-1)!}{k!(n-k-1)!} * (|\Delta|^2)^k$$

The term $(|\Delta|^2)^k$ corresponds to the number of temporal variations of states used in the conjunctive dependencies. This expression can be simplified to:

$$n * \big((1 + |\Delta|^2)^{n-1} - 1\big)$$

Proof.

$$n \sum_{k=1}^{n-1} \binom{n-1}{k} * (|\Delta|^2)^k = n\big((\sum_{k=0}^{n-1} \binom{n-1}{k} * (|\Delta|^2)^k) - 1\big)$$

The binomial theorem states that:

$$\sum_{k=0}^{n} \binom{n}{k} * r^k = (1+r)^n$$

The direct application of this theorem to our sum gives result:

$$n * \big((1 + |\Delta|^2)^{n-1} - 1\big)$$

□

This number is a lower bound of the total search space since it does not consider disjunctive logical relations. This search space size, however, indicates the complexity of a brute force exploration algorithm showing that such an approach can not be considered for even small sets of states streams and small temporal constraints. What can be concluded is that the realistic exploration of the search space defined by our problem can not be achieved without efficient search space pruning criteria.

6.2 Preliminaries

In the last section, we defined the problem of mining Complex Temporal Dependencies from a set of state streams. The study of the search space of conjunctive temporal dependencies showed that a naive brute force approach could not apply to even small sets of state streams.

As discussed in Chapter 4, several properties can be used to prune search spaces for sequential pattern mining, that is based, mostly, on count-based interestingness measure. We motivated in Chapter 5 the use of an intersection-based interestingness measure for complex dependencies mining. In this section, we discuss the applicability of some of the available pruning mechanism to our problem, given the intersection-based confidence measure. More precisely, we are particularly interested in the *Apriori* property and closure checking.

6.2.1 Is the Apriori property suitable for complex temporal dependencies mining?

For sequential pattern mining, the Apriori pruning mechanism relies on the downward closure property of the count-based support of sequential pattern. This property states that any sub-sequence of a frequent sequential pattern is also frequent. It is based on the monotonic decrease of the count-based support interestingness measure with pattern extensions. For instance, if a sequential pattern ABC have a support of s, any of its sub-sequences have a support equal or greater than s. The *Apriori* pruning mechanism uses this property as follows: any super-sequence of an infrequent sequential pattern does not need to be generated and tested. The question we pose in this subsection is: can we use an *Apriori*-like pruning mechanism to discover the subset of complex temporal dependencies corresponding to our problem?

To respond to this question, let us first consider the differences between our problem and that of sequential patterns/rule mining. As discussed in Chapter 4, sequential rule mining often use two user-given parameters: minimum support and minimum confidence. Without digging into details, sequential rule mining approaches mainly use the set of frequent sequences to build valid sequential rules. In this process, the *Apriori* pruning mechanism intervenes in the support checking step (whatever the used algorithmic approach) thanks to the downward closure property of the support measure. For example, rules that can be constructed from a sequence ABC (i.e. $AB \rightarrow C$ or $A \rightarrow BC$) will not be generated and tested if A or B are not frequent. While sharing the same objective (discovering interesting rules/dependencies) our problem is different in that, *inter alia*, there is no equivalent concept of "frequent sequence". A sequence ABC can be interesting w.r.t the minimum support while, with our problem settings, $A \wedge B \wedge C$ is not even assessed nor corresponds to a particular interest. This would require the use of minimum user-given intersection-length thresholds which is not of interest for our problem. Indeed, we aim to avoid any user-given interest threshold and rely on the statistical assessment of confidence values. Therefore, an *Apriori*-like pruning mechanism is interesting if it can be applied straightforwardly to the intersection-based confidence measure.

As our objective is to discover a set of significant complex temporal dependencies, one straightforward algorithmic approach is to use incremental dependencies extensions to build statistically significant complex temporal dependencies. In our problem context, this means performing incremental premise extensions (with supplementary conjunctive and disjunctive relations) and prune while finding non-valid dependencies. For example, a dependency $A \wedge B \wedge C \rightarrow D$ is obtained by assessing successively $A \rightarrow D$, $A \wedge B \rightarrow D$ and $A \wedge B \wedge C \rightarrow D$. If, for example, $A \wedge B \rightarrow D$ is not statistically valid the potential downward closure property would permit to ensure that $A \wedge B \wedge C \rightarrow D$ is not statistically valid and avoid assessing it. In this context, the *Apriori*-like pruning mechanism can be formulated as follows:

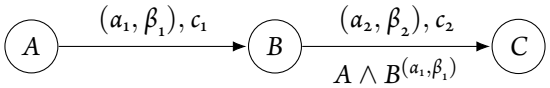
If a dependency R is not statistically valid, then all its super-dependencies are not statistically valid.

In order to apply this approach, the statistical validity defining the interest of a dependency must be downward closed w.r.t to dependencies extensions:

Sub-dependencies of a statistically valid dependency are also statistically valid.

Two conditions are necessary to verify this property for our problem: (1) the interest threshold must be monotonic w.r.t premise length (2) the interest measure must be inversely monotonic to the interest threshold. More specifically, if confidence is monotonically increasing with premise extensions, thresholds must by monotonically decreasing and vice versa. The inverse monotonicity condition is necessary to guarantee the validity preservation since we use dynamic statistical thresholds that depend on premise and conclusion lengths. The use of a static (or fixed) validity threshold (frequent sequence mining) can be seen as a specific case since it is a constant value. In our context, while the first condition is verified, unfortunately, the second one is not (cf. Section 5.3.1): the confidence measure is not monotonic with premise extensions. Therefore, we conclude that an Apriori-like pruning mechanism can not be applied with a premise extension exploration strategy.

In this work, we propose to consider a different exploration approach that is more directed towards the discovery of conditional temporal models using Complex Temporal Dependencies. As described in Section 5.4, a conditional temporal relation, modelled as a directed graph, is built in order to preserve conditionality of temporal relations. For more clarity, let us consider the following example.



In this example, we argue that the temporal phenomenon involving A, B, and C is of interest only if each of its conditional "steps" corresponds to a statistically valid temporal dependency. That is to say that if the relation between A and B is not statistically valid, then there is no interest in assessing the relation between A, B then C. This assumption permits to define, somehow, a downward closure based on the graph representation of temporal phenomena. It can be formalised as follows:

P a temporal phenomenon and $G = \{V, E\}$ a conditional temporal graph modelling P. P is considered as interesting if every $e \in E$ corresponds to statistically valid non-conjunctive CTD

With the former example, the temporal relationship between A, B and C is interesting if $A \rightarrow B^{(\alpha_1,\beta_1)}$ and $A \wedge B^{(\alpha_1,\beta_1)} \rightarrow C^{(\alpha_2,\beta_2)}$ are statistically valid. An Apriori-like mechanism can be derived from this statement for incremental construction of statistically valid temporal relationships. For instance, let $A \rightarrow B$ a temporal dependency and $(A \rightarrow B)_r$ its stream representative. If $A \rightarrow B$ is statistically valid, then dependencies taking the form $(A \rightarrow B)_r \rightarrow X \equiv A \wedge B \rightarrow X$ with $X \in \mathcal{S}$ can be tested. Otherwise, they are pruned.

6.2.2 "Closure" checking with an interval-based confidence measure

The concept of closed pattern is useful to perform both exploration search space pruning and to reduce information redundancy or pattern flooding without loss of expressive power. Let us first recall the definition of closed sequential patterns.

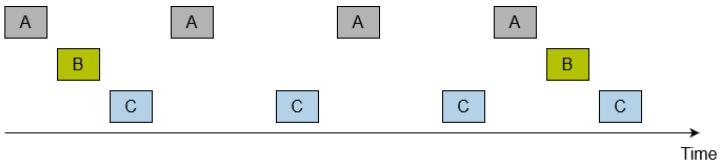


Figure 6.2.1: Succession of intervals A, B, and C. Successions ABC and AC are closed. AB and BC are not.

Definition 29 (Closed sequential pattern (from [162]))
Let FS be the set of frequent sequential patterns including all sequences having a support no less than the minimum support. The set of closed sequential pattern CS is defined as follows: $CS = \{a | a \in FS$ and $\nexists \beta \in FS$ such that a is a subset of β and a and β have the same support$\}$

Let us consider the following simple illustrative example. Let $ABCD$ be a frequent sequential pattern having a support of s. If CD have the same support s, then CD is not closed. This can be interpreted as follows: all occurrences of CD contribute to the occurrence of the larger pattern $ABCD$. In other words, all $ABCD$ and CD report on the same temporal phenomenon occurrences. Thus, each frequent pattern obtained by extending CD is a sub-pattern of a frequent pattern extending $ABCD$. Therefore, extending CD can be avoided and $ABCD$ considered as a unique result that maintains the expressive power of CD or any of its extensions. We aim to extend this reasoning for intersection based dependencies.

Let us first illustrate this with a simple example. We describe in Figure 6.2.1 3 streams A, B and C. From a counting support perspective, in this configuration, succession of intervals ABC and AC are closed while BC and AB are not: AB and BC have both a support of 2 that is equal to the support of ABC. Indeed, one can notice that occurrences of events A, B and C that contributes to the support of AB and BC are the those contributing the the support of ABC. Let us now extend this to intersection based reasoning. The following set of 4 dependencies can be extracted from the previous streams:

$$R_1 = A \rightarrow B^{(a,a)}, conf = 0.5$$
$$R_2 = A \wedge B^{(a,a)} \rightarrow C^{(2a,2a)}, conf = 1$$
$$R_3 = B \rightarrow C^{(a,a)}, conf = 1$$
$$R_4 = A \rightarrow C^{(2a,2a)}, conf = 1$$

We describe in Figure 6.2.1 the representative streams of each of these dependencies. What can be noticed is that both R_1 and R_2 have *correspondent* representative streams. More precisely the following equality is obtained:

$$(R_1)_r \cap (R_2)_r = (R_1)_r = (R_2)_r$$

This can be interpreted as follows: active durations of streams A and B that contribute to dependency R_1 are the same that contribute to dependency R_2. Thus, it can be concluded that these two dependencies are generated by the same temporal phenomenon. This reasoning can be also applied to R_3 using a supplementary temporal transformation on its representative. Indeed, as B in R_1 use an (a, a)-transformation, and B is the temporal reference of R_2, assessing if these two dependencies describe the same phenomenon can be achieved by comparing the representative streams of $(R_1)_r$ and $(R_3)_r^{(a,a)}$. On the other hand, it can be easily noticed in Figure 6.2.2 that R_2 does not contain all information provided by R_4: R_4 involves more active intervals of A. The equality of representative

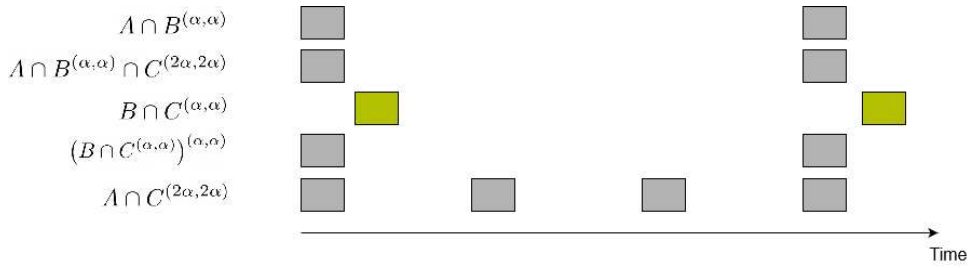


Figure 6.2.2: Interval streams representatives of dependencies $A \rightarrow B^{(a,a)}$, $A \wedge B^{(a,a)} \rightarrow C^{(2a,2a)}$, $B \rightarrow C^{(a,a)}$ and $A \rightarrow C^{(2a,2a)}$ extracted from streams in Figure 6.2.1

streams can also be expressed in terms confidence measures of dependencies between R_1 and R_2 as follows:

$$conf((R_1)_r \rightarrow (R_2)_r) = 1 \text{ and } conf((R_1)_r \rightarrow (R_2)_r) = 1$$

These relations states that $(R_1)_r$ and $(R_2)_r$ are mutually and strongly correlated. However, the strict equality is too restricting to be applied with noisy real world data. We propose then to use a relaxation parameter ε to allow a certain degree of error that may correspond to noise or to temporal variability (i.e the same phenomenon can be occurring with slight temporal variations). In order to exempt the user from providing the epsilon parameter that may need expert knowledge, one approach is to use the statistical threshold on confidence measure. It corresponds to the minimum intersection length that can be considered as statistically valid given the statistical test of independence. The loss of a smaller amount w.r.t the maximal confidence value $(=1)$ can be considered as *non-significant*. Follows the definition of the relaxed correspondence relationship.

Definition 30 (Interval Streams correspondence relationship)
Let A and B two interval streams. A and B are correspondent if:

$$conf(A \rightarrow B) \leq 1 - \varepsilon$$
$$\text{and } conf(B \rightarrow A) \leq 1 - \varepsilon$$

with ε a relaxation parameter.

This relationship is used to detect closed-like complex temporal dependencies. We refer this property of temporal dependencies as closed-like as it permits only to infer that two dependencies are describing the same temporal phenomenon but are is not lossless. Indeed, a closed-like dependency does not provide information about the confidence of sub-dependencies. For instance, R_2 does not provide information about confidence value of R_1.

Definition 31 (Closed-like Complex Temporal Dependency)
Let $\mathcal{S}$ be a set of interval-based streams and $\mathcal{R}$ a set of Complex Temporal Dependencies over $\mathcal{S}$, a dependency $R \in \mathcal{R}$ and S the temporal reference state of R. R is closed-like in $\mathcal{R}$ if $\nexists R' \in \mathcal{R}$ such that R is a sub-dependency of R' and $(R')_r$ is correspondent to $(R)_r^{(a,\beta)}$ with (a, β) the temporal transformation of S in R'.

While not preserving confidence information in all cases, this property is useful for a discovery process to implement two main pruning strategies: sub-temporal relations pruning and dependencies merge.

The dependencies merge consists of merging dependencies of the form $A \rightarrow B$ and $B \rightarrow C$ if their representative streams are correspondent. This would permit to avoid testing a dependency $A \wedge B \rightarrow C$ and reducing its cost to the computation of the closed-like checking and the dependency confidence rather than exploring the entire temporal search space. The second way to benefit from the closed-like CTD property is that of sub-temporal relations pruning. We will detail this in the next section as we describe the CTD-Miner algorithm.

6.3 CTD-Miner

In this section, we introduce the Complex Temporal Dependencies Miner (CTD-Miner) that is devised to discover statistically valid Complex Temporal Dependencies to build a graph-based description of conditional temporal phenomena that exists in a set of state stream $\mathcal{S}$. It uses a breadth-first strategy based on discovering statistically valid Complex Temporal Dependencies of incremental sizes such that the stream representatives of dependencies assessed in an iteration k are used as premises for iteration $k+1$. In this process, the *Apriori*-like pruning is used to eliminate non-valid dependencies and benefits from the closed-like property of Complex Temporal Dependencies. Hereafter, we describe more precisely each of the main steps of CTD-Miner.

Before detailing the main steps CTD-Miner, let us first describe our algorithm inputs and output. $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ is a lexicographically ordered set of n temporally ordered interval-based state streams. Each stream in $\mathcal{S}$ must have a unique label. We will assume that the stream length of state streams are known *à priori*. $\mathcal{T} \in \mathbb{N}^*$ is an observation duration. If not available, this parameter can be directly computed from $\mathcal{S}$ with:

$$\mathcal{T}_{min} = t_{max} - t_{min} \mid \forall S \in \mathcal{S}, \forall [b, e] \in S, t_{min} \leq b, t_{max} \geq e$$

$\mathcal{T}_{min}$ is the minimum observation duration that contains all intervals of streams in $\mathcal{S}$. Its computation can be done in $\Theta(n)$[1]. $\Delta = [min, max]$ such that $min, max \in \mathbb{Z}^2$, is the constraint on (α, β)-transformation: $min \leq \alpha \leq max$ and $min \leq \beta \leq max$. We stress the fact the no order constraint is imposed on the pair α and β (we will come back and motivate this point in the next chapter). The output of CTD-Miner is a set of complex temporal dependencies permitting to build a conditional temporal relations graph.

CDT-Miner uses a breadth-first strategy to explore the conjunctive complex temporal dependencies search space. As we described earlier, this exploration is not based on dependencies extensions (extending premises or conclusions of previously assessed dependencies) but on conditional relations extensions using dependencies stream representatives. We describe in the next paragraph the primary process of CTD-Miner before providing more specific details.

The incremental construction of conditional temporal relations starts considering all streams in $\mathcal{S}$ as premise candidates and stored in the "candidates" set (line 2). Objects in this set are structured as follows:

$$\langle \big(D_i, conf(D_i), (D_i)_r\big) \rangle$$

D_i is a conjunctive temporal dependency such that $\forall i, D_{i+1} = (D_i)_r \rightarrow S$ with $S \in \mathcal{S}$. This structure describes a conditional temporal relation graph: each element corresponds to an edge in the conditional graph. The initial-

[1] Computing $\mathcal{T}_{min}$ consists of finding both the minimum and maximum time stamps in temporally ordered intervals sequences. Temporal order permits to avoid iterating over all intervals of sequences in $\mathcal{S}$

Algorithm 1: CTD-Miner

Input: $\mathcal{S}$: a lexicographically ordered set of state streams
$\mathcal{T}$: observation duration
$\Delta = [min, max]$: temporal constraint on time lags
Output: $\mathcal{R}$: conditional temporal relations

```
1  R ← ∅
2  candidates ← {∀S ∈ S | ⟨(S → ∅, 1, S)⟩}
3  while |candidates| > 0 do
4      nextCandidates ← ∅
5      for p ∈ candidates do
6          extended ← False
7          process ← PrePruning(p, nextCandidates ∪ R)
8          if process then
9              for s ∈ S do
10                 if NonCyclic(p,s) then
11                     Δ' ← Update(Δ, p)
12                     results ← TimeLagDiscovery(p, s, Δ', T)
13                     if |results| > 0 then
14                         extended ← True
15                         newCandidates ← newCandidates ∪ results
16             if |p| > 1 and not extended then
17                 R ← AddAndEnsureClosure(R, p)
18     newCandidates ← MergeDependencies(newPremises)
19     candidates ← newCandidates
20 R ← BuildDisjunctions(R)
21 return R
```

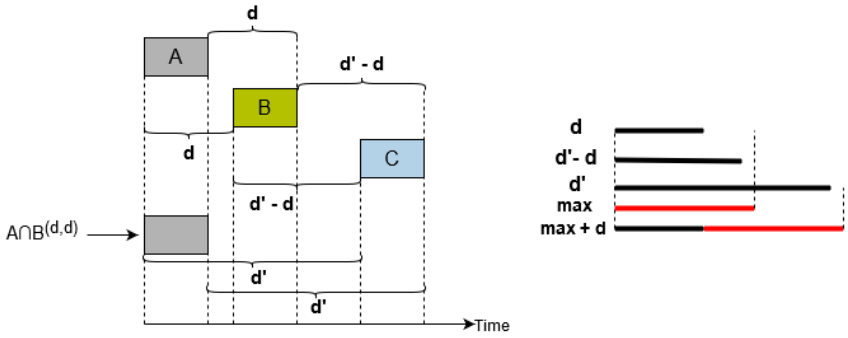


Figure 6.3.1: 3 interval streams portions. Their endpoints time lags are reported at the right of the figure with a temporal constraint $\Delta = [0, max]$.

ization of this structure is constituted by objects of the form:

$$(S_i \rightarrow \emptyset, -, S_i)$$

where $S_i \in \mathcal{S}$.

In the main While loop (line 3 to 21), for each pair (p, s) of premise candidate in *candidates* and state stream in $\mathcal{S}$, the algorithm calls a quantitative pairwise dependency method, called generically *TimeLagDiscovery* (line 13). This method is devised to discover "interesting" dependencies of the form $p \rightarrow s^{(\alpha,\beta)}$ given a temporal constraint Δ and an observation duration $\mathcal{T}$. A result given by this approach must have the same structure than objects in *candidates*. If an "interesting" relation is found from a candidate p and a conclusion stream s, it is added at the end of p with its confidence and representative stream. For a simplicity purpose, we assume, for now, that CTD-Miner uses a hypothetical efficient approach. One can also observe that multiple temporal relations can be found for a single pair (p, s). The obtained results are added to the next iteration candidates (line 15) in order to be extended. If no result is found, p is added to the result set $\mathcal{R}$ (line 18) following the Apriori-like principle we discussed in the last section. The end of the While loop is guaranteed by the non-cyclic condition that is checked at line 10. This is achieved by checking if the label of s is included in the set of labels of p (in $\Theta(|p|)$).

Temporal Constraint Update

In section 6.1.1, we discussed the temporal constraint application. We argued that a window-like temporal constraint can be limiting for the discovery of large patterns that describe phenomena that last in time more than the temporal constraint duration. In our context, using a such temporal constraint, i.e $\Delta = [min, max]$ for the (α, β)-transformations approach, limits the duration of dependencies to $max - min$ time units. This can be easily noticed in Figure 6.3.1. It describes the following temporal phenomenon: state A is active, then state B after a duration d from A, then state C after a duration d' from A. If one considers a temporal constraint $\Delta = [0, max]$ such that $max < d'$ (depicted in red) , two dependencies can be discovered $A \rightarrow B^{(d,d)}$ and $B \rightarrow C^{(d'-d,d'-d)}$. However, $A \wedge B^{(d,d)} \rightarrow C^{(d',d')}$ is out of the temporal scope defined by Δ as $d' < max$. We consider that this type of dependencies, and by extension large durations temporal relations descriptions, are more interesting than a set of temporally limited dependencies.

In order to consider the Δ constraint as a gap-like one, CTD-Miner, in line 11, updates locally the Δ constraint that is to use the time lag discovery algorithm with respect to previously discovered dependencies. More precisely, the main idea is to define the temporal reference of the search space with respect to the latest non-transformed

stream that is added to the premise conjunction. In Figure 6.3.1, the temporal search space of dependencies of the form $A \wedge B^{(d,d)} \rightarrow X$ is shifted by d time units, i.e $\Delta' = [0 + d, max + d]$ (depicted in red in the figure), in order to capture all dependencies that would have been discovered for a dependency of type $B \rightarrow X$ without increasing the temporal search space exploration cost. In other terms, the temporal constraint update permits to consider B as a reference for the temporal search space. It is also to notice that this approach maintains the temporal relation between non-reference states. One can convince himself observing in Figure 6.3.1 that $d' = d + (d - d')$. Thus, if d and d' are discovered (i.e time lags for $A \rightarrow B$ and $A \wedge B^{(d,d)} \rightarrow C$) it is easy to infer the time lag between B and C in this description (i.e $d' - d$).

For simplicity, we used in the former example temporal transformations where expansion and reduction have the same value. In order to make it more general for cases where $\alpha \neq \beta$, we propose the following temporal constraint update for a dependency $A \rightarrow B^{(\alpha,\beta)}$ and $\Delta = [min, max]$:

$$\Delta' = [min + min(\alpha, \beta), max + max(\alpha, \beta)]$$

The justification for this formula follows mainly the same reasoning as the last paragraph but considering the first and last endpoints. The lower bound of Δ is updated w.r.t to the lowest transformation parameter and the higher bound w.r.t to the highest transformation parameter in order the guarantee that at least all "interesting" transformations that would have been discovered without the conjunctive relations can be discovered with it.

One limitation that is introduced by updating the temporal constraint for a premise $A \wedge B^{(\alpha,\beta)}$ is that these temporal relations with time lags corresponding to (α', β') such that $\alpha' \leq \alpha$ and $\beta' \leq \beta$ cannot not be discovered by extending the latest dependency. Such temporal relations will be discovered independently. This discovery leads to information redundancy if they correspond to the same temporal phenomenon. This problem is tacked with the closed-like pruning that we introduce in the next subsection.

6.3.1 Closed-like based pruning

As discussed in Section 6.2.2, the closed-like property of Complex Temporal Dependencies can be used to detect if the portions of state streams contributing to two different dependencies are describing the same temporal phenomenon. CTD-Miner uses this property on the purpose of pruning the search space and eliminate redundancies in the results set.

Let us first consider the example depicted in Figure 6.2.1 and described in Section 6.2.2. Due to the lexicographic order of state streams, the dependency $A \rightarrow B^{(\alpha,\alpha)}$ is discovered first. In this example, all intervals of B contribute to the former dependency. Therefore, the representative stream of $A \rightarrow B^{(\alpha,\alpha)}$ is correspondent to $B^{(\alpha,\alpha)}$. Hence, all dependencies of the form $B \rightarrow X$ with $X \in \mathcal{S}$ and their temporal relation extensions can be pruned as all the information provided by B is maintained by $A \rightarrow B^{(\alpha,\alpha)}$. Besides, CTD-Miner uses the temporal constraint update routine to make sure that dependencies having B as a reference state are included in the temporal search space of dependencies of the form $A \wedge B^{(\alpha,\alpha)} \rightarrow X$. This idea is implemented by the *PrePrunning* procedure at line 7 in Algorithm 1 and described by Algorithm 2. It checks if a temporal relation c is closed-like in a set of reference temporal relations. For each existing temporal relation r, *PrePruning* tests, first, if states in c are included in r (*IsIncluded* line 3). This operation is done in $\Theta(\#r)$ and consists on testing if states appearing in c are included in r and that their temporal transformations are consistent. For instance, if r contains a conjunction $A \wedge B^{(\alpha,\beta)} \wedge C^{(\alpha',\beta')}$, and c includes $B \wedge C^{(\alpha'',\beta'')}$, *isIncluded* tests if $\alpha' - \alpha = \alpha''$ and $\beta' - \beta = \beta''$. Then, if c is included in r, *PrePruning* calculates the transformation of the representative of c with the temporal transformation

Algorithm 2: PrePruning

```
   Input: c : a temporal relation
          R' : a reference set of temporal relations
1  ref_c ← temporal reference of c
2  for r ∈ R' do
3      if isIncluded(c, r) then
4          (α, β) ← temporal transformation of ref_c in r
5          c_r^(α,β) ← Transform(c_r, (α, β))
6          if isClosedLike(r_r, c_r^(α,β)) then
7              return True
8  return False
```

of its temporal reference in r (lines 4 and 5). Finally, it calls *isClosedLike* that returns a boolean indicating if $c_r^{(\alpha,\beta)}$ and r_r are correspondent. If it is the case, the procedure returns True and terminate. If non of the elements of $\mathcal{R}'$ are closed-like w.r.t c (i.e c is closed-like in $\mathcal{R}'$), *PrePruning* returns False.

The second manner CTD-Miner use the closed-like relationship is that of merging dependencies. It is implemented by *MergeDependencies* in line 18 of Algorithm 1. It takes as input the set of newly discovered temporal relationships at a given iteration of the main while loop. This function is devised to merge any couple of dependencies of type $A \rightarrow B$ and $B \rightarrow C$ where the conclusion stream of the first dependency is the reference state of the second. More precisely, it tests if the representative streams of the dependencies are correspondent (using the closed-like relationship). If so, a new dependency of the form $A \wedge B \rightarrow C$ is produced by extending the first dependency that is deleted from the new premises set. This process permits to avoid performing the time discovery algorithm (with the set of conclusion candidates given the non-cyclic condition).

At the end of the main While loop (line 18 in Algorithm 1), $\mathcal{R}$ contains a set of closed-like temporal relations discovered from $\mathcal{S}$. $\mathcal{R}$ is sufficient to build a conditional temporal relations graph $G(\mathcal{R}) = \{V, E\}$ where V contains state labels contributing to temporal relations in $\mathcal{R}$ and V is built using information in R. For each $R = \langle \{D_i, conf(D_i), (D_i)_r\} \rangle \in \mathcal{R}$, each dependency D_i is added to $G(\mathcal{R})$ as an edge in E containing the premise of D_i and its confidence. This representation reports on conditional conjunctive temporal. However, building such representation using the process described higher may include redundant information. For example, if two temporal relations have the same prefix, edges corresponding to the "similar" temporal relations portions will be duplicated. Moreover, the conjunctive-only representation does not provide insight about disjunctive relations confidences. These problems are partially tackled by the *BuildDijsunction* procedure (line 19) that we describe in the next subsection.

6.3.2 Computing disjunctive relations

The set of conjunctive complex temporal dependencies given by the incremental construction of temporal relationship permits to build a first version of the conditional temporal graph. However, as mentioned before, this data structure does not provide any information about disjunctive relationships confidences and may contain edges duplications. In this subsection, we motivate and describe the *BuildDisjunction* (described in Algorithm 3) procedure that is executed at the end, the incremental construction of conditional temporal relationships. It is first devised to eliminate potential duplicate edges corresponding to common prefixes and compute confidences

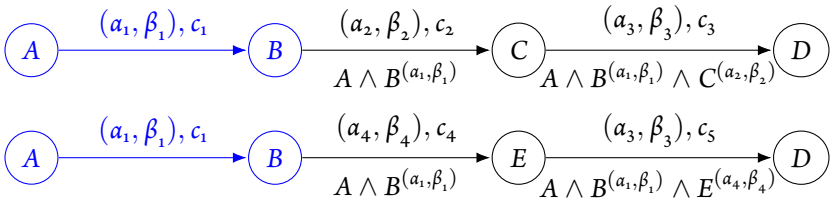
of a specific and useful subset of dependencies with disjunctive relationships.

Algorithm 3: BuildDisjunctions

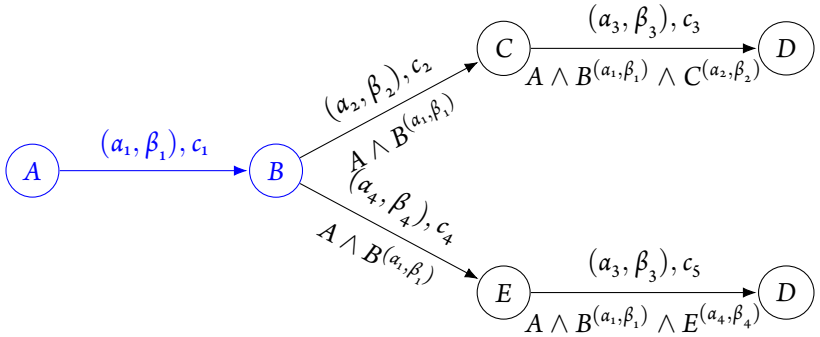
Input: $\mathcal{R}'$: a set of closed-like conjunctive temporal relations

for $r \in \mathcal{R}$ **do**
  for $r' \in reverse\mathcal{R}$ **do**
    if $r = r'$ **then**
      break
    $d_1, i_1, d_2, i_2 \leftarrow GetCommonPrefixIndexes(r, -1, r', -1)$
    if $i_1 \neq -1$ *and* $i_2 \neq -1$ **then**
      $suffix_1 \leftarrow GetSuffix(d_1, i_1)$
      $d_2.AddDisjunction(i_2, suffix_1)$
      $\mathcal{R} \leftarrow \mathcal{R}/d_1$
      return $BuildDisjunctions(\mathcal{R})$
return $\mathcal{R}$

Let us first illustrate the edge redundancy problem with the following two temporal relations graph representations:



The blue nodes and edges correspond to a common prefix of these two dependencies. What can be observed is that the edge linking A and B in these two temporal relations are redundant. A single graph representation, including these two temporal relations, will, then, contain duplicate information: two identical edges between A and B describing the same temporal relation (in term of intervals intersection). Therefore, the *BuildDisjunction* function performs a prefix factorisation to obtain the following temporal relation object:



This object cannot be considered yet a conditional temporal graph: node D is duplicated. This structure can be called a common-prefix conditional dependency tree. For this work, we consider that this type of structures have enough expressive power and is implemented in Algorithm 3.

6.3.3 Time Lag Discovery in CTD-Miner

The core operation executed by the incremental conditional temporal relations build is that of the discovery of quantitative correlations between two dependencies. In Algorithm 1, this step, called generically *TimeLagDiscovery* (line 12) is devised to detect quantitative temporal dependencies between two interval-based streams that are assessed w.r.t an interestingness measure and a temporal constraint. CTD-Miner can use any approach devised to detect meaningful time-lagged dependencies between interval-based streams with the condition of providing temporal information of the form of an (α, β) pair. These time lag information can be used to compute intersection-based confidences even if the approach does not consider such an assessment approach. For instance, CTD-Miner can be applied with a classical frequent pattern mining algorithm that uses a count-based interestingness measure, e.g. [78] or, as motivated later, intersection-based assessment as TEDDY [127] or ITLD that will be introduced in the next chapter. We will study in the experiments chapter the difference implied by the choice of the assessment approach in term of quality of results provided by CTD-Miner. In this subsection, we focus on intersection-based approaches and show that the time lag discovery step is critical to CTD-Miner in that it determines both the quality and performance of its result.

As showed before, a time lag discovery algorithm that can be used in our context is devised to explore a search space defined by the temporal constraint Δ. Since our dependency model expresses temporal information as a pair of values (α, β), a naive exploration of such search space for a dependency $p \rightarrow s$ can be done in $\Theta(|\Delta|^2 * E)$ where E is the cost of assessing a dependency $p \rightarrow s^{(\alpha,\beta)}$ (or computing the interestingness measure). For instance, if the intersection based confidence is used, the confidence computation can be done in $\Theta\big(max(\#p, \#s)\big)$. Since such exploration is executed at a high frequency in the incremental construction process that itself may be used in an online context, the complexity of the lag discovery process affects the efficiency of CTD-Miner greatly. Therefore, time lag discovery algorithms must be able to perform an efficient exploration thanks to pruning criteria and complexity reduction through efficient heuristics.

The second way the time lag discovery approach impacts CTD-Miner is linked to the Apriori-like incremental construction of conditional temporal relations. As described before, interesting temporal dependencies found at an iteration k are used as premises candidates at iteration $k + 1$. This implies that the more dependencies found at a given iteration, the greater is the cost of exploring the logical and temporal search spaces at the next iteration due to the combinatorial explosion of dependencies numbers. For instance, with a classical frequent sequence mining, the user must define a support threshold that is high enough to prune non-frequent (i.e. non-interesting) temporal relations but not too restrictive to be able to discover interesting relations. With the intersection-based statistical assessment approach, the challenge is to obtain the better compromise between precision and recall: maximising numbers of discovered true positives (valid time lags reporting on "real" temporal relations) while minimising numbers false positives (valid time lags that do not correspond to "real" phenomena). The rate of false positives affects CTD-Miner as it reduces the pruning power of the Apriori-like assumption significantly. This aspect will be developed in more details in the next chapter.

The final point we discuss in this subsection is that of specificity of temporal relationships when using the intersection-based assessment. An exploration of the temporal search space based on computing confidences and statistical validity thresholds for each pair of temporal transformation is not sufficient to obtain concise results. Due to the confidence monotonicity of temporal dependencies w.r.t conclusion length, several temporal transformations may lead to statistically valid temporal dependencies describing the same temporal transformation. One can convince himself of the latter statement with the following example. Let A and B two states, and a temporal phenomenon that links A and B with a time lag corresponding to (α, β). The confidence of $A \rightarrow B^{(\alpha,\beta)}$

is maximal and equals to 1. For each (α', β') such that $B^{(\alpha,\beta)} \cap B^{(\alpha',\beta')} = B^{(\alpha,\beta)}$, the confidence of $A \rightarrow B^{(\alpha',\beta')}$ equals to the maximal value 1. These two dependencies reports on the same temporal phenomenon but the first one is more specific in that is describes more precisely the temporal aspect of the relation. A time lag discovery algorithm used by CTD-Miner must be able to provide the most specific temporal dependencies; that is to say that it is capable of providing one dependency per statistically valid temporal phenomenon. The duplication of temporal dependencies reporting on the same interesting phenomenon also induces a combinatorial explosion causing a decrease in CTD-Miner's efficiency and pattern flooding in the result set.

As reported in Chapter 4, to the best of our knowledge, it exists two quantitative temporal pattern discovery algorithms that can be applied in the context of dependencies between interval-based streams: PIVOTMiner [78] that uses a count-based assessment approach and TEDDY [127] whose authors introduced the intersection-based statistical assessment we use in this work. In the next chapter, we will introduce an alternative algorithm to ITLD.

6.4 Conclusion & Discussion

In this chapter, we proposed the CTD-Miner algorithm. It is devised to compute a set of statistically valid dependencies permitting to build conditional temporal models reporting concisely on a set of temporal phenomena described by a set of interval-based streams. It uses an incremental construction approach that is enhanced using both an Apriori-like pruning mechanism and a closed-like relationship. The obtained Complex Temporal Dependencies can be used to build conditional temporal state graphs modelling temporal phenomena that exists in a given state stream set. Several directions can be investigated to extend this work.

The first aspect that can be investigated is that of getting rid of the non-cyclic constraint. One straightforward approach to obtain complex temporal dependencies, including several mentions of a given state is to duplicate it, with a different label, in the input streams set. We believe that this may introduce large amounts of redundant and trivial dependencies since each interval stream is correlated with itself (with a confidence of 1). Another approach can be that of using stream intersection to avoid taking into account in confidence measure the intersection of an interval with itself.

Secondly, CTD-Miner can be extended in order to fully take advantage of the disjunction operator included in the Complex Temporal Dependency model. The building disjunction approach proposed in CTD-Miner can be extended to build disjunctive "blocks" that allows expressing partial order. This would require supplementary calculations of confidence measures. For instance, in the example used in Section 6.3.2 with an initially active state A, the described model can provide precise confidences of observing states B, C, D and E:

- B was observed to be active after a duration corresponding to (α_1, β_1) with confidence c_1
- C was observed to be active after a duration corresponding to (α_2, β_2) with confidence c_2 if preceded by B.
- E was observed to be active after a duration corresponding to (α_4, β_4) with confidence c_4 if preceded by B.
- D was observed to be active after a duration corresponding to (α_3, β_3) with confidence c_3 if preceded by B and C.
- D was observed to be active after a duration corresponding to (α_3, β_3) with confidence c_5 if preceded by B and E.

In this example, an active state A is sufficient to provide without ambiguity time lags and confidences of further states B, C and E but provides two different descriptions for state D. For the latest state, one can distinguish two main cases: (1) active A is sufficient to describe the time lag and confidence for the occurrence of a further state D or (2) these time lag and confidence needs an active state C or E to be specified. We believe that these two cases correspond to two different semantic interpretations:

1. *ABCD* and *ABED* are two variations of the same temporal phenomenon/behaviour.
2. *ABCD* and *ABED* are two distinct temporal phenomena/behaviours sharing the same prefix.

These two cases distinguish two disjunctive blocks. An efficient disjunctions building must be able to distinguish between these two cases and recompute related confidences.

7

Mining significant quantitative temporal dependencies between pairs of interval-based streams

Contents

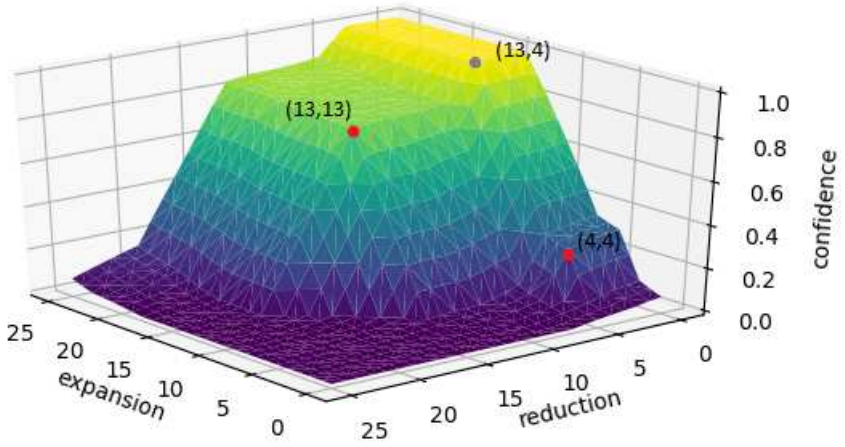


Figure 7.1.1: Search space for a pair of interval streams with two dependencies $A \rightarrow B^{(4,4)}$ and $A \rightarrow B^{(13,13)}$. $\Delta = [0, 25]$

7.1 Background & Problem statement

The problem of discovering quantitative dependencies between two interval-based streams A and B consists of detecting a set of time delays describing each a linear temporal relationship "of interest" between intervals of A and B. More specifically, we aim to extract a set of dependencies $\mathcal{D}$ of the form $A \rightarrow B^{(\alpha,\beta)}$ such that:

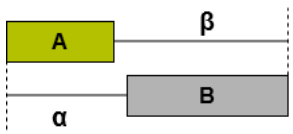
1. $A \rightarrow B^{(\alpha,\beta)}$ is valid w.r.t the intersection-based statistical assessment of the confidence measure.
2. $A \rightarrow B^{(\alpha,\beta)}$ is the most specific dependency describing a particular linear temporal relationship between A and B.

An efficient algorithm, TEDDY, was proposed in the seminal work [127] introducing this problem. In this subsection, we discuss each of the before-mentioned components of this exploration problem and TEDDY's solutions before concluding with the formal problem statement.

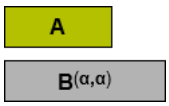
7.1.1 Discovering the set statistically valid dependencies

Given a temporal constraint $\Delta = [min, max]$, discovering statistically valid temporal dependencies between a premise A and a conclusion B is consists of exploring the search space composed of all dependencies $A \rightarrow B^{(\alpha,\beta)}$ such that $\alpha \in \Delta$ and $\beta \in \Delta$. This exploration consists of computing both the confidence value, done in $\Theta\big(max(\#A, \#B)\big)$, and the statistical threshold, in $\Theta(1)$, for each temporal dependencies. The naive exploration of such search space has a complexity in $\Theta\big(max(\#A, \#B)) * |max - min|^2\big)$ since every temporal transformation (α, β) is to be assessed. We describe in Figure 7.1.1 the search space for a pair of state streams with two temporal dependencies and $\Delta = [0, 25]$.

In [127], authors considered the exploration a sub-search space such that for every temporal transformation $\alpha \geq \beta \geq 0$. Such temporal transformation corresponds to cases where the conclusion streams are shifted ($\alpha = \beta$) or extended ($\alpha > \beta$). In this chapter, we will not consider such condition on extension and reduction values in order to obtain the most precise time delay description if active intervals of the premise are correlated with bigger conclusions intervals. This case can be observed in Figure 7.1.2) where the delay between A and B (B begins α



(a) Original intervals A and B



(b) Temporal transformation of B with the $\alpha \geq \beta$ constraint

(c) Temporal transformation of B without the $\alpha \geq \beta$ constraint

Figure 7.1.2: The $\alpha \geq \beta$ condition on temporal transformation do not permit to obtain the most precise time lag description

time units after the beginning of A and ends β time units after the end of A) is better described by $A \rightarrow B^{(\alpha,\beta)}$ while only $A \rightarrow B^{(\alpha,\alpha)}$ provides less "precise" characterization of the temporal relation.

In the same work, authors proposed TEDDY that explores (using pruning criteria) the defined search space (with $\alpha \geq \beta$) as a semi-lattice defined by the inclusion property of temporal transformations. A temporal transformation (α, β) is said to be included in (α', β') if $\alpha \leq \alpha'$ and $\beta \geq \beta'$ and noted as follows $(\alpha, \beta) subseteq (\alpha', \beta')$. An example of such lattice is provided in Figure 7.1.3 for $\Delta = [0, 5]$. In the worst case, the exploration of such search space have the same complexity as a brute force approach (i.e quadratic w.r.t to $|\Delta|$) but uses pruning criteria to accelerate the exploration process. Two main pruning approaches are proposed for TEDDY. The first uses confidence monotonicity of the confidence measure w.r.t conclusion length to prune unpromising temporal transformation candidates thanks to a lower bound on the statistical thresholds and a higher bound on confidence length. More precisely, for any temporal transformation with $\alpha \geq \beta$, the following lower bound on the statistical thresholds is verified:

$$LowerBound(\alpha, \beta) \geq min\big(1, th(0, 0)\big)$$

where $th(x, y)$ is the statistical threshold for a dependency $A \rightarrow B^{(x,y)}$. With transformations (α, β) such that $\alpha \geq \beta$, $\mathbf{len}(B^{(\alpha,\beta)}) \geq \mathbf{len}(B)$. Due to confidence monotonicity and the higher bound on the confidence measure (cf Figure 5.2.7) statistical thresholds used corresponding to temporal transformations included in the semi-lattice are lower bounded by $th(0, 0)$ (as $B^{(0,0)}$ have the lowest length) or 1 if $th(0, 0) > 1$. Given a dependency $A \rightarrow B^{(\alpha,\beta)}$ of depth d, it is possible to eliminate a non-promising transformation candidate (α_c, β_c) of the same depth using a confidence gain/loss higher bound:

$$maxGain = (|\alpha - \alpha_c| + |\beta - \beta_c|)\frac{\#B}{\mathbf{len}(A)}$$

for a dependency $A \rightarrow B$. Thus, if $\mathbf{conf}(A \rightarrow B^{(\alpha,\beta)}) + maxGain > boundMinConf$, candidate (α_c, β_c) can be eliminated as it is not possible for a dependency $A \rightarrow B^{(\alpha_c,\beta_c)}$ to be statistically valid. As we do not use in this work the $\alpha \geq \beta$ constraint this pruning approach cannot be straightforwardly be applied. The second pruning mechanism is based on a dominance relationship that controls confidences loss w.r.t to parents in the semi-lattice. We discuss this pruning technique in the next sub-section.

7.1.2 Specific temporal dependencies selection

The exploration of the search space defined in the later section permits to obtain a set of (α, β)-transformations such that each dependency $A \rightarrow B^{(\alpha,\beta)}$ is statistically valid. However, this result set may contain several statistically valid dependencies describing the same temporal correlation between the premise and the conclusion.

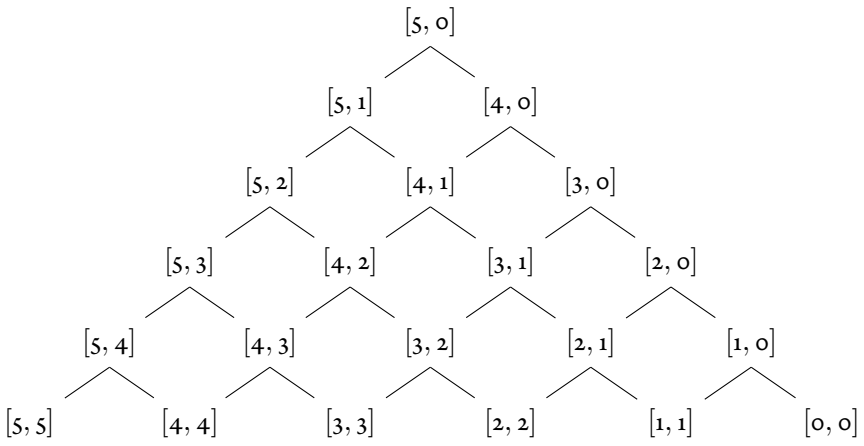


Figure 7.1.3: Semi-lattice of temporal transformations defined by the inclusion relations for $\Delta = [0, 5]$

One of the causes of this information redundancy is the confidence monotonicity w.r.t to conclusion length, especially when an inclusion relation stands between temporal transformations. Figure 7.1.4 provides an example of such a case. Both dependencies $A \rightarrow B^{(\alpha,\beta)}$ and $A \rightarrow B^{(\alpha',\beta')}$ maximize the confidence value (=1 and are both statistically valid) and describe the same temporal relation between A and B. The first dependency is considered as more specific: the intersection of A and $B^{(\alpha,\beta)}$ maximizes the confidence value in a "tight" manner, in that, no active length of $B^{(\alpha,\beta)}$ can be considered as "superfluous" for the intersection $A \cap B^{(\alpha,\beta)}$. For temporal transformations inclusion cases, authors of [127] proposed a dominance relationship permitting to compare the specificity of temporal dependencies.

Definition 32 (Dominance relationship [127])
Let $d_1 = A \rightarrow B^{(\alpha,\beta)}$ and $d_2 = A \rightarrow B^{(\alpha',\beta')}$ be two temporal dependencies. d_1 dominates d_2 if $(\alpha, \beta) \subseteq (\alpha', \beta')$ and

$$1 - \frac{\textbf{conf}(A \rightarrow B^{(\alpha,\beta)})}{\textbf{conf}(A \rightarrow B^{(\alpha',\beta')})} < 1 - \frac{\textbf{len}(B^{(\alpha,\beta)})}{\textbf{len}(B^{(\alpha',\beta')})}$$

This dominance relationship permits to refine the conclusion's intervals while controlling the loss of confidence. For $(\alpha, \beta) \subseteq (\alpha', \beta')$, if (α, β) dominates (α', β'), the confidence value will be reduced in the same proportions than the length of the conclusion. If it is not the case, the conclusion's length is more, proportionally, reduced than the confidence value since the reduced active intervals do not intervene in the intersection with the premise. Figure 7.1.4 provides a simple case where the loss of confidence between dependencies with (α, β) and (α', β') is equal to 0 while the reduction in B's length is not. In [127], this relationship is used in the semi-lattice level-wise search space exploration to stop the refinement process, thus efficiently prune the search space, if a significant loss is observed. One limitation of this approach appears when multiple significant dependencies stand between a pair of streams as in Figure 7.1.1. In this example, the most specific dependencies are $A \rightarrow B^{(4,4)}$ and $A \rightarrow B^{(13,13)}$ with temporal transformations that are singletons in the semi-lattice. If the latter is explored in a level-wise manner, dependency $A \rightarrow B^{(13,4)}$ will be tested first as it is more general than the first two. What can be noticed here is that direct children of $(13, 4)$ will not be dominant as the loss in confidence measure while refining intervals of B on both intervals endpoints is significant.

One other case of specificity that is not treated by the before-mentioned dominance relationship is cases where a specific temporal relation corresponds to an (α, β)-transformation such that $\alpha < \beta$. In such case, a level-wise

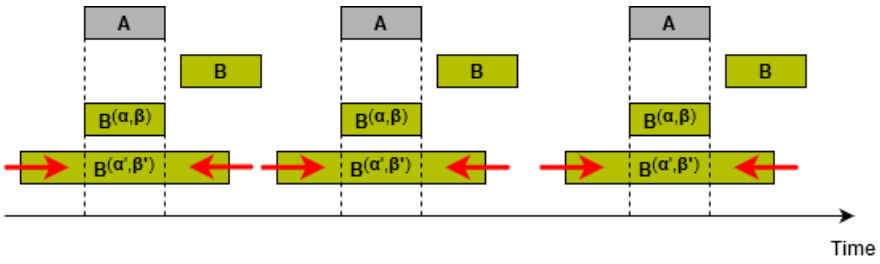


Figure 7.1.4: $A \rightarrow B^{(\alpha,\beta)}$ is more specific than $A \rightarrow B^{(\alpha',\beta')}$. The confidences of these two dependencies are equal to 1

exploration of the semi-lattice will provide as the most specific dependencies a set of dependencies of the form $A \rightarrow B^{(\theta,\theta)}$ such that $\alpha \leq \theta \leq \beta$ if those are its most specific ancestors included in the semi-lattice. For example, in the semi-lattice depicted in Figure 7.1.3, if exists a dependency $A \rightarrow B^{(1,3)}$ three dependencies will be returned: $A \rightarrow B^{(1,1)}$, $A \rightarrow B^{(2,2)}$ and $A \rightarrow B^{(3,3)}$. The dominance relationship can not be used in this case since there is no inclusion relationship between $(1,1)$, $(2,2)$ and $(3,3)$, yet, these dependencies describe, partially, the same temporal relation.

7.1.3 Problem statement

Given the exploration problem and the specific dependencies selection, we define the problem treated in this chapter as follows.

Problem 6 (Quantitative temporal dependencies mining)
Let A and B be two state streams and $\Delta = [min, max]$ a constraint on (α, β)-temporal transformations. Discover $\mathcal{R}$ the set of statistically valid and most specific temporal dependencies $A \rightarrow B^{(\alpha,\beta)}$.

In the remainder of this chapter, we will propose an alternative approach to the level-wise semi-lattice exploration of the search space of time-delayed dependencies based on the analysis of elementary confidence variations.

7.2 The maximal elementary confidence variations exploration strategy

As described in the last section, the main idea behind the dominance relationship proposed in [127] is the refinement of large statistically valid temporal-transformations (i.e. more general (α, β) values) while controlling the loss on confidence value. The exploration of dependencies with more specific time lags can be stopped when the observed loss on the confidence measure is considered as "significant" w.r.t the dominance relationship. For instance, in Figure 7.1.1, the level-wise exploration of the semi-lattice begins by testing the most general time lag, i.e. $[25, 0]$, and perform a refinement by exploring more specific children in a level-wise manner. Taking into account the dominance-based pruning, every refinement on the left of time lags $(\alpha, 4)$ with $\alpha \in [13, 25]$ induces a significant loss as well as every refinement on the right of time lags $(13, \beta)$ with $\beta \in [0, 4]$. For every loss, a search space pruning is performed, and none of its children is tested any more. Therefore, the level-wise algorithm, TEDDY, proposed in [127], will provide $(13, 4)$ as the most specific time lag. Several observations can be made:

1. $(13, 4)$ is the aggregation of actual specific time lags $(13, 13)$ and $(4, 4)$.

2. The level-wise exploration tested 65 time lags (with a confidence computation in $\Theta\big(max(\#A, \#B)\big)$) for this simple case $((25 - 13 + 1) * (4 - 0 + 1) = 13 * 5 = 65)$ over the 338 possible ones (for $\Delta = [0, 25]$)

In this section, we propose an alternative approach to the refinement process of TEDDY permitting to address the specificity problem for cases with multiple interesting temporal time lags stand between a pair of interval streams and that reduces the exploration complexity, from quadratic to linear w.r.t Δ.

The refinement process based on the dominance relationship used in the level-wise discovery process, assesses confidence losses based on elementary confidence variations.

Definition 33 (Elementary confidence variations)

Let A and B two interval-based streams and (α, β) a temporal transformation. Four elementary confidence variations can be defined for a dependency $A \rightarrow B^{(\alpha,\beta)}$:

- Elementary left-gain:$LG(\alpha, \beta) = \textbf{conf}(A \rightarrow B^{(\alpha+1,\beta)}) - \textbf{conf}(A \rightarrow B^{(\alpha,\beta)})$
- Elementary left-loss: $LL(\alpha, \beta) = \textbf{conf}(A \rightarrow B^{(\alpha,\beta)}) - \textbf{conf}(A \rightarrow B^{(\alpha-1,\beta)})$
- Elementary right-gain: $RG(\alpha, \beta) = \textbf{conf}(A \rightarrow B^{(\alpha,\beta-1)}) - \textbf{conf}(A \rightarrow B^{(\alpha,\beta)})$
- Elementary right-loss $RL(\alpha, \beta) = \textbf{conf}(A \rightarrow B^{(\alpha,\beta)}) - \textbf{conf}(A \rightarrow B^{(\alpha,\beta+1)})$

Assuming a maximal confidence variation assessment criteria permitting to decide whether a confidence variation is "significant" or not[1], the characteristic of specific temporal dependencies w.r.t to elementary gains and losses is the following:

- Each elementary confidence loss is "significant"
- Each elementary confidence gain is not "significant"

These two properties can be easily induced from the "tight" confidence maximization strategy (cf Figure 7.1.4). This also can be observed in Figure 7.1.1 where 3 temporal transformations verify these conditions: $(13, 13)$, $(4, 4)$ which are the most specific time lags and $(13, 4)$ which is the aggregation of the first two. What can also be noticed in this very example is that significant losses corresponding to specific dependencies can be "detected" for different transformation values. For instance, for specific time lag $(4, 4)$ the loss on confidence value induced by conclusion's intervals reduction on the left (subtracting a unit to the α expansion value) can be detected with $(4, 4)$, $(4, 3)$ ... $(4, 0)$. The same can be stated for confidence loss induced by intervals' reduction on the right (adding a unit to the β reduction value): $(4, 4)$, $(3, 4)$, ..., $(13, 4)$ ... $(20, 4)$. The inverse observation can also be made for specific time lags expansions: confidence gains are not *significant*. While the significance of elementary confidence variations for the same parameter is equivalent for numerous temporal transformations (as a sparse the temporal phenomena is used to generate the example data) the evaluation of confidence variations for the same expansion (or reduction) parameter can be non-equivalent given the sparsity and density of the interval streams. For instance, applying large temporal transformation for dense interval streams may induce interval overlaps which induces an underestimation of the actual influence of an elementary transformation with regards to the real (i.e non-transformed) temporal data. This statement is also valid with (α, β)-transformation with $\alpha < \beta$ with which intervals may be removed.

[1]This aspect of our exploration strategy will be treated in the next subsection.

To provide a precise assessment of time lag specificity, an approach needs to analyze confidence variation that reflects actual temporal relations between the original streams. One interesting property permitting to solve this problem is confidence variations upper-bound.

Property 6 (Maximal Confidence Variation)

Let A and B two interval-based streams. Elementary confidence variations for dependencies $A \rightarrow B^{(\alpha,\beta)}$ are upper bounded by confidence variation for $A \rightarrow B^{(\alpha,\alpha)}$:

$$\text{Maximum left-gain: } LG(\alpha,\beta) \leq LG(\alpha,\alpha)$$
$$\text{Maximum left-loss: } LL(\alpha,\beta) \leq LL(\alpha,\alpha)$$
$$\text{Maximum right-gain: } RG(\alpha,\beta) \leq RG(\alpha,\alpha)$$
$$\text{Maximum right-loss: } RL(\alpha,\beta) \leq RL(\alpha,\alpha)$$

Proof. Let A and B be two state streams. By definition of interval streams, for any interval $I = [t_i, t_{i+1})$ of B, $[t_i - 1, t_i)$ and $[t_{i+1}, t_{i+1} + 1)$ are inactive intervals (otherwise, they would be merged with I). Let (α, β) a temporal transformation. We can distinguish three cases when it comes to confidence variations for $A \rightarrow B^{(\alpha,\beta)}$:

- $\alpha > \beta$: some intervals of B may be merged in $B^{(\alpha,\beta)}$ due to extra length induced by an expansion greater than reduction. The computation of elementary confidence variation on $B^{(\alpha,\beta)}$ may induce a loss of information as it misses the contribution of potentially merged intervals.
- $\alpha < \beta$: some intervals of B may not be included in $B^{(\alpha,\beta)}$ if their length is lesser than $\beta - \alpha$. Their contribution to elementary confidence variation is missed.
- $\alpha = \beta$: the length and the time lag between intervals in B are preserved in $B^{(\alpha,\beta)}$. Every original interval contributes to the elementary confidence variation as the time lag between intervals equals at least to 1 time unit.

We conclude that elementary confidence variations for a temporal transformation (α, β) are upper-bounded by elementary confidence variations with temporal transformations (α, α) for left gains/losses and (β, β) for right gains/losses □

This property suggests that the assessment of confidence variations with temporal shifts (i.e. transformations of the form (α, α)) provides better insights on specific temporal time lags between intervals endpoints. Indeed, temporal shifts cannot induce intervals merging (information of interval overlaps is lost) or removal as the time lag between intervals, their length and numbers are preserved.

Based on this property, we propose an exploration strategy of the temporal search space devised to discover specific expansions and reductions values rather than specific temporal transformations. The main idea is to assess for each value $\alpha \in \Delta$ the "significance" of maximal elementary confidence variations, i.e left and right gains and losses, in order to detect independently specific expansions and reductions values. These specific expansion and reduction values can, then, be used to build and assess the statistical validity of specific temporal dependencies. Following this idea, the value $\alpha \in \Delta$ is said to be:

- a specific expansion if $LL(\alpha, \alpha)$ is "significant" and $LG(\alpha, \alpha)$ is not.

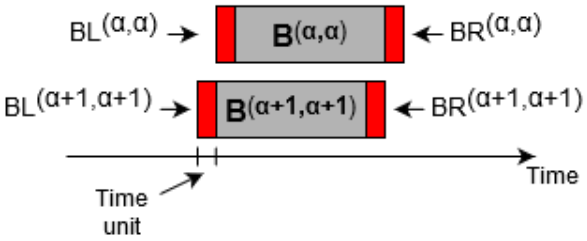
- a specific reduction if $RL(a, a)$ is "significant" and $RG(a, a)$ is not.

This straightforward assessment of confidence variations for each value $a \in \Delta$ needs 5 computations of confidences in $\Theta(max(\#A, \#B))$: computing a reference confidence $conf(A \rightarrow B^{(a,a)}$ and 4 confidences with elementary transformations $conf(A \rightarrow B^{(a+/-1,a+/-1)}$. Therefore, discovering specific expansion and reduction values for a temporal constraint $\Delta = [min, max]$ can be done in $\Theta(5 * |Delta| * max(\#A, \#B))$ which is linear with respect to Δ. Moreover, maximal confidence variations have the following property.

Property 7 (Maximal Elementary gain/loss equality)
Let A and B be two interval-based streams. The following equalities stand for maximal elementary confidence variations for any dependency $A \rightarrow B$:

$$LG(a, a) = LL(a + 1, a + 1)$$
$$RG(a, a) = RL(a - 1, a - 1)$$

Proof. By definition, every interval of B (or its temporal shift) $[t_i, t_{i+1})$ has at least contiguous non-active intervals of length 1 at his both sides.Let us note BL (resp. BR) the interval stream containing all non-active intervals of B that are contiguous to every interval in B on the left (resp. on the right). $BL^{(a,a)}$ (resp. $BR^{(a,a)}$) is then the interval stream containing non-valid elementary intervals of $B^{(a,a)}$ that are contiguous on the left (resp. the rights) to its valid intervals. The following is a graphical representation of BL and BR for a shift value of a and $a + 1$.

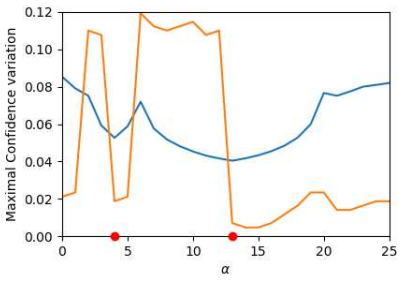


In this graphical representation, we can easily observe that the elementary confidence left gain for $A \rightarrow B^{(a,a)}$ is induced exclusively by $BL^{(a,a)}$ that contains $\#B$ intervals since maximal elementary confidence variations are assessed with respect to conclusion shifts (i.e (a, a)-transformations that preserves the original length and time lags between the conclusion' intervals). Therefore, the left gain $LG(a, a)$ equals to $\frac{\textbf{len}(BL^{(a,a)} \cap A)}{\textbf{len}(A)}$. It is also to notice that intervals of $BL^{(a,a)}$ correspond the starting elementary interval of each interval in $B^{(a+1,a+1)}$. Therefore, the left elementary loss for $A \rightarrow B^{(a+1,a+1)}$ is also induced by $BL^{(a,a)}$ and equals to $\frac{\textbf{len}(BL^{(a,a)} \cap A)}{\textbf{len}(A)}$. The result follows. The equality $RG(a, a)$ and $RL(a - 1, a - 1)$ can be proofed following the same reasoning. □

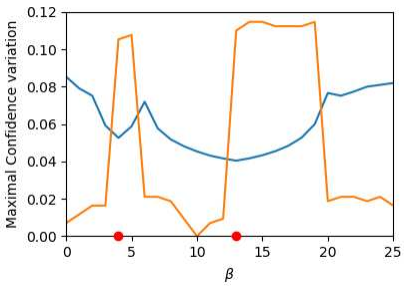
These equalities permits to reduce the cost of specific expansions and reductions in Δ to $\Theta(3*|\Delta|*max(\#A, \#B))$ as each value $a \in \Delta$ shares a confidence variation assessment with its direct successor and direct predecessor. The non-overlapping and removed intervals guarantee for temporal shifts permits also to infer the following equality:

$$\textbf{conf}(A \rightarrow B^{(a+1,a+1)}) = \textbf{conf}(A \rightarrow B^{(a,a)}) + LG(a, a) - RL(a, a) \quad (7.1)$$

This equality can be easily proofed similarly to the gain/loss equality property of maximal confidence variation. Using this equality, the discovery of specific expansions and reduction values in $\Delta = [min, max]$ needs the computation of 1 confidence reference for the first value in Δ, i.e $conf(A \rightarrow B^{(min,min)})$, as the following confidence



(a) Maximal confidence left gains



(b) Maximal confidence right losses

Figure 7.2.1: Maximal confidences variations computed for the example depicted in Figure 7.1.1 and corresponding statistical thresholds (blue lines). Red points correspond to specific expansion and reduction values.

references can be computed in $\Theta(1)$ given the precedent reference and its left gain and right loss. Therefore, the number of confidence computations needed for the exploration of Δ is $2 * |\Delta| + 1$. For instance, computed maximal confidence gains and losses for the example described in Figure 7.1.1 with $\Delta = [0, 25]$ are depicted in Figure 7.2.1. In the two figures, one can observe two specific temporal dependency 'characteristic forms' composed of 'significant' confidence variations that are clearly distinguishable from "non-significant" ones. These characteristic forms permits to detect specific expansions and reduction values corresponding to values in Δ where losses are "significant" and gains are not. These specific values are then used to build specific dependencies using temporal order: specific expansion 4 is associated with specific reduction 4 to obtain $A \rightarrow B^{(4,4)}$ and specific expansion 13 is associated with specific reduction 13 to obtain $A \rightarrow B^{(13,13)}$.

The last element we deal with in this section is how to assess maximal confidence variations. In [127], the refinement process is controlled by the dominance relationship that compares the reduction of the conclusion intervals to the loss in confidence measure. For maximal confidence variations, we propose to use a different approach based on the statistical validity test used for temporal dependencies. Indeed, the statistical assessment problem can be formalized as an interval-based streams correlation problem. For a dependency $A \rightarrow B^{a,a}$, the reference confidence value is computed with $A \cap B^{(a,a)}$. We aim to determine if the loss in intersection value while adding or subtracting a unit from expansion or reduction is statistically significant in a given observation duration $\mathcal{T}_{obs}$. This can be achieved with the assessment of the correlation between the stream $A \cap B^{(a,a)}$ and a stream, noted B^*, of length and size $\#B$ composed of added or removed conclusion intervals of length 1 that are produced by the elementary transformation. This is equivalent to calculate a statistical validity threshold for a dependency $(A \cap B^{(a,a)}) \rightarrow B^*$. The null hypothesis states that the intersection between $A \cap B^{(a,a)}$ and B^* are statistically independent. The contingency tables of the expected and observed outcomes can be derived straightforwardly from that used for temporal dependencies (cf. Tables 5.2.2 and 5.2.1). Statistical thresholds given by this approach are depicted in blue lines in Figure 7.2.1.

7.3 Interval Time Lag Discovery (ITLD) algorithm

In this section, we describe the Interval Time Lag Discovery algorithm (ITLD) that implements the exploration strategy introduced in the previous section. ILTD is described in Algorithm 4. It takes as parameters p the premise interval-based stream, c the conclusion streams, a temporal constraint $\Delta = [min, max]$ and an observation dura-

Algorithm 4: ITLD

Data: p, c: premise and conclusion streams, $\Delta = [min, max]$
T_{obs}: observation duration
Result: R : set of specific dependencies of the form $p \rightarrow c$

```
1  R ← ∅
2  G, L, TH ← ⟨⟩
3  reference = Conf(p, c, (min, min))
4  for a = min → max do
5      gain ← Conf(p, c,(a + 1, a)) - reference
6      Add gain to G
7      loss ← reference - Conf(p,c,(a, a + 1))
8      Add loss to L
9      Add th(len(p) * reference, #c) to TH
10     reference → reference + gain - loss
11 sigExp, sigRed ← GetSigVal(G, L, TH)
12 R ← R ∪ GetSpecDep(sigExp, sigRed) return R
```

tion $\mathcal{T}_{obs}$.

First, ITLD initializes the results set R as an empty set, G, L, TH, respectively, the maximal confidence left gains, maximal confidence right losses and the statistical thresholds, as empty sequences (lines 1 and 2). The reference confidence is also initialized with $\mathbf{conf}(p \rightarrow c^{(min,min)}$ which to be considered to compute confidence variations for $p \rightarrow c^{(min,min)}$ (line 3). In the main loop (lines 4 to 10) computes the maximal confidence left gain (line 5), the right loss (line 6) and the statistical thresholds for maximal confidence variations corresponding to temporal transformations (a, a) (line 7). These values are appended respectively to G, L and TH. It is to notice that a single validity threshold is computed for both gains and losses since the variation significance is calculated with respect to the same confidence. In the 8th line, ITLD updates the reference confidence value for the next iteration by applying equality 7.2.

Procedure GetSigVal(G, L, TH)

Data: G : expansion gains
L : reduction losses
TH : statistical thresholds
Result: sigExp, sigRed: temporally ordered specific values of expansions and reductions

```
1  sigExp, sigRed ← ⟨⟩
2  if L[0] > TH[0] then
3      Add min to sigRed
4  for i = 1 → max − min do
5      if G[i] < TH[i] and G[i − 1] > TH[i − 1] then
6          Add min + i to specExp
7      if L[i] > TH[i] and L[i − 1] < TH[i − 1] then
8          Add min + i to specRed
9  if G[len(G) − 1] > TH[len(TH) − 1] then
10     Add max to sigExp
11 return sigExp, sigRed
```

The second step of ITLD is the extraction of significant expansion and reduction values using the procedure *GetSpecVal* (line 9). This operation, done in $\Theta(|\Delta|)$, consists in verifying if each value in Δ correspond to a specific or loss using the statistical thresholds :

- For each element $i \in G$: $G[i] < TH[i]$ and $G[i-1] > TH[i-1]$
- For each element $i \in L$: $L[i] < TH[i]$ and $L[i-1] > TH[i-1]$

If the gain (res. loss) value at i is specific, $min + i$ is added to *sigExp* (res. *sigRed*). The first value in L and last in G are tested with a relaxed version of the higher conditions to capture dependencies with potential specific values that are out of Δ:

- $L[0] > TH[0]$ means that the specific loss $l \leq min \Rightarrow min$ is added to *sigExp*
- $G[|G|-1] > TH[0]$ means that the specific gain $l \geq max \Rightarrow max$ is added to *sigExp*

For instance, if we consider a temporal constraint $\Delta = [0, 10]$ with the example depicted 7.2.1.a the real specific value 14 is out of bounds of Δ but statistically significant confidence variations are detected: *max* is considered as a specific gain to maintain this information.

Procedure GetSpecDep(sigExp, sigRed)

Data: sigExp, sigRed: temporally ordered specific values of expansions and reductions
Result: specDep: the set of specific dependencies

```
1  specDep ← {}
2  if |sigExp| == |sigRed| then
3      i ← 0
4      while i < |sigExp| do
5          (α, β) ← (sigExp[i], sigRed[i])
6          if Conf(A → B(α, β)) > th(len(A), len(B(α, β)) then
7              Add A → B(α, β) to specDep
8          i++
9  else
10     (α, β) ← (sigExp[0], sigRed[|sigRed| − 1]
11     if Conf(A → B(α, β) > th(len(A, len(B(α, β) then
12         Add A → B(α, β) to specDep
13 return specDep
```

The last step *GetSpecDep* (Procedure GetSpecDep) builds specific dependencies candidates using temporal order (first specific expansion with first specific reduction etc.) and tests if they are statistically valid (line 2 to 8). If an unequal number of specific expansion and reductions is found, the most general dependency (with the greatest expansion and lowest reduction, line 10 to 12) is considered and tested permitting to avoid losing temporal information.

7.4 Conclusion & Discussion

In this chapter, we proposed the Interval Time Lag Discovery Algorithm (ITLD) that is devised to discover quantitative temporal dependencies between two interval-based streams. It uses a maximal confidence variation

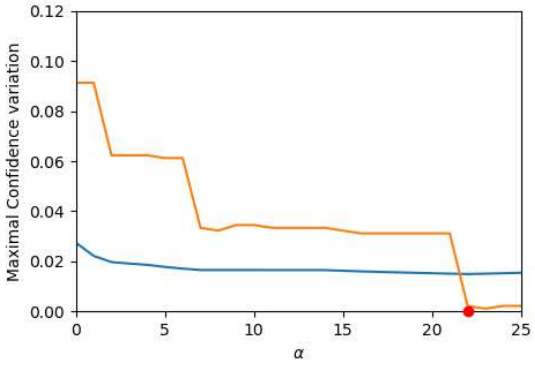


Figure 7.4.1: Maximal left gain with 3 temporal relations started by the same state

property to perform a linear exploration of a quadratic temporal search space defined by a temporal constraint Δ. We argued that this exploration strategy permits to obtain more specific temporal relations in cases where multiple dependencies stand between a pair of state streams in comparison with a semi-lattice level-wise exploration. We will evaluate our approach in the following chapter and compare it the existing methods.

In conclusion, we discuss several ways to extend our work and open questions that might worth attention. First, as evoked in the discussion section of Chapter 5, the information provided by ITLD or TEDDY [127] is not complete. It characterize the temporal information between intervals with two parameters: the time lag between the first endpoints and time lag between second endpoints. We believe that the maximal confidence variation exploration strategy might permit to obtain a piece of supplementary quantitative information about duration without additional computational cost. Typical durations of premise intervals involved in a temporal dependency might be extracted from the maximal confidence variation values. The width of characteristic forms of significant dependencies formed by equivalent confidence variation (e.g. Figure 7.2.1) can indicate the duration of premise intervals involved in a given dependency. However, this approach can not be straightforwardly applied for all cases, as we will see in the next chapter.

The second open question that remains unsolved by our approach concerns cases where multiple temporal dependencies are temporally overlapping. For instance, this case may occur when a state starts two different temporal relations. Figure 7.4.1 describes the maximal confidence gains for an example of such configuration where three characteristic forms can be observed.

Another idea that can be investigated is that of storage and query of temporal dependencies. The question posed is the following. Given distinct and two non-overlapping observation durations $\mathcal{T}_1$ and $\mathcal{T}_2$. Let us assume that ITLD provides $\mathcal{R}_1$ and $\mathcal{R}_2$ the sets of statistically valid pairwise dependencies for respectively $\mathcal{T}_1$ and $\mathcal{T}_2$. It is possible to infer the set specific dependencies and their confidences for $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ from $\mathcal{R}_1$ and $\mathcal{R}_2$ without running the temporal search space exploration? In other words, is it possible to perform aggregation of results given for a different period without loss of information? This would permit to build a temporal dependencies base that could be queried to obtain insight about an environment for different chunks of time without additional exploration cost. This idea can be extended to Complex Temporal Dependencies.

Finally, ITLD can be extended to be applied in a fully online setting in that maximal confidence values may be updated without computing confidences an overall sequence. More precisely, for a given dependency, the online extension of ITLD might maintain a vector of $|\Delta|$ values corresponding to maximal confidence variations that

should be updated given new intervals in the premise or the conclusion stream.

8 Evaluation

Contents

8.1 Introduction

This chapter aims to evaluate our approach devised to discover complex temporal dependencies between interval-based streams. We evaluate both the qualitative and quantitative performance of ITLD and CTD-Miner using synthetic data provided by a testbed tool that makes it possible to run a multitude of simulation scenarios. We also use real-world motion data generated from a sensor system composed of outdoor video cameras and using real-time video processing.

As stated in Chapter 4, two available approaches can be applied directly to discover quantitative temporal relationships between interval-streams. The first, PIVOTMiner [78], use an occurrence count-based assessment with a user-given interestingness parameter. This approach was designed for sequences databases but can be easily adapted to process streams as it is not endpoint sensitive. The second approach that we extend in this thesis, TEDDY [127], uses the statistical assessment on the intersection-based interestingness measure. We compare both qualitatively and quantitatively ITLD to these algorithms and analyze the efficiency of their use in CTD-Miner. More specifically, the performance analysis presented in this chapter aims mainly to respond to the following general questions:

How do the intersection-based assessment compare to count-based assessment in terms of discovery of temporal phenomena ? We argued in Chapter 5 that intervals intersection may provide a better assessment of temporal relationships compared to occurrence counting-based interestingness measure. We aim hereafter to provide experimental evidence to this claim.

How qualitatively efficient is our approach ? We proposed in Chapter 7 the maximal confidence variation exploration strategy and its implementation, ITLD, in the aim of using it in CTD-Miner process to enhance both execution times and accuracy. In this chapter, we study whether this approach permits to enhance the quality of results for pairwise and multiple state dependencies in comparison with existing approaches and w.r.t different datasets' characteristics. Also, we study whether pruning techniques used in CTD-Miner affect the quality of the provided results.

How effective are CTD-Miner and ITLD in terms of execution time? We evaluate the execution time of ITLD w.r.t to different datasets' characteristics and compare it to the quadratic baseline algorithm and TEDDY that performs a semi-lattice level-wise exploration using pruning techniques. Also, we evaluate the behaviour of CTD-Miner with the available time lag discovery approaches.

Is our approach robust to noise? We analyze if our approach is capable of discovering accurate temporal dependencies with noisy data.

At what extent do our approach scale? A straightforward rule of thumb used in the analysis of data streams is the following: the analysis execution time must be lesser than acquisition time. We aim to evaluate the number of state streams that can be processed thanks to our approach within this rule.

All algorithms used in this chapter were implemented[1] in Python and tested on a Core i7 2.1Ghz with 8GB memory running Windows 10.

[1] Implementations are available at: https://github.com/AElOuassouli/Quantitaive-Interval-Stream-Mining

8.2 Preliminaries

This section aims to define several terms we use in this chapter.

Stream density refers to the quantity of active interval length over the total observation duration $\mathcal{T}_{obs}$. This metric permits to quantify active activity in a given duration: a higher density means a high degree of activity. We emphasize that a high number of temporal phenomena occurrences induces dense streams, but dense streams do not imply straightforwardly high occurrence numbers. For a stream A the density of a stream is given by $\frac{\textbf{len}(A)}{\mathcal{T}_{obs}}$.

Temporal Variability refers to slight variations in temporal phenomena in terms of quantitative temporal aspect. For instance, pedestrians form a "cluster" of actors that similarly perform a trajectory. This behaviour can be reported on by a typical temporal pattern that does not precisely describe each exact occurrence of pedestrian trajectory quantitatively but is representative of the general behaviour of the pedestrians "cluster".

Noise in interval-based stream. Interval-based streams support binary information. A given time unit is whether active or inactive. A noisy interval-based stream is a stream that includes a certain amount of meaningless information including be "positive" noise, i.e. additional active intervals, and data corruption, i.e. deactivation of true active intervals.

Operations on state streams. This term refers indistinctly to operations used to compute dependencies' confidence values, including intersection and temporal transformations. These two operations are linear with respect to the number of streams' intervals.

8.2.1 Matching conditions and accuracy metrics

Knowledge discovery algorithms are hard to assess qualitatively in real-world contexts due to the *unknown* property of the potentially discovered insight. Generally speaking, two general approaches can be used in this context. The first, that is used extensively with *machine learning* methods, is to train a model on a portion of the datasets and verify if this model has enough predictive power in another portion (that can be novel data). The second is to use datasets describing known configurations with known ground truth (i.e. the model to be discovered is known) and compare it to the provided results. In this chapter, we use this second approach as it tackles the quality assessment problem directly. The study of the predictive power in our context is a different problem inducing more parameters that can be tackled in further work (does the activity is constant in a given time span ? at what extent false prediction can be used for concept drifts detection ?)

In order to assess the accuracy of algorithms, we use mainly use the F_1 score accuracy metric. It is defined as the harmonic mean of precision and recall. *Precision* describes the proportion of accurate results in the set of discovered dependencies: $precision = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$. *Recall* reports on the proportion of accurate discovered results in the set of accurate dependencies: $recall = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$. The F_1 score is used to seek good balance between precision and recall. It is given by:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

In order to differentiate between *positives* and *negatives* w.r.t the ground truth we use several matching conditions that are more or less restrictive. Given a true dependency $D_t = A_t \rightarrow B_t^{(a_t,\beta_t)}$ and a discovered dependency $D_d = A_2 \rightarrow B_2^{(a_2,\beta_2)}$ we define the following matching conditions for temporal dependencies.

- Qualitative matching: D_t and D_d are equivalent if $A_t = A_d$ and $B_t = B_d$.
- Exact matching: D_t and D_d are equivalent if they match qualitatively and $(\alpha_t, \beta_t) = (\alpha_d, \beta_d)$.
- Relaxed θ Matching: for $\theta > 0$, D_t and D_d are equivalent if they match qualitatively and $|\alpha_t - \alpha_d| \leq \theta$ and $|\beta_t - \beta_d| \leq \theta$.

The qualitative matching assesses the accuracy of a discovered dependency regarding the presence of a correlation between dependencies: *is the algorithm capable of reporting on qualitative correlations ?*. The exact matching reports on the ability to discover the exact truth: *is the algorithm capable of discovering the ground truth from the data ?*. The relaxed matching is a trade off between the restrictive exact matching and the permissive qualitative matching. It reports on the capability of discovering dependencies that are enough close to the ground truth in quantitative terms: *is the algorithm capable of discovering dependencies with approximative temporal information ?*

8.3 Performance study

8.3.1 Datasets description

Simulation tool

In order to generate synthetic datasets, we designed a graphical motion simulation tool[2]. A screen-shot of its user interface[3] is provided in Figure 8.3.1. This tool allows building motion simulation scenarios that can be run with regards to different temporal characteristics. More precisely, simulation scenarios are defined w.r.t spacial and temporal configurations. The spacial configuration definition consists of defining graphically an environment that is characterized by:

- A set of directed segments that model a spacial configuration. It models all possible paths that can be taken in a given environment by placing in the grid a set of nodes (black points in cells' centre) and defining directed segments (black lines) between pairs. For instance, a building configuration can be represented as follows: nodes represent specific places (e.g. rooms, intersection) and segments corridors between these specific places maintaining the distance information (e.g. corridor between the kitchen and the entrance hall). All segments do not need to be connected.
- A set of motion sensors. In the grid, the user can place sensors (coloured squares) that will be used to report on motion in their area during the simulation. Active sensors are coloured in grey if they are active and in red otherwise.
- A set of trajectories. Given the set of segments representing all possible paths, the tool permits to define specific trajectories for the simulation as an ordered set of connected segments (each segment is connected to its direct successor and predecessor).

The graphical definition of an environment configuration is used to define a set of behaviours composing a simulation scenario. A behaviour is defined with the following parameters:

- A trajectory t

[2]https://github.com/AElOuassouli/Quantitative-Interval-Stream-Mining/tree/master/Simulation%20Tool
[3]A graphical library developed by Lionel Robinault was partly used in the implementation of the user interface.

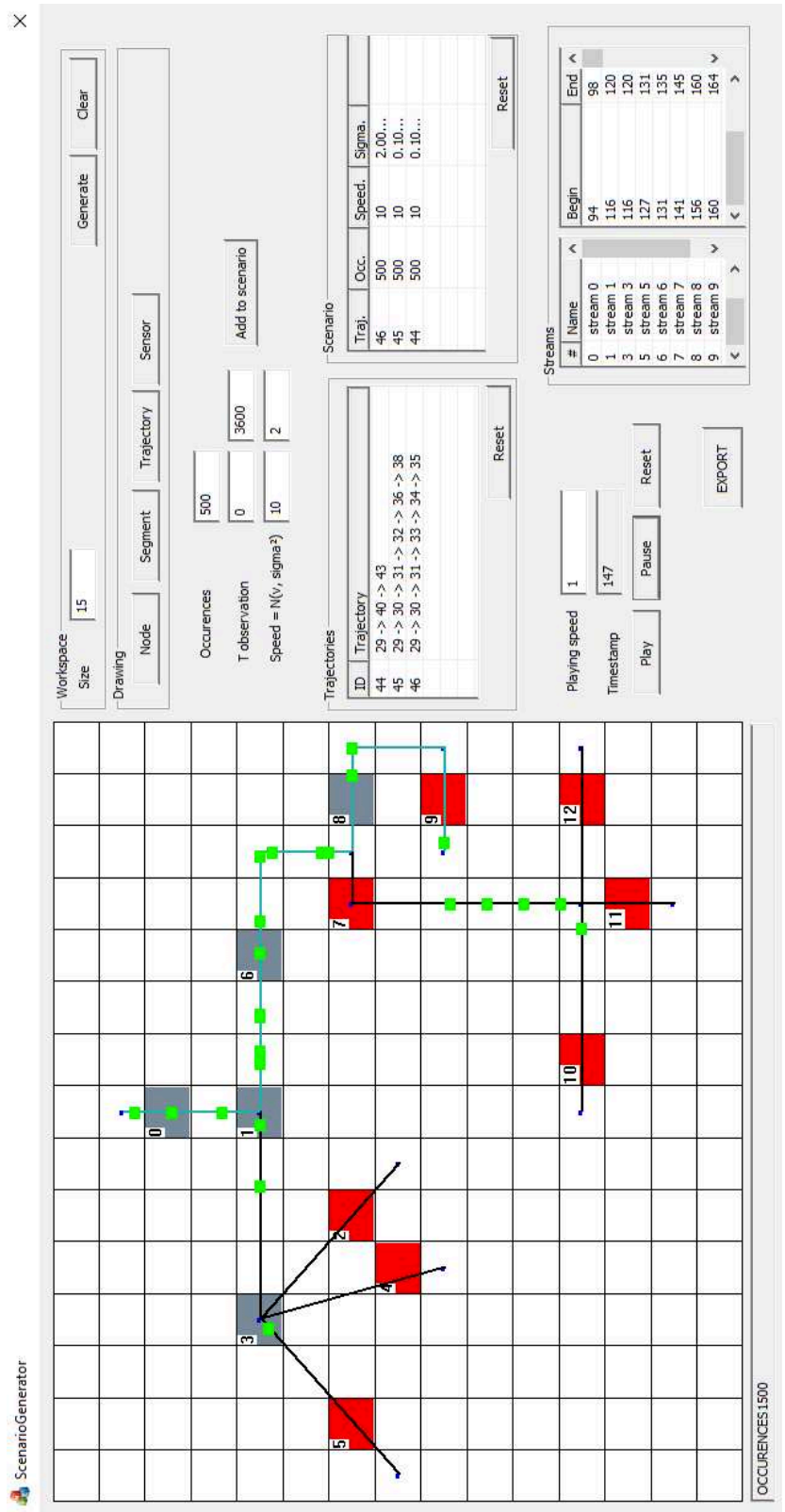


Figure 8.3.1: The synthetic data generation tool

- An activity time span $\mathcal{T} = [t_b, t_e)$.
- An occurrence number. It defines the number of times the trajectory t will be taken during $\mathcal{T}$. An occurrence corresponds to an actor instance performing a trajectory represented graphically by green squares.
- A speed distribution. In real-world contexts, the same phenomenon can occur with a slight temporal variation. For example, two pedestrians can perform the same trajectory with slightly different velocities. Inspire by [156], we modelled the velocity of a specific behaviour as a normal distribution $V \sim \mathcal{N}(v,\ \sigma^2)$ with v a typical occurrence speed and σ^2 the variance of the distribution.

Given a set of behaviours, the simulation process used in this tool is described in the following. For each occurrence of each behaviour, the simulator creates an occurrence instance. It computes its starting time stamp in the behaviour's activity time span $\mathcal{T}$ w.r.t a to a uniform distribution[4] and its velocity v w.r.t to the normal distribution specified in the parameters. Given the starting timestamp and the velocity v, the simulator iterate over time, time unit per time unit, to compute the instance's next pixel coordinates (i.e. positions in the absolute graphical area, not coordinate in the grid). In this process, if an actor instance enters an empty sensor's S area at timestamp t_b, an interval is initialized with its first endpoint $[t_b, -)$. If the instance leaves the sensor area at t_e the last interval is complete $[t_b, t_e)$ and added to the interval stream of S. This process is repeated until reaching the final point of the trajectory. The result of this process is a set of interval-based sequences corresponding each to a motion sensor. This simulation process can be played in the graphical area.

subsubsectionGenerated datasets

We describe hereafter the generated datasets used in this section.

Linear Trajectory. Table 8.3.1 describes a set of datasets obtained from the simulation tool that will be used in this chapter. We defined a linear trajectory with ten equidistant sensors, as shown by Figure 8.3.2. In the rest of this chapter, we will refer to the state corresponding to the sensor with index i in as $Sensor_i$. We ran 11 simulations varying the number of occurrences for the same duration $\mathcal{T} = 10000$ and object speed (cf Table 8.3.1). Each dataset contains ten state streams. The typical time lag between successive state is $\approx (4, 4)$. As shown in Figure 8.3.3 intervals number increases with occurrences for sparse data (100 to 5000 occurrences) and decreases for high-density event occurrences due to intervals overlap. Overlapping intervals correspond to situations where two actor instances pass simultaneously, with a given delay, through a sensor area. In the other hand, active length always increases when the event occurrences increases and ranges between 170 and 7850.

[4]The uniform distribution is used for simplicity and is sufficient for our experimental purposes. A more neat approach is to consider arrival times (occurrence starting timestamps) as a Poisson process. The generation process even more sophisticated if

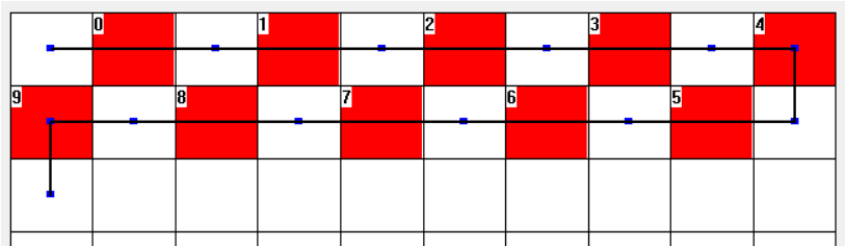


Figure 8.3.2: The linear trajectory used to generated synthetic datasets. It is directed from sensor 0 to sensor 9. (Screenshot from the simulation tool)

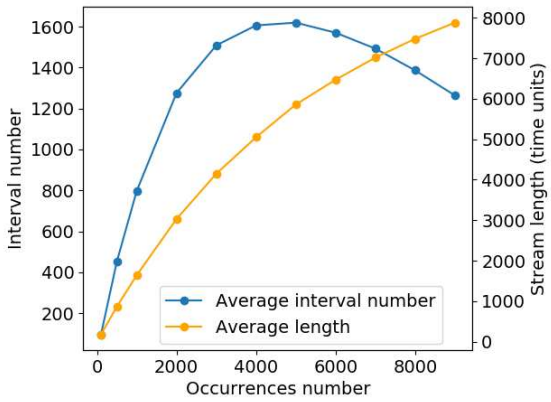


Figure 8.3.3: Average stream length and interval number with respect to occurrence number in the Linear trajectory datasets described in Table 8.3.1

Table 8.3.1: 11 simulated data sets with increasing occurrences number (*Occ*). *#Int*: average number of intervals per stream, *Len*: average streams length *Den*: average density % of $\mathcal{T}$. $\mathcal{T} = 10000$ time units

Occ	#Int	Len	Den
100	95	170	1.7%
500	452	870	8.7%
1000	798	1640	16.4%
2000	1275	3030	30.3%

Occ	#Int	Len	Den
3000	1509	4150	41.5%
4000	1606	5050	50.5%
5000	1619	5860	58.6%
6000	1570	6471	64.5%

Occ	#Int	Len	Den
7000	1492	6990	69.9%
8000	1386	7460	74.6%
9000	1263	7850	78.5%

Linear trajectory with temporal variability. We generated a set of datasets using the same trajectory as above but with temporal variability. The speed of occurrences was set to $v = 10$, observation duration to $T_{obs} = 10000$ and an occurrence number to 1000. We generated 17 datasets corresponding to a different variance σ values ranging between $[0, 2]$ containing streams with an average density of 25% (considered as sparse streams).

Linear trajectory with noise. We simulated a scenario composed of the same linear trajectory sensed by 10 equidistant sensors for $T_{obs} = 10000$ and occurrence number of 1000 and without temporal variability $\sigma^2 = 0$. We obtained a dataset with an average density of 16%. Starting from this dataset, we constructed 10 datasets by adding a uniform random noise in each dataset by introducing a proportion $\tau \in \{0.01, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4\}$ of false information (activating inactive intervals and *vice-versa.*). We describe in Figure 8.3.4 the number of intervals w.r.t their durations in the first stream of the linear trajectory in the generated datasets. We can observe that the noiseless dataset contain mainly intervals of 2 time units and no intervals of duration 1. Intervals of length 1 correspond to spurious intervals corresponding to mainly additional noise intervals. It is also to notice that the proportion of noise intervals of length 1 exceeds number of intervals of length 2 considered as true active intervals for $\tau > 10$.

8.3.2 Interval Time Lag Discovery (ITLD) assessment

This section aims to evaluate ITLD our quantitative time lag discovery algorithm and compare it to the two existing approaches: TEDDY [127] and PIVOTMiner [78]. All these works are devised to discover pairwise quantitative relationships between interval-based data and models quantitative information as a pair (α, β).

we consider some correlations between occurrence (e.g. people tends to walk in pairs)

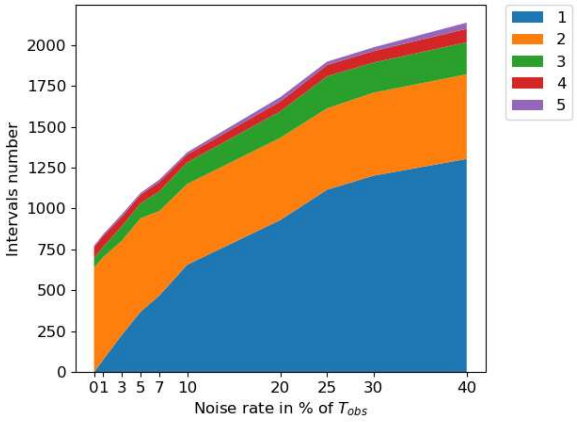


Figure 8.3.4: Number of intervals w.r.t to their durations for $Sensor_1$ in the Linear Trajectory with noise datasets

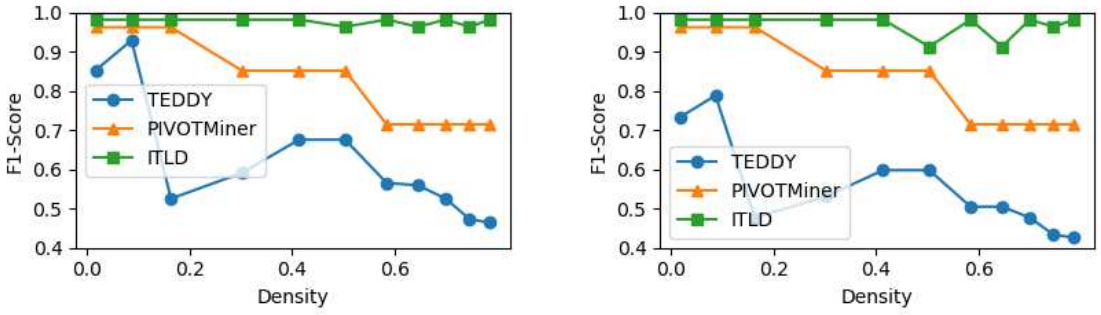
PIVOTMiner was designed for sequences databases and uses occurrence counting minimum support as an interestingness measure. To the best of our knowledge, it is the only approach that can be easily adapter to discovering quantitative relationships between interval-based streams. The approach used in this algorithm is not endpoint sensitive in that it is based on the analysis of relative quantitative relationships rather than absolute positions in a given timeline. As described in Chapter 4, it uses a geometrical approach consisting in projecting intervals $[t_b, t_e)$ in a bi-dimensional space (first endpoint t_b, second endpoint t_e) and run a clustering algorithm to detect typical quantitative information. In order to fairly compare it with TEDDY and ITLD, we adapted this algorithm to interval-based streams and included the temporal constraint $\Delta = [min, max]$ as for TEDDY and ITLD. Parameters used in PIVOTMiner version we use in this work are described by the following:

- $\Delta = [min, max]$ the temporal constraint
- *min-supp*: The minimum support threshold
- ε: a neighbouring range for the density-based clustering algorithm (DBSCAN)

In the following, we evaluate the quality of results as well as the performance of the three available algorithms of quantitative temporal pattern mining algorithms. More precisely, we analyse their robustness to streams' density, temporal variability and noise. We also evaluate their performance with regards to numbers of streams intervals and temporal constraint.

Robustness to streams' density

We executed ITLD, PIVOTMiner and TEDDY a temporal constraint of $\Delta = [0, 15]$ on the *Linear Trajectory* datasets. We firstly used $\varepsilon = 1$ and *min-supp*$= \frac{2}{3}$ for PIVOTMiner. The obtained F_1 scores are reported in Figure 8.3.5 for qualitative and exact matching.



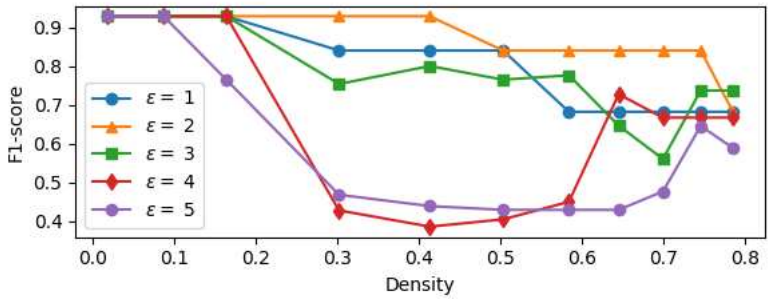
(a) Qualitative Matching **(b)** Exact Matching

Figure 8.3.5: F1-Scores w.r.t to density of streams

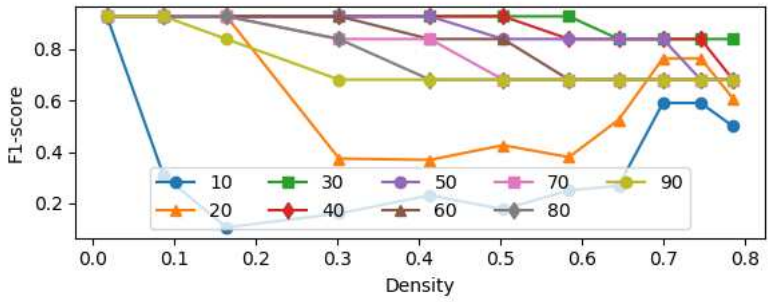
This experiment shows that ITLD is robust to density even for exact matching, in comparison with PIVOTMiner and TEDDY. This latter can detect significant dependencies (high recalls) but outputs a significant amount of false positives impacting the results' precision. We observed that these false positives often have large temporal transformations causing a significant increase in intervals lengths and intervals overlaps, particularly with dense datasets. As a consequence, large intersections are obtained, and by extension high confidence values, which can lead to statistically valid dependencies. Large temporal transformations deform, to some extent, the original linear temporal information that is to be discovered. Besides, large temporal transformations induce intervals overlap and fusion, which lead to a lower number of intervals in the conclusion stream. As a consequence, the dominance-based pruning is less efficient in such cases as the observed loss/gain in confidence measure may not include the contribution of all intervals in the original stream. ITLD tackles this problem as it considers confidence variations for non-deformed intervals: number and lengths of the conclusion is maintained.

With the specified parameters, PIVOTMiner behaves well for sparse data and gives similar results to ITLD, but lower recalls are obtained for dense data. This is explained by the fact that overlaps caused by great densities cause a fall in interval number and by extension, the occurrence count of interval relationships. Thus, PIVOTMiner requires to lower the minimum support, which may cause a pattern flooding problem, or a more permissive ε value (i.e. the distance between points for the clustering algorithm). For a more fair comparison, we executed PIVOTMiner varying ε and the minimum support. The results are reported in Fig. 8.3.6. Even for the best ε value ($\varepsilon = 2$), PIVOTMiner is sensitive to density. Low minimum supports causes pattern flooding (10%, 20%), and the best qualitative results are given for 30%. These results show that using automatic statistical tests permits to adapt the significance threshold to activity intensity, contrary to user-given significance thresholds approaches requiring continuous parameter tuning.

We also report in Figure 8.3.7 the observed execution times of ITLD, TEDDY and PIVOTMiner. What is evaluated here is the efficiency of these algorithms with regards to streams interval number. This figure shows that ITLD is slightly outperforming TEDDY for these datasets. While ITLD benefits from its linear complexity with respect to interval numbers and temporal constraint, TEDDY's process, that is quadratic w.r.t temporal constraint



(a) ε variation. *min-supp* $= 2/3$



(b) *min-supp* variation. $\varepsilon = 1$

Figure 8.3.6: F1 scores (qualitative matching) for PIVOTMiner

and linear in numbers of intervals, is enhanced by its early pruning techniques. On the other hand, PIVOTMiner that is quadratic w.r.t the number of intervals is slower than the other two algorithms even with the temporal constraint.

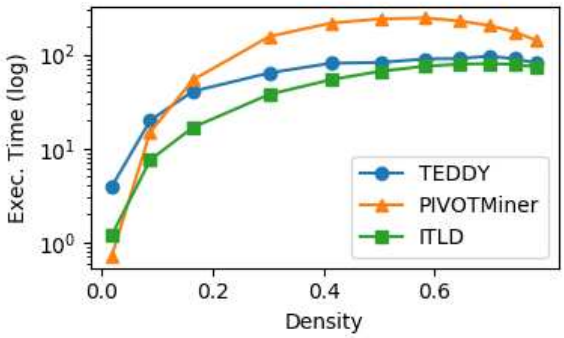


Figure 8.3.7: Execution Time w.r.t density of streams

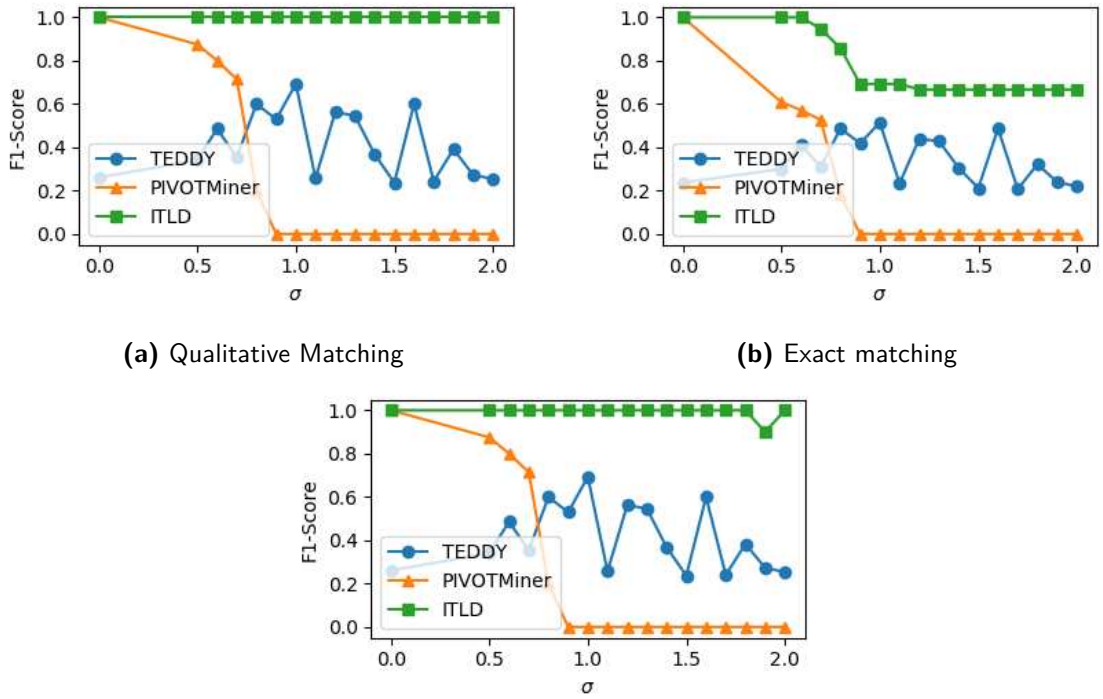
Conclusions

- ITLD is robust to streams density
- The use of the intersection-based confidence measure and a statistical interestingness assessment permits to obtain, at least, precise results in comparison with using user-given thresholds on occurrence counting supports.

- The maximum confidence variation approach allows ITLD to obtain results with betters recalls in comparison with TEDDY.
- ITLD's performances with respect to stream's density show the better trade-off between accuracy and execution time for this dataset. w

Robustness to temporal variability

We executed each algorithm with a temporal constraint $\Delta = [0, 15]$. We also used $\varepsilon = 1$ and a minimum support of $\frac{2}{3}$ for PIVOTMiner. Figure 8.3.8 reports on F1-score obtained for the **Linear trajectory with temporal variability** datasets for qualitative, exact and relaxed matching ($\theta = 2$).



(a) Qualitative Matching

(b) Exact matching

(c) Relaxed matching $\theta = 2$

Figure 8.3.8: F1 scores over velocity variance σ

The resulting F_1 scores show that ITLD is the most robust to temporal variability for all matching approaches. It is to notice that even if ITLD is less accurate for high variance values given the exact matching approach, it provides results that are close to the ground truth (or typical behaviours with quantitative information close to the average expected value). This can be observed with the relaxed matching with $\theta = 2$. We describe in Figures 8.3.9.a and Figure 8.3.9.b maximal confidence variations for a dependency $Sensor_1 \rightarrow Sensor_2$ such that $Sensor_1$ is directly followed by $Sensor_2$ in the linear trajectory with $\sigma^2 = 0$ and $\sigma^2 = 2$. What can be noticed is that the characteristic form of specific temporal dependencies is "flat" without temporal variability and tends toward a Gaussian curve with $\sigma^2 = 2$. Also, the characteristic form has a slightly larger width with temporal variability (cf expansions in 8.3.9.b) causing the detection of a specific time lag value that is slightly different from that observed without temporal variability. This explains the decrease in the F_1 score with exact matching. Figure 8.3.9.c and Figure 8.3.9.d describe maximal confidence variations for $Sensor_2 \rightarrow Sensor_1$ and show that no specific

transformation value is detected: no dependency stands for B followed by A (the dataset was generated in a one way trajectory).

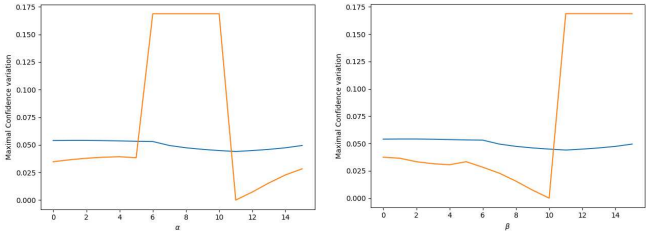
On the other hand, TEDDY provides a result sets with maximal precisions but low recalls for all variances σ^2 as shown in Figure 8.3.10 for a variance of $\sigma^2 = 0$. This is due to a large number of false positives. For example, for the successive first states of the linear trajectory $Sensor_1$ and $Sensor_2$, TEDDY provided five dependencies for $\sigma^2 = 0$: $A \rightarrow B^{(11,11)}, conf = 1$ corresponding to the ground truth and four other dependencies $B \rightarrow A^{(10,4)}, conf = 0.44$, $B \rightarrow A^{(11,5)}, conf = 0.45$, $B \rightarrow A^{(12,6)}, conf = 0.45$ and $B \rightarrow A^{(14,8)}, conf = 0.45$ that are false positives. Indeed, no occurrence of this temporal relationship was included in the simulation. The corresponding maximal confidence variations provided by ITLD are depicted in Figure 8.3.9.a and Figure 8.3.9.c.

PIVOTMiner accuracy decreases with temporal variability for all matching approaches even for a permissive minimum support of $\frac{2}{3}$. This is due to the clustering approach used in this algorithm. Increasing the variance of the speed normal distribution increases the distance between intervals projections in the (first endpoint, second endpoint) space. Thus, with a restrictive $\varepsilon = 1$, no cluster satisfies the minimum threshold for high variance values. Figure 8.3.11 describes results provided by DBSCAN with different variance and ε values for $Sensor_1 \rightarrow Sensor_2$ with an expected time lag of $(11, 11)$. Figure 8.3.11.a shows that DBSCAN provided a unique dense cluster (i.e red point at $(11, 11)$) for a null variance (no temporal variability) with an $\varepsilon = 1$. In this dataset, all intervals of $Sensor_1$ are related to an interval of $Sensor_2$ with an exact $(11, 11)$ time lag. With a variance equals to 2 (Figure 8.3.11.b), DBSCAN do not provide any frequent cluster. We can observe that the area surrounding the coordinates $(11, 11)$ contains a multitude of points due to temporal variability. This corresponds to the normal distribution of velocities we used to generate the dataset. In order to obtain results, we increased the value of ε and were not able to to obtain frequent clusters until an $\varepsilon = 4$ as shown in Figure 8.3.11.c. Another way to obtain results with PIVOTMiner is to lower the minimum threshold. For $\varepsilon = 1$ and $\sigma^2 = 2$, we obtained a frequent cluster for a *min-supp*$= \frac{1}{4}$ containing few of the intervals corresponding to the behaviour (cf Figure 8.3.11.d) and even multiple clusters for a minimum support of $\frac{1}{6}$.

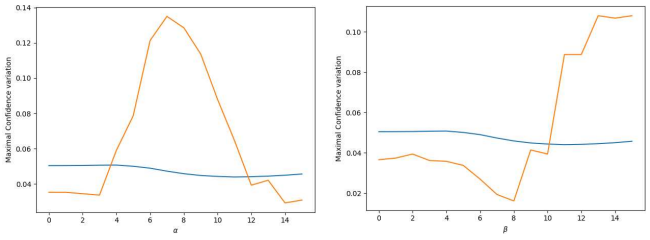
This experiment suggests that using user-given interestingness thresholds makes it harder to detect unknown temporal phenomena. Indeed, the user must have prior knowledge about searched temporal phenomena in order to set appropriate thresholds. For instance, in the results provided higher, knowing the approximate time lag distribution of the typical behaviour helps to set the ε parameter. However, this temporal distribution may evolve in time, requiring the adaptation of the algorithm's parameters. Besides, multiple temporal relationships, each with a different distribution (e.g. different variance values), may exist between pair of interval-based streams: finding each temporal relation may require several executions with different optimal parameters. This problem is somewhat solved by the statistical approach used by ITLD and TEDDY as interestingness thresholds are calculated adaptively (for each pair of state streams) w.r.t a statistical significance level that has to be set once by the user.

Conclusions

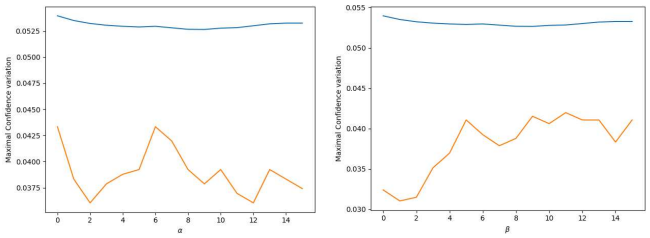
- This experiment suggests that ITLD is robust to temporal variability that follows a normal distribution.
- TEDDY has high precision, but low recall for the datasets used in this experiment: In comparison with TEDDY, ITLD provides a better trade-off between precision and recall.
- ITLD permits to obtain accurate results without user-given parameters contrary to PIVOTMiner. Using



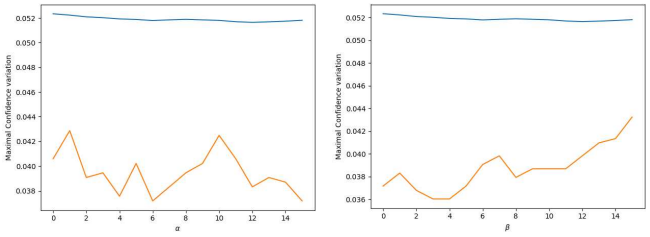
(a) $A \rightarrow B$ with $\sigma^2 = 0$



(b) $A \rightarrow B$ with $\sigma^2 = 2$



(c) $B \rightarrow A$ with $\sigma^2 = 0$



(d) $B \rightarrow A$ with $\sigma^2 = 2$

Figure 8.3.9: Maximal confidence variations for a pair of state streams A and B such that A follows directly B is the linear trajectory for $\sigma^2 = 0$ and $\sigma^2 = 2$

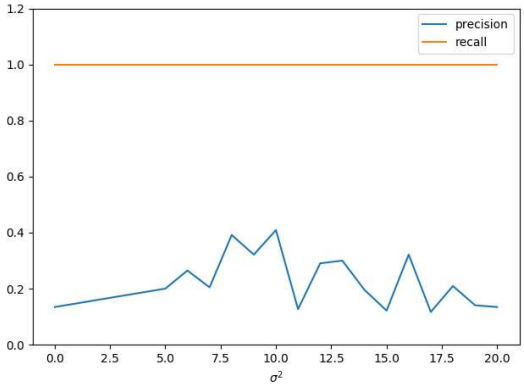
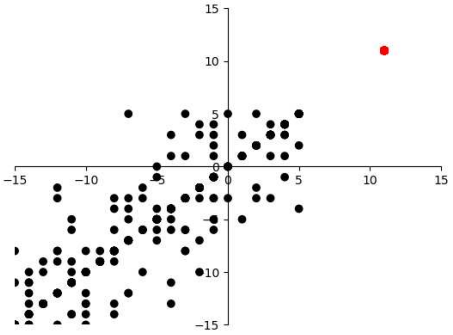
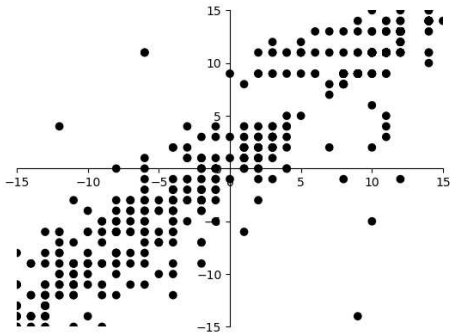


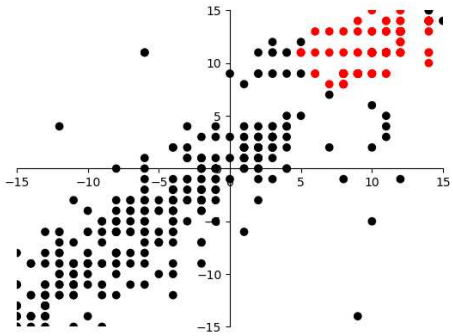
Figure 8.3.10: Precision and recall obtained with TEDDY for $\sigma^2 = 0$ and $\Delta = [0, 15]$



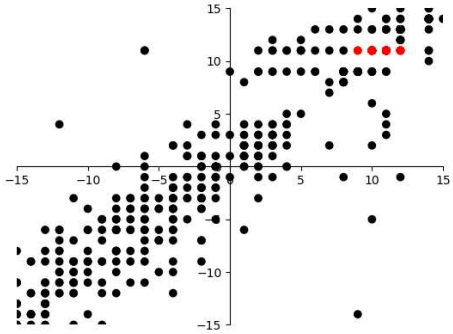
(a) $\varepsilon = 1,\ \sigma^2 = 0,\ \textit{min-supp} = \frac{2}{3}$



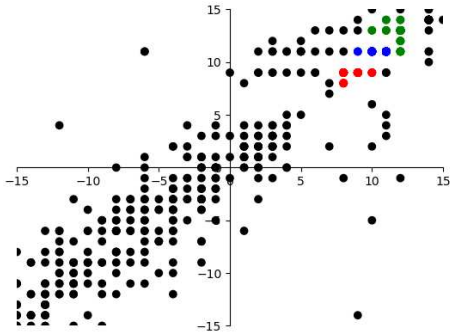
(b) $\varepsilon = 1,\ \sigma^2 = 2,\ \textit{min-supp} = \frac{2}{3}$



(c) $\varepsilon = 4,\ \sigma^2 = 2,\ \textit{min-supp} = \frac{2}{3}$



(d) $\varepsilon = 1,\ \sigma^2 = 2,\ \textit{min-supp} = \frac{1}{4}$



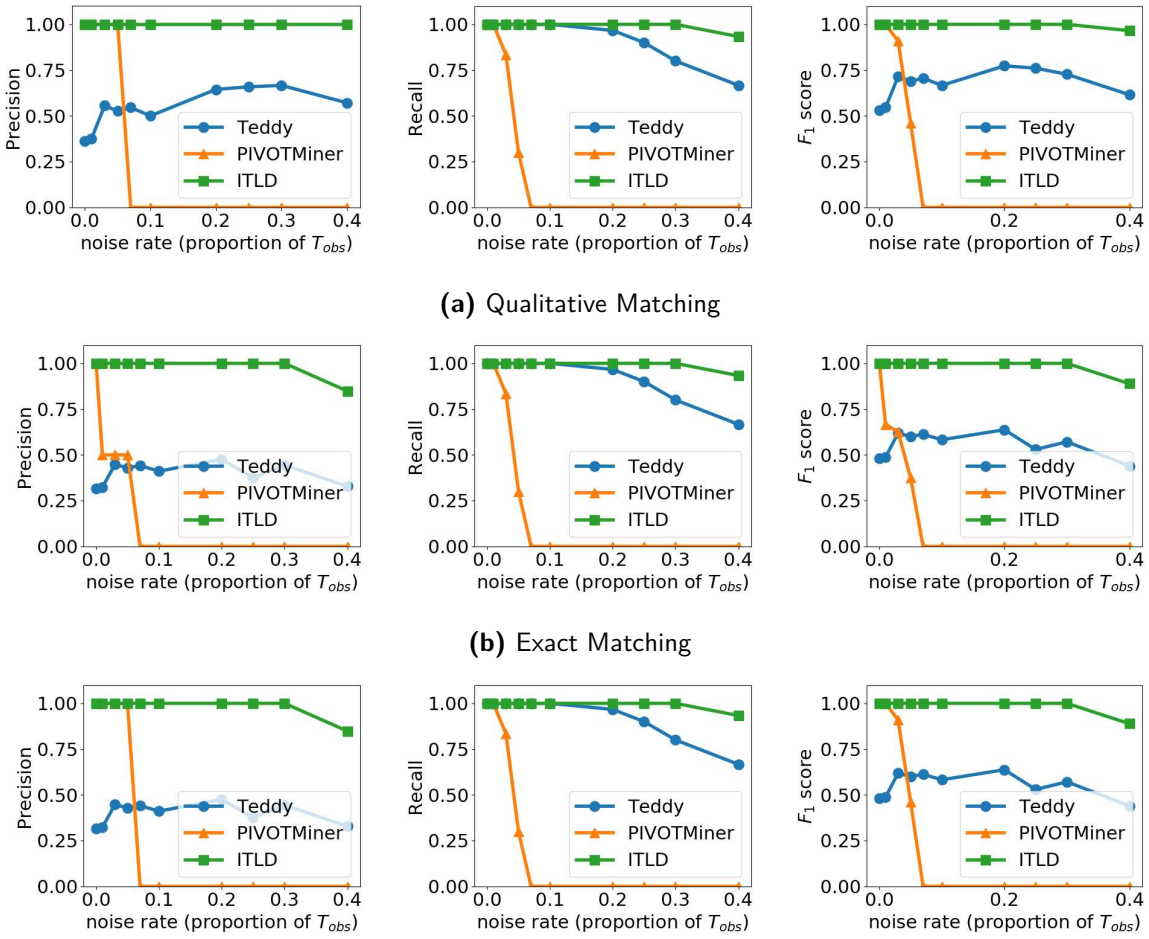
(e) $\varepsilon = 1,\ \sigma^2 = 2,\ \textit{min-supp} = \frac{1}{6}$

Figure 8.3.11: Clustering results given by DBSCAN for PIVOTMiner for different variance σ^2, ε and minimum support *min-supp* for an expected time lag of $(11, 11)$. Coloured points correspond to frequent clusters. $\Delta = [0, 15]$

statistical interestingness threshold is useful when no prior knowledge is provided about temporal phenomena to be discovered.

Robustness to noise

To evaluate the robustness to noise of our approach, we used the **Linear Trajectory with noise** dataset. We used $\Delta = [0, 15]$ and PIVOTMiner was executed with *min-supp*$= \frac{2}{3}$ and $\varepsilon = 1$. We report in Figure 8.3.12 on F_1 scores obtained for noise rates ranging between $[0.01, 0.4]$. Results provided by this experiment suggest that ITLD is more robust to noise in comparison with TEDDY and PIVOTMiner. ITLD's results score well in both precision and recall, even for a noise rate of 40%.



(a) Qualitative Matching

(b) Exact Matching

(c) Relaxed Matching. $\theta = 0.5$

Figure 8.3.12: Precision, recalls and F_1 scores w.r.t noise rate.

The PIVOTMiner algorithm was able to provide results with high precision for low noise rates w.r.t qualitative and the restrictive relaxed matching ($\theta = 0.5$). For these results, we observed that time lags given by PIVOTMiner were slightly different from the expected temporal information. Indeed, PIVOTMiner uses clusters centroids coordinates as representative time lag. Thus, with noise, the centroid coordinate derivative slightly from the expected result, causing a decrease in exact precision. For higher noise rates, PIVOTMiner was not able to provide satisfactory results with constant minimum support, as shown by the resulting recall scores. This is simply due to the constant the interestingness measure and clustering parameters as discussed higher.

As for previous experiments, TEDDY's precision is impacted by amounts of false positives. Thus, we focus on the current analysis on recalls on which TEDDY scores well usually. What can be observed is that the recalls of both TEDDY and PIVOTMiner decrease more with noise rates in comparison with ITLD. TEDDY's recalls decreases starting from a noise rate of $\tau = 0.2$ while ITLD's first recall and precision impact was observed for $\tau = 0.4$. We recall that the length of false information in state streams for $\tau = 0.4$ corresponds to 4000 that is higher than the average original streams length 1640.

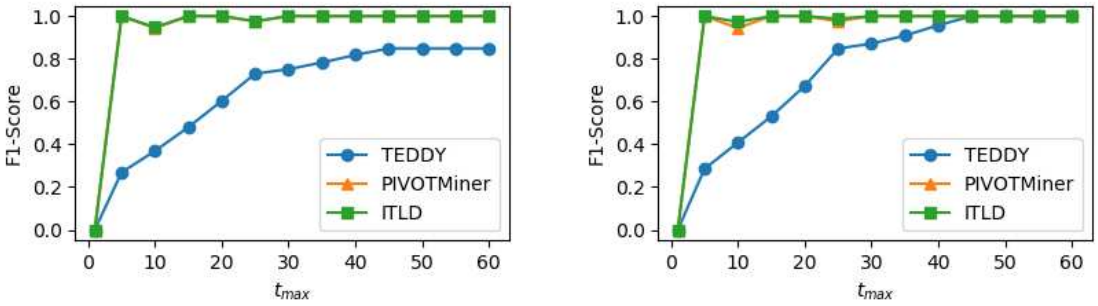
We also observed for the three algorithms a decrease of confidence and support values with noise rates. This is simply due to the number of spurious intervals introduced by noise in the streams (cf Figure 8.3.4. This also explains the empty results set provided by PIVOTMiner for high noise rates: the proportion of true active intervals decreases with high noise rates.

Conclusions

- This experiment suggests that ITLD is more robust to noise in comparison with TEDDY.
- The use of statistical assessment makes it possible to obtain accurate results with noisy data in comparison with user-given interestingness measures that need parameters tuning.

Influence of the temporal constraint

We used the 1000 occurrences dataset of *Linear trajectory* to evaluate the influence of the temporal constraint on the quality of results given by the three algorithms. These were executed for $\Delta = [0, max]$ with $max \in [1, 60]$. In this dataset, the expected temporal transformation between successive states in the linear trajectory is around $(4, 4)$. We report on F_1 scores in Figure 8.3.13.



(a) Exact Matching **(b)** Relaxed Matching $\theta = 1$

Figure 8.3.13: Execution time and F1-Scores w.r.t to t_{max} ($t_{min} = 0$)

As this streams in this dataset are sparse and noiseless, both PIVOTMiner and ITLD obtained maximum F1-Scores for reasons discussed higher. TEDDY obtains high recalls and a growing precision with t_{max} for this data set. That is partly explained by the proportion of true positives that grows with t_{max} due to the simulation scenario: large temporal constraint makes it possible to capture dependencies between a state and a larger set of its following states in the linear trajectory. TEDDY is capable of finding true positives but fails somehow to detect true negatives for $t_{max} < 30$. This may be due to the fact that TEDDY is designed to extract correlations based on intersection length and do not integrate the succession information directly. ITLD benefits from the intersection model assessment and, at the same time, takes into account directly temporal successions as it uses elementary confidence variations that can be seen as the number of impacted intervals. We also observed that

TEDDY provides less false negatives for large temporal constraints with $t_{max} \geq 40$, which suggests that the quality of results given by TEDDY are dependent on the temporal constraint.

The ground truth of the dataset used in this experiments contains several dependencies with $\alpha < \beta$. For such cases, ITLD and PIVOTMiner were able to provide a precise result. Due to its semi-lattice exploration strategy, results given by TEDDY contained multiple dependencies reporting on such cases. For example, with $\Delta = [0, 60]$, PIVOTMiner and ITLD provided $Sensor_7 \rightarrow Sensor_8^{(3,4)}$ while TEDDY's results contained two dependencies $Sensor_7 \rightarrow Sensor_8^{(3,3)}$ and $Sensor_7 \rightarrow Sensor_8^{(4,4)}$. In real-world scenarios, temporal relationships with $\alpha < \beta$ may exist in large numbers and/or with large $\beta - \alpha$ leading to a huge number of duplicated information and pattern flooding.

We report in Figure 8.3.14 on execution times w.r.t Δ. It shows that ITLD outperforms TEDDY: for $\Delta = [0, 60]$ ITLD runs seven times faster than TEDDY. This is due to the linear complexity of ITLD w.r.t the temporal constraint. As expected, running times of PIVOTMiner were marginally affected by the size of Δ as this constraint is simply used to reduce the number of vectors for the clustering step.

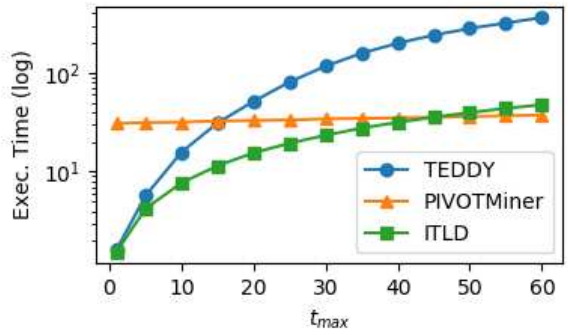


Figure 8.3.14: Execution Time w.r.t t_{max} in $\Delta = [0, t_{max}$

Conclusions

- The accuracy of ITLD is not affected by the temporal constraint Δ
- ITLD provides precise temporal information when $\alpha < \beta$ compared to TEDDY.
- ITLD outperforms TEDDY for large temporal constraints due to its linear complexity.

8.3.3 Complex Temporal Dependencies discovery

In this subsection, we evaluate the multiple states dependency discovery approach, CTD-Miner, that we introduced in Chapter 6. We examine the efficiency of its pruning criteria and the impact of the used time lag discovery algorithm. We will evaluate performances of CTD-Miner with both ITLD and TEDDY as they are the only available approaches using streams intersection as interestingness criteria and statistical assessment. We chose not to include PIVOTMiner in this evaluation as it uses occurrence counting support and requires additional user-given parameters. We also assess its performance w.r.t the influence of the temporal constraint and density of streams.

Hereafter, we will use ITLD as the standard quantitative pairwise dependency algorithm for CTD-Miner (ITLD is inserted at line 12 in Algorithm 1). Otherwise, it will be mentioned explicitly. For the need of our experiments, we will use different configurations of CTD-Miner:

- **CTD-Miner**: CTD-Miner as described in Algorithm 1.
- **Merging** Performs the incremental construction with the merging routine. Pre-pruning (line 7) is removed from Algorithm 1.
- **WP** (without pruning). Performs a simple incremental construction of Complex Temporal Dependencies given the Apriori-like assumption. (line 7 and 18 are removed from Algorithm 1
- **Pruning**: Merging dependencies (Line 18 in Algorithm 1) is removed from CTD-Miner. Only Pre-pruning is used to reduce the search space.
- **CTD-Miner-TEDDY**: CTD-Miner with TEDDY as a time lag discovery algorithm.

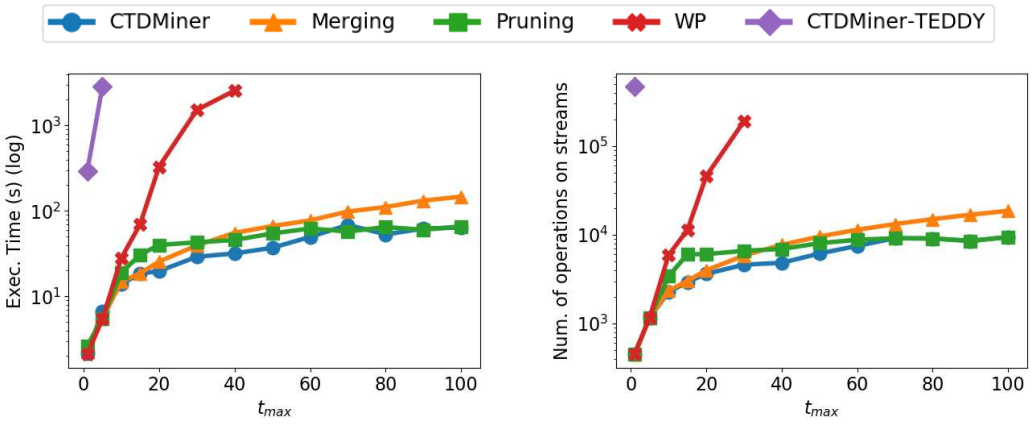
In order to assess the efficiency of these algorithms, we will use the number of state streams operations in addition to execution times. It indicates precisely the main cost of the exploration process, including the closed-like verifications for pruning and merging in addition to time lag discovery.

For all executions, we limited the execution of the CTD discovery algorithms to 40 minutes and used the statistical threshold for the correspondence relationship.

Influence of the temporal constraint

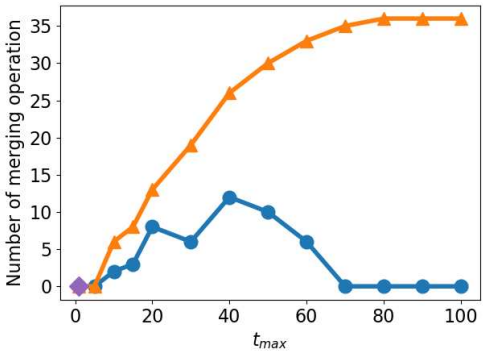
In order to evaluate the influence of the temporal constraint on the five version of CTD-Miner described higher, we use the 1000 occurrences stream of *Liner Trajectory - slow*. We first executed these algorithms with a list of streams ordered w.r.t their position in the trajectory $[Sensor_0, Sensor_1, Sensor_2 \ldots, Sensor_9]$ for $\Delta = [0, t_{max}]$ with $t_{max} \in [0, 100]$. The results of this experiments are reported in Figure 8.3.15.

First, we observe that *CTD-Miner* associated with one or both pruning techniques and *ITLD* as a time lag discovery approach (*CTD-Miner, Merging, Pruning, WP*) runs significantly faster than *CTD-Miner-TEDDY* for this dataset. *CTD-Miner-TEDDY* reached the 40 minutes execution limits for respectively $t_{max} = 10$ and $t_{max} = 50$. It suffers from the precision of TEDDY (cf. the previous section) that generate a large number of false positives: as discovered false positives at iteration i are considered as premises for iteration $i + 1$, the number of dependencies to be tested at each iteration increases exponentially. In Figure 8.3.15.b, we observe that *CTD-Miner-TEDDY* provided 12 results for $\Delta = [0, 1]$ while non is expected. This experiment shows that the precision of a time lag discovery approach is critical to a multiple state dependency discovery algorithm. Indeed, it depends largely on its precision score. We can also notice that the closed-like pruning and merging are not efficient in this case.

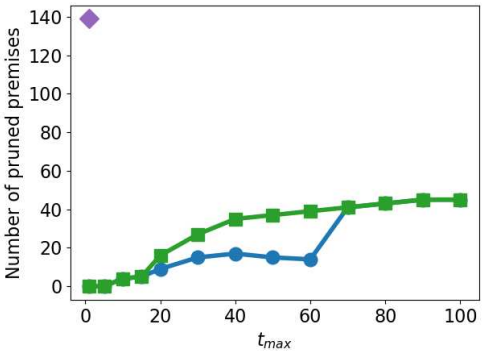


(a) Execution time

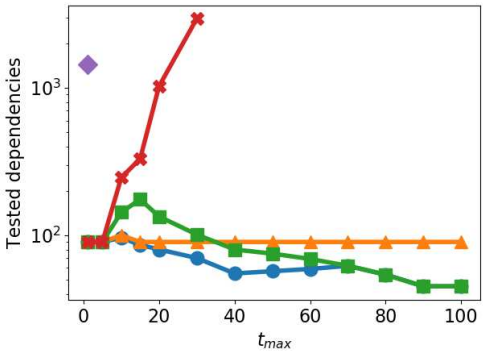
(b) Number of state streams operations



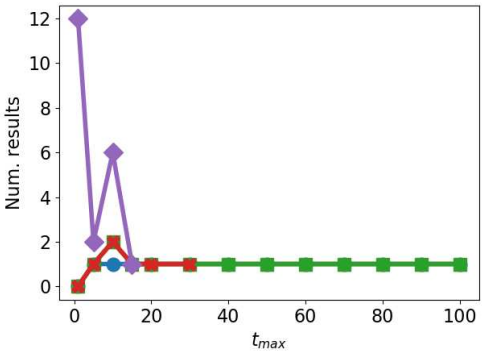
(c) Execution time



(d) Execution time



(e) Number of state streams operations



(f) Number of results

Figure 8.3.15: Execution times and number of results of the five versions of CTD-Miner w.r.t the temporal constraint $\Delta = [0, t_{max}]$.

This experiment suggests that the closed-like pruning permits to accelerate the discovery process significantly while not affecting the results' quality. We observe in Figure 8.3.15.a that *CTD-Miner* using pruning and/or merging outperforms significantly *WP*, that reached the 40 minutes execution limit for $t_{max} = 40$, with higher values of t_{max} (e.g a ratio of ≈ 50 in execution time for $t_max = 30$). In Figure 8.3.15.b reporting on numbers of state streams operations, one can notice that the number of executed state streams operations follows the same evolution of execution times and maintains the order of magnitude ratios. This shows that the efficiency of the CTD exploration is determined mainly by these operations execution number. We will focus on this performance metric as it does not depend on hardware specifications and configurations leading to variations in execution times[5]. In Figure 8.3.15.f, we observe that closed-like pruning criterion preserves the quality of the results. The single result provided by CTD-Miner with ITLD corresponds to the conditional relationship describing the linear trajectory.

This experiment permits also to compare the cost the closed-like based pre-pruning and merging w.r.t to the temporal constraint. In Figure 8.3.15.b, we observe that *Merging* is slightly more efficient than *Pruning* for $t_{max} \leq 30$ and inversely for $t_{max} > 30$. This is explained by the following.

Merging executes the time lag dependency algorithm 90 $(10*9)$ times to compute all pairwise dependencies in the first iteration as it can be seen in Figure8.3.15.e. The obtained candidate set contains all necessary information to compute, by dependencies merging, all necessary CTDs composing the conditional temporal relationship describing the entire trajectory without further iterations. For $t_{max} = 10$, this result set is exclusively composed of dependencies of the form $Sensor_i \rightarrow Sensor_{i+1}$ where $Sensor_{i+1}$ is the direct predecessor of $Sensor_i$ is the linear trajectory. As a consequence, the cost of merging dependencies in the result set is optimal. For larger values of t_{max}, the first iteration result contains additional dependencies. For instance, with $t_{max} = 20$, each sensor streams is temporally correlated with its two direct successors. A portion of the corresponding exploration tree is depicted in Figure 8.3.16. This number increases each time a state can be related to more dependencies. As a consequence, with larger candidates set provided by the first iteration, the merging process has a greater cost slightly as it can be observed in Figure 8.3.15.c. However, it remains negligible in comparison with the influence of t_{max} on the time lag discovery algorithm.

On the other hand, *Pruning* relies on pruning premise candidates. For instance, if a dependency $Sensor_0 \rightarrow Sensor_1$ is discovered, all dependencies of the form $Sensor_1 \rightarrow Sensor_i$ are pruned as $Sensor_1$ is a correspondent sub-dependency of $Sensor_0 \rightarrow Sensor_1$. This process is used in all iterations to prune the search space as it can observed in Figure 8.3.17. Also, the efficiency of *Pruning* increases with t_{max} as more premises can be pruned as indicated in the former figure with $t_{max} = 10$ and $t_{max} = 20$. As a consequence, number of time lag discovery algorithm executions, that are the most costly process in CTDs discovery, decrease with higher values of t_{max} as shown in Figure 8.3.15.d. This explains the relative over-performance of *Pruning* and over *Merging*.

The use of both closed-like premise pruning and dependency merging makes *CTD-Miner* more efficient in comparison with using a single search space reduction approach. As it can be noticed in Figure 8.3.15.e, CTD-Miner benefits from dependencies merging when premise pruning is not efficient (e.g $t_{max} = 10$ and $t_{max} = 15$) and from premise pruning when it permits to reduce the search space (for high values of t_{max}).

We also executed the four CTD-Miner versions with ITLD on a streams list that not consider temporal streams order to evaluate the order influence on the discovery process and pruning efficiency. First, we executed CTD-

[5]The correct use of execution times to compare algorithms with close performances requires multiple experiment executions to obtain a sufficiently significant assessment and avoid independent running time variations.

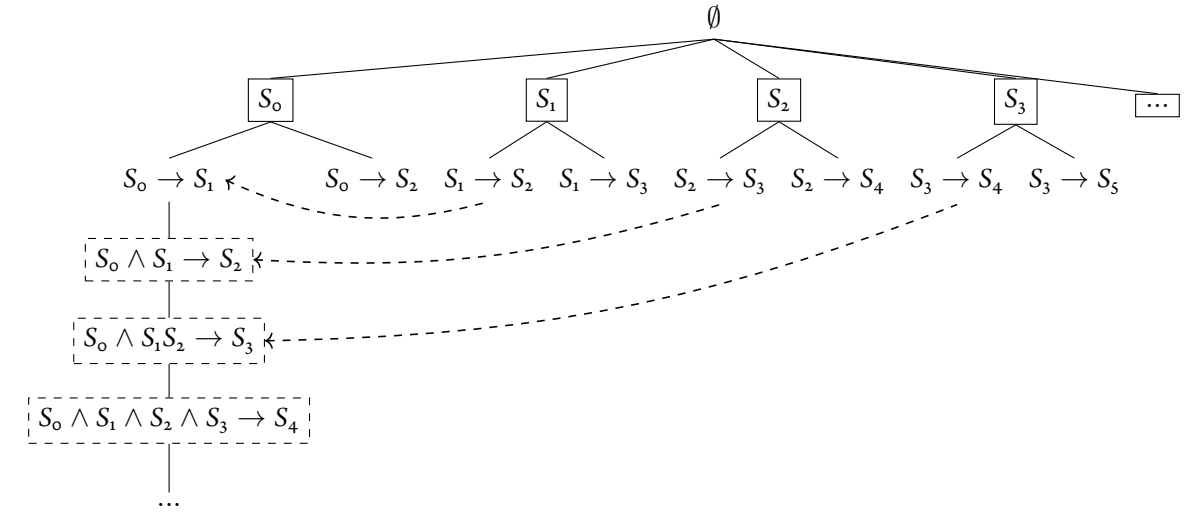
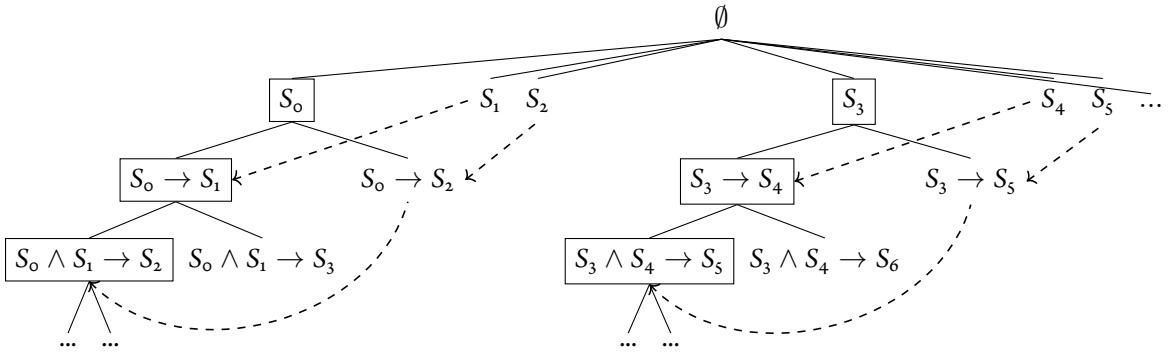
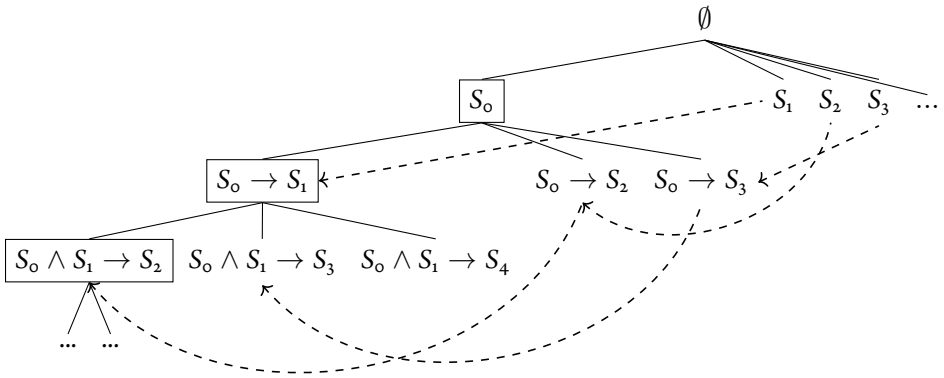


Figure 8.3.16: A portion of the exploration tree of *Merging* with $\Delta = [0, 10]$. Nodes correspond to dependencies. Solid frames refer to tested premises. Dashed frames indicate dependencies obtained by merging. Dashed arrows denote "merged with".

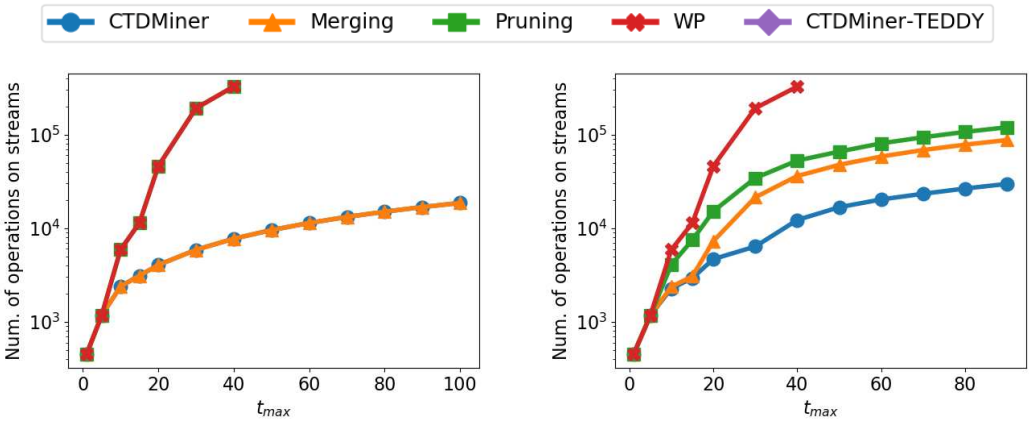


(a) $t_{max} = 10$



(b) $t_{max} = 20$

Figure 8.3.17: A portion of the exploration tree of *Merging* with $\Delta = [0, 10]$ and $\Delta = [0, 20]$. Nodes correspond to dependencies. Solid frames refer to tested premises. Dashed arrows denote "correspondent sub-dependency of".



(a) Reverse order **(b)** Random order. Average on 7 executions

Figure 8.3.18: Number of streams operation w.r.t t_{max} and input streams list order

Miner with the reverse order. Figure 8.3.18.a shows that in that particular case, *Pruning* is equivalent to *WP*: any premise pruning is not able to reduce the search space. In the other hand, CTD-Miner benefits from dependencies merging. Second, we executed seven times the CTD-Miner versions on a streams list with randomised orders. For all executions, we noticed that CTD-Miner outperforms both *Pruning* and *Merging* with all randomised orders. Figure 8.3.18.b describes the simple average of streams operations. This indicates that using both search space reduction approaches permits to enhance the scalability of CTD-Miner with regards to the temporal constraint size.

To conclude, our example shows that the use of a max-gap like temporal constraint permits to enhance the performance of our approach. This has two main advantages. First, interesting temporal phenomena are not bounded temporally. This is a useful property when the overall duration of interesting phenomena are not known *à priori*. Second, it permits to reduce the cost of the search space exploration considerably. For instance, with the synthetic we used in the subsection, it is not necessary to define a sizeable temporal search space, e.g. $\Delta = [0, 100]$, to capture temporally large temporal phenomena. For instance, using $\Delta = [0, 10]$ is sufficient to obtain a description of a trajectory that is executed in 100 time units.

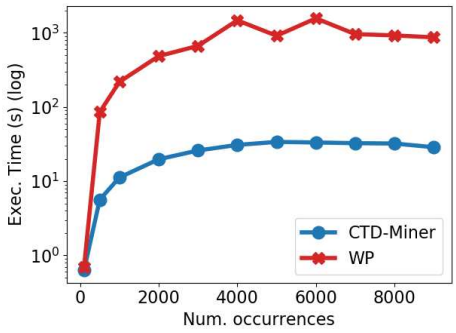


Figure 8.3.19: Execution times of CTD-Miner and *WP* w.r.t to occurrence number

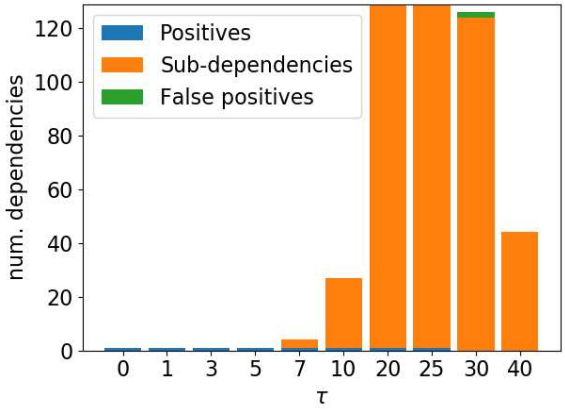


Figure 8.3.20: Number of conditional temporal relationships provided by CTD-Miner w.r.t to noise rate.

Robustness to density

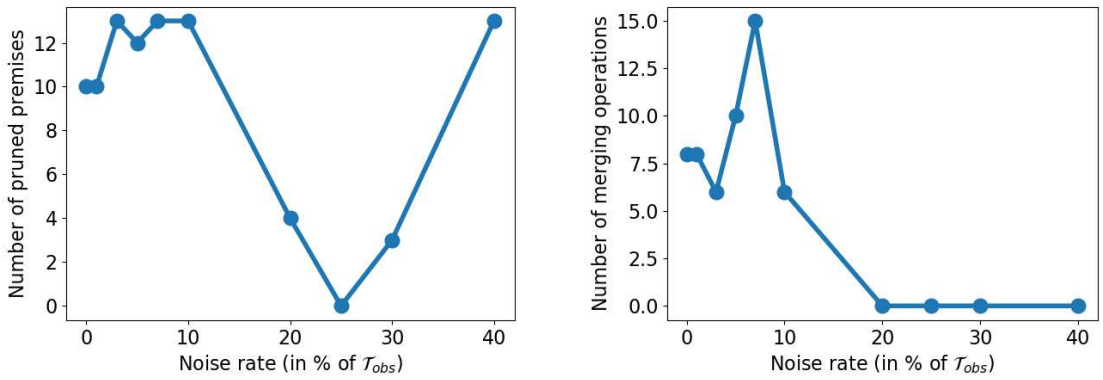
We executed *CTD-Miner* and *WP* on the Linear Trajectory datasets obtained with a variation of simulation number of occurrences for $\Delta = [0, 9]$. As expected, with this dataset, our approach, with and without pruning, was able to find the single conditional relationship with all dataset without providing any false positives. This is due to the fact that it uses ITLD as a time lag discovery algorithm that is itself robust to density.

This experiment also permits to evaluate the behaviour of CTD-Miner w.r.t number of interval. Execution times reported in Figure 8.3.19 shows that the use of premise pruning and dependencies merging permits to reduce execution times significantly for this dataset by up two orders of magnitude. This suggests that our approach can scale well with the number of intervals in a dataset.

Robustness to noise

The robustness to noise of CTD-Miner was evaluated with the Linear Trajectory with noise datasets with a constant temporal constraint $\Delta = [0, 9]$. We recall that the noise rate τ denotes the percentage of $\mathcal{T}_{obs} = 10000$ containing false information. For instance, if $\tau = 10$, 1000 elementary intervals in $\mathcal{T}_{obs}$ contains whether a spurious active interval or deactivated interval. We refer the reader to Figure 8.3.4 to intervals duration proportion in the used datasets.

We report in Figure 8.3.20 on the number of conditional temporal relationships provided by CTD-Miner for



(a) Number of pruned premises

(b) Number of merging operations

Figure 8.3.21: Number of pruned premises and merging operation in CTD-Miner w.r.t noise rate.

different noise rates. We distinguish between 3 types of results:

- True positives are results that well describes qualitatively the succession of states corresponding to the linear trajectory.
- Sub-dependencies are results that describes a portion of the trajectory qualitatively while maintaining the temporal order.
- Negatives are results including at least one non-correctly ordered relationship. For instance, if *ABCDE* correspond to the expected state order, *ABDCE* will be considered as a false positive.

This experiment shows that CTD-Miner was able to efficiently detect the entire trajectory (the true positive) up to a noise rate of 25%. Also, no false positive nor sub-dependency was given with a noise rate less or equal to 5%. Second, with noise rates greater than 7%, CTD-Miner provided sub-dependencies with relatively large amounts for 20%, 25% and 30% and a single false positive for 30%. The number of sub-dependencies shows that the correspondence relationship is less efficient with a large amount of noise. Spurious intervals and deactivated one are uniformly distributed over the observation duration. As a consequence, the resulting intersection between related states includes extra-length due to spurious intervals that are not related to any further streams and less intersection due to deactivated true active intervals. This results on a greater difference between representative streams of a dependency and its sub-dependencies. Thus, less premise pruning and merging operations can be performed. This can be observed in Figure 8.3.21.

To conclude comments on this experiment, we can argue that CTD-Miner robustness to noise is satisfactory with regards to the ratio of correct information over the amount of noise (spurious intervals + deactivated intervals). We also consider that obtaining portions of true temporal relationships as results for a large amount of noise remains acceptable. The resulting pattern flooding can be addressed in a pre-processing step that groups highly similar results. However, we believe that the use of the correspondence relationship can be improved with a better relaxation parameter other than the statistical threshold.

Robustness to temporal variability

One important aspect to evaluate is the robustness to temporal variability. We showed higher that ITLD permits to discover pairwise dependencies with temporal variability efficiently. We showed that temporal information

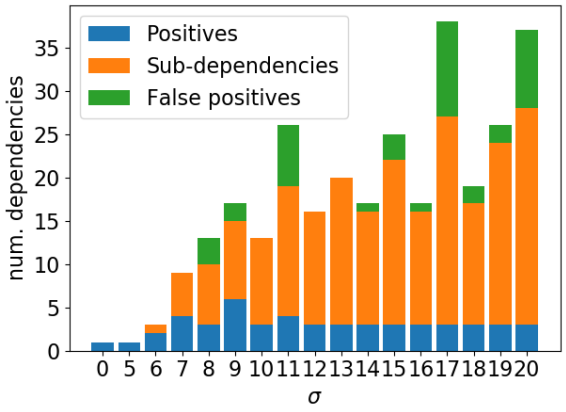


Figure 8.3.22: Number of conditional temporal relationships provided by CTD-Miner w.r.t to temporal variability

remains consistent with respect to the ground truth but with slight variations. One question that is posed for multiple state dependency discovery is: *How often slight variations in pairwise dependencies time lag information affect the incremental construction of complex temporal dependencies ?*.

We executed CTD-Miner on the Linear *trajectory with temporal variability* datasets with $\Delta = [0, 15]$ and report in Figure 8.3.22 on provided results. We use the same results classification as for the noise experiment. In this figure, we observe that CTD-Miner provided with non-negligible amounts of false positives with temporal variability. This is due to the detection of several dependencies by ITLD that do not coincide with the expected order. For example, the following dependency was discovered (for $\sigma^2 = 2$) with a confidence of 0.14:

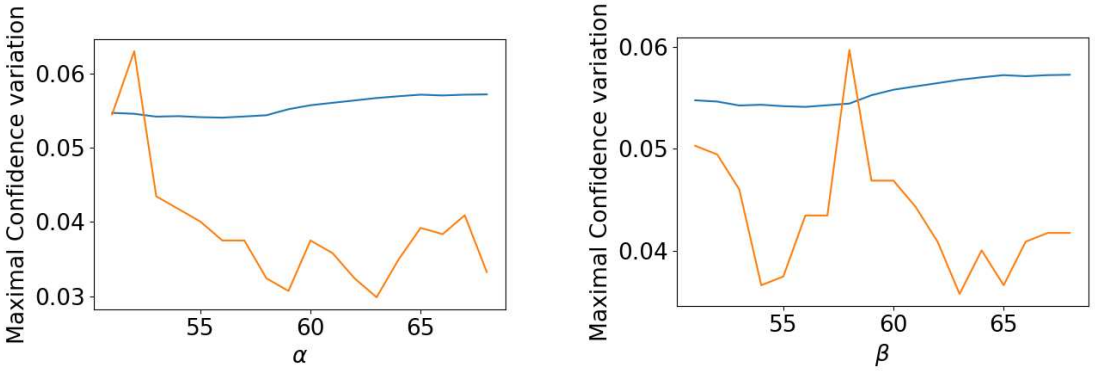
$$Sensor_1 \wedge Sensor_2^{(13,10)} \wedge Sensor_3^{(25,20)} \wedge Sensor_4^{(38,30)} \wedge Sensor_6^{(53,51)} \rightarrow Sensor_5^{(53,58)}$$

The corresponding maximal confidence variations are depicted in Figure 8.3.23. This can be explained by the following. Dependency $Sensor_1 \wedge Sensor_2^{(13,10)} \wedge Sensor_3^{(25,20)} \wedge Sensor_4^{(38,30)} \rightarrow Sensor_6^{(53,51)}$ was discovered with a confidence of 0.5. It do not capture the entire temporal relationship between $Sensor_4$ and $Sensor_6$ due the limiting temporal constraint. As a consequence, the resulting intersection takes into account active length of $Sensor_6$ that report on the fastest trajectory occurrences. Corresponding intervals intersect with small portions of intervals of $Sensor_5$ reporting on the slowest behaviours.

We also noticed that CTD-Miner was able to detect the entire trajectory for all variances σ. However, we observed that the linear trajectory was reported at least twice, with variation in temporal information, for a variance greater than 6. For instance, with $\sigma^2 = 2$, we obtained 3 conjunctive results describing the entire trajectory. They share the same prefix from $Sensor_1$ to $Sensor_5$ in terms of quantitative information but differ on the rest of the trajectory due to the detection of 3 dependencies with $Sensor_6$:

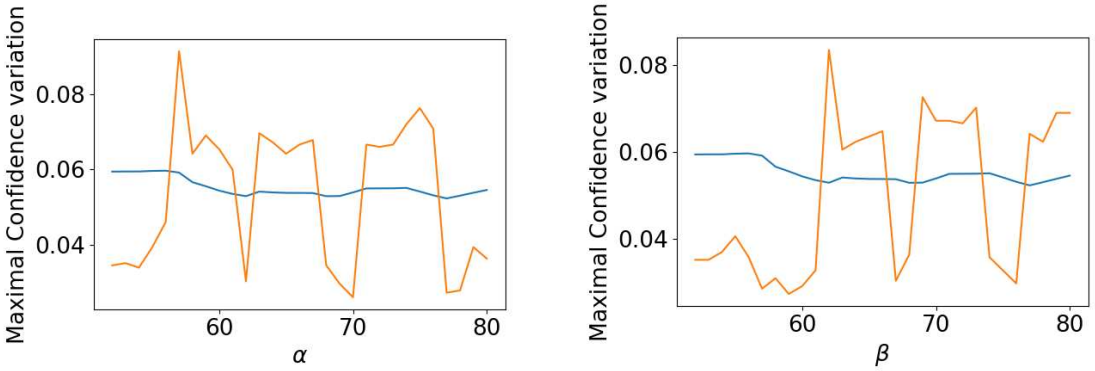
$$Sensor_0 \wedge Sensor_1^{(11,12)} \wedge Sensor_2^{(25,21)} \wedge Sensor_3^{(38,31)} \wedge Sensor_4^{(50,42)} \wedge Sensor_5^{(65,52)} \rightarrow Sensor_6^{(62,62)}$$
$$Sensor_0 \wedge Sensor_1^{(11,12)} \wedge Sensor_2^{(25,21)} \wedge Sensor_3^{(38,31)} \wedge Sensor_4^{(50,42)} \wedge Sensor_5^{(65,52)} \rightarrow Sensor_6^{(68,69)}$$
$$Sensor_0 \wedge Sensor_1^{(11,12)} \wedge Sensor_2^{(25,21)} \wedge Sensor_3^{(38,31)} \wedge Sensor_4^{(50,42)} \wedge Sensor_5^{(65,52)} \rightarrow Sensor_6^{(77,77)}$$

The corresponding maximum confidence variations are depicted in Figure 8.3.24. This is explained by the na-



(a) Expansions α **(b)** Reductions β

Figure 8.3.23: Elementary confidence variations and statistical threshold (blue line) for dependency $Sensor_1 \wedge Sensor_2^{(13,10)} \wedge Sensor_3^{(25,20)} \wedge Sensor_4^{(38,30)} \wedge Sensor_6^{(53,51)} \rightarrow Sensor_5^{(53,58)}$



(a) Expansions α **(b)** reductions β

Figure 8.3.24: Elementary confidence variations and statistical threshold (blue line) for dependency $Sensor_0 \wedge Sensor_1^{(11,12)} \wedge Sensor_2^{(25,21)} \wedge Sensor_3^{(38,31)} \wedge Sensor_4^{(50,42)} \wedge Sensor_5^{(65,52)} \rightarrow Sensor_6^{(\alpha,\beta)}$

ture of the dataset that includes temporal variability. Slight variations in terms of time lags between successive intervals accumulate at each extension (building CTD of length n from CTD of length $n-1$) until obtaining a multitude of specific dependencies characteristic shapes in maximal confidence variations. In such cases, each specific dependency is extended on its own, resulting in qualitatively duplicated conditional temporal relationships. In such cases, disjunctive relationships permit to reduce information redundancy and provide a more condensed representation that is easier to interpret. For instance, results that were provided for σ^2 are reduced to 4 tree structures that sum up the entire behaviours. One interesting fact to also observe is the adaptation of temporal constraint to temporal variability. In Figure 8.3.24, one can notice that the explored search space size is greater than the initially set one $\Delta = [0, 15]$. Using pairwise dependencies to discover temporal relationships, and by extension of limited Δ, with this dataset would not have permitted to discover the overall activity described in the dataset.

8.4 Results with the real world motion dataset

8.4.1 Dataset description

The dataset described in this section was provided by the Foxstream company[6]. We gathered data from a sensor system composed of 4 outdoor cameras situated in an office area. Each of these cameras captures a portion of the office area. Figure 8.4.1 shows snapshots of the four used cameras. Starting from images taken by these cameras, we defined ten areas, displayed with red polygons in Figure 8.4.1, corresponding to physical regions of interest and labelled as 1-1, 1-2, ..., 4-3. Figure 8.4.2 shows the position of these areas from an areal view. Starting from images taken by these cameras, we used "real-time" motion detection. Without much details, this process is performed as follows. For each area, a background image corresponding to a static scene is computed and kept in memory. This background image is compared to incoming frames: if a sufficient amount of incoming frames are different enough from the background image an event *Motion begin* is produced. An event *Motion end* is triggered if incoming frames are similar again to the background image. We collected data for three weeks and considered time portions corresponding to office hours (18 days from 6 am to 8 pm) and obtained ten time point event sequences. In order to obtain interval-based streams, we defined for each area X an environment state *Motion in area X* noted **M-X** and defined with the simple predicate:

$$\textbf{M-X}(t, X) ::= last(t, X).v == \textit{Motion begin}$$

This predicates states that an area X is in state *Motion in area X*, or *M-X*, at timestamp t (i.e time unit $[t, t+1)$) if the event of X preceding t equals *Motion begin*.

The 10 resulting state streams are described in Table 8.4.1. This data set describes motion activity in the office area during 18 working days between 6 am and 8 pm. First observations showed that it contains a significant amount of noise (e.g. detection of shadows, sudden luminosity changes) and several omissions were observed (e.g. when a car passes through an analysis zone with a great speed). Moreover, the resulting streams are extremely sparse.

Table 8.4.1: Dataset corresponding to the experiment described in Fig.8.4.2

Name	#Int	Len
1-1	1403	4303
1-2	851	3257

Name	#Int	Len
2-1	8640	47881
2-2	2099	4947
2-3	8548	26847

Name	#Int	Len
3-1	3909	14644
3-2	3699	13423

Name	#Int	Len
4-1	9686	30257
4-2	4578	13397
4-3	9825	21273

While not being heterogeneous, this dataset is interesting as it permits us to evaluate and validate our approach qualitatively with non-favourable settings. It describes a temporal phenomenon that is inherently temporally variable: cars in this environment have the same behaviour in qualitative and quantitative aspects but with slight temporal variations. Second, this dataset is noisy due to omissions and false detections. Indeed, motion detection, or more generally video processing, is still a challenging problem whose improving performance in the real world depends on several real-world conditions (e.g. change in weather, luminosity). Being able to extract accurate temporal knowledge from such dataset suggests that our approach can be applied in real-world contexts and

Figure 8.4.1: Four outdoor camera views. Red polygons describe motion analysis areas.

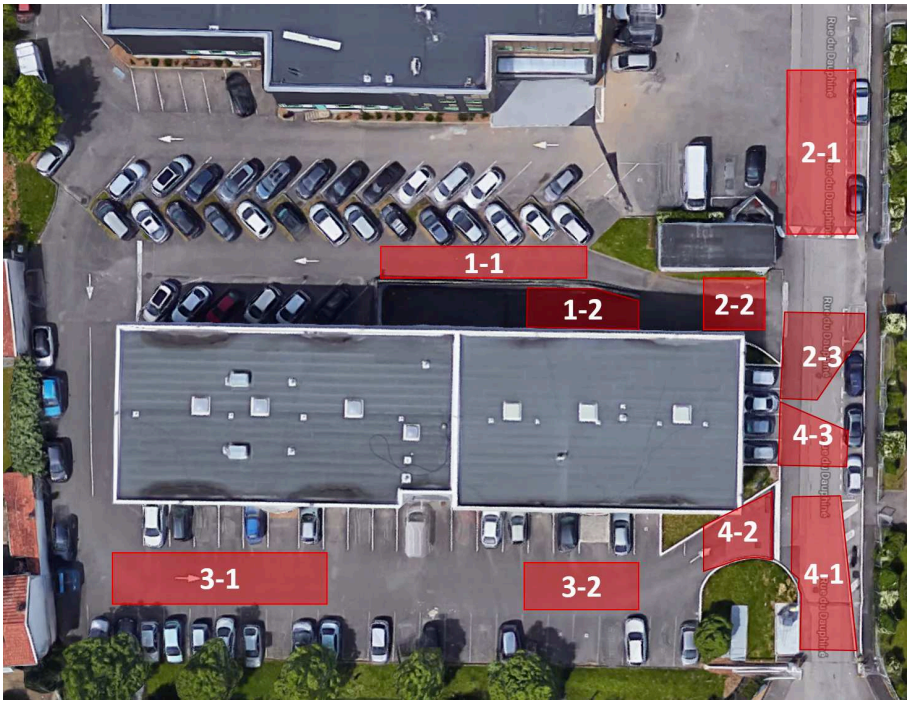


Figure 8.4.2: Position of motion analysis areas (aerial view)

Table 8.4.2: Observed ground truth on the real motion dataset. Row contains premises and columns conclusions. For example, 1-1 $\rightarrow$ 3-1 is expected while 3-1 $\rightarrow$ 1-1 is not

	1-1	1-2	2-1	2-2	2-3	3-1	3-2	4-1	4-2	4-3
1-1	-					X	X			
1-2		-		X	X			X		X
2-1			-		X			X		X
2-2	X	X	X	-	X	X		X		X
2-3	X	X	X	X	-	X		X		X
3-1						-	X	X	X	
3-2							-	X	X	
4-1	X	X	X	X	X		X	-	X	X
4-2							X	X	-	
4-3	X	X	X	X	X	X		X	X	-

that data gathered from video processing can be integrated into a knowledge discovery process.

8.4.2 Pairwise dependencies

In order to assess the accuracy of ITLD in comparison with TEDDY and PIVOTMiner on the real motion dataset, we constructed an approximate qualitative truth based on ground observations of trajectories taken by cars in the office area that can be observed for a temporal constraint of $\Delta = [0, 40]$. We emphasise that this ground truth is used as an indication for the quality of results and do not provide a strong guarantee of completeness (some positives may not be described in the state streams as well as some real positives may have been missed during observations). It is described in Table 8.4.2. We compute accuracy scores based on the qualitative matching.

We firstly executed PIVOTMiner on this dataset using minimum support of 0.1 and $\varepsilon = 1$. The exploration took 1704 seconds and provided 15 results with a maximal precision ($= 1$) a very low recall of 0.25. As we discussed earlier in this chapter, using user-given thresholds is a problematic task if no prior knowledge of temporal phenomena to be found is available. With PIVOTMiner, setting optimal parameters depends on various factors and data characteristics. For instance, temporal variability influences the ε parameter (*how dense are clusters describing a temporal relationship ?*) and trajectories occurrences influence the minimum support (*do all "significant" temporal relationships occur with more than* 10% *of a particular state intervals ?*). In our case, a user needs to execute the discovery process varying the interestingness parameters to obtain satisfactory results.

Second, we compared the efficiency of both ITLD and TEDDY in order to validate our observations on synthetic data. First, we executed both algorithms on the streams set without any pre-processing or parameters adaptation. We obtained results described in Table 8.4.3. We observe that ITLD ran five times faster than TEDDY due to its linear complexity. The two algorithms also obtained similar F_1 scores with a slightly better accuracy for ITLD. We analyse these results in more details in the following.

Table 8.4.3: Results on the FOX data set for $\Delta = [0, 40]$ (Qualitative matching)

	#Res	Precision	Recall	F1-Score	Accuracy	Run. time (s)
TEDDY	94	0.49	0.98	0.65	0.45	760
ITLD	34	0.76	0.55	0.64	0.68	**145**

[6] www.foxstream.fr

What can be first observed is that TEDDY provided positives for all pairs of state streams in the dataset. The more precise analysis of the obtained results shows that "false positives" provided by TEDDY have often large temporal transformations with, in some cases, high confidence values. Follows some example:

$$\text{1-1} \rightarrow \text{2-1}^{(40,0)}, conf = 0.42$$
$$\text{1-1} \rightarrow \text{4-3}^{(40,0)}, conf = 0.47$$
$$\text{3-2} \rightarrow \text{1-2}^{(40,0)}, conf = 0.05$$
$$\text{4-2} \rightarrow \text{2-1}^{(40,1)}, conf = 0.39$$

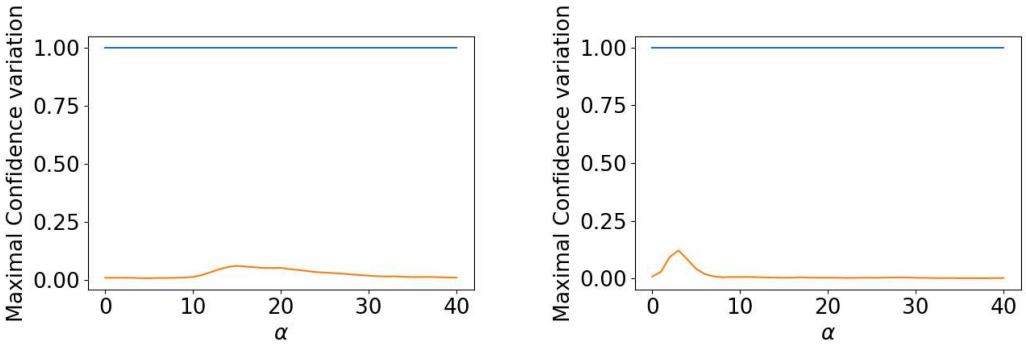
These false positives are due to two main factors. First, large temporal transformation lead to a multiplication of active length with factors up to 40. For instance, an active interval of duration 1 second may have a duration of 41 if a temporal transformation $(40, 0)$ is applied. These deformation of original state streams may lead to intersections that are not reflecting any "significant" temporal relationships. Second, state streams in the used dataset are very sparse. As a consequence, even small intersection lengths due to noise can lead to statistically valid dependencies. Using the entire observation duration to compute the validity threshold leads to very low values. However, we also observed that true positives provided by TEDDY are quantitatively accurate with satisfactory time lag characterizations. For example, the following dependencies describe a car leaving the office area starting from analysis zone 2-1:

$$\text{2-1} \rightarrow \text{2-3}^{(7,2)}, conf = 0.36$$
$$\text{2-1} \rightarrow \text{4-3}^{(12,4)}, conf = 0.37$$
$$\text{2-1} \rightarrow \text{4-1}^{(14,7)}, conf = 0.31$$

We observe that expansions α and reductions β are ordered following the spatial configuration. As a consequence, TEDDY outperforms significantly ITLD in terms of recall.

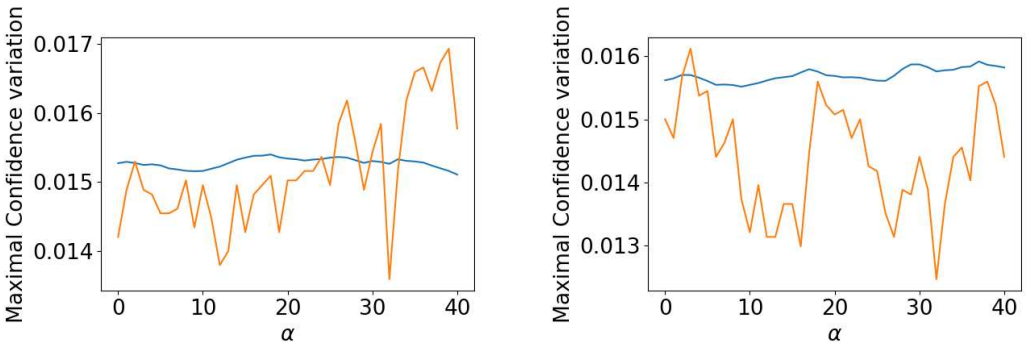
We made two main observations on dependencies provided by ITLD that explains low accuracy scores. First, the use of the statistical rule of thumbs (values of the expected contingency table ≥ 5 is too restrictive for the maximal confidence variation assessment on very sparse data streams. For example, Figure 8.4.3 provides two cases where these conditions lead to not considering the statistical test as valid. In these figures, the characteristic form of specific dependencies is present but not detected due to the application of the rule of thumbs.

The second observation is also linked to the density of data streams. We noticed that the provided statistical thresholds are very low. Indeed, maximal confidence variations assess the statistical significance of observing a confidence gain due to adding or removing active intervals of length 1 from the conclusion intervals. These quantities are negligible compared to the overall observation duration leading to very low statistical thresholds. This has two consequences. First, 7 "false positives" that do not correspond to a physical trajectory are obtained. Figure 8.4.4 provide the maximal confidence variations (for expansions) of two of these cases. Second, true positives have large time lag information due to too permissive thresholds. One descriptive example of such cases is $\text{2-1} \rightarrow \text{2-3}^{(40,0)}$. Corresponding maximal confidence variations and thresholds are depicted in Figure 8.4.5.



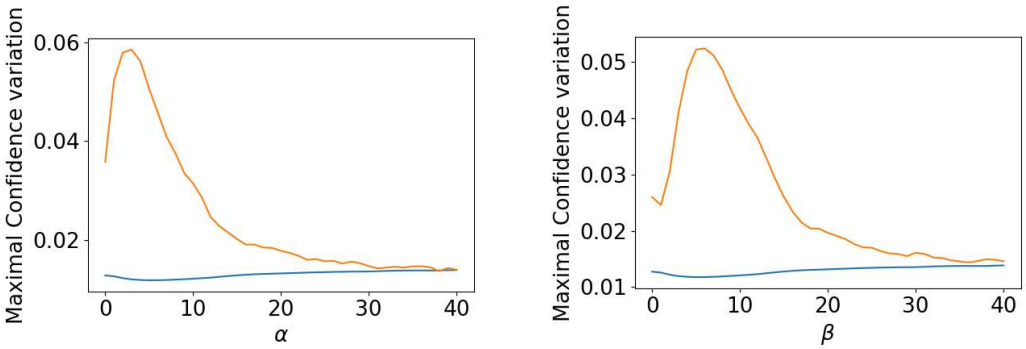
(a) 1-1 → 3-1 **(b)** 2-2 → 1-1

Figure 8.4.3: Maximal confidence variations for expansion value for 2 expected dependencies. Thresholds ≥ 1 corresponds to configuration where the rule of thumbs is not verified



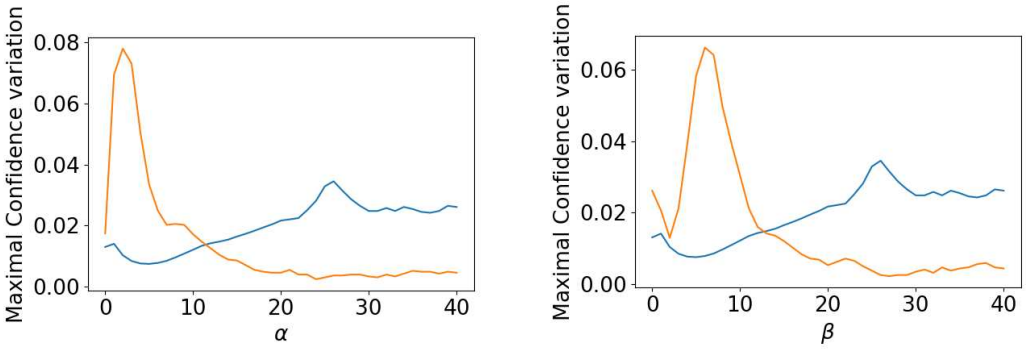
(a) 3-1 → 2-1 **(b)** 4-2 → 2-1

Figure 8.4.4: Maximal confidence variations for expansion value for 2 "false positives" provided by ITLD.



(a) 3-1 → 2-1 **(b)** 4-2 → 2-1

Figure 8.4.5: Maximal confidence variations for dependency 2-1 → 2-3$^{(40,0)}$



(a) Confidence variations for expansions **(b)** Confidence variations for reductions

Figure 8.4.6: Maximal confidence variations for dependency 1-1 → 2-2

In order to validate our first observations, we ran ITLD and TEDDY with the same parameters using a significance level of 0.005 for the statistical test (corresponding to a χ^2 value of 7.879) in order to obtain more restrictive statistical thresholds. Resulting results are described in Table 8.4.6. As expected, the precision of ITLD was enhanced by more restrictive statistical test (precision of 0.92). It is to notice that TEDDY's result was not impacted qualitatively (the same results were provided with similar time lags) nor in terms of execution time.

Table 8.4.4: Results on the FOX data set for $\Delta = [0, 40]$ (Qualitative matching) with a 0.005 statistical significance level.

	#Res	Precision	Recall	F1-Score	Accuracy	Run. time (s)
TEDDY	94	0.49	0.98	0.65	0.45	739
ITLD	27	0.96	0.55	0.70	0.75	**143**

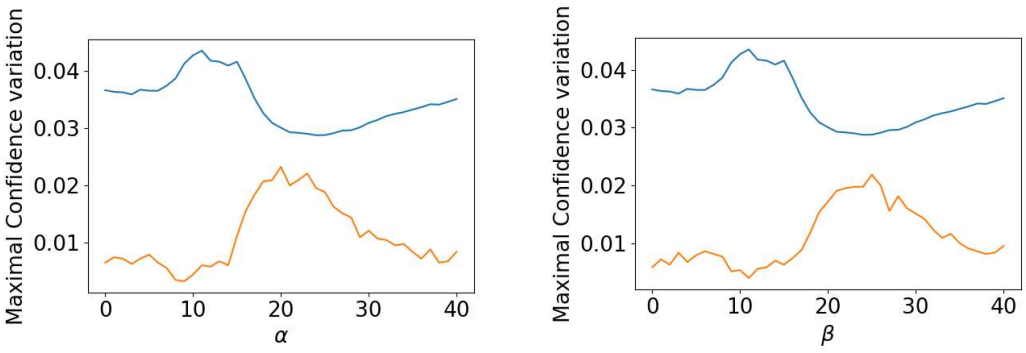
We also executed ITLD without considering the rule of thumbs for maximal confidence variations with a significance level of 0.005. The rule thumbs are applied in the dependency building and statistical validity verification to ensure the validity of the statistical test. This execution provided results as described in Table 8.4.5.

Table 8.4.5: Results on the FOX data set for $\Delta = [0, 40]$ (Qualitative matching) with a 0.005 statistical significance level.

	#Res	Precision	Recall	F1-Score	Accuracy	Run. time (s)
ITLD	40	0.92	0.76	0.84	0.84	**161**

What can be remarked is the significant enhancement of recall on the basis of the approximate ground truth. The obtained F_1 score suggests that ITLD is capable of providing results with a good balance between precision and recall for this dataset. We also remarked that 7 negatives that were missed by ITLD involve stream 1-2 that is the less dense (cf. Table 8.4.1). For example, specific reduction and expansion values for dependency 1-2 → 2-2 were detected by the maximal confidence variation approach (as shown in Figure 8.4.6) without the restrictive rule of thumbs but was not considered as a valid dependency due to its statistical assessment (that includes the rule of thumbs).

To perform a complete and fair assessment and comparison of ITLD and TEDDY, we executed both approaches while including a pre-processing stage that computes an *effective* observation duration for each pair of



(a) Confidence variations for expansions **(b)** Confidence variations for reductions

Figure 8.4.7: Maximal confidence variations for dependency 1-1 $\rightarrow$ 3-2 with *effective* observation duration obtained for a maximal gap of 80 time units.

streams. The effective observation duration is computed in linear complexity w.r.t to a maximal inactivity gap between intervals. It simply removes from the overall observation duration $\mathcal{T}_{obs}$ any time span between active intervals exceeding a maximal gap defined by the user. In order to not influence the results of the discovery process qualitatively, the maximal gap constraint must be greater or equal to the size of the temporal constraint Δ. This permit to avoid obtaining streams intersection length that would not be obtained with original streams. The obtained *effective* observation duration is used to compute statistical thresholds. We emphasise that different *effective* duration are computed for each pair of state streams. From a higher level of abstraction, it comes to consider a condensed version of each pair of state streams. However, the discovery process ran with such observation duration loses the meaning of the statistical test of independence. Indeed, it gets rid of the observation duration parameter that is essential for the interpretation of statistically valid dependencies. A dependency is valid w.r.t a given observation duration. We executed both ITLD and TEDDY on a dataset with a maximal gap constraint of $2 * |\Delta|$, non-application of the rule of thumbs with maximal confidence variations for ITLD and a significance level of 0.05. The obtained results are described in Table 8.4.6.

Table 8.4.6: Results on the FOX data set for $\Delta = [0, 40]$ (Qualitative matching) with a 0.05 statistical significance level and computation of *effective* observation durations.

	#Res	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Run. time (s)**
TEDDY	48	0.94	0.96	0.95	0.94	817
ITLD	34	0.97	0.70	0.81	0.83	**180**

They show that TEDDY results quality outperform that of ITLD, especially in terms of recall. This result suggests that TEDDY was able to reduce amounts of false positives due to low observation durations. False negatives provided by ITLD were mainly caused by high statistical thresholds that are due to low observation durations used in the statistical test. This means that maximal confidence variations were not statistically valid for the computed duration. An example of such case is depicted in Figure 8.4.7.

However, these results need to be nuanced. Results provided both by TEDDY, and ITLD depends on the arbitrary choice of the maximal gap duration for the *effective* observation duration computation. This can be easily observed with results described in Table 8.4.7 obtained for a maximal duration gap of 400.

Table 8.4.7: Results on the FOX data set for $\Delta = [0, 40]$ (Qualitative matching) with a 0.05 statistical significance level and computation of *effective* observation durations.

	#Res	Precision	Recall	F1-Score	Accuracy	Run. time (s)
TEDDY	91	0.50	0.97	0.67	0.49	790
ITLD	43	0.92	0.82	0.88	0.88	**177**

Finally, we emphasise that the execution duration of both ITLD, TEDDY and PIVOTMiner are negligible in comparison with the entire observation duration. This is a prerequisite for being able to apply the discovery process in a streaming context. Also, ITLD outperforms TEDDY in terms of execution time for $\Delta = [0, 40]$ for all configuration by a factor of at least 4 due to its linear complexity w.r.t the size of the temporal search space.

8.4.3 Complex Temporal Dependencies

We also used the real-world motion dataset to evaluate the ability of our approach to discover multiple state dependencies. In this case, finding quantitative temporal dependencies aims to discover "interesting" trajectories occurring in the office area that are mainly behaved by cars. We expected to obtain conditional temporal relationships describing how cars do enter into the office area to park and how they leave it. These simple behaviours must respect the spatial configuration (e.g. a dependency between 3-1 and 2-1 is impossible with a time lag of 1 second). In the following, we execute different configurations of CTD-Miner to evaluate its behaviour with real-world data. We recall that the used dataset is too sparse, as discussed in the last section. We will use a χ^2 value of 7.879, corresponding to a confidence level of 0.005 for reasons exposed higher. Also, CTD-Miner uses ITLD version where the statistical rule of thumbs is not applied to validity thresholds on maximal confidence variations.

We experimented CTD-Miner with a subset of state streams in the real motion dataset $\mathcal{S} = \{1\text{-}1, 1\text{-}2, 2\text{-}1, 3\text{-}1, 3\text{-}2, 4\text{-}1\}$. These streams were chosen such that there is no pair of analysis areas are directly adjacent for interpretation simplicity. This permits to obtain dependencies where time lag are likely to be expressed with temporal transformations (a, β) where $a \neq 0$ and $\beta \neq 0$.

We executed CTD-Miner as well as CTD-Miner-TEDDY with a temporal constraint $\Delta = [0, 60]$. *CTD-Miner-TEDDY* completed the exploration process in $\approx$ 1180 seconds, performed 105 temporal search space explorations and 66 premise pruning. It provided single result depicted in Figure 8.4.8. What can be noticed is that this conditional includes all state streams in the input set and uses large temporal transformation. This conditional relationship does not correspond to any significant trajectory in the office area. Therefore, we conclude that it does not provide any significant or interesting knowledge of temporal phenomena occurring in the environment. We also executed CTD-Miner-TEDDY with a statistical significance level of 0.05. Same results (in terms of temporal search space explorations, premise pruning numbers and result set) were given. This experiment suggests that the use of temporal exploration strategy of TEDDY do not permit to provide precise insight, at least with low-density datasets efficiently.

We executed CTD-Miner with ITLD as a quantitative dependency mining algorithm, as described in the first paragraph of this section. The exploration process completed in $\approx$ 150 seconds ($\approx$ 7 times faster than CTD-

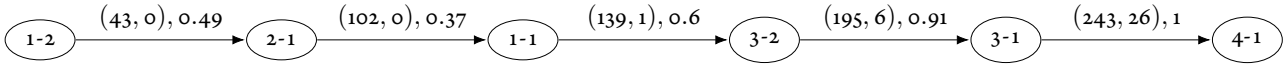


Figure 8.4.8: Conditional temporal relationships obtained by CTD-Miner-TEDDY for a set of streams $\mathcal{S} = \{1\text{-}1, 1\text{-}2, 2\text{-}1, 3\text{-}1, 3\text{-}2, 4\text{-}1\}$ with $\Delta = [0, 60]$

Miner-TEDDY), performed 68 temporal search space explorations and 1 premise pruning. The provided results are depicted in Figure 8.4.9. First, we noticed that all the provided results correspond to observable trajectories without describing the entire set of possible trajectories. We will come back to this point later in this section. Second, we can notice with conditional relationships d and e that temporal transformations are not necessarily increasing with conjunctive relationships order. This is due to two combined reasons: temporal variability and temporal constraint update approach. As the used dataset describe temporally variable phenomena (motion), relatively large transformation is obtained. For instance, in pattern (d), the relationship between 3-1 and 4-1 is temporally described by $(17, 8)$. In order to extend this conjunctive relationship, CTD-Miner update the temporal constraint to $\Delta = [0+8, 60+17]$ for reasons we described in Chapter 7. As a consequence, it is possible to extend the conjunctive relationship between 3-1 and 4-1 with a state occurring between the previous states as 3-2. This suggests that the temporal constraint update approach can be enhanced to ensure temporal order in conditional temporal relationships, especially with temporally variable phenomena. This can be achieved by using separate temporal constraint for expansion and reduction values (e.g in the former example, using $\Delta_\alpha = [0 + 17, 60 + 17]$ for expansions and $\Delta_\beta = [0 + 8, 60 + 8]$ for reductions). The other approach can be re-computing confidences of conjunctive relationships following temporal order. Results provided by CTD-Miner with ITLD described in Figure 8.4.9 do not represent all possible trajectories in the office area. The reasons behind this quality of results are the low density of data and the application of the statistical rule of thumbs. By definition, the representative stream of any temporal dependencies has a length that is less or equal to the premise stream as a stream intersection provides it. As state streams in the used dataset are non-dense, the stream representative of, for instance, pairwise dependencies is even less dense. As a consequence, in the case of low-density datasets, dependencies with numerous conjunctive relationships are more likely to not verify a condition of the statical rule of thumbs. To verify this statement, we executed CTD-Miner without the verification of the former conditions. Results described in Figure 8.4.10 were obtained (execution time $\approx$ 154 seconds, 83 temporal search space explorations, 4 premise pruning). This result set constitutes a more complete and precise description of possible trajectories in the office area, confirming our prior observation. This experiment permits to highlight the main limitation of our approach with low-density data (i.e. valid and non-valid durations are extremely unbalanced) that can be summed up with the following question. Does the statistical χ^2 test of independence on maximal confidence variations and confidence values still valid without a minimal quantity of information, or validity duration (materialised by the use of the rule of thumbs)? One direction that can be explored is that of using other statistical independence tests that are more suitable for such configurations (e.g. Fisher exact test). Nevertheless, even with this theoretical issue, our approach permitted to obtain a satisfactory environment description that is worth commenting. Follows several comments of this result set showing how conditional temporal relationships may be used and interpreted.

"Surprising" results interpretation. First, we can observe that trajectories starting from area 1-2 (that is the less dense state stream in dataset) are obtained (pattern (a)) including a temporal relationship between 1-2 and 1-1 that is extended to 3-1. This relationship do not correspond to an actual trajectory but may indicate that moving objects that passes through 1-1 may activate sometimes analysis area 1-2 (for example under the effect of car shadows passing through 2-2). A similar observation can be stated for relationships between 2-1 and 1-1 in pattern (e) (possible cause: the top of cars going from 4-1 to 1-1 may activate zone 2-1 cf. Figure 8.4.1).

Confidences interpretation example. Let us consider pattern (d). The first conjunction of this conditional relationship state that half of the active length of stream 1-1 is correlated with stream 3-1. This statement is likely to be consistent with real behaviours: an equivalent number of parking spaces are available in both in the central

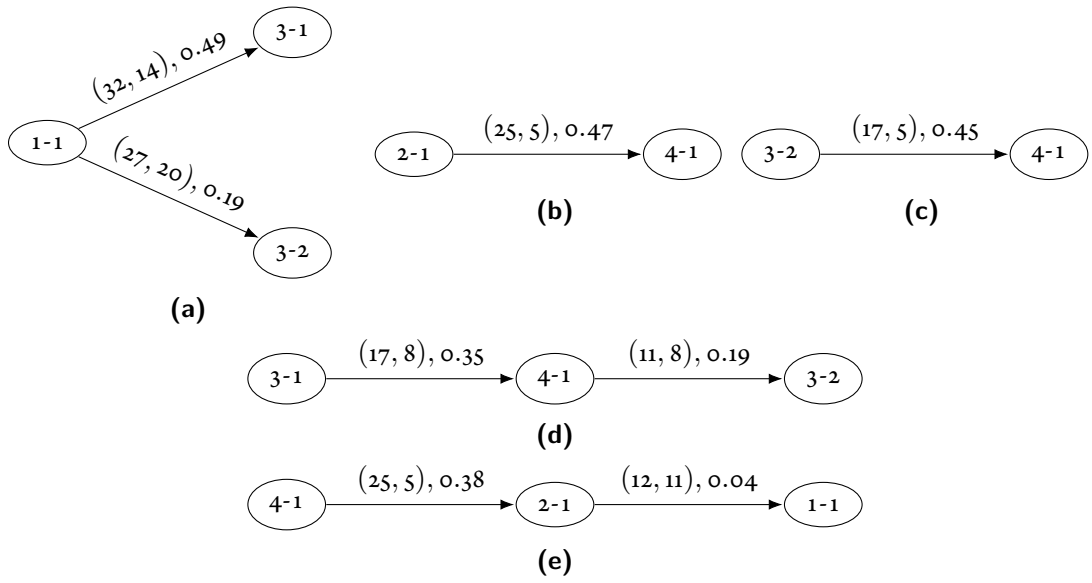


Figure 8.4.9: Conditional temporal relationships obtained by CTD-Miner for a set of streams $\mathcal{S} = \{1\text{-}1, 1\text{-}2, 2\text{-}1, 3\text{-}1, 3\text{-}2, 4\text{-}1\}$ with $\Delta = [0, 60]$

main building north and south. Approximatively one third of the former activity induces motion in area 3-2: the remaining cars park in spaces near are 3-1. The proportion of remaining cars leaving the office area generates a proportion of 0.6 of the remaining active length: when reaching 3-2 from 1-1, the probability for a car of leaving the office area is higher than parking.

Information redundancy. In the result set described in Figure 8.4.10, we observe that temporal relationships between 3-2 and 4-1 appears in two conditional temporal relationships (c) and (d) that are both closed-like and can be considered as non-redundant. This result is to be interpreted as follows: all environment actors performing the sub-trajectory 3-2 to 4-1 do not necessarily come from 1-1. In a higher level of abstraction, the former statement can be expressed as: there are two types of cars leaving the office area passing through 3-1: those which had parked in the building north and those that had parked in the building south. It is to notice that the time lag characterisation in the pattern (c) $(17, 5)$ is more general compared to its correspondent in the pattern (d) $(6, 7)$ (obtained by subtraction) suggesting a behavioural difference.

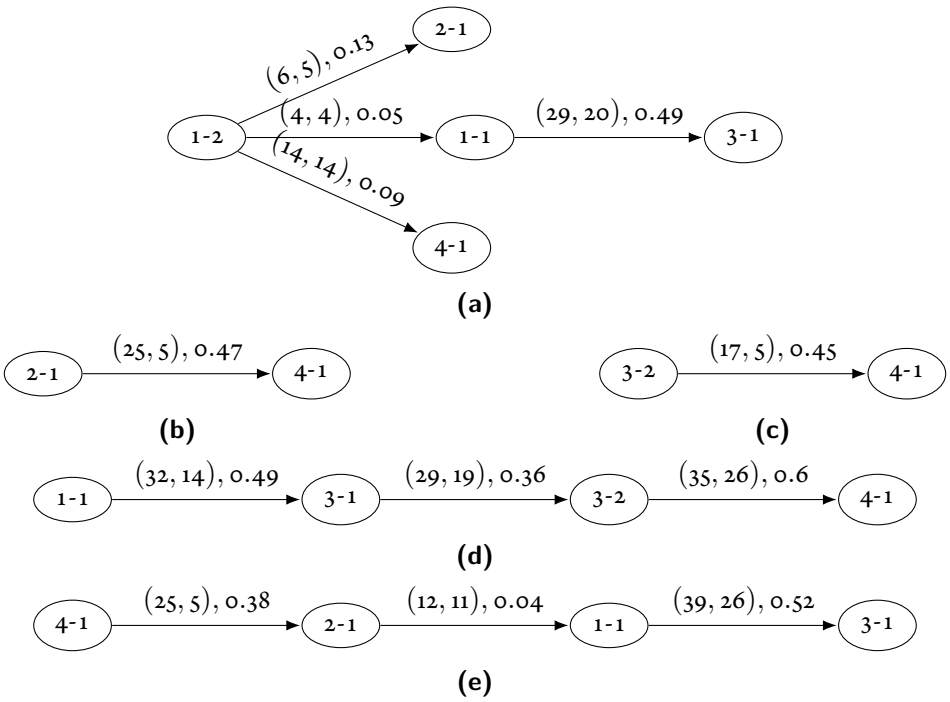


Figure 8.4.10: Conditional temporal relationships obtained by CTD-Miner for a set of streams $\mathcal{S} = \{1\text{-}1, 1\text{-}2, 2\text{-}1, 3\text{-}1, 3\text{-}2, 4\text{-}1\}$ with $\Delta = [0, 60]$ without application of statistical rule of thumbs.

9

Conclusion and future directions

9.1 Conclusion

In this thesis, we have been interested in the general problem of discovering interesting temporal knowledge from data provided by heterogeneous data sources. More precisely, the work presented in this dissertation is structured around two main questions: (1) how to use heterogeneous data streams in a same knowledge discovery process? and more specifically, at what extent heterogeneous data can be unified into a single data model that maintains temporality? (2) how to extract quantitative temporal knowledge from a set of interval-based streams?

The first part of this dissertation aims to introduce and motivate the primary approach of our work and provide an answer to the first question. Chapter 2, analyses several conceptual characteristics of temporal data, including time domains, time primitives, data models and datasets activities separation. We argued that the characteristics of any temporal data model are guided by their generation process and semantics of described temporal phenomena. As a consequence, heterogeneity of temporal data in a data source system (e.g. sensor system) may be caused by each of these mentioned parameters. In Chapter 3, we described the general conceptual approach we adopted to solve the heterogeneity problem. We proposed to use data models unification based on Temporal Abstraction in order to apply a classical temporal pattern mining approach. It aims to convert heterogeneous raw data streams into a set of interval-based streams, called state streams. A state refers to a temporal configuration of data referring to a high-level environment state of interest for the application domain. They are defined as predicates, on data from one or multiple raw data streams, that can use expert knowledge or discretization approaches (e.g. for time series). This process can also include specific pattern retrieval approaches to include complex information into the discovery process, especially for unstructured data as videos, images or texts. For instance, we show in the Chapter 8 that temporal knowledge (trajectories in this case) can be extracted from data gathered from video processing, specifically motion detection that is still a challenging task. Besides making it possible to solve heterogeneity problems (unification of time domains, time primitives, data models and formats), this approach makes it possible to obtain a description of a given environment through states that compose a high level, understandable, vocabulary for pattern languages of discovery processes. We also believe that this approach may

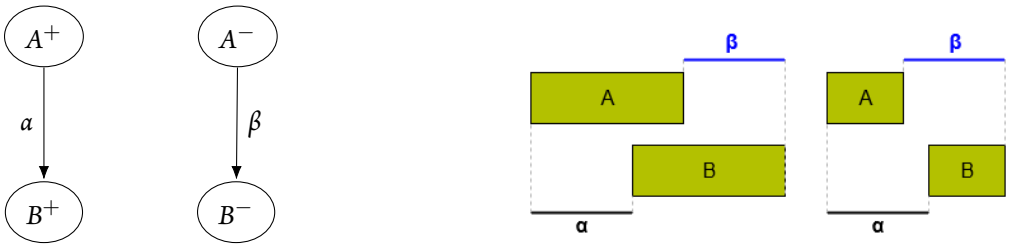
facilitate the discovery of multi-domain and holistic knowledge that may highlight hidden correlations between multiple domains of knowledge.

The second part of this thesis includes its main contributions. It tackles the problem of discovering temporal knowledge from a set of interval-based data. In Chapter 5, we proposed a generalisation of the intersection-based quantitative pairwise dependency model previously proposed by [127]. Contrary to the majority of existing approaches regardless of their activity separation format (attribute of subject centred), the authors proposed to use the intersection of interval-based streams as interestingness measure. We argued that this approach could provide a better assessment of temporal relationships in comparison with occurrence counting (for frequent pattern mining), especially with dense attribute centred interval-based streams where a single interval may report on multiple temporal relationships occurrences. Also, this approach uses a Pearson χ^2 independence test to assess the interestingness of pairwise temporal dependencies, making it possible to avoid using user-given thresholds. In this work, we proposed *Complex Temporal Dependencies* that aims to model temporal relationships between multiple states. It includes conjunction and disjunction operators, similarly to a normal conjunctive form, that is expressive enough to describe a large set of temporal relationships and complex configuration. We also showed how this dependency model could be used to build temporal models that maintain the conditionality of temporal relationships. We argued that being able to report on such conditional relationships allows a better understanding of temporal phenomena and behaviours through consistent qualitative, quantitative and confidence information. In Chapter 6, we proposed *CTD-Miner* an efficient algorithm devised to discover CTDs from a set of interval-based streams. It is a breadth-first algorithm using incremental construction of conditional Complex Temporal Dependencies. In this process, a corresponding Apriori-like pruning is used to reduce the search space considerably. We also proposed a closed-like property permitting to prune the search space based on previously discovered dependencies as well as reducing redundancy in the result set. The process of CTD-Miner uses a pairwise dependency discovery algorithm. At the best of our knowledge, only two existent approaches can be directly or easily adapted to mining quantitative relationships between pairwise dependencies. The first, named TEDDY [127], uses the intersection-based pairwise dependency model based on which we proposed CTDs. The second, PIVOTMiner [78], is a frequent pattern mining using user-given thresholds. In this dissertation, we proposed in Chapter 7 an alternative approach *Interval Time Lag Discovery* (ITLD). It is based on the analysis of maximal confidence variations heuristic permitting to perform a linear exploration of the quadratic search space w.r.t the temporal constraint. Maximal confidence variations approach is based on the analysis of intersection gains/losses, and by extension on confidence measure, induced by elementary translations of the conclusion stream. This approach permits to maintain the number of intervals in an interval-based stream that avoids overlapping and cancellation of intervals under the effect of temporal transformation. Finally, in Chapter 8, we evaluate our approach using both simulated datasets and motion data gathered from a sensor system composed of outdoor cameras. The experiments we conducted firstly shows that ITLD provides accurate results in comparison with existing approaches. They also suggest the advantage of using statistical assessment in that it permits to exempt the user from disposing of prior knowledge guiding its choice of interestingness threshold. Also, the intersection-based assessment had shown better robustness to noise that occurrence counting support. We also reported on performances of CTD-Miner. The experiments had shown that the pruning criteria based on the closed-like property permit to efficiently accelerate the discovery process and reduce the number of redundant results.

In this thesis, we have motivated the use of interval-based data as a unique model for a high-level representation of information provided by heterogeneous data streams and proposed a pattern language as well as efficient discovery algorithms devised to quantitative temporal knowledge discovery. Encouraging results permit to validate the accuracy of our approach from a pattern mining perspective suggesting future studies, directions, and problems to be addressed. We discuss several of these in the following.

Conducting qualitative experiments and use cases using heterogeneous systems. Although we experimented our approach in a real-world motion dataset, one weakness of our study is the lack of experimentation on more real-world heterogeneous data streams systems. Another experimentation scenario that is worth conducting is that including states corresponding to multiple expertise domains providing different perspectives on the same set of temporal facts. In order to make future studies easier to perform, we began full implementation of the framework presented in Chapter 3 that includes streaming management, online interval-based streams construction, a knowledge discovery module and also an extension of the simulation tool we used in this work.

Mining non-ambiguous quantitative interval-based patterns. As we have discussed in Chapter 4, there is no known approach permitting to discover fully non-ambiguous interval-based patterns. Existing quantitative pattern languages [127] [78] use time lag information represented as a pair (α, β) where values express the time lag between first and second interval endpoints respectively. While providing sufficient temporal information to describe a large set of temporal insight, this representation does not permit to fully characterize temporal relationships between intervals as it can be remarked in the following example.



A fully non-ambiguous pairwise relationship between A and B would require at least an additional temporal information of intervals duration (e.g between A^+ and A^-) and/or the gap between between a first and second endpoint or *vice-versa*. This supplementary information may be also included in the Complex Temporal Dependency Model. One direction that would permit to achieve this is made possible by the maximal confidence variation approach. We believe that width of characteristic forms of specific dependencies may correspond to the duration of premises intervals. Therefore, the problem is to evaluate the consistency of the former claim with noisy data and temporal variability.

On-line discovery of intersection-based dependencies. In this work, we focused on processing off-line data in order to validate our approach. One open question that remains unsolved is the ability of on-line and continuous mining of intersection-based dependencies. This task can be handy with huge data sources systems with a big amount of streams providing data continuously that cannot be entirely stored in memory. This seems to be achievable with pairwise dependencies using a continuous update of the maximal confidence variation values and statistical validity thresholds. One approach is that of updating in $\Theta(|\Delta|)$ these values for each incoming interval. However, the generalization to online Complex Temporal Dependencies mining seems to be a more difficult task to achieve and worth being investigated.

Concept drift detection. CTD-Miner permits to discover conditional temporal relationships that permit to build a temporal model of a given environment. This model can be used to perform forecasting of events consisting of a set of probabilities of occurrence of future states. One interesting task is to be able to assess the validity of this temporal model over time and detect partial or total behavioural changes. This is a challenging task since it requires to assess whether unsuccessful predictions are due to ordinary noise, outliers or real progressive/sudden change in environment activity.

Temporal knowledge storage and query. In a context where storage is not feasible due to vast amounts of raw data streams and on-line knowledge discovery is performed, one interesting feature is the storage and query of insights discovered over time. For instance, let us consider that a monitoring system stores a temporal model with a granularity of one hour in a knowledge database. One query that could be interesting to execute is: *What is the temporal model of environment activity for the observation duration between 8:00 am and 8:00 pm?*. The underlying challenge to address this query is to be able to *merge* several conditional temporal models into a unique one with a certain level of accuracy. With pairwise dependencies, we believe that the maximal confidence variation approach may permit to solve this problem.

References

[1] Serge Abiteboul. Querying semi-structured data. In Foto N. Afrati and Phokion G. Kolaitis, editors, *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 1997. doi: 10.1007/3-540-62222-5_33. URL https://doi.org/10.1007/3-540-62222-5_33.

[2] Charu C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015. ISBN 978-3-319-14141-1. doi: 10.1007/978-3-319-14142-8. URL https://doi.org/10.1007/978-3-319-14142-8.

[3] Charu C. Aggarwal and ChengXiang Zhai. An introduction to text mining. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 1–10. Springer, 2012. doi: 10.1007/978-1-4614-3223-4_1. URL https://doi.org/10.1007/978-1-4614-3223-4_1.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 3–14. IEEE Computer Society, 1995. doi: 10.1109/ICDE.1995.380415. URL https://doi.org/10.1109/ICDE.1995.380415.

[5] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993.*, pages 207–216. ACM Press, 1993. doi: 10.1145/170035.170072. URL https://doi.org/10.1145/170035.170072.

[6] Helena Ahonen-Myka. Mining all maximal frequent word sequences in a set of sentences. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 255–256. ACM, 2005. doi: 10.1145/1099554.1099614. URL https://doi.org/10.1145/1099554.1099614.

[7] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction Series. Springer, 2011. ISBN 978-0-85729-078-6. doi: 10.1007/978-0-85729-079-3. URL https://doi.org/10.1007/978-0-85729-079-3.

[8] Hunor Albert-Lorincz and Jean-François Boulicaut. Mining frequent sequential patterns under regular expressions: A highly adaptive strategy for pushing contraints. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, pages 316–320. SIAM, 2003. doi: 10.1137/1.9781611972733.37. URL https://doi.org/10.1137/1.9781611972733.37.

[9] James F. Allen. An interval-based representation of temporal knowledge. In Patrick J. Hayes, editor, *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24-28, 1981*, pages 221–226. William Kaufmann, 1981. URL http://ijcai.org/Proceedings/81-1/Papers/045.pdf.

[10] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983. doi: 10.1145/182.358434. URL http://doi.acm.org/10.1145/182.358434.

[11] James F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, 1984. doi: 10.1016/0004-3702(84)90008-0. URL https://doi.org/10.1016/0004-3702(84)90008-0.

[12] James F. Allen and Patrick J. Hayes. A common-sense theory of time. In *IJCAI*, pages 528–531. Morgan Kaufmann, 1985.

[13] James F. Allen and Patrick J. Hayes. Moments, points in an interval-based temporal logic. *Computational Intelligence*, 5:225–238, 1989.

[14] Miguel R. Álvarez, Paulo Félix, and Purificación Cariñena. Discovering metric temporal constraint networks on temporal databases. *Artificial Intelligence in Medicine*, 58(3):139–154, 2013. doi: 10.1016/j.artmed.2013.03.006. URL https://doi.org/10.1016/j.artmed.2013.03.006.

[15] Gad Ariav. A temporally oriented data model. *ACM Trans. Database Syst.*, 11(4):499–527, 1986. doi: 10.1145/7239.7350. URL https://doi.org/10.1145/7239.7350.

[16] Sujeevan Aseervatham, Aomar Osmani, and Emmanuel Viennet. bitspade: A lattice-based sequential pattern mining algorithm using bitmap representation. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 792–797. IEEE Computer Society, 2006. doi: 10.1109/ICDM.2006.28. URL https://doi.org/10.1109/ICDM.2006.28.

[17] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 429–435. ACM, 2002. doi: 10.1145/775047.775109. URL https://doi.org/10.1145/775047.775109.

[18] Silvana Badaloni and Massimiliano Giacomin. The algebra ia^{fuz}: a framework for qualitative fuzzy temporal reasoning. *Artif. Intell.*, 170(10):872–908, 2006. doi: 10.1016/j.artint.2006.04.001. URL https://doi.org/10.1016/j.artint.2006.04.001.

[19] Jaume Baixeries, Gemma Casas-Garriga, and José L. Balcázar. A best-first strategy for finding frequent sets. In Danièle Hérin and Djamel A. Zighed, editors, *Extraction et gestion des connaissances (EGC'2002), Actes des deuxièmes journées Extraction et Gestion des Connaissances, Montpellier, France, 21-23 janvier 2002*, volume 1 of *Extraction des Connaissances et Apprentissage*, pages 101–106. Hermes Science Publications, 2002.

[20] Kaustubh Beedkar, Klaus Berberich, Rainer Gemulla, and Iris Miliaraki. Closing the gap: Sequence mining at scale. *ACM Trans. Database Syst.*, 40(2):8:1–8:44, 2015. doi: 10.1145/2757217. URL https://doi.org/10.1145/2757217.

[21] Kirk D. Borne. Scientific data mining in astronomy. In Hillol Kargupta, Jiawei Han, Philip S. Yu, Rajeev Motwani, and Vipin Kumar, editors, *Next Generation of Data Mining.*, Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press / Chapman and Hall / Taylor & Francis, 2008. doi: 10.1201/9781420085877.ch5. URL https://doi.org/10.1201/9781420085877.ch5.

[22] Peter Buneman. Semistructured data. In Alberto O. Mendelzon and Z. Meral Özsoyoglu, editors, *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA*, pages 117–121. ACM Press, 1997. doi: 10.1145/263661.263675. URL https://doi.org/10.1145/263661.263675.

[23] Gemma Casas-Garriga. Discovering unbounded episodes in sequential data. In Nada Lavrac, Dragan Gamberger, Hendrik Blockeel, and Ljupco Todorovski, editors, *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, volume 2838 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2003. doi: 10.1007/978-3-540-39804-2_10. URL https://doi.org/10.1007/978-3-540-39804-2_10.

[24] Gemma Casas-Garriga. Summarizing sequential data with closed partial orders. In Hillol Kargupta, Jaideep Srivastava, Chandrika Kamath, and Arnold Goodman, editors, *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, pages 380–391. SIAM, 2005. doi: 10.1137/1.9781611972757.34. URL https://doi.org/10.1137/1.9781611972757.34.

[25] Yen-Liang Chen and Tony Cheng-Kui Huang. Discovering fuzzy time-interval sequential patterns in sequence databases. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 35(5):959–972, 2005. doi: 10.1109/TSMCB.2005.847741. URL https://doi.org/10.1109/TSMCB.2005.847741.

[26] Yen-Liang Chen, Mei-Ching Chiang, and Ming-Tat Ko. Discovering time-interval sequential patterns in sequence databases. *Expert Syst. Appl.*, 25(3):343–354, 2003. doi: 10.1016/S0957-4174(03)00075-7. URL https://doi.org/10.1016/S0957-4174(03)00075-7.

[27] Yen-Liang Chen, Shin-yi Wu, and Yu-Cheng Wang. Discovering multi-label temporal patterns in sequence databases. *Inf. Sci.*, 181(3):398–418, 2011. doi: 10.1016/j.ins.2010.09.024. URL https://doi.org/10.1016/j.ins.2010.09.024.

[28] Yi-Cheng Chen, Ji-Chiang Jiang, Wen-Chih Peng, and Suh-Yin Lee. An efficient algorithm for mining time interval-based patterns in large database. In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 49–58. ACM, 2010. doi: 10.1145/1871437.1871448. URL https://doi.org/10.1145/1871437.1871448.

[29] Yi-Cheng Chen, Julia Tzu-Ya Weng, and Lin Hui. A novel algorithm for mining closed temporal patterns from interval-based data. *Knowl. Inf. Syst.*, 46(1):151–183, 2016. doi: 10.1007/s10115-014-0815-2. URL https://doi.org/10.1007/s10115-014-0815-2.

[30] Yi-Chun Chen and Guanling Lee. An efficient projected database method for mining sequential association rules. In *Fifth IEEE International Conference on Digital Information Management, ICDIM 2010, July 5-8, 2010, Lakehead University, Thunder Bay, Canada*, pages 274–278. IEEE, 2010. doi: 10.1109/ICDIM.2010.5664724. URL https://doi.org/10.1109/ICDIM.2010.5664724.

[31] Damien Cram, Benoit Mathern, and Alain Mille. A complete chronicle discovery approach: application to activity analysis. *Expert Systems*, 29(4):321–346, 2012. doi: 10.1111/j.1468-0394.2011.00591.x. URL https://doi.org/10.1111/j.1468-0394.2011.00591.x.

[32] Yann Dauxais, Thomas Guyet, David Gross-Amblard, and André Happe. Discriminant chronicles mining - application to care pathways analytics. In Annette ten Teije, Christian Popow, John H. Holmes, and Lucia Sacchi, editors, *Artificial Intelligence in Medicine - 16th Conference on Artificial Intelligence in Medicine, AIME 2017, Vienna, Austria, June 21-24, 2017, Proceedings*, volume 10259 of *Lecture Notes in Computer Science*, pages 234–244. Springer, 2017. doi: 10.1007/978-3-319-59758-4_26. URL https://doi.org/10.1007/978-3-319-59758-4_26.

[33] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[34] Christophe Dousson and Thang Vu Duong. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 620–626. Morgan Kaufmann, 1999. URL http://ijcai.org/Proceedings/99-1/Papers/089.pdf.

[35] Christophe Dousson, Paul Gaborit, and Malik Ghallab. Situation recognition: Representation and algorithms. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 166–174. Morgan Kaufmann, 1993. URL http://ijcai.org/Proceedings/93-1/Papers/024.pdf.

[36] Didier Dubois and Henri Prade. Processing fuzzy temporal knowledge. *IEEE Trans. Systems, Man, and Cybernetics*, 19(4):729–744, 1989. doi: 10.1109/21.35337. URL https://doi.org/10.1109/21.35337.

[37] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice-Hall, 2002. ISBN 0-13-088892-3. URL http://www.seas.smu.edu/%7Emhd/book.

[38] Liran Einav and Jonathan Levin. Economics in the age of big data. *Science*, 346(6210):1243089, 2014.

[39] En-Zheng Guan, Xiao-Yu Chang, Zhe Wang, and Chun-Guang Zhou. Mining maximal sequential patterns. In *2005 International Conference on Neural Networks and Brain*, volume 1, pages 525–528, Oct 2005. doi: 10.1109/ICNNB.2005.1614668.

[40] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, 1996. URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/1230.

[41] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 82–88. AAAI Press, 1996. URL http://www.aaai.org/Library/KDD/1996/kdd96-014.php.

[42] Michael Fisher. Temporal representation and reasoning. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 513–550. Elsevier, 2008. doi: 10.1016/S1574-6526(07)03012-X. URL https://doi.org/10.1016/S1574-6526(07)03012-X.

[43] Philippe Fournier-Viger, Usef Faghihi, Roger Nkambou, and Engelbert Mephu Nguifo. CMRULES: an efficient algorithm for mining sequential rules common to several sequences. In Hans W. Guesgen and R. Charles Murray, editors, *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference, May 19-21, 2010, Daytona Beach, Florida, USA*. AAAI Press, 2010. URL http://www.aaai.org/ocs/index.php/FLAIRS/2010/paper/view/1390.

[44] Philippe Fournier-Viger, Roger Nkambou, and Vincent Shin-Mu Tseng. Rulegrowth: mining sequential rules common to several sequences by pattern-growth. In William C. Chu, W. Eric Wong, Mathew J. Palakal, and Chih-Cheng Hung, editors, *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), TaiChung, Taiwan, March 21 - 24, 2011*, pages 956–961. ACM, 2011. doi: 10.1145/1982185.1982394. URL https://doi.org/10.1145/1982185.1982394.

[45] Philippe Fournier-Viger, Cheng-Wei Wu, and Vincent S. Tseng. Mining maximal sequential patterns without candidate maintenance. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar R. Zaïane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications, 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I*, volume 8346 of *Lecture Notes in Computer Science*, pages 169–180. Springer, 2013. doi: 10.1007/978-3-642-53914-5_15. URL https://doi.org/10.1007/978-3-642-53914-5_15.

[46] Philippe Fournier-Viger, Antonio Gomariz, Michal Sebek, and Martin Hlosta. VGEN: fast vertical mining of sequential generator patterns. In Ladjel Bellatreche and Mukesh K. Mohania, editors, *Data Warehousing and Knowledge Discovery - 16th International Conference, DaWaK 2014, Munich, Germany, September 2-4, 2014. Proceedings*, volume 8646 of *Lecture Notes in Computer Science*, pages 476–488. Springer, 2014. doi: 10.1007/978-3-319-10160-6_42. URL https://doi.org/10.1007/978-3-319-10160-6_42.

[47] Philippe Fournier-Viger, Ted Gueniche, Souleymane Zida, and Vincent S. Tseng. Erminer: Sequential rule mining using equivalence classes. In Hendrik Blockeel, Matthijs van Leeuwen, and Veronica Vinciotti, editors, *Advances in Intelligent Data Analysis XIII - 13th International Symposium, IDA 2014, Leuven,*

Belgium, October 30 - November 1, 2014. Proceedings, volume 8819 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2014. doi: 10.1007/978-3-319-12571-8_10. URL https://doi.org/10.1007/978-3-319-12571-8_10.

[48] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S. Tseng. VMSP: efficient vertical mining of maximal sequential patterns. In Marina Sokolova and Peter van Beek, editors, *Advances in Artificial Intelligence - 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings*, volume 8436 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2014. doi: 10.1007/978-3-319-06483-3_8. URL https://doi.org/10.1007/978-3-319-06483-3_8.

[49] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[50] Andrew U Frank. Different types of "times" in gis. *Spatial and temporal reasoning in geographic information systems*, pages 40–62, 1998.

[51] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, 13(3):57–70, 1992. URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/1011.

[52] Christian Freksa. Temporal reasoning based on semi-intervals. *Artif. Intell.*, 54(1):199–227, 1992.

[53] Fabio Fumarola, Pasqua Fabiana Lanotte, Michelangelo Ceci, and Donato Malerba. Clofast: closed sequential pattern mining using sparse and vertical id-lists. *Knowl. Inf. Syst.*, 48(2):429–463, 2016. doi: 10.1007/s10115-015-0884-x. URL https://doi.org/10.1007/s10115-015-0884-x.

[54] Aymen Gammoudi, Allel Hadjali, and Boutheina Ben Yaghlane. Modeling temporal relations between fuzzy TIME intervals: A disjunctive view. In *2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 50–57. IEEE, 2016. doi: 10.1109/FUZZ-IEEE.2016.7737667. URL https://doi.org/10.1109/FUZZ-IEEE.2016.7737667.

[55] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *Int J. Information Management*, 35(2):137–144, 2015. doi: 10.1016/j.ijinfomgt.2014.10.007. URL https://doi.org/10.1016/j.ijinfomgt.2014.10.007.

[56] Chuancong Gao, Jianyong Wang, Yukai He, and Lizhu Zhou. Efficient mining of frequent sequence generators. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 1051–1052. ACM, 2008. doi: 10.1145/1367497.1367651. URL https://doi.org/10.1145/1367497.1367651.

[57] René Arnulfo García-Hernández, Jos'e Francisco Mart'inez Trinidad, and Jesús Ariel Carrasco-Ochoa. A new algorithm for fast discovery of maximal sequential patterns in a document collection. In Alexander F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing, 7th International Conference, CICLing 2006, Mexico City, Mexico, February 19-25, 2006, Proceedings*, volume 3878 of *Lecture Notes in Computer Science*, pages 514–523. Springer, 2006. doi: 10.1007/11671299_53. URL https://doi.org/10.1007/11671299_53.

[58] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. SPIRIT: sequential pattern mining with regular expression constraints. In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie, editors, *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 223–234. Morgan Kaufmann, 1999. URL http://www.vldb.org/conf/1999/P22.pdf.

[59] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowl. Data Eng.*, 14(3):530–552, 2002. doi: 10.1109/TKDE.2002.1000341. URL https://doi.org/10.1109/TKDE.2002.1000341.

[60] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006. doi: 10.1145/1132960.1132963. URL https://doi.org/10.1145/1132960.1132963.

[61] Fatma Ghorbel, Fayçal Hamdi, and Elisabeth Métais. Temporal relations between imprecise time intervals: Representation and reasoning. In Dominik Endres, Mehwish Alam, and Diana Sotropa, editors, *Graph-Based Representation and Reasoning - 24th International Conference on Conceptual Structures, ICCS 2019, Marburg, Germany, July 1-4, 2019, Proceedings*, volume 11530 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2019. doi: 10.1007/978-3-030-23182-8_7. URL https://doi.org/10.1007/978-3-030-23182-8_7.

[62] Fosca Giannotti, Mirco Nanni, and Dino Pedreschi. Efficient mining of temporally annotated sequences. In Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors, *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 348–359. SIAM, 2006. doi: 10.1137/1.9781611972764.31. URL https://doi.org/10.1137/1.9781611972764.31.

[63] Fosca Giannotti, Mirco Nanni, Dino Pedreschi, and Fabio Pinelli. Mining sequences with temporal annotations. In Hisham Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 593–597. ACM, 2006. doi: 10.1145/1141277.1141413. URL https://doi.org/10.1145/1141277.1141413.

[64] Jim Giles. Computational social science: Making the links. *Nature*, 488(7412):448–450, 2012. doi: 10.1038/488448a. URL https://doi.org/10.1038/488448a.

[65] Antonio Gomariz, Manuel Campos, Roque Marín, and Bart Goethals. Clasp: An efficient algorithm for mining frequent closed sequences. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I*, volume 7818 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2013. doi: 10.1007/978-3-642-37453-1_5. URL https://doi.org/10.1007/978-3-642-37453-1_5.

[66] Leticia I. Gómez and Alejandro A. Vaisman. Re-spam: Using regular expressions for sequential pattern mining in trajectory databases. In *Workshops Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 395–398. IEEE Computer Society, 2008. doi: 10.1109/ICDMW.2008.14. URL https://doi.org/10.1109/ICDMW.2008.14.

[67] Iqbal A. Goralwalla, Yuri Leontiev, M. Tamer Özsu, and Duane Szafron. Modeling temporal primitives: Back to basics. In Forouzan Golshani and Kia Makki, editors, *Proceedings of the Sixth International Conference on Information and Knowledge Management (CIKM'97), Las Vegas, Nevada, USA, November 10-14, 1997*, pages 24–31. ACM, 1997. doi: 10.1145/266714.266847. URL https://doi.org/10.1145/266714.266847.

[68] Iqbal A. Goralwalla, M. Tamer Özsu, and Duane Szafron. An object-oriented framework for temporal data models. In *Temporal Databases, Dagstuhl*, pages 1–35, 1997. doi: 10.1007/BFb0053696. URL https://doi.org/10.1007/BFb0053696.

[69] Karam Gouda, Mosab Hassaan, and Mohammed J. Zaki. Prism: An effective approach for frequent sequence mining via prime-block encoding. *J. Comput. Syst. Sci.*, 76(1):88–102, 2010. doi: 10.1016/j.jcss.2009.05.008. URL https://doi.org/10.1016/j.jcss.2009.05.008.

[70] Valerio Grossi, Beatrice Rapisarda, Fosca Giannotti, and Dino Pedreschi. Data science at sobigdata: the european research infrastructure for social mining and big data analytics. *Int. J. Data Sci. Anal.*, 6(3):205–216, 2018. doi: 10.1007/s41060-018-0126-x. URL https://doi.org/10.1007/s41060-018-0126-x.

[71] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.*, 26(9):2250–2267, 2014. doi: 10.1109/TKDE.2013.184. URL https://doi.org/10.1109/TKDE.2013.184.

[72] Thomas Guyet and Rene Quiniou. Mining temporal patterns with quantitative intervals. In *Workshops Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 218–227. IEEE Computer Society, 2008. doi: 10.1109/ICDMW.2008.16. URL https://doi.org/10.1109/ICDMW.2008.16.

[73] Thomas Guyet and Rene Quiniou. Extracting temporal patterns from interval-based sequences. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1306–1311. IJCAI/AAAI, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-221. URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-221.

[74] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 1–12. ACM, 2000. doi: 10.1145/342009.335372. URL https://doi.org/10.1145/342009.335372.

[75] David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, 2001. ISBN 9780262082907. URL https://mitpress.mit.edu/books/principles-data-mining.

[76] Sherri K. Harms and Jitender S. Deogun. Sequential association rule mining with time lags. *J. Intell. Inf. Syst.*, 22(1):7–22, 2004. doi: 10.1023/A:1025824629047. URL https://doi.org/10.1023/A:1025824629047.

[77] Sherri K. Harms, Jitender S. Deogun, Jamil Saquer, and Tsegaye Tadesse. Discovering representative episodal association rules from event sequences using frequent closed episode sets and event constraints. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 603–606. IEEE Computer Society, 2001. doi: 10.1109/ICDM.2001.989576. URL https://doi.org/10.1109/ICDM.2001.989576.

[78] Marwan Hassani, Yifeng Lu, Jens Wischnewsky, and Thomas Seidl. A geometric approach for mining sequential patterns in interval-based data streams. In *2016 IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE) 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 2128–2135. IEEE, 2016. doi: 10.1109/FUZZ-IEEE.2016.7737954. URL https://doi.org/10.1109/FUZZ-IEEE.2016.7737954.

[79] Yukai He, Jianyong Wang, and Lizhu Zhou. Efficient incremental mining of frequent sequence generators. In Jeffrey Xu Yu, Myoung-Ho Kim, and Rainer Unland, editors, *Database Systems for Advanced Applications - 16th International Conference, DASFAA 2011, Hong Kong, China, April 22-25, 2011, Proceedings, Part I*, volume 6587 of *Lecture Notes in Computer Science*, pages 168–182. Springer, 2011. doi: 10.1007/978-3-642-20149-3_14. URL https://doi.org/10.1007/978-3-642-20149-3_14.

[80] Yu Hirate and Hayato Yamana. Sequential pattern mining with time intervals. In Wee Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang, editors, *Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference, PAKDD 2006, Singapore, April 9-12, 2006, Proceedings*, volume 3918 of *Lecture Notes in Computer Science*, pages 775–779. Springer, 2006. doi: 10.1007/11731139_90. URL https://doi.org/10.1007/11731139_90.

[81] Frank Höppner and Frank Klawonn. Finding informative rules in interval sequences. In Frank Hoffmann, David J. Hand, Niall M. Adams, Douglas H. Fisher, and Gabriela Guimarães, editors, *Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001, Cascais, Portugal, September 13-15, 2001, Proceedings*, volume 2189 of *Lecture Notes in Computer Science*, pages 125–134. Springer, 2001. doi: 10.1007/3-540-44816-0_13. URL https://doi.org/10.1007/3-540-44816-0_13.

[82] Frank Höppner and Sebastian Peter. Temporal interval pattern languages to characterize time flow. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 4(3):196–212, 2014. doi: 10.1002/widm.1122. URL https://doi.org/10.1002/widm.1122.

[83] Zhengxing Huang, Xudong Lu, and Huilong Duan. On mining clinical pathway patterns from medical behaviors. *Artificial Intelligence in Medicine*, 56(1):35–50, 2012. doi: 10.1016/j.artmed.2012.06.002. URL https://doi.org/10.1016/j.artmed.2012.06.002.

[84] ito, Wynne Hsu, and Mong-Li Lee. Mining relationships among interval-based events for classification. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 393–404. ACM, 2008. doi: 10.1145/1376616.1376658. URL https://doi.org/10.1145/1376616.1376658.

[85] Po-shan Kam and Ada Wai-Chee Fu. Discovering temporal patterns for interval-based events. In Yahiko Kambayashi, Mukesh K. Mohania, and A Min Tjoa, editors, *Data Warehousing and Knowledge Discovery, Second International Conference, DaWaK 2000, London, UK, September 4-6, 2000, Proceedings*, volume 1874 of *Lecture Notes in Computer Science*, pages 317–326. Springer, 2000. doi: 10.1007/3-540-44466-1_32. URL https://doi.org/10.1007/3-540-44466-1_32.

[86] HC Koh and G Tan. Data mining applications in healthcare. *Journal of healthcare information management: JHIM*, 19(2):64–72, 2005.

[87] Kun Lan, Dan-tong Wang, Simon Fong, Liansheng Liu, Kelvin K. L. Wong, and Nilanjan Dey. A survey of data mining and deep learning in bioinformatics. *J. Medical Systems*, 42(8):139:1–139:20, 2018. doi: 10.1007/s10916-018-1003-9. URL https://doi.org/10.1007/s10916-018-1003-9.

[88] Srivatsan Laxman and P Shanti Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, 2006.

[89] Srivatsan Laxman, P. S. Sastry, and K. P. Unnikrishnan. Discovering frequent episodes and learning hidden markov models: A formal connection. *IEEE Trans. Knowl. Data Eng.*, 17(11):1505–1517, 2005. doi: 10.1109/TKDE.2005.181. URL https://doi.org/10.1109/TKDE.2005.181.

[90] Bac Le, Hai V. Duong, Tin C. Truong, and Philippe Fournier-Viger. Fclosm, fgensm: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. *Knowl. Inf. Syst.*, 53(1):71–107, 2017. doi: 10.1007/s10115-017-1032-6. URL https://doi.org/10.1007/s10115-017-1032-6.

[91] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

[92] Jay Lee, Hung-An Kao, and Shanhu Yang. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia Cirp*, 16:3–8, 2014.

[93] Tao Li and Sheng Ma. Mining temporal patterns without predefined time windows. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 451–454. IEEE Computer Society, 2004. doi: 10.1109/ICDM.2004.10016. URL https://doi.org/10.1109/ICDM.2004.10016.

[94] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan-chi Chiu. A symbolic representation of time series, with implications for streaming algorithms. In Mohammed Javeed Zaki and Charu C. Aggarwal, editors, *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD 2003, San Diego, California, USA, June 13, 2003*, pages 2–11. ACM, 2003. doi: 10.1145/882082.882086. URL https://doi.org/10.1145/882082.882086.

[95] Ming-Yen Lin and Suh-Yin Lee. Fast discovery of sequential patterns by memory indexing. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, *Data Warehousing and Knowledge Discovery, 4th International Conference, DaWaK 2002, Aix-en-Provence, France, September 4-6, 2002, Proceedings*, volume 2454 of *Lecture Notes in Computer Science*, pages 150–160. Springer, 2002. doi: 10.1007/3-540-46145-0_15. URL https://doi.org/10.1007/3-540-46145-0_15.

[96] David Lo, Siau-Cheng Khoo, and Jinyan Li. Mining and ranking generators of sequential patterns. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pages 553–564. SIAM, 2008. doi: 10.1137/1.9781611972788.51. URL https://doi.org/10.1137/1.9781611972788.51.

[97] David Lo, Siau-Cheng Khoo, and Limsoon Wong. Non-redundant sequential rules - theory and algorithm. *Inf. Syst.*, 34(4-5):438–453, 2009. doi: 10.1016/j.is.2009.01.002. URL https://doi.org/10.1016/j.is.2009.01.002.

[98] Congnan Luo and Soon M. Chung. Parallel mining of maximal sequential patterns using multiple samples. *The Journal of Supercomputing*, 59(2):852–881, 2012. doi: 10.1007/s11227-010-0476-1. URL https://doi.org/10.1007/s11227-010-0476-1.

[99] Congnan Luo and Soon Myoung Chung. Efficient mining of maximal sequential patterns using multiple samples. In Hillol Kargupta, Jaideep Srivastava, Chandrika Kamath, and Arnold Goodman, editors, *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, pages 415–426. SIAM, 2005. doi: 10.1137/1.9781611972757.37. URL https://doi.org/10.1137/1.9781611972757.37.

[100] Sheng Ma and Joseph L. Hellerstein. Mining partially periodic event patterns with unknown periods. In Dimitrios Georgakopoulos and Alexander Buchmann, editors, *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 205–214. IEEE Computer Society, 2001. doi: 10.1109/ICDE.2001.914829. URL https://doi.org/10.1109/ICDE.2001.914829.

[101] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997. doi: 10.1023/A:1009796218281. URL https://doi.org/10.1023/A:1009796218281.

[102] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997. doi: 10.1023/A:1009796218281. URL https://doi.org/10.1023/A:1009796218281.

[103] Mohsen Marjani, Fariza Nasaruddin, Abdullah Gani, Ahmad Karim, Ibrahim Abaker Targio Hashem, Aisha Siddiqa, and Ibrar Yaqoob. Big iot data analytics: Architecture, opportunities, and open research challenges. *IEEE Access*, 5:5247–5261, 2017. doi: 10.1109/ACCESS.2017.2689040. URL https://doi.org/10.1109/ACCESS.2017.2689040.

[104] Fabian Mörchen. Algorithms for time series knowledge mining. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 668–673. ACM, 2006. doi: 10.1145/1150402.1150485. URL https://doi.org/10.1145/1150402.1150485.

[105] Fabian Mörchen. *Time series knowlegde mining*. PhD thesis, University of Marburg, 2006. URL http://d-nb.info/980289726.

[106] Fabian Mörchen and Dmitriy Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 315–326. SIAM, 2010. doi: 10.1137/1.9781611972801.28. URL https://doi.org/10.1137/1.9781611972801.28.

[107] Robert Moskovitch and Yuval Shahar. Medical temporal-knowledge discovery via temporal abstraction. In *AMIA 2009, American Medical Informatics Association Annual Symposium, San Francisco, CA, USA, November 14-18, 2009*. AMIA, 2009. URL http://knowledge.amia.org/amia-55142-a2009a-1.626575/t-001-1.627318/f-001-1.627319/a-088-1.627488/a-089-1.627485.

[108] Robert Moskovitch and Yuval Shahar. Classification of multivariate time series via temporal abstraction and time intervals mining. *Knowl. Inf. Syst.*, 45(1):35–74, 2015. doi: 10.1007/s10115-014-0784-5. URL https://doi.org/10.1007/s10115-014-0784-5.

[109] Robert Moskovitch and Yuval Shahar. Classification-driven temporal discretization of multivariate time series. *Data Min. Knowl. Discov.*, 29(4):871–913, 2015. doi: 10.1007/s10618-014-0380-z. URL https://doi.org/10.1007/s10618-014-0380-z.

[110] Robert Moskovitch and Yuval Shahar. Fast time intervals mining using the transitivity of temporal relations. *Knowl. Inf. Syst.*, 42(1):21–48, 2015. doi: 10.1007/s10115-013-0707-x. URL https://doi.org/10.1007/s10115-013-0707-x.

[111] Gábor Nagypál and Boris Motik. A fuzzy model for representing uncertain, subjective, and vague temporal knowledge in ontologies. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*, volume 2888 of *Lecture Notes in Computer Science*, pages 906–923. Springer, 2003. doi: 10.1007/978-3-540-39964-3_57. URL https://doi.org/10.1007/978-3-540-39964-3_57.

[112] Fumiya Nakagaito, Tomonobu Ozaki, and Takenao Ohkawa. Discovery of quantitative sequential patterns from event sequences. In Yücel Saygin, Jeffrey Xu Yu, Hillol Kargupta, Wei Wang, Sanjay Ranka, Philip S. Yu, and Xindong Wu, editors, *ICDM Workshops 2009, IEEE International Conference on Data Mining Workshops, Miami, Florida, USA, 6 December 2009*, pages 31–36. IEEE Computer Society, 2009. doi: 10.1109/ICDMW.2009.13. URL https://doi.org/10.1109/ICDMW.2009.13.

[113] Amy Nordum. Popular internet of things forecast of 50 billion devices by 2020 is outdated, august 2016. *URL https://spectrum. ieee. org/tech-talk/telecom/internet/popular-internet-of-things-forecastof-50-billion-devices-by-2020-is-outdated*.

[114] Amy Nordum. Popular internet of things forecast of 50 billion devices by 2020 is outdated, 2016. URL https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.

[115] Eiman Al Nuaimi, Hind Al Neyadi, Nader Mohamed, and Jameela Al-Jaroodi. Applications of big data to smart cities. *J. Internet Services and Applications*, 6(1):25:1–25:15, 2015. doi: 10.1186/s13174-015-0041-5. URL https://doi.org/10.1186/s13174-015-0041-5.

[116] Hans Jürgen Ohlbach. Relations between fuzzy time intervals. In *11th International Symposium on Temporal Representation and Reasoning (TIME 2004), 1-3 July 2004, Tatihou Island, Normandie, France*, pages 44–51. IEEE Computer Society, 2004. doi: 10.1109/TIME.2004.1314418. URL https://doi.org/10.1109/TIME.2004.1314418.

[117] Panagiotis Papapetrou, George Kollios, Stan Sclaroff, and Dimitrios Gunopulos. Mining frequent arrangements of temporal intervals. *Knowl. Inf. Syst.*, 21(2):133–171, 2009. doi: 10.1007/s10115-009-0196-0. URL https://doi.org/10.1007/s10115-009-0196-0.

[118] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In Catriel Beeri and Peter Buneman, editors, *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings.*, volume 1540 of *Lecture Notes in Computer Science*, pages 398–416. Springer, 1999. doi: 10.1007/3-540-49257-7_25. URL https://doi.org/10.1007/3-540-49257-7_25.

[119] Dhaval Patel. *Mining patterns in complex data*. PhD thesis, 2011.

[120] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In Dimitrios Georgakopoulos and Alexander Buchmann, editors, *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 215–224. IEEE Computer Society, 2001. doi: 10.1109/ICDE.2001.914830. URL https://doi.org/10.1109/ICDE.2001.914830.

[121] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11):1424–1440, 2004. doi: 10.1109/TKDE.2004.77. URL https://doi.org/10.1109/TKDE.2004.77.

[122] Jian Pei, Jiawei Han, and Wei Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *J. Intell. Inf. Syst.*, 28(2):133–160, 2007. doi: 10.1007/s10844-006-0006-z. URL https://doi.org/10.1007/s10844-006-0006-z.

[123] Sebastian Peter, Frank Höppner, and Michael R. Berthold. Pattern graphs: A knowledge-based tool for multivariate temporal pattern retrieval. In *6th IEEE International Conference on Intelligent Systems, IS 2012, Sofia, Bulgaria, September 6-8, 2012*, pages 67–73. IEEE, 2012. doi: 10.1109/IS.2012.6335193. URL https://doi.org/10.1109/IS.2012.6335193.

[124] Sebastian Peter, Frank Höppner, and Michael R. Berthold. Learning pattern graphs for multivariate temporal pattern retrieval. In Jaakko Hollmén, Frank Klawonn, and Allan Tucker, editors, *Advances in Intelligent Data Analysis XI - 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings*, volume 7619 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2012. doi: 10.1007/978-3-642-34156-4_25. URL https://doi.org/10.1007/978-3-642-34156-4_25.

[125] Thi-Thiet Pham, Jiawei Luo, and Bay Vo. An effective algorithm for mining closed sequential patterns and their minimal generators based on prefix trees. *IJIIDS*, 7(4):324–339, 2013. doi: 10.1504/IJIIDS.2013.056314. URL https://doi.org/10.1504/IJIIDS.2013.056314.

[126] Clifton Phua, Vincent C. S. Lee, Kate Smith-Miles, and Ross W. Gayler. A comprehensive survey of data mining-based fraud detection research. *CoRR*, abs/1009.6119, 2010. URL http://arxiv.org/abs/1009.6119.

[127] Marc Plantevit, Céline Robardet, and Vasile-Marian Scuturici. Graph dependency construction based on interval-event dependencies detection in data streams. *Intell. Data Anal.*, 20(2):223–256, 2016. doi: 10.3233/IDA-160803. URL https://doi.org/10.3233/IDA-160803.

[128] Parisa Rashidi and Diane J. Cook. Mining sensor streams for discovering human activity patterns over time. In Geoffrey I. Webb, Bing Liu, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu, editors, *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 431–440. IEEE Computer Society, 2010. doi: 10.1109/ICDM.2010.40. URL https://doi.org/10.1109/ICDM.2010.40.

[129] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Trans. Knowl. Data Eng.*, 23(4):527–539, 2011. doi: 10.1109/TKDE.2010.148. URL https://doi.org/10.1109/TKDE.2010.148.

[130] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer, 2011. doi: 10.1007/978-0-387-85820-3_1. URL https://doi.org/10.1007/978-0-387-85820-3_1.

[131] John F. Roddick and Myra Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Trans. Knowl. Data Eng.*, 14(4):750–767, 2002. doi: 10.1109/TKDE.2002.1019212. URL https://doi.org/10.1109/TKDE.2002.1019212.

[132] Guangchen Ruan, Hui Zhang, and Beth Plale. Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data. In Jimmy J. Lin, Jian Pei, Xiaohua Hu, Wo Chang, Raghunath Nambiar, Charu C. Aggarwal, Nick Cercone, Vasant G. Honavar, Jun Huan, Bamshad Mobasher, and Saumyadipta Pyne, editors, *2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*, pages 32–39. IEEE Computer Society, 2014. doi: 10.1109/BigData.2014.7004410. URL https://doi.org/10.1109/BigData.2014.7004410.

[133] Keyvan Mir Mohammad Sadeghi and Ben Goertzel. Uncertain interval algebra via fuzzy/probabilistic modeling. In *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2014, Beijing, China, July 6-11, 2014*, pages 591–598. IEEE, 2014. doi: 10.1109/FUZZ-IEEE.2014.6891863. URL https://doi.org/10.1109/FUZZ-IEEE.2014.6891863.

[134] Eliana Salvemini, Fabio Fumarola, Donato Malerba, and Jiawei Han. FAST sequence mining based on sparse id-lists. In Marzena Kryszkiewicz, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Ras, editors, *Foundations of Intelligent Systems - 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings*, volume 6804 of *Lecture Notes in Computer Science*, pages 316–325. Springer, 2011. doi: 10.1007/978-3-642-21916-0_35. URL https://doi.org/10.1007/978-3-642-21916-0_35.

[135] Steven Schockaert and Martine De Cock. Temporal reasoning about fuzzy intervals. *Artif. Intell.*, 172(8-9):1158–1193, 2008. doi: 10.1016/j.artint.2008.01.001. URL https://doi.org/10.1016/j.artint.2008.01.001.

[136] Guido Sciavicco. Reasoning with time intervals: A logical and computational perspective. *ISRN Artificial Intelligence*, 2012, 2012.

[137] Arie Segev and Arie Shoshani. Logical modeling of temporal data. In Umeshwar Dayal and Irving L. Traiger, editors, *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, CA, USA, May 27-29, 1987*, pages 454–466. ACM Press, 1987. doi: 10.1145/38713.38760. URL https://doi.org/10.1145/38713.38760.

[138] Chayma Sellami, Carlos Miranda, Ahmed Samet, Mohamed Anis Bach Tobji, and François de Beuvron. On mining frequent chronicles for machine failure prediction. *Journal of Intelligent Manufacturing*, Sep 2019. ISSN 1572-8145. doi: 10.1007/s10845-019-01492-x. URL https://doi.org/10.1007/s10845-019-01492-x.

[139] Yuval Shahar. A framework for knowledge-based temporal abstraction. *Artif. Intell.*, 90(1-2):79–133, 1997. doi: 10.1016/S0004-3702(96)00025-2. URL https://doi.org/10.1016/S0004-3702(96)00025-2.

[140] Richard T. Snodgrass. Temporal databases. In Andrew U. Frank, Irene Campari, and Ubaldo Formentini, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning, Pisa, Italy, September 21-23, 1992, Proceedings*, volume 639 of *Lecture Notes in Computer Science*, pages 22–64. Springer, 1992. doi: 10.1007/3-540-55966-3_2. URL https://doi.org/10.1007/3-540-55966-3_2.

[141] Shijie Song, Huaping Hu, and Shiyao Jin. HVSM: A new sequential pattern mining algorithm using bitmap representation. In Xue Li, Shuliang Wang, and Zhao Yang Dong, editors, *Advanced Data Mining and Applications, First International Conference, ADMA 2005, Wuhan, China, July 22-24, 2005, Proceedings*, volume

3584 of *Lecture Notes in Computer Science*, pages 455–463. Springer, 2005. doi: 10.1007/11527503_55. URL https://doi.org/10.1007/11527503_55.

[142] Myra Spiliopoulou. Managing interesting rules in sequence mining. In Jan M. Zytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings*, volume 1704 of *Lecture Notes in Computer Science*, pages 554–560. Springer, 1999. doi: 10.1007/978-3-540-48247-5_73. URL https://doi.org/10.1007/978-3-540-48247-5_73.

[143] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter M. G. Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 1996. doi: 10.1007/BFb0014140. URL https://doi.org/10.1007/BFb0014140.

[144] Nicu-Razvan Stancioiu, Lhouari Nourine, Jean-Marc Petit, Vasile-Marian Scuturici, Dominique Fouchez, Emmanuel Gangler, and Philippe Gris. Discovering injective mapping between relations in astrophysics databases. In Dimitris Kotzinos, Dominique Laurent, Jean-Marc Petit, Nicolas Spyratos, and Yuzuru Tanaka, editors, *Information Search, Integration, and Personlization - 11th International Workshop, ISIP 2016, Lyon, France, November 1-4, 2016, Revised Selected Papers*, volume 760 of *Communications in Computer and Information Science*, pages 18–32. Springer, 2016. doi: 10.1007/978-3-319-68282-2_2. URL https://doi.org/10.1007/978-3-319-68282-2_2.

[145] Liang Tang, Tao Li, and Larisa Shwartz. Discovering lag intervals for temporal dependencies. In Qiang Yang, Deepak Agarwal, and Jian Pei, editors, *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 633–641. ACM, 2012. doi: 10.1145/2339530.2339633. URL https://doi.org/10.1145/2339530.2339633.

[146] Anne Tchounikine, Maryvonne Miquel, Thierry Pécout, and Jean-Luc Bonnaud. Prosopographical data analysis. application to the angevin officers (xiii?xv centuries). *JDMDH*, 2018, 2018. URL https://jdmdh.episciences.org/4553.

[147] Divya Tomar and Sonali Agarwal. A survey on data mining approaches for healthcare. *International Journal of Bio-Science and Bio-Technology*, 5(5):241–266, 2013.

[148] Minh-Thai Tran, Bac Le, and Bay Vo. Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently. *Eng. Appl. of AI*, 38:183–189, 2015. doi: 10.1016/j.engappai.2014.10.021. URL https://doi.org/10.1016/j.engappai.2014.10.021.

[149] Thien-Trang Van, Bay Vo, and Bac Le. Mining sequential rules based on prefix-tree. In Ngoc Thanh Nguyen, Bogdan Trawinski, and Jason J. Jung, editors, *New Challenges for Intelligent Information and Database Systems [original works presented during a poster session organized within the 3rd Asian Conference on Intelligent Information and Database Systems, 20-22 April 2011, Daegu, Korea]*, volume 351 of *Studies in Computational Intelligence*, pages 147–156. Springer, 2011. doi: 10.1007/978-3-642-19953-0_15. URL https://doi.org/10.1007/978-3-642-19953-0_15.

[150] Thien-Trang Van, Bay Vo, and Bac Le. Imsr_pretree: an improved algorithm for mining sequential rules based on the prefix-tree. *Vietnam J. Computer Science*, 1(2):97–105, 2014. doi: 10.1007/s40595-013-0012-3. URL https://doi.org/10.1007/s40595-013-0012-3.

[151] Alexandre Vautier, Marie-Odile Cordier, and Rene Quiniou. An inductive database for mining temporal patterns in event sequences. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1640–1641. Professional Book Center, 2005. URL http://ijcai.org/Proceedings/05/Papers/post-0342.pdf.

[152] Roy Villafane, Kien A. Hua, Duc A. Tran, and Basab Maulik. Knowledge discovery from series of interval events. *J. Intell. Inf. Syst.*, 15(1):71–89, 2000. doi: 10.1023/A:1008781812242. URL https://doi.org/10.1023/A:1008781812242.

[153] Jason Tsong-Li Wang, Mohammed Javeed Zaki, Hannu Toivonen, and Dennis E. Shasha. Introduction to data mining in bioinformatics. In Jason Tsong-Li Wang, Mohammed Javeed Zaki, Hannu Toivonen, and Dennis E. Shasha, editors, *Data Mining in Bioinformatics*, pages 3–8. Springer, 2005.

[154] Jianyong Wang and Jiawei Han. BIDE: efficient mining of frequent closed sequences. In Z. Meral Özsoyoglu and Stanley B. Zdonik, editors, *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*, pages 79–90. IEEE Computer Society, 2004. doi: 10.1109/ICDE.2004.1319986. URL https://doi.org/10.1109/ICDE.2004.1319986.

[155] Jianyong Wang, Jiawei Han, and Chun Li. Frequent closed sequence mining without candidate maintenance. *IEEE Trans. Knowl. Data Eng.*, 19(8):1042–1056, 2007. doi: 10.1109/TKDE.2007.1043. URL https://doi.org/10.1109/TKDE.2007.1043.

[156] Wentao Wang, Chunqiu Zeng, and Tao Li. Discovering multiple time lags of temporal dependencies from fluctuating events. In Yi Cai, Yoshiharu Ishikawa, and Jianliang Xu, editors, *Web and Big Data - Second International Joint Conference, APWeb-WAIM 2018, Macau, China, July 23-25, 2018, Proceedings, Part II*, volume 10988 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2018. doi: 10.1007/978-3-319-96893-3_10. URL https://doi.org/10.1007/978-3-319-96893-3_10.

[157] Gary M. Weiss. Data mining in telecommunications. In Oded Maimon and Lior Rokach, editors, *The Data Mining and Knowledge Discovery Handbook.*, pages 1189–1201. Springer, 2005.

[158] Andrew Whitmore, Anurag Agarwal, and Li Da Xu. The internet of things - A survey of topics and trends. *Information Systems Frontiers*, 17(2):261–274, 2015. doi: 10.1007/s10796-014-9489-2. URL https://doi.org/10.1007/s10796-014-9489-2.

[159] Edi Winarko and John F. Roddick. ARMADA - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data Knowl. Eng.*, 63(1):76–90, 2007. doi: 10.1016/j.datak.2006.10.009. URL https://doi.org/10.1016/j.datak.2006.10.009.

[160] Shin-yi Wu and Yen-Liang Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE Trans. Knowl. Data Eng.*, 19(6):742–758, 2007. doi: 10.1109/TKDE.2007.190613. URL https://doi.org/10.1109/TKDE.2007.190613.

[161] Shin-yi Wu and Yen-Liang Chen. Discovering hybrid temporal patterns from sequences consisting of point- and interval-based events. *Data Knowl. Eng.*, 68(11):1309–1330, 2009. doi: 10.1016/j.datak.2009.06.010. URL https://doi.org/10.1016/j.datak.2009.06.010.

[162] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, pages 166–177. SIAM, 2003. doi: 10.1137/1.9781611972733.15. URL https://doi.org/10.1137/1.9781611972733.15.

[163] Abdulsalam Yassine, Shailendra Singh, and Atif Alamri. Mining human activity patterns from smart home big data for health care applications. *IEEE Access*, 5:13131–13141, 2017. doi: 10.1109/ACCESS.2017.2719921. URL https://doi.org/10.1109/ACCESS.2017.2719921.

[164] Illhoi Yoo, Patricia Alafaireet, Miroslav Marinov, Keila Pena-Hernandez, Rajitha Gopidi, Jia-Fu Chang, and Lei Hua. Data mining in healthcare and biomedicine: A survey of the literature. *J. Medical Systems*, 36(4):2431–2448, 2012. doi: 10.1007/s10916-011-9710-5. URL https://doi.org/10.1007/s10916-011-9710-5.

[165] Mariko Yoshida, Tetsuya Iizuka, Hisako Shiohara, and Masanori Ishiguro. Mining sequential patterns including time intervals. In Belur V. Dasarathy, editor, *Data Mining and Knowledge Discovery: Theory, Tools, and Technology II, Orlando, FL, USA, April 24, 2000*, volume 4057 of *SPIE Proceedings*, pages 213–220. SPIE, 2000. doi: 10.1117/12.381735. URL https://doi.org/10.1117/12.381735.

[166] Mohammed J. Zaki and Wagner Meira. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2014. ISBN 9780521766333. URL http://www.dataminingbook.info/.

[167] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12 (3):372–390, 2000. doi: 10.1109/69.846291. URL https://doi.org/10.1109/69.846291.

[168] Mohammed Javeed Zaki. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001. doi: 10.1023/A:1007652502315. URL https://doi.org/10.1023/A:1007652502315.

[169] Mohammed Javeed Zaki and Ching-Jui Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. Knowl. Data Eng.*, 17(4):462–478, 2005. doi: 10.1109/TKDE.2005.60. URL https://doi.org/10.1109/TKDE.2005.60.

[170] Chunqiu Zeng, Liang Tang, Tao Li, Larisa Shwartz, and Genady Grabarnik. Mining temporal lag from fluctuating events for correlation and root cause analysis. In Danny Raz, Michele Nogueira, Edmundo Roberto Mauro Madeira, Brendan Jennings, Lisandro Zambenedetti Granville, and Luciano Paschoal Gaspary, editors, *10th International Conference on Network and Service Management, CNSM 2014 and Workshop, Rio de Janeiro, Brazil, November 17-21, 2014*, pages 19–27. IEEE Computer Society, 2014. doi: 10.1109/CNSM.2014.7014137. URL https://doi.org/10.1109/CNSM.2014.7014137.

[171] Chunqiu Zeng, Liang Tang, Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Ya. Grabarnik. An integrated framework for mining temporal logs from fluctuating events. *IEEE Trans. Services Computing*, 12(2):199–213, 2019. doi: 10.1109/TSC.2016.2598747. URL https://doi.org/10.1109/TSC.2016.2598747.

[172] Li Zhang, Guoqing Chen, Tom Brijs, and Xing Zhang. Discovering during-temporal patterns (dtps) in large temporal databases. *Expert Syst. Appl.*, 34(2):1178–1189, 2008. doi: 10.1016/j.eswa.2006.12.024. URL https://doi.org/10.1016/j.eswa.2006.12.024.

[173] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996.*, pages 103–114. ACM Press, 1996. doi: 10.1145/233269.233324. URL https://doi.org/10.1145/233269.233324.

UNIVERSITÉ DE LYON

INSA

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : EL OUASSOULI
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 21/07/2020

Prénoms : Amine

TITRE : Discovering Complex Quantitative Dependencies between Interval-based State Streams

NATURE : Doctorat

Numéro d'ordre : AAAALYSEIXXXX

Ecole doctorale : INFORMATIQUE ET MATHEMATIQUE DE LYON – ED512

Spécialité : Informatique

RESUME :
Les avancées significatives qu'ont connu les technologies de capteurs, leur utilisation croissante ainsi que leur intégration dans les systèmes d'information permettent d'obtenir des descriptions temporelles riches d'environnements réels. L'information générée par de telles sources de données peut être qualifiée d'hétérogène sur plusieurs plans : types de mesures physiques, domaines et primitives temporelles, modèles de données etc. Dans ce contexte, l'application de méthodes de fouille de motifs constitue une opportunité pour la découverte de relations temporelles non-triviales, directement utilisables et facilement interprétables décrivant des phénomènes complexes. Nous proposons d'utiliser un ensemble d'abstraction temporelles pour construire une représentation unifiée, sous forme des flux d'intervalles (ou états), de l'information générée par un système hétérogène. Cette approche permet d'obtenir une description temporelle de l'environnent étudié à travers des attributs (ou états), dits de haut niveau, pouvant être utilisés dans la construction des motifs temporelles. A partir de cette représentation, nous nous intéressons à la découverte de dépendances temporelles quantitatives (avec information de délais) entre plusieurs flux d'intervalles. Nous introduisons le modèle de dépendances Complex Temporal Dependency (CTD) défini de manière similaire à une forme normale conjonctive. Ce modèle permets d'exprimer un ensemble riche de relations temporelles complexes. Pour ce modèle de dépendances nous proposons des algorithmes efficaces de découverte : CTD-Miner et ITLD - Interval Time Lag Discovery. Finalement, nous évaluons les performances de notre proposition ainsi que la qualité des résultats obtenus à travers des données issues de simulations ainsi que des données réelles collectées à partir de caméras et d'analyse vidéo.

MOTS-CLÉS : Fouille de données temporelles – Flux de données – Intervalles temporels

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse:
Dr. Vasile-Marian SCUTURICI (INSA-LYON)

Président de jury :

Composition du jury :
Pr. Farouk TOUMANI (Université Blaise Pascal)
Dr. Reza AKBARINIA (INRIA, LIRMM)
Pr. Fréderique LAFOREST (INSA-LYON)
Pr. Karine ZEITOUNI (Université de Versailles)
Dr. Lionel ROBINAULT (FOXSTREAM)
Dr. Vasile-Marian SCUTURICI (INSA-LYON)