# Algorithms and Measures in Sequence Data Mining

Ion Răilean
PhD Thesis

TELECOM Bretagne

**N◦ d'ordre** : 20XXtelbXXXXXX

**Sous le sceau de l'Université européenne de Bretagne**

**Technical University of Cluj-Napoca**

**Télécom Bretagne**

**En habilitation conjointe avec l'Université de Bretagne Sud**

Doctoral School – SIDOC - SICMA

# Algorithms and Measures in Sequence Data Mining

**PhD Thesis**

Mention : « Inginerie Electronică Şi Telecomunicaţii (UTCN)
Sciences et Technologies de l'Information et de la Communication (TB) »

Presented by **Ion Răilean**

Department : Communications (UTCN); LUSSI (TB)

Laboratory : UMR CNRS 3192 Lab-STICC

Thesis Director : Prof. Dr. Ing. Monica Borda

Co-directeur : Prof. Philipppe Lenca
Maître de conf. Sorin Moga

Defended on September 28th 2012

**Jury :**

|  |  |
|---|---|
| Repporters : | Prof. Surname Name |
|  | Surname Name,University, France |
| Examinators : | Surname Name, University |
|  | Surname Name, University |
|  | Surname Name, University |
| Directors : | Prof. Dr. Ing. Monica Borda, Technical University of Cluj-Napoca, Romania |
|  | Prof. Philippe Lenca, Télécom Bretagne, France |
| Guests : | Surname Name, University |
|  | Surname Name, University |

# Contents

**Contents** iii

**Abstract** v

**Résumé** vii

**1 Introduction** 1

**2 Sequential Rules and Patterns Mining** 5
- 2.1 Introduction 5
- 2.2 State of the Art 7
- 2.3 Theoretical Fundamentals 9
- 2.4 Closeness Preference Interestingness Measure for Sequential Rules Selection 12
  - 2.4.1 Definition 12
  - 2.4.2 CP Properties 16
  - 2.4.3 Toy Example 17
  - 2.4.4 Closeness Preference Measure in Real Application 18
- 2.5 Modified Closeness Preference Measure for Patterns extraction 27
  - 2.5.1 Closeness Weight implementation 28
  - 2.5.2 $MCP_{sc}$: Support - Confidence - Closeness Preference based measure 30
  - 2.5.3 $MCP_{s\ func}$: Support - Closeness Preference based measure combined with an additional weighting function to fulfil the anti-monotone property 31
  - 2.5.4 Pattern Mining and $MCP_{sc}$ with $MCP_{s\ func}$ Properties 33
  - 2.5.5 Toy Example 36
  - 2.5.6 Real Database analysis 41
- 2.6 Conclusions 45

**3 Time Series** 47
- 3.1 Introduction 47
- 3.2 State of the Art 48
- 3.3 Theoretical Fundamentals 51
  - 3.3.1 Wavelets 51
  - 3.3.2 Neural Networks 59
  - 3.3.3 Genetic Algorithms 65
- 3.4 Proposed Forecasting Model 70
  - 3.4.1 Sidebar 70
  - 3.4.2 Data for Analysing 72
  - 3.4.3 Wavelet Decomposition 73
  - 3.4.4 Neural Network Design 76
  - 3.4.5 Genetic Algorithm Optimization 87

3.5 Evaluation of the Prediction Model . . . . . . . . . . . . . . . . . . . . . . . 92
3.5.1 Additional Forecasting Models . . . . . . . . . . . . . . . . . . . . . 93
3.5.2 Measuring the Accuracy . . . . . . . . . . . . . . . . . . . . . . . . . 94
3.5.3 WiMAX Traffic ($W67$ and $W504$ databases) . . . . . . . . . . . . . 96
3.5.4 EUR/USD Currency Exchange ($EUVol$ and $EURate$ databases) . . 99
3.6 Conlcusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101

**4 Conclusions and Perspectives** **103**

**Index** **107**

**Bibliography** **107**

**List of Figures** **122**

**List of Tables** **124**

**Annexes** **125**
A Classification of sequential pattern mining algorithms . . . . . . . . . . . . 125
A.1 First category . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
A.2 Second category . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
A.3 Third category . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127
B Relational representation of existing algorithms. Used databases . . . . . . 128
C Differences regarding complexity . . . . . . . . . . . . . . . . . . . . . . . 134

# Abstract

The increasing amount of information makes sequential data mining an important domain of research. A vast number of data mining models and approaches have been developed in order to extract interesting and useful patterns of data. Most models are used for strategic purposes resulting in using of the *time* parameter. However, the extensive field of data mining applications requires new models to be introduced. The current thesis proposed models for temporal sequential data mining having as a goal the forecasting process. We focus our study on sequential temporal database analysis and on time-series data. In sequential database analysis we propose several interestingness measures for rules selection and patterns extraction. Their goal is to advantage those rules/patterns whose time-distance between the itemsets is small. The extracted information is used to predict user's future requests in a web log database, obtaining a higher performance in comparison to other compared models. In time-series analysis we propose a forecasting model based on Neural Networks, Genetic Algorithms, and Wavelet Transform. We apply it on a WiMAX network traffic and EUR/USD currency exchange data in order to compare its prediction performance with those obtained using other existing models. Different ways of changing parameters adapted to a given situation and the corresponding simulations are presented. It was shown that the proposed model outperforms the existing ones from the prediction point of view on the used time-series. As a whole, this thesis proposes forecasting models for different types of temporal sequential data with different characteristics and behaviour.

# Résumé

La quantité croissante d'information rend l'exploration de données séquentielles un domaine important de la recherche. Un grand nombre de modèles d'exploration de données et d'approches ont été développés afin d'extraire des patterns intéressants et utiles de données. La plupart des modèles sont utilisés avec des objectifs stratégiques résultant en l'utilisation du paramètre de temps. Cependant, le vaste champ d'applications de data mining nécessite l'introduction de nouveaux modèles. La thèse en cours propose des modèles pour l'exploration de données temporelle séquentielle ayant comme objectif le processus de prévision. Notre étude se concentre sur l'analyse de base de données séquentielles et temporelles et de données time-series. Dans l'analyse de base de données séquentielles, nous proposons plusieurs mesures d'intérêt pour la sélection des règles et l'extraction des patterns. Leur but est de mettre en valeur ces règles / patterns dont la distance-temps entre les itemsets est petit. L'information extraite est utilisée pour prédire les futures demandes des utilisateurs dans une base de données web log, l'obtention d'un rendement plus élevé en comparaison avec d'autres modèles. En ce qui concerne l'analyse des séries temporelles, nous proposons un modèle de prévision basé sur les réseaux de neurones, algorithmes génétiques, et la transformée en ondelettes. Nous l'appliquons sur un trafic réseau WiMAX et de données de change EUR/USD afin de comparer ses performances de prédiction à celles obtenues avec d'autres modèles existants. Différentes façons de modification des paramètres adaptés à une situation donnée et les simulations correspondantes sont présentées. Nous avons montré que le modèle proposé surpasse ceux déjà existants du point de vue de la prédiction sur les time-series que nous avons utilisées. Dans son ensemble, cette thèse propose des modèles de prévision pour différents types de données séquentielles et temporelles avec des caractéristiques et des comportements différents.

# Introduction

The rapid growth of online information due to Internet, computer storage and the widespread use of database technologies, have created a massive need for data analysis methods in order to find relationships in it and seek solutions to difficult problems. With this goal, *Data Mining* techniques are used, which are the processes of analysing the information and extract interesting knowledge from it (*patterns*) that might otherwise remain unknown.

The extracted information from applying data mining analysis, is mostly used for strategic purposes. For example, many card-thieves have the same behaviour – they purchase expensive luxury goods, which are unusual to the card-holder. By using data mining systems and discovering the unexpected behaviour in the account operations, the losses could be minimized [Liu 08, Singh 09a]. Telecommunication engineers use data mining to predict errors and failures in their systems [Weiss 01, Devitt 05]. Google (and other websites) uses data mining techniques in advertising, showing us those ads which we are more interested in, after analysing our search queries on its engine [Tuğ 06, Liu 07]. Technical analysts use data mining in the financial market to identify patterns in order to forecast future price movements to obtain higher profits [Panda 07, Aggarwal 09, Rai 11]. Biologists use data mining to prevent a disease, after studying a set of symptoms experienced by the patients [Ordonez 06, Ohsaki 07, Klema 08]. The list of application areas for data mining is vast, and is expected to grow rapidly in the years to come.

Nowadays, there exist a large number of data mining models to extract valid, novel, potentially useful, and ultimately understandable patterns in data [Fayyad 96]. Important data mining models are *Clustering*, *Association/Sequential Rules extraction*, *Sequential Analysis*. Clustering analyses a set of data and divides it into groups of similar objects based on the features present in a set of data from the same class [Jain 88, Agrawal 98, Bradley 98]. It is used in Internet (discovering groups of similar access patterns), biology (classification of plants and animals according to their features). Association and sequential rules extraction imply the finding of certain relationships among a set of objects in a database (*i.e.* these objects occur together or one implies the other) [Agrawal 93, Agrawal 94]. These are mostly used in mining transaction data (*e.g.* if a customer bought items A and B, then he will by the item C next), medical diagnosis (if a patient has the symptoms A and B, then he is having the disease C). Sequential analysis aims to discover frequent (or interesting from the user's point of view) patterns that occur in a sequential database [Agrawal 95]. It is used in biology (in finding DNA sequences, gene structures), Weblog click streams (to discover user access patterns).

An interesting parameter in databases used by data mining analysis is the *time* parameter. It denotes the time period during which a certain action is true (*e.g.* the time-interval of having breakfast or searching the internet); or the time instance at which an action was performed (*e.g.* he went out at 9h00 AM, the items have been bought yesterday). Many data mining problems involve this temporal aspect resulting in *Temporal Data Mining* (TDM) [Lin 02]. It has the capacity to look for interesting correlations, rules, patterns in large temporal data sets, which might be missed if temporal component is not taken into consideration [Roddick 02]. Typically, the sequential rules and sequential patterns mining are those models which associate a time-stamp to its elements. The problem of TDM consists in mining patterns from ordered data with temporal interdependencies, which could be either *sequences* or numerical *time-series* [Antunes 01, Laxman 06, Mehta 11]. A sequence is composed by a series of nominal symbols from a particular alphabet (*e.g.* customer transactions logs, protein sequences), while a time-series is a *sequence* of continuous, real-valued elements (*e.g.* temperature values monitoring, price of a stock). In the current work we view as sequential data both: sequences and time-series. Both these data types are sequential by their nature, the only difference being the alphabet used by sequences and the real-valued elements used by time-series.

Initial work in TDM reasoning that refined the temporal relationships used in later TDM research was defined in [Allen 83, Dean 87, Freksa 92]. The goal in sequential TDM is to extract temporal patterns that are used to support diagnosis and to predict future behaviours. The sequential TDM is based on conventional sequential pattern mining algorithms while taking into consideration the temporal aspect also. Such conventional algorithms are *AprioriAll* [Agrawal 95], *Generalized Sequential Patterns* [Srikant 96], *pattern-growth approach* [Pei 04], *CMRules* [Fournier-Viger 12]. Their goal is to extract patterns and rules after a processing with an *Interestingness Measure* (IM) (or several IM used as post-treatment on the extracted information). The task of the IM is to determine the most useful patterns and rules from the user's perspective by selecting them from a large set of candidates. In order to take into account the temporal aspect in the mining process, one has to consider the time parameter into such an IM. In this way, the IM will use in its computation formula the time distances between the itemsets of a pattern or rule. Some of the IMs which take into consideration the temporal aspect of a pattern or rule are *generalized information gain* [Yang 02] (penalizing the gaps between pattern occurrences), *Time-interval Weighted-support* [Chang 11] (which uses time-interval decreasing weighted functions disadvantaging the distance between the itemsets of a pattern).

In comparison to sequential TDM, the time-series analysis has a long history [Box 70]. It is used for providing explanations for data pattern changes and to forecast future values of data points being analysed. Contrary to sequential TDM, the observations in time-series are taken at a constant time-interval having a natural temporal ordering. This allows them to be frequently plotted using line charts resulting in a more understandable view of the data. There are many models for time-series analysis, from which we distinguish *Case-Based Reasoning* models [Kolodner 92, Riordan 02]; *Rule-Based Forecasting* models [Webb 03]; *Statistical* models [Mandal 06, abd T. Nochai 06]; *Artificial Intelligence* models [Refenes 93, Abrahart 98].

The current problem in temporal sequential data mining is that extensive field of data mining applications and the challenge of prediction processes brought to the point where different models are required for data analysis and forecasting of different data types. This is needed despite significant advances in research over the last few decades. In this work we propose models for sequential temporal data mining having as a goal the forecasting aspect of the TDM [Lin 02]. We center our research on sequential temporal database analysis, by introducing new time-based interestingness measures for sequential rules and patterns extraction used for prediction purposes, and on time-series analysis by proposing a forecasting model.

The current thesis is divided into two main parts. Each one deals with a type of sequential data, *i.e.* sequences and time-series.

The goal of the first part of our research is to introduce Interestingness Measures for sequential rule and pattern mining that take into consideration the time parameter and are used to build accurate forecasting models. In sequential rule mining, we propose the *Closeness Preference* (CP) IM. It takes into consideration the time-distance between the antecedent and consequent of a sequential rule in order to advantage those which occur closer one to another. In the sequential pattern mining, we present two types of *Modified Closeness Preference* (MCP) measure: the first one is based on *Support* and *Confidence* ($MCP_{sc}$), while the second one is based on *Support* combined with a new *weighting function* in order to fulfil the anti-monotone property of the IM ($MCP_{s\ func}$). Their goal is to rank at the top the patterns which have their itemsets closer with respect to time. They are pre-processing measures used instead of *Support*. These extracted sequential patterns and rules could be used in market basket analysis (to predict an item being purchased), web analysis (for predicting the pages that will be visited), network security (to prevent intrusion from an unusual activity).

In the second part of our research, *i.e.* in case of time-series analysis, our goal is to propose and study models used in forecasting of time-series with different characteristic. We present the prediction performance of different models applied on different series in order to see if there are forecasting models that could be used in different domains. We study how the selection of parameters could improve the performance of a prediction model, and how a constant upgrading with the latest available information and retraining the model influences the effectiveness of the prediction. We propose a forecasting model based on time-series decomposition in wavelet domain (*Stationary Wavelet Transform* (SWT) using *Artificial Neural Networks* (ANN) and their optimization using *Genetic Algorithms* (GA). The proposed approaches are used in domains where the forecasting of future real-valued data samples is needed (*e.g.* trend detection, transaction volumes of a retail company).

Each part is based on results on real datasets. The thesis ends with a conclusion of our work followed by future perspectives.

# Sequential Rules and Patterns Mining

## 2.1 Introduction

Sequential pattern mining, introduced in [Agrawal 95] and considered as one of the most challenging problems in data mining [Yang 06], aims to extract the relationships between occurrences of sequential events. It has large applications, such as the analysis of DNA sequences, stock marketing, web access patterns, transactional databases, security of the network systems [Zhao 03].

A *sequential pattern* or a *sequence* $AB$ is simply an ordered list of itemsets [Agrawal 95]. From sequential patterns one can obtain sequential rules of the form $A \rightarrow B$, where $A$ (the antecedent) and $B$ (the consequent) are two itemsets. Sequential pattern and rule mining aims at extracting those patterns and rules whose number of appearances, *i.e.* their *Support*, is higher than a minimum *Support* threshold [Agrawal 95]. The *Support* of the rule $A \rightarrow B$ is defined as the fraction of total sequences which support this rule. Consequently, the *Support* of a pattern $AB$ is defined as the fraction of the total sequences which support this pattern.

A typical example of a sequential pattern is a customer who, after buying a laser printer, returns to buy a scanner (after one month) and then a CD burner (after another one month) [Chang 11], which could be written as:

$$printer \xrightarrow{1\ month} scanner \xrightarrow{1\ month} CD\ burner$$

From this, we can construct sequential rules of the form:

$$printer \xrightarrow{1\ month} scanner$$

$$printer \xrightarrow{2\ months} CD\ burner$$

$$scanner \xrightarrow{1\ month} CD\ burner$$

In this example, the pattern and the rules "tell" the store's manager which items were bought and in which order.

In order to extract the rules and patterns, many algorithms have been introduced for sequential mining over the last decade. Besides the first ones, *AprioriAll*, *AprioriSome*, and *DynamicSome* presented in [Agrawal 95] many improvements have been proposed such as with *Generalized Sequential Patterns* [Srikant 96], *SPADE* [Zaki 01], *GO-SPADE* [Leleu 03], the *pattern-growth approach* [Pei 04], *ExMiner* [Quang 06], *WS-pan* [Yun 06], *Graph for Time Constraints* [Masseglia 09], *CMRules* [Fournier-Viger 12]. A nice taxonomy of sequential pattern mining techniques and algorithms is presented in [Mabroukeh 10]. The most common approach consists in discovering ordered itemsets which appear frequently in the sequence database.

During the mining process, one can obtain many patterns and rules which are not relevant from the user's point of view. One of the most interesting and difficult approach to reduce the number of patterns and rules, is to construct *Interestingness Measures* (IM) used to show the quality of a given pattern/rule and to select and rank the patterns/rules according to their potential interest to the user. Many studies have been made regarding the formalization of rule/pattern interestingness, and concerning the substitution to human's evaluation of them using formalized interestingness measures [Ohsaki 07]. These measures could be categorized as follows [Geng 06]:

- *Objective Measures*: based on probability, or on the form of the rules (such as peculiarity, surprisingness, conciseness);
- *Subjective Measures*: which take into account both the data and the user's knowledge (such as surprisingness, novelty);
- *Semantic Measures*: which consider the semantics and explanations of the patterns or rules (such as utility, actionability).

In the current research we focus on objective measures. Based on the data distribution, the objective interestingness measures can evaluate a rule via its statistical factors. Depending on the user's point of view, each objective IM reflects his/her interests on the data [Lenca 08].

The most used measures for evaluating the quality of the extracted information is *Support* (in case of pattern mining), and *Support* and *Confidence* (in case of rule mining) [Mannila 97] . However it is well-known that the *Support* and the *Confidence* measures are not enough to consider the interestingness of those sequential rules and patterns. Another aspect consists in the fact that sequential patterns/rules and items within them have been treated uniformly, while, in reality these have different importance. Other criteria are needed in order to discover those patterns/rules which are interesting from the user and domain application points of view. This is why lately different approaches have been presented regarding *interestingness* related to the sequential extracted data to select or rank it. Many of those take into consideration the weights of the items and the time-distance between the itemsets [Lo 05, Yun 08]. Indeed, it could be desirable to weight recent events more heavily than remote events [Lo 05, Yun 08, Chang 11]. If we retake our example with the buyer, then, the rule $\textit{printer} \xrightarrow{\textit{1 month}} \textit{scanner}$ is more important than $\textit{printer} \xrightarrow{\textit{2 months}} \textit{CD burner}$, because the time distance between the events is smaller in the first case. Temporal constraints which bound the distance between a pair of events have been introduced in [Bettini 98b]. Such constraints like regularity has recently attracted attention [Tanbeer 09, Surana 11, Amphawan 12]. It can also be useful to use interestingness measures in addition to *Support* that help to select the good rules. For example [Zhao 08] considers the *Lift* [Brin 97] in addition to the *Support* and the *Confidence* to evaluate positive and negative sequential rules.

During the data mining process, interestingness measures could be used mostly in 2 ways, as presented in Figure 2.1. First of all, measures could be used to prune uninteresting patterns/rules during the mining process in order to diminish the search space and improve the mining efficiency. This is the role of the *pre-processing* measures most of which are based on the anti-monotone property of a measure stating that the measure's value of a pattern must be no greater than the measure's value of its subpatterns [Agrawal 94] An example of this measure is *Support*, as presented in [Agrawal 94], where the user can set a threshold to eliminate the information which appears not enough times in the database. Then, the measures can be used to filter the extracted information in a *post-processing step*, in order to obtain the final patterns and rules, as in the case of *Confidence* and *Lift* [Mannila 97].

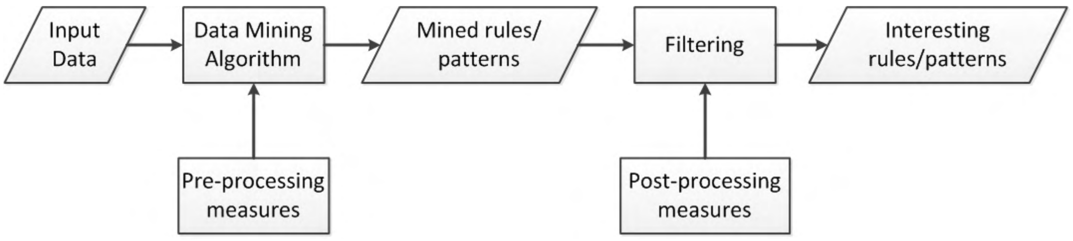


Figure 2.1 : Roles of interestingness measures in data mining process.

In our research we take into consideration both types of measures: *pre-* and *post-processing* ones. We consider the time-distance between the events also.

Regarding the sequential rule mining, we here propose post-processing interestingness measure, *Closeness Preference* (*CP*), that takes into consideration the time interval to meet the user's preference of selecting the rules with closer antecedent and consequent in a post-processing step (Section 2.4).

For the sequential pattern mining, we present two types of a *Modified CP* measure that are used in the extraction and ranking of sequential patterns (Section 2.5). The first one is *Support-Confidence-CP* based measure ($MCP_{sc}$ (*Modified CP*), Subsection 2.5.2), while the second one is *Support-CP* based combined with a weighting function that as results fulfils the anti-monotone property ($MCP_{s\ func}$, Subsection 2.5.3). The aim of both measures is to select the patterns with closer itemsets and are used instead of the *Support*, thus, they are pre-processing measures, the difference being that the first one does not fulfil the *apriori* principle, while the second does. Our results on a real dataset show that the new measures are able to evidentiate the sequential patterns with closer events, and present different results in comparison to the *Support* mining (Subsections 2.4.4 and 2.5.6). Such patterns and rules could be used in web analysis (for predicting the pages that will be visited), marketing (to find the next items that would be bought), network security (to prevent intrusion from unwanted packages).

## 2.2 State of the Art

In the current part we briefly review the related works developed to improve the usefulness of sequential rules and patterns following some user's expectations (*e.g.* taking into account time interval, importance between items and interestingness measures to focus on a special kind of rules). As it has been stated in Section 2.1 we are concerned about Interestingness Measures, and not the extraction algorithms. However, a review of existing methods for patterns and rules extraction are presented in the Annexes 4.

We first review some post-processing techniques used in sequential data mining. Obviously the *Confidence* ($P(B|A)$) measure is often used (where $P(X)$ is the probability of $X$). It measures the accuracy of a given rule. But it can produce misleading results when the *Support* of the rule is higher than the rule's *Confidence*. For example, if the itemset $A$ happens in 25% of the cases, while the itemset $B$ in 90%, then the events $A$ and $B$ might be completely independent, and the rule $A \rightarrow B$ might not be statistically significant even if its *Confidence* is high. *Confidence* and *Lift* ($\frac{P(AB)}{P(A) \cdot P(B)}$) are used for example to find the unexpected sequential rules [Spiliopoulou 99], to discover atherosclerosis risks [Klema 08], or to monitor a network [Costa 09]. The *J-Measure* [Smyth 91]

$$\textit{J-Measure} = P(A)\left(P(B|A) \cdot \log\left(\frac{P(B|A)}{P(B)}\right) + (1 - P(B|A)) \cdot \log\left(\frac{1-P(B|A)}{1-P(B)}\right)\right)$$

is used to rank the rules in time-series [Das 98, Höppner 02] or in interval sequences [Höppner 01] due to its good properties. The *generalized information gain* which penalizes the gaps between pattern occurrences is proposed in [Yang 02]:

*The Generalized Information Gain of $D$ (a sequence of events) with respect to $P$ (a pattern) is defined as $I(P) \times (S_D(P) - 1) - L_D(P)$, where $I(P)$, $S_D(P)$, and $L_D(P)$ are the information of $P$, the Support of $P$ within $D$, and the information loss of $D$ with respect to $P$, respectively.*

The *Sequential Implication Intensity* measure which evaluates the statistical significance of the rules in comparison with a probabilistic model is proposed in [Blanchard 08]:

$$SII(A \xrightarrow{\omega} B) = 1 - \sum_{k=0}^{N_{A\bar{B}}(\omega)} \mathrm{C}_{n_A}^{k} (e^{-\frac{\omega}{L}n_B})^k (1 - e^{-\frac{\omega}{L}n_B})^{n_A - k}$$

where $\omega$ is a time window, while $n_A$, $n_B$, $n_{A\bar{B}}$ are the number of the itemsets $A$, $B$ in the database and of the counterexamples respectively.

An interesting empirical comparison between several interestingness measures for association rules and sequential patterns for web mining is done in [Huang 02, Huang 07]. It is shown that some measures are much more effective than others: some give the best ranking performances of the sequential patterns, some contain more variety of high-ranked patterns while others are more specialized. Thus, a user should use a measure that is suitable for his application and interest.

Let's now consider pre-processing approaches. A practical application to identify patterns in network alarm logs in order to predict telecommunication equipment failures is presented in [Weiss 01]. Patterns are generated with a genetic algorithm where the fitness of the prediction pattern is based on its *F-measure* (*i.e.* on both its *precision* and *recall*). However most of the pre-processing approaches are based on the *anti-monotone property* (property defined in Section 2.1) as it was defined for the case of the *Support* in [Agrawal 94]. Also based on *precision* and *recall* the *statistical support* for mining sequential patterns on data streams is proposed in [Laur 05].

$$F_\beta = (1 + \beta^2)PR/(R + \beta^2 P)$$

where $P$ is the *precision*, *i.e.* $\frac{TP}{TP+FP}$[1], $R$ is the *recall*, *i.e.* $TP/(TP + FN)$, while $\beta$ is a parameter adjusted by the user. This measure is anti-monotone and is used in place

[1] $TP$ - true positive; $TN$ - true negative; $FP$ - false positive; $FN$ - false negative

of the *Support*. In [Giannella 04] the frequent-pattern mining framework was extended to mine time-sensitive patterns with approximate support guarantee in data streams. This framework offers several kinds of pruning strategies to mine a variety of frequent patterns associated with time. *Sequential Interestingness* measure [Sakurai 07] takes into consideration the supports of a pattern and its subpattern and has thus the anti-monotone property:

$$inst(s) = \min_{s_p \subseteq s} \left\{ \left( \frac{1}{\mathit{freq}(s_p)} \right)^{\alpha} \right\} \times \frac{(\mathit{freq}(s))^{1+\alpha}}{N}$$

where $\alpha \geq 0$ is a sequential interestingness parameter, $s_p$ is a sequential sub-pattern of a sequential pattern $s$, *freq*() is a function that calculates the frequency of a sequential pattern in sequential data, and $N$ is the number of sequential data.

*Weighted Interesting Sequential* pattern mining with *sequential s-confidence* and *w-confidence* measures, which support the apriori principle, are proposed in [Yun 07]. The anti-monotone *Weighted Support*, a *Support*-based measure, takes into consideration the weights (to be set by the user) of the itemsets to extract weighted sequential patterns [Yun 08]. In [Chang 11] this approach is extended with the *Time-interval Weighted Support* (TiW-support) measure in order to consider the time-intervals between itemsets:

$$\mathit{TiW\text{-}Supp}(X) = \frac{\sum_{S:(X \subseteq S) \bigwedge (S \in SDB)} W(S)}{\sum_{S:S \in SDB} W(S)}$$

where $W(S)$ is the *time-interval weight of a sequence*. *TiW-support* uses three kinds of time-interval weighted functions where weights could decrease in general with or without ceiling or in log scale. In addition it takes into consideration the number of items of a sequence (more items in the itemset having a higher influence on the measure's value,*i.e.* it might increase or decrease the final result (according to the time-intervals in the pattern) with a greater ratio than the smaller itemsets). This measure appears to fulfil the *a priori* principle, however, the author does not explain which distance to choose from a pattern which appears several times inside a sequence. For example the pattern $a\ b$ inside $a\ (bc)\ x\ a\ x\ b$, *i.e.* we are not told if we should choose the first $a\ b$ pair or the second one, because if we choose the other one, than the weight of the next pattern (in this case $a\ (bc)$) will be higher, thus, the anti-monotone principle will not be fulfilled.

## 2.3 Theoretical Fundamentals

We here present the basic concepts in time stamp sequential data mining. Definitions are exemplified with a toy sequential database (defined next) presented in Table 2.1.

Let's denote by $I = \{i_1, i_2, ..., i_k\}$ the set of all items. An *itemset* is a non-empty subset of *I*. A *sequence* is an ordered list of itemsets, denoted by $S = \{(s_1, t_1), (s_2, t_2), ..., (s_n, t_n)\}$, where $s_i$ is an itemset and $t_i$ its corresponding time stamp, $1 \leq i \leq n$. Obviously $t_i < t_{i+1}$ for $1 \leq i < n$. An item can occur at most once in an itemset of a sequence but an item can occur in several itemsets of a sequence. A *sequence database* is a set of sequences, denoted by $SD = \{S_1, S_2, ..., S_m\}$, each one being associated with an *ID*. A *sequential rule* $A \rightarrow B$ is a relationship between two itemsets $A$ and $B$ where $B$ occurs at a time stamp $t_B > t_A$, and where $A$ and $B$ belong to the same sequence. A *sequential pattern* $P_1 P_2 ... P_n$ is a relationship between the itemsets $P_i$ where $P_{i+1}$ occurs at a time stamp $t_{i+1} > t_i$, and where $P_i$ and $P_{i+1}$ belong to the same sequence.

Table 2.1 : Toy example DB.

| ID \ Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 10 | a | (de) | b | a | b | e | |
| 20 | (bc) | e | (ab) | e | b | c | c |
| 30 | c | d | e | b | a | | |
| 40 | (ab) | (cd) | b | | | | |
| 50 | c | (ade) | (ad) | a | e | c | |
| 60 | e | b | c | e | b | | |
| 70 | d | d | c | b | d | | |
| 80 | (abd) | c | b | | | | |
| 90 | a | b | (de) | a | c | e | |
| 100 | e | c | b | | | | |

Let's consider the set of items $I = \{a, b, c, d, e\}$ and the sequential database $SD$ in Table 2.1 which contains 10 sequences: $S_{10}$, $S_{20}$, $\ldots S_{100}$. We consider in this toy example only discrete time-stamps without loss of generality. The second sequence $S_{20} = \{(bc, 1),\ (e, 2),\ (ab, 3),\ (e, 4),\ (b, 5),\ (c, 6),\ (c, 7)\}$ is composed of seven itemsets while $S_{60} = \{(e, 1), (b, 2), (c, 3), (e, 4), (b, 5)\}$ contains five itemsets.

From $S_{20}$ we might form for example the following sequential rules: $b \to e$ (from $(bc, 1)$ with $(e, 2)$, or from $(ab, 3)$ with $(e, 4)$), $bc \to a$ (from $(bc, 1)$ with $(ab, 3)$), $e \to c$ (from $(e, 2)$ or $(e, 4)$ with $(c, 6)$ or $(c, 7)$), or the patterns $b\ e\ c$ (from $(b, 1)$, $(e, 2)$ and $(c, 6)$ or $(c, 7)$), $(ab)\ b$ (from $((ab), 3)$ and $(b, 5)$). In the same way, the rules $c \to d$, $c \to e$, $c \to b$ and $c \to a$ can be extracted from $S_{30}$. In our analysis we consider a rule and a pattern valid also if $t_{i+1} \geq t_i$, *e.g.* the rule $b \to c$ and the pattern $b\ c\ e$, formed from $(b, 1)$, $(c, 1)$, and $(e, 2)$ are good ones.

One of the issues of the existing measures presented in Section 2.2 is that in real applications events that appear after 2 time units might be as important as the ones that occur after 5 time units. And then further events may be of continuous decreasing interest to the user. For example, the rules *printer* $\xrightarrow{\textit{1 day}}$ *scanner* and *printer* $\xrightarrow{\textit{3 days}}$ *CD burner* might be equivalent to the user, *i.e.* having the same importance as the time difference between the antecedent and consequent is not so big from the user's point of view. While the rule *printer* $\xrightarrow{\textit{1 day}}$ *scanner* would be of much greater importance than the rule *printer* $\xrightarrow{\textit{10 days}}$ *CD burner*, because the time-difference in this case is not negligible to the user. One of the solutions to achieve this is using time-window sequential rules and patterns mining. But in order to differentiate between these time intervals the user would have to execute the extraction algorithms separately for a given time-interval. For example, one has to execute the algorithm by taking into consideration the rules happening inside a time-interval of 10 days to select and advantage the rules *printer* $\xrightarrow{\textit{1 day}}$ *scanner* and *printer* $\xrightarrow{\textit{3 days}}$ *CD burner*, and a different simulation with a different time-interval to select the rule *printer* $\xrightarrow{\textit{10 days}}$ *CD burner* and to give it less importance. If a more precise differentiation of the rules is needed, then more simulations have to be performed. These point of views, has not been considered previously and are the basis of our proposition.

Thus, the user may be interested in taking into account the time-difference between the consequent itemsets, *i.e.* the antecedent and the consequent of the rules and the time-difference between the appearance of 2 consequent itemsets in a pattern. For example, from this point of view the rules obtained from $S_{30}$ $c \to d$ ($c$ and $d$ are very close with respect to time) and $c \to a$ ($c$ and $a$ are further) are not equivalent. In the same way the pattern $c\ e\ a$ ($c$, $e$, and $a$ are very close with respect to time) and $(bc)\ e\ c$ ($e$ and $c$ are further) are not equivalent also. Indeed it could be desirable to have the rules/patterns ranked at the same level if the time-difference between the itemsets is no greater than a certain time, denoted by us with $\sigma_t$, and then, to decrease their importance with a certain speed w.r.t. to time by imposing a time-window $\omega_t$ where the value of the measure passes below the 50% of the maximum value. These considerations lead us to define the notion of closeness in time.

**Definition 1:** ***Time Closeness - sequential rules.*** Let $\omega_t$ be a time interval and $W$ a time window of size $\omega_t$. We say that the itemsets $A$ and $B$ with time-stamps $t_A$ and $t_B$ are $\omega_t$-close *iff* $|t_A - t_B| \leq \omega_t$. When considering a sequential rule $A \to B$, *i.e.* $t_B \geq t_A$, $A$ and $B$ are $\omega_t$-close *iff* $t_B - t_A \leq \omega_t$.

**Definition 2:** ***Closeness Measure - sequential rules.*** Let $\sigma_t$ be a user-preference time-interval, $\sigma_t < \omega_t$. We define a closeness measure for a $\omega_t$-close rule $A \to B$ as a decreasing function of $t_B - t_A$ and $1/\sigma_t$ such that if $t_B - t_A \leq \sigma_t$ then the measure should decrease slowly while if $t_B - t_A > \sigma_t$ then the measure should decrease rapidly.

In the following only $\omega_t$-close rules will be considered (Definition 1). The set of $\omega_t$-close rules will be ranked according to the closeness measure (Definition 2): the closer $B$ to $A$ the higher the measure's value of the $\omega_t$-close rule $A \to B$ is.

Let's take as an example the sequence $S_{90} = \{(a,1), (b,2), (de,3), (a,4), (c,5), (e,6)\}$ (Table 2.1). Some of the rules that could be formed from $S_{90}$ are: $a \to b$ (from $(a,1)$ with $(b,2)$), $a \to (de)$ (from $(a,1)$ with $(de,3)$), $a \to a$ (from $(a,1)$ and $(a,4)$), $a \to c$ (from $(a,1)$ with $(c,5)$), $a \to e$ (from $(a,1)$ with $(e,6)$). Let now set $\omega_t = 4$ and $\sigma_t = 2$. All the rules are 4-close rule except $a \to e$ (the time difference between the itemsets is 5). The rules $a \to b$ and $a \to (de)$ will have close measure's values (the time-difference between the itemsets is less or equal to $\sigma_t$), the value for $a \to b$ being slightly greater (the time-difference for this rule is smaller). The measure for $a \to (de)$ must be much greater than for $a \to a$ (the time-difference are smaller and greater respectively than $\sigma_t$).

Regarding patterns, we state the following Definition:

**Definition 3:** ***Time-Closeness Weight - sequential patterns.*** Let $\sigma_t$ and $\omega_t$ be two user-preference time-intervals, *s.t.* $\sigma_t < \omega_t$, and $\omega_t$ being the time after which the value of the weight passes below 50%. We define a time-closeness weight for a pattern $P_1P_2...P_n$ as a decreasing function of $t_{i+1} - t_i$ and $1/\sigma_t$ and $1/\omega_t$ such that if $t_{i+1} - t_i \leq \sigma_t$, then the weight should decrease slowly, while if $t_{i+1} - t_i > \sigma_t$ then the weight should decrease faster. The speed of the decreasing depends on the time-interval $\omega_t - \sigma_t$, *i.e.*: a higher value results in a slower decreasing, while a small value results in a faster decreasing of the time-closeness weight.

The set of obtained patterns will be ranked according to the time-closeness weight (Definition 3): the closer the itemsets, the higher the measure's value is.

From the same database in Table 2.1 , we consider $S_{20}$ where we might form the following sequential patterns from: $(bc)$ $e$ $c$ (from $(bc, 1)$, $(e, 2)$ or $(e, 4)$, and $(c, 6)$ or $(c, 7)$), $(ab)$ $b$ (from $(ab, 3)$ with $(b, 5)$), $(ab)$ $e$ (from $(ab, 3)$ with $(e, 4)$), $c$ $e$ $a$ (from $(c, 1)$, $(e, 2)$, and $(a, 3)$). If we take an $\omega_t = 2$ and a value for $\sigma_t = 1$, then the measure's value for the pattern $(bc)$ $e$ $c$ should be lower than for $c$ $e$ $a$ (the time difference between the itemsets of the first pattern is higher). The measure of the pattern $(ab)$ $e$ is higher than for the pattern $(ab)$ $b$ (the time difference of the itemsets from $(ab)$ $e$ is $\leq \sigma_t$, which is not the case for $(ab)$ $b$).

## 2.4 Closeness Preference Interestingness Measure for Sequential Rules Selection

In this Section we present the closeness IM for sequential rules mining, its properties, and how to use it in order to rank or compare the $\omega_t$-close rules. A toy example is presented followed by simulations on a real Web-log database used for prediction purposes. The results show that the proposed measure is able to advantage and select the rules with closer itemsets which give higher prediction performance than the existing sequential forecasting algorithms and measures.

### 2.4.1 Definition

Let's consider a sequential rule $A \to B$. We define an interestingness measure which will favour the rules with a relation of dependency between the itemsets $A$ and $B$ (requirement 1) where the consequent $B$ is as close as possible to the antecedent $A$ in most of the cases (requirement 2).

The first requirement implies to consider $R(A) \cdot R(B)$, where $R(X) = \frac{n_X}{|DB|}$, $n_X$ being the number of sequences where the itemset $X$ appears, and $|DB|$ is the number of sequences in the database $DB$. In order to penalize very frequent itemsets in the database the higher $R(A) \cdot R(B)$, the lower the closeness measure's value will be (following the properties proposed in [Piatetsky-Shapiro 91]).

The second requirement considers two parameters: the size of the (sliding) window $\omega_t$ (in order to consider only $\omega_t$-close rules – Definition 1) and the user-predefined time-interval $\sigma_t$ (Definition 2). It will measure the strength of the closeness of the itemsets $B$ to $A$ over all sequences of the database. We denote by $C_{\omega_t,\sigma_t}(B|A)$ this closeness.

As a whole the proposed *Closeness (User) Preference* interestingness measure $CP$ is defined as $CP(A \to B) = \frac{C_{\omega_t,\sigma_t}(B|A)}{R(A) \cdot R(B)}$.

Let's now define the strength of the closeness $C_{\omega_t,\sigma_t}(B|A)$ between $A$ and $B$.

First of all each $B$ following $A$ is weighted by a decreasing function $f(t, s, \omega_t)$ of its distance in time from $A$ inside the time-window $\omega_t$:

$$f(t, s, \omega_t) = \frac{1}{1 + s^{t-\omega_t}} \tag{2.1}$$

where $t = t_B - t_A$, $s$ is the slope of the plot ($s > 1$) and is directly proportional with the user-preference time-interval $\sigma_t$ (Definition 2.3). The greater $s$ implies that the rules with the consequent closer to the end of $\omega_t$ will be much more penalized than the others. As $s$ and $\omega_t$ are user-fixed parameters, we simply denote the function by $f(t)$. Figure 2.2 illustrates the behaviour of this function for $s = 3, 7, 25$ and $\omega_t = 5$.

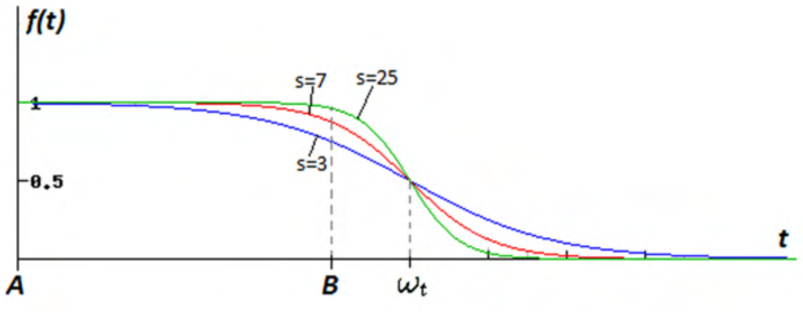


Figure 2.2 : Weight distance function according to $\omega_t$ and slope $s$.

For a given value for $\sigma_t$ and the minimum value $f(\sigma_t)$ of $f(t)$ in the interval $[0, \sigma_t]$, the slope $s$ is given by:

$$s = \sqrt[\sigma_t - \omega_t]{\frac{1}{f(\sigma_t)} - 1} \tag{2.2}$$

The strength of the time-relation between a single antecedent $A$ and one or several consequents $B$ inside the same time-window of width $\omega_t$, is the average value of all the $f(t)$ function's values:

$$CP_{B|A} = \frac{1}{n_{t_{B|A}}} \sum_{j=1}^{n_{t_{B|A}}} \frac{1}{1 + s^{t_{B_j} - \omega_t}} \tag{2.3}$$

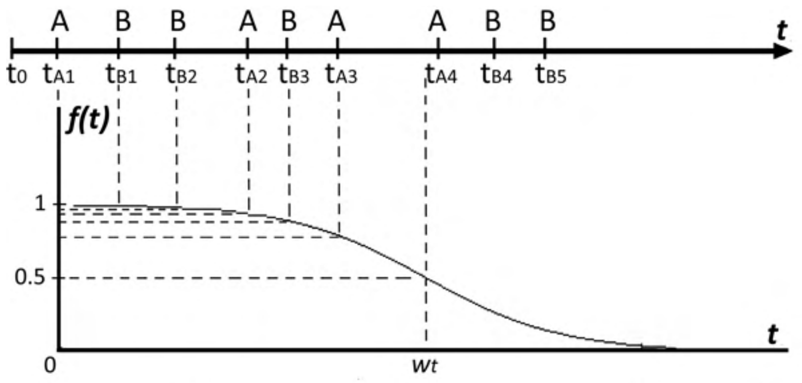
where $n_{t_{B|A}}$ is the number of $B$ between 2 consecutive $A$ inside the same window and where $t_{B_j}$ is the time distance of each $B$ from the beginning of the window (starting from the first $A$). If there is only one itemset $A$ in the time-interval $[0, \omega_t]$, then $n_{t_{B|A}}$ is the number of $B$ inside the window. Next, this calculation is done for each $A$ inside $\omega_t$ (see Figure 2.3 for an illustration):

$$CP_\omega = \frac{1}{n_{t_A}} \sum_{i=1}^{n_{t_A}} CP_{B|A_i} \tag{2.4}$$

where $n_{t_A}$ is the number of $A$ inside $[0, \omega_t]$, and $CP_{B|A_i}$ is the expression from Equation (2.3) for each $A_i$. Note that each $A$ not followed (w.r.t $\omega_t$) by a $B$ will penalize the rule's evaluation.

We then extend this approach to all time-windows of size $\omega_t$ of a sequence. The strength of $A \to B$ in a sequence is the average value of all windows in it:

$$CP_S = \frac{1}{n_{\omega_t}} \sum_{k=1}^{n_{\omega_t}} CP_{\omega_k} \tag{2.5}$$



$$CP_\omega = \frac{1}{3}\left(\frac{1}{2}\left(\frac{1}{1+s^{(t_{B1}-t_{A1})-\omega_t}} + \frac{1}{1+s^{(t_{B2}-t_{A1})-\omega_t}}\right) + \frac{1}{1}\left(\frac{1}{1+s^{(t_{B3}-t_{A1})-\omega_t}}\right) + 0\right)$$

Figure 2.3 : An example of the $CP_\omega$ calculation.

where $n_{\omega_t}$ is the number of time-windows of size $\omega_t$ in a single sequence containing the rule $A \to B$, and $CP_{\omega_k}$ is defined as in Equation (2.21) for each window $k$. This calculation is illustrated in Figure 2.4.

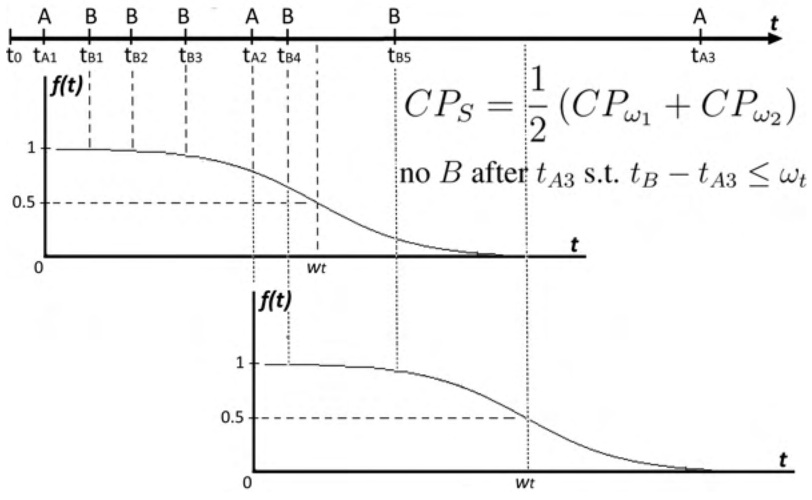

A B B B A B B A
t
t0 tA1 tB1 tB2 tB3 tA2 tB4 tB5 tA3
f(t)
1
0.5
0
wt
t
$CP_S = \frac{1}{2}(CP_{\omega_1} + CP_{\omega_2})$
no $B$ after $t_{A3}$ s.t. $t_B - t_{A3} \leq \omega_t$
f(t)
1
0.5
0
wt
t

Figure 2.4 : An example of the $CP_S$ calculation.

Last the average strength of the rule $A \to B$ is calculated from all the sequences containing $A \to B$ to the entire database and the closeness index $C_{\omega_t,\sigma_t}(B|A)$ is defined by:

$$C_{\omega_t,\sigma_t}(B|A) = \frac{1}{|DB|}\sum_{m=1}^{n_{AB_\omega}} CP_{S_m} \tag{2.6}$$

where $n_{AB_\omega}$ is the total number of sequences where the rule $A \to B$ holds at least once in the interval $[0, \omega_t]$; and $CP_{S_m}$ is the expression from the Equation (2.5) for each sequence $m$.

Finally the *Closeness (User) Preference* interestingness measure $CP$ is defined as the following:

$$CP(A \to B) = \frac{\frac{1}{|DB|}\sum_{m=1}^{n_{AB_\omega}}\left[\frac{1}{n_{\omega_t|m}}\sum_{k=1}^{n_{\omega_t|m}}\left[\frac{1}{n_{t_A|k}}\sum_{i=1}^{n_{t_A|k}}\frac{1}{n_{t_{B|A_i}}}\sum_{j=1}^{n_{t_{B|A_i}}}\frac{1}{1+s^{t_j-\omega_t}}\right]\right]}{R(A)\cdot R(B)} \tag{2.7}$$

The goal of the proposed measure is to select the "strong" rules with respect to the probability of the antecedent's and consequent's appearances in the database, and with respect to the temporal proximity between the itemsets of the rule. The numerator of the Equation (2.7), $C_{\omega_t,\sigma_t}(B|A)$, is actually the frequency of an $\omega_t$-close rule $A \to B$ taking into account the temporal proximity ($C_{\omega_t,\sigma_t}(B|A) < R(AB)$), where $R(AB) = \frac{n_{AB}}{|DB|}$, $n_{AB}$ being the number of sequences where the rule $A \to B$ is sustained. It results that the $CP$ measure has some similarities with the *Lift* measure (*i.e.* $\frac{P(AB)}{P(A)\cdot P(B)}$, where $P(X)$ is the probability of the itemset $X$ in the database).

The $CP$ measure can thus be used in order to rank the rules. In addition if one fixes a threshold it can be used to select the rules. Such a threshold $\theta^*$ could be the average value of the weighting function (Equation (2.1)) over $\omega_t$. It is the area delimited by the curve of the function and the x-axis divided by $\omega_t$, *i.e.* its integral. In order to calculate this integral we denote: $t_j$ by $x$; $s^{-\omega_t}$ by a constant $c$; $s^{t_j}$ by $a^x$. Results that the total area of the function from Figure 2.2 is:

$$\begin{aligned} \int_0^{\omega_t}\frac{1}{1+ca^x}\,dx &= \int_0^{\omega_t}\frac{1+ca^x-ca^x}{1+ca^x}\,dx = \\ = \int_0^{\omega_t}\left[1-\frac{ca^x}{ca^x+1}\right]dx &= \int_0^{\omega_t}\left[1-\frac{ca^x\ln(a)}{(ca^x+1)\ln(a)}\right]dx = \\ &= \int_0^{\omega_t}\left[1-\frac{1}{\ln(a)}\cdot\frac{ca^x\ln(a)}{ca^x+1}\right]dx \end{aligned} \tag{2.8}$$

As we know:

$$(u^x)' = u^x\ln(u) \tag{2.9}$$

$$\frac{(u(x))'}{u(x)} = (\ln(u(x)))' \tag{2.10}$$

So, it can be written:

$$(ca^x+1)' = ca^x\ln(a) \tag{2.11}$$

By substituting Equations (2.10) and (2.11) in (2.8), we obtain:

$$\begin{aligned}\int_0^{\omega_t}\left[1-\frac{1}{\ln(a)}\cdot\frac{(ca^x+1)'}{ca^x+1}\right]dx &= \int_0^{\omega_t}\left[1-\frac{1}{\ln(a)}\cdot\ln(ca^x+1)\right]dx = \\ =\int_0^{\omega_t}x'\,dx-\frac{1}{\ln(a)}\int_0^{\omega_t}(\ln(ca^x+1))'\,dx &= \left[x-\frac{\ln(ca^x+1)}{\ln(a)}\right]\Bigg|_0^{\omega_t} = \\ =\omega_t+\frac{\ln(c+1)-\ln(ca^{\omega_t}+1)}{\ln(a)} &= \omega_t+\frac{\ln\left(\frac{c+1}{ca^{\omega_t}+1}\right)}{\ln(a)}\end{aligned} \tag{2.12}$$

By resubstituting the values of $a$ and $c$, we obtain the final propose threshold value of the $CP$:

$$\theta^* = \frac{1}{\omega_t}\int_0^{\omega_t}\frac{1}{1+s^{t-\omega_t}}\,dx = \frac{\omega_t+\frac{\ln\left(\frac{s^{-\omega_t}+1}{2}\right)}{\ln(s)}}{\omega_t} \tag{2.13}$$

This average value will select the rules with very close itemsets. Of course depending of user's goal another threshold can be used *(by setting the threshold higher or lower)*.

## 2.4.2 CP Properties

We now discuss some properties of the $CP$ measure. Many studies on a large number of interestingness measures, especially for association rules, and on their properties has attracted significant attention in the literature (*e.g.* [Tan 04, Geng 06, Lenca 08]). [Piatetsky-Shapiro 91] proposed three properties that must be satisfied by any reasonable interestingness measure: a measure should value 0 if $A$ and $B$ are statistically independent *i.e.* when $P(AB) = P(A)\cdot P(B)$ [property $P_1$]; a measure should monotonically increase with $P(AB)$ when $P(A)$ and $P(B)$ remain the same [property $P_2$]; a measure should monotonically decrease with $P(A)$ (or $P(B)$) when $P(AB)$ and $P(B)$ (or $P(A)$) remain the same [property $P_3$]. [2]

We focus on $P_2$ and $P_3$ properties. Indeed, $CP$ measure is not concerned by the property $P_1$ as it was not designed for. However, as $CP$ is a measure to be used in a post-analysis phase to select the rules one can first filter the rules with the help of a measure that fulfils property $P_1$. This is the case of the *Lift* measure which values 1 at the independence (property $P_1$ can be extended to any constant value [Lenca 08]).

Regarding the property $P_2$: if $R(AB)$ increases then the number $n_{AB_\omega}$ of sequences containing $\omega_t$-close rules might only increase (in the worst case it remains the same). As $CP$ sums only positive numbers the numerator of Equation (2.7) can only increase while the denominator is constant. $CP$ has the property $P_2$.

A similar reasoning is done to show that $CP$ has the property $P_3$. If $R(AB)$ does not change then no new $\omega_t$-close rules can appear and the numerator of Equation (2.7) is constant while the denominator will increase with $R(A)$. Thus $CP$ decreases. The same reasoning is done for $R(B)$.

[2] Note that here we do not consider $P(X)$ but instead $R(X)$ as defined in Subsection 2.4.1.

### 2.4.3 Toy Example

Let's consider the toy example database of Table 2.1. In order to extract the rules we use the post-processing from the *Generalized Sequential Pattern* algorithm [Srikant 96]. It was selected because of its simplicity and its property to extract all the existing sequential patterns that pass a given *Support* threshold. The algorithm passes the database several times. At each pass there is a *Join Phase*, and *Prune Phase*. For the first phase the set $L_{k-1}$ of all frequent $k-1$ sequences is joined with itself in order to generate a superset of the set of all frequent $k$-sequences (where $k$ is the number of items). For the *Prune Phase* the candidate sequences whose *Support* count is less than the minimum *Support* are deleted. A *hash-tree* data structure is used for an efficient candidate counting. Using *GSP* we extract all the possible sequential patterns that have 2 itemsets (note, that there might be several items in the antecedent or consequent itemset) and analyse them as rules. With a *Support* threshold of 50% the *GSP* algorithm extracts 15 sequential rules.

These rules are sorted with respect to their *Lift* ($Lift(A \to B) = \frac{R(AB)}{R(A) \cdot R(B)}$), *Confidence* ($conf(A \to B) = \frac{R(AB)}{R(A)}$) and $CP$ values. Note, that we adapt the formulas for *Support*, *Confidence*, and *Lift* for a sequential rule $A \to B$, *i.e.*:

$$supp = \frac{R(AB)}{|DB|}; \qquad conf = \frac{R(AB)}{R(A)}; \qquad Lift = \frac{R(AB)}{R(A) \cdot R(B)} \tag{2.14}$$

For $CP$ we select $\omega_t = 6$, $\sigma_t = 4$, and $f(\sigma_t) = 0.8$. According to Equation (2.19), it results that we obtain the following value for the slope of the function $f(t)$: $s = \sqrt[4-6]{\frac{1}{0.8} - 1} = 2$. Thus, the $CP$ measure is parametrized with $\omega_t = 6$ and $s = 2$ (which lead to the function presented in Figure 2.5 and a medium value $\theta^*$ of 0.837). The results are shown in Table 2.2.

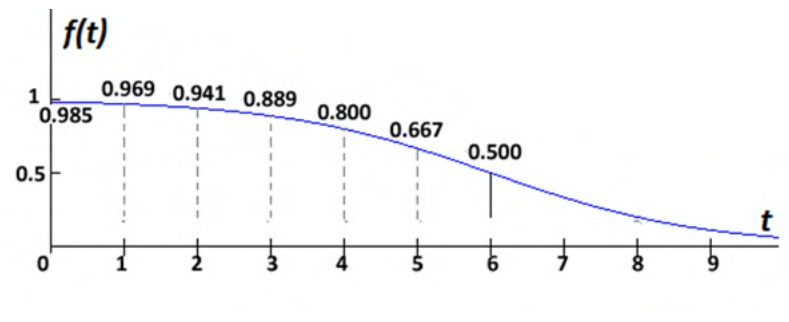
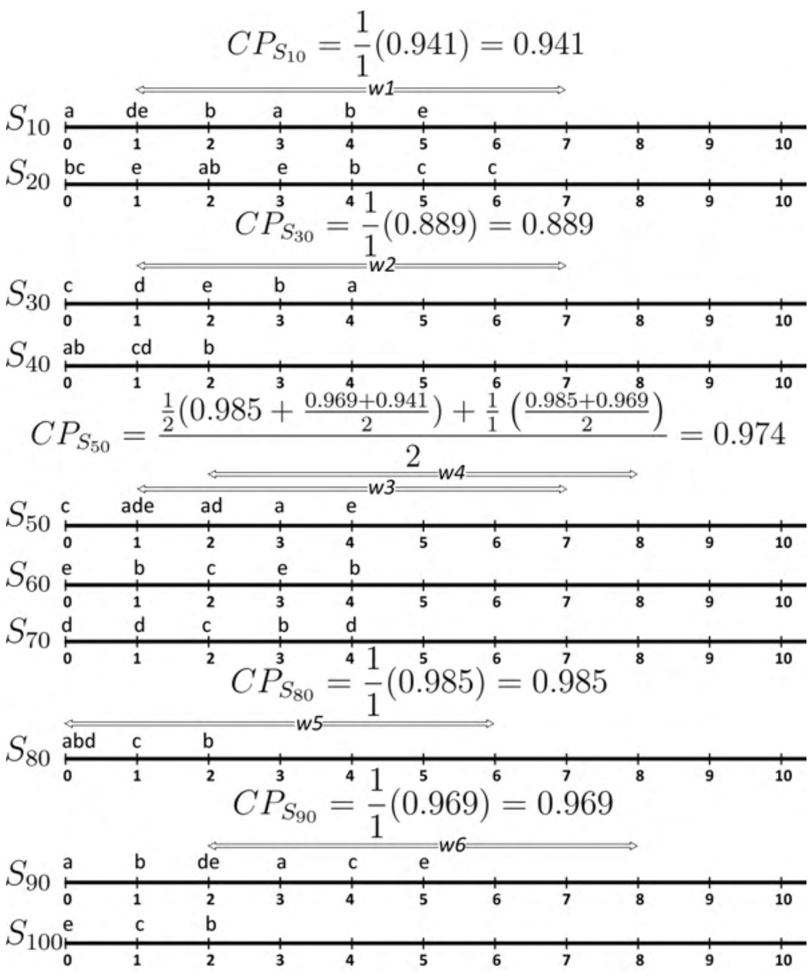


Figure 2.5 : The function $f(t)$ with $s = 2$ and $\omega_t = 6$.

Let's illustrate the case of the rule $d \to a$. The contribution $CP_{S_i}$ for each sequence $S_i$ containing $d \to a$ (Equation (2.5)) is illustrated in Figure 2.6. These values are then averaged (Equation (2.6)) over the sequences containing $d \to a$ (the items $a$ and $d$ appear in 7 sequences out of 10):

$$CP(d \to a) = \frac{\frac{1}{10}(0.941+0.889+0.974+0.985+0.969)}{\frac{7}{10} \cdot \frac{7}{10}} = 0.97$$



$CP_{S_{10}} = \frac{1}{1}(0.941) = 0.941$
w1
$S_{10}$: a (0), de (1), b (2), a (3), b (4), e (5)
$S_{20}$: bc (0), e (1), ab (2), e (3), b (4), c (5), c (6)
$CP_{S_{30}} = \frac{1}{1}(0.889) = 0.889$
w2
$S_{30}$: c (0), d (1), e (2), b (3), a (4)
$S_{40}$: ab (0), cd (1), b (2)
$CP_{S_{50}} = \frac{\frac{1}{2}(0.985 + \frac{0.969+0.941}{2}) + \frac{1}{1}\left(\frac{0.985+0.969}{2}\right)}{2} = 0.974$
w4
w3
$S_{50}$: c (0), ade (1), ad (2), a (3), e (4)
$S_{60}$: e (0), b (1), c (2), e (3), b (4)
$S_{70}$: d (0), d (1), c (2), b (3), d (4)
$CP_{S_{80}} = \frac{1}{1}(0.985) = 0.985$
w5
$S_{80}$: abd (0), c (1), b (2)
$CP_{S_{90}} = \frac{1}{1}(0.969) = 0.969$
w6
$S_{90}$: a (0), b (1), de (2), a (3), c (4), e (5)
$S_{100}$: e (0), c (1), b (2)
0 1 2 3 4 5 6 7 8 9 10

Figure 2.6 : $CP_\omega$ and $CP_S$ calculation for the rule $d \rightarrow a$.

Let's consider some rules with the highest *Lift* in Table 2.2, for example rules 1 and 4. Following our goals the $CP$ measure is not in favour of $e \rightarrow a$ but is in favour of $d \rightarrow a$. Indeed the rule $e \rightarrow a$ (respectively $d \rightarrow a$) is supported five times (respectively six times) but is not supported seven times (respectively four times). In addition the item $a$ occurs in average further from item $e$ than from item $d$. Here $CP$ disagrees with *Lift* as it was expected in order to advantage the rules with closer itemsets. Nevertheless they are concordant when rules can be considered as *bad* rules[3].

### 2.4.4 Closeness Preference Measure in Real Application

In this section we experimentally study the ability of the $CP$ measure to select *good* prediction rules with close itemsets. The results show in particular that few and simple rules with high $CP$ evaluation may produce an accurate rules set. For that purpose we

[3] *Kendall's Rank Correlation* coefficient [Kendall 38] on 170 rules extracted from *ClickStream* database (presented in the next section) is 0.68.

Table 2.2 : Sequential rules of the toy database with their *CP* and *Lift* values.

| Rule ID | Rule | CP | Lift | Confidence |
|---|---|---|---|---|
| 1 | d → a | 0.970 | 1.020 | 0.714 |
| 2 | e → e | 0.900 | 1.020 | 0.714 |
| 3 | a → d | 0.677 | 1.020 | 0.714 |
| 4 | e → a | 0.584 | 1.020 | 0.714 |
| 5 | b → a | 0.601 | 0.952 | 0.667 |
| 6 | c → b | 0.742 | 0.864 | 0.778 |
| 7 | a → b | 0.682 | 0.794 | 0.714 |
| 8 | e → b | 0.675 | 0.794 | 0.714 |
| 9 | d → b | 0.665 | 0.794 | 0.714 |
| 10 | a → c | 0.638 | 0.794 | 0.714 |
| 11 | d → c | 0.625 | 0.794 | 0.714 |
| 12 | c → e | 0.577 | 0.794 | 0.556 |
| 13 | e → c | 0.514 | 0.794 | 0.714 |
| 14 | b → b | 0.570 | 0.617 | 0.556 |
| 15 | b → c | 0.368 | 0.617 | 0.556 |

use the *ClickStream* data provided for the ECML/PKDD 2005 data mining challenge[4]. Indeed in such data one can be interested in predicting the next page that will be visited by a user based on a history of visited pages [Labsky 05]. Rules with close itemsets make here sense and thus $CP$ should be well-adapted.

#### 2.4.4.1 Purpose

As a forecasting problem we build a *simple* prediction model used to forecast the users' future requests in the *ClickStream* database, i.e. the visited page $V_n$ which appears after a visited page $V_i$ in order to provide reasonable recommendations to meet the user's requirements. So, we are searching for the rules of the form $V_i \rightarrow V_n$. In order to evaluate the performance of the extracted rules, a classical *Training - Testing* phase methodology is applied. Thus, the database is splitted into a *training* part and into a *testing* part. From the training part we extract the rules using the *GSP* algorithm with a *Support* pruning, followed by a filtering with $CP$, *Confidence*, or *Lift*. On the testing part we verify the forecasting performance of the obtained sequential rules making a comparison between $CP$, *Confidence* and *Lift* (Subsection 2.4.4.4), and between the proposed forecasting model with existing ones (Subsection 2.4.4.5) where we also compare the simple rules with the prediction performance of the complex rules of the form $V_1V_2V_3...V_{n-1} \rightarrow V_n$. The order of the $V_1, V_2, ..., V_{n-1}$ is not important, while $V_n$ should be inside the same time window $\omega_t$ proceeding $V_1, V_2, ..., V_{n-1}$.

The comparison with other existing models is done with the results from [Labsky 05] and [Liu 07]. Labsky et al [Labsky 05] use the same database and are motivated by two

[4] `http://lisp.vse.cz/challenge/CURRENT/index2.html`

goals: to predict the next visited page $V_n$ after observing a sequence of pages $V_1V_2...V_{n-1}$, and to find interesting patterns in the visited page sequences. We are focusing on his first objective, as it is similar to ours. Three algorithms are used [Labsky 05]: one statistical – *Markov n-gram* models, and two rule-based – *set covering* and *compositional* algorithms. Only *Accuracy* is used as a prediction performance measure and the best result, an accuracy of 0.67, was obtained using the *Markov N-gram* model. Liu and Keselj [Liu 07] propose an approach for user navigation patterns classification and user's future requests prediction. They use a database from a server log access file at Dalhousie University, similar to *ClickStream* Database[5]. Their model is based on combined mining of the log data and the contents of the corresponding web pages, which are captured through extraction of character *N-grams*. For the prediction process the authors do not split the entire database in Training/Testing but they split each session in two parts. One for analyzing (training) in order to predict the second part. The best result in prediction was obtained using *4 N-gram* size model, with an *Accuracy* of 0.644.

We evaluated the performance of our approach using *prediction accuracy* measures, such as *Accuracy*, *Precision*, *Recall*, and *F1-Score*.

#### 2.4.4.2 ClickStream database

The *ClickStream* database contains about 3.6 millions records (24 days) from a www shop web server log. Each record contains the time, IP address, session ID, page request URL, and referee. Most of the sessions are very short (with an average of 16 pages).

For our purpose only the *time*, the *session ID*, and the *visited page* are relevant. Data are then grouped into *sessions* (Figure 2.7). A *session* was described as a set of visited pages with the same session ID from the moment a user entered the site until the moment he left it. There are 522 410 sessions. Only 203 887 of them are of length greater than 1 (*i.e.* they contain more than 1 visited page). More basic analysis of the *ClickStream* database is provided in [Naito 05, Labsky 05].

The database is splitted into a *training* set (first 100K sessions) and *testing* set (next 60K) in order to make our result compatible to those from [Labsky 05].

#### 2.4.4.3 Rules extraction and validation algorithm

For the sequential rules extraction we have implemented the *GSP* algorithm (described briefly in Subsection 2.4.3). We used a *Support* threshold value of 1.5%. This threshold leads to several interesting profiles as shown in [Hofgesang 05]. We then obtain 170 rules from the *training* set. Each of these rules are then evaluated with the $CP$ measure. The time windows size $\omega_t$ used in the $CP$ measure was set to 360 and 420 seconds. Indeed these times are the most frequent time-interval a user spend on the shop site [Naito 05]. Last we study two user scenarios to set the slope $s$ of the $f(t)$ function (Equation (2.1)): one where the importance of the next visited page decreases rapidly ($s = 1.05$) and an another one where it decreases slower ($s = 1.25$) (Figure 2.8). The efficiency of the set of rules (or of any subset of rules) can then be studied on the *testing* set. We used the

[5]The [Liu 07] didn't provide us their DB for experiments and comparison

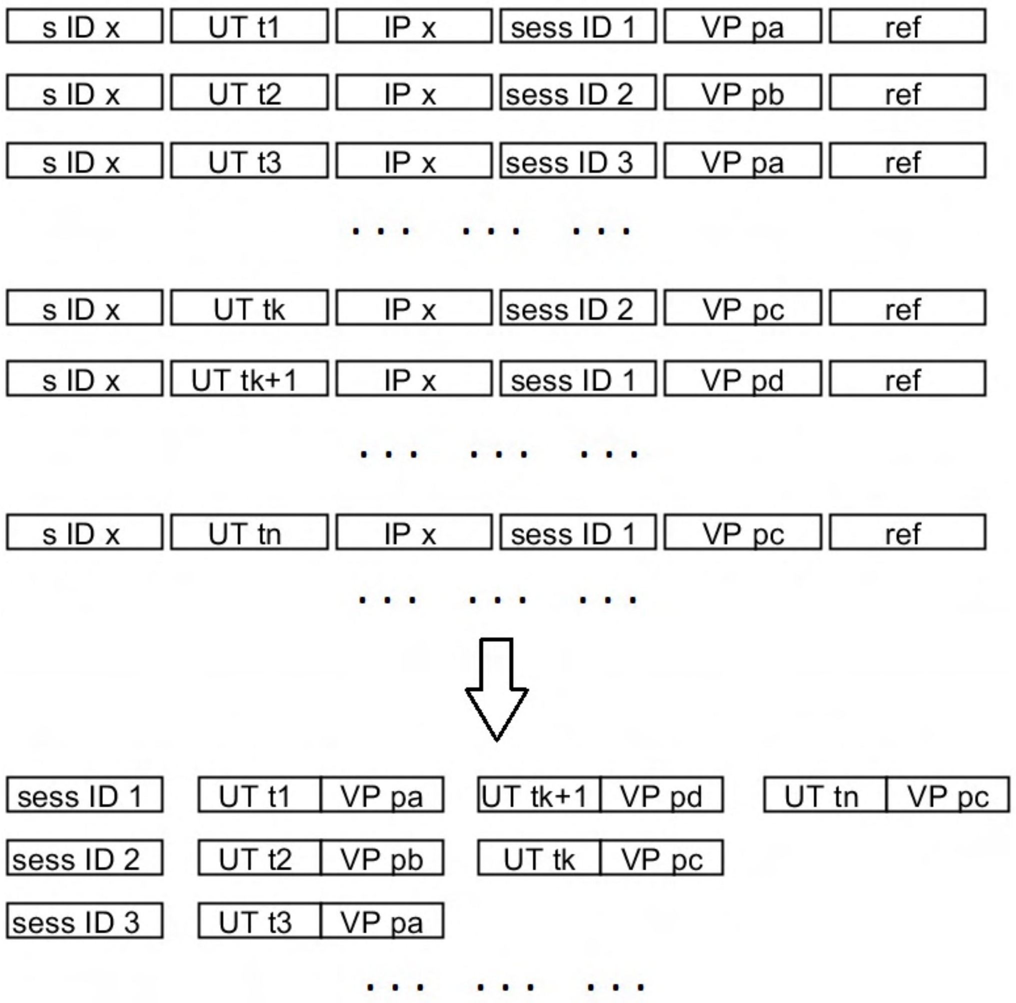

s ID x | UT t1 | IP x | sess ID 1 | VP pa | ref
s ID x | UT t2 | IP x | sess ID 2 | VP pb | ref
s ID x | UT t3 | IP x | sess ID 3 | VP pa | ref
s ID x | UT tk | IP x | sess ID 2 | VP pc | ref
s ID x | UT tk+1 | IP x | sess ID 1 | VP pd | ref
s ID x | UT tn | IP x | sess ID 1 | VP pc | ref
sess ID 1 | UT t1 | VP pa | UT tk+1 | VP pd | UT tn | VP pc
sess ID 2 | UT t2 | VP pb | UT tk | VP pc
sess ID 3 | UT t3 | VP pa

Figure 2.7 : Data transformation of the *ClickStream* database. Each row of the initial log file contains: shop ID *s ID*; unixtime *UT t*; IP address *IP*; session *sess ID*; visited page *VP*; referrer *ref*. From these, we group the rows into sessions, thus, a session will contain several visited pages and the corresponding time. The notation with $x$ presents different shop IDs or IP addresses (that of course might be the same, but these are not taken into consideration by us).

classical measures used in supervised learning *i.e. accuracy* (A), *precision* (P), *recall* (R) and *F1-Score* (F1).

From the obtained rules we use those which were selected by each of these measure and pass a certain threshold value. The *GSP* algorithm will extract the rules of the form $V_i \rightarrow V_n$. However, we would like also to know if we could increase the performance of the prediction by computing the so called *complex rules* of the form $V_1V_2...V_{n-1} \rightarrow V_n$. For this, we take all the rules of the simple form $V_i \rightarrow V_n$, after which we made combinations

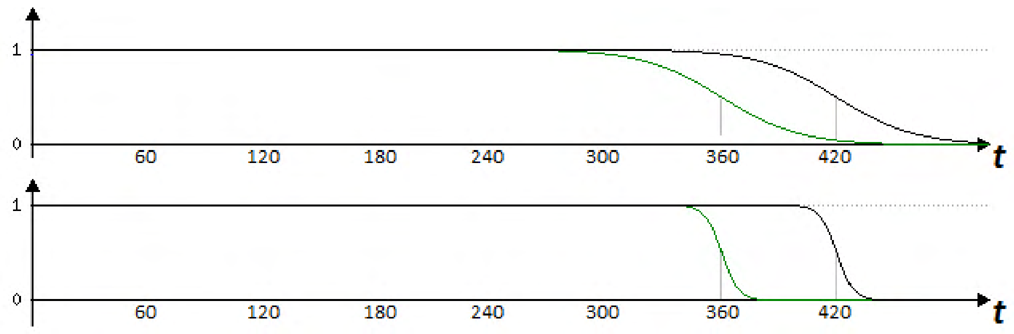


Figure 2.8 : The functions to calculate the proposed IM for the ClickStream dataset: top (from left to right): $slope = 1,05$, $\omega_t = 360, 420$; bottom (from left to right): $slope = 1,25$, $\omega_t = 360, 420$.

between all $V_i$ itemsets. For example, having the rules $A \rightarrow Y$, $B \rightarrow Y$, and $C \rightarrow Y$, we get the *complex rules* $AB \rightarrow Y$, $AC \rightarrow Y$, $BC \rightarrow Y$, $ABC \rightarrow Y$. All these *simple* and *complex rules* are applied on the test database. In the testing phase, the rules are considered to be valid if all of its itemsets are occuring in the time-window of length $\omega_t$.

As in [Labsky 05, Liu 07], we evaluated our results using the *Accuracy* (Equation (2.15)) measure:

$$Accuracy = \frac{TP + TN}{TN + FN + FP + TP} \tag{2.15}$$

where *TN* is *True Negative*, *FN* is *False Negative*, *FP* is *False Positive*, and *TP* is *True Positive*

We compute also other three popular performance indicators used in data mining [Goutte 05, Feng 07, Devitt 05]: *Precision (Prec), Recall (Rec) and F1-Score (F1).*

$$Prec = \frac{TP}{FP + TP}, \quad Rec = \frac{TP}{FN + TP}, \quad F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \tag{2.16}$$

#### 2.4.4.4 $CP$ vs *Lift* and *Confidence* in rule ranking and forecasting

In order to see the differences between the proposed measure with *Confidence* and *Lift* we have computed the *Kendall's $\tau$ Rank Correlation* coefficient [Kendall 38] based on the measures' results from the 170 extracted rules:

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)} \tag{2.17}$$

where $C$ is the number of *concordant* pairs, and $D$ the number of *discordant* pairs, while $\frac{1}{2}n(n-1)$ is the total number of pairs. The coefficient takes values in the interval $[-1; 1]$.

There are low *Kendall's $\tau$* correlation values between $CP$ and the *Confidence* (see Table 2.3) showing the properties' and the rule ranking differences between these two measures. These differences are also shown in Figure 2.9 from the left, where the graph of the rules' *Confidence* versus $CP$ are presented. The mathematical similarity between $CP$

(see Equation (2.7)) and *Lift* (see Equation (2.14)) induce higher *Kendall's* $\tau$ correlation values (see Table 2.3). Nevertheless, the differences between the rules' ranking given by $CP$ and *Lift* measures are showed by the corresponding plot from Figure 2.9 from the right, where the rules' values of the *Lift* and $CP$ are shown.

Table 2.3 : *Kendall's* $\tau$ *Correlation* coefficients between $CP$ and *Confidence* with *Lift* computed on the *ClickStream* database.

| | $\omega_t = 360s$ $s = 1.05$ | $\omega_t = 360s$ $s = 1.25$ | $\omega_t = 420s$ $s = 1.05$ | $\omega_t = 420s$ $s = 1.25$ |
|---|---|---|---|---|
| Confidence | 0.408 | 0.404 | 0.399 | 0.394 |
| Lift | 0.674 | 0.689 | 0.701 | 0.706 |

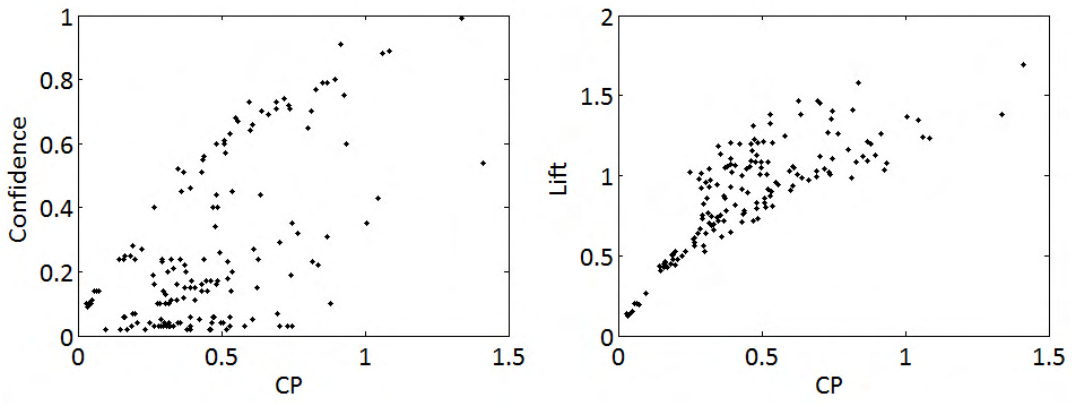


Figure 2.9 : The values for *Confidence*, *Lift*, and $CP$. $CP$ parameters were: $\omega_t = 360$ *slope* $= 1,25$. From 170 rules only 158 are shown because of the very high values of the $CP$ of the remaining rules resulting in a graphic concentration at the origins.

The differences are observed from the histograms of these values. Let's take a closer look to the rules generated from Figure 2.10. In case of the $CP$ measure we have the width of the time window $\omega_t = 360, 420$. We should also point out that the rules found in $\omega_t = 360$ would be found in the rules from $\omega_t = 420$ also. *Confidence* and *Lift* histograms are obviously the same, because these measures do not take into consideration the time-distance between events. In the histograms from our *IM*, the medium *IM* values increase (from aproximately $0,4305$ to $0,6045$ in the next histograms). This fact proves that the new *IM* is able to distinguish and advantage the rules which have the events closer one to another relative to the $\omega_t$ value. Also, by increasing the slope coefficient in our *IM* calculation from $1,05$ to $1,25$, we would be able to give similar importance to the rules which have the consequences inside the same window, because in this case the function in the *IM* calculation would decrease later with a faster rate.

The differences between the *Kendall's* $\tau$ *Correlation* coefficients and the plots from Figures 2.9 and 2.10, allow us to affirm that our $CP$ measure shows different results and

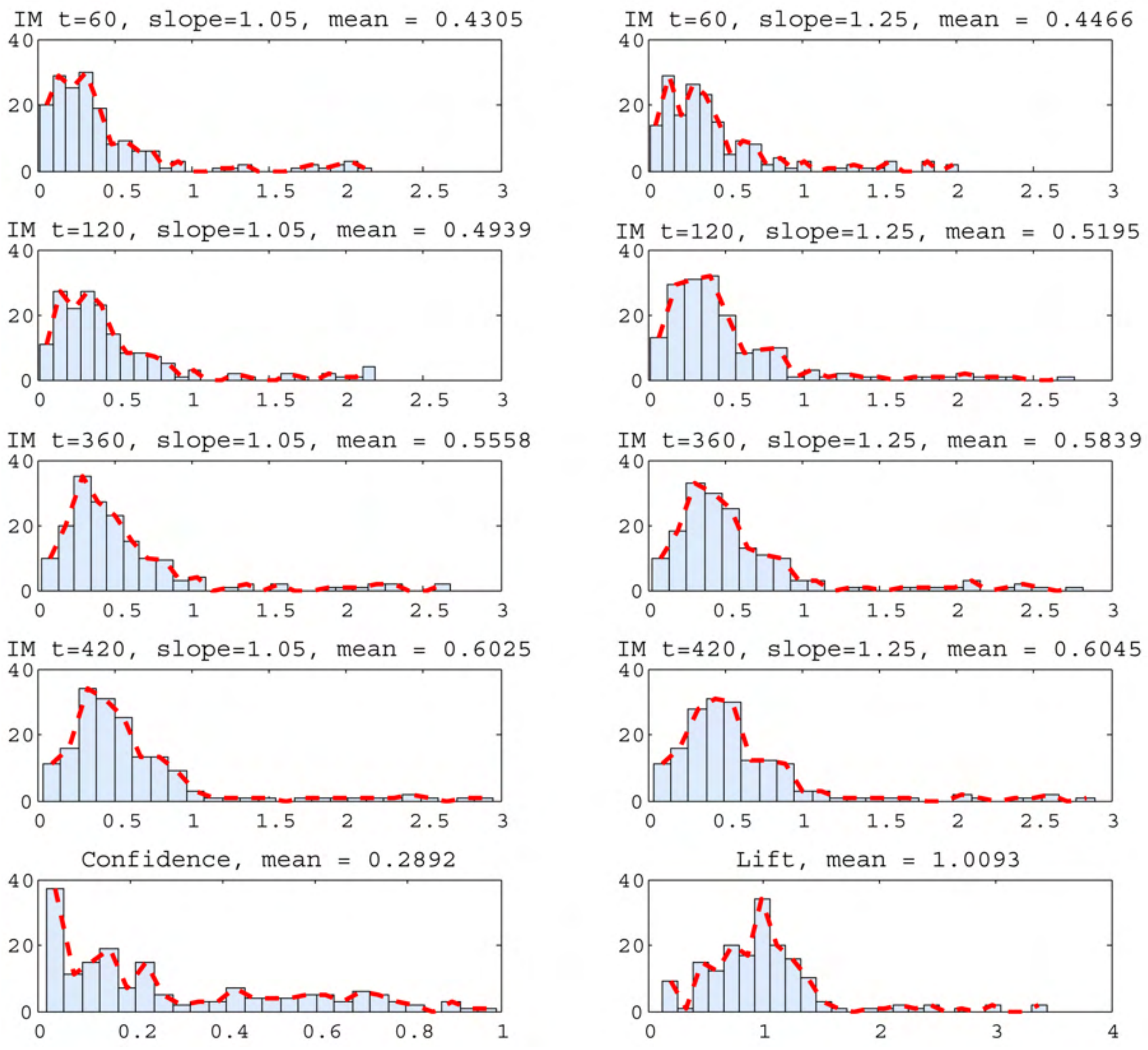


Figure 2.10 : The histograms of the ClickStream dataset with $s = 1.5\%$, $\omega_t = 360, 420$, and *slope* $= 1,05; 1,25$. Notice the increasing of the rules' values in the proposed IM while increasing the window's width and slope.

properties compared with *Confidence* and *Lift*. This difference is much higher because $CP$ has a temporal aspect. Also, if we increase the $\omega_t$ value much more, than it should be supposed that the similarity between $CP$ and *Lift* should be higher. however, according to our experiments, if we increased $\omega_t$ the $\tau$ value didn't improve significantly. For example, if we take an $\omega_t = 900$ seconds, then we obtain a value for $\tau$ between $CP$ and *Lift* around 0.670, while for $\omega_t = 1800$, we get $\tau = 0.650$. There is no big difference in the graphs of the $CP$ versus *Lift* also.

Earlier we have showed that there are differences between $CP$ measure and the *Confidence* with the *Lift*. Now, we want to know if these differences of the $CP$ imply better prediction performance. We present the results by using an $\omega_t = 360$ and a *slope* $s = 1.25$ for the $CP$ parameters. The pruning is done with a $CP$ threshold of 0.8 and we obtain 31 simple rules. In order to obtain the same number of rules, we must set a threshod of 0.6 for the *Confidence* and of 1.240 for the *Lift* respectively.

Table 2.4 : *Accuracy* (A), *Precision* (P), *Recall* (R), and *F1-Score*(F1) for $\omega_t = 360$. The $CP$'s *slope* $s = 1.25$. The number of *simple rules* tested was 31 for each measure.

| *Measure* | *CP* | *Conf* | *Lift* |
|---|---|---|---|
| Thr values | *0.8* | *0.6* | *1.240* |
| A | 0.757 | 0.508 | 0.709 |
| P | 0.785 | 0.809 | 0.571 |
| R | 0.676 | 0.349 | 0.785 |
| F1 | 0.662 | 0.416 | 0.565 |

The obtained results (see Table 2.4) show that $CP$ is better than *Confidence* in *Accuracy*, *Recall*, and *F1-Score* performance criterias, while is worse in *Precision* (i.e. $CP$ gives more *False Positives*). If we look at the formulas of the prediction performances measures, we can deduce that in comparison to *Confidence* the $CP$ measure has higher values for the ratio $FP/TP$, while lower values for the ratio $FN/TP$. This is because of the time-window, and of the fact that *Confidence* takes into consideration only $R(AB)$ and $R(A)$ parameters (resulting in lower $FP$ for the *Confidence*), while $CP$ considers $R(B)$ also.

In comparison to *Lift*, the $CP$ performance measures are higher with: 6.7% for the *Accuracy*, 37% for the *Precision*, and 17% for the *F1-Score*. While it is lower with 14% in case of the *Recall* (i.e. it gives more *False Negatives*). In this case we have a lower value for the ratio $FP/TP$, while a higher ratio $FN/TP$. The reason for this is the $\omega_t$ parameter in $CP$, which has more restriction in allowing a rule to be selected and to perform a prediction, thus resulting in less *false positives* and more *false negatives*.

#### 2.4.4.5 Forecasting model using $CP$ measure vs existing prediction methods that use sequential patterns/rules processing

From the initial 170 simple rules, we have selected those with a $CP$ value higher than a threshold value $\theta$. Eight threshold values 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, $\theta^*$ and 1.0 ($\theta^*$ corresponds to the medium value of the function $f(t)$ –see Equation (2.13)) for the $CP$ measure have been used for our $s$ and $w_t$ choices.

The prediction model using the $CP$ measure allows us to retain the rules that give an *Accuracy* of up to 0.886 (see Tables 2.5 and 2.6). Labsky et.al. [Labsky 05], have obtained a value for *Accuracy* of 0.61 using *N-gram model*, 0.59 using *set covering algorithm*, and 0.49 using the *compositional algorithm*. However, in [Labsky 05] the authors only consider in rules the consequent that appear at the next time stamp while we consider time-interval, this is why there is a big difference in the performance results. While Liu and Keselj

[Liu 07] have the highest prediction *Accuracy* of 0.643 in case of the *3 N-gram size model*, and 0.644 when applying *4 N-gram size model*.

This difference in Accuracy is because of the $CP$'s capacity in filtering the rules of the form $V_i \to V_n$ which have closer antecedents and consequents. The rules with higher $CP$ value are expected to have better prediction performance in the future. However, in this way we might loose several interesting *complex* rules $V_1V_2...V_{n-1} \to V_n$ (that are found by the models used by other authors) that would give better results than *some simple* rules, but this is a risk we take in order to simplify the proposed approach, as an initial goal, in case of very large databases. Figure 2.11 illustrates these results and extend the Table 2.5 (for $s = 1.25$ and $\omega_t = 360$; the conclusions are similar for the other parameters values). It plots $A$, $P$, $R$, and $F1$ as a function of the number of rules with the highest $CP$ values. The initial decreasing spike is only due to the fact that when adding new rules these rules introduce false positive and false negative cases. This is afterwards redressed in average with additional rules. The optimal set of rules are of size around 18-20. This size corresponds to the number of rules selected with a threshold value of $\theta^*$ (Equation (2.13)).

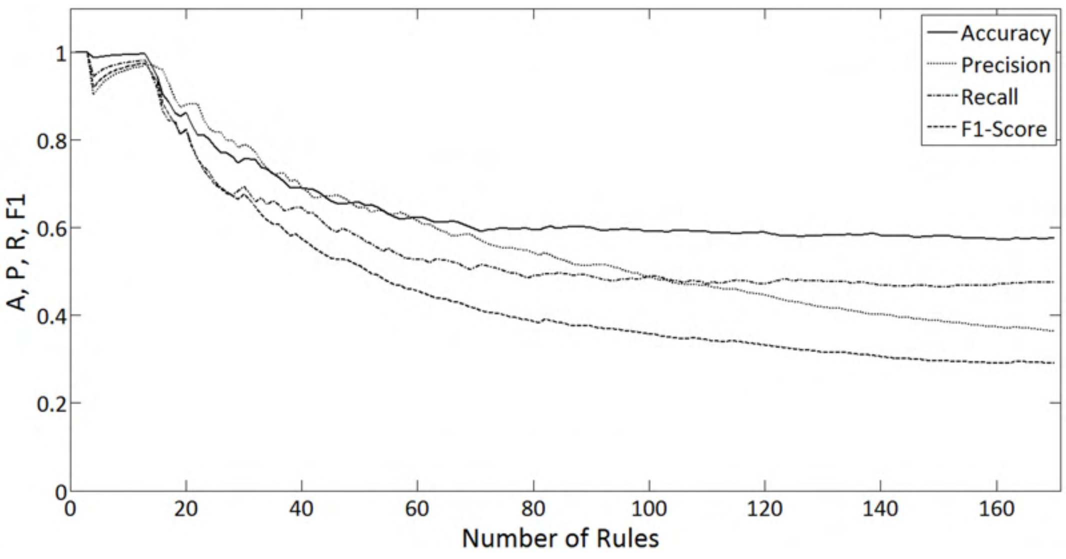


Figure 2.11 : *Accuracy*, *Precision*, *Recall*, and *F1-Score* for the subset of rules containing the highest $CP$.

The values for *Precision*, *Recall*, and *F1-Score* are also presented in Tables 2.5 and 2.6.

We made prediction experiments using both *simple* and *complex* rules. The results for $\omega_t = 360$ and $slope = 1.25$ are presented in Figure 2.12. The $OX$ axis presents the number of *simple rules* and the number of *initial simple rules* used to form the *complex* ones (e.g. from 18 simple we formed 22 complex, from 25 we formed 59, and so on). According to the graphs, we can observe that the *Accuracy* of the *simple rules* begins to decrease faster and to make an important difference with the *complex* rules starting from about 18-20 rules. We can observe that this value corresponds to the number of rules selected after a threshold value of $\theta^*$ as a medium value of the function $f(t)$ (Equation (2.13)). In case of the *Precision*, the *simple rules* have lower performance. This means that these give

Table 2.5 : Number of rules *nr*, *Accuracy* (A), *Precision* (P), *Recall* (R), and *F1-Score* (F1) for $\omega_t = 360$ according to *slope* $s$ and *threshold* $\theta$ set for the *CP*.

| $s$ | $\theta$ | *0.4* | *0.5* | *0.6* | *0.7* | *0.8* | *0.9* | *0.960* | *1.0* |
|---|---|---|---|---|---|---|---|---|---|
| 1.05 | nr | *93* | *64* | *47* | *39* | *28* | *21* | *18* | *18* |
| | A | 0.596 | 0.611 | 0.655 | 0.692 | 0.749 | 0.837 | 0.863 | 0.863 |
| | P | 0.514 | 0.586 | 0.670 | 0.710 | 0.801 | 0.883 | 0.897 | 0.897 |
| | R | 0.480 | 0.540 | 0.599 | 0.647 | 0.693 | 0.787 | 0.843 | 0.843 |
| | F1 | 0.370 | 0.440 | 0.530 | 0.587 | 0.677 | 0.789 | 0.839 | 0.839 |
| 1.25 | $\theta$ | *0.4* | *0.5* | *0.6* | *0.7* | *0.8* | *0.9* | *0.991* | *1.0* |
| | nr | *97* | *70* | *52* | *41* | *31* | *21* | *18* | *18* |
| | A | 0.596 | 0.597 | 0.653 | 0.689 | 0.757 | 0.837 | 0.863 | 0.863 |
| | P | 0.499 | 0.578 | 0.635 | 0.682 | 0.785 | 0.883 | 0.897 | 0.897 |
| | R | 0.485 | 0.511 | 0.563 | 0.634 | 0.676 | 0.787 | 0.843 | 0.843 |
| | F1 | 0.364 | 0.416 | 0.497 | 0.565 | 0.662 | 0.789 | 0.839 | 0.839 |

Table 2.6 : Number of rules *nr*, *Accuracy* (A), *Precision* (P), *Recall* (R), and *F1-Score* (F1) for $\omega_t = 420$ according to *slope* $s$ and *threshold* $\theta$ set for the *CP*.

| $s$ | $\theta$ | *0.4* | *0.5* | *0.6* | *0.7* | *0.8* | *0.9* | *0.966* | *1.0* |
|---|---|---|---|---|---|---|---|---|---|
| 1.05 | nr | *103* | *78* | *54* | *44* | *31* | *23* | *18* | *17* |
| | A | 0.598 | 0.605 | 0.636 | 0.667 | 0.755 | 0.810 | 0.862 | 0.886 |
| | P | 0.481 | 0.543 | 0.641 | 0.673 | 0.785 | 0.847 | 0.897 | 0.929 |
| | R | 0.467 | 0.484 | 0.544 | 0.606 | 0.675 | 0.843 | 0.843 | 0.844 |
| | F1 | 0.351 | 0.391 | 0.484 | 0.536 | 0.661 | 0.838 | 0.838 | 0.859 |
| 1.25 | $\theta$ | *0.4* | *0.5* | *0.6* | *0.7* | *0.8* | *0.9* | *0.992* | *1.0* |
| | nr | *107* | *82* | *55* | *45* | *32* | *25* | *18* | *17* |
| | A | 0.598 | 0.610 | 0.629 | 0.658 | 0.759 | 0.783 | 0.862 | 0.886 |
| | P | 0.467 | 0.542 | 0.643 | 0.671 | 0.773 | 0.817 | 0.897 | 0.929 |
| | R | 0.467 | 0.479 | 0.535 | 0.596 | 0.632 | 0.704 | 0.843 | 0.844 |
| | F1 | 0.343 | 0.389 | 0.476 | 0.529 | 0.639 | 0.697 | 0.838 | 0.859 |

more *False Positives* than the *complex rules*. For the *Recall* the facts change: now the *complex rules* are the ones which give less performance, i.e. more *False Negatives*. The fast decreasing in *Recall* and *Precision* graphics around 4 rules tells us about the fact that the *CP* measure still can give a high importance to some rules that are not quite good in the prediction process. The *F1-Score* graph is actually a generalization between *Precision* and *Recall*.

## 2.5 Modified Closeness Preference Measure for Patterns extraction

In this Section we propose interestingness measures that will favour the smaller time-distance between the itemsets of a pattern (Section 2.4 was concerned by sequential rules).

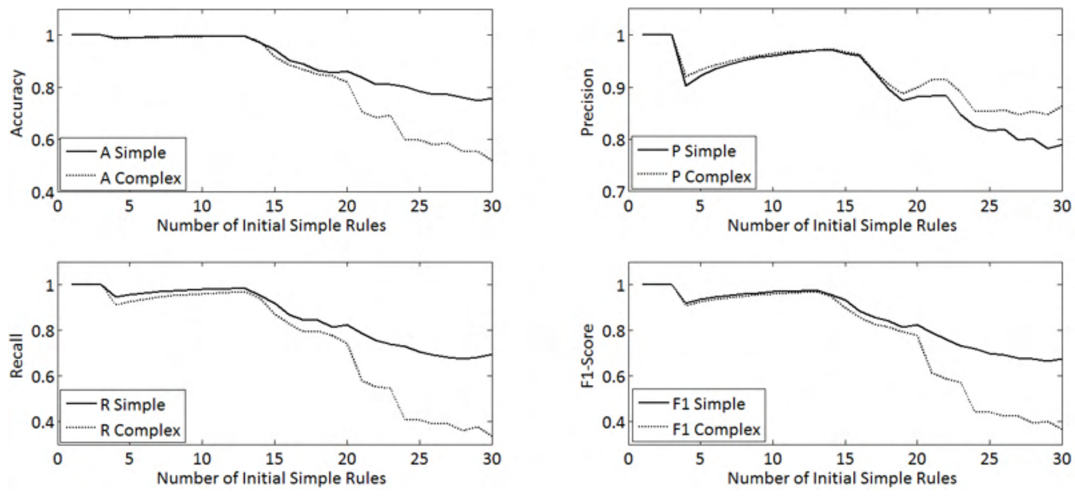


Figure 2.12 : A comparison between *simple* and *complex rules*. Note, that the $OX$ axis represents the number of *simple rules*. The information from the axis does not correspond to the number of *complex rules*, but it corresponds to the initial number of *simple rules* from which the *complex* ones were derived. $CP$ parameters were: $\omega_t = 360$, $slope = 1.25$.

These should be used in a pre-processing step of the pattern mining process. One of the measure fulfils the *anti-monotone* property (defined in Section 2.1), while the other doesn't. This results in the fact that the search space of the first measure will be much larger than for the second one. The mining principles of the measures are also presented followed by comparison with *Support* on a toy example and a real database.

## 2.5.1 Closeness Weight implementation

As it has been stated in Definition 3, a function has to be introduced that will advantage the smaller time-difference between the itemsets of a pattern. We take into consideration the time between 2 consecutive itemsets, it results that we may consider that each pattern of length $n$ is made of $n-1$ subpatterns of length 2. So, our measure will have in its formula a coefficient of the form $C(\omega_t, \sigma_t)_{pattern} = \frac{1}{n-1}\sum_{i=1}^{n-1} C(\omega_t, \sigma_t)_{pattern\ length\ 2}$.

Now, we denote a pattern as $P_1P_2...P_n$, where $P_i$ is an itemset, and $1 \geq i \leq n$. Thus, we define the closeness weight between 2 itemsets $P_i$ and $P_{i+1}$ of a pattern using the weighting function proposed in Equation (2.1) and Figure 2.2 (page 13), where the influence on the measure's value of the itemset $P_{i+1}$ is weighted according to its time distance from $P_i$:

$$f(t, s, \omega_t) = \frac{1}{1 + s^{t-\omega_t}} \tag{2.18}$$

where $t = t_{i+1} - t_i$, $s$ is the slope of the plot ($s > 1$) and is directly proportional with the user-preference time-interval $\sigma_t$ (Definition 3). The greater $s$ implies that the itemset $P_{i+1}$ closer to the end of $\omega_t$ in relation to the itemset $P_i$ will be much more penalized. As $s$ and $\omega_t$ are user-fixed parameters, we simply denote the function by $f(t)$.

For a given value for $\sigma_t$ and the minimum value $f(\sigma_t)$ of $f(t)$ in the interval $[0, \sigma_t]$, the slope $s$ is given by (reminder of Equation (2.19)):

$$s = \sqrt[\sigma_t - \omega_t]{\frac{1}{f(\sigma_t)} - 1} \tag{2.19}$$

What we search for are the patterns whose itemsets are as close as possible, but we make an average of this closeness between itemsets. For example, let's suppose that we have a sequence $s = (a_1 b_1)\ c_2\ c_3\ a_4\ c_5\ d_6\ c_7\ d_8\ c_9$. We search the pattern $a\ c\ d$ inside this sequence. First, we find the subpattern $a\ c$, but we do not know to which $a$ and $c$ our pattern refers to. To solve this issue, we apply the following technique: we calculate an average value of time-distance weights of all $c$ with respect to all $a$ until the end of the sequence or until a new instance of the searched pattern occurs (*e.g.* if the pattern $s$ continued with $a_{10}\ c_{11}\ d_{12}$, then we take for the first instance of $a\ c\ d$ only the itemsets until $c_9$). This implies that we consider all $c$ between $a_1$ and $a_4$ with respect to $a_1$ meaning $c_2$ and $c_3$; and all $c$ between $a_4$ and the end of the sequence, meaning $c_5$, $c_7$ and $c_9$. The distance between $a_4$ and $c_5$, $c_7$ and $c_9$ is considered twice: once with a window having as origin $a_1$; and the second time with a window starting from $a_4$. In this way we disadvantage the itemsets which are further from the origin, but at the same time we advantage the closeness between these itemsets and the next searched ones appearing closer to the end of the sequence (*e.g.* as $d$ is expected to appear further than $c$ w.r.t. $a$, through this technique we disadvantage the time-difference from $a_1$, but advantage the time-distance from $a_4$).

Next, we find the subpattern $c\ d$, and we compute a medium value of time-distances of all $d$ between $c_2$ and $c_3$ (*i.e.* 0), between $c_3$ and $c_5$ (0 again), $c_5$ and $c_7$ (*i.e.* $d_6$), $c_7$ and $c_9$ (*i.e.* $d_8$), and $c_9$ until the end of the sequence (*i.e.* 0). Another aspect that should be mentioned is why we consider the elements *after* the last itemset of the searched pattern (*e.g.* in our case we still consider $c_9$ even if it is after the last $d$, *i.e.* $d_9$, of the searched pattern $a\ c\ d$). This is done in order to disadvantage the itemsets which are independent from each other and occur randomly inside a sequence. Thus, if $c$ occurs independently of $a$ and $d$, then it might appear many times before, between, and after $a$ and $d$, and this fact will diminish the final measure value.

Thus, first we define the strength of the time-relation between a single itemset $P_i$ and an (or several) itemset(s) $P_{i+1}$ as follows:

$$C_{P_{i+1}|P_i} = \frac{1}{n_{t_{P_{i+1}|P_i}}} \sum_{j=1}^{n_{t_{P_{i+1}|P_i}}} \frac{1}{1 + s^{t_{P_{i+1,j}} - \omega_t}} \tag{2.20}$$

where $n_{t_{P_{i+1}|P_i}}$ is the number of $P_{i+1}$ between 2 consecutive $P_i$, or, if there is no another $P_i$, then we look until the end of the sequence or until a new appearance of the searched pattern; while $t_{P_{i+1,j}}$ is the time distance of each $P_{i+1}$ from the beginning of the window (starting from the considered $P_i$). In the above example, we would have $P_i = a$, $P_{i+1} = c$, and $P_{i+2} = d$ and this expression calculates *e.g.* $c_2$ and $c_3$ with respect to $a_1$, or $c_5$, $c_7$ and $c_9$ with respect to $a_4$.

Secondly, we have to calculate an average value of each $C_{c|a}$:

$$C_{mult\ P_{i+1}|P_i} = \frac{1}{n_{t_{P_i}}} \sum_{m=1}^{n_{t_{P_i}}} CP_{P_{i+1}|P_{i,m}} \tag{2.21}$$

where $n_{t_{P_i}}$ is the number of $P_i$ between the previous and next itemsets, and $C_{P_{i+1}|P_{i,m}}$ is the expression from the Equation (2.20) for each $P_i$. In our example this expression would calculate a medium value of $c_2$ and $c_3$ with respect to $a_1$ and $c_5$, $c_7$ and $c_9$ with respect to $a_4$.

As it has been mentioned earlier, we consider that each pattern of length $n$ is made of $n-1$ subpatterns of length 2. By taking into consideration this aspect together with the Equation (2.21), we can define a function $f_{pattern}$ in the following way[6]:

$$f_{pattern} = \frac{1}{nr_{\text{2-length subpattern}}} \cdot \sum_{k=1}^{nr_{\text{2-length subpattern}}} \left[ \frac{1}{n_{t_{P_{i,k}}}} \sum_{m=1}^{n_{t_{P_{i,k}}}} \frac{1}{n_{t_{P_{i+1}|P_{i,k,m}}}} \sum_{j=1}^{n_{t_{P_{i+1}|P_{i,k,m}}}} \frac{1}{1+s^{t_{P_{i+1,j}}-\omega_t}} \right] \tag{2.22}$$

### 2.5.2 $MCP_{sc}$: Support - Confidence - Closeness Preference based measure

In the following part we present the first proposed measure for the patterns extraction and ranking. We are showing the thinking process taken in its computation.

In order to consider the statistical appearance of a pattern, we combine the $f_{pattern}$ formula (Equation (2.22)) with *Support* and *Confidence* for a pattern. We introduce some modifications in case of the *Confidence* measure to make it usable in the pattern processing.

As we know, the *Support* is defined as follows:

$$supp_{pattern} = \frac{nr_{pattern\ occurrences}}{|DB|} \tag{2.23}$$

The *Confidence* is defined for sequential rules as $\frac{R(AB)}{R(A)}$ (Equation (2.14)), where $R(AB)$ is the number of sequences containing the rule $A \rightarrow B$, and $R(A)$ is the number of sequences containing the antecedent $A$ in the database. For sequential pattern however, we do not know what an *antecedent* stands for, this is why we first of all propose the following definition of an *incomplete pattern*:

> **Definition 4:** An *incomplete sequential pattern* is the relation between a pattern $p$ and a sequence $S$ of a form $I(p; S)$ where $p$ is the searched pattern through the sequence $S$ such that several first itemsets of $p$ (at most $n-1$, where $n$ is the total number of itemsets of $p$) occur inside the sequence $S$.

For example, let's suppose that we are searching for the pattern $acf$ in the sequence $abcdef$. In this case, we do find it in the sequence, meaning that it would not be considered

[6]We should point out that we take into consideration the weighted value of the 2-length subpatterns even if it is outside $\omega_t$ value (in comparison to the expression from $CP$ calculation for sequential rules, Equation (2.3)).

as *incomplete*. While if we are looking for the pattern $bcz$, then it would be an *incomplete* one. However, if we are searching for the pattern $xaz$, then we cannot find a relation of the form $I(p;S)$ after analyzing the sequence $abcdef$, because the first item $x$ cannot be found in the sequence.

Thus, we define the equation for *Confidence* in the following way:

$$conf_{pattern} = \frac{nr_{pattern\ occurrences}}{nr_{pattern\ occurrences} + nr_{incomplete\ patterns}} \tag{2.24}$$

Another aspect, is that we might have several occurences of a pattern in a single sequence (note, that in spite of this fact the value of $nr_{pattern\ occurences}$ ($nr_{p\ o}$) and $nr_{incomplete\ patterns}$ ($nr_{i\ p}$) does not change). Taking into consideration this remark and the Equations (2.22), (2.23), and (2.23), we can obtain the final proposed measure (*Modified Support - Confidence - based Closeness Preference*) $MCP_{sc}$:

$$MCP_{sc} = \frac{nr_{p\ o}}{nr_{p\ o} + nr_{i\ p}} \cdot \frac{nr_{p\ o}}{|DB|} \cdot \frac{1}{nr_{p\ o}} \sum_{s=1}^{nr_{p\ o}} \left[ \frac{1}{nr_{p\ in\ seq}} \sum_{t=1}^{nr_{p\ in\ seq}} f_{pattern} \right] \tag{2.25}$$

where $f_{pattern}$ is calculated as in Equation (2.22), $|DB|$ is the total size of the database, $nr_{p\ in\ seq}$ is the number of how many times the pattern $p$ is found in a given sequence.

The proposed measure is to be used as a pre-processing one instead of *Support*. The question is how we value single itemsets pattern which could not be decomposed into 2-length subbpatterns (these are formed at the initial stages of the $GSP$ algorithm).

In the case of a pattern with a single itemset, e.g. ($abc$) instead of $abc$, the value of $f_{pattern}$ from Equation (2.22), is replaced simply by $\frac{1}{1+s^{t_j - \omega_t}}$:

$$MCP_{single\ itemset\ pattern} = \frac{po}{|DB|} \cdot \frac{1}{1 + s^{t_j - \omega_t}} \tag{2.26}$$

as $nr_{ip}$ is always 0.

### 2.5.3 $MCP_{s\ func}$: Support - Closeness Preference based measure combined with an additional weighting function to fulfil the anti-monotone property

In this Subsection we introduce a measure also based on the $f_{pattern}$ expression from the Equation (2.22), but which fulfils the *anti-monotone* property.

As it has been mentioned, the *Apriori* principle states that the measure of a pattern cannot be greater than that of its subpattern. Thus, we have to meet the following aspect $M_{pattern} \leq M_{subpattern}$.

One of the ways that seem to fulfil this property is if instead of the medium value of the expressions presented in the Equation (2.22), we compute their product. So, as in the previous case, we use the same function $f_{pattern}$, but its influence on the $MCP_{s\ func}$ measure will be:

$$\prod_{s=1}^{nr_{seq\ with\ p}} \left[ \frac{1}{nr_{p\ in\ seq}} \sum_{t=1}^{nr_{p\ in\ seq}} f_{pattern_t} \right] \tag{2.27}$$

Even if it seems that the patterns' values should always have a measure lower than that of its subpatterns because of the fact that values inside the $\prod$ operation is $< 1$, there are several problems which arise. There might be subpatterns with their $MCP_{s\ func}$ value lower than that of the next formed pattern.

Let's analyze deeper our problem, depicted in Figure 2.13. First of all, it should be taken into consideration that $f_{pattern}$ is applied *only* on a given instance of a pattern, and later we make a medium value on all the values from the pattern's appearances in the sequences, and a product of all the values from the sequences containing the pattern. Because of the fact that $xyz$ might happen more times than $xyzt$, we might have $xyz$ final $f_{pattern}$ value obtained from all patterns and sequences lower than that of $xyzt$. So, we should define a function such that:

- its value for $xyzt$ and $xyzr$ is lower than for $xyz$
- its value for $xyzr$ is lower than for $xyzt$

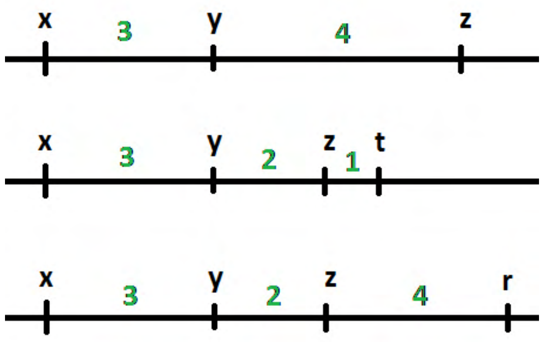


Figure 2.13 : An example of the time-differences problem.

The only way to define such a function is when taking into consideration the following parameters: the *size* of the pattern, and the *medium time-interval* in a pattern. So, the function $f(nr, \Delta_{t_{average}})$ should:

- $f(nr, \Delta'_{t_{average}}) < f(nr-1, \Delta''_{t_{average}})$, regardless of the $\Delta'_{t_{average}}$ value
- $f(nr, \Delta'_{t_{average}}) < f(nr, \Delta''_{t_{average}})$ for $\Delta'_{t_{average}} > \Delta''_{t_{average}}$

We propose the function in Figure 2.14. With such a function way we make sure that the first important parameter is the number of items $nr$, after which we take into consideration the $\Delta_{t_{average}}$, s.t. it never passes the value of the function in the point $nr-1$, in this way we guarantee the apriori principle.

The formula of the proposed function is:

$$f(nr, \Delta_{t_{average}}) = 0.9^{nr} + (0.9^{nr-1} - 0.9^{nr}) \cdot \Delta_{t_{average}} \tag{2.28}$$

where $\Delta_{t_{average}}$ will be replaced by the Equation (2.27). The value of 0.9 was selected because it has almost a constant decreasing with $nr$ .

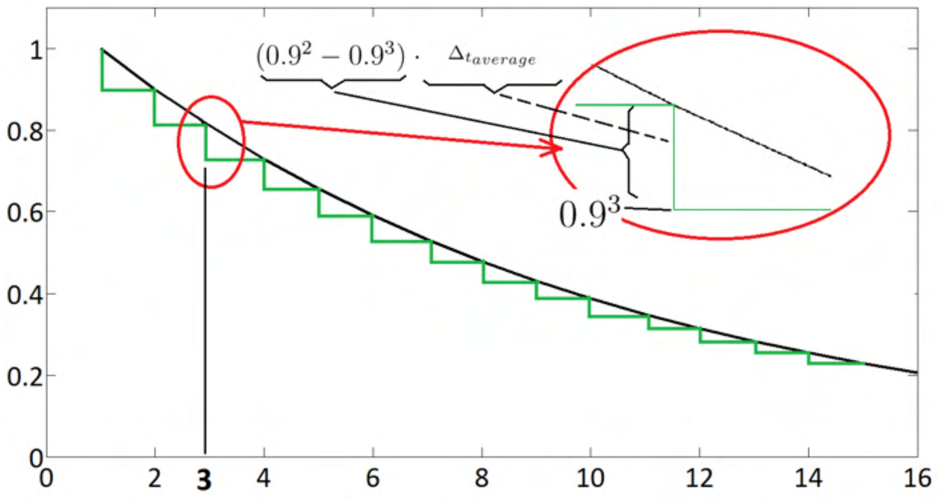


Figure 2.14 : An explanation of the *coefficient function* to meet our requirements. We can see, that if $\Delta_{t_{average}}$ is higher, it takes the upper value of the ceiling, otherwise, it takes the bottom value, but it never passes to the previous coefficient values of its subpatterns.

Another aspect consists in how to disadvantage the patterns which are not frequent in the database, but have high $f_{pattern}$ values. For example, there might be cases where $f_{pattern}$ has high values, but the pattern itself is not actually statistically valid (i.e. it occurs too few times in the entire database). In order to overcome this issue, we simply integrate the *Support* measure as defined in the Equation (2.23).

It results, that the final formula for the algorithmical measure is as follows:

$$MCP_{s\ func} = \frac{nr_{p\ o}}{|DB|} \cdot \left[ 0.9^{nr} + (0.9^{nr-1} - 0.9^{nr}) \cdot \prod_{s=1}^{nr_{p\ o}} \left[ \frac{1}{nr_{p\ in\ seq}} \sum_{t=1}^{nr_{p\ in\ seq}} f_{pattern_t} \right] \right] \tag{2.29}$$

## 2.5.4 Pattern Mining and $MCP_{sc}$ with $MCP_{s\ func}$ Properties

In the current part we present the way of mining the sequential patterns according to the proposed measures. We do not focus on the speed of the extraction, but only on the time-closeness weight of a given pattern.

The pattern formation is done using the *Join-Phase* from the well-known *Generalized Sequential Patterns* algorithm [Srikant 96]. In this phase the set of $L_{k-1}$ previous patterns is joined with itself in order to generate a superset of the set of the future possible $k$-patterns. While the *Prune Phase* is done differently for $MCP_{sc}$ and $MCP_{s\ func}$.

Thus, at the beginning we set a threshold $MCP_{thr}$ in order to select the *valid* patterns. In case of the $MCP_{s\ func}$ the process is quite simple because it fulfils the apriori principle as shown earlier. So, we compute the value of the measure of a given pattern, and if it passes the threshold value, then it is considered as a valid one and passes to the next joining phase, otherwise the pattern is discarded.

In case of the $MCP_{sc}$, the measure does not fulfil the apriori principle, and we have to keep all the patterns from the current joining phase to the next joining phase. This aspect requires a lot of time-processing. At the end, the patterns which pass the $MCP_{thr}$ value will be considered valid.

In order to show that $MCP_{sc}$ does not fulfil the apriori principle, we state the following property:

> **Property 1:** A A sequential subpattern $S1 = st_1, st_2, ..., st_{n-1}$ with $MCP_{sc} < MCP_{sc\ thr}$ might give a sequential pattern $S = st_1, st_2, ..., st_{n-1}, st_n$ with $MCP_{sc} \geq MCP_{sc\ thr}$ iff the $MCP_{sc}$ value of the second subpattern $S2 = st_2, ..., st_{n-1}, st_n$ used in forming the final pattern $S = st_1, st_2, ..., st_{n-1}, st_n$ is greater than $MCP_{sc\ thr}$

The above Property 2.5.4 is valid when $MCP_{sc\ S2} < MCP_{sc\ thr}$ and $MCP_{sc\ S1} \geq MCP_{sc\ thr}$ also. This theorem states the fact that a pattern obtained from previous stage might give a valid pattern in the next stage of pattern composition even if its $MCP_{sc}$ value is smaller than $MCP_{sc\ thr}$. It implies the fact that in comparison to *Support* pruning (where a pattern is certainly not valid if the two subpatterns forming it have a value less then $supp_{thr}$), we have to keep the entire set of subpatterns. The only patterns which do not pass to the next step are those which have the $MCP_{sc} = 0$, meaning that there are no instances of such patterns in the entire database.

***Proof***. Let's denote by $\alpha_k = \frac{1}{nt_{P_{i,k}}} \sum_{m=1}^{nt_{P_{i,k}}} \frac{1}{nt_{P_{i+1}|P_{i,k,m}}} \sum_{j=1}^{nt_{P_{i+1}|P_{i,k,m}}} \frac{1}{1+s^{t_{P_{i+1,j}}-\omega t}}$. Then, the Equation (2.25) might be rewritten as:

$$MCP_{sc} = \left[\frac{1}{nr_s}\sum_{s=1}^{nr_s}\left[\frac{1}{nr_p}\sum_{t=1}^{nr_p}\left[\frac{1}{nr_{2-len\ p}}\sum_{k=1}^{nr_{2-len\ p}}\alpha_k\right]\right]\right]\cdot\frac{n_c\cdot n_c}{(n_c+n_i)|DB|} \tag{2.30}$$

where $n_c$ and $n_i$ are the number of complete and incomplete patterns respectively.

We suppose that we have 2 patterns with $MCP_{sc1}$ and $MCP_{sc2}$.

The values for $MCP_{sc}$ might be written as:

$$\begin{aligned} MCP_{sc1} &= \left[\frac{1}{nr_{s1}}\sum_{s=1}^{nr_{s1}}\left[\frac{1}{nr_{p1}}\sum_{t=1}^{nr_{p1}}\left[\frac{1}{nr_1}\sum_{nr_1}(\alpha_1+\alpha_2+...+\alpha_{n-1})\right]\right]\right]\cdot\frac{n_{c1}\cdot n_{c1}}{(n_{c1}+n_{i1})|DB|} \\ MCP_{sc2} &= \left[\frac{1}{nr_{s2}}\sum_{s=1}^{nr_{s2}}\left[\frac{1}{nr_{p2}}\sum_{t=1}^{nr_{p2}}\left[\frac{1}{nr_2}\sum_{nr_2}(\alpha_2+...+\alpha_{n-1}+\alpha_n)\right]\right]\right]\cdot\frac{n_{c2}\cdot n_{c2}}{(n_{c2}+n_{i2})|DB|} \end{aligned} \tag{2.31}$$

Results the following $MCP_{sc\ 1+2}$ value:

$$MCP_{sc\ 1+2} = \left[\frac{1}{nr_{s1+2}}\sum_{s=1}^{nr_{s1+2}}\left[\frac{1}{nr_{p1+2}}\sum_{t=1}^{nr_{p1+2}}\left[\frac{1}{nr_{1+2}}\sum_{nr_{1+2}}(\alpha_1+\alpha_2+...+\alpha_{n-1}+\alpha_n)\right]\right]\right]\cdot \\ \cdot\frac{n_{c1+2}\cdot n_{c1+2}}{(n_{c1+2}+n_{i1+2})|DB|} \tag{2.32}$$

Let's consider that we have $MCP_{sc1} < MCP_{sc\ thr}$ and $MCP_{sc2} \geq MCP_{sc\ thr}$. Based on the same Equations (2.31) and (2.32) we can see that even if the $nr_{1+2}$ value might be higher than $nr_1$ and $nr_2$, the final medium coefficients values of $\alpha$ between the first and second subpatterns also might be higher than that of the first pattern, while lower than that of the second pattern. This could result in a value $MCP_{sc\ 1+2} \geq MCP_{sc\ thr}$.

The search space of $MCP_{sc}$ is large, but will we get the same patterns by diminishing the searched patterns after a pre-processing with *Support*, followed by a post-processing with $MCP_{sc}$? We base on the assumption that $MCP_{sc}$ contains in its formula (Equation (2.25)) the relation for *Support*, *i.e.* $\frac{nr_{p\ o}}{|DB|}$. Let's rewrite the Equation (2.25) in the following form:

$$MCP_{sc} = conf \cdot supp \cdot g_{pattern} \tag{2.33}$$

where $g_{pattern} = \frac{nr_{p\ o}}{|DB|} \cdot \frac{1}{nr_{p\ o}} \sum_{s=1}^{nr_{p\ o}} \left[ \frac{1}{nr_{p\ in\ seq}} \sum_{t=1}^{nr_{p\ in\ seq}} f_{pattern} \right]$. We have $conf$ and $supp$ in the interval $(0;1)$. It results that $MCP_{sc} \in ((0;1) \cdot supp)$. We denote $conf \cdot g_{pattern} = coeff \in (0;1)$. In this way there could be patterns that do not pass the support threshold, but pass the $MCP_{sc}$ threshold, patterns that will be lost, but could be interesting from our point of view (*i.e.* closer itemsets giving a high value for $coeff$ resulting in an overall high $MCP_{sc}$).

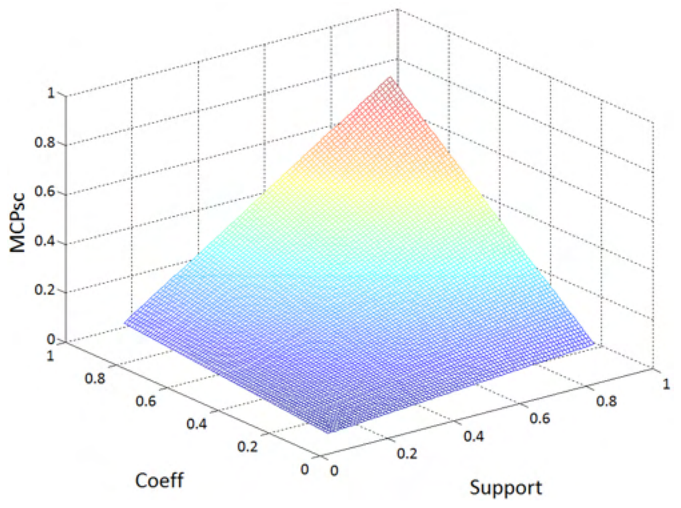


Figure 2.15 : $MCP_{sc}$ as a function of $coeff$ and $supp$. Both $coeff$ and $supp$ varied from 0.1 to 0.9.

For example, Figure 2.15 shows a $3D$ plot for the $MCP_{sc} = coeff \cdot supp$, where $coeff$ and $supp$ varied from 0.1 to 0.9. In case if one sets a $supp_{thr} = 0.4$ and $MCP_{sc\ thr} = 0.2$ there could be patterns with a $supp < supp_{thr}$, but with $MCP_{sc} > MCP_{sc\ thr}$ that will be lost (Figure 2.16).

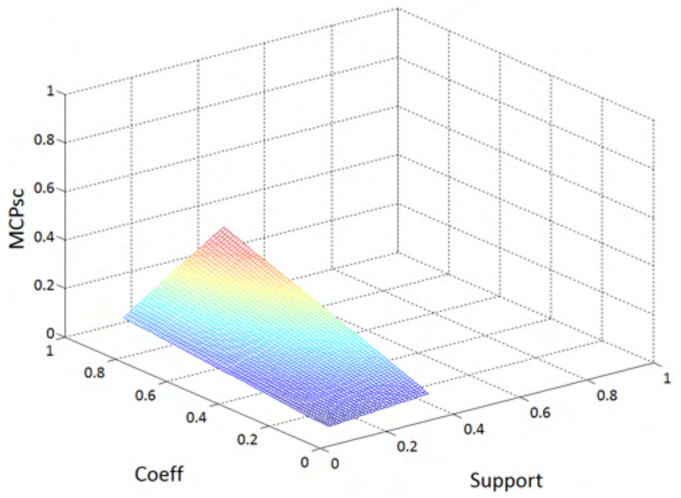


Figure 2.16 : Lost patterns for a $supp_{thr} = 0.4$. Note, that in case of $MCP_{sc\ thr} = 0.2$ the patterns with measure's values $> thr$ at the upper part of the graph will be lost.

### 2.5.5 Toy Example

We make a comparison on a toy example database between patterns extraction done by $MCP_{sc}$, $MCP_{s\ func}$, and *Support*.

Let's consider the 10 sequences from Table 2.1. We consider $s = 2$ and the time window length $\omega_t = 2$, the function beeing presented in Figure 2.17 with its corresponding values.

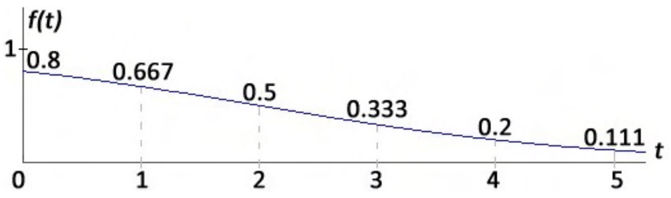


Figure 2.17 : The function used in toy example. The *slope* $s = 2$ and the time window $\omega_t = 2$.

After applying the extraction algorithm, one of the obtained pattern is $d\ b\ a$. Let's show step by step how to calculate its $MCP_{sc}$ value. We are based on Equation (2.25). First of all, we find the sequences where the pattern $d\ b\ a$ appears, *i.e.* $S_{10}$, $S_{30}$, and $S_{80}$, resulting in $n_{p\ o} = 3$. From each of these sequences we calculate the expression $\frac{1}{nr_{p\ in\ seq}} \sum_{t=1}^{nr_{p\ in\ seq}} f_{pattern}$. In all sequences we have just one occurrence of the pattern, *i.e.* $nr_{p\ in\ seq} = 1$[7] In order to calculate $f_{pattern}$ we use Equation (2.22). The searched pattern $d\ b\ a$ is made of the following 2-length sub-patterns $d\ b$ and $b\ a$, thus

[7]If we would have had a sequence $a\ (de)\ b\ a\ b\ e\ d\ b\ e\ a$, then $nr_{p\ in\ seq} = 2$, and in the case of the pattern $d\ b\ a$ we would calculate an $f_{pattern}$ for $a\ (de)\ b\ a\ b\ e$, and another $f_{pattern}$ for $d\ b\ e\ a$.

$nr_{2\text{-}length\ subpattern} = 2$. It results:

$$\begin{aligned} f_{pattern}^{S_{10}} &= \frac{1}{2}\left[\frac{1}{1}\frac{0.667+0.333}{2} + \frac{1}{2}(0.667+0)\right] = 0.417 \\ f_{pattern}^{S_{30}} &= \frac{1}{2}\left[\frac{1}{1}\cdot 0.5 + \frac{1}{2}\cdot 0.667\right] = 0.583 \\ f_{pattern}^{S_{80}} &= \frac{1}{2}\left[\frac{1}{1}\frac{0.8+0.5}{2} + \frac{1}{2}(0.8+0)\right] = 0.525 \end{aligned} \tag{2.34}$$

The number of incomplete patterns is $nr_{i\ p} = 4$, as we have appearances of $d$ or $d\ b$ only in $S_{40}$, $S_{50}$, $S_{70}$, and $S_{90}$; while $|DB| = 10$. Results the final $MCP_{sc}$ value:

$$\begin{aligned} MCP_{sc} &= \frac{3}{3+4}\cdot\frac{3}{10}\cdot\frac{1}{3}\left(f_{pattern}^{S_{10}} + f_{pattern}^{S_{30}} + f_{pattern}^{S_{80}}\right) \\ &= \frac{3}{70}(0.417+0.583+0.525) = 0.06536 \end{aligned}$$

Next, we show how we calculate the $MCP_{s\ func}$ value for the same pattern, using Equation (2.29). The values of $f_{pattern}$ are the same with those from the $MCP_{sc}$ calculation (Equation (2.34)). The value for $nr_{p\ o}$ is 3 (there are 3 sequences in the database containing the pattern $d\ b\ a$), and the value for $nr$ is 3, because we have 3 items in the pattern. Replacing the numbers in Equation (2.29), we obtain:

$$\begin{aligned} MCP_{s\ func} &= \frac{3}{10}\cdot\left[0.9^3 + \left(0.9^{3-1} - 0.9^3\right)\cdot f_{pattern}^{S_{10}}\cdot f_{pattern}^{S_{30}}\cdot f_{pattern}^{S_{80}}\right] \\ &= 0.3\cdot(0.729 + 0.081\cdot 0.128) = 0.2218 \end{aligned}$$

The calculation of *Support* is simple, we have 3 sequences containing the pattern $d\ b\ a$ from a total of 10 sequences. Thus, $supp = \frac{3}{10} = 0.3$.

#### 2.5.5.1 $MCP_{sc}$ and $MCP_{s\ func}$ vs *Support*

Next, we make the comparison between $MCP_{sc}$, $MCP_{s\ func}$ and *Support*.

In case of the $MCP_{sc}$ and *Support* extraction comparison, we will take into consideration two different aspects: the number of patterns kept at each stage for future processings, and the number of selected final patterns passing the threshold value at each step.

For this, we have set the threshold values in order to obtain the same number of final patterns. In case of the $MCP_{sc}$: $MCP_{sc\ thr} = 0.0345$, and $supp_{thr} = 20\%$. We have obtained in both cases 163 patterns. In Table 2.7 we have the following information: the total number of patterns to be checked at each stage, the number of patterns to be kept for the next stage, and the number of patterns that pass the threshold value. In case of the *Support* prunning it should be obvious that the number of patterns for the next stage is identical with the number of those which pass $supp_{thr}$ value.

After analyzing the information from Table 2.7, we can observe that the number of patterns kept for the future stages using $MCP_{sc}$ is higher than the ones using *Support*. This means that the $MCP_{sc}$ search space is larger then the one of the *Support* because

Table 2.7 : The number of obtained sequential patterns at each stage of the GSP algorithm using classic *Support* processing and $MCP_{sc}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than threshold value.

| Stage ID | *Support* processing | | | $MCP_{sc}$ processing | | |
|---|---|---|---|---|---|---|
| | NAP | NKPF | NPT | NAP | NKPF | NPT |
| 1 | 40 | 28 | 28 | 40 | 32 | 31 |
| 2 | 157 | 74 | 74 | 197 | 153 | 57 |
| 3 | 216 | 51 | 51 | 490 | 289 | 39 |
| 4 | 27 | 10 | 10 | 189 | 107 | 34 |
| 5 | 0 | 0 | 0 | 34 | 32 | 2 |
| 6 | - | - | - | 4 | 4 | 0 |
| 7 | - | - | - | 0 | 0 | 0 |

it does not fulfil the anti-monotone property. However, at each stage the number of good patterns that pass the threshold values are smaller when using the $MCP_{sc}$ then in the case where *Support* is used, resulting that the $MCP_{sc}$ is more selective. Another aspect observed is that the $MCP$ tends to advantage longer patterns. We can see, that there are 36 patterns with length $\geq 4$, while the *Support* passes just 10 of such patterns.

From these 163 patterns, 95 of them could be found in both $MCP_{sc}$ and *Support* processing. So, there is approximately 60% of common patterns.

In case of the $MCP_{s\ func}$ against *Support*, because of the fact that the proposed measure is directly proportional with the *Support*, and that the toy example database is quite small, the number of extracted and analysed rules in this case are the same between *Support* and $MCP_{s\ func}$ (as in Table 2.7). This is the reason why in Subsections 2.5.5.2 and 2.5.5.3 we consider $MCP_{sc}$ only.

Now, let's take a look at the values of these measures given to certain patterns. The $MCP_{s\ func\ thr}$ was set to 0.1, while the thresholds for the $MCP_{sc}$ and *Support* are the same: 0.0345 and 20% respectively.

Some results are presented in Table 2.8. We can observe that for the patterns where the *Support* gives the same results, in the case of the proposed measures the values and ranking are different. Also, even if the *Support* values are the same (i.e. the patterns 2, 3, and 4, or 5 and 6), the values of the $MCP_{sc}$ and $MCP_{s\ func}$ are different, ranking the patterns in different order, because they take into consideration the time-differences between the itemsets. Also, in case of the patterns 5 and 6, we can see that the pattern 5 is a subpattern for the pattern 6. Despite this, $MCP_{sc5} < MCP_{sc6}$, while $MCP_{s\ func5} > MCP_{s\ func6}$, showing once more time that $MCP_{sc}$ does not satisfy the anti-monotone property, while $MCP_{s\ func}$ does fulfil it. Note, the lower value for the $MCP_{sc}$ is because we have integrated the *Confidence* measure in its formula also.

Table 2.8 : Differences in measure's values between *Support*, $MCP_{sc}$, and $MCP_{s\ func}$.

| Pattern ID | Pattern | *Support* | $MCP_{sc}$ | $MCP_{s\ func}$ |
|---|---|---|---|---|
| 1 | $d\ e\ a$ | 0.4 | 0.126 | 0.294 |
| 2 | $a\ d\ e$ | 0.3 | 0.086 | 0.226 |
| 3 | $a\ (de)\ e$ | 0.3 | 0.037 | 0.197 |
| 4 | $d\ c\ b$ | 0.3 | 0.084 | 0.225 |
| 5 | $a\ d\ c\ b$ | 0.2 | 0.041 | 0.139 |
| 6 | $(ab)\ d\ c\ b$ | 0.2 | 0.095 | 0.125 |

### 2.5.5.2 $MCP_{sc}$ vs *windowing Support*

In the current section we compare $MCP_{sc}$ against *Support* GSP processing if we use the same window interval for the *Support*. In the case of $MCP_{sc}$ we have set an $\omega_t = 2$. We use the same $\omega_t = 2$ for *Support* also.

In this case, with the same value for *Support* $= 20\%$, we have obtained 121 rules. In order to obtain 121 patterns with $MCP_{sc}$ we set its value to $MCP_{sc\ thr} = 0.0495$. The information with the number of patterns at each stage is shown in Table 2.9. From these 121 patterns only 71 are common, which is about 60%. As in the previous case, the search space for $MCP_{sc}$ is larger and the proposed measure tends to advantage longer patterns.

Table 2.9 : The number of obtained sequential patterns at each stage of the GSP algorithm using windowed *Support* $\omega_t$ processing and $MCP_{sc}$ processing. *NAP* - Number of analysed patterns; *NKPF* - Number of kept patterns for the future processing; *NPT* - Number of patterns with IM value greater than the threshold value.

| Stage ID | *Support* processing | | | $MCP_{sc}$ processing | | |
|---|---|---|---|---|---|---|
| | NAP | NKPF | NPT | NAP | NKPF | NPT |
| 1 | 40 | 25 | 25 | 40 | 32 | 29 |
| 2 | 130 | 55 | 55 | 195 | 152 | 47 |
| 3 | 146 | 35 | 35 | 431 | 263 | 28 |
| 4 | 18 | 6 | 6 | 144 | 88 | 16 |
| 5 | 0 | 0 | 0 | 23 | 21 | 1 |
| 6 | - | - | - | 2 | 2 | 0 |
| 7 | - | - | - | 0 | 0 | 0 |

### 2.5.5.3 $MCP_{sc}$ processing vs *Support*, *Confidence*, and $f_{pattern}$ processing

In the current part our aim was to compare the result of the $MCP_{sc}$ (noted further as *Case I*) with the case if we would applied separately the measures that are used in the Equation of the $MCP_{sc}$ (Equation (2.25)) (noted further as *Case II*). So, in the second case there are 3 stages in processing: *pre-processing with Support*, *processing with Confidence*, and *post-processing with a variant of CP* addapted for the patterns, as in Equation (2.22) of the $f_{pattern}$. The differences in the processing are shown in Figure 2.18.

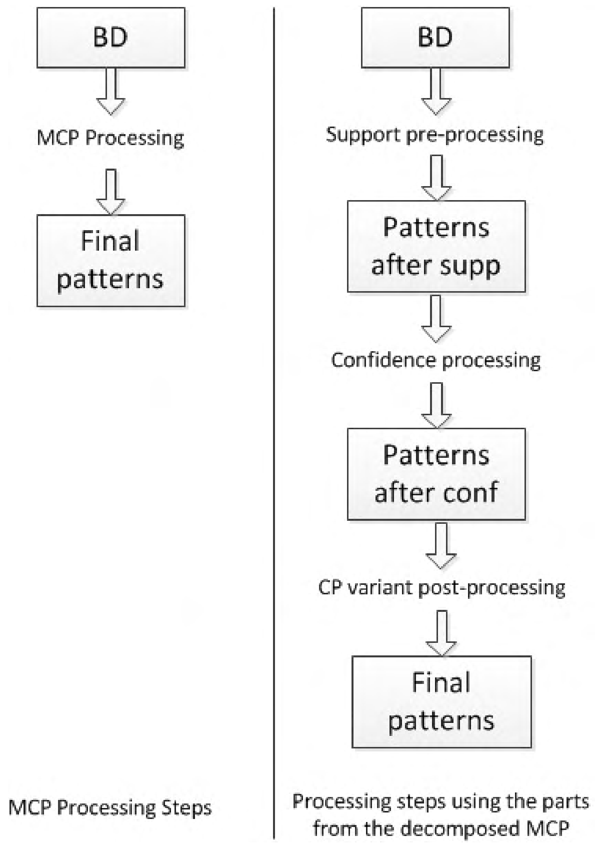


Figure 2.18 : A comparison of the processing steps between $MCP_{sc}$ and the decomposition of the $MCP_{sc}$.

In order to obtain approximately the same number of the final patterns, for the *Case I* we set $MCP_{sc\ thr} = 0.050$ (119 patterns obtained), while for the *Case II* we set the following values: *Support* $= 10\%$ (2571 patterns extracted); *conf* $= 20\%$ (533 patterns remaining); *variant of CP* $= 0.58333$ (122 final patterns remaining).

In order to analyse the results, we have arranged the final patterns in a decreasing order of their final IM values, and took the number of common patterns that are in the first top-$i$ ranked, where $i = \{1..119\}$. The graph with the result showing the number of common patterns is presented in Figure 2.19, while their percentage is in Figure 2.20.

From the results we can notice that the final patterns are different. The highest percentage number of the common patterns is achieved when we take into consideration the top- ranked $72 - 80$ patterns, thus obtaining $51\% - 52\%$ of the same patterns.

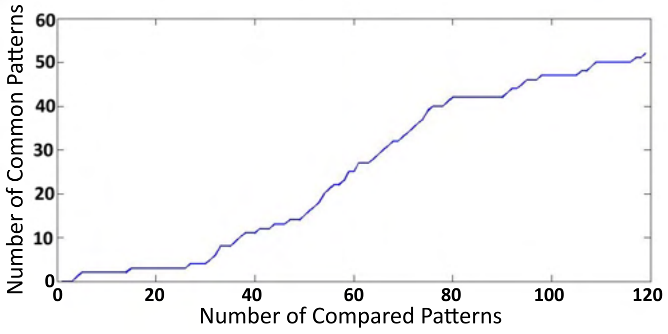


Figure 2.19 : The number of common patterns between *Case I* and *Case II* according to the top-$i$ ranked patterns arranged in a decreasing order of their IM.

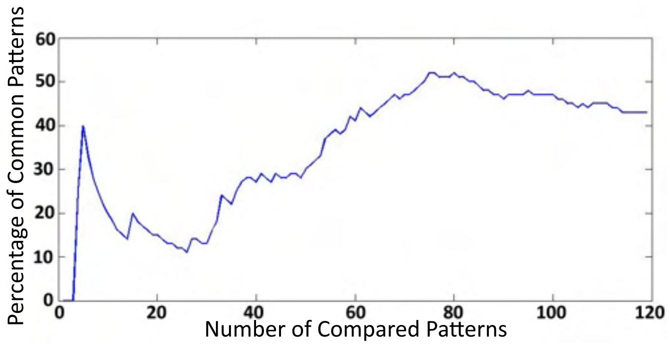


Figure 2.20 : The number of the common patterns in percentage % between *Case I* and *Case II* according to the first taken patterns arranged in a decreasing order of their IM.

### 2.5.6 Real Database analysis

The goal of our case study was to make a comparison between the patterns' extraction using the proposed interestingness measures, i.e. $MCP_{sc}$ and $MCP_{s\ func}$, and *Support*. In order to form the sequential patterns we have implemented the *Generalized Sequential Pattern* algorithm [Srikant 96] presented in the subsection 2.5.4 and adapted for using $MCP_{sc}$ and $MCP_{s\ func}$ measures instead of *Support*. We use the same *ClickStream* database presented in Subsection 2.4.4.2 for rule selection.

#### 2.5.6.1 $MCP_{sc}$ vs *Support* on *ClickStream* database

Because of the fact that $MCP_{sc}$ does not have the anti-monotone property, it means that the processing-time is very high. This is why we have performed a simulation with the following parameters for the function of the $MCP_{sc}$: $\omega_t = 60$, *slope* $s = 1.05$, and $MCP_{sc\ thr} = 0.02$.

The results are shown in Table 2.10. 193 rules have been obtained in the case of the $MCP_{sc}$ processing. In order to obtain the same number of patterns with *Support*, a value of $supp_{thr} = 17.52\%$ was set. We can observe a lot of patterns kept for the future processing, which shows the time-inefficiency of the $MCP_{sc}$, however, in comparison to the *Support* pruning it selects the patterns with closer itemsets. A graphical comparison between the number of common patterns according to the top-$i$ ranked selected patterns by $MCP_{sc}$ and *Support* are given in Figures 2.21 and 2.22. This number increases almost linearly with the number of compared patterns staying at around 70% of the number of compared patterns.

Table 2.10 : The number of obtained sequential patterns at each stage of the GSP algorithm using classic *Support* processing and $MCP_{sc}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than the threshold value.

| Stage ID | $MCP_{sc}$ processing | | | *Support* processing | | |
|---|---|---|---|---|---|---|
| | NAP | NKPF | NPT | NAP | NKPF | NPT |
| 1 | 61 | 61 | 20 | 6 | 6 | 6 |
| 2 | 3094 | 1348 | 26 | 57 | 19 | 19 |
| 3 | 2383 | 2256 | 44 | 58 | 34 | 34 |
| 4 | 3903 | 3779 | 53 | 69 | 50 | 50 |
| 5 | 4735 | 4603 | 39 | 91 | 57 | 57 |
| 6 | 3348 | 3253 | 11 | 93 | 26 | 26 |
| 7 | 891 | 867 | 0 | 25 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 |

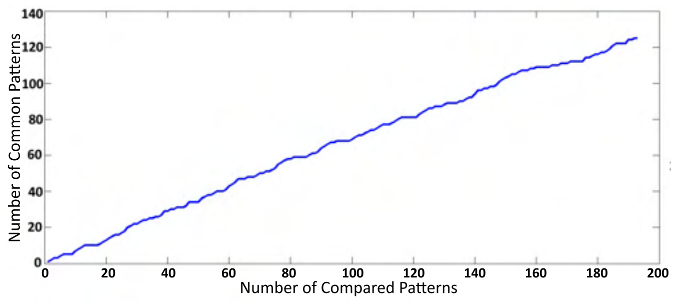


Figure 2.21 : The number of common patterns between $MCP_{sc}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM.

In comparison to the Toy example from Subsection 2.5.5, here is the *Support* who advantages longer patterns (we consider longer patterns those whose number of items$\geq$ 4) *i.e.* 134 vs 103 patterns. This is explained by the fact that longer patterns tend to have larger time-intervals between the itemsets, while this fact is disadvantaged by $MCP_{sc}$ measure. Another point to be mentioned is the much greater number of checked patterns

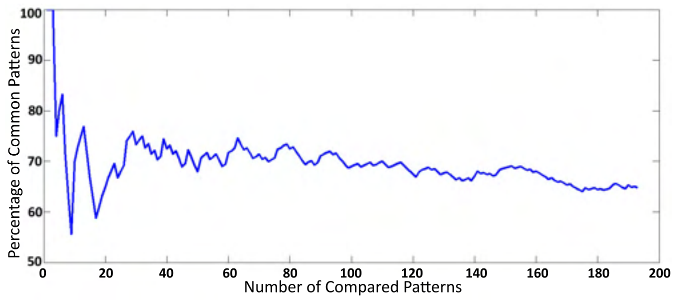


Figure 2.22 : The number of common patterns in percentage % between $MCP_{sc}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM.

at each stage when using $MCP_{sc}$, explained by the fact that it does not fulfil the anti-monotone property in comparison to *Support*.

#### 2.5.6.2 $MCP_{sc}$ vs $MCP_{s\ func}$ on *ClickStream* database

For the $MCP_{sc}$ processing the same parameters are used: $\omega_t = 60$, *slope* $s = 1.05$, and $MCP_{sc\ thr} = 0.02$. In case of $MCP_{s\ func}$ in order to obtain 193 patterns also, we set its threshold of $MCP_{s\ func\ thr} = 0.1072$. Its $\omega_t$ and $s$ values were as in the case of $MCP_{sc}$. The results are shown in Table 2.11. A graphical comparison between the number of common patterns according to the top-$i$ ranked selected patterns by $MCP_{sc}$ and $MCP_{s\ func}$ are given in Figures 2.23 and 2.24.

Table 2.11 : The number of obtained sequential patterns at each stage of the GSP algorithm using $MCP_{sc}$ and $MCP_{s\ func}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than the threshold value.

| Stage ID | $MCP_{sc}$ processing | | | $MCP_{s\ func}$ processing | | |
|---|---|---|---|---|---|---|
| | NAP | NKPF | NPT | NAP | NKPF | NPT |
| 1 | 61 | 61 | 20 | 61 | 10 | 10 |
| 2 | 3094 | 1348 | 26 | 1714 | 27 | 27 |
| 3 | 2383 | 2256 | 44 | 112 | 47 | 47 |
| 4 | 3903 | 3779 | 53 | 113 | 57 | 57 |
| 5 | 4735 | 4603 | 39 | 109 | 50 | 50 |
| 6 | 3348 | 3253 | 11 | 76 | 2 | 2 |
| 7 | 891 | 867 | 0 | 2 | 0 | 0 |
| 8 | 0 | 0 | 0 | - | - | - |

In this case, the number of longer patterns (*i.e.* with number of items $\geq 4$) is approximately the same (103 in case of $MCP_{sc}$ vs 109 in case of $MCP_{s\ func}$). However, the number of similar extracted patterns is around 70%, the differences resulting from the *coefficient function* from Equation (2.28) used in $MCP_{s\ func}$ formula. As in the case of *Support*, the anti-monotone property of $MCP_{s\ func}$ facilitates the amount of verified patterns at each stage in comparison to $MCP_{sc}$.

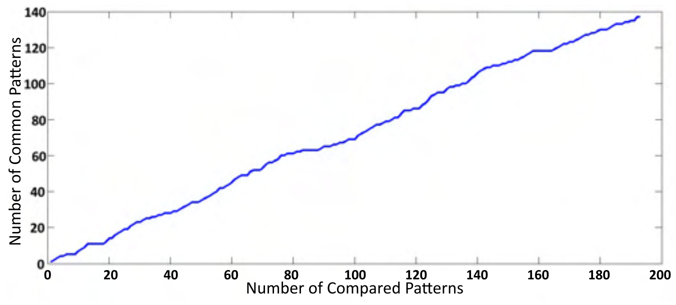


Figure 2.23 : The number of common patterns between $MCP_{sc}$ and $MCP_{s\ func}$ according to the first taken patterns arranged in a decreasing order of their IM.

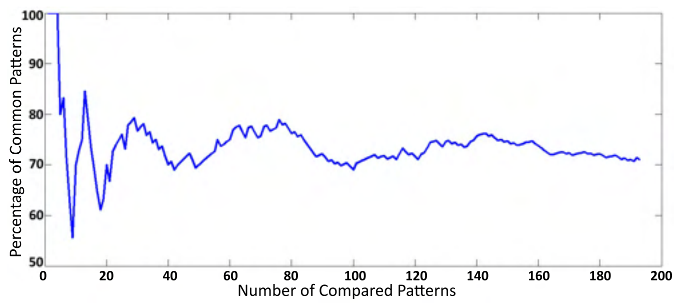


Figure 2.24 : The number of common patterns in percentage % between $MCP_{sc}$ and $MCP_{s\ func}$ according to the first taken patterns arranged in a decreasing order of their IM.

#### 2.5.6.3 $MCP_{s\ func}$ vs *Support* on *ClickStream* database

We have performed a simulation with the following parameters for $MCP_{s\ func}$: $\omega_t = 360$, *slope* $s = 1.05$, and $MCP_{s\ func\ thr} = 0.01$. 629 patterns were obtained. In order to compare it to *Support* processing, *i.e.* to obtain the same number of patterns, a value of $supp_{thr} = 12.85\%$ was set. The results are shown in Table 2.12. As it is expected, the *Support* measure is the one advantaging longer patterns in comparison to $MCP_{s\ func}$ (524 vs 480). The explanation is the same as for $MCP_{sc}$: *Support* does not take into account the time-distance between its itemsets. A graphical comparison between the number of common patterns according to the top-$i$ ranked patterns by $MCP_{sc}$ and *Support* are given in Figures 2.25 and 2.26. This number represents around $80\%-85\%$ from the total number of compared patterns, which is higher then for the $MCP_{sc}$ vs *Support* comparison (where we had a value of 70%). This is due to the anti-monotone property of the $MCP_{sc}$, making it "closer" in "properties" to *Support* measure.

Table 2.12 : The number of obtained sequential patterns at each stage of the GSP algorithm using classic *Support* processing and $MCP_{s\ func}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than the threshold value.

| Stage ID | $MCP_{s\ func}$ processing | | | *Support* processing | | |
|---|---|---|---|---|---|---|
| | NAP | NKPF | NPT | NAP | NKPF | NPT |
| 1 | 61 | 11 | 11 | 10 | 10 | 10 |
| 2 | 1859 | 41 | 41 | 155 | 28 | 28 |
| 3 | 197 | 96 | 96 | 122 | 67 | 67 |
| 4 | 332 | 161 | 161 | 184 | 98 | 98 |
| 5 | 397 | 148 | 148 | 209 | 127 | 127 |
| 6 | 289 | 142 | 142 | 239 | 155 | 155 |
| 7 | 260 | 30 | 30 | 281 | 128 | 128 |
| 8 | 25 | 0 | 0 | 187 | 14 | 14 |
| 9 | - | - | - | 14 | 2 | 2 |
| 10 | - | - | - | 2 | 0 | 0 |

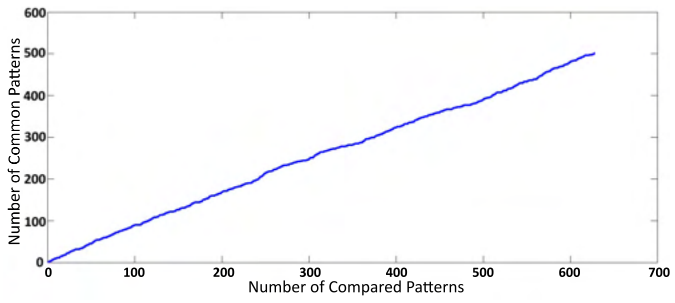


Figure 2.25 : The number of common patterns between $MCP_{s\ func}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM.

## 2.6 Conclusions

The temporal aspect in sequential rules and patterns makes an important topic in sequential data mining. Especially, the time closeness between itemsets of a sequential data could be important for the end-user, if he is interested in predicting the events (*i.e.* itemsets) which might appear in the forthcoming time-interval. We thus proposed Interestingness measures for sequential rules and patterns mining that could be used to build accurate forecasting models.

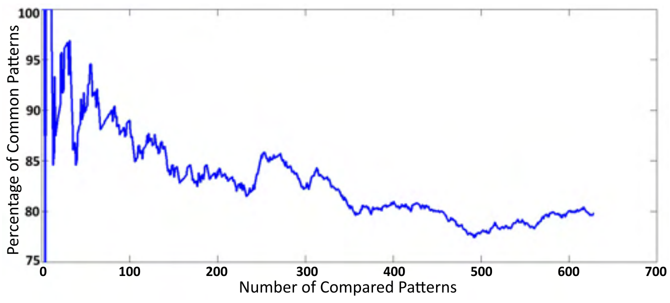


Figure 2.26 : The number of common patterns in percentage % between $MCP_{s\ func}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM.

In case of sequential rules selection, we have introduced the *Closeness Preference*, an interestingness measure that will favour the rules with close itemsets (within a user-predefined time-interval and with the help of a user preference function). This measure is used as a post-processing filter. It can thus be used alone or it can be used to complement traditional measures like the *Lift*. Some properties of this measure are discussed and we have shown that it satisfies 2 out of 3 important properties proposed by Piatetsky-Shapiro [Piatetsky-Shapiro 91]. A case study on web logs data shows that the Closeness Preference measure is helpful to find small and efficient set of simple sequential rules. We could conclude that we have met 3 important goals by using our measure for a prediction problem: (1) a higher *Accuracy* in comparison to other exiting algorithms; (2) simplicity of the algorithm by obtaining a greater performance with the *simple* rules compared to the *complex* ones; (3) an usage of the *time parameter* through a maximum time-interval $\omega_t$ and a time importance variation inside $\omega_t$ by using a sloping coefficient for the function 2.1.

In case of sequential patterns mining we proposed 2 derivatives of the $CP$ measure, *i.e.* $MCP_{sc}$ and $MCP_{s\ func}$, that also have as goal to advantage the shorter time-distance between the itemsets of a pattern. Both measures are used in pre-processing step of the algorithm, but only one of it fulfils the anti-monotone property. This fact results in an issue for the second measure which does not fulfil the apriori principle: the search space and patterns' check is large and time consuming. A comparison of the top-$i$ ranked extracted patterns using these measures and *Support* was presented using the *ClickStream* database. The results have shown that the number of common patterns between the 2 measures and *Support* is around $70\% - 85\%$ from the total number of compared patterns, where higher values being specific to the comparison between the measure satisfying the anti-monotone property, *i.e.* $MCP_{s\ func}$ and support. This is due to the fact that the anti-monotone property makes $MCP_{s\ func}$ "closer" in "properties" to *Support* measure. Another point is that both $MCP_{sc}$ and $MCP_{s\ func}$ tend to rank at the top shorter patterns. This is explained by the fact that longer patterns tend to have larger time-intervals between the itemsets, and this fact is disadvantaged by the proposed measures. Thus, the proposed pre-processing measures do advantage those sequential patterns which have shorter time-interval between its itemsets.

# Time Series

## 3.1 Introduction

A time-series represents a set of historical data measured sequentially through time. According to the way the data is recorded, we distinguish two types of series: *continuous*, where the observed data is continuously measured (*e.g.* seismogram records), and *discrete*, where the data is taken at a discrete set of time points (*e.g.* daily revenue of an on-line shop, or the temperature measured at hourly intervals). Of course, any continuous time-series can be sampled and transformed into a discretized one, where little information is lost if the sampling intervals are small enough. In our work, we are going to discuss about discrete time-series.

Time-series analysis has become one of the most important and broadly used branches of research applied to extract meaningful information and characteristics from the data. It has large applications such as economic planning, stock market analysis, process and quality control, signal processing, inventory studies. An application of time-series analysis is *forecasting* (or *prediction*).

Two important domains where the time-series forecasting is applicable are communications and finance. In case of communications, globally for a radio network, and particularly for a WiMAX network (Worldwide Interoperability for Microwave Access), predicting the future traffic level is mandatory in order to keep satisfactory quality of services, to characterize its performance and it is of significant interest in its control, design and management. In case of finance, particularly for foreign currency exchange rate or stock market volatility, its forecasting is of high interest in the financial world in order to increase gains and to have a better risk management. The main idea behind time-series analysis used in forecasting is that successive observations are *dependent* on each other, thus, one should consider the order in which the observations are collected and to find some similar patterns that will help in prediction. Thus, a *model* is applied on previously collected historical values, in order to estimate the future values of the time-series. In our work we discuss about WiMAX and EUR/USD currency exchange forecasting.

According to our knowledge and experience, there are several problems in time-series analysis. The first issue (*i1*) is finding a model that could be suitable to analyse and predict different types of time-series. For example, we suppose that the same model cannot be applied on a network traffic (where a trend line could be observed because of the increasing/decreasing number of subscribers) and on a Foreign Exchange market (where there are a lot of speculators influencing randomly the price between currencies). Our goal is to find such a model. The second issue (*i2*) consists in finding better parameters involved in a prediction model. For example, if we use a neural network to simulate a WiMAX traffic during a day, then we have to set the type of the network, number of layers, of neurons in the hidden layers, the number of training epochs, because different values would result in different approximation of the desired output series. These setting would be different from those used in simulation of the traffic during one week. The third issue (*i3*) consists in the fact that the effectiveness of a prediction model changes as soon as the behaviour of the time-series varies, *i.e.* the phenomena that generates a time-series change in the long run. If we suppose that the analysed WiMAX traffic experiences some increase in users, or installation of new base stations (radio receiver/transmitter), then a change in signal characteristic has to be expected. This implies a retraining of the prediction model, changing of its parameters, or finding an easier and better model instead for reproducing and forecasting the wanted time-series.

In the current work we propose a model to forecast future evolution of a time-series based on its historical information. We formulate our problem as follows: given some observed labels $y_1$, $y_2$, ..., $y_n$ of a time-series of length $n$, apply the proposed model, and predict the labels $y_{n+1}$, ..., $y_{n+k}$ of the future $k$ labels. Our model is based on time-series decomposition in wavelet domain (*Stationary Wavelet Transform* (SWT) using *Artificial Neural Networks* (ANN) and their optimization using *Genetic Algorithms* (GA). The originality of our proposed approach in comparison to other works is based on time-series decomposition in wavelet domain and the way we use the ANNs in the forecasting step (the configuration of inputs, of weights, the selection of data for each level of wavelet decomposition, the optimization with GAs). We compare existing models with the proposed one. As analysed data we used two time-series from different domains: WiMAX network traffic data and EUR/USD currency exchange. Our results present the performance of different forecasting models in case of different time-series (*i1*). Different settings of the model are also studied in order to see how the selection of parameters could improve its performance (*i2*). Finally, we see how a constant upgrading with the latest available information and retraining the model plays an important role in the effectiveness of the future estimation of the time-series (*i3*). In this way if the behaviour of the series changes, then we keep our model up-do-date by analysing each time newer parts of the series.

## 3.2 State of the Art

There are numerous existing models for time-series forecasting, which can be grouped into four categories [Rong 08]:

1. *Case-Based Reasoning model* (CBR): is a means for solving a new problem by using or adapting solution to old problems - its essence in analogy. The basic principle of CBR is that similar problems have similar solutions.

2. *Rule-Based Forecasting model* (RBF): is a set of rules that use the experts' domain knowledge and the characteristics of the data to produce a forecast from a combination of simple extrapolation methods. It consists of 99 rules that use 28 features of time-series, such as: causal forces, functional form, types of data, length of series, instability. Its application depends upon features of the time-series.

3. *Statistical model*: is a probability distribution model such as the single regressive model, exponential smoothing, autoregressive integrated moving average model.

4. *Soft Computing model*: is based on information processing in biological systems solving the problems by exploiting the tolerance for imprecision, uncertainty, partial truth, and approximation in order to obtain robustness and low solution cost.

Example of such models are neural networks and their amelioration or mixture with other methods, fuzzy logic, support vector machine.

Next, we present how different time-series were analysed and forecasted using the above models (problems *i1* and *i3* from our goals). A fuzzy CBR system [Kolodner 92] for weather prediction is proposed by [Riordan 02]. The knowledge about temporal features used by human forecasters is encoded in a fuzzy similarity measure, used later to locate the *k*-nearest neighbours from the historical database. The accuracy of 0 to 6 hour predictions was up to 85%. The prediction of bank lending decisions using CBR approach taking into considerations subjective factors also (*e.g.* Economic sentiment) is presented in [Teodoro 06]. The data from the Euro-system survey for Portugal was used. The results show that the system can forecast with quite a high precision (90%) the decision of economic agents. A CBR model for individual demand forecasting is proposed in [Rong 08]. The forecast process is divided into 3 stages: case representation, similarity search, and case forecasting and adjustment. A distance-based formula is presented, which compares time-series according to their slope and length. The empirical results show that this method is able to produce forecasts with a high degree of accuracy (a medium absolute error (MASE) of 0.470 in comparison to the moving average algorithm which obtained a MASE of 0.745). A CBR system to forecast the presence of oil slicks is shown in [Corchado 08]. Information as salinity, temperature, pressure, number and area of slicks, is also used. The percentage of correct predictions was up to 87%.

Adya and Armstrong [Adya 00] describe the utilization of a RBF approach. This technique was applied on 126 annual time-series from Makridakis time-series Competition. Mean absolute percentage error (MAPE) was used as a performance indicator. A value of 8.92 was obtained for 1 year prediction. A RBF model used to estimate the cost per megabyte of hard disk drives is shown in [Webb 03]. At the moment the paper was written, the results suggested a fall rate in price of about 48% per year. A comparison between RBF and neural networks, Box-Jenkins models, to be used under given conditions (markets, game theory) is presented in [Armstrong 06]. The results show that RBF models could be applied to cross-sectional data (*i.e.* observations of many objects at a given time) with a 10% error reduction in comparison to *ex ante* forecasting accuracy of alternative methods, and to time-series data with about 60% error reduction.

A prediction of applying least square method (LSM) and linear regression method (LRM) models is described in [Moungnoul 05]. For better performance, a combination between these two is used. LSM is a prediction model using trend line by least square method, which has the minimum summary of square of difference between trend line's data and collected data. LRM is a method for defining the relation of two variables. The model was applied on GSM traffic data. Errors around 25% were obtained. A statistical model used to forecast monthly large wildfire events in Western United States is presented in [Preisler 06]. The method is based on logistic regression techniques with spline functions to accommodate non-linear relationships between fire-danger predictors and probability of large fire events. Simulations on datasets from federal land management agency fire reports showed that the model is able to miss only 2% of the large fire events during the month of August. The autoregressive integrated moving average (ARIMA) model was used to forecast the Indian sugar-cane production in [Mandal 06]. Standard errors between 0.116 and 0.134 were obtained. A study of predicting oil palm price of Thailand using ARIMA model is shown in [abd T. Nochai 06]. The obtained MAPE values were up to 5.27%. Models based on the ensemble multiple linear regression (EMR) and projection pursuit regression (PPR) techniques were developed to forecast the south-west monsoon rainfall over India [Rajeevan 07]. A root mean square error between 4.56 and 6.75% was obtained.

Regarding the soft-computing methods, in [Tan 93] the advantages of the artificial neural networks over traditional rule-based systems are shown in case of a financial trading system to improve trading profitability (the authors presents only the concept, no real data simulations are performed). A comparison between ARIMA and neural networks modelling is presented in [Mohammed 04, Feng 05]. [Mohammed 04] applied the models on a Financial Balance Sheet's data of a commercial bank in Egypt, obtaining an accuracy higher with about 10% using the neural networks. While [Feng 05] made simulations on four wireless network traffic traces obtaining values for $MSE$ with 8% and for *Normalized* $MSE$ ($NMSE$) with 10% higher in case of ANN in comparison to ARIMA approach. It was also shown that linear models reach their limitation with non-linearity in the data, while ANNs are successful in handling non-linear problem optimization and prediction. This is due to several distinguishing features of ANNs, making them valuable and attractive for a forecasting task. First of all, they can be treated as multivariate non-linear non-parametric statistical methods [White 89, Ripley 93, Zhang 98]. After that, ANNs can generalize, are non-linear, and are universal function approximators [Zhang 98]. The fact that neural networks produce better results than a standard statistical time-series predictor, has been demonstrated in [Refenes 93, Abrahart 98, Ibarra-Berastegi 07]. [Abrahart 98] provides a forecasting benchmark for river flow prediction reaching a *mean absolute error* ($MAE$) smaller with about 19% for the ANN in comparison to ARMA. [Ibarra-Berastegi 07] presents a chemical modelling problem, showing that the neural networks were able to explain 92% of the overall variability of the data in comparison to the *multiple linear regression* (MLR) model, where a value of 0.72% of the variability was expressed.

When the time-series is non-stationary, it is very difficult to identify a proper global model [Zhang 01]. To overcome this problem, an efficient way is to use the wavelet decomposition technique in the preprocessing step, because it provides a useful decomposition of time-series, in terms of both time and frequency, permitting to effectively diagnose the main component and to extract abstract local information from the time-series. Thus, Wavelet Transform (WT) has been frequently used for time-series analysis and forecasting in recent years [Wang 02, Papagiannaki 05, Feng 05]. Mitra [Mitra 06] presents in his work the capability of a technique using wavelet multi-resolution and neural networks to forecast the daily spot foreign exchange rates of major internationally traded currencies. The original exchange rate series to be modelled is first decomposed into various frequency resolution related components using wavelets. Next neural networks are applied for modelling components of the decomposed series resulting in a high performance prediction with a $MSE$ value as low as 0.001. A prediction technique for one day ahead energy prices using WT is described in [Nguyen 08]. The results demonstrate that the use of wavelets as a pre-processing procedure of forecasting data improves the performance of prediction models, the authors obtaining an improvement of $MSE$ up to 20%.

One of the problems in neural networks consists in their design configuration (size, weights). This is why the optimization of Artificial Neural Networks using Genetic Algorithms applied in forecasting have been proposed in many papers [Ojeda 95, Yu 07, Fiszelew 07] (problem (*i2*) from our goals). The simultaneous optimization of the network architecture and the training weights is presented in [Zhang 93]. The simulation results on two benchmark problems of different complexity (*majority* and *parity*) have suggested that the proposed method is able to find minimal size network. In [Ojeda 95] a method of determining the optimal size of the hidden layer and the number of connections between the layers is presented, which is used for processing ECG signals. In [Ileană 04] an approach using genetic computing is given that is used for establishment of the optimum number of layers and the number of neurons on a layer, for a given problem. An approach using GA algorithm to select the optimal architecture for ANN model in hot rolling process is shown in [Son 04]. A genetic algorithm-based ANN model for the turning process in manufacturing industry is presented in [Venkatesan 09].

Concluding, we can state that a better performance in forecasting is obtained using soft computing methods. These models were used in our research also (*i.e.* artificial neural networks which could be optimized using genetic algorithms, and wavelet transform of the time-series).

## 3.3 Theoretical Fundamentals

In the current section we present theoretical fundamentals related to the proposed prediction approach. We present basic aspects about *wavelets transform*, *neural networks*, and *genetic algorithms*.

### 3.3.1 Wavelets

#### 3.3.1.1 Introduction

Wavelets are mathematical functions that cut up a signal into several frequency components in order to produce a good local representation of a signal in both time and frequency domain. If we look at a series with a large *window*, we would notice gross features. Similarly, if we look at a series with a small *window*, we would notice small features [Graps 95]. The goal of the wavelet transform is to see both *the forest and the trees* [Grosse 04].

The *wavelet analysis* process consists in using a wavelet prototype function, called an *analysing wavelet* or *mother wavelet* on a time-series, *i.e.* it is a convolution of a series $s(t)$ with a set of functions generated from the mother wavelet. There are two types of analysis: *temporal* and *frequency*. The temporal analysis is achieved with the translations of the prototype wavelet, while the frequency analysis is realised with a dilations of the same wavelet.

The wavelet transforms are broadly divided into three classes: *continuous*, *discrete*, and *multi-resolution-based*.

#### 3.3.1.2 Continuous Wavelet Transforms (CoWT)

In continuous wavelet transforms, a given time-series $s(t)$ of finite energy is projected on a continuous family of frequency bands. The series can be represented on a frequency band

$[f,\ 2f]$ for all positive frequencies $f > 0$. Next, the original $s(t)$ can be reconstructed by a suitable integration over all the resulting frequency components.

The most common approach of building the wavelets is the one proposed by Daubechies [Daubechies 92], where we have 2 orthonormal *parent wavelets* (*i.e.* they are orthogonal and normalized, meaning that their inner product $\langle f, g \rangle$ is zero): the *scaling function* $\phi$ (*father wavelet*) and the *analysing function* $\psi$ (*mother wavelet*)[1] . For a function $\psi(t)$, in order to be *admissible* as a *wavelet*, it should have the average value zero and be localized in time and frequency space [Farge 92]. Thus, the complex-valued function $\psi$ should satisfy the following conditions:

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty \tag{3.3}$$

$$c_\psi = 2\pi \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty \tag{3.4}$$

where $\Psi$ represents the Fourier transform of $\psi$, and $c_\psi$ are the wavelet coefficients which are used later in the reconstruction of the initial time-series $s(t)$. The Equation (3.3) requires the function $\psi$ to have finite energy, while Equation (3.4) is the admissibility condition, stating that if $\Psi(\omega)$ is smooth, then $\Psi(0) = 0$.

The different wavelets in the basis are generated through translations of the father wavelet $\phi$ and dilations and translations of the mother wavelet $\psi$. The wavelet transform of a real series $s(t)$ is defined as:

$$S(b, a) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi' \left( \frac{t-b}{a} \right) s(t) dt \tag{3.5}$$

$\psi'$ is the complex conjugate of $\psi$. The pair $(a,\ b)$ defines a point in the right half-plane $\mathbb{R}_+ \times \mathbb{R}$. The parameter $b$ corresponds to the time shift, while $a$ to the scale of the

[1] Orthonormal bases of wavelets generated by one function were constructed for different function spaces. However, Mallat [Mallat 89] unified the construction of these bases for $L^2(\mathbb{R})$ using the multiresolution approximation (MRA) defined as a sequence of closed subspaces $(V_m)_{m\in\mathbb{Z}}$ of $L^2(\mathbb{R})$ [Goodman 93]. $(V_m)_{m\in\mathbb{Z}}$ has to satisfy the next properties:

1. $V_m \subset V_{m+1}, \qquad m \in \mathbb{Z}$
2. $\bigcup_{m\in\mathbb{Z}} V_m$ is dense in $L^2(\mathbb{R})$ and $\bigcap_{m\in\mathbb{Z}} V_m = \{0\}$
3. $g \in V_m \leftrightarrow D_2 g \in V_{m+1}, \qquad m \in \mathbb{Z}$, where $D_A g(t) := g(at), t \in \mathbb{R}$, for any positive number $a$
4. $g \in V_m \leftrightarrow T_{2^{-m}n} f \in V_m$, $(m, n) \in \mathbb{Z}^2$, where $T_\tau(t) := g(t - \tau)$, $x \in \mathbb{R}$, for $\forall \tau \in \mathbb{R}$
5. there exists an isomorphism $I$ from $V_0$ onto $I^2(\mathbb{Z})$ which commutes with the action of $\mathbb{Z}$

For the MRA, it was shown by Mallat [Mallat 89] that there exists the *father wavelet* $\phi \in V_0$ such that $(T_n\phi)_{n\in\mathbb{Z}}$ is an orthonormal basis of $V_0$. If

$$\phi_{m,n}(t) := \sqrt{2^m}\phi(2^m t - n), \qquad (m, n) \in \mathbb{Z}^2 \tag{3.1}$$

then for $\forall m \in \mathbb{Z}$, $(\phi_{m,n})_{n\in\mathbb{Z}}$ is an orthonormal basis of $V_m$. We suppose $W_m$ as being the orthogonal complement of $V_m$ in $V_{m+1}$. Then, the *mother wavelet* $\psi \in W_0$ exists, *s.t.* $(T_n\phi)_{n\in\mathbb{Z}}$ is an orthonormal basis of $W_0$, and if:

$$\psi_{m,n}(t) := \sqrt{2^m}\psi(2^m t - n), \qquad (m, n) \in \mathbb{Z}^2 \tag{3.2}$$

then $(\psi_{m,n})_{n\in\mathbb{Z}}$ is a complete orthonormal set in $W_m$.

analysing wavelet. Thus, when the scaling parameter $a$ is varied from high to low values, then the wavelet function $\psi'\left(\frac{t-b}{a}\right)$ will be compressed, *i.e.* the corresponding wavelet transform changes from low-frequency to high-frequency signal structures. A shifting by $b$ and rescaling by $a$ could be written as:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{3.6}$$

In this case, the Equation (3.5) can be rewritten as:

$$S(b,a) = \int_{-\infty}^{\infty} \psi'_{a,b}(t)s(t)dt \tag{3.7}$$

With the above relations, the original signal $s(t)$ can be obtained from $S(b,a)$ – its wavelet transform, using the following equation:

$$s(t) = \frac{1}{c_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(b,a)\psi_{a,b}(t)\frac{da\ db}{a^2} \tag{3.8}$$

where $c_\psi$ are obtained from the Equation (3.4).

The most famous pair of father and mother wavelets is the *Daubechies 4 tap wavelet* (Figure 3.1).

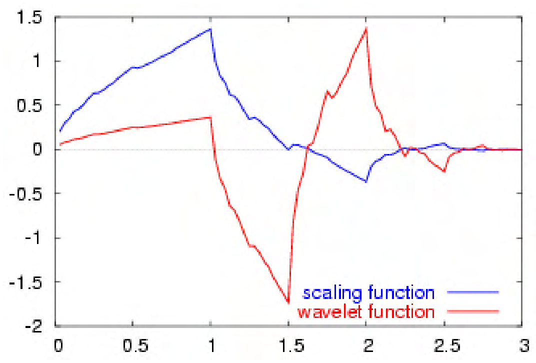


Figure 3.1 : Daubechies 4-*tap* (*db*2) wavelet. The scaling function is $\phi$ (father wavelet), while the wavelet function is $\psi$ (mother wavelet).

### 3.3.1.3 Discrete Wavelet Transform (DWT)

The CoWT has an issue of redundancy with digital computers, because the parameters $(a,b)$ take continuous values, and thus, its computation may consume significant amount of time and resources, depending on the resolution required. This is the reason why usually we are more concerned with discretely sampled, rather than continuous functions [Abramovich 00].

The DWT analyses a discrete signal $s[n]$, taking a series of $N$ samples and producing $N$ new values called wavelet coefficients. There are 2 types of coefficients: *approximation* and *detail*. If we consider the series $s[n]$ and the wavelet and scale functions $\psi_{j_0,k}[n]$ and $\phi_{j,k}[n]$ being discrete defined in $[0, M-1]$ (*i.e.* totally $M$ points), then the *approximation* coefficients are given by:

$$W_\phi[j_0, k] = \frac{1}{\sqrt{M}} \sum_n s[n]\phi_{j_0,k}[n] \tag{3.9}$$

and the *detail* coefficients by:

$$W_\psi[j, k] = \frac{1}{\sqrt{M}} \sum_n s[n]\psi_{j,k}[n]; \qquad j \geq j_0 \tag{3.10}$$

The initial approximated signal $s[n]$ will be given in this case by:

$$s[n] = \frac{1}{\sqrt{M}} \sum_k W_\phi[j_0, k]\phi_{j_0,k}[n] + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_k W_\psi[j, k]\psi_{j,k}[n] \tag{3.11}$$

**Multi-resolution Analysis :**

In case of the CoWT, we compute the wavelet transform by changing the scale of the analysis window, shifting the window in time, multiplying it by the signal, and integrating over all times. In case of the DWT however, filters of different cut-off frequencies are used to analyse the signal at different scales. Thus, the DWT is formed around a set of filtering operations in order to determine the coefficients from Equations (3.9) and (3.10). These coefficients could be written in the following form:

$$W_\phi[j, k] = h_\phi[-n] * W_\phi[j+1, n]|_{n=2k, k\geq 0} \tag{3.12}$$

$$W_\psi[j, k] = h_\psi[-n] * W_\psi[j+1, n]|_{n=2k, k\geq 0} \tag{3.13}$$

where $*$ is the convolution operation, $h_\phi$ is the high-pass filter, and $h_\psi$ is the low-pass filter respectively.

This way of obtaining a DWT is referred to as the *Mallat's algorithm* in the literature. The original signal is first passed through two complementary filters, low-pass and high-pass (as illustrated in Figure 3.2) after which is decomposed into two components, sub-sampled by 2, a low-frequency content and a high-frequency content of the same size. The low-frequency content of the signal (*i.e.*the *approximation*) gives a signal its basic identity, the high-frequency content on the other hand (*i.e. detail*) presents the finer details of data. Due to successive sub-sampling by 2, the length of the signal must be a power of 2, or at least a multiple of power of 2, and it determines the number of levels the signal can be decomposed into.

For a multilevel decomposition, the approximation series after the first filtering is further decomposed to obtain the approximation and detail series at the next level of decomposition [Mitra 06]. This multilevel decomposition is iterated until the predetermined or desired decomposition level is attained. In this way, we obtain, using a sequence of low-pass and high-pass filters, the wavelet decomposition of a signal at different frequency level. At the end of an $r^{th}$ level decomposition, we get the following decomposition of the original series:

- $a(r)$: approximation at the coarsest level $r^{th}$ of decomposition

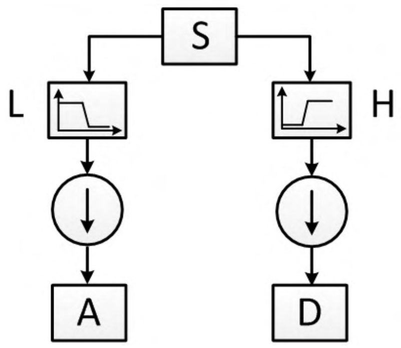


Figure 3.2 : Low-pass and high-pass filtering of a signal. $S$ denotes the original signal, $A$ denotes the approximation and $D$, the detail; $\downarrow$ represents the sub-sampling of the signal.

- $d(r)$: details at the coarsest level $r^{th}$ of decomposition
- $d(r-1)$: details at the $(r-1)^{th}$ level of decomposition
- $d(1)$: details at the finest level of decomposition

A two-level wavelet decomposition is illustrated in Figure 3.3, where $S$ denotes the original signal, and $L$ and $H$ denote the low-pass and high-pass filters respectively.

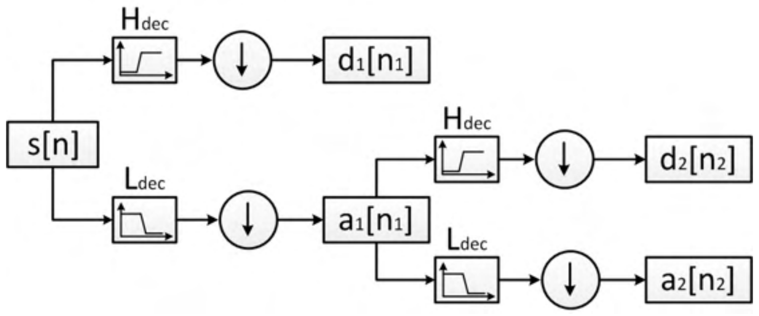


Figure 3.3 : A two-level wavelet decomposition process; $\downarrow$ represents the down-sampling of the signal.

In order to assemble back those components into the original signal, we use the *reconstruction*, or *synthesis* using the relation from the Equation (3.11). Where wavelet analysis involves filtering and down-sampling, the wavelet reconstruction process consists of up-sampling and filtering [MathWorks 12c]. Up-sampling is the process of lengthening a signal component by inserting zeros between samples (Figure 3.4). An example of 2 level signal reconstruction is presented in Figure 3.5. Note, that the reconstruction filters have their coefficients opposite to the decomposition filters. An example of the *Daubechies 4* level filters is presented in Figure 3.6.

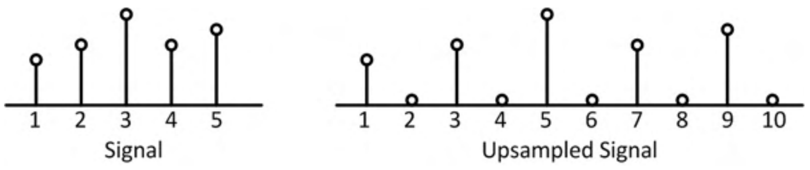


Figure 3.4 : Up-sampling of a signal [MathWorks 12c].

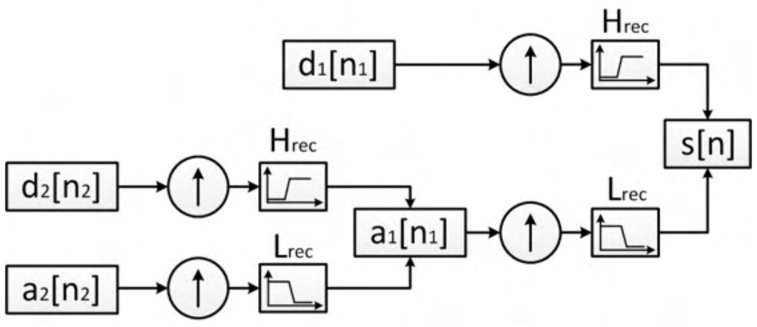


Figure 3.5 : A two-level wavelet reconstruction process; ↑ represents the up-sampling of the signal.

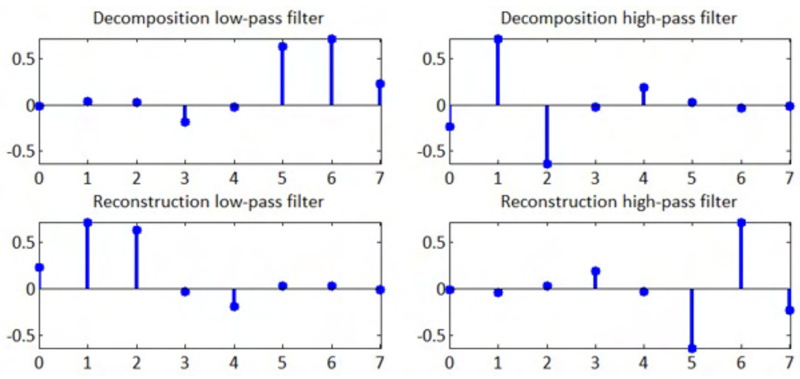


Figure 3.6 : Decomposition and Reconstruction filters for the *db*4 wavelet 8-*tap*).

The disadvantage of the Mallat's algorithm is the decreasing of the length of the coefficient sequences with the increasing of the iteration index due to the use of decimators. Because of the decimation step in the DWT analysis, it makes the standard DWT time shift-variant. Another way to implement the MRA is to use the *à trous* algorithm, proposed by Shensa [Shensa 92], which corresponds to the computation of the Stationary Wavelet Transform (SWT). In this case, the use of decimators is avoided, but different low-pass and high-pass filters are used at each iteration. At each level the filters are up-sampled versions of the corresponding filters from the previous level. Thus, the difference between

SWT and DWT is that the signal is not sub-sampled in the SWT case, but the filters are up-sampled at each level. In the SWT each set of coefficients contains the same number of samples as the input signal. For example, for a decomposition of $N$ levels, there is a redundancy of $2N$, because no sub-sampling is performed. This redundant representation makes it shift-invariant, becoming suitable for edge detection, de-noising.

#### 3.3.1.4 Wavelet types

One of the problems in the wavelet analysis consists in choosing the wavelet function $\psi$. There are different types of wavelets which are more or less suitable for a given problem [Farge 92, Torrence 98]:

- *Orthogonal*, *non-orthogonal*, or *biorthogonal*. Orthogonal wavelets are obtained when the scaling function $\phi(t)$ and analysis function $\psi(t)$ are orthonormal. Using the orthogonal analysis, the number of convolutions at each scale is proportional to the width of the wavelet basis at the given scale. This is used for signal processing, because it gives the most compact representation of the signal. However, in case of time-series analysis, an aperiodic shift in the time-series produces a different wavelet spectrum. The non-orthogonal analysis instead, is redundant at large scales, and are useful for time-series analysis, where continuous and smooth variations in wavelet amplitude are expected. In case of the biorthogonal wavelets, the orthogonal condition is relaxed to biorthogonal conditions, there are two multiresolution analyses [Strang 96]. For the biorthogonal wavelet filter, the low-pass and high-pass filters do not have the same length. The low-pass is always symmetric, while the high-pass filter can be either symmetric or a-symmetric.

- *Complex* or *real*. The issues of the standard DWT for analysis has three main disadvantages [Shukla 03]: shift-sensitivity, poor directionality, and lack of phase information. These facts restrict the scope of DWT for certain signal and image processing (edge detection, motion estimation). Stationary Wavelet Transform (SWT) reduce only the first disadvantage of shift-sensitivity, but brings very high redundancy and computation. In order to overcome these issues, the complex wavelet transforms are used. They use complex-valued filtering that decomposes the real/complex time-series into real and imaginary parts in transform domain. These parts return information about amplitude and phase, being suited for oscillatory behaviours, *i.e.* non-stationary series. While a real wavelet function gives information of a single component and is better applied to isolate peaks or discontinuities in a time-series.

- *Width*. It is defined as the $e$-folding time of the wavelet amplitude. A narrow function (in time) has good time resolution, but poor frequency one. From the other part, the wider function has poor time resolution, but good frequency resolution.

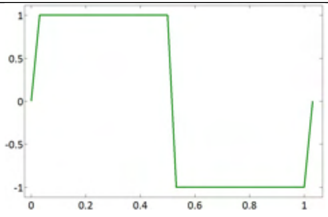
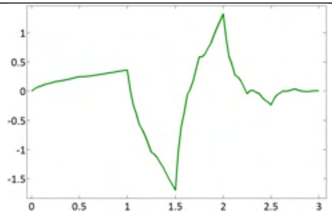
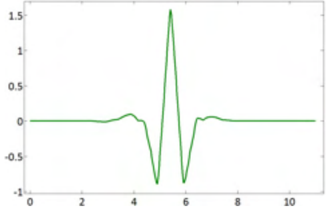
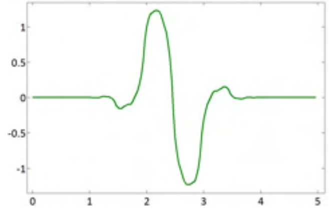
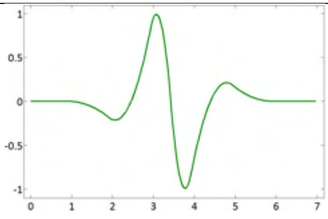
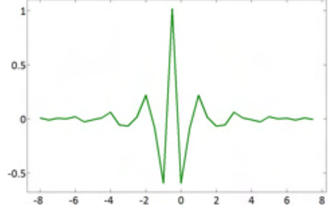
- *Shape*. A wavelet function has as goal to reflect the features of the time-series. Thus, for a series with spikes or sharp jumps, a boxcar-like function should be used (*e.g.* Haar wavelet). For a smooth varying time-series, a smooth wavelet function has to be applied (*e.g.* damped cosine wavelet).

- *Vanishing Moments.* An important parameter of a wavelet function is its number of *vanishing moments*, which gives the order of the wavelet and is a necessary condition for the smoothness of the wavelet function [Daubechies 92]. Thus, if the wavelet can generate polynomials up to degree $p-1$, then it can have $p$ vanishing moments. The *vanishing* aspect means that the wavelet coefficients are zero for those polynomials. More vanishing moments result in better approximation of the time-series.

There are many wavelet families which depend on the choice of the mother wavelet function. At the same time, the choice of the mother wavelet depends on a given application.

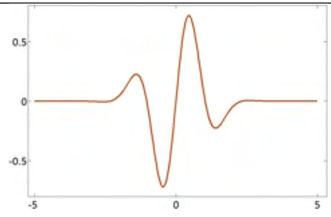
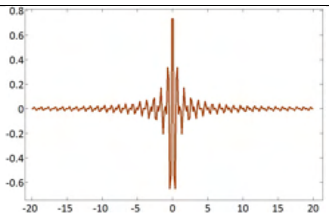
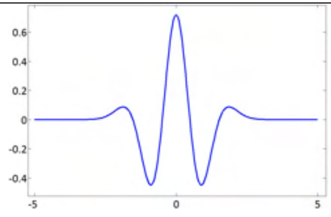
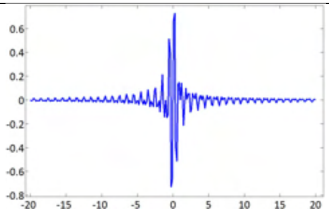
Some examples of wavelet shapes are presented in Table 3.1 for real functions, and in Table 3.2 for complex wavelet functions.

Table 3.1 : Examples of real wavelet functions. The number represents the order of the given wavelet. In case of the *biorthogonal* and *reversed biorthogonal* wavelets, the first number is the number of vanishing moments in the synthesis wavelet, while the second one is the number of vanishing moments in the analysis wavelet.

| WT Name | $\psi$ plot | WT Name | $\psi$ plot |
|---|---|---|---|
| Haar |  | Symlet 2 |  |
| Coiflet 2 |  | Biorthogonal 1.3 |  |
| Reversed Biorthogonal 3.3 |  | Meyer |  |

The *Haar* wavelet is the first and the simplest orthonormal (*i.e.* orthogonal and normal) wavelet. Being a step function, it is an asymmetric wavelet. The *Daubechies* family wavelets are asymmetric and orthogonal and they vary according to their number of vanishing moments. *Biorthogonal* and *Reversed Biorthogonal* family wavelets are symmetric and exhibit the property of linear phase, needed in signal and image reconstruction. The *Symlets* and *Coiflets* families are *near* symmetric (*i.e.* almost symmetric) and orthogonal.

Table 3.2 : Examples of complex wavelet functions. The number represents the order of the given wavelet.

| WT Name | $\psi$ plot | WT Name | $\psi$ plot |
|---|---|---|---|
| Cgau 3 Real |  | Shan 3 Real |  |
| Cgau 3 Imag |  | Shan 3 Imag |  |

They are modified versions of *Daubechies* wavelets. *Meyer* wavelets are symmetric and orthogonal.

The *Complex Gaussian* and *Complex Shannon* wavelets are families of complex symmetric wavelets which are neither orthogonal, nor biorthogonal. They are built from the complex Gaussian function and from the frequency B-spline wavelets respectively [Crowley 96].

## 3.3.2 Neural Networks

### 3.3.2.1 Introduction

The *Artificial Neural Network* (ANN) is an important paradigm for the cognitive machine [Ojeda 95]. The goal of the ANN is to model complex systems allowing it to be used in solving difficult tasks such as pattern recognition, classification, prediction [Gorman 88, Sbirrazzuoli 97, Beasley 97]. The ANN is an emulation of the biological neural system with the principal characteristic the distributed information or knowledge in connection or weight between simple elements, the *artificial neuron*. This element basically consists of *input signals* which are multiplied by *weights* (*i.e.* strengths of the respective signals) and summed up followed by a transfer function in order to compute the *output* of the neuron. This artificial neuron is associated with other similar elements and are interconnected in the network, the *Neural Network*. There are different types of interconnections between neurons, *i.e. architectures* of the ANN, requiring different types of algorithms to find its weights, but despite to be an apparently complex system, a neural network is relatively simple.

A neuron with a single R-element input vector is shown in Figure 3.7. The individual element inputs $p_1, p_2, ..., p_R$ are multiplied by weights $w_{1,1}, w_{1,2}, ..., w_{1,R}$, and the weighted values are passed to the summator. Their sum is $Wp$, the dot product of the (single row) matrix $W$ and the vector $p$. The neuron has a bias $b$, which is summed with the weighted inputs to form the net input $n$. This sum, $n$, is the argument of the transfer function $f$ in order to compute the output $a$:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b \tag{3.14}$$

$$a = f(Wp + b) \tag{3.15}$$

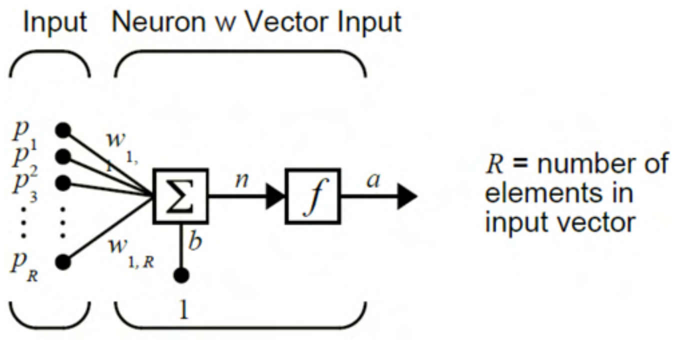


Figure 3.7 : Single Neuron Model [MathWorks 12b].

As in most of the systems, the Neural Networks have their own advantages and disadvantages [sit 12a]:

1. Advantages:
    - A neural network can perform tasks that a linear program cannot,
    - When an element of the neural network fails, it can continue without any problem by their parallel nature,
    - A neural network learns and does not need to be programmed,
    - It can be implemented in any application,
    - It can be implemented without any problem.
2. Disadvantages:
    - The neural network needs training to operate,
    - The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated,
    - Requires high processing time for large neural networks.

A neural network is characterized by three things [Marchiori 08]:

1. Its *architecture*: the pattern of nodes and connections between them (Subsection 3.3.2.2),
2. Its *learning algorithm*, or *training method*: the method for determining the weights of the connections (Subsection 3.3.2.3),
3. Its *transfer function*: the function that produces an output based on the input values received by a node (Subsection 3.3.2.4),

#### 3.3.2.2 Network Architecture

There are three different classes of network architectures:

1. Single-layer feed-forward
2. Multi-layer feed-forward
3. Recurrent

**Single-Layer Feed-Forward Networks :**
Two or more of the neurons shown before can be combined in a layer. A one layer network with $R$ input elements and $S$ neurons is shown in Figure 3.8.

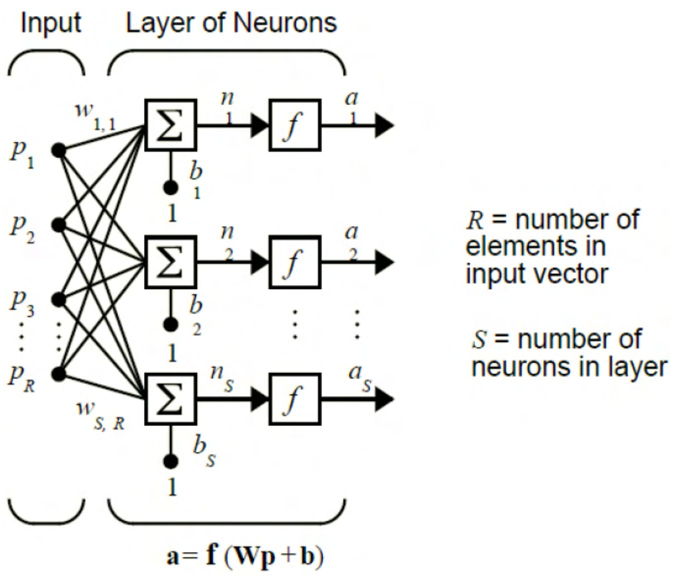

Input
Layer of Neurons
$w_{1,1}$
$p_1$
$p_2$
$p_3$
$p_R$
$w_{S,R}$
$\Sigma$
$n_1$
$n_2$
$n_S$
$f$
$a_1$
$a_2$
$a_S$
$b_1$
$b_2$
$b_S$
1
$R$ = number of elements in input vector
$S$ = number of neurons in layer
$\mathbf{a} = \mathbf{f}(\mathbf{Wp}+\mathbf{b})$

Figure 3.8 : Single Layer Neural Network [MathWorks 12b].

In this network, each element of the input vector $p$ is connected to each neuron input through the weight matrix $W$. The $i^{th}$ neuron has a summer that gathers its weighted inputs and bias to form its own scalar output $n(i)$. The various $n(i)$ taken together form an $S$-element net input vector $n$. Finally, the neuron layer outputs form a column vector $a$.

**Multi-Layer Feed-Forward Networks :**
A network can have several layers [MathWorks 12b]. Each layer has a weight matrix $W$, a bias vector $b$, and an output vector $a$. A 3 layer network, is shown in Figure 3.9.

**Recurrent Networks :**
In a recurrent network, the weight matrix for each layer contains input weights from all the neurons in the network, not just neurons from the previous layer. A fully network is one where every neuron receives input from all other neurons in the system. Such a network cannot be easily arranged in layers. A simple recurrent network is shown in Figure 3.10.

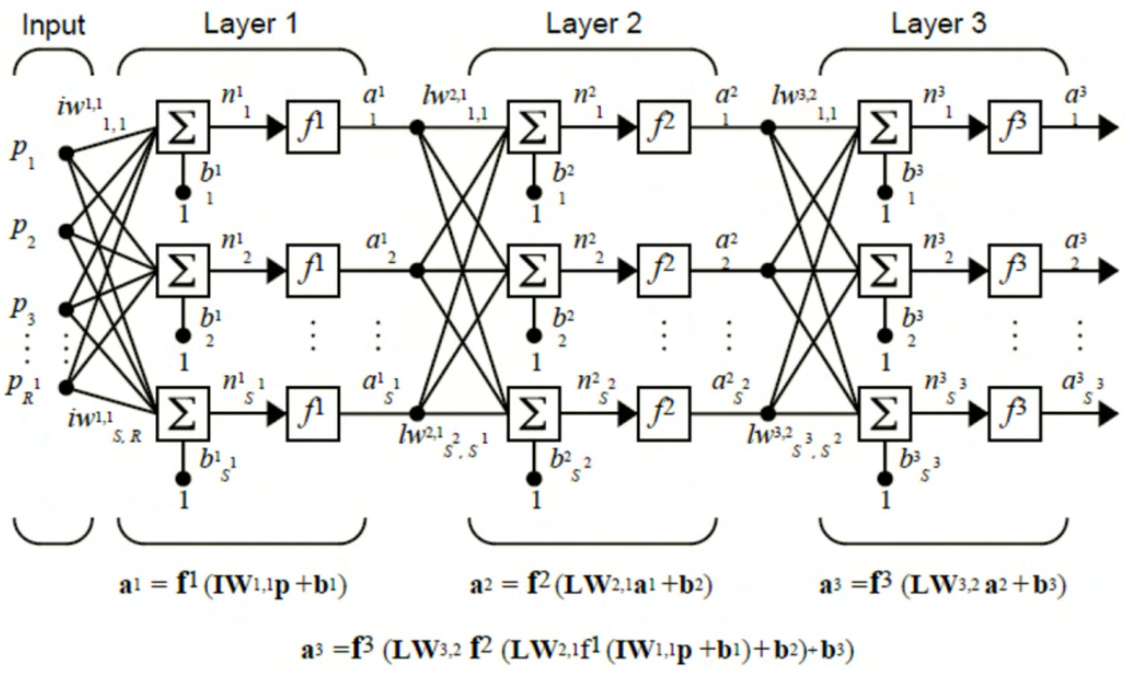

Input
Layer 1
Layer 2
Layer 3
$a^1 = f^1(IW^{1,1}p + b^1)$
$a^2 = f^2(LW^{2,1}a^1 + b^2)$
$a^3 = f^3(LW^{3,2}a^2 + b^3)$
$a^3 = f^3(LW^{3,2} f^2(LW^{2,1} f^1(IW^{1,1}p + b^1) + b^2) + b^3)$

Figure 3.9 : Three Layer Neural Network [MathWorks 12b].

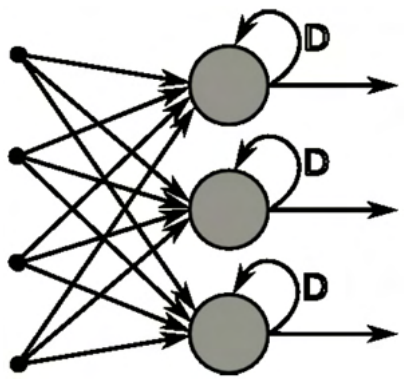


Figure 3.10 : Simple Recurrent Neural Network.

#### 3.3.2.3 Learning and Training the Neural Network

When adjusting the weights of an ANN we are able to obtain the desired output for a given inputs. However, when there are many neurons, it would be complicated to find the necessary weight values by hand. This is why various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using *a priori* knowledge. Another way, is to *train* the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule [Singh 09b, Krishna 11].

The training procedure dictates a global algorithm that affects all the weights and biases of a given network, such as *Bayesian regulation back-propagation*, *gradient descent back-propagation*, *Levenberg-Marquardt back-propagation*, *unsupervised random order weight/bias training*. Training the network is time consuming. It usually learns after several epochs, depending on how large the network is. Thus, a large network requires more training time compared to a smaller one. The main idea in training is to minimize a performance function (the global error) while updating the weights and biases of the network. Basically, the network is trained for several epochs and stopped after reaching the maximum epoch. Another way to stop the training process is to check if the differences between network output and known outcome is less than a specified value [MathWorks 12b, Thalatam 10].

The learning functions (*e.g. Hebb weight learning rule*, *Kohonen weight learning function*, *batch self-organizing map weight learning function*) can be applied to individual weights and biases within a network [MathWorks 12b]. We can categorize the learning as follows:

- *Supervised learning*: in which the network is trained by providing it with input and matching output patterns;
- *Unsupervised learning*: in which the output unit is trained to respond to clusters of pattern within the inputs;
- *Reinforcement learning*: it can be considered as an intermediate of the supervised and unsupervised processes. Some action on the environment is performed and a feedback response is received, after which the learning system grades its action as *good (rewarding)*, or *bad (punishable)* based on the environmental response, and adjusts its parameters accordingly.

#### 3.3.2.4 Transfer Functions of the Neural Networks

There are two functions that determine the way signals are processed by neurons: *activation* and *output* functions. The *activation function* acts on the input vector and influences the total signal a neuron receives. The *output function*, operates on a scalar activation, influences the scalar output. The composition of the activation and the output function is called the *transfer function*. For some transfer functions there is no natural division between activation and output functions.

There are many types of transfer functions, but only three of them are presented next.

**Hard-Limit Transfer Function :**

The *hard-limit transfer function*, shown in Figure 3.11, is given by the following equation:

$$a = f(n) = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n \geq 0 \end{cases} \tag{3.16}$$

It limits the output of the neuron to either 0, if the net input argument $n$ is less than 0; or 1, if $n$ is greater than or equal to 0. There is also a *symmetric hard-limit transfer function* that forces a neuron to output a 1 if its net input reaches a threshold, otherwise it outputs $-1$.

**Linear Transfer Function :**

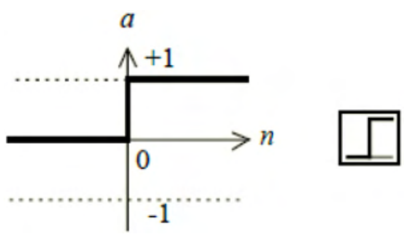


Figure 3.11 : Hard-Limit transfer function.

The linear transfer function from Figure 3.12 calculates the neuron's output by simply returning the value passed to it:

$$a = f(n) = n \tag{3.17}$$

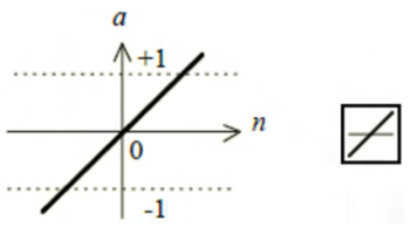


Figure 3.12 : Linear transfer function.

**Log-Sigmoid Transfer Function :**

The sigmoid transfer function shown in Figure 3.13, takes the input, which may have any value between plus and minus infinity:

$$a = f(n) = \frac{1}{1 + \exp(-n)} \tag{3.18}$$

It generates a continuous valued output between 0 and 1. Is defined as a strictly increasing function that exhibits smoothness and asymptotic properties, being also differentiable.

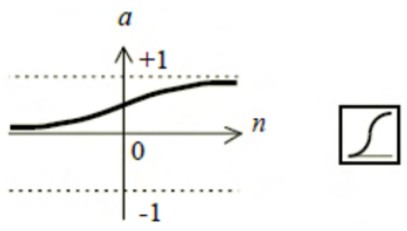


Figure 3.13 : Log-Sigmoid transfer function.

### 3.3.3 Genetic Algorithms

#### 3.3.3.1 Introduction

*Genetic Algorithms* (*GA*s), introduced by Holland in years 1970 (with a revised work in [Holland 92]), are search algorithms based on the mechanics of natural selection and genetics as observed in the biological world providing efficient, effective techniques for optimization and machine learning applications [Goldberg 88, Man 96, Paulinas 07, Musliu 12]. The natural selection is based on the principle that strong species have greater opportunity to pass their genes to future generations, while weaker species are faced with extinction by natural selection. Sometimes, several random changes could occur in genes, which will result in the appearance of new future species in case if these genes bring supplementary advantages for survival. Through time, species with more adapted genes to the surrounding environment become dominant in the population. Thus, a GA performs similar actions as in real world by repeatedly modifying a population of individual solutions. At each step, it selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution for survival.

In GA terminology, the *population* members are *strings* or *chromosomes*, which are binary representations of solution vectors $x_i$ ($x_i \in P$, $1 \leq i \leq N$, where $P$ is the population of size $N$). These chromosomes are made of discrete units called *genes* $g_k$ ($1 \leq k \leq m$, where $m$ is the length of a chromosome, *i.e.* $x_i = g_1, g_2, ..., g_m$). A GA undertakes to select subsets (usually pairs of the form $x_i$ with $x_j$) of solutions from a *population*, called *parents*, to combine them to produce new solutions called *children* or *offsprings*. The rules of combination to form children are based on the genetic notion of *crossover*, which consists of interchanging solution values of particular variables (*i.e.* interchanging $g_k$), together with occasional operations such as random value changes, called *mutations*. The *children* produced by the mating of parents, and that pass a survivability test, are then available to be chosen as *parents* for the next *generation*. This survivability test consists in optimization of a function $f$ – the *fitness function* – which has as an input variable a certain solution vector $x_i$.

There are several steps in order to apply a GA:

1. Encoding technique: *gene*, *chromosome*

2. Initialization procedure: *creation*

3. Evaluation function: *environment*

4. Selection of parents: *reproduction*

5. Genetic operators: *mutation*, *recombination*

6. Parameter settings: *practice* and *art*

The standard genetic algorithm is illustrated in Figure 3.14.

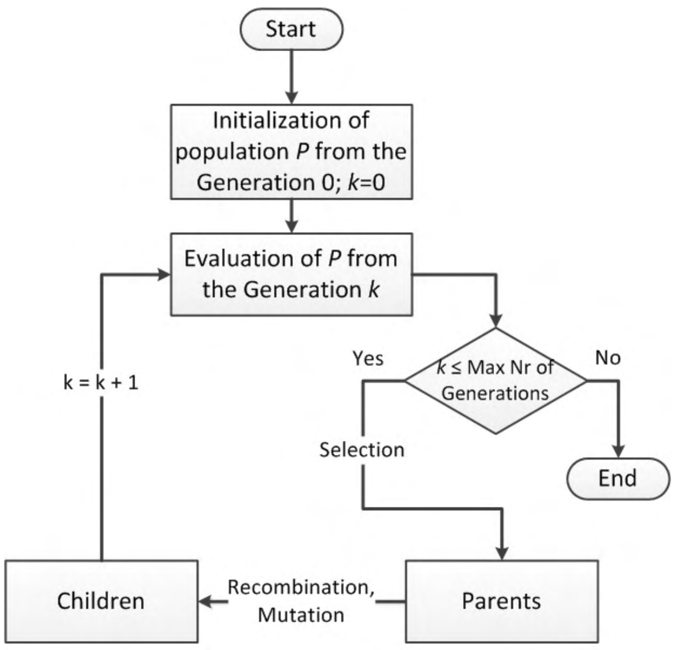

Start
Initialization of population P from the Generation 0; k=0
Evaluation of P from the Generation k
k ≤ Max Nr of Generations
Yes
No
End
Selection
k = k + 1
Children
Recombination, Mutation
Parents

Figure 3.14 : Block diagram of a standard Genetic Algorithm.

#### 3.3.3.2 Encoding and Initialization Techniques

In the initialization step, the first thing to do is to decide the coding structure. Coding for a solution $x_i$, is usually made using *binary encoding*, described as a string of binary symbols 0 and 1. The initial population is selected completely at random, with each symbol of each solution having a 50% chance of taking the value 0 or 1.

**Example 1:** gene $g_k$ represents a datum: car brand (001), engine type (010), color (101).
A chromosome $x_i$ would be an array of genes:
(car brand, engine type color) $\leftrightarrow$ (001, 010, 101).

Another type of encoding is *value encoding*, where values such as real numbers, or characters, or objects are used and the binary encoding would be difficult to apply.

**Example 2:** Chromosome 1: (0.12, 3.45, 6.78).
Chromosome 2: (A, E, I, O, U, Y).

In *permutation encoding* a chromosome is formed of a string of numbers giving a position in a sequence. It is used in ordering problems mostly (*e.g.* the Travelling Salesman problem).

**Example 3:** Chromosome 1: (1, 3, 5, 2, 4).
Chromosome 2: (1, 2, 4, 3, 5).

#### 3.3.3.3 Evaluation Function and Selection of Parents

At each step, the GA uses the current population to create the children that make up the next generation. The algorithm selects a group of *chromosomes*, who contribute their *genes*, the entries of their vectors, to their children. The problem is how to select these chromosomes. According to Darwin's evolution theory, the best ones should survive and create new offspring. In case of GA, the selection is done according to the fitness function (*i.e.* the function to be optimized): the parents with better fitness values have greater chance to be selected. There are many methods how to select the best *chromosomes*, for example *roulette wheel selection*, *Boltzman selection*, *tournament selection*, *rank selection*, *steady state selection* [sit 12c].

**Roulette Wheel Selection :**

In Roulette Wheel Selection a subject representing $p\%$ of total fitness has $p\%$ chances to be selected for mating. It is like a roulette wheel where all chromosomes of the population are placed, everyone of them having its place as big as its fitness function, as shown in Figure 3.15. When a marble is thrown there and selects the chromosome. The chromosome with bigger fitness will be selected more times.

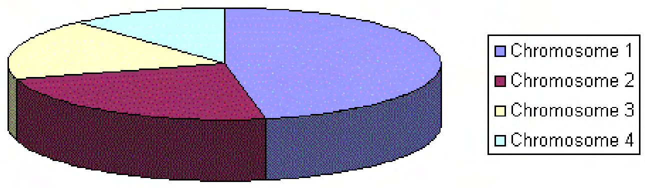


Figure 3.15 : Roulette wheel selection [sit 12c].

**Example 4:** Let's suppose 4 chromosomes $c_1$, $c_2$, $c_3$, and $c_4$ needed to maximize a function $f$. The *score* of each chromosome (*i.e.* its fitness value) will be 1, 1.25, 1.5, and 6.25 respectively. The total fitness will be 10 in this case, and the chromosomes will have the following percentages of the total fitness: 10%, 12.5%, 15%, and 62.5% respectively. Thus, for the next generation, $c_1$ will have 10% chance to be selected, while $c_4$ will have a chance of 62.5%.

**Rank Selection :**

The roulette wheel selection will have problems when the fitness differs very much. Supposing that the best chromosome fitness is 90%, then the other chromosomes will have very few chances to be selected. Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2, *etc.*, and the best will have fitness $N$ (number of chromosomes in population). You can see in Figures 3.16 and 3.17 how the situation changes after changing fitness to order number. After this, all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from the others.

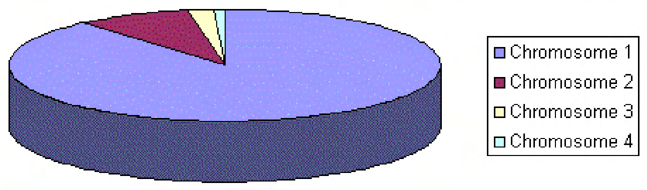


Figure 3.16 : Situation before ranking (graph of fitnesses) [sit 12c].

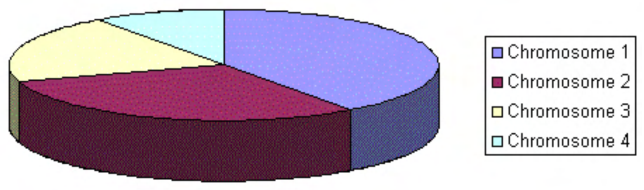


Figure 3.17 : Situation after ranking (graph of order numbers) [sit 12c].

**Example 5:** Let's consider the same 4 chromosomes from Example 3.3.3.3. After ranking them decreasingly according to their score, we obtain the next order: $c_4$, $c_3$, $c_2$, $c_1$ resulting in the following fitness values: 4, 3, 2, 1. The total fitness will be 10, and the chromosomes will have the following percentages of the total fitness: 40%, 30%, 20%, and 10% respectively. Thus, for the next generation, $c_1$ will have 10% chance to be selected, while $c_4$ will have a chance of 40%.

**Steady-State Selection :**

In the *Steady-State* one (or a few) member at random is selected, replicated, and another (or a few) random member is replaced with the copy in each time step [Rogers 99]. The replacement is done even if the offsprings are worse than the chromosomes they replace. The main idea of this selection is that big part of chromosomes should survive to the next generation.

**Elitism :**

Elitism provides a means for reducing genetic drift by ensuring that the best chromosomes are allowed to pass/copy their traits to the next generation [Ahn 03]. Some genes of chromosomes may turn out to be more important to the final solution than others. Thus, using the elitism, the best a few best chromosomes are copied to a new population. The rest is done in a classical way. Elitism can very rapidly increase the performance of GA, because it prevents losing the best found solution [sit 12b].

### 3.3.3.4 Genetic Operators

The genetic algorithm creates three types of children for the next generation [MathWorks 12a]:

**Elite children :** are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.

**Crossover children :** are created by combining the genes of a pair of parents. We distinguish *one-point crossover*, *two-point crossover*, *uniform crossover*.

- *One-point crossover*: the child chromosome is formed by appending the last $t$ genes of the first parent chromosome to the first $m-t$ genes of the second parent chromosome (where $m$ represents the total number of elements of a parent, and $t$ is the number of elements to be exchanged between parents).

  **Example 6:** Considering the following chromosome parents, where | is the crossover point:
  Chromosome 1: 01100010|00111110101
  Chromosome 2: 11001110|10100111011
  The following offsprings will be produced:
  Offspring 1: 01100010|10100111011
  Offspring 2: 11001110|00111110101

- *Two-point crossover*: the child chromosome is formed by interchanging the genes between two crossover points.

  **Example 7:** Considering the following chromosome parents, where | is the crossover point:
  Chromosome 1: 01100010|00111110| 101
  Chromosome 2: 11001110|10100111|011
  The following offsprings will be produced:
  Offspring 1: 01100010|10100111|101
  Offspring 2: 11001110|00111110|011

- *Uniform crossover*: it decides (with a given probability known as *mixing ratio*) which genes from a parent will contribute to form the children. It allows the parents to be mixed at gene level rather than the segment level.

  **Example 8:** Considering the following chromosome parents:
  Chromosome 1: 0110001000111110101
  Chromosome 2: 1100111010100111011
  Considering a mixing ratio of 50% means that approximately half of the genes will be taken from each of the parents. The following offsprings could be produced:
  Offspring 1: 0110101000100110101
  Offspring 2: 1100011010111111011

**Mutation children :** are created by introducing random changes, or mutations, to a single parent. It operates independently on each individual by perturbing with a certain probability each bit string. A usual way to mutate is to generate a random value $v$ between 1 and $N$ and then make a random change in the $v^{th}$ element of the string with a probabilistic bit mutation.

**Example 9:** Considering the following offspring:
Initial offspring: 01100010001111110101
If the generated value $v = 5$, then at the $5^{th}$ gene of the offspring a mutation will occur:
Mutated offspring: 01101010001111110101

## 3.4 Proposed Forecasting Model

The current section describes the proposed forecasting model for a time-series. The architecture of Neural Networks is presented that depends on the number of series coming from the Wavelet Transform decomposition of the input data. An optimization of the ANN with Genetic Algorithms is shown. Different settings of the model according to different time-range forecasting is described.

### 3.4.1 Sidebar

As it was stated in Section 3.1 we propose a time-series forecasting model in order to answer the following questions: is there a model that could be used in forecasting different time-series (*i1*), what parameters to use in order to improve the performance of the prediction model (*i2*), and how a constant upgrading with the latest available information and retraining the model influences the forecasting (*i3*). The proposed model, presented in this section, is based on *Stationary Wavelet Transform* time-series processing with *Artificial Neural Networks* and their optimization using *Genetic Algorithms*.

The model is implemented using the usual training-testing phases in order to evaluate its performance. Thus, the time-series will be divided into a training and testing part. The training part will be analysed and will *teach* the model. The learned model will be used to predict future information, which will be compared to the remained testing part from the time-series.

The implementation is done as follows. From the *Input Data for Analysing* we have to extract the *training* and *testing* series (Figure 3.18). This extraction is done according to the following choices:

- type of data,
- time-range of prediction,
- type of forecasting model.

After this division of data, the next step is to make the *wavelet decomposition* of level $n$ of both training and testing time-series. These 2 steps are described in section 3.4.3.

The $n^{th}$ level of decomposition gives us $n+1$ series for processing: 1 approximation and $n$ detail. Each of these is used to forecast the time-series corresponding to the same level from the predicted signal (*i.e.* the $3^{rd}$ level details from training and testing series, are responsible for prediction of the $3^{rd}$ level details from the forecasted one). The forecasting is done using $n + 1$ *Artificial Neural Networks*, one for each level of decomposition. But, before applying the wavelet to the ANN, we need to group parts of the signals into neurons. This is done according to our choice of time-shifting between different parts of the series.

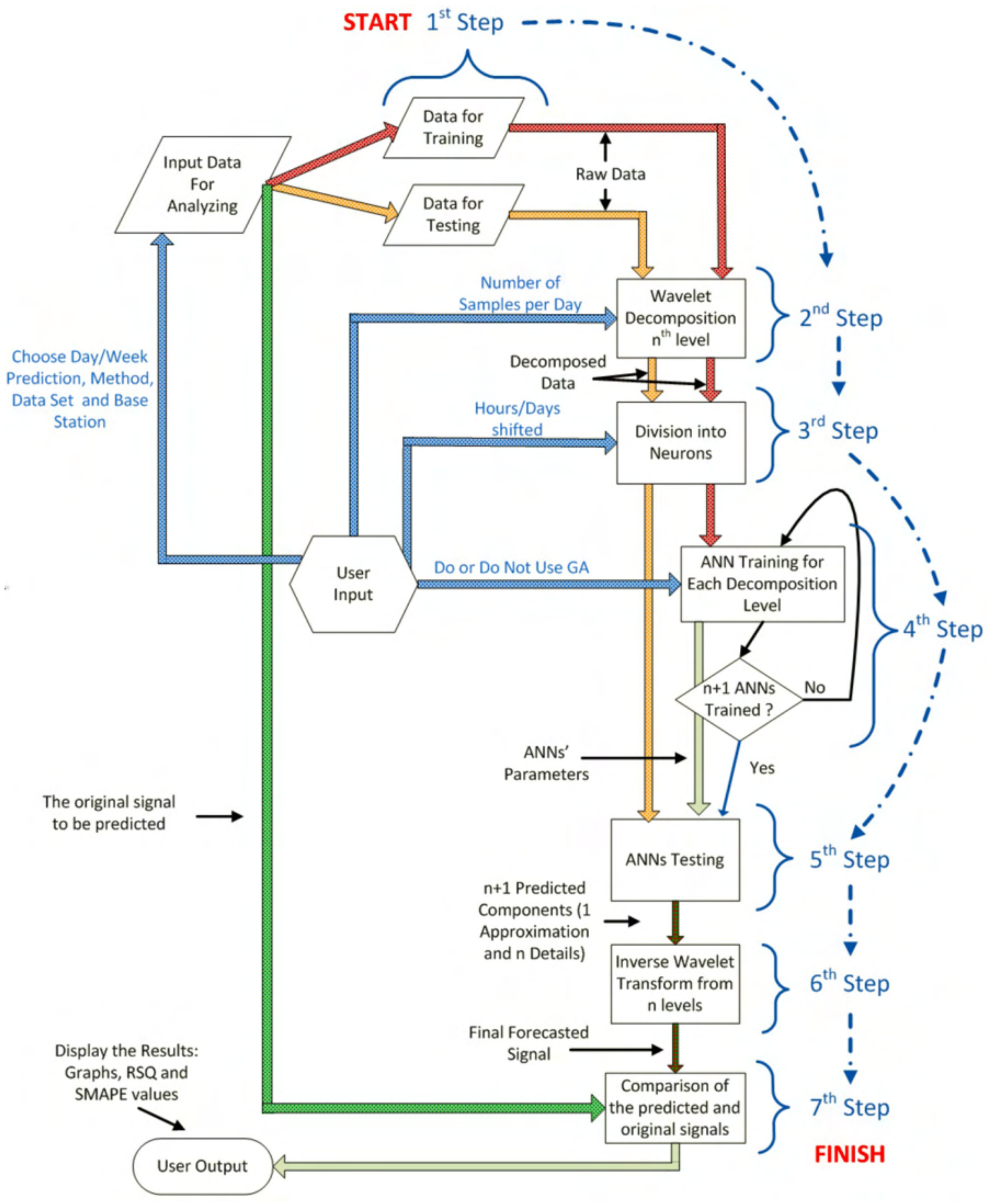

START
1st Step
Data for Training
Data for Testing
Input Data For Analyzing
Raw Data
Number of Samples per Day
Wavelet Decomposition nth level
2nd Step
Choose Day/Week Prediction, Method, Data Set and Base Station
Decomposed Data
Hours/Days shifted
Division into Neurons
3rd Step
User Input
Do or Do Not Use GA
ANN Training for Each Decomposition Level
4th Step
n+1 ANNs Trained ?
No
Yes
ANNs' Parameters
The original signal to be predicted
ANNs Testing
5th Step
n+1 Predicted Components (1 Approximation and n Details)
Inverse Wavelet Transform from n levels
6th Step
Final Forecasted Signal
Display the Results: Graphs, RSQ and SMAPE values
Comparison of the predicted and original signals
7th Step
FINISH
User Output

Figure 3.18 : Block diagram of the forecasting process.

Another user's input is the ANN's training, which can be done in 2 ways:

- using usual Neural Network training (*e.g.* with the Matlab®'s Toolbox), or
- using the *Genetic Algorithm*.

These points correspond to steps 3, 4, and 5 from Figure 3.18 and are explained in sections 3.4.4 and 3.4.5.

The next step, represents the reconstruction of the predicted signal by applying the *Inverse Stationary Wavelet Transform*. For this phase, we use one forecasted approximation signal, and $n$ forecasted detail signals. After reconstruction, we make a comparison between the real and predicted signals.

A simplified diagram of these steps is presented in Figure 3.19.

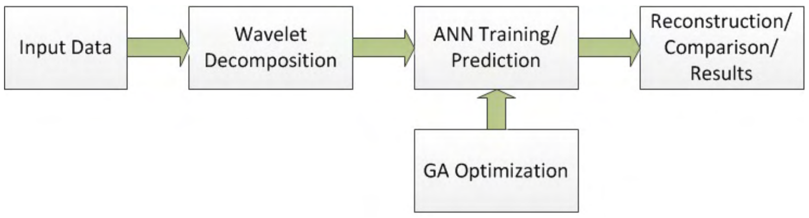


Figure 3.19 : Simplified block diagram of the forecasting process.

### 3.4.2 Data for Analysing

Two sets of time-series were used. The first one consists of WiMAX traffic volume, while the second is the EUR/USD currency exchange volume and price.

The WiMAX traffic series consists of 2 subsets. The first one was obtained by monitoring the incoming and outgoing traffic from 67 Base Stations for 8 weeks: from March 17th till May 11th. For each Base Station we have its own series. It consists of numerical values representing the total number of packets from the uplink or downlink traffic during a time interval of 15 minutes. It can be easily deducted that for a given BS we have the following number of samples: 96[Samples/Day], 972[Samples/Week], and a total number of 5376[Samples]. This database will be names as $W67$. The second WiMAX data set was obtained by monitoring the traffic from 504 BS for 5 days. In comparison to the previous set, the numerical values for this one are the total number of packets from the uplink or downlink traffic during 30 minutes. Results 48[Samples/Day]. It will be noted as $W504$ database.

Regarding the EUR/USD time-series, we used the volume and exchange rate of the EUR/USD pair. The period of collection is of 15 weeks and the values are recorded every 15 minutes. We have 96[Samples/Day], 672[Samples/Week] and a total number of $10,080$[Samples]. The volume database will be noted by us as $EUVol$, while the exchange rate DB as $EURate$.

### 3.4.3 Wavelet Decomposition

Our understanding in using Wavelet Decomposition was that the observed traffic can be thought of as a mixture of some distinct process components at different scales and volatility levels. Since the observed time-series is a mixture of such complex processes, a forecaster who is unable to identify the separate scale-related components of the series, is unable to produce models which are capable of giving accurate forecasts. So, our approach consists into decomposing the original time-series into scale or frequency related components and model each component separately, in order to obtain more accurate models. With this hypothesis, after obtaining the wavelet decomposition, neural network predictive models for each of the decomposed level of the original series were designed.

We test the prediction performance after processing the time-series with various types of mother wavelets such as *Daubechies* (db), *Coiflet* (coif), *Symlet* (sym), *Biorthogonal* (bior), and *Reverse Biorthogonal* (rbio). When applying the WT on our data, we have to choose the level of decomposition $n$. The level $n$ of decomposition depends on the number of samples per day we can choose: 96, 32, or 16 samples. For a discrete signal, in order to be able to apply the Stationary Wavelet Transform, if a decomposition at level $n$ is required, then $2^n$ must divide evenly into the length of the signal. So, for our data, if we have 96 samples per day, then $n$ cannot be greater than 5. For 32 and 16 samples per day, $n$ is 4 and 3 respectively. The decision of choosing 32 and 16 samples per day is because of the fact that there should be some periodicities in the WiMAX traffic, which are better noticed if we modify the sampling from 15 minutes to 45 and 90 minutes, each sample being obtained by making the mean value between other 3 or 6 consecutive samples [Stolojescu 09]. It was supposed that the networks would perform better if we have some "periodicities" in the traffic.

An important aspect when applying wavelets is to choose which part of the data should be used for decomposition. We constructed 3 different models regarding this aspect: 2 for one day prediction, and the $3^{rd}$ for week prediction described next. These approaches contribute to the analysis of the issue (*i2*) described in Section 3.1.

#### 3.4.3.1 Single Day Prediction model 1 - Similar Days Selection

This method is used for prediction of a given day from the last week. The same days of the week are selected for analysis as the one we want to forecast. For example, let's consider the $W67$ database to predict how the traffic on Wednesday at the $33^{rd}$ BS is. Taking into account that we have 8 weeks period, then, only the time-series from all Wednesdays during weeks 1 to 7 will be taken for processing. The advantage in this technique is that usually the user's behaviour is modelled during certain week days, but the disadvantage is that the number of subscribers is always changing.

For training time-series, we concatenated the raw series representing the samples from Wednesdays taken from weeks 1 till 6. This information is used, after wavelet decomposition, for ANN inputs represented in steps 3 and 4 of the block diagram (Figure 3.18). For ANN target in training process, the time-series from the Wednesday of the $7^{th}$ week was used (Figure 3.20). For testing data, we need at ANN's inputs the SWT of the concatenated days from weeks 2 to 7. What we expect from the output of the ANN is the SWT of Wednesday traffic from $8^{th}$ week (Figure 3.21). An example of approximation and detail

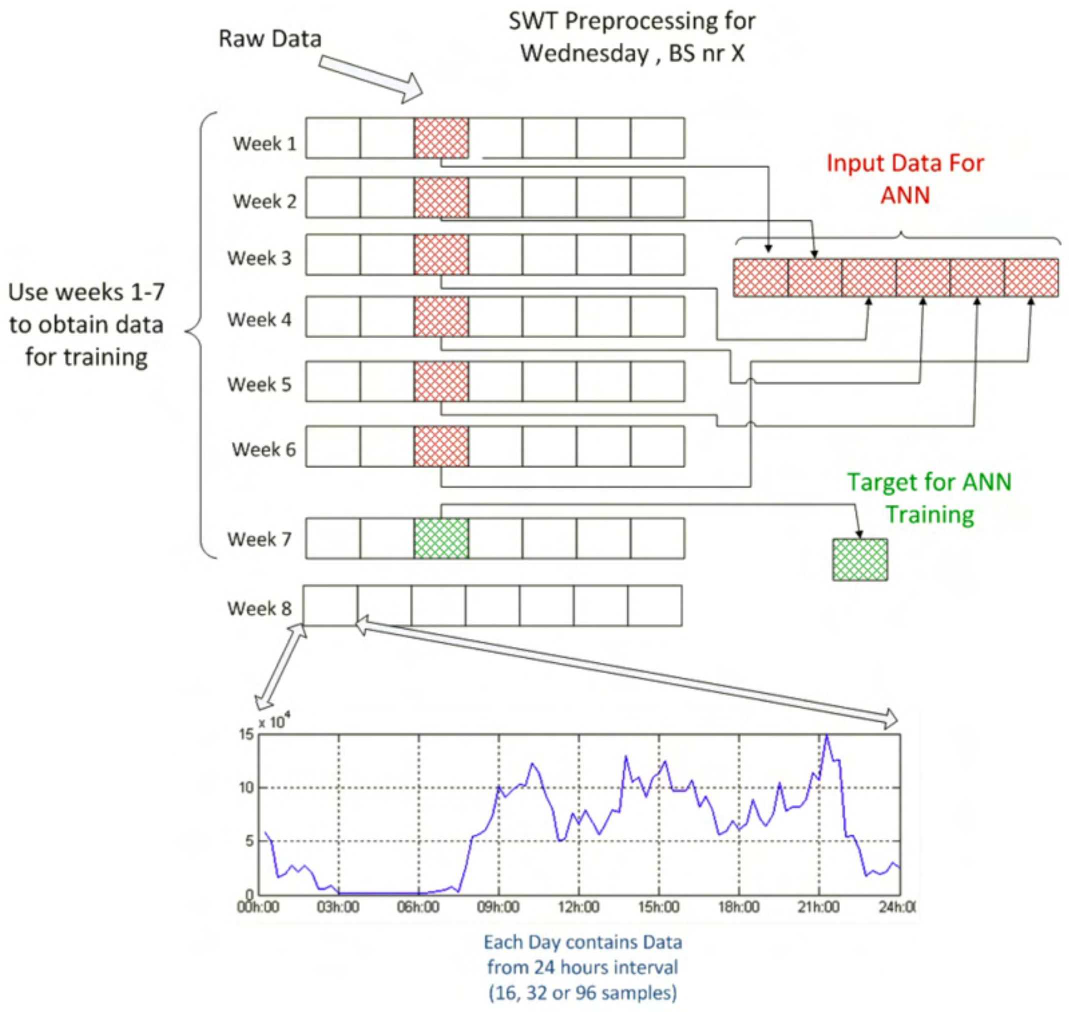


Figure 3.20 : SWT preprocessing for ANN training, used in 1st method for day prediction.

signals is shown in Figure 3.22.

Regarding the $W504$ dataset, it has traffic information from 5 days only. Thus, we did not use this method in prediction.

#### 3.4.3.2 Single Day Prediction model 2 - All Days Selection

The second approach in this prediction, was to take into consideration all the available data before required day for forecasting. In comparison to the previous case (Subsection 3.4.3.1), here our series is not interrupted and we do always know about the amount of traffic coming from new subscribers.

Let's take the same example: prediction for Wednesday, 8th week for the $W67$ DB. For the training of neural network, we use for ANNs' inputs the data beginning from Monday of the 1st week till the day before forecasting from the 7th week, *i.e.* Tuesday. For the

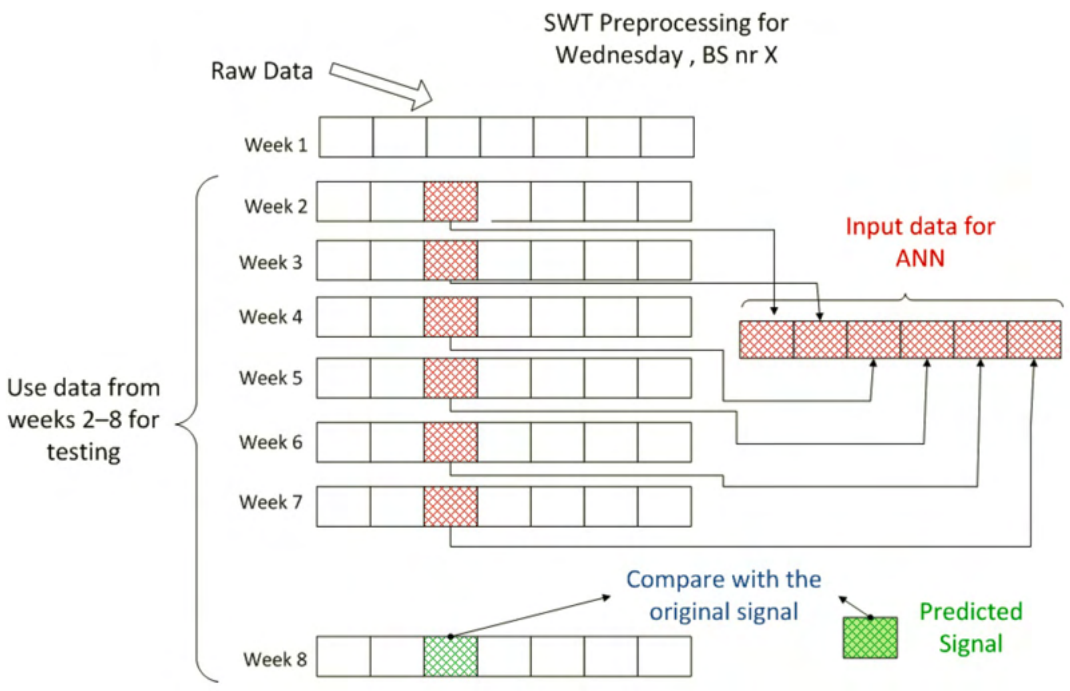

SWT Preprocessing for Wednesday , BS nr X
Raw Data
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Use data from weeks 2–8 for testing
Input data for ANN
Compare with the original signal
Predicted Signal

Figure 3.21 : SWT preprocessing for ANN testing, used in 1st method for day prediction.

ANNs' targets the signal representing WiMAX traffic on Wednesday, 7th week was used (Figure 3.23).

For testing phase we need at ANN inputs the SWT from Monday of the 2nd week, till Tuesday of the 8th week (Figure 3.24). We expect from the ANNs' outputs the SWT transform of the time-series, which after Inverse SWT processing, would give us the traffic during the forecasted day (*i.e.* Wednesday, from the 8th week).

This technique of forecasting is applied to the $W504$ data set also. No day ahead forecasting was performed with the EUR/USD currency exchange databases.

### 3.4.3.3 One Week Prediction

Based on the Hann and Steurer [Hann 96] remarks in a research regarding weekly exchange rate data forecasting that ANN outperforms the linear models, we made prediction of one week ahead. Also, in this case we do have the correlation between given days of the weeks (explained further in ANN approaching), and we know the rising of customer demands, because we are able to process every day from all weeks.

We take once again as an example the $W67$ database, and we forecast the traffic from the last 8th week of a given Base Station (Figures 3.25 and 3.26). As in the previous cases, the role of SWT is to make the decomposition of data before entering the ANNs. As an input, we had the wavelet transforms obtained after the decomposition of the time-series from weeks 1 till 6, while the series from the 7th week was taken as a target. In testing

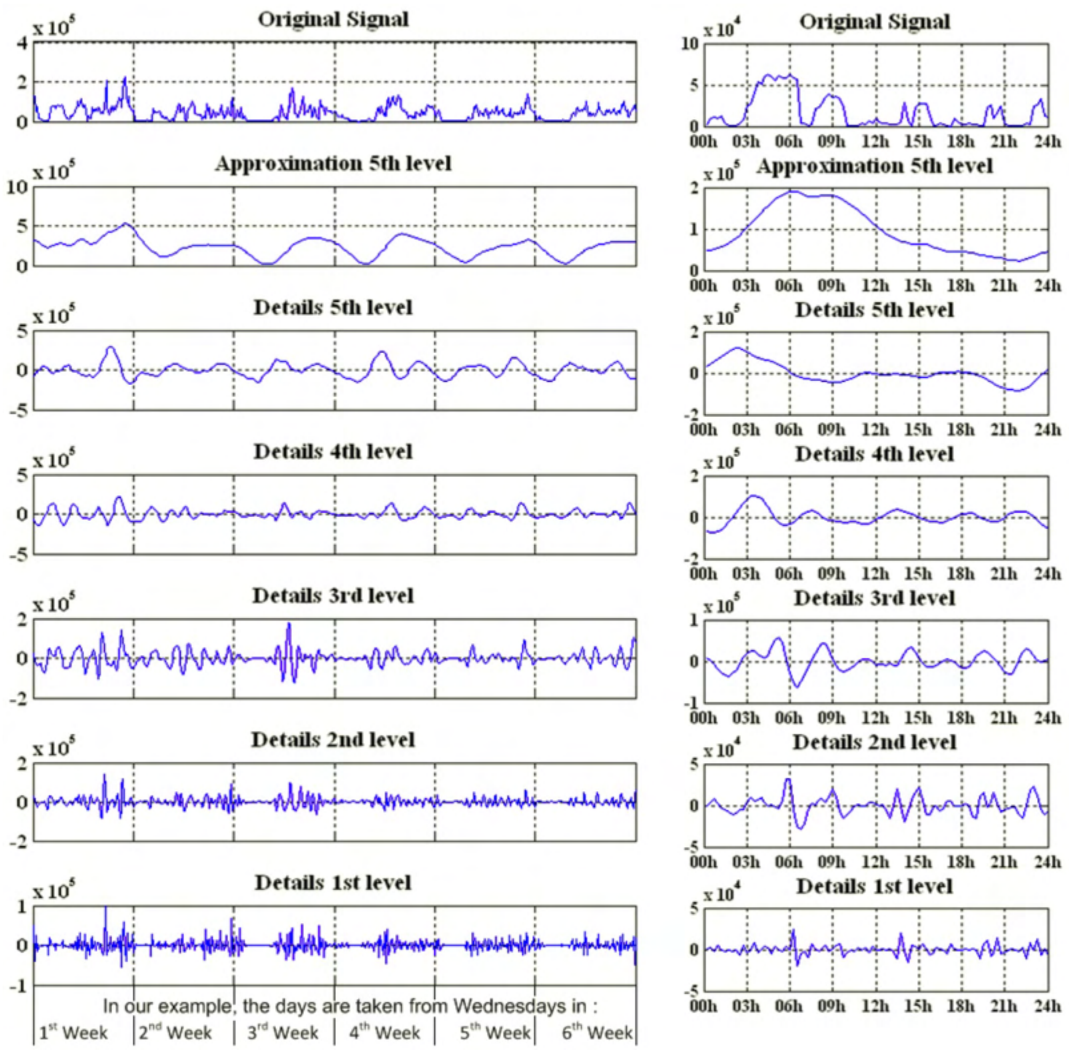


Figure 3.22 : Approximation and Detail time-series for training, used in $1^{st}$ method for future ANN inputs (to the left), and for ANN target (to the right).

process, the ANNs inputs contained the SWT of the series from weeks $2^{nd}$ till $7^{th}$. The traffic from the last week was obtained at the output.

### 3.4.4 Neural Network Design

After wavelet decomposition, our next task is to design the Artificial Neural Networks (Figure 3.19). In order to implement a neural network, first of all we had to choose its model. As it has been mentioned in Section 3.3.2.2, the 2 most important types of ANNs are *Feed-Forward Neural Networks* and *Recurrent Neural Networks*. Feed-Forward ANNs were applied in our forecasting techniques, because according to [Dematos 96], this model

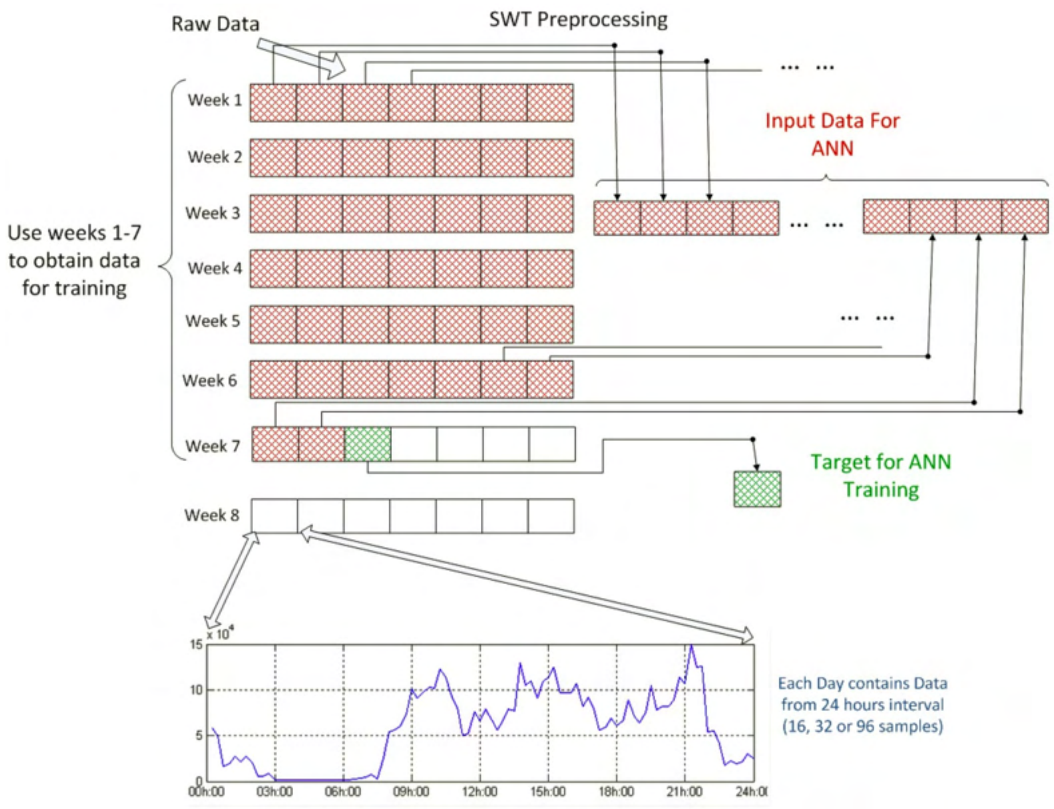


Figure 3.23 : SWT preprocessing for ANN training, used in 2nd method for day prediction.

is relatively accurate in forecasting, despite being quite simple and easy to use. Anyway, the recurrent network forecast performance was lower than that of the feed-forward model. It might be because of the fact that recurrent networks pass the data from back to forward as well as from forward to back, and can become "confused" or unstable.

Further designing of feed-forward network implies the establishment of:

- the number of layers,
- the number of neurons in each layer.

These aspects are very important if we want to minimize the generalization error, the learning time, and the ANN dimension.

In [Ileană 04] is pointed out the fact that the choice of the number of layers is made knowing that one hidden layer network is able to approximate most of the nonlinear functions demanded by practice. This fact has been observed by us also in our earlier studies on neural networks, that's why we have chosen single hidden layer ANN.

Concerning the *dimension of each neuron layer* the situation is as follows:

- *input* and *output* layers are imposed by the problem to be solved,

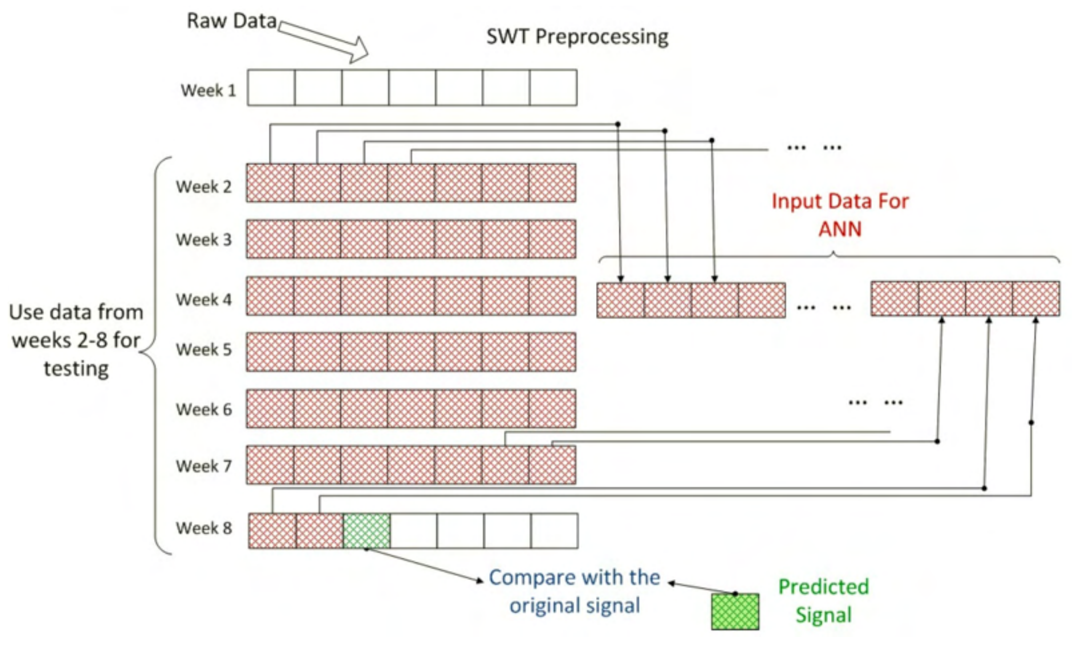


Figure 3.24 : SWT preprocessing for ANN testing, used in $2^{nd}$ method for day prediction.

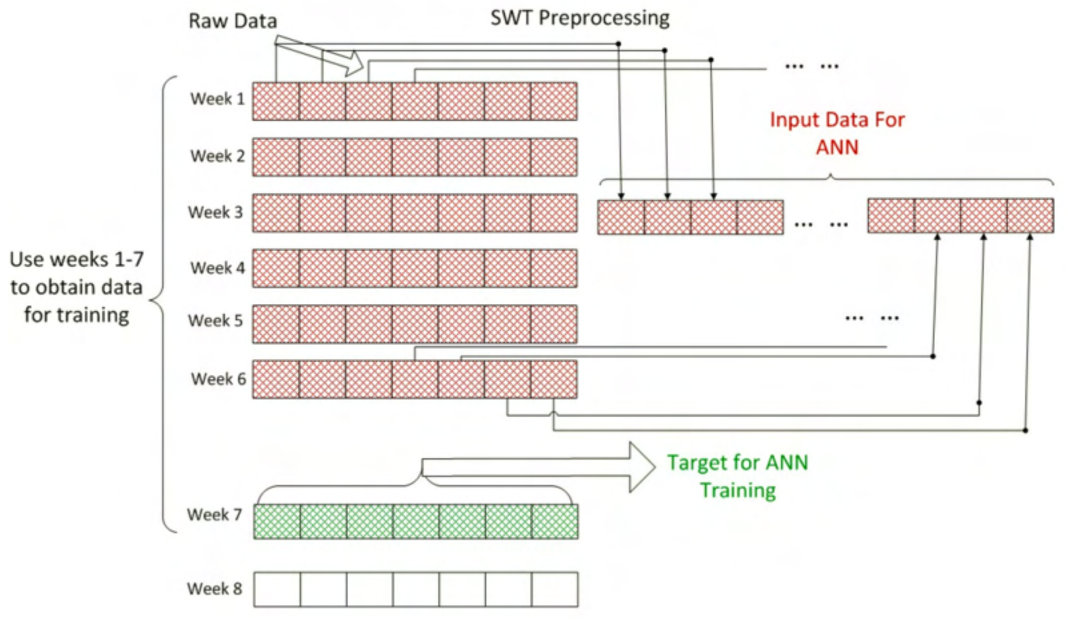


Figure 3.25 : SWT preprocessing for ANN training, used in One Week Forecasting.

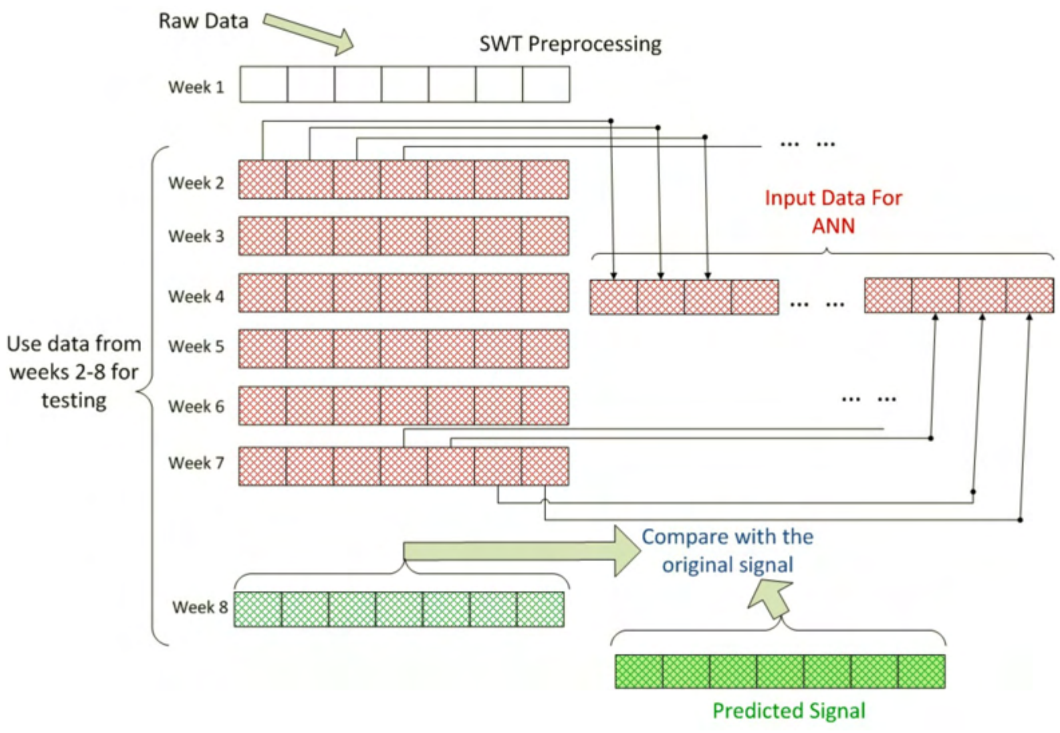


Figure 3.26 : SWT preprocessing for ANN testing, used in One Week Forecasting.

- the dimension of the *hidden* layer is essential for efficiency of the network.

#### 3.4.4.1 Single Day Prediction model 1 - Similar Days Selection

In case of day forecasting we predict a *single-vector time-series* (data from one day, or data from one week). This aspect imposes us the condition of using a single vector-neuron for output[2], which contains in case of day prediction an array of 96, 32, or 16 samples. In case of week forecasting, the number of samples is 672, 224, or 112 values[3].

For the number of input neurons we take into consideration different periods of the day and different user's behaviours. One of the possibilities for grouping the input data is to perform a temporal synchronization of all inputs (Figure 3.27). We take the time-series from an entire day from hours $00h00$ till $24h00$ as each vector-input of the ANN. By this we make sure that the morning, noon and evening periods from known series, are responsible for the same periods of the day from the forecasted series.

Another possibility of choosing the information for our input neurons arises if we think about the behaviour of each person. According to [Knauth 96] our life cycle is divided into 3 parts: 8 hours of sleep, 8 hours of work, and the rest of 8 hours we use for resting. But

[2]We used Matlab® in our simulations, and in its neural toolbox a neuron is seen actually as a vector of elements.

[3]For further explanation, we use the case of 96[Samples/Day]

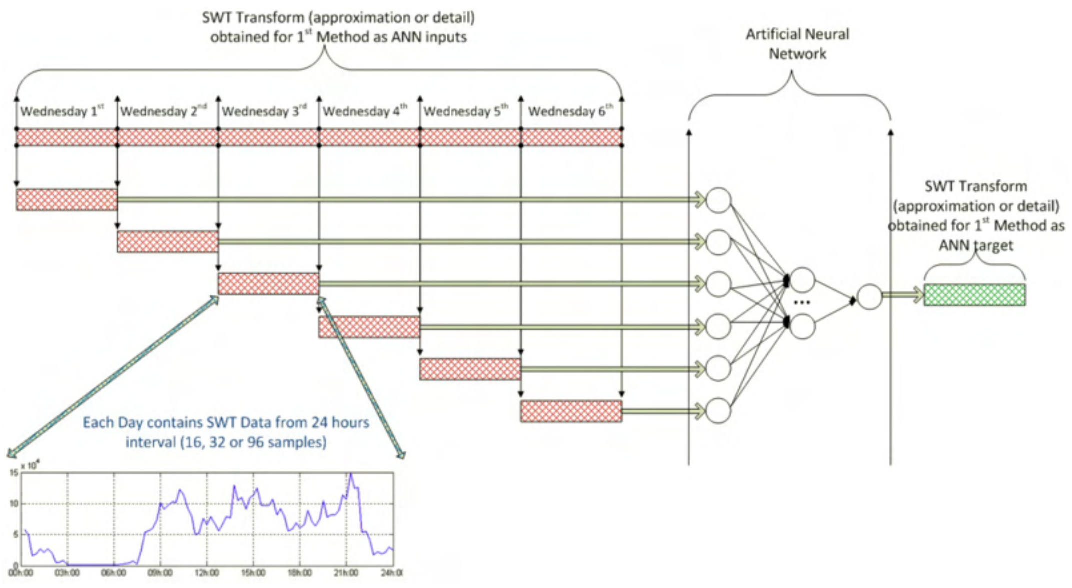
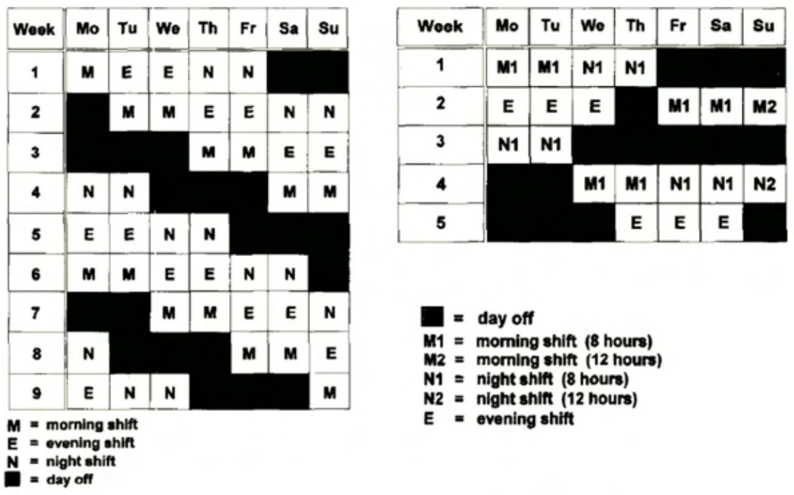

SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs
Artificial Neural Network
Wednesday 1st
Wednesday 2nd
Wednesday 3rd
Wednesday 4th
Wednesday 5th
Wednesday 6th
SWT Transform (approximation or detail) obtained for 1st Method as ANN target
Each Day contains SWT Data from 24 hours interval (16, 32 or 96 samples)

Figure 3.27 : 24 Hours Shift for neuron decomposition.

there are some industries where the use of these hours is shifted, like the ones presented in Figure 3.28. Taking into consideration this information, we made a time superposition between these 3 parts, design shown in Figure 3.29. The data for vector-neurons is taken by shifting with an 8 hours time interval, but the length of each of them still is taken from 24 hours. So, we have 2/3 of similar data for two adjacent vector-neurons.



| Week | Mo | Tu | We | Th | Fr | Sa | Su |
|---|---|---|---|---|---|---|---|
| 1 | M | E | E | N | N | ■ | ■ |
| 2 | ■ | M | M | E | E | N | N |
| 3 | ■ | ■ | ■ | M | M | E | E |
| 4 | N | N | ■ | ■ | ■ | M | M |
| 5 | E | E | N | N | ■ | ■ | ■ |
| 6 | M | M | E | E | N | N | ■ |
| 7 | ■ | ■ | M | M | E | E | N |
| 8 | N | ■ | ■ | ■ | M | M | E |
| 9 | E | N | N | ■ | ■ | ■ | M |

M = morning shift
E = evening shift
N = night shift
■ = day off

| Week | Mo | Tu | We | Th | Fr | Sa | Su |
|---|---|---|---|---|---|---|---|
| 1 | M1 | M1 | N1 | N1 | ■ | ■ | ■ |
| 2 | E | E | E | ■ | M1 | M1 | M2 |
| 3 | N1 | N1 | ■ | ■ | ■ | ■ | ■ |
| 4 | ■ | ■ | M1 | M1 | N1 | N1 | N2 |
| 5 | ■ | ■ | ■ | E | E | E | ■ |

■ = day off
M1 = morning shift (8 hours)
M2 = morning shift (12 hours)
N1 = night shift (8 hours)
N2 = night shift (12 hours)
E = evening shift

Figure 3.28 : Continuous shift system in the steel industry (to the left), and in the chemical industry (to the right).

Thus, the number of input vector-neurons can be deducted according to the next

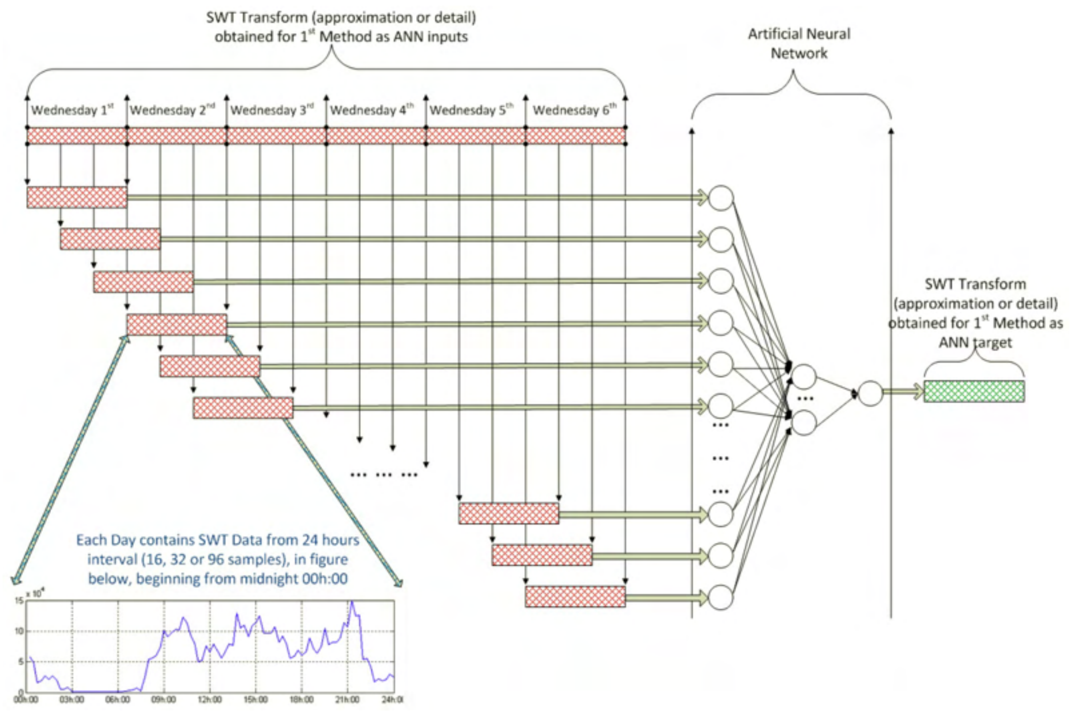

SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs
Wednesday 1st
Wednesday 2nd
Wednesday 3rd
Wednesday 4th
Wednesday 5th
Wednesday 6th
Artificial Neural Network
SWT Transform (approximation or detail) obtained for 1st Method as ANN target
Each Day contains SWT Data from 24 hours interval (16, 32 or 96 samples), in figure below, beginning from midnight 00h:00
00h:00
03h:00
06h:00
09h:00
12h:00
15h:00
18h:00
21h:00
24h:00

Figure 3.29 : 8 Hours Shift for neuron decomposition.

formula (for the $W67$ DB):

$$[\text{NrOfDays - 1}] \cdot \frac{24\text{h}}{[\text{NrOfShiftHours}]} + 1 = 6 \cdot \frac{24\text{h}}{8\text{h}} + 1 = 16\,[\text{Vector-Neurons}] \tag{3.19}$$

Based on similar research as in [Knauth 96], Tucker et.al. [Tucker 99] describes a recent survey of the range of shift systems that are currently operating, and found that about one third of continuous systems now involve 12 hours shifts. The organisations that participated for this research were manufacturing companies (steel, chemicals, aluminium, oil, chipboard, food, glass fibre) along with one engineering company. Based on this information, we made a 12 *hours* temporal synchronization between ANN's neurons. Half of the information of two adjacent ANN's vector-inputs are identical. The number of vector-neurons, according to the formula 3.19, is:

$$[\text{NrOfNeurons}] = 6 \cdot \tfrac{24\text{h}}{12\text{h}} + 1 = 11\,[\text{Vector-Neurons}]$$

#### 3.4.4.2 Single Day Prediction model 2 - All Days Selection

This approach is similar to the first model of day prediction (Subsection 3.4.4.1). The main difference is the length of series received transformed by the SWT prepared for neural network's inputs. If in the previous method it represents the time-series from 6 days interval, then in this case (if we suppose the $3^{rd}$ day of the week, Wednesday, for the $W67$ DB), we have the samples from:

$$[\text{NrOfDays}] = 6[\text{Weeks}] \cdot 7[\text{Days/Week}] + [\text{Monday} + \text{Tuesday}] = 44[\text{Days}]$$

For a shifting of 12 hours in this case, by using Equation (3.19), the number of input neurons are:

$$[\text{NrOfDays} - 1] \cdot \frac{24\text{h}}{[\text{NrOfShiftHours}]} + 1 = [44 - 1] \cdot \frac{24\text{h}}{12\text{h}} + 1 = 87[\text{Vector-Neurons}]$$

and for 8 hours shifting:

$$[44 - 1] \cdot \frac{24\text{h}}{8\text{h}} + 1 = 130[\text{Vector-Neurons}]$$

We did show how we have chosen the number of input and output neurons. Next, we present how the number of neurons from the hidden layer were selected.

If the number of input and output neurons are kind of "decided" by the information we have, then the hidden layer stays completely at our choice of designing. In [Ileană 04] is said that the network must satisfy at least 2 criteria:

- the network must be able to learn the input data,
- the network must be able to generalize for similar input data that was not used in training set.

The accomplishment degree of these requirements depends on the network complexity (*i.e.* number of neurons), training data set and number of iterations for training. Figure 3.30 presents the dependence of ANN's performance as a function of complexity, while Figure 3.31 shows the same dependence of the number of iterations used for training.

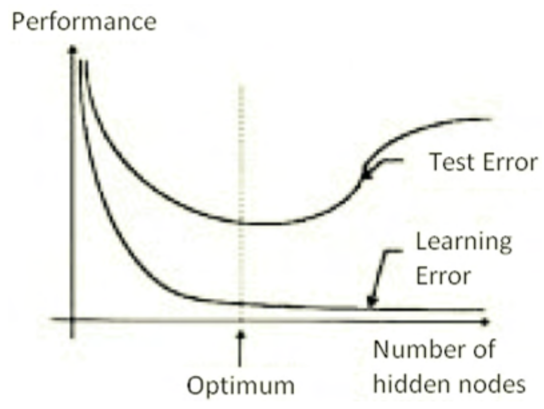


Figure 3.30 : Performance's dependence of network complexity.

Based on the above remarks, we diminished the complexity of the ANN, leading to a better generalization and an increased training speed. After simulations and tests, presented in Figure 3.32, we got the conclusion that the best value for the number of

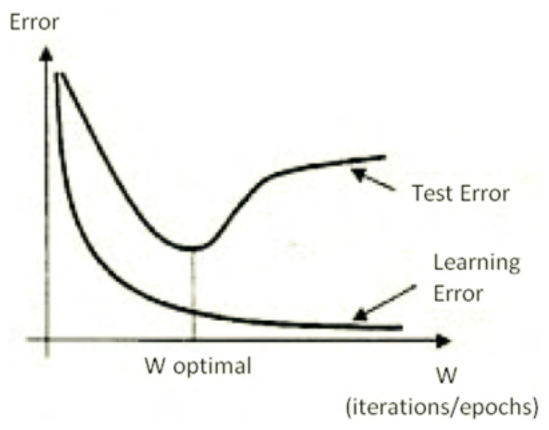


Figure 3.31 : Performance's dependence of number of iterations used for training.

neurons in the hidden layer is 3 (more information about SMAPE value can be found in subsection 3.5.2). Initially, it seemed to be good for the first method for day prediction, where we had at least 6 input vector-neurons, but it wouldn't be a good value for the second method, where the smallest number of input vector-neurons is 42, this being in case for 24 hours shifting only, while beginning with 12 hours shifting, the inputs increase to already a value greater then 80. However, by increasing the size of the hidden layer, the network was not be able to make a good generalization of the signal and of its main characteristics (*i.e.* periodicities, trend). At the output we had a lot of noise, because the ANN has been over-learned with the training signal. A comparison between 3 and 15 neurons for the intermediate layer is shown in Figure 3.33.

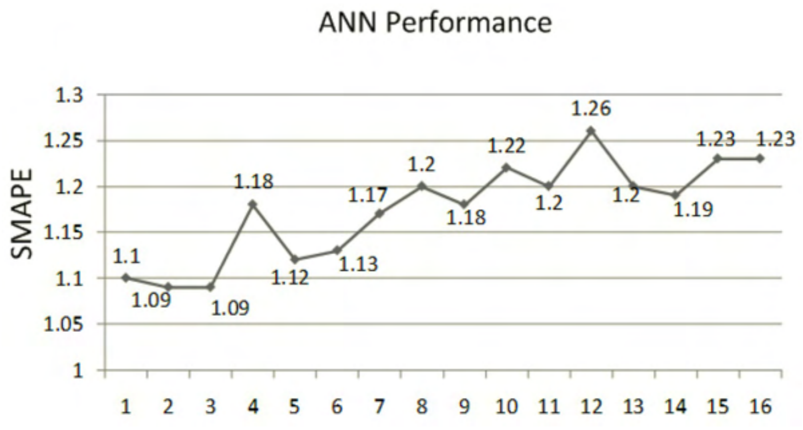


Figure 3.32 : Obtained SMAPE values as a function of neurons number in the hidden layer for One Day Prediction ANN.

For the $W504$ data set, the idea is the same. The only difference is that we have 5 days for processing instead of 8 weeks. At the ANNs' inputs, the wavelet transforms from the

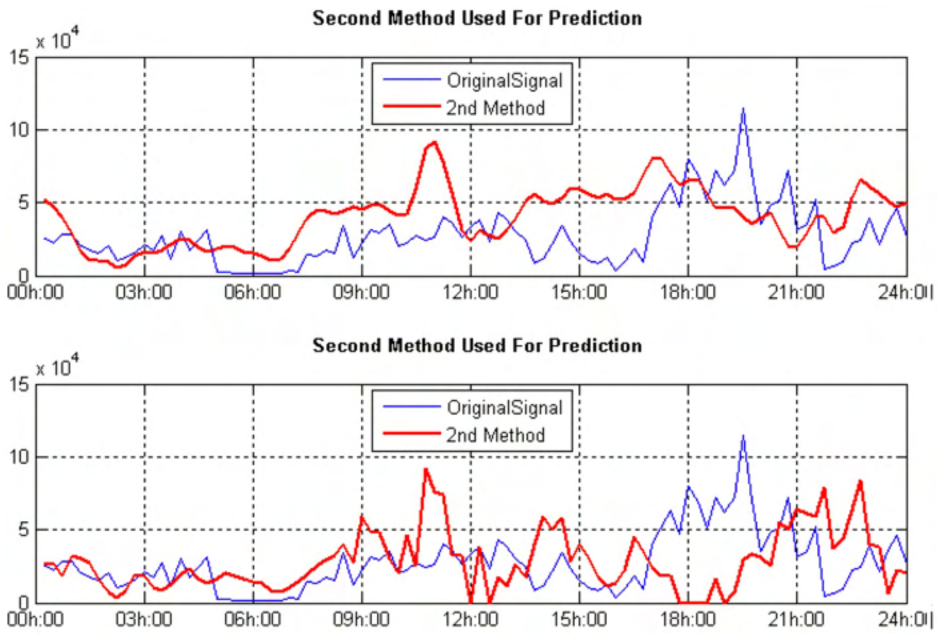


Figure 3.33 : Forecasting capability when using 3 neurons (above) and 15 neurons (below) for the size of the hidden layer in One Day Prediction.

first 3 days are used, and as a target the wavelet decompositions from $4^{th}$ day are applied. In the testing phase, the inputs contained days $2^{nd}$ till $4^{th}$, and we obtained the $5^{th}$ day decomposition at the output.

### 3.4.4.3 One Week Prediction

We forecasted the data for the entire last week. The number of vector-neurons at the output is 1, the reason being the same as in One Day Forecasting.

In order to figure out how many inputs should we have, we must know which samples to take from the wavelet decomposition signals and how to group these samples into neurons. We took into consideration the behaviour of different companies regarding the days of working, or resting, given to their employees. In [Tucker 99] is pointed out that there is a form of system involving work with four shifts system - for example, two days shifts, followed immediately by two night shifts, followed by several rest days. This type of day working is usually popular in medicine, and restaurant business. So, the idea is that we have many industries in which the working and rest days have a variable schedule. According to the information just described, we made a variable day shifting at the inputs of our ANN also. An example of 2 days shifting for the $W67$ database is depicted in Figure 3.34. The first input has the information from the first 7 days of the traffic. The next input, contains also the time-series from 7 days, but shifted with 2 days. The third vector-neuron, contains the information shifted with 4 days, and so on. This is done until the end of the entire wavelet decomposition signal, obtained from the first 6 weeks for the ANN's inputs (in case of $W67$ dataset).

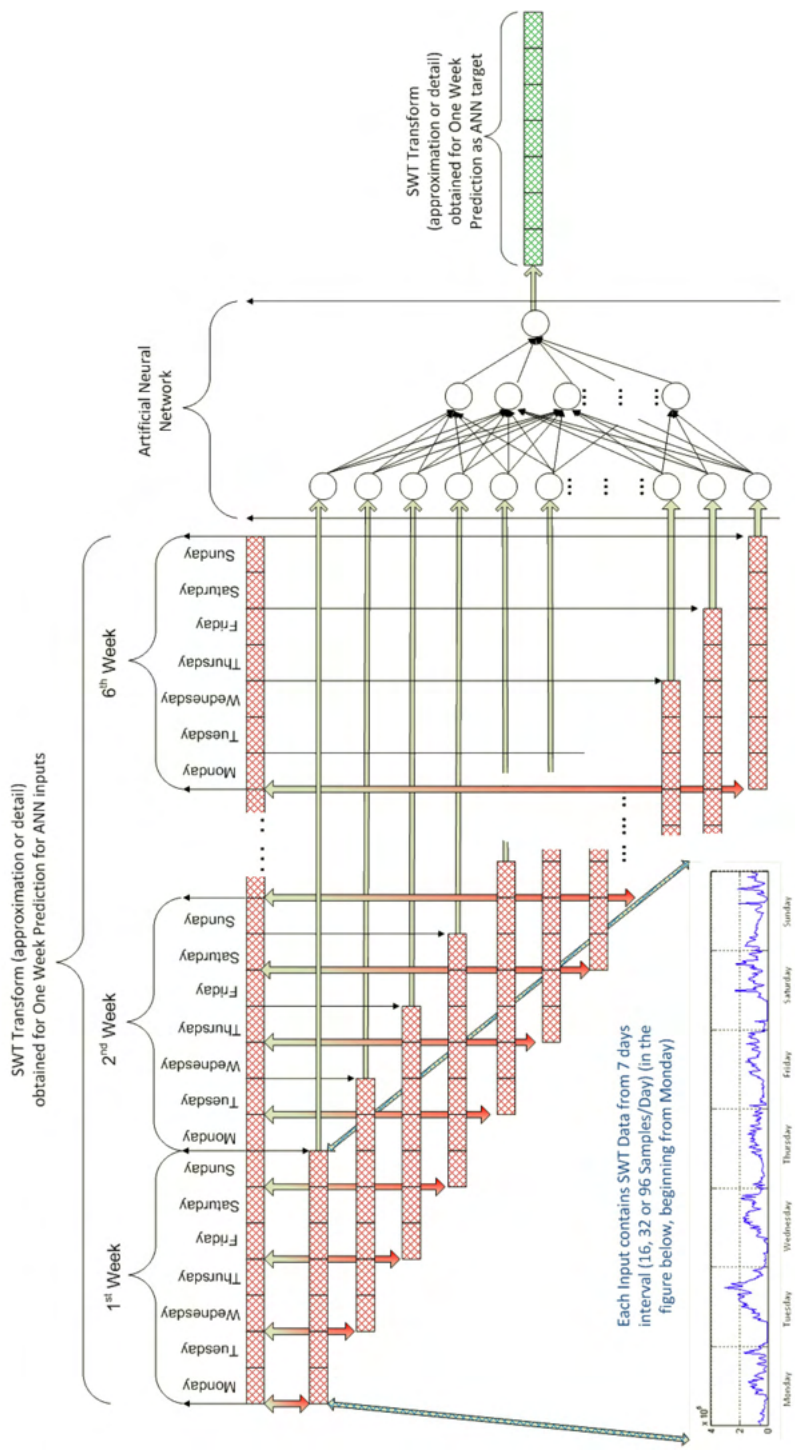

SWT Transform (approximation or detail) obtained for One Week Prediction as ANN target
Artificial Neural Network
SWT Transform (approximation or detail) obtained for One Week Prediction for ANN inputs
1st Week
2nd Week
6th Week
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Each Input contains SWT Data from 7 days interval (16, 32 or 96 Samples/Day) (in the figure below, beginning from Monday)

Figure 3.34 : 2 Days Shift for input vector-neuron decomposition, One Week Prediction.

The number of input vector-neurons, was calculated according to the next relation:

$$\left\lfloor [\text{NrOfWeeks} - 1] \cdot \frac{7[\text{Days}]}{[\text{NrOfShiftDays}]} \right\rfloor + 1 = \left\lfloor 5 \cdot \frac{7}{2} \right\rfloor + 1 = 18[\text{Vector-Neurons}] \tag{3.20}$$

where $\lfloor x \rfloor$ is the *floor function*, also called the greatest integer function or integer value, and gives the largest integer less than or equal to $x$.

For the number of neurons in the hidden layer, several tests were done by comparing the ANN capacity of prediction for different sizes of the hidden layer. The results are shown in Figure 3.35. According to these values, the number of vector-neurons for the hidden layer is 12.

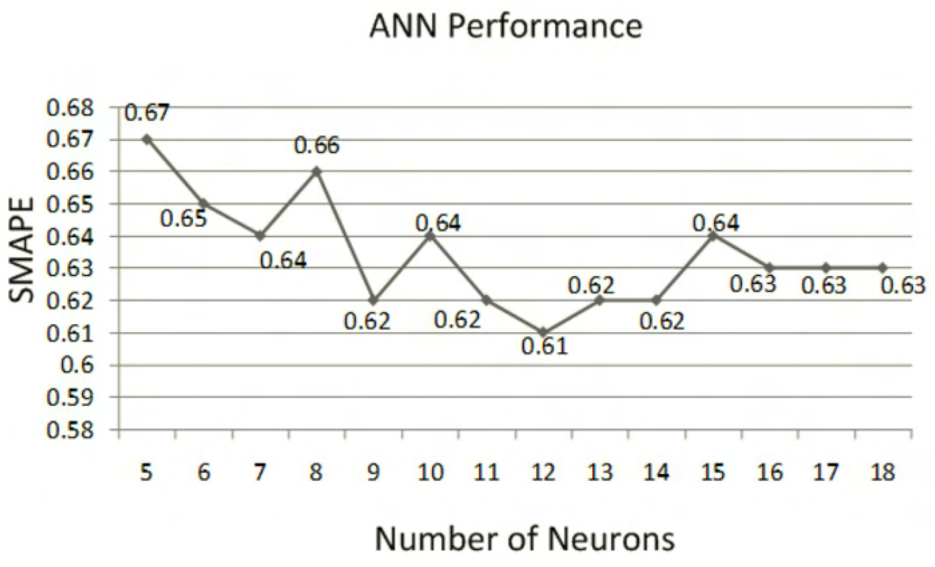


Figure 3.35 : Obtained SMAPE values as a function of neurons number in the hidden layer for One Week Prediction ANN.

So far, we have described the proposed solution for designing the neural networks, *i.e.* their configuration, the number of neurons at each layer. Another problem that stays in the ANN design, consists in choosing its *training periods* and its training *performance*. These two parameters are related to each other, meaning that more epochs will result in a better performance usually. However, a better training performance (implying more iterations) doesn't mean a correct trained network (Figure 3.31). It is considered that during training, the Mean Squared Error (MSE) between the output signal and the target should be about 5% [MathWorks 12b]. So, we set the limits for MSE to be 0.05. Regarding the number of epochs, we just made sure that they are sufficient to be able to achieve in most of the cases the MSE value written above: 350 $\leftrightarrow$ for One Day Prediction; while 150 for approximation with $150 + i \cdot 250$ for details, where $i$ represents the level of the detail signal $\leftrightarrow$ for Week Prediction.

Another important parameter in ANN design is the decision of the training function to be used.

There are several training functions:

- ***Batch Gradient Descent (traingd)***: in this function the weights and biases are updated in the direction of the negative gradient of the performance function.
- ***Batch Gradient Descent with Momentum (traingdm)***: momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low-pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With a momentum a network can slide through such a minimum.
- ***Backpropagation training with an adaptive learning rate (traingda)***: in standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. The performance can be improved if we allow the learning rate to change during training process. A near-optimal is obtained for the local terrain. When a larger learning rate could result in stable learning, the learning rate is increased. When the learning rate is too high to guarantee a decrease in error, it gets decreased until stable learning resumes.
- ***Combination between Adaptive Learning Rate with Momentum training (trainingdx)***: as its definition says, it will apply both the adaptive learning and momentum.

After several tests, we saw that *traingd* gets blocked fast and was not able to a further converging of training (it usually stopped at a MSE value of 0.74). *Traingdm*: even if it says that it doesn't get stocked in a local minimum, in our case we had several unsuccessful simulations regarding this aspect (the ANN did stop at a performance value of 0.95 also, and was not able to convergence further). Both *traingda* and *traingdx* managed to train the network, but *traingda* was a little bit slower then *traingdx* (for example, after 900 iterations of training, *traingda* reached 0.573 performance value, while *traingdx* had an MSE of 0.532). Thus, we used the *adaptive learning rate with momentum training* function for our networks.

### 3.4.5 Genetic Algorithm Optimization

One of the problems in ANN designing is that during under-training or over-training the network with its usual training algorithm, its weights are not optimized, causing in undesired prediction signal. This is why an optimization technique of its weights would be required. We optimized our artificial neural networks using genetic algorithms (GA) (Figure 3.19).

We applied the GA in order to find the optimal weights between the input and hidden layer. The proposed designing of the Genetic Algorithm is as follows:

1. Each individual contained a set of weights for all the links between layers

2. Each gene represented a single weight (real value)
3. We had a population size of 100 individuals, meaning 100 different possibilities at each generation for the network
4. The number of generations was 100 $\leftrightarrow$ less generations resulted in not finding an acceptable solution for our problem, while more generations results in a longer time processing. Anyway, above this value, we didn't manage to observe better performance of the final result
5. The fitness function represented the summation between the two training data sets, calculated as follows:

$$\begin{aligned}[\boldsymbol{Fitness}] = {} & \frac{1}{[\mathit{NrOfSamples/Day}]} \sum_{i=1}^{[\mathit{NrOfSamples/Day}]} (x_i^{\mathit{training1}} - x_i^{\mathit{original1}}) \\ & + \frac{1}{[\mathit{NrOfSamples/Day}]} \sum_{i=1}^{[\mathit{NrOfSamples/Day}]} (x_i^{\mathit{training2}} - x_i^{\mathit{original2}})\end{aligned} \tag{3.21}$$

6. The crossover function was *Scattered*, which creates a random binary vector and selects the genes where the vector is a 1 from the first parent, and where the vector is a 0 from the second parent, after, combines the genes to form the child [MathWorks 12a]. For example, if $p_1$ and $p_2$ are the parents

$$p_1 = [a\ b\ c\ d\ e\ f\ g]; \qquad p_2 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$$

   and the binary vector is [11001000], then the function returns the following child:

$$c = [a\ b\ 3\ 4\ e\ 6\ 7\ 8]$$

   The reason of choosing this crossover function is because the others would do 2 point or single point crossover between parents, and while we had so many variables, there was a great probability that the *good* weights would not be situated next to each other, thus, the performance of the algorithm would decrease

7. The number of *Elite Count* was set to 2. It represents the individuals with better fitness (*i.e.* a smaller error value) which pass to the next population. It would seem wiser to set this number to a greater value, as we used 100 individuals, but, because of many variables (genes), we let more parents to mutation and crossover
8. The mutation function was *Gaussian*, having as input 2 parameters: the *Scale* and the *Shrink*. The *Scale* parameter determines the standard deviation at the first generation. The *Shrink* parameter controls how the standard deviation shrinks as generation go by using the recursive formula:

$$\sigma_k = \sigma_{k-1}(1 - \mathrm{Shrink}\frac{k}{\mathit{Generations}})$$

   where the standard deviation is:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Both values are 1.

9. The crossover fraction was set to 0.75, 0.80, 0.85 as follows: we run the algorithm 3 times for each ANN, but the initial population for the second time, was actually the final population obtained from the first rulation, and, consequently, the initial population for the last time was the final population from the second time of applying the GA. The reason of doing this, was because with each application the randomizing values and vectors used for mutation and crossover are different, so, this implied a larger domain of activity and searching for the genetic algorithm. Also, by using the Gaussian function, at the end of the generations, the mutation values was 0, but applying once again the algorithm, this value was restarted

10. Because the training and testing error were used for function optimization, we changed the input data used, in order to have another week with unused sequences in training for the final prediction and performance comparison

GA was applied only for the first model for day prediction described in subsection 3.4.4.1, *i.e.* for $W67$ and $W504$ databases. The reason is the great number of links between the hidden and input layers for the second model of day prediction, resulting in a high time-consuming operation. Also, in order to diminish the time needed for processing and the number of variables needed (represented by genes), the hidden layer contained just 2 neurons instead of 3. If applying the genetic algorithm to the second model day prediction and one week prediction, then, according to the next equation:

$$[\text{NrOfLinks}] = [\text{NrInputNeurons}] \cdot [\text{NrHiddenNeurons}] \tag{3.22}$$

there are at least $42 \cdot 3 = 126$ and $6 \cdot 10 = 60$ links, implying the same number of variables, resulting in a time-consuming issue. While for the first prediction method (WiMAX traffic), there are $6 \cdot 2 = 12$ links for 24 hours shifting.

For the $W67$ database we have 8 weeks of time-series. The first model for day prediction uses 1 day from each week. In order to use the GA algorithm, we need 2 different time-series for training. For the calculation of the first part in the fitness function, at the input of the ANN the samples from wavelet transform of the weeks $1^{st}$ till $5^{th}$ were used. As a target, the wavelet transform of the day from week number 6 was used. The second part of the fitness function, was calculated by applying to the network's inputs the time-series from the second training signal: wavelet decomposition of the days starting with $2^{nd}$ till $6^{th}$ week as input, and the wavelet decomposition for the day from the $7^{th}$ week. So, during one generation of a given individual, supposing 10 links between first 2 layers, we perform the following steps (Figure 3.36):

- Take those 10 genes of the individual and apply as weights to the network;
- Input data from weeks $1^{st}$ till $5^{th}$, compare the output with the day from the $6^{th}$ week, retain the first part of the fitness function;

- During the same generation, using the same weights, input data from weeks $2^{nd}$ till $6^{th}$, compare the output with the day from the $7^{th}$ week, retain the second part of the fitness function;
- Compute the final value for the fitness function;
- Apply mutation and crossover to these weights (genes), use their new values (children) in next generation.

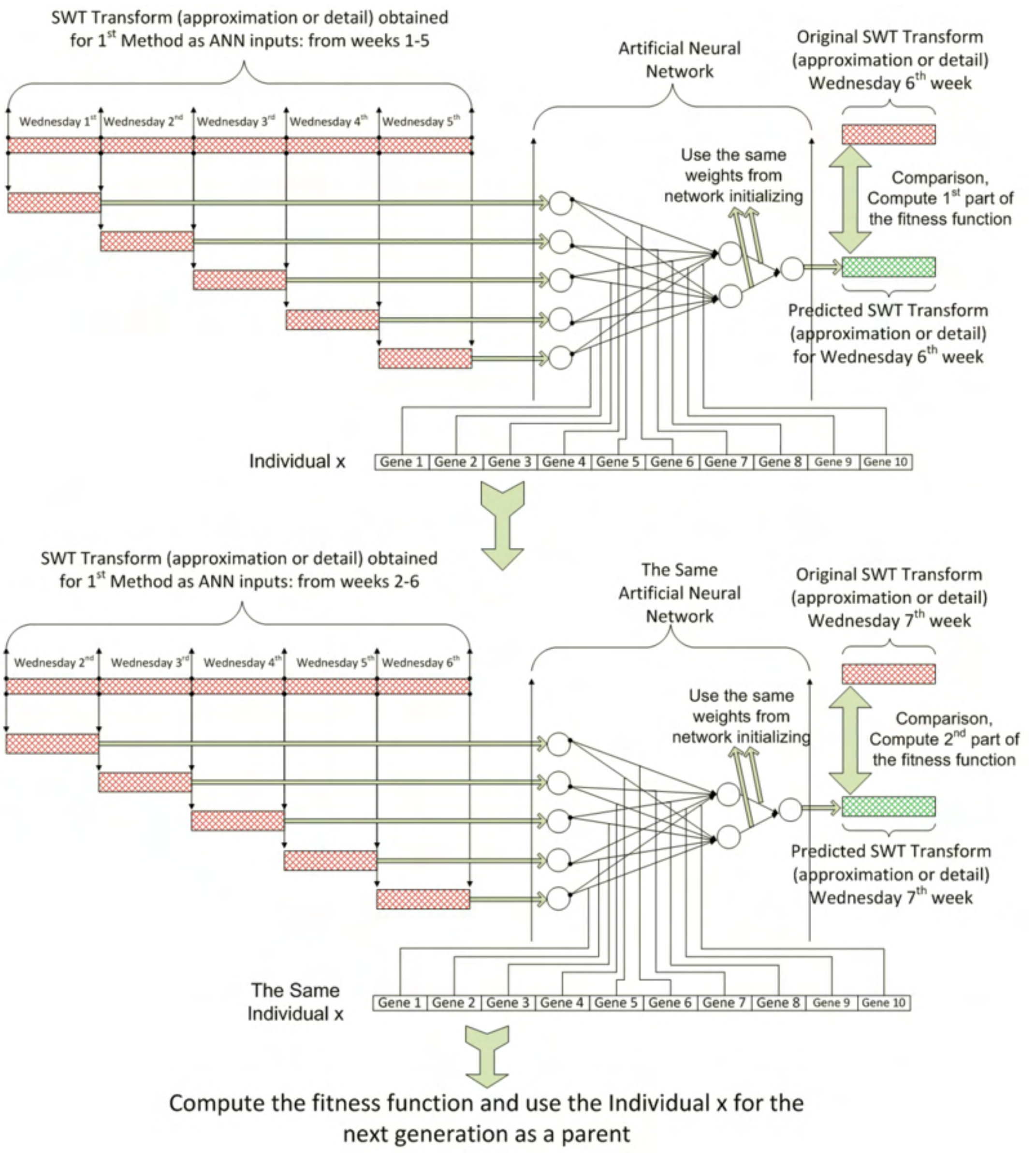

SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs: from weeks 1-5
Artificial Neural Network
Original SWT Transform (approximation or detail) Wednesday 6th week
Wednesday 1st
Wednesday 2nd
Wednesday 3rd
Wednesday 4th
Wednesday 5th
Use the same weights from network initializing
Comparison, Compute 1st part of the fitness function
Predicted SWT Transform (approximation or detail) for Wednesday 6th week
Individual x
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6
Gene 7
Gene 8
Gene 9
Gene 10
SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs: from weeks 2-6
The Same Artificial Neural Network
Original SWT Transform (approximation or detail) Wednesday 7th week
Wednesday 2nd
Wednesday 3rd
Wednesday 4th
Wednesday 5th
Wednesday 6th
Use the same weights from network initializing
Comparison, Compute 2nd part of the fitness function
Predicted SWT Transform (approximation or detail) Wednesday 7th week
The Same Individual x
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6
Gene 7
Gene 8
Gene 9
Gene 10
Compute the fitness function and use the Individual x for the next generation as a parent

Figure 3.36 : Applying Genetic Algorithm on Neural Networks in case of $W67$ database. The signals from the last week will be used for testing.

Note: the input time-series was normalized to have values in the interval $[-1, +1]$.

Regarding the values from those two links between the hidden and output layer, we let them the same as obtained from the net initializing.

The question is why to complicate? Using this GA method for training, ensures us better approximation of the signals. It is like training the same network with two different sets of data, kind of *double-training*, which cannot be accomplished by using the usual Neural Network training.

A comparison between training an ANN using GA method with the usual training of the ANN, is shown in Figure 3.37. It can be observed, that in the case of the GA algorithm, it approximates both signals, because we were able to apply both of them in training process. While with the usual training of the ANN, it approximates just one target signal. Another aspect is that after each startup, the usual ANN training initialized its weights differently, while at the end, the GA algorithm was able to *learn* better the characteristics of the signal.

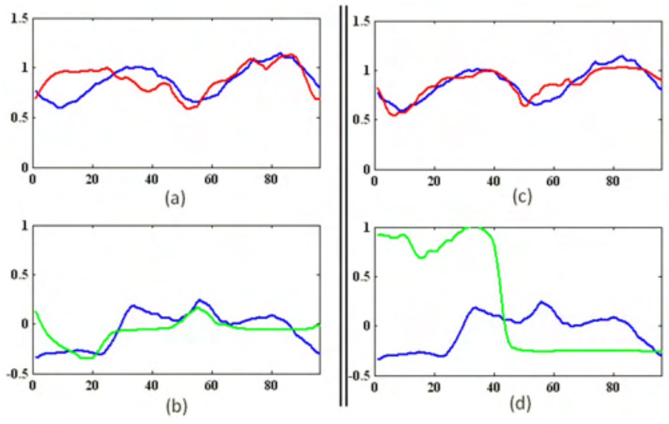


Figure 3.37 : GA Training ((a) and (b)) vs usual ANN Training ((c) and (d)) (for the approximation signals chosen from a random Base Station). The blue signals are the original ones, the red and green signals represent the output of the network after training.

The testing process, shown in Figure 3.38, uses as input data starting from the $3^{rd}$ until the $7^{th}$ weeks. The signal predicted is one taken from the last week of our data.

The using of GA optimization for the $W504$ data set, is almost similar to the one described above. Anyway, some aspects should be pointed out. Two data sets are needed: 2 for training and 1 in forecasting. Because there were only 5 days, the methodology of distributing the information is as follows (Figure 3.39):

- Take the wavelet decomposition from the time-series of days 1 and 2 as ANN inputs, $3^{rd}$ day's series use as a target, compute the first part of the fitness function
- Take the wavelet decomposition from the time-series of days 2 and 3 as ANN inputs, $4^{th}$ day's series use as a target, compute the second part of the fitness function

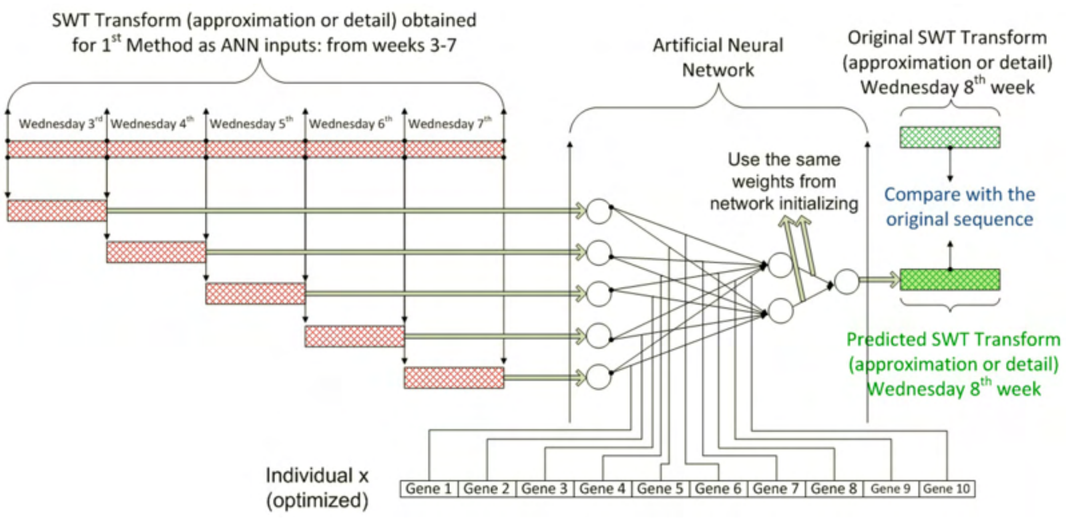

SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs: from weeks 3-7
Wednesday 3rd
Wednesday 4th
Wednesday 5th
Wednesday 6th
Wednesday 7th
Artificial Neural Network
Use the same weights from network initializing
Original SWT Transform (approximation or detail) Wednesday 8th week
Compare with the original sequence
Predicted SWT Transform (approximation or detail) Wednesday 8th week
Individual x (optimized)
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6
Gene 7
Gene 8
Gene 9
Gene 10

Figure 3.38 : Prediction of the signal using optimized Artificial Neural Networks for the $W67$ database.

- After training, use the wavelet decomposition from days 3 and 4, predict the last day (Figure 3.40)

In this case, the minimum number of links between the input and hidden layers were 4 (for 24 hours shifting during neuron decomposition). According to the previous use of GA with the first data set, it results that we had 4 genes which encoded the weights from these links, the 2 weights between the hidden and output layer were let by default during ANN initializing. But, because in this case we had only 4 genes, the 2 weights from the next layer were of a much greater importance. This is the reason why we encoded them also. Finally, 6 genes instead of 4 were used, which encoded all ANN's weights (this example refers to 24 hours shifting, for 12 hours shifting we would have had $6 + 2 = 8$ genes, and so on).

## 3.5 Evaluation of the Prediction Model

The current section is the evaluation of the prediction model (last step from block diagram in Figure 3.19). We reconstruct our signal from the predicted SWT signals and compare it to the original one in order to answer the questions from Section 3.1. We compare our model with other forecasting techniques on different types of time-series in order to analyse the problem (*i1*) (Sections 3.5.3.2 and 3.5.4). Different parameters are changed to meet the requirements of the problem (*i2*) concerned about the change of the settings of the forecasting model (Section 3.5.3). And, a retraining of the ANNs with the incoming time-series is presented in order study the last problem (*i3*), regarding the effectiveness of the model when the time-series changes its characteristics (Sections 3.5.3 and 3.5.4).

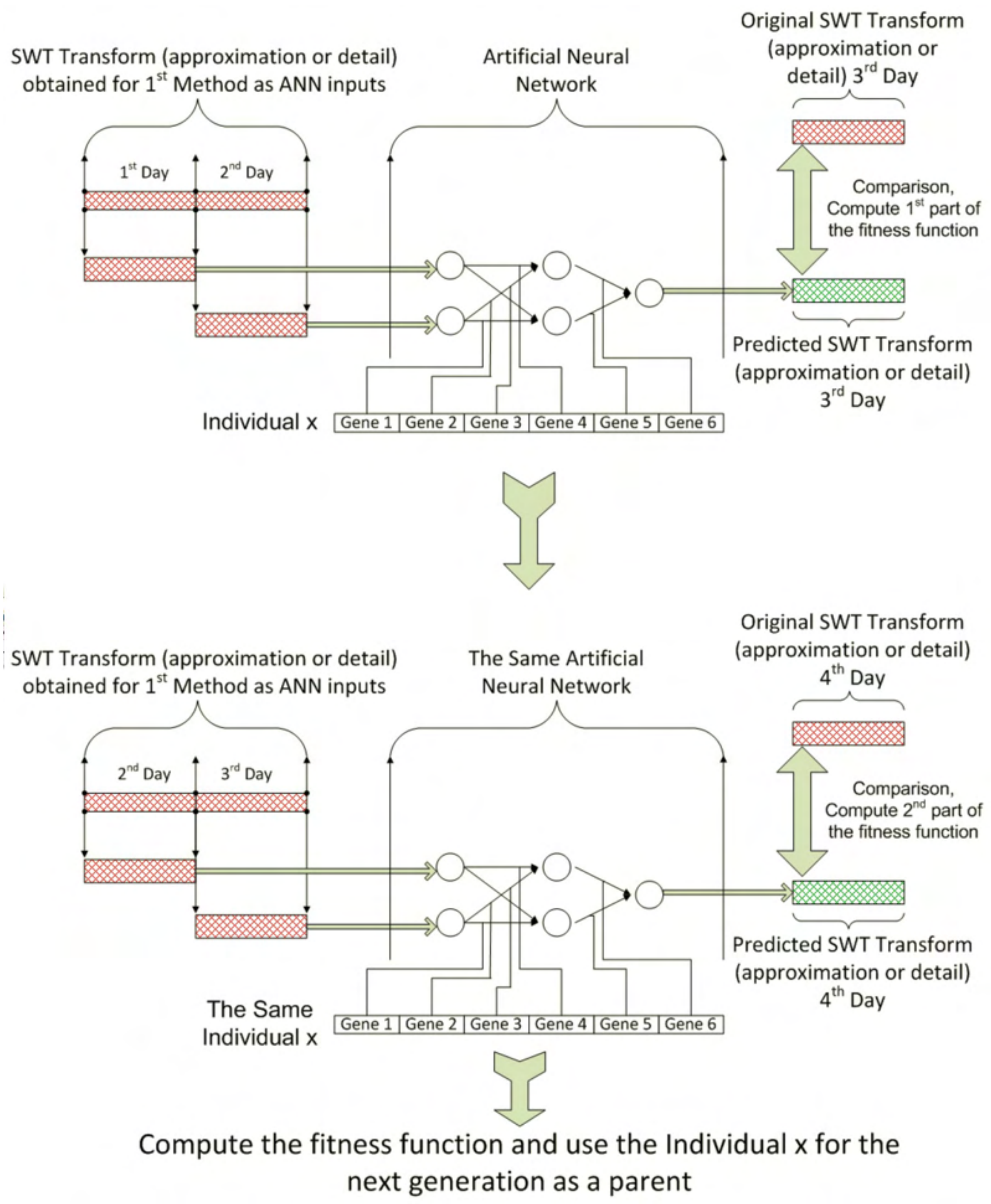

SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs
Artificial Neural Network
Original SWT Transform (approximation or detail) 3rd Day
1st Day
2nd Day
Comparison, Compute 1st part of the fitness function
Predicted SWT Transform (approximation or detail) 3rd Day
Individual x
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6
SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs
The Same Artificial Neural Network
Original SWT Transform (approximation or detail) 4th Day
2nd Day
3rd Day
Comparison, Compute 2nd part of the fitness function
Predicted SWT Transform (approximation or detail) 4th Day
The Same Individual x
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6
Compute the fitness function and use the Individual x for the next generation as a parent

Figure 3.39 : Applying Genetic Algorithm on Neural Networks for the $W504$ database (24 hours shifting).

### 3.5.1 Additional Forecasting Models

In our analysis we made comparisons with other forecasting models and to see the performance of different wavelet families in signal decomposition. The following forecasting models were used for this purpose:

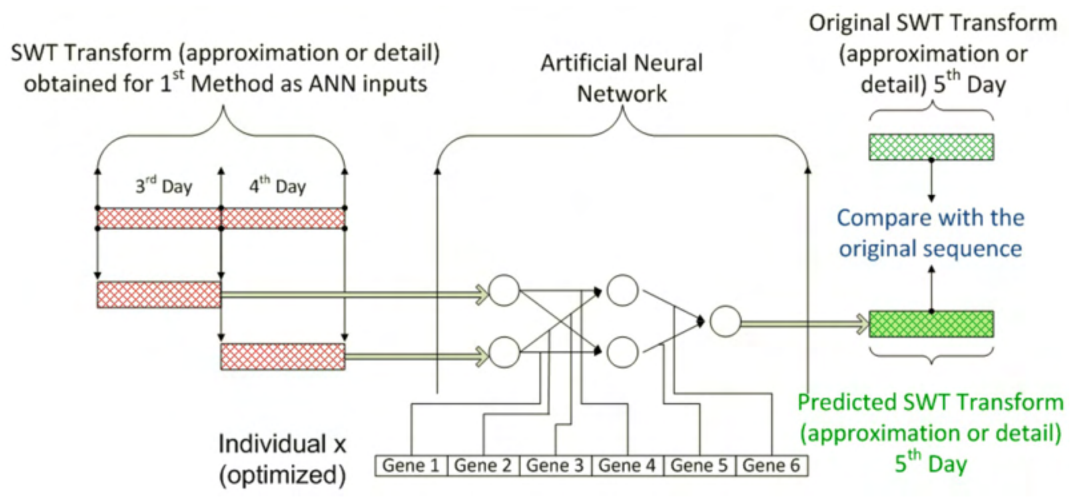

SWT Transform (approximation or detail) obtained for 1st Method as ANN inputs
3rd Day
4th Day
Artificial Neural Network
Original SWT Transform (approximation or detail) 5th Day
Compare with the original sequence
Predicted SWT Transform (approximation or detail) 5th Day
Individual x (optimized)
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6

Figure 3.40 : Prediction of the signal using optimized Artificial Neural Networks for the $W504$ database (24 hours shifting).

- *Linear Regression* (LR): a statistical tool for modelling where the output is a linear combination of inputs:

  $$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

  where $y$ represents the output data, $\beta$ is the weight vector, $\beta_0$ is called bias of the model, and $x$ represents the input data ($x_0 = 1$ for bias). The parameters of the linear regression model are usually estimated using the least-squares method.

- *Random Walk* (RW): it is the simplest model from time-series models and it is characterized by the fact that the changes in the time-series follow a random direction that is unpredictable:

  $$y(t) = y(t-1) + \alpha$$

  *i.e.* the prediction of next values equal the previous values plus the average change one period to the next, $\alpha$. This model assumes that, from one period to the next, the original time-series takes a random *step* away from its last recorded position.

- *ARIMA*: which is based on Box-Jenkins Methodology [Box 70], used to build the time-series model in a sequence of steps which are repeated until the optimum model is achieved.

### 3.5.2 Measuring the Accuracy

The following widely used statistical measures for error that are used to identify a method or the optimum value of the parameter within a method were computed:

- **Mean Absolute Error** (MAE) value is the average absolute error value, closer this value is to zero, the better the forecast is:

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|F_t - X_t| \tag{3.23}$$

where $F_t$ is the prediction and $X_t$ is the true value

- **Mean Square Error** (MSE) with **Root MSE** (RMSE) and its normalized function (NMSE) is a measure of the absolute error, as the prediction accuracy increases, the MSE becomes smaller. If NMSE> 1, it means that the prediction performance is worse than that of the trivial predictor:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{t=1}^{T}(X_t - F_t)^2}{\sum_{t=1}^{T}(X_t - \overline{X_t})^2}} \tag{3.24}$$

- **R-Square** (RSQ): is the coefficient of determination $R^2$, in statistics, is the proportion of variability in a data set that is accounted for by a statistical model. In this definition, the term *variability* is defined as the sum of squares. A version of its calculation is:

$$R^2 = \frac{SS_R}{SS_T} \tag{3.25}$$

where

$$SS_T = \sum_t (X_t - \overline{X_t})^2; \qquad SS_R = \sum_t (F_t - \overline{F_t})^2 \tag{3.26}$$

in which $X_t$, $F_t$ are the original data values and modelled values (predicted) respectively, while $\overline{X_t}$ and $\overline{F_t}$ are the means of the observed data and modelled (predicted) values, respectively. $SS_T$ is the total sum of squares, $SS_R$ is the regression sum of squares. In ideal case $RSQ = 1$.

- **Mean Absolute Percent Error** (MAPE): calculates the mean absolute error in percent between the real and forecasted signals:

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{X_t - F_t}{X_t}\right| \times 100\% \tag{3.27}$$

- **Symmetric Mean Absolute Error** (SMAPE): calculates the symmetric absolute error in percent between the actual $X$ and the forecast $F$ across all observations $t$ of the test set of size $T$:

$$SMAPE = \frac{1}{T}\sum_{t=1}^{T}\frac{|X_t - F_t|}{(X_t + F_t)/2} \tag{3.28}$$

### 3.5.3 WiMAX Traffic ($W67$ and $W504$ databases)

#### 3.5.3.1 One Day Prediction

For the *One Day Prediction model* we analysed the issue (*i2*) defined in Section 3.1, *i.e.* finding better parameters for a better prediction result.

We could select the following parameters: the BS number (from 67 or 504 possible), the number of sample per day (16, 32, or 96), the number of hours shifted (4, 8, 12, or 24), the day of the week (from Monday till Sunday), and between applying or not the GA optimization. Previously, the configuration of the ANN was analysed also (Subsections 3.4.4 and 3.4.5). We compared the performances of the model 1 and model 2 for day forecasting, and the performance of the ANN optimization using GA. After combining the above possibilities and eliminating the erroneous data, we considered $8,232$ final number of simulations. The results are presented in Tables 3.3 and 3.4.

Table 3.3 : Medium $RSQ$ and $SMAPE$, Day Prediction.

| Model | Hours shifted | Measured value | 96 samples | 32 samples | 16 samples |
|---|---|---|---|---|---|
| Similar Days | 4 hours | RSQ | 1.212 | 1.249 | 1.317 |
| | | SMAPE | 0.905 | 0.927 | 0.802 |
| | 8 hours | RSQ | 1.159 | 1.120 | 1.205 |
| | | SMAPE | 0.886 | 0.858 | 0.756 |
| | 12 hours | RSQ | 1.330 | 1.219 | 1.287 |
| | | SMAPE | 0.904 | 0.924 | 0.814 |
| | 24 hours | RSQ | 1.178 | 1.201 | 1.263 |
| | | SMAPE | 0.910 | 0.946 | 0.780 |
| All Days | 4 hours | RSQ | 0.715 | 0.739 | 0.711 |
| | | SMAPE | 0.868 | 0.843 | 0.732 |
| | 8 hours | RSQ | 0.820 | 0.782 | 0.792 |
| | | SMAPE | 0.861 | 0.849 | 0.720 |
| | 12 hours | RSQ | 0.634 | 0.767 | 0.690 |
| | | SMAPE | 0.866 | 0.857 | 0.738 |
| | 24 hours | RSQ | 0.741 | 0.781 | 0.703 |
| | | SMAPE | 0.869 | 0.876 | 0.752 |

By comparing the results from these tables, we can see that applying second model for day prediction (all days selection), gave us better results with about $2-6\%$ in comparison with the first forecasting model (*Similar Days Selection*). Regarding the use of the genetic optimization for ANN, we can see that the results for $SMAPE$ are worse with about 10% compared to those obtained using ordinary ANN training, But the RSQ values are closer to the ideal 1 when we applied the genetic optimization.

Speaking about the number of samples per day, the best result was obtained when using 16[Samples/Day]. But in this case, the forecasting technique was not able to predict the sudden increasing and the peaks of our WiMAX traffic. This was because of the way the 16 samples per day were obtained: by making a medium value from other 6 consecutive

Table 3.4 : Medium *RSQ* and *SMAPE*, *Similar Days Selection* Model, using Genetic Algorithms optimization.

| Hours shifted | Measured value | 96 samples | 32 samples | 16 samples |
|---|---|---|---|---|
| 4 hours | RSQ | 1.127 | 1.200 | 1.176 |
| | SMAPE | 1.001 | 0.913 | 0.867 |
| 8 hours | RSQ | 1.108 | 1.129 | 1.099 |
| | SMAPE | 0.983 | 0.924 | 0.820 |
| 12 hours | RSQ | 1.211 | 1.155 | 1.168 |
| | SMAPE | 1.008 | 0.951 | 0.851 |
| 24 hours | RSQ | 1.087 | 1.189 | 1.215 |
| | SMAPE | 1.067 | 0.950 | 0.872 |

original samples. Regarding this aspect, 96 samples per day were better to be used instead. In both cases, the optimal number of shifted hours is 8.

#### 3.5.3.2 Week Prediction

Using *Week Prediction* we analyse the issues (*i1*) – concerned about a model to be used for different time-series characteristics; (*i2*) – finding better parameters of the model; (*i3*) – answering if the upgrading of the model with the latest information improves its performance.

The problem (*i1*) was studied using 2 time-series from different domains: *WiMAX* network traffic and *EUR/USD* currency exchange. A comparison of the proposed model using neural networks and existing models (*i.e.* linear regression, random walk, ARIMA) was performed. For the problem (*i2*) the following parameters have been changed: the number of samples per day, the number of shifted days for the ANN input, the type of wavelet. The configuration of the ANN was analysed earlier (Subsection 3.4.4.3). In order to answer the last question (*i3*), we used for the proposed model different input series. Thus, the first method using ANNs, *i.e. ANN No Sliding*: we train the networks once for each decomposition level, as described in subsections 3.4.3.3 and 3.4.4.3. For inputs, we have the first $(n-2k)$ weeks, where $n$ is the total number of weeks, and $k$ is the number of weeks we want to forecast (*i.e.* 1 in this case). The target consists of the data taken from the weeks $(n-2k+1)$ to $(n-k)$. The data used for ANN's input during the testing phase is the information from the weeks $(k+1)$ to $(n-k)$. The output signal is compared to the real data of the last $k$ weeks. The next method, ANN *Known Sliding*, uses sliding, retraining the network with the real information. The entire signal is divided into smaller parts. Each of these sequences will predict a small part of the final forecasted signal. The information for networks' retraining is always taken from the real data. The last method, ANN *UnKnown Sliding*, proposes a forecasting using sliding with unknown data. The only difference consists in the fact that the information used for the next simulation and retraining is taken not from the original signal, but from the previously predicted one.

We had $1,449$ simulations by changing the BS number from the $W67$ database, the number of samples per day, the number of shifted days, and after eliminating the erroneous data. The results for $RSQ$ and $SMAPE$ are presented in Tables 3.5 and 3.6.

Table 3.5 : Medium $RSQ$ and $SMAPE$ by comparing the number of shifted days in one week *No Sliding* prediction model for the $W67$ database.

| Samp/Day | Shifted Days | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 96 | RSQ | 1.243 | 1.195 | 1.659 | 1.281 | 1.308 | 1.077 | 1.252 |
| | SMAPE | 0.895 | 1.036 | 1.057 | 1.247 | 1.081 | 0.819 | 0.894 |
| 32 | RSQ | 1.318 | 1.276 | 1.415 | 1.392 | 1.401 | 1.106 | 1.380 |
| | SMAPE | 0.992 | 1.000 | 0.989 | 1.129 | 1.001 | 0.857 | 0.896 |
| 16 | RSQ | 1.663 | 1.397 | 1.891 | 1.805 | 1.742 | 1.215 | 1.712 |
| | SMAPE | 0.840 | 1.012 | 0.961 | 1.093 | 1.036 | 0.845 | 0.793 |

Table 3.6 : Comparison between the three models based on ANN.

| ANN Type | $RSQ$ | $SMAPE$ | $MAPE$ | $RMSE$ | $MAE$ |
|---|---|---|---|---|---|
| No Sliding | 1.137 | 0.946 | 48,728 | 1.430 | 0.602 |
| Known Sliding | 0.960 | 1.006 | 25,218 | 1.399 | 0.573 |
| UnKnown Sliding | 0.845 | 1.052 | 33,783 | 1.738 | 0.648 |

From the results we can notice that the best performance is obtained using one or six days shifting. Also, the best solutions for forecasting were obtained using 2 or 6 days shifting with 16[Samples/Day]. Anyway, as in the one day prediction model, using 16 samples do not give the peaks of the signals which could results in system failures in case of a very high users' demand.

Regarding the WT, we propose various types of mother wavelets such as Daubechies (db), Coiflet (coif), Symlet (sym), Biorthogonal (bior), and Reverse Biorthogonal (rbio). In Table 3.7 we present the results. $MAE$, $MSE$, $RSQ$, $SMAPE$, and $RMSE$ are the average value for all 67 Base Stations for 2 weeks forecasting from ANN and RW prediction. While $SMAPE\ L$, $MAPE\ L$, and $MAE\ L$ are the average value corresponding to the ANN, RW, LR, and ARIMA obtained from the mean of the original signal and the mean of the forecasted signal, because ARIMA and LR cannot be used to obtain forecasts for every moment of time as ANN and RW do. For linear models the trajectory of the forecasts is represented through a sloping line which represents the weekly increase.

According to the results, the wavelet of Haar (db1), which is the simplest of the Daubechies family and rbio1.1 give the best prediction performance. The results also indicate that with the increase of the filters' length (support of the mother wavelets), the performance of the wavelet transform deteriorates.

A comparison between all the mentioned forecasting models is presented in Table 3.8. The db1 mother wavelet was used. The results prove that ANN performs better than other prediction techniques. The linear regression model gives good forecasting results also.

Table 3.7 : Comparison between wavelets in week prediction, $W67$ WiMAX traffic database.

| Wavelet | $RSQ$ | $SMAPE$ | $MAPE$ | $MSE$ | $RMSE$ | $MAE$ | $SMAPEL$ | $MAPEL$ | $MAEL$ |
|---|---|---|---|---|---|---|---|---|---|
| coif 1 | 1.4450 | 1.0900 | 0.2113 | 11.720 | 2.8000 | 1.0304 | 0.8900 | 0.0020 | 0.9599 |
| coif 2 | 1.4930 | 1.2200 | 0.2285 | 12.950 | 2.8300 | 0.8748 | 0.8370 | 0.0019 | 0.7191 |
| db 1 | 1.1680 | 1.0800 | 0.2367 | 8.0600 | 2.4300 | 0.7685 | 0.8120 | 0.0016 | 0.7327 |
| db 2 | 1.3640 | 1.1500 | 0.2451 | 10.520 | 2.6900 | 0.8408 | 0.8550 | 0.0019 | 0.7768 |
| db 3 | 1.3580 | 1.1200 | 0.2117 | 9.7600 | 2.6400 | 0.8193 | 0.8570 | 0.0018 | 0.7678 |
| db 4 | 1.490 | 1.1100 | 0.2159 | 10.610 | 2.5800 | 0.7985 | 0.8340 | 0.0018 | 0.7563 |
| db 5 | 1.4350 | 1.1100 | 0.2190 | 12.560 | 2.7500 | 0.8339 | 0.8230 | 0.0019 | 0.7730 |
| bior 3.1 | 0.6950 | 1.1300 | 0.3152 | 9.8600 | 2.5200 | 0.8402 | 0.860 | 0.0018 | 0.7071 |
| rbio 1.1 | 1.2000 | 1.0800 | 0.2215 | 10.000 | 2.6100 | 0.7948 | 0.8200 | 0.0017 | 0.8947 |
| rbio 2.2 | 1.4820 | 1.1900 | 0.3202 | 10.290 | 2.7100 | 0.8747 | 0.8910 | 0.0018 | 0.7690 |
| rbio 3.3 | 1.9520 | 1.2100 | 0.2623 | 10.330 | 2.8800 | 0.9509 | 0.9070 | 0.0022 | 1.0690 |
| sym 2 | 1.3650 | 1.2600 | 0.2146 | 13.200 | 2.8900 | 0.8854 | 0.8950 | 0.0019 | 0.7412 |

Table 3.8 : Comparison between the forecasting techniques for 2 weeks prediction, $W67$ WiMAX traffic database.

| Forecasting Model | $SMAPE\ L$ | $MAPE\ L$ | $MAE\ L$ |
|---|---|---|---|
| ANN No Sliding | 0.472 | 0.0011 | 0.4428 |
| ANN Known Sliding | 0.509 | 0.0009 | 0.4241 |
| ANN UnKnown Sliding | 0.722 | 0.0017 | 0.6681 |
| ARIMA | 0.772 | 0.0027 | 0.9990 |
| Linear Regression | 0.523 | 0.0031 | 0.3868 |
| Random Walk with Wavelets | 4.440 | 0.0030 | 1.3633 |

### 3.5.4 EUR/USD Currency Exchange ($EUVol$ and $EURate$ databases)

For EUR/USD currency pair 2 weeks ahead forecasting was performed. The wavelets' results for the volume data are shown in Table 3.9. The best forecasting performance is obtained using the mother wavelets coif2 and sym2.

The comparison between the models is shown in Table 3.10. Coif2 mother wavelet was used. Best results are obtained with ANN and LR. However, we should apply LR model for long time interval prediction if we are interested in the tendency of the signal, and not in possible peaks.

The wavelets' results for the EUR/USD exchange rate data are shown in Table 3.11. According to the results, no preferable wavelet transform was found for this data set.

The comparison between the models is shown in Table 3.12. Db1 mother wavelet was used. The smallest error value have been obtained using neural networks (No Sliding). Its performance is much higher for this data set than the results using other forecasting techniques. In comparison to the previous data sets, the LR model cannot be used for long time interval forecasting in this case.

Concluding, we can observe that the proposed model using ANN outperforms other forecasting ones for different time-series characteristics (*i1*). The better performance after

Table 3.9 : Comparison between wavelets in week prediction, $EUVol$ database.

| Wavelet | $RSQ$ | $SMAPE$ | $MAPE$ | $MSE$ | $RMSE$ | $MAE$ | $SMAPEL$ | $MAPEL$ | $MAEL$ |
|---|---|---|---|---|---|---|---|---|---|
| coif 1 | 0.6880 | 0.5560 | 0.1152 | 1.6270 | 1.2200 | 0.3586 | 0.5160 | 0.0821 | 0.4240 |
| coif 2 | 0.4550 | 0.5220 | 0.0793 | 1.089 | 1.0380 | 0.2967 | 0.4530 | 0.0732 | 0.3799 |
| db 1 | 0.6250 | 0.5200 | 0.0839 | 1.3560 | 1.1260 | 0.3175 | 0.4540 | 0.0713 | 0.3690 |
| db 2 | 0.7150 | 0.5780 | 0.1088 | 1.6100 | 1.2190 | 0.3550 | 0.4970 | 0.0812 | 0.4199 |
| db 3 | 0.5860 | 0.5850 | 0.1156 | 1.4990 | 1.1880 | 0.3618 | 0.5310 | 0.0864 | 0.4461 |
| db 4 | 0.871 | 0.6000 | 0.1114 | 1.6040 | 1.2390 | 0.3745 | 0.5270 | 0.0863 | 0.4459 |
| db 5 | 0.808 | 0.587 | 0.1121 | 1.5570 | 1.2250 | 0.3700 | 0.54600 | 0.0912 | 0.4710 |
| bior 3.1 | 0.6280 | 0.5340 | 0.1137 | 1.5520 | 1.1730 | 0.3390 | 0.4330 | 0.0712 | 0.3681 |
| rbio 1.1 | 0.6150 | 0.5190 | 0.0958 | 1.2860 | 1.0960 | 0.3208 | 0.4570 | 0.0716 | 0.3704 |
| rbio 2.2 | 0.5410 | 0.5550 | 0.1087 | 1.4400 | 1.1490 | 0.3406 | 0.4910 | 0.0790 | 0.4081 |
| rbio 3.3 | 0.7720 | 0.5950 | 0.0993 | 1.4180 | 1.1670 | 0.3480 | 0.4550 | 0.0716 | 0.3705 |
| sym 2 | 0.4760 | 0.4990 | 0.0890 | 1.1170 | 1.0370 | 0.2962 | 0.4530 | 0.0736 | 0.3813 |

Table 3.10 : Comparison between the forecasting techniques for 2 weeks prediction, $EUVol$ database.

| Forecasting Model | $SMAPE\ L$ | $MAPE\ L$ | $MAE\ L$ |
|---|---|---|---|
| ANN No Sliding | 0.1690 | 0.0200 | 0.1079 |
| ANN Known Sliding | 0.1530 | 0.0178 | 0.9570 |
| ANN UnKnown Sliding | 0.2670 | 0.0344 | 0.1812 |
| ARIMA | 1.1350 | 0.3245 | 1.7054 |
| Linear Regression | 0.1910 | 0.0243 | 0.1109 |
| Random Walk with Wavelets | 0.9400 | 0.2670 | 1.4173 |

Table 3.11 : Comparison between wavelets in week prediction, $EURate$ database.

| Wavelet | $RSQ$ | $SMAPE$ | $MAPE$ | $MSE$ | $RMSE$ | $MAE$ | $SMAPEL$ | $MAPEL$ | $MAEL$ |
|---|---|---|---|---|---|---|---|---|---|
| coif 1 | 1.4612 | 0.0087 | 0.6698 | 11.733 | 3.2000 | 0.0087 | 0.2745 | 0.3149 | 0.4105 |
| coif 2 | 2.4350 | 0.0089 | 0.6876 | 10.296 | 3.0502 | 0.0090 | 0.2739 | 0.3135 | 0.4088 |
| db 1 | 0.5011 | 0.0093 | 0.7199 | 11.078 | 3.1762 | 0.0094 | 0.2675 | 0.3025 | 0.3945 |
| db 2 | 0.9468 | 0.0112 | 0.8723 | 12.636 | 3.4910 | 0.0114 | 0.2772 | 0.3179 | 0.4145 |
| db 3 | 1.7864 | 0.0071 | 0.5491 | 8.3235 | 2.6839 | 0.0072 | 0.2725 | 0.3123 | 0.4073 |
| db 4 | 3.9849 | 0.0069 | 0.5308 | 10.6729 | 2.9384 | 0.0070 | 0.2726 | 0.3128 | 0.4079 |
| db 5 | 3.3660 | 0.0068 | 0.5255 | 10.6936 | 2.8954 | 0.0069 | 0.2731 | 0.3137 | 0.4090 |
| bior 3.1 | 2.4196 | 0.0085 | 0.6609 | 11.4006 | 3.2603 | 0.0086 | 0.2727 | 0.3119 | 0.4066 |
| rbio 1.1 | 0.5584 | 0.0103 | 0.7971 | 11.3382 | 3.2730 | 0.0104 | 0.2681 | 0.3030 | 0.3951 |
| rbio 2.2 | 0.8704 | 0.0080 | 0.6172 | 10.6246 | 3.0167 | 0.0080 | 0.2722 | 0.3112 | 0.4058 |
| rbio 3.3 | 0.7793 | 0.0087 | 0.6723 | 10.6935 | 3.0813 | 0.0088 | 0.2652 | 0.2991 | 0.3897 |
| sym 2 | 0.9531 | 0.0089 | 0.6893 | 11.0653 | 3.1447 | 0.0090 | 0.2758 | 0.3168 | 0.4131 |

parameters' selection was obtained using $db1$ wavelet with a 2 or 6 days shifting of the time-series for the ANN inputs with 16 samples per day in case of *WiMAX* time-series.

Table 3.12 : Comparison between the forecasting techniques for 2 weeks prediction, *EURate* database.

| Forecasting Model | $SMAPE\ L$ | $MAPE\ L$ | $MAE\ L$ |
|---|---|---|---|
| ANN No Sliding | 0.00524 | 0.40321 | 0.00536 |
| ANN Known Sliding | 0.01244 | 0.95650 | 0.01258 |
| ANN UnKnown Sliding | 0.00957 | 0.72781 | 0.00965 |
| ARIMA | 0.02159 | 1.63279 | 0.02163 |
| Linear Regression | 1.17423 | 56.4945 | 0.74656 |
| Random Walk with Wavelets | 0.08567 | 6.17345 | 0.08111 |

For the *EUR/USD* time-series better prediction performance was obtained using $coif2$ and $sym2$ wavelets. Regarding the third problem (*i3*) we have shown that a constant upgrading and retraining of the model improves its performance when a change of time-series characteristics is observed for *WiMAX* and *EUVol* , as we have obtained a higher prediction accuracy with *ANN Known Sliding* modification. In case of *EURate* time-series this type of upgrading diminished the performance, where the *ANN No Sliding* modification performed better than *ANN Known Sliding*.

## 3.6 Conlcusions

The searching for models used in forecasting the future evolution of a time-series has gained large interest lately. In this chapter we proposed a time-series forecasting model in order to answer the following questions: finding a model that could be used in a time-series forecasting for different series characteristics (*i1*); the selection of parameters of the proposed approach and their influence on the final prediction performance (*i2*); and the influence of the constant upgrading with the latest available information and retraining the model on the forecasting *(i3)*.

The proposed forecasting model was built by using neural networks, wavelet transform, and genetic algorithms (in order to optimize the neural network training). In order to answer the question *(i1)* two types of data was used from different domains: WiMAX traffic and EUR/USD currency exchange. A comparison between our approach, ARIMA, Linear Regression, Random Walk was done. The prediction performance was calculated using the following performance errors: $SMAPE$, $RSQ$, $MAPE$, $RMSE$, $MAE$. For both data-types our results have shown that the proposed model outperforms other existing algorithms in forecasting. However, if we are interested in the tendency of the analysed data, then ARIMA and Linear Regression models are the one to be taken into consideration.

For the second question *(i2)* the following parameters were changed and verified: the type of wavelet function, the time-shifting of the data used for the inputs of the neural networks, the predicted time-interval, the type of the ANN training (using or not the optimization with genetic algorithms). Best results were obtained using the simplest Haar mother wavelet (*i.e. db1*), especially those wavelets with good time-frequency localization which have a reduced number of vanishing moments, such as *rbio1.1* or *db3*. For the input time-shifting interval best results were obtained using 8-hours shifting in case of day prediction (for WiMAX data), with an increased performance of $1\% - 3\%$ . In case of week prediction, better results are obtained using 1, 2 and 6 days shifting, with an increased performance of $5\% - 15\%$. The genetic algorithms were used to find out the optimal values for ANN's weights. This optimization increased the $SMAPE$ error with $2\% - 7\%$, but decreased the $RSQ$ error, setting it closer to the ideal value of 1.

In case of the third question *i3*, we have compared the prediction performance of our approach when we retrain the algorithm with newest available information. In case of WiMAX and EUR/USD volume databases, the best performance was obtained using this approach, however in case of $SMAPE$ and $MAE$ calculation, the results were comparable with those from using Linear Regression model. In case of EUR/USD exchange rate the best results were obtained when not using the retraining process, and the Linear Regression model was the worst performer from all tested algorithms.

# Conclusions and Perspectives

In our work we propose forecasting models for 2 types of sequential data that consider the temporal interdependencies of its items: *rules/patterns* from a *sequential database*, and *time-series*.

For sequential rules selection we introduced the *Closeness Preference* measure. Its goal is to advantage those rules whose antecedent and consequent are as close as possible. A weighting function is used that has as parameters the time-distance between the itemsets, the slope $s$ and a maximum allowed time-window $\omega_t$, where $s$ and $\omega_t$ are set by the user according to his preferences. The measure satisfies the $2^{nd}$ and $3^{rd}$ properties of Piatetsky-Shapiro, showing its quality factors of coverage. It has been tested on a web log database having as a goal to predict the future user's request after analysing previous ones. The following parameters were set for $CP$: time-window $\omega_t = \{360, 420\}$ seconds, and the slope $s = \{1.05, 1.25\}$. The threshold value varied from 0.4 to 1.0 containing a value $\theta^*$ corresponding to a recommended threshold value. The results have shown that using $CP$ measure we are able to select those rules which give high *Accuracy*, *Precision*, *Recall* and *F1-Score* in a forecasting process (up to 0.89). We compared the prediction performance of the extracted simple rules (*i.e.* the rules of the form $V_i \rightarrow V_n$) with complex rules (*i.e.* the rules of the form $V_1V_2...V_{n-1} \rightarrow V_n$). Using simple rules a higher *Accuracy*, *Precision* and *F1-Score* (but a lower *Recall*) was obtained, proving an existence of a simple and accurate predictive model.

For sequential patterns mining, 2 *Modified CP* measures were introduced, that also advantage the time-closeness between the itemsets of a pattern. These measures are used in the pre-processing step of the *Generalized Sequential Pattern* extraction algorithm instead of *Support* measure. The first measure based on *Support* and *Confidence*, *i.e.* $MCP_{sc}$, does not satisfy the *a-priori* principle in comparison to the second one, *i.e.* $MCP_{s\ func}$, that contains another weighting function in its computation. A comparison of the extracted patterns after applying these 2 measures and *Support* is presented on a web log database in order to see the differences of extracted patterns. The results show a number of common patterns between the 2 measures and *Support* of around $70\% - 85\%$ from the total number of compared patterns. Higher similarity with *Support* is experienced by $MCP_{s\ func}$. This is explained by its anti-monotone property, setting it "closer" to *Support*. In comparison to *Support*, the $MCP_{sc}$ and $MCP_{s\ func}$ measures advantage shorter patterns. This is due to the fact that these patterns have higher probability of having shorter time-intervals between the itemsets in comparison to longer ones. The extracted patterns using the proposed measures could be used to forecast the short time-range events. For example, in case of market basket data if one buys the item $A$ followed by an item $B$ during a time-interval $\omega_t$, the manager might find a beginning of a time-closeness pattern $A \rightarrow B \rightarrow C$. In this case it will be supposed that the buyer will come after the item $C$ within a time-window $\omega_t$. However, if the item $B$ is bought within an interval much further than $\omega_t$, then the mentioned pattern will no longer be applied.

For the time-series analysis, we proposed a forecasting model in order to study and answer the following issues: finding a model that could be used in forecasting of the time-series with different characteristics (*i1*); to obtain a better forecasting performance after the change of model's parameters (*i2*); to see if the constant upgrading of the model with the latest information from a changing time-series characteristics improves its prediction accuracy. The proposed model was built using *Artificial Neural Networks*, *Stationary Wavelet Transform*, and *Genetic Algorithms*. As databases for analysis we used *WiMAX* network traffic (day and week forecasting), and *EUR/USD* currency exchange volume and rate (week forecasting). In order to test our model by analysing the first problem (*i1*) comparisons with Linear Regression, Random Walk, and ARIMA models were performed. The prediction performance was evaluated using $SMAPE$, *RSQ*, $MAPE$, $RMSE$, and $MAE$ errors. In case of *WiMAX* traffic a performance higher between $5\% - 35\%$ was obtained by the proposed model. For the $EUVol$ time-series a performance around 15% higher was seen by our model, and a much higher performance for the *EURate* series. For both types of data we have shown that the proposed model outperforms the existing ones. The second problem (*i2*) was studied changing the following parameters: the number of the base station, the types of days for analysis (first or second model for day forecasting), the number of samples per day, the number of hours shifted (in case of day prediction) or days shifted (in case of week selection), the wavelet transform used, and between applying or not the GA optimization (for day prediction). We have performed day prediction for *WiMAX* series and week prediction for *WiMAX* and *EUR/USD* series. The presented results were obtained after computing a medium value from of the prediction from all base stations. In case of day forecasting, best performance was obtained using 8 hours shifting with 16 samples per day. Between the first and the second model for day prediction, the second one gave a better result with about $2\% - 6\%$. The GA optimization did not give a better value for $SMAPE$ (worse with $2 - 7\%$), but a better value for *RSQ* (with about 5%). In case of week forecasting, best results were obtained using 1, 2 and 6 days shifting, with the *Haar* wavelet in case of *WiMAX*, and *coif2* and *sym2* in case of *EUR/USD*. Summing up the above results, we say that a better performance is obtained using the second model for day prediction, genetically optimized neural networks; 16 samples per day (resulting in 3 levels wavelet transform); 8 hours shifting for day prediction and 1, 2 or 6 days shifting for week prediction; with *Haar*, *coif2* or *sym2* wavelets. Another aspect is that we need different configurations for different time-series characteristics in order to obtain a better prediction performance. For the question (*i3*) the prediction performance of the proposed model was compared when we did and didn't retrain the model with new available series. For *WiMAX* and *EUVol* databases the results have shown an improvement after retraining. For *WiMAX* traffic an increase of $MAE$ and $MAPE$ with 4% was observed, but a decreasing in $SMAPE$ with 7%. For $EUVol$ time-series it was $SMAPE$ and $MAPE$ which gave better results (10%), a worse value for $MAE$ (more than 10%). For *EURate* time-series the retraining process didn't prove itself to be a good solution. All the performance parameters worsened twice.

Despite of the above results, it should be kept in mind that different settings and configurations of the proposed models must be tested in case of the different sequential databases and time-series characteristics.

As perspectives, one of our current work in progress consists in adapting the proposed interestingness measures for extraction of sequential patterns for time-series analysis and forecasting and to compare them with the proposed $ANN$ model. Another goal consists in verifying the prediction performance for a short-range forecasting using the $MCP_{sc}$ and $MCP_{s\ func}$ measures. A way of selecting the time-window parameters $\omega_t$ and the slope of the function $s$ to the user should be given according to different types of sequential databases and time-series.

# Bibliography

[abd T. Nochai 06] R. Nochai abd T. Nochai. *ARIMA model for forecasting oil palm price.* In Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statisticd and Applications, pp. 1–7, Universiti Sains Malaysia, Penang, 2006. 2, 50

[Abrahart 98] R.J. Abrahart & L. See. *Neural Network vs. ARMA Modelling: constructing benchmark case studies of river flow prediction.* In Proceedings of the Third International Conference on GeoComputation, September 1998. 2, 50

[Abramovich 00] F. Abramovich, T.C. Bailey & T. Sapatinas. *Wavelet analysis and its statistical applications.* The Statistician, vol. 49, no. 1, pp. 1–29, 2000. 53

[Adya 00] M. Adya & J.S. Armstrong. *An application of rule-based forecasting to a situation lacking domain knowledge.* International Journal of Forecasting, vol. 16, no. 4, pp. 477–484, 2000. 49

[Aggarwal 09] S. K. Aggarwal, L. M. Saini & A. Kumar. *Electricity price forecasting in deregulated markets: A review and evaluation.* Electrical Power and Energy Systems, vol. 31, no. 1, pp. 13–22, 2009. 1

[Agrawal 93] R. Agrawal, T. Imielinksi & A. Swami. *Mining Association Rules between Sets of Items in Large Databases.* In Proceedings of the 1993 ACM SIGMOD International Conference on Management of data, pp. 207–216, 1993. 1

[Agrawal 94] R. Agrawal & R. Srikant. *Fast Algorithms for Mining Association Rules in Large Databases.* In Jorge B. Bocca, Matthias Jarke & Carlo Zaniolo, editors, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, Santiago de Chile, Chile, pp. 487–499. Morgan Kaufmann, 1994. 1, 7, 8, 128

[Agrawal 95] R. Agrawal & R. Srikant. *Mining Sequential Patterns.* In Philip S. Yu & Arbee L. P. Chen, editors, Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, pp. 3–14. IEEE Computer Society, 1995. 1, 2, 5, 128

[Agrawal 98] R. Agrawal, J. Gehrke, D. Gunopulos & P. Raghavan. *Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications.* In Proceedings of the International Conference on Management of Data, volume 27, pp. 94–105, 1998. 1

[Ahn 03] C.W. Ahn & R.S. Ramakrishna. *Elitism-based compact genetic algorithms.* IEEE Transactions on Evoluationary Computation, vol. 7, no. 4, pp. 367–385, 2003. 68

[Allen 83] J. F. Allen. *Maintaining knowledge about temporal intervals.* Communications of the ACM, vol. 26, no. 11, pp. 832–843, November 1983. 2

[Amphawan 10] K. Amphawan, A. Surarerks & P. Lenca. *Mining Periodic-Frequent Itemsets with Approximate Periodicity Using Interval Transaction-Ids List Tree.* In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 245–248, 2010. 127, 132

[Amphawan 12] K. Amphawan, P. Lenca & A. Surarerks. *Mining top-k regular-frequent itemsets using database partitioning and support estimation.* Expert Systems With Applications, vol. 39, pp. 1924–1936, 2012. 6

[Antunes 01] C. M. Antunes & A. L. Oliveira. *Temporal data mining: An overview.* In KDD Workshop on Temporal Data Mining, 2001. 2

[Armstrong 06] J. Armstrong. *Findings from Evidence-based Forecasting: Methods for Reducing Forecast Error.* International Journal of Forecasting, vol. 22, pp. 583–598, 2006. 49

[Beasley 97] J. S. Beasley, A. W. Righter, C. J. Apodaca, S. Pour-Mozafari & D. Huggett. *$I_{DD}$ Pulse Response Testing Applied to Complex CMOS ICs.* In Proceedings of the International Test Conference, Washington, DC, USA, 1997. IEEE COmputer Society Press. 59

[Bettini 98a] C. Bettini, X. S. Wang & S. Jajodia. *Mining Temporal Relationships with Multiple Granularities in Time Sequences.* Data Engineering Bulletin, vol. 21, pp. 32–38, 1998. 126

[Bettini 98b] C. Bettini, X.S. Wang, S. Jajodia & J.-L. Lin. *Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences.* IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 2, pp. 222–237, 1998. 6, 126, 133

[Blanchard 08] J. Blanchard, F. Guillet & R. Gras. *Assessing the interestingness of temporal rules with Sequential Implication Intensity.* In Régis Gras, Einoshin Suzuki, Fabrice Guillet & Filippo Spagnolo, editors, Statistical Implicative Analysis, volume 127 of *Studies in Computational Intelligence*, pp. 55–71. Springer, 2008. 8

[Box 70] G. Box & G. Jenkins. Time series analysis: Forecasing and control. Holden-Day, San Francisco, 1970. 2, 94

[Bradley 98] P. S. Bradley, U. Fayyad & C. Reina. *Scaling Clustering Algorithms to Large Databases.* In Proceedings of the 4th International Conference on Knowledge Discovery adn Data Mining (KDD), pp. 94–105. ACM Press, 1998. 1

[Brahmi 10] I. Brahmi, S. B. Yahia & P. Poncelet. *MAD-IDS: novel intrusion detection system using mobile agents and data mining approaches.* In Proceedings of the Pacific Asia conference on Intelligence and Security Informatics, pp. 73–76. Springer-Verlag, 2010. 132

[Brin 97] S. Brin, R. Motwani, J. D. Ullman & S. Tsur. *Dynamic Itemset Counting and Implication Rules for Market Basket Data.* In Proceedings ACM Special Interest Group on Management Of Data International Conference, pp. 255–264, Tucson, Arizona, USA, 1997. ACM. 6

[Chang 11] J.H. Chang. *Mining weighted sequential patterns in a sequence database with a time-interval weight.* Knowledge Based Systems, vol. 24, no. 1, pp. 1–9, February 2011. 2, 5, 6, 9

[Chi 04] Y. Chi, H. Wang, P. S. Yu & R. R. Muntz. *Moment: maintaining closed frequent itemsets over a stream sliding window.* In Fourth IEEE International Conference on Data Mining, pp. 59–66, 2004. 132

[Chuzanova 98] N. A. Chuzanova, A. J. Jones & S. Margetts. *Feature selection for genetic sequence classification.* Bioinformatics, vol. 14, pp. 139–143, 1998. 126, 131

[Corchado 08] J. M. Corchado, A. Mata, F. de Paz & D. del Pozo. *A case-based reasoning system to forecast the presence of oil slicks.* In IADIS European Conference Data Mining, pp. 3–10, 2008. 49

[Costa 09] R. Costa, N. Cachulo & P. Cortez. *An Intelligent Alarm Management System for Large-Scale Telecommunication Companies.* In Proceedings of the 14th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence, pp. 386–399, Berlin, Heidelberg, 2009. Springer-Verlag. 8

[Crowley 96] P. J. Crowley. *The B-spline Wavelet Recurrence Relation and B-spline Wavelet Interpolation.* Honors Projects, no. paper 9, 1996. 59

[Das 98] G. Das, K.I. Lin, H. Mannila, G. Renganathan & P. Smyth. *Rule Discovery From Time Series.* In Knowledge Discovery and Data Mining, pp. 16–22. AAAI Press, 1998. 8

[Daubechies 92] I. Daubechies. Ten Lectures on Wavelets. CBMS-NSF Regional Conference Series in Applied Mathematics. Society of Industrial and Applied Mathematics, Philadelphia, PA, USA, May 1992. 52, 58

[Dean 87] T. L. Dean & D. McDermott. *Temporal data base management.* Artificial Intelligence, vol. 32, no. 1, pp. 1–55, 1987. 2

## BIBLIOGRAPHY

[Dematos 96] G. Dematos, M.S. Boyd, B. Kermanshahi, N. Kohzadi & I. Kaastra. *Feedforward versus recurrent neural networks for forecasting monthly japanese yen exchange rates.* Asia-Pacific Financial Markets, vol. 3, no. 1, pp. 59–75, 1996. 76

[Deshpande 02] M. Deshpande & G. Karypis. *Evaluation of Techniques for Classifying Biological Sequences.* In Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp. 417–431, 2002. 126, 132, 133

[Deshpande 04] M. Deshpande & G. Karypis. *Selective Markov models for predicting Web page accesses.* ACM Transactions on Internet Technology, vol. 4, no. 2, pp. 163–184, May 2004. 127, 133

[Devitt 05] A. Devitt, J. Duffin & R. Moloney. *Topographical proximity for mining network alarm data.* In in ACM SIGCOMM workshop on Mining network data, pp. 22–26, 2005. 1, 22

[Duskin 09] O. Duskin & D. G. Feitelson. *Distinguishing humans from robots in web search logs: preliminary results using query rates and intervals.* In Proceedings workshop on Web Search Click Data, pp. 15–19. ACM Press, 2009. 127

[Farge 92] M. Farge. *Wavelet Transforms and their applications to turbulence.* Annual Review of Fluid Mechanics, vol. 24, pp. 395–457, 1992. 52, 57

[Fayyad 96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. American Association for Artificial Intelligence, 1996. 1

[Feng 05] H. Feng & Y. Shu. *Study on network traffic prediciton techniques.* In Wireless Communications, Networking and Mobile Computing, volume 2, pp. 1041–1044, September 2005. 50

[Feng 07] D. Feng, G. Burns & E. Hovy. *Extracting Data Records from Unstructured Biomedical Full Text.* In Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 837–846. Association for Computational Linguistics, 2007. 22

[Fiszelew 07] A. Fiszelew, P. Britos, A. Ochoa, H. Merlino, E. Fernández & R. García-Martínez. *Finding Optimal Neural Network Architecture Using Genetic Algorithms.* Advances in Computer Science and Engineering, Research in Computing Science, vol. 27, pp. 15–24, 2007. 51

[Fournier-Viger 12] P. Fournier-Viger, U. Faghihi, R. Nkambou & E. M. Nguifo. *CMRules: Mining sequential rules common to several sequences.* Knowledge-Based Systems, vol. 25, no. 1, pp. 63–76, 2012. 2, 5

[Freksa 92] C. Freksa. *Temporal Reasoning Based on Semi-Intervals.* Artificial Intelligence, vol. 54, pp. 199–227, 1992. 2

[Geng 06] L. Geng & H.J. Hamilton. *Interestingness Measures for Data Mining: A Survey.* ACM Computing Surveys , vol. 38, no. 3, September 2006. 6, 16

[Giannella 04] C. Giannella, J. Han, J. Pei, X. Yan & P.S. Yu. *Mining frequent patterns in data streams at multiple time granularities.* In Kargupta Hillol & Joshi Anupam, editors, Data Mining: Next Generation Challenges and Future Directions, volume 212, pp. 105–124. MIT/AAAI Press, 2004. 9, 130

[Goldberg 88] D. E. Goldberg & J. H. Holland. *Genetic Algorithms and Machine Learning.* Machine Learning, vol. 3, pp. 95–99, 1988. 65

[Goodman 93] T.N.T. Goodman, S.L. Lee & W.S. Tang. *Wavelets in wandering subspaces.* Transactions of the American Mathematical Society, vol. 338, no. 2, pp. 639–654, August 1993. 52

[Gorman 88] R. P. Gorman & T. J. Sejnowski. *Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets.* Neural Networks, vol. 1, pp. 75–89, 1988. 59

[Goutte 05] C. Goutte & E. Gaussier. *A probabilistic interpretation of precision, recall, and f-score, with implication for evaluation.* In Proceedings of the 27th European Conference on Information Retrieval, pp. 345–359, 2005. 22

[Graps 95] A. Graps. *An Introduction to Wavelets.* Technical report, Institute of Electrical and Electronics Engineers, 1995. 51

[Graves 06] A. Graves, S. Fernández, F. Gomez & J. Schmidhuber. *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.* In Proceedings of the 23rd international conference on Machine learning, pp. 369–376. ACM Press, 2006. 126, 133

[Grosse 04] C.U. Grosse, F. Finck, J.H. Kurz & H.W. Reinhardt. *Improvements of AT technique using wavelet algorithms, coherence functions and automatic data analysis.* Construction and building materials, vol. 18, no. 3, pp. 203–213, April 2004. 51

[Han 00a] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal & M.-C. Hsu. *FreeSpan: frequent pattern-projected sequential pattern mining.* In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 355–359. ACM Press, 2000. 131

[Han 00b] J. Han, J. Pei & Y. Yin. *Mining frequent patterns without candidate generation.* SIGMOD Record, vol. 29, no. 2, pp. 1–12, May 2000. 131

[Han 07] J. Han, H. Cheng, D. Xin & X. Yan. *Frequent pattern mining: current status and future directions.* Data Mining and Knowledge Discovery, vol. 15, pp. 55–86, 2007.

[Hann 96] T.H. Hann & E. Steurer. *Much ado about nothing? Exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data.* Neurocomputing, vol. 10, no. 4, pp. 323–339, 1996. 75

[Hofgesang 05] Peter I. Hofgesang & Wojtek Kowalczyk. *Analysing Clickstream Data: From Anomaly Detection to Visitor Profiling.* In Proceedings of the ECML/PKDD Discovery Challenge, A Collaborative Effort in Knowledge Discovery from Databases, 2005, pp. 21–30, 2005. 20

[Holland 92] J. H. Holland. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, USA, 1992. 65

[Höppner 01] F. Höppner & F. Klawonn. *Finding Informative Rules in Interval Sequences.* In Frank Hoffmann, David J. Hand, Niall M. Adams, Douglas H. Fisher & Gabriela Guimarães, editors, International Conference in Intelligent Data Analysis, volume 2189 of *Lecture Notes in Computer Sciences*, pp. 125–134, Cascais, Portugal, September 2001. Springer-Verlag Berlin Heidelberg. 8

[Höppner 02] F. Höppner. *Learning Dependencies in Multivariate Time Series.* In ECAI Workshop on Knowledge Discovery in (Spatio-) Temporal Data, 2002, pp. 25–31, 2002. 8

[Huang 02] X. Huang, A. An & N. Cercone. *Comparison of interestingness functions for learning web usage patterns.* In ACM International Conference on Information and Knowledge Management, pp. 617–620, McLean, VA, USA, November 2002. ACM. 8

[Huang 07] X. Huang. *Comparison Of Interestingness Measures For Web Usage Mining: An Empirical Study.* International Journal of Information Technology & Decision Making, vol. 6, no. 01, pp. 15–41, 2007. 8

[Ibarra-Berastegi 07] G. Ibarra-Berastegi, A. Elias, R. Arias & A. Barona. *Artificial Neural Networks vs Linear Regression in a Fluid Mechanics and Chemical Modelling Problem: Elimination of Hydrogen Sulphide in a Lab-Scale Biofilter.* In Computer Systems and Applications, pp. 584–587, May 2007. 50

[Ileană 04] I. Ileană, C. Rotar & A. Incze. *The optimization of feed forward neural networks structure using genetic algorithms* . In Proceedings of the International Conference on Theory and Application of Mathematics and Informatics, pp. 223–234, Thessaloniki, Greece, 2004. 51, 77, 82

[Jain 88] A. K. Jain & R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988. 1

[Kapoor 06] V. Kapoor, P. Poncelet, F. Trousset & M. Teisseire. *Privacy preserving sequential pattern mining in distributed databases.* In Proceedings of the 15th ACM international conference on Information and knowledge management, pp. 758–767. ACM Press, 2006. 133

[Kendall 38] M. G. Kendall. *A New Measure of Rank Correlation.* Biometrika, vol. 30, no. 1/2, pp. 81–93, 1938. 18, 22

[Klema 08] J. Klema, L. Novakova, F. Karel, O. Stepankova & F. Zelezny. *Sequential Data Mining: A Comparative Case Study in Development of Atherosclerosis Risk Factors.* IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 38, no. 1, pp. 3 –15, 2008. 1, 8

[Knauth 96] P. Knauth. *Designing better shift systems.* Applied Ergonomics, vol. 27, no. 1, pp. 39–44, 1996. 79, 81

[Kolodner 92] J. L. Kolodner. *An Introduction to Case-Based Reasoning.* Artificial Intelligence Review, vol. 6, pp. 3–34, 1992. 2, 49

[Krishna 11] B.R. Krishna, R. Devi, A.Kumar, B.V. Sudhakar & P. Venkatesh. *A New Application of Data Mining.* International Jounral of Comptuer Science and Technology, vol. 2, pp. 101–104, 2011. 62

[Labsky 05] M. Labsky, V. Las & P. Berka. *Mining Click-stream Data With Statistical and Rule-based Methods.* In Proceedings of the ECML/PKDD Discovery Challenge, A Collaborative Effort in Knowledge Discovery from Databases, 2005, pp. 31–42, 2005. 19, 20, 22, 25

[Lam 08] T. W. Lam, W. K. Sung, S. L. Tam, C. K. Wong & S. M. Yiu. *Compressed indexing and local alignment of DNA.* Bioinformatics, vol. 24, no. 6, pp. 791–797, March 2008. 126, 128

[Lane 99] T. Lane & C. E. Brodley. *Temporal sequence learning and data reduction for anomaly detection.* ACM Transactions on Information and System Security, vol. 2, no. 3, pp. 295–331, August 1999. 127

[Laur 00] P. A. Laur, F. Masseglia & P. Poncelet. *Schema Mining: Finding Structural Regularity among Semistructured Data.* In Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 498–503. Springer-Verlag, 2000. 133

[Laur 05] P.-A. Laur, J.-E. Symphor, R. Nock & P. Poncelet. *Statistical Supports for Frequent Itemsets on Data Streams.* In Petra Perner & Atsushi Imiya, editors, Proceedings of the Machine Learning and Data Mining in Pattern Recognition, 4th International Conference, Leipzig, Germany, volume 3587 of *Lecture Notes in Computer Science*, pp. 395–404. Springer, 2005. 8

[Laxman 06] S. Laxman & P.S. Sastry. *A survey of temporal data mining.* Sādhanā, vol. 31, no. 2, pp. 173–198, April 2006. 2

[Leleu 03] M. Leleu, C. Rigotti, J.-F. Boulicaut & G. Euvrard. *GO-SPADE: Mining Sequential Patterns over Datasets with Consecutive Repetitions.* In P. Perner & A. Rosenfeld, editors, Proceedings of the Machine Learning and Data Mining in Pattern Recognition, Third International Conference, Leipzig, Germany, volume 2734 of *Lecture Notes in Computer Science*, pp. 293–306. Springer, 2003. 5

[Lenca 08] P. Lenca, P. Meyer, B. Vaillant & S. Lallich. *On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid.* European Journal of Operational Research, vol. 184, no. 2, pp. 610–626, 2008. 6, 16

[Leslie 04] C. Leslie, R. Kuang & K. Bennett. *Fast string kernels using inexact matching for protein sequences.* Journal of Machine Learning Research, vol. 5, pp. 1435–1455, 2004. 126, 133

[Li 04] H.-F. Li, S.-Y. Lee & M.-K. Shan. *An Efficient Algorithm for Mining Frequent Itemsets over the entire History of Data Streams.* In Proceedings of First International Workshop on Knowledge Discovery in Data Streams, 2004. 134

[Li 08a] D. Li, A. Laurent & P. Poncelet. *Recognizing unexpected recurrence behaviors with fuzzy measures in sequence databases.* In Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology, pp. 37–43. ACM Press, 2008. 133

[Li 08b] H.-F. Li, M.-K. Shan & S.-Y. Lee. *DSM-FI: an efficient algorithm for mining frequent itemsets in data streams.* Knowledge and Information Systems, vol. 17, no. 1, pp. 79–97, October 2008. 130

[Lin 98] M.-Y. Lin & S.-Y. Lee. *Incremental update on sequential patterns in large databases.* In Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence, pp. 24–31, 1998. 130

[Lin 02] W. Lin, M. A. Orgun & G. J. Williams. *An Overview of Temporal Data Mining.* In Proceedings of the 1st Australian data mining workshop, 2002. 1, 2

[Liu 07] H. Liu & V. Keselj. *Combined mining of Web server logs and web contents for classifying user navigation patterns and predicting users' future requests.* Data and Knowledge Engineering, vol. 61, pp. 304–330, 2007. 1, 19, 20, 22, 26

[Liu 08] X. Liu, P. Zhang & D. Zeng. *Sequence Matching for Suspicious Activity Detection in Anti-Money Laundering.* In Proceedings of the IEEE ISI PAISI, PACCF, and SOCO international workshops on Intelligence and Security Informatics, pp. 50–61. Springer-Verlag, 2008. 1, 127

[Lo 05] S. Lo. *Binary Prediction Based on Weighted Sequential Mining Method.* In Andrzej Skowron, Rakesh Agrawal, Michael Luck, Takahira Yamaguchi, Pierre Morizet-Mahoudeaux, Jiming Liu & Ning Zhong, editors, IEEE / WIC / ACM International Conference on Web Intelligence, Compiegne, France, pp. 755–761. IEEE Computer Society, 2005. 6

[Mabroukeh 10] N. R. Mabroukeh & C. I. Ezeife. *A taxonomy of sequential pattern mining algorithms.* ACM Computing Surveys, vol. 43, no. 1, p. 3, 2010. 5

[Mallat 89] S.G. Mallat. *Multiresolution Approximations and Wavelet Orthonormal Bases of $L^2(\mathbb{R})$.* Transactions of the American Mathematical Society, vol. 315, no. 1, pp. 69–87, September 1989. 52

[Man 96] K. F. Man, K. S. Tang & S. Kwong. *Gentic Algorithms: Concepts and Applications.* IEE Transactions on Industrial Electronics, vol. 43, no. 5, pp. 519–534, 1996. 65

[Mandal 06] B. N. Mandal. *Forecasting sugar-cane production in India with ARIMA model.* Technical report, Indian Agricultural Statistics Research Institute, New Delhi, 2006. 2, 49

[Mannila 97] H. Mannila, H. Toivonen & A.I. Verkamo. *Discovery of Frequent Episodes in Event Sequences.* Data Mining and Knowledge Discovery, vol. 1, no. 3, pp. 259–289, 1997. 6, 7

[Marchiori 08] E. Marchiori & K. Jong. *Neural Networks*, 2008. 60

[Masseglia 98] F. Masseglia, C. Fabienne & P. Poncelet. *The PSP Approach for Mining Sequential Patterns.* In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery, pp. 176–184. Springer-Verlag, 1998. 133

[Masseglia 02] F. Masseglia, M. Teisseire & P. Poncelet. *Real Time Web Usage Mining with a Distributed Navigation Analysis.* In Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, pp. 169–. IEEE Computer Society Press, 2002. 131

[Masseglia 03] F. Masseglia, P. Pascal & M. Teisseire. *Incremental mining of sequential patterns in large databases.* Data and Knowledge Engineering, vol. 46, no. 1, pp. 97–121, July 2003. 131

[Masseglia 09] F. Masseglia, P. Poncelet & M. Teisseire. *Efficient mining of sequential patterns with time constraints: Reducing the combinations.* Expert Systems with Applications, vol. 36, no. 2, pp. 2677–2690, 2009. 5, 131

## BIBLIOGRAPHY

[MathWorks 12a] MathWorks. *Matlab R2012a Documentation → Global Optimization Toolbox*. Technical report, 2012. 68, 88

[MathWorks 12b] MathWorks. *Matlab R2012a Documentation → Neural Networks Toolbox*. Technical report, 2012. 60, 61, 62, 63, 86, 120

[MathWorks 12c] MathWorks. *Matlab R2012a Documentation → Wavelet Toolbox*. Technical report, 2012. 55, 56, 120

[Mehta 11] N. Mehta & S. Dang. *Temporal Sequential Pattern in Data Mining Tasks*. International Jounral of Comptuer Science and Engineering, vol. 3, no. 2, pp. 2674–2678, 2011. 2

[Minh 06] Q. T. Minh, S. Oyanagi & K. Yamazaki. *Mining the k-most interesting frequent patterns sequentially*. In Proceedings of the 7th international conference on Intelligent Data Engineering and Automated Learning, pp. 620–628. Springer-Verlag, 2006. 127, 132, 133, 134

[Mitra 06] S. Mitra & A. Mitra. *Modeling exchange rates using wavelet decomposed genetic neural networks*. Statistical Methodology, vol. 3, pp. 103–124, 2006. 50, 54

[Mohammed 04] A. Mohammed & A. Omran. *The ARIMA Versus Artificial Neural Networks Modeling*. Central Laboratory for Agricultural Expert System, 2004. 50

[Moungnoul 05] P. Moungnoul, N. Laipat, T.T. Hung & T. Paungma. *GSM Traffic Forecast by Combining Forecasting Technique*. In Information, Communications and Signal Processing, pp. 429–433, 2005. 49

[Musliu 12] N. Musliu. *Genetic Algorithms - A Tutorial*, 2012. 65

[Naito 05] D. Naito, K. Yamamoto, K. Yada, N. Matsumura, K. Ohno & H. Tamura. *Does WEB Log Data Reveal Consumer Behavior?* In Proceedings of the ECML/PKDD Discovery Challenge, A Collaborative Effort in Knowledge Discovery from Databases, 2005, pp. 43–54, 2005. 20

[Nguyen 08] H.T. Nguyen & I.T. Nabney. *Combining the Wavelet Transform and Forecasting Models to Predict Gas Forward Prices*. In International Conference on Machine Learning and Applications, pp. 311–317, Washington, DC, USA, 2008. IEEE COmputer Society Press. 50

[Ohsaki 07] M. Ohsaki, H. Abe, S. Tsumoto, H. Yokoi & T. Yamaguchi. *Evaluation of rule interestingness measures in medical knowledge discovery in databases*. Artificial Intelligence in Medicine, vol. 41, no. 3, pp. 177–196, 2007. 1, 6

[Ojeda 95] R.G. Ojeda, F.M. de Azevedo & J.M. Barreto. *Genetic algorithms in the optima choice of neural networks for signal processing*. In 38th Midwest Symposium on Circuits Systems, Rio de Janeiro, 1995. 51, 59

[Ordonez 06] C. Ordonez. *Association Rule Discovery With the Train and Test Approach for Heart Disease Prediction*. IEEE Transactions on Information Technology in Biomedicine, vol. 10, no. 2, pp. 334–343, April 2006. 1

[Panda 07] C. Panda & V. Narasimhan. *Forecasting exchange rate better with artificial neural network*. Journal of Policy Modeling, vol. 29, pp. 227–236, 2007. 1

[Papagiannaki 05] K. Papagiannaki, N. Taft, Z.-L. Zhang & C. Diot. *Long-term forecasting of Internet backbone traffic*. Neural Networks, IEEE Transactions on, vol. 16, no. 5, pp. 1110–1124, September 2005. 50

[Parthasarathy 99] S. Parthasarathy, M. J. Zaki, M. Ogihara & S. Dwarkadas. *Incremental and interactive sequence mining*. In Proceedings of the eighth international conference on Information and knowledge management, pp. 251–258. ACM Press, 1999. 132

[Paulinas 07] M. Paulinas & A. Usinskas. *A survey of genetic algorithms applications for image enchancement and segmentation*. Information Technology and Control, vol. 36, no. 3, pp. 278–284, 2007. 65

[Pei 04] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal & M.-C. Hsu. *Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach.* IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 11, pp. 1424–1440, 2004. 2, 5, 126, 132

[Piatetsky-Shapiro 91] G. Piatetsky-Shapiro. *Discovery, Analysis, and Presentation of Strong Rules.* In Knowledge Discovery in Databases, pp. 229–248. AAAI/MIT Press, 1991. 12, 16, 46

[Preisler 06] H. K. Preisler & A. L. Westerling. *Statistical Model for Forecasting Monthly Large Wildfire Events in Western United States.* Journal of Applied Meteorology and Climatology, vol. 46, pp. 1020–1030, 2006. 49

[Quang 06] T.M. Quang, S. Oyanagi & K. Yamazaki. *ExMiner: An Efficient Algorithm for Mining Top-K Frequent Patterns.* In Advanced Data Mining and Applications, volume 4093 of *Lecture Notes in Computer Science*, pp. 436–447. Springer-Verlag, 2006. 5, 127, 130, 134

[Rai 11] P. Rai & K. Rai. *Comparison of Stock Prediction Using Different Neural Network Types.* International Journal of Advanced Engineering and Application, pp. 157–160, 2011. 1

[Raïssi 07] C. Raïssi, P. Poncelet & M. Teisseire. *Towards a new approach for mining frequent itemsets on data stream.* Journal of Intelligent Information Systems, vol. 28, pp. 23–36, 2007. 127, 130

[Raïssi 08] C. Raïssi, T. Calders & P. Poncelet. *Mining Conjuctive Sequential Patterns.* Data Mining and Knowledge Discovery, vol. 17, no. 1, pp. 77–93, August 2008. 130

[Rajeevan 07] M. Rajeevan, D. S. Pai, R. A. Kumar & B. Lal. *New statistical models for long-range forecasting of southwest monsoon rainfall over India.* Climate Dynamics, vol. 28, no. 7-8, pp. 813–828, 2007. 50

[Refenes 93] A.N. Refenes. *Constructive learning and its application to currency exchange rate forecasting.* In R.R. Trippi & E. Turban, editors, Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real-World Performance, pp. 465–493. Eds IL: Probus, 1993. 2, 50

[Riordan 02] D. Riordan & B. K. Hansen. *A fuzzy case-based system for weather prediction.* Engineering Intelligent Systems, vol. 3, pp. 139–146, 2002. 2, 49

[Ripley 93] B.D. Ripley. *Statistical aspects of neural networks.* In Network and Chaos - Statistical and Probabilistic Aspects, pp. 40–123. Chapman and Hall/CRC, July 1993. 50

[Roddick 02] J. F. Roddick & M. Spiliopoulou. *A Survey of Temporal Knowledge Discovery Paradigms and Methods.* IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 4, pp. 750–767, 2002. 2

[Rogers 99] A. Rogers & A. Prügel-Bennett. *Genetic drift in genetic algorithm selection schemes.* IEEE Transactions on Evoluationary Computation, vol. 3, no. 3, pp. 298–303, 1999. 68

[Rong 08] Z. Rong, C. Rongqiu, X. Xia & W. Guoping. *A case-based reasoning system for individual demand forecasting.* In Wireless Communications, Networking and Mobile Computing, pp. 1–6, 2008. 48, 49

[Sakurai 07] S. Sakurai, Y. Kitahara & R. Orihara. *Sequential pattern mining based on a new criteria and attribute constraints.* In Systems, Man, and Cybernetics, pp. 516–521. IEEE, 2007. 9

[Sbirrazzuoli 97] N. Sbirrazzuoli & D. Brunel. *Computational Neural Networks for Mapping Calorimetric Data: Application of Feed-Forward Neural Networks to Kinetic Parameters Determination and Signals Filtering.* Neural Computing and Applications, vol. 5, pp. 20–32, 1997. 59

[Shensa 92] M.J. Shensa. *The Discrete Wavelet Transform: Wedding the A Trous and Mallat Algorithms.* IEEE Transactions on Signal Processing, vol. 40, no. 10, pp. 2464–2482, October 1992. 56

[Shukla 03] P. D. Shukla. *Complex Wavelet Transforms and Their Applications.* PhD thesis, University of Strathclyde, Signal Processing Division, Department of Electronic and Electrical Engineering, Glasgow, Scotland, United Kingdom, 2003. 57

[Singh 09a] G. Singh, F. Masseglia, C. Fiot, A. Marascu & P. Poncelet. *Data Mining for Intrusion Detection: From Outliers to True Intrusions.* In Proccedings of Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, volume 5476, pp. 891–898, 2009. 1, 128

[Singh 09b] Y. Singh & A.S. Chauhan. *Neeural Networks in Data Mining.* Journal of Theoretical and Applied Information Technology, vol. 5, no. 1, pp. 37–42, 2009. 62

[sit 12a] *Artificial Neural Networks @http://www.learnartificialneuralnetworks.com/*, accessed in June, 2012. 60

[sit 12b] *Genetic Algorithm Viewer @http://www.rennard.org/alife/english/gavgb.html*, accessed in June, 2012. 68

[sit 12c] *Introduction to Genetic Algorithms → Selection @http://www.obitko.com/tutorials/genetic-algorithms/selection.php*, accessed in June, 2012. 67, 68, 121

[Smyth 91] P. Smyth & R. M. Goodman. *Rule Induction Using Information Theory.* In Knowledge Discovery in Databases, pp. 159–176. 1991. 8

[Son 04] J.S. Son, D.M. Lee, I.S. Kim & S.K. Choi. *A study on genetic algorithm to select architecture of a optimal neural network in the hot rolling process.* Journal of Materials Processing Technology, vol. 153-54, pp. 643–648, 2004. 51

[Spiliopoulou 99] M. Spiliopoulou. *Managing Interesting Rules in Sequence Mining.* In Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery, pp. 554–560, London, UK, 1999. Springer-Verlag. 8

[Srikant 96] R. Srikant & R. Agrawal. *Mining Sequential Patterns: Generalizations and Performance Improvements.* In Peter M. G. Apers, Mokrane Bouzeghoub & Georges Gardarin, editors, Proceedings of the Advances in Database Technology, 5th International Conference on Extending Database Technology, Avignon, France, volume 1057 of *Lecture Notes in Computer Science*, pp. 3–17. Springer, 1996. 2, 5, 17, 33, 41, 131

[Stolojescu 09] C. Stolojescu, I. Firoiu & A. Isar. *Forecasting of WiMAX BS Traffic: Observations and Initial Models.* Technical report, Alcatel-Lucent Timisoara, 2009. 73

[Strang 96] G. Strang & T. Nguyen. Wavelets and Filter Banks. Wellesley - Cambridge Press, 2nd edition, 1996. 57

[Surana 11] Akshat Surana, R. Uday Kiran & P. Krishna Reddy. *An Efficient Approach to Mine Periodic-Frequent Patterns in Transactional Databases.* In Longbing Cao, Joshua Zhexue Huang, James Bailey, Yun Sing Koh & Jun Luo, editors, PAKDD Workshops, volume 7104 of *Lecture Notes in Computer Science*, pp. 254–266. Springer, 2011. 6

[Tan 93] C.N.W. Tan. *Incorporating Artificial Neural Networks into a Rule-Based Financial Trading System.* In Proceedings of the First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, pp. 292–293. IEEE COmputer Society Press, 1993. 50

[Tan 04] P.-N. Tan, V. Kumar & J. Srivastava. *Selecting the right objective measure for association analysis.* Information Systems, vol. 29, no. 4, pp. 293–313, 2004. 16

[Tanbeer 08] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong & Y.-K. Lee. *CP-tree: a tree structure for singla-pass frequent pattern mining.* In Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining, pp. 1022–1027. Springer-Verlag, 2008. 130

[Tanbeer 09] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong & Y. K Lee. *Discovering Periodic-Frequent Patterns in Transactional Databases.* In Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp. 242–253. Springer-Verlag, 2009. 6, 127, 132

[Teng 03] W.-G. Teng, M.-S. Chen & P. S. Yu. *A regression-based temporal pattern mining scheme for data streams.* In Proceedings of the 29th international conference on Very large data bases, volume 29, pp. 193–104, 2003. 131

[Teodoro 06] M. F. Teodoro & L. M. Botelho. *A case-based reasoning approach for predicting bank lending decisions.* Technical report, ISCTE Business School, 2006. 49

[Thalatam 10] M.N.V. Thalatam, P.V. Rao, K.V.S.R.P. Varma, N.V.R. Murty & A. Apparao. *Prediction of Protein Secondary Structure using Artificial Neural Network.* International Jounral of Comptuer Science and Engineering, vol. 2, no. 5, pp. 1615–1621, 2010. 63

[Torrence 98] C. Torrence & G. P. Compo. *A Practical Guide to Wavelet Analysis.* Bulletin of the American Meteorological Society, vol. 79, no. 1, pp. 61–78, 1998. 57

[Tucker 99] P. Tucker, L. Smith, I. Macdonald & S. Folkard. *Distribution of rest days in 12 hour shift systems: impacts on health, wellbeing, and on shift alertness.* Occupational and Environmental Medicine, vol. 56, pp. 206–214, 1999. 81, 84

[Tuğ 06] E. Tuğ, M. Şakiroğlu & A. Arslan. *Automatic discovery of the sequential accesses from web log data files via a genetic algorithm.* Knowledge Based Systems, vol. 19, no. 3, pp. 180–186, 2006. 1

[Tzvetkov 05] P. Tzvetkov, X. Yan & J. Han. *TSP: Mining top-k closed sequential patterns.* Knowledge and Information Systems, vol. 7, no. 4, pp. 438–457, May 2005. 134

[Venkatesan 09] D. Venkatesan, K. Kannan & R. Saravanan. *A genetic algorithm-based artificial neural network model for the optimization of machining processes.* Neural Computing and Applications, vol. 18, no. 2, pp. 135–140, February 2009. 51

[Wang 96] K. Wang & J. Tan. *Incremental Discovery of Sequential Patterns.* In Proceedings of ACM SIGMOD Data Mining Workshop: Research Issues on Data Mining and Knowledge Discovery, pp. 95–102. ACM Press, 1996. 133

[Wang 02] X. Wang & X. Shan. *A wavelet-based method to predict Internet traffic.* In Communications, Circuits and Systems and West Sino Expositions, volume 1, pp. 690–694, 2002. 50

[Webb 03] G. K. Webb. *A rule based forecast of computer hard drive costs.* Issues in Information Systems, vol. 4, pp. 337–343, 2003. 2, 49

[Weiss 01] G. Weiss. *Predicting Telecommunication Equipment Failures from Sequences of Network Alarms.* In Handbook of Data Mining and Knowledge Discovery, pp. 891–896. Oxford University Press, 2001. 1, 8

[White 89] H. White. *Learning in Artificial Neural Networks: A Statistical Perspective.* Neural Computation, vol. 1, no. 4, pp. 425–464, 1989. 50

[Wong 06] R. C.-W. Wong & A. W.-C. Fu. *Mining top-K frequent itemsets from data streams.* Data Mining and Knowledge Discovery, vol. 13, no. 2, pp. 193–217, 2006. 134

[Yang 02] J. Yang, W. Wang & P. S. Yu. *InfoMiner+: Mining Partial Periodic Patterns with Gap Penalties.* In IEEE International Conference on Data Mining, pp. 725–728, Maebashi City, Japan, 2002. IEEE Computer Society. 2, 8

[Yang 06] Q. Yang & X. Wu. *10 Challenging Problems in Data Mining Research.* International Journal of Information Technology and Decision Making, vol. 5, no. 4, pp. 597–604, 2006. 5

[Yu 07] J.-C. Yu & Y.-L. Tseng. *Evolutionary Engineering Optimization Using Recursive Regional Neural Network and Genetic Algorithm.* In Proceedings of the Second International Conference on Innovative Computing, Information and Control, p. 325, Washington, DC, USA, 2007. IEEE COmputer Society Press. 51

[Yun 06] U. Yun & J. J. Leggett. *WSpan: Weighted Sequential pattern mining in large sequence databases.* In 3rd International IEEE Conference Intelligent Systems, pp. 512 – 517, September 2006. 5

[Yun 07] U. Yun. *WIS: Weighted Interesting Sequential Pattern Mining with a Similar Level of Support and/or Weight.* ETRI Journal, vol. 29, no. 3, pp. 336–352, 2007. 9

[Yun 08] U. Yun. *A new framework for detecting weighted sequential patterns in large sequence databases* . Knowledge Based Systems , vol. 21, no. 2, pp. 110–122, 2008. 6, 9

[Zaki 01] M.J. Zaki. *SPADE: An Efficient Algorithm for Mining Frequent Sequences.* Machine Learning, vol. 42, no. 1/2, pp. 31–60, 2001. 5, 133

[Zhang 93] B.-T. Zhang & H. Mühlenbein. *Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor.* Complex Systems, vol. 7, no. 3, pp. 199–220, 1993. 51

[Zhang 98] G. Zhang, E.B. Patuwo & Y.M. Hu. *Forecasting with artificial neural networks: The state of the art.* International Journal of Forecasting, vol. 14, no. 1, pp. 35–62, March 1998. 50

[Zhang 01] B. Zhang, R. Coggins, M. A. Jabri, D. R. Dersch & B. Flower. *Multiresolution forecasting for futures trading using wavelet decompositions.* IEEE Transactions on Neural Networks, vol. 12, no. 4, pp. 765–775, 2001. 50

[Zhao 03] Q. Zhao & S. S. Bhowmick. *Sequential Pattern Mining: A Survey.* Technical report, CAIS, Nanyang Technology University, No 2003118, 2003. 5, 125

[Zhao 08] Y. Zhao, H. Zhang, L. Cao, C. Zhang & H. Bohlscheid. *Efficient Mining of Event-Oriented Negative Sequential Rules.* In Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, pp. 336–342, Washington, DC, USA, 2008. IEEE Computer Society. 6

# List of Figures

2.1 Roles of interestingness measures in data mining process. . . . . . . . . . . 7

2.2 Weight distance function according to $\omega_t$ and slope $s$. . . . . . . . . . . . . 13

2.3 An example of the $CP_\omega$ calculation. . . . . . . . . . . . . . . . . . . . . . . . . 14

2.4 An example of the $CP_S$ calculation. . . . . . . . . . . . . . . . . . . . . . . . . 14

2.5 The function $f(t)$ with $s = 2$ and $\omega_t = 6$. . . . . . . . . . . . . . . . . . . . 17

2.6 $CP_\omega$ and $CP_S$ calculation for the rule $d \rightarrow a$. . . . . . . . . . . . . . . . . 18

2.7 Data transformation of the *ClickStream* database. Each row of the initial log file contains: shop ID *s ID*; unixtime *UT t*; IP address *IP*; session *sess ID*; visited page *VP*; referrer *ref*. From these, we group the rows into sessions, thus, a session will contain several visited pages and the corresponding time. The notation with $x$ presents different shop IDs or IP addresses (that of course might be the same, but these are not taken into consideration by us). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

2.8 The functions to calculate the proposed IM for the ClickStream dataset: top (from left to right): $slope = 1,05$, $\omega_t = 360, 420$; bottom (from left to right): $slope = 1,25$, $\omega_t = 360, 420$. . . . . . . . . . . . . . . . . . . . . . . 22

2.9 The values for *Confidence*, *Lift*, and $CP$. $CP$ parameters were: $\omega_t = 360$ $slope = 1,25$. From 170 rules only 158 are shown because of the very high values of the $CP$ of the remaining rules resulting in a graphic concentration at the origins. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

2.10 The histograms of the ClickStream dataset with $s = 1.5\%$, $\omega_t = 360, 420$, and $slope = 1,05; 1,25$. Notice the increasing of the rules' values in the proposed IM while increasing the window's width and slope. . . . . . . . . 24

2.11 *Accuracy*, *Precision*, *Recall*, and *F1-Score* for the subset of rules containing the highest $CP$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

2.12 A comparison between *simple* and *complex rules*. Note, that the $OX$ axis represents the number of *simple rules*. The information from the axis does not correspond to the number of *complex rules*, but it corresponds to the initial number of *simple rules* from which the *complex* ones were derived. $CP$ parameters were: $\omega_t = 360$, $slope = 1.25$. . . . . . . . . . . . . . . . . . 28

2.13 An example of the time-differences problem. . . . . . . . . . . . . . . . . . 32

2.14 An explanation of the *coefficient function* to meet our requirements. We can see, that if $\Delta_{t_{average}}$ is higher, it takes the upper value of the ceiling, otherwise, it takes the bottom value, but it never passes to the previous coefficient values of its subpatterns. . . . . . . . . . . . . . . . . . . . . . . . 33

2.15 $MCP_{sc}$ as a function of $coeff$ and $supp$. Both $coeff$ and $supp$ varied from 0.1 to 0.9. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

2.16 Lost patterns for a $supp_{thr} = 0.4$. Note, that in case of $MCP_{sc\ thr} = 0.2$ the patterns with measure's values $> thr$ at the upper part of the graph will be lost. . . . . 36
2.17 The function used in toy example. The *slope* $s = 2$ and the time window $\omega_t = 2$. . . . . 36
2.18 A comparison of the processing steps between $MCP_{sc}$ and the decomposition of the $MCP_{sc}$. . . . . 40
2.19 The number of common patterns between *Case I* and *Case II* according to the top-$i$ ranked patterns arranged in a decreasing order of their IM. . . . 41
2.20 The number of the common patterns in percentage % between *Case I* and *Case II* according to the first taken patterns arranged in a decreasing order of their IM. . . . . 41
2.21 The number of common patterns between $MCP_{sc}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM. . . . 42
2.22 The number of common patterns in percentage % between $MCP_{sc}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM. . . . . 43
2.23 The number of common patterns between $MCP_{sc}$ and $MCP_{s\ func}$ according to the first taken patterns arranged in a decreasing order of their IM. . . . . 44
2.24 The number of common patterns in percentage % between $MCP_{sc}$ and $MCP_{s\ func}$ according to the first taken patterns arranged in a decreasing order of their IM. . . . . 44
2.25 The number of common patterns between $MCP_{s\ func}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM. . . . . 45
2.26 The number of common patterns in percentage % between $MCP_{s\ func}$ and *Support* according to the first taken patterns arranged in a decreasing order of their IM. . . . . 46

3.1 Daubechies 4-*tap* (*db*2) wavelet. The scaling function is $\phi$ (father wavelet), while the wavelet function is $\psi$ (mother wavelet). . . . . 53
3.2 Low-pass and high-pass filtering of a signal. $S$ denotes the original signal, $A$ denotes the approximation and $D$, the detail; $\downarrow$ represents the sub-sampling of the signal. . . . . 55
3.3 A two-level wavelet decomposition process; $\downarrow$ represents the down-sampling of the signal. . . . . 55
3.4 Up-sampling of a signal [MathWorks 12c]. . . . . 56
3.5 A two-level wavelet reconstruction process; $\uparrow$ represents the up-sampling of the signal. . . . . 56
3.6 Decomposition and Reconstruction filters for the *db*4 wavelet 8-*tap*). . . . . 56
3.7 Single Neuron Model [MathWorks 12b]. . . . . 60
3.8 Single Layer Neural Network [MathWorks 12b]. . . . . 61
3.9 Three Layer Neural Network [MathWorks 12b]. . . . . 62
3.10 Simple Recurrent Neural Network. . . . . 62
3.11 Hard-Limit transfer function. . . . . 64

3.12 Linear transfer function. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
3.13 Log-Sigmoid transfer function. . . . . . . . . . . . . . . . . . . . . . . . . . 64
3.14 Block diagram of a standard Genetic Algorithm. . . . . . . . . . . . . . . . 66
3.15 Roulette wheel selection [sit 12c]. . . . . . . . . . . . . . . . . . . . . . . . 67
3.16 Situation before ranking (graph of fitnesses) [sit 12c]. . . . . . . . . . . . . 68
3.17 Situation after ranking (graph of order numbers) [sit 12c]. . . . . . . . . . 68
3.18 Block diagram of the forecasting process. . . . . . . . . . . . . . . . . . . . 71
3.19 Simplified block diagram of the forecasting process. . . . . . . . . . . . . . 72
3.20 SWT preprocessing for ANN training, used in $1^{st}$ method for day prediction. 74
3.21 SWT preprocessing for ANN testing, used in $1^{st}$ method for day prediction. 75
3.22 Approximation and Detail time-series for training, used in $1^{st}$ method for future ANN inputs (to the left), and for ANN target (to the right). . . . . 76
3.23 SWT preprocessing for ANN training, used in $2^{nd}$ method for day prediction. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
3.24 SWT preprocessing for ANN testing, used in $2^{nd}$ method for day prediction. 78
3.25 SWT preprocessing for ANN training, used in One Week Forecasting. . . . 78
3.26 SWT preprocessing for ANN testing, used in One Week Forecasting. . . . 79
3.27 24 Hours Shift for neuron decomposition. . . . . . . . . . . . . . . . . . . . 80
3.28 Continuous shift system in the steel industry (to the left), and in the chemical industry (to the right). . . . . . . . . . . . . . . . . . . . . . . . . . 80
3.29 8 Hours Shift for neuron decomposition. . . . . . . . . . . . . . . . . . . . 81
3.30 Performance's dependence of network complexity. . . . . . . . . . . . . . . 82
3.31 Performance's dependence of number of iterations used for training. . . . . 83
3.32 Obtained SMAPE values as a function of neurons number in the hidden layer for One Day Prediction ANN. . . . . . . . . . . . . . . . . . . . . . . 83
3.33 Forecasting capability when using 3 neurons (above) and 15 neurons (below) for the size of the hidden layer in One Day Prediction. . . . . . . . . . . 84
3.34 2 Days Shift for input vector-neuron decomposition, One Week Prediction. 85
3.35 Obtained SMAPE values as a function of neurons number in the hidden layer for One Week Prediction ANN. . . . . . . . . . . . . . . . . . . . . . . 86
3.36 Applying Genetic Algorithm on Neural Networks in case of $W67$ database. The signals from the last week will be used for testing. . . . . . . . . . . 90
3.37 GA Training ((a) and (b)) vs usual ANN Training ((c) and (d)) (for the approximation signals chosen from a random Base Station). The blue signals are the original ones, the red and green signals represent the output of the network after training. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
3.38 Prediction of the signal using optimized Artificial Neural Networks for the $W67$ database. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
3.39 Applying Genetic Algorithm on Neural Networks for the $W504$ database (24 hours shifting). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
3.40 Prediction of the signal using optimized Artificial Neural Networks for the $W504$ database (24 hours shifting). . . . . . . . . . . . . . . . . . . . . . . 94

## LIST OF FIGURES

1 Relations between existing method for Mining Sequential Patterns. The arrow indicates the *parrent - child* relation between algorithms (which method is an extension/improvement/new idea from a previous one). The dashed line shows the methods from the same "family". . . . . . . . . . . . . . . . . 129
2 First family . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
3 Second family . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
4 Families 3, 4, and 5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
5 Family 6 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
6 Times Comparison of the algorithms. This is done relative to the *Apriori* method. The placements of the algorithms do not represent the real time ratios, they represent just that one is slower or faster than another. The times between different classes are not compared (for example, *FreeSpan* seems to be at the same level as *ExMiner* it does not mean that they have the same processing time) . . . . . . . . . . . . . . . . . . . . . . . . . . . 137

# List of Tables

2.1 Toy example DB. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Sequential rules of the toy database with their *CP* and *Lift* values. . . . . . 19

2.3 *Kendall's* $\tau$ *Correlation* coefficients between *CP* and *Confidence* with *Lift* computed on the *ClickStream* database. . . . . . . . . . . . . . . . . . . . . . 23

2.4 *Accuracy* (A), *Precision* (P), *Recall* (R), and *F1-Score*(F1) for $\omega_t = 360$. The *CP*'s *slope* $s = 1.25$. The number of *simple rules* tested was 31 for each measure. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.5 Number of rules *nr*, *Accuracy* (A), *Precision* (P), *Recall* (R), and *F1-Score* (F1) for $\omega_t = 360$ according to *slope* *s* and *threshold* $\theta$ set for the *CP*. . . . 27

2.6 Number of rules *nr*, *Accuracy* (A), *Precision* (P), *Recall* (R), and *F1-Score* (F1) for $\omega_t = 420$ according to *slope* *s* and *threshold* $\theta$ set for the *CP*. . . . 27

2.7 The number of obtained sequential patterns at each stage of the GSP algorithm using classic *Support* processing and $MCP_{sc}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than threshold value. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

2.8 Differences in measure's values between *Support*, $MCP_{sc}$, and $MCP_{s\ func}$. 39

2.9 The number of obtained sequential patterns at each stage of the GSP algorithm using windowed *Support* $\omega_t$ processing and $MCP_{sc}$ processing. *NAP* - Number of analysed patterns; *NKPF* - Number of kept patterns for the future processing; *NPT* - Number of patterns with IM value greater than the threshold value. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

2.10 The number of obtained sequential patterns at each stage of the GSP algorithm using classic *Support* processing and $MCP_{sc}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than the threshold value. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

2.11 The number of obtained sequential patterns at each stage of the GSP algorithm using $MCP_{sc}$ and $MCP_{s\ func}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than the threshold value. . . . 43

2.12 The number of obtained sequential patterns at each stage of the GSP algorithm using classic *Support* processing and $MCP_{s\ func}$ processing. *NAP* - Number of analyzed patterns; *NKPF* - Number of kept patterns for the future processings; *NPT* - Number of patterns with IM value greater than the threshold value. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

3.1 Examples of real wavelet functions. The number represents the order of the given wavelet. In case of the *biorthogonal* and *reversed biorthogonal* wavelets, the first number is the number of vanishing moments in the synthesis wavelet, while the second one is the number of vanishing moments in the analysis wavelet. . . . . 58
3.2 Examples of complex wavelet functions. The number represents the order of the given wavelet. . . . . 59
3.3 Medium $RSQ$ and $SMAPE$, Day Prediction. . . . . 96
3.4 Medium $RSQ$ and $SMAPE$, *Similar Days Selection* Model, using Genetic Algorithms optimization. . . . . 97
3.5 Medium $RSQ$ and $SMAPE$ by comparing the number of shifted days in one week *No Sliding* prediction model for the $W67$ database. . . . . 98
3.6 Comparison between the three models based on ANN. . . . . 98
3.7 Comparison between wavelets in week prediction, $W67$ WiMAX traffic database. . . . . 99
3.8 Comparison between the forecasting techniques for 2 weeks prediction, $W67$ WiMAX traffic database. . . . . 99
3.9 Comparison between wavelets in week prediction, $EUVol$ database. . . . . 100
3.10 Comparison between the forecasting techniques for 2 weeks prediction, $EUVol$ database. . . . . 100
3.11 Comparison between wavelets in week prediction, *EURate* database. . . . . 100
3.12 Comparison between the forecasting techniques for 2 weeks prediction, *EURate* database. . . . . 101

# Annexes

## Publications

I. Railean, S. Moga, M. Borda, *Forecasting by Neural Networks in the Wavelet Domain*, Acta Technica Napocensis Electronics - Telecomunications, 2009, 50(4): 15-27.

I. Railean, C. Stolojescu, S. Moga, P. Lenca: *WiMAX Traffic Forecasting based on Neural Networks in Wavelet Domain*, Proceedings of the 4$^{th}$ IEEE International Conference on Research Challenges in Information Science, Nice, 2010.

I. Railean, S. Moga, M. Borda, C. Stolojescu: *Neural Networks vs Genetically Optimized Neural Networks in Times Series Prediction*, International Conference on Stochastic Modelling Techniques and Data Analysis, Chania, Crete, 2010

C. Stolojescu, I. Railean, S. Moga, P. Lenca, A. Isar; *A Wavelet Based Prediction Method for Times Series*, Proceedings of the Stochastic Modeling Techniques and Data Analysis International Conference, Chania, Crete, 2010

C. Stolojescu, I. Railean, S. Moga, A. Isar: *Comparison of Wavelet Families with Application to WiMAX Traffic Forecasting*, Proceedings of the 12th International Conference on Optimization of Electrical and Electronic Equipment, Brasov, Romania, 2010, pp. 932-937

## Preface

In the annexes chapter we analyse the existing algorithms for sequential patterns extraction. We classify them according to their goal, their relations, and their differences regarding complexity and mining speed.

## A Classification of sequential pattern mining algorithms

Many previous studies contributed to finding different algorithms for mining the sequential data. Zhao and Bhowmick [Zhao 03] categorize the patterns that could be obtained from different types of time-series data into four categories:

- *Trend analysis*: finds the evolution patterns of attributes over time;
- *Similarity search*: finds sequences that differ only slightly;

- *Sequential patterns*: finds the relationships between occurrences of sequential events, in order to see if there is any specific order of the occurrences;
- *Periodical patterns*: recurring patterns in the time series databases, periodicity can be daily, weekly, monthly, seasonal or yearly.

What we are looking for is the classification of the algorithms according to their final goal. In this way we could define three important categories and tasks of the algorithms used for sequential data mining.

## A.1 First category

The methods from the first category are used in the *classification of sequences*: Leslie et.al. [Leslie 04], propose an extended *k-mer* based kernel framework for use with SVMs for classification of protein sequence data. These kernels (*restricted gappy kernels*, *substitution kernels*, and *wildcard kernels*) are based on feature spaces indexed by k-length subsequences, and use biologically-inspired models of inexact matching. An evaluation of *K-nearest neighbor*, *Markov models* (simple Markov chains of various orders, sequence classifiers derived from *Interpolated Markov Models*, *Selective Markov Models*, SVM) are presented in [Deshpande 02].

The formal feature selection method based on the *Gamma* (or near-neighbor) test is developed by Chuzanova et al. [Chuzanova 98]. The process of finding the best subset of the features is speeded up by using *genetic algorithms* and a *kd-tree* technique for the construction of the nearest-neighbor lists. Graves et al. proposes an AI approach using *Recurrent Neural Networks* for phonetic labeling on the TIMIT speech corpus [Graves 06].

## A.2 Second category

The second category finds *periodic frequent patterns in a database*. In the case when we have a very long itemset (i.e. DNA sequence), then different comppressed indexes methods are applied, such as BWT (*Burrows− Wheeler Transform*), CSA(*Compressed Suffix Array*), FM-index [Lam 08], in order to speed up the process of finding the local alignments of another (much shorter) itemset.

However, if our database consists of time sequences, then we do not need a compression for the itemset, what we need is a compression of the entire database, taking into account only the elements appearing more frequently. Bettini et al. [Bettini 98a], [Bettini 98b], propose a strategy for finding the solution of event−mining problems, which relies on the many optimization opportunities provided by the temporal constraints of the event structures. They use a *timed automata with granularities* (TAGs) (for testing whether a specific temporal pattern, appears frequently in a time sequence), and *heuristics* (aim at reducing the number of candidate eventy types and reducing the time spent by the TAGs testing whether a candidate sype does appear frequently in the sequence), which has been proven to be more performant then GSP (*Generalized Sequential Pattern*). These optimizations consist in identifying the possible inconsistencies in the given event structure before starting the process, and reducing the length of the sequence.

Pai et al. [Pei 04] explore a *pattern-growth approach*: sequence databases are recursively projected into a set of smaller databases based on the current sequential patterns, and sequential patterns are grown in each projected database by exploring only locally frequent fragments. The following methods were proposed based on this ideas:

- *FreeSpan* (Frequent pattern−projected Sequential pattern mining), which reduces the efforts of candidate subsequence generation;
- *PrefixSpan* (Prefix−projected Sequential pattern mining), which offers ordered growth and reduced projected databases;
- *Pseudo−projection* technique was developed in PrefixSpan to further improve the performance.

Raissi et al. [Raïssi 07] propose a *Frequent itemsets mining on data streams* (FIDS) algorithm. The advantage of the algorithm is that the users can issue requests for frequent itemsets over an arbtirary time interval at any time. In [Tanbeer 09] a tree−based structure is used, called *Periodic−frequent pattern tree* (PF-tree), that captures the database contents in a highly compact manner, and enables a pattern growth mining technique to generate the complete set of periodic-frequent patterns in database for user−given periodicity and support thresholds. However, a newer method based on the same principles, but with fewer modifications, are depicted by Amphawan et al. [Amphawan 10]. The main difference is that in this case the authors do not use the exact periodicity as in [Tanbeer 09], but an approximate periodicity.

A problem with the frequent patterns findings, is that the user should provide a support threshold which is very difficult to identify withoud knowledge about the dataset in advance. This is where the mining *top−k* frequent patterns are used, where the users control the number of pattern to be discovered for analyzing [Minh 06]. Minh et al. [Minh 06] propose an optimization method of ExMiner [Quang 06], called *OExMiner*, to mine the top-k frequent patterns from a large scale dataset efficiently and effectively. Another 3 methods have been proposed by them also: *Seq-Miner*, and *Seq−BOMA*, being slight modifications of the OExMiner.

## A.3 Third category

To the third category, could be attached the methods used in *prediction*. For example, predicting a user's behavior on a Web site in need to personalize and influence user's browsing experience, or detecting suspicious activity detection methods [Lane 99], [Duskin 09]. In case of the web access prediction, Deshpande and Karypis [Deshpande 04] combine different order Markov models, so that the resulting model has a low state−space complexity and, at the same time, retains the coverage and the accuracy of the All−Kth−Order Markov models. The key idea is that many of the states of the different order Markove models can be eliminated without affecting the performance of the overall scheme. They also use *Selective Markove models*, in order to reduce the state complexity and improve the prediction accuracy of the resulting model. Three schemes are presented for pruning the states of the All−Kth−Order Markove model: (1) frequency pruning, (2) confidence pruning, (3) error pruning.

An example of predicting the money laundering, is presented by Liu et al. [Liu 08]. They combine sequence matching and classification methods. There are four steps

- data acquisition and parsing, in order to determine the peer group (the following attributes are used: industry, account type, and company size);
- high-risk sequence selection;
- similarity calculation between high-risk sequences and reference sequences: the similarities are calculated between each query sequence and each reference sequence (Euclidean distance measure is used);
- sequences classification: the process consists in ordering the similarity of each candidate sequence, giving a threshold *Ts*, choosing the relatively higher ones, and flag them in the original database as suspicious.

Of course, that many of the algorithms could be used in both: periodic frequent patterns finding and prediction.

## B Relational representation of existing algorithms. Used databases

In Figure 1 you can observe the existing methods for mining sequential patterns and the relations between them. The arrow indicates the *parrent - child* relation, for example *PF-Tree* algorithm was developed by taking the main ideas from *FP-Growth* algorithm, *Seq-Miner* from *OExMiner*, and so on. The dashed line represent that the connected methods are from the same "family", meaning that they explore the same global idea.

Now, let's describe briefly each of the algorithms:

- *Apriori* [Agrawal 94]. The algorithm consists of two steps: the first one counts item occurences to determine the large 1-itemsets. A subsequent pass $k$ consists of 2 phases: (1) the large itemsets $L_{k-1}$ found in the $k-1$th pass are used to generate the candidate itemsets $C_k$ using the apriori function; (2) next, the database is scanned and the support of candidates in $C_k$ is counted.
- *AprioriAll* [Agrawal 95]. In each pass the large sequences from the previous pass are used to generate the candidate sequences and then measure their support by making a pass over the database. At the end of the pass, the support of the candidates is used to determine the large sequences. In the first pass, the output of the l-itemset phase is used to initializ the set of large 1-sequences. The candidates are stored in hash-tree to quickly find all candidates contained in a customer sequence.
- *BWT-SW*, exploits BWT index of a text T to speed up the dynamic programming for finding all local alignments [Lam 08].
- *COD* (Common Outlier Detection) [Singh 09a]. The principle of the algorithm is to perform successive clustering steps on usage patterns of different partners sites, until the number of common outliers meets the number of alarms desired by the user. The data sets for the analysis were taken from two different research organizations. The first log file the total number of objects is 30,454. The second log file contains 72,381 objects.

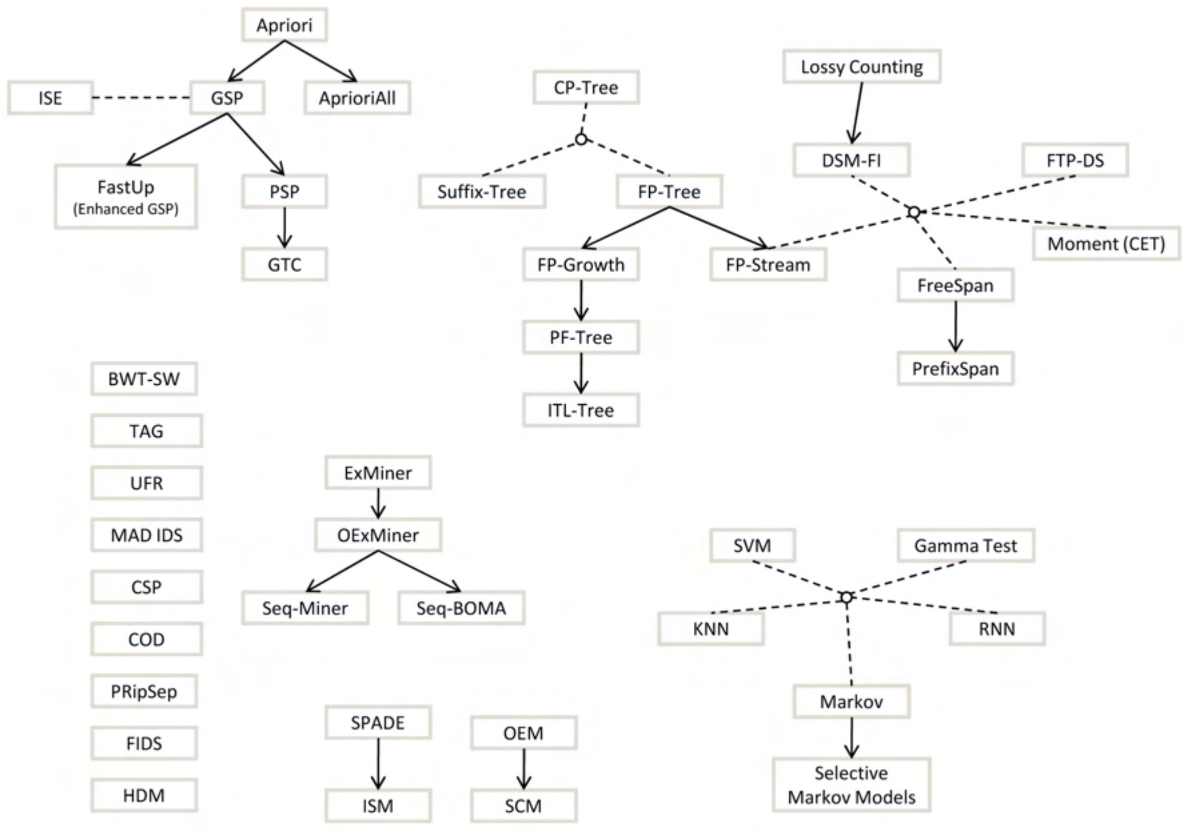


Figure 1 : Relations between existing method for Mining Sequential Patterns. The arrow indicates the *parrent - child* relation between algorithms (which method is an extension/improvement/new idea from a previous one). The dashed line shows the methods from the same "family".

- *CP-Tree* (Compact Pattern Tree) [Tanbeer 08]. It captures database information with one scan (Insertion phase), and provides the same mining performance as the FP-Growth method (Restructuring phase) by dynamic tree restructuring process. It can give also full functionalities for interactive and incremental mining. The experiments were performed on two real dense (chess and mushroom) and one synthetic sparse datasets.

- *CSP* (Conjunctive Sequence Pattern mining) [Raïssi 08]. The basic idea of the algorithm is to alternate between a sequence mining task and a generation of all possible conjunctions. The data used in experiments was a synthetic data set containing 200,000 sequences based on 10,000 items, and the UNIX User Data set containing 9 sets of sanitized user data drawn from the command histories of 8 UNIX computer users at Purdue University.

- *DSM-FI* (Data Stream Mining for Frequent Itemsets), based on Lossy Counting [Li 08b]. It is composed of four steps: (1) reading a block of transactions; (2) constructing the summary data structure; (3) pruning the infrequent information from the summary data structure; (4) and top-down frequent itemset discovery scheme. Steps 1 and 2 are performed in sequence for a new block. Steps 3 and 4 are usually performed periodically or when it is needed. Two synthetic data sets were used. The first one had an average transaction size T of 10 items and the average size of frequent itemset I is 5-items. In the second data set, the average transaction size T and average frequent itemset size I are set to 30 and 20, respectively. Both of them have 1,000,000 transactions.

- *ExMiner* [Quang 06].

- *FastUP* (Enhanced GSP) [Lin 98]. It is used for the continuously updating databases. The idea is to count over appended data sequences instead of the entire updated database, and fast filtering of patterns found in last mining and successive reductions in candidate sequences. The algorithm starts with finding frequent 1-sequences in updated database. After that, at each step $k$, it finds frequent $k$-sequences in updated database.

- *FIDS* (Frequent Itemsets Mining on Data Streams) [Raïssi 07]. It is used for the continuously updating databases. The algorithm has 2 main steps: (1) the insertion of each itemset of the studied batch in teh data structure $Lattice_{reg}$ using a region principle; (2) the extraction of the maximal subset. The stream data was generated from Web Server Log Data of the ECML/PKDD Challenge 2005. These data comes from a Czech company running several internet shops. The log data cover the traffic on the web server of about 3 weeks. This represents about three million records.

- *FP-Stream* [Giannella 04]. A compact tree is used to represent the frequent pattern set. Each node in teh frequent pattern tree represents a pattern and its frequency is recorded in the node. This tree shares the similar structure with FP-tree. The difference is that it stores frequent patterns instead of streaming data. An interesting idea is the use of natural and logarithmic titled-time windows. The stream data was generated by the IBM synthetic market-basket data generator. In all the experiments 3M transactions were generated using 1K distinct items.

- *FP-Tree* (Frequent Pattern Tree), is an extended prefix-tree structure, and developes FP-Growth [Han 00b]. There are 2 basic steps: (1) scan the transaction database once, collect the set of frequent items F and their supports; sort F in support descending order as L, the list of frequente items. (2) Create the root of an FP-tree, T, and label it as "null". For each transaction *Trans* in DB do: select and sort the frequent items in *Trans* according to the order of $L$. There are 2 synthetic data sets. The first one has 1K items, with an average transaction size and the average maximal potentially frequent itemset size are set to 25 and 10, respectively, while the number of transactions in the dataset is set to 10K. The second data set has 10K items.

- *FreeSpan* (Frequent pattern-projected Sequential pattern mining) [Han 00a]. Given a sequence database S and the support threshold epsilon, FreeSpan mine the complete set of sequential patterns as follows. (1) Scan S, find the set of frequent items in S, and (in frequency descending order) sort them into $f_list$. (2) Perform alternative-level projection mining which consists of the following steps: (a) construct a frequent item matrix by scanning the database once, (b) generate length-2 sequential patterns and the annotations on item-repeating patterns and projected databased, (c) scan database to generate item-repeating patterns and projected databases, and (d) do matrix projection mining on projected databases recursively, if there are still longer candidate patterns to be mined. The synthetic datasets used were generated using the standard procedure described by Srikant and Agrawal. The number of items were set to 10,000.

- *FTP-DS* (Frequent Temporal Patterns of Data Streams) [Teng 03]. It scans online transaction flows and generates candidate frequent patterns in real time. It has also a regression based compact pattern representation. There were 2 synthetic datasets, with 200 (100) items, 3 (5) average number of items per transaction, 1000 customers, and 500,000 transactions. A real data set, AlarmLog, was used also. It had 287 items, 1.6 numbers of items per transaction, 5788 customers, and 128,815 transactions.

- *Gamma test for feature selection* [Chuzanova 98]. It was used for feature selection and LSU rRNA classification according to RDP phylogenetic classes.

- *GSP* (Generalized Sequential Patterns) [Srikant 96]. The algorithm tries to generate as few candidates as possible while maintaining completeness. Another aspect is the way the support count for the candidate sequences is determined.

- *GTC* (Graph for Time Constraints), is the implementation of the TCLW algorithm (Time Constraints Level Wise), and the structure used for organizing candidate sequences is a prefix tree structure as in PSP [Masseglia 09].

- *HDM* (Heuristic Based Distributed Miner) [Masseglia 02]. Is used to get frequent behaviour patterns answering the Web Usage Mining problem in real time.

- *ISE* (Incremental Sequence Extraction) [Masseglia 03]. It is used for the continuously updating databases. The main feature is that the set of candidate sequences to be tested is reduced. It reuses the minimal information from the old frequent sequences, i.e. the support of frequent sequences. A synthetic data set was used.

- *ISM* (Interactive Sequence Mining), extension of SPADE, [Parthasarathy 99]. It is used for the continuously updating databases. The idea in interactive sequence mining is that the end user is able to query the database dor association rules at different values of support and confidence. Synthetic datasets were used.

- *ITL-Tree* (Interval Transaction-ids List Tree) [Amphawan 10]. Is a variation of PF-tree, but where an approximate periodicity is found instead of the exact periodicity as in PF-tree. Synthetic and real (Mushroom) datasets were used.

- *KNN* [Deshpande 02]

- *MAD-IDS* (Mobile Agent Using Data mining based Intrusion Detection System) [Brahmi 10]. It is an Intrusion Detection System. Its distributed structure comprises different agents which are able to move from one station to another, called: Sniffer, Filter, Misuse Detection, Anomaly Detection, Rule Mining and Reporter Agent. The traffic data DARPA was used.

- *Markov* [Deshpande 02]

- *Moment* (Maintaining Closed Frequent Itemsets by Incremental Updates) [Chi 04]. An efficient in-memory data structure, the closed enumeration tree (CET) is used to record all closed frequent itemsets in the current sliding window. CET also monitors the itemsets that form the boundary between closed frequent itemsets and the rest of the itemsets. A synthetic and a real dataset were used. The real one contains a few months of clickstream data from an e-commerce web sites. There are 59,602 transactions in the dataset. The sliding window was set to 50,000 and the experiment was done on 100 consecutive sliding windows. The number of distinct items is 497, the maximal transaction size is 267, the average transaction size is 2.5.

- *OExMiner* [Minh 06]. Is an optimized version of ExMiner, used to mine the top-k frequent patterns from a large scale dataset. It reduces the search space in comparison to the previous version.

- *PF-Tree* (Periodic Frequent pattern Tree), uses FP-Growth mining technique, [Tanbeer 09]. The PF-Tree contains a prefix-tree and a periodic-frequent item list, called the PF-list, consisting of each distinct item with relative support, periodicity, and a pointer pointing to the first node in the PF-tree carrying the item. The items in the tree are arranged in support-descending item order. Several synthetic and real datasets (chess, mushroom, and kosarak) were analyzed.

- *PrefixSpan* [Pei 04]. In comparison to FreeSpan it tries to avoid checking every possible combination of a potential candidate sequence. For example, since items within an element of a sequence can be listed in any order, without loss of generality, one can assume that they are always listed alphabetically. In this case the concept of prefix and suffix is introduced. A real data set was used, Gazelle from Blue Martini. It contains 29,369 customers' Web click-stream data provided by Blue

Martini Software company. It contains 35,722 sessions and 87,546 page views. The synthetic data set contains 200,000 customers, and the number of items is 10,000. The average number of items in a transaction is 2.5, and the average number of transactions in a sequence is 10.

- *PSP*, resumes general principles of GSP, but different intermediary data structure [Masseglia 98].
- *PriPSeP* (Privacy Preserving Sequential Patterns) [Kapoor 06].
- *RNN* [Graves 06].
- *SCM* (Schema Mining), based on OEM (Object Exchange Model), but makes structural regularities of semistructured data [Laur 00]. There are two steps: mapping phase, and mining phase.
- *Selective Markov Models*, based on All-Kth Order Markov models, contains *Frequency Pruned MM*, *Confidence Pruned MM*, *Error-Pruned MM* [Deshpande 04].
- *Seq-BOMA* [Minh 06]. It builds in adnvance a "large" FP-tree which can be used to mine top-k frequent patterns with any different values of top-k in order to save the computation time.
- *Seq-Miner* [Minh 06]. Only a very small number of items (compared to k) are examined to generate top-k frequent patterns in comparison to OExMiner.
- *SPADE* [Zaki 01]. The idea is to break the large search space into small, manageable chunks that can be processed independently in memory. This is accomplished via suffix-based equivalence classes. Synthetic data sets were used in the analysis.
- *String Kernels* (restricted gappy kernels, substitution kernels, wildcard kernels) [Leslie 04].
- *SVM* [Deshpande 02].
- *SuffixTree* [Wang 96].
- *TAG and heuristics* (Timed Automata with Granularities) [Bettini 98b]. The data set was the closing prices of 439 stocks for 517 trading days during the period between January 3, 1994, and January 11, 1996.
- *UFR* (Unexpected Fuzzy Recurrence) [Li 08a]. The algorithm accepts a belief base B, a sequence database D, and a minimum fuzzy degree threshold $\varsigma$ as input data, and outputs all unexpected sequences in D with respect to B and $\varsigma$. The evaluation was done on Web access logs. One was a large access log file of an online forum site, and another one was a large access log file of a mixed homepage hosting server.

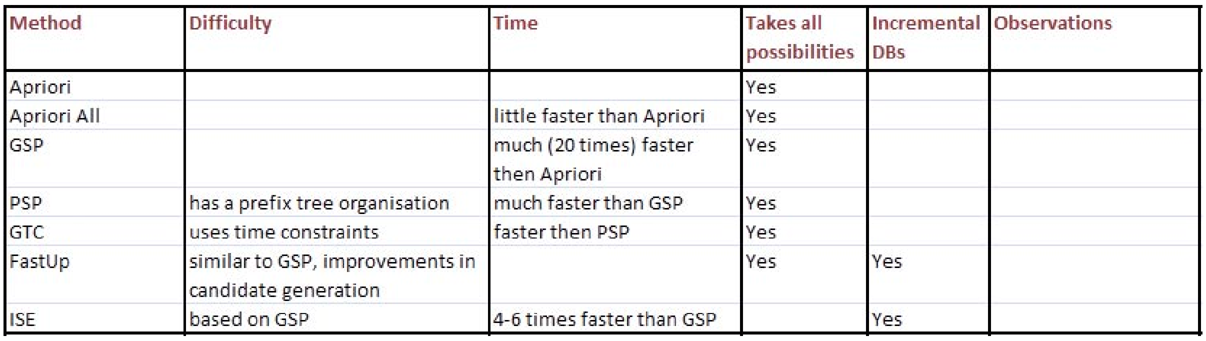
**Algorithms with unset IMs :**

As it has been presented, we are interested in algorithms and Interestingness Measures for sequential data mining. However, there are some algorithms that do not require a set IM for patterns and rules extraction. The only algorithms that we were able to find regarding this aspect, are the one used in *top-k* sequential mining. Almost all of them are based on *Lossy Counting* algorithms [Li 04]. The following methods are described: *TSP* [Tzvetkov 05], and the ones based on *ExMiner* algorithm [Quang 06]: *OExMiner, Seq-Miner, Seq-BOMA* [Minh 06].

However, there is one based also on the *Chernoff bound* with a guarantee of the output quality and also a bound on the memory usage [Wong 06].

## C Differences regarding complexity

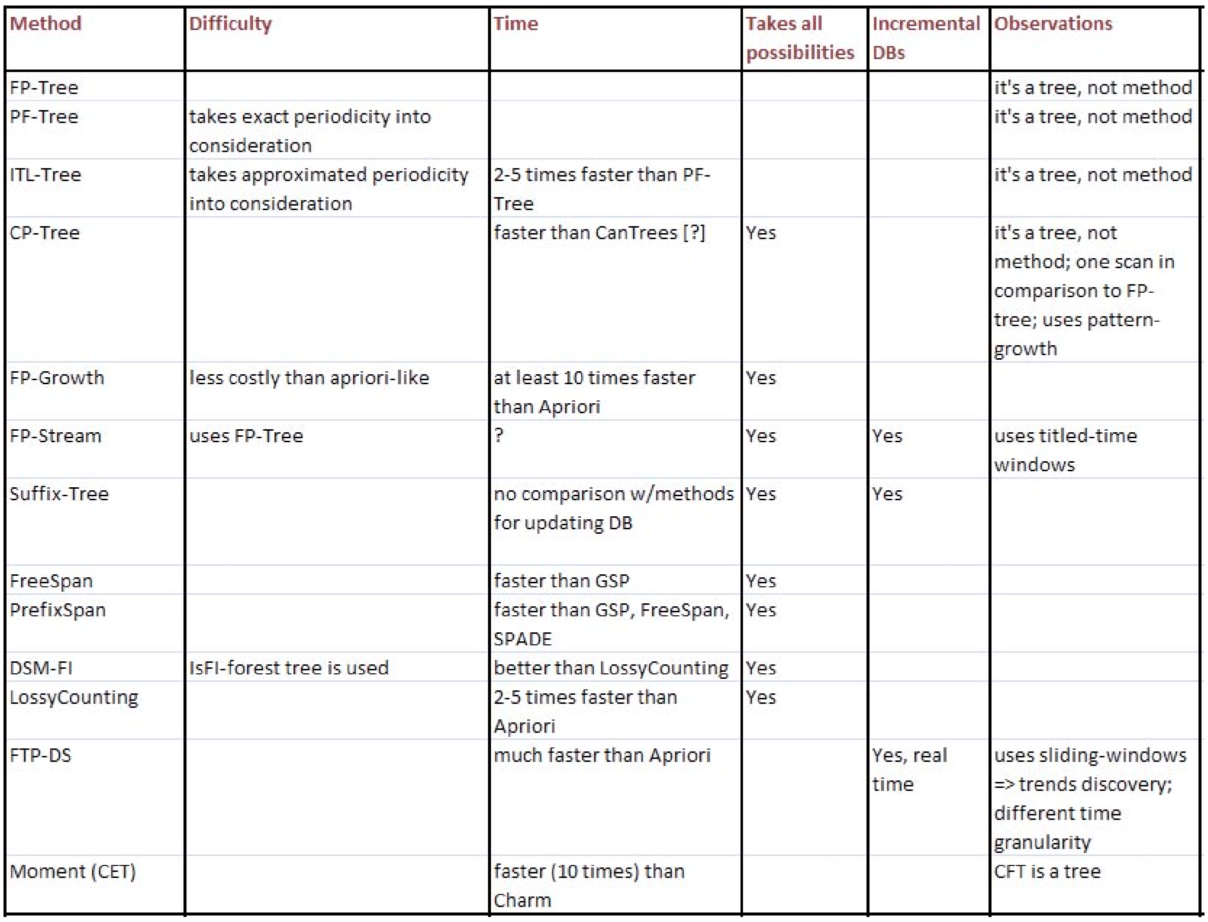
Comparisons between the algorithms from the same "family" are presented in Figures 2, 3, 4, and 5. You can notice the following characteristics: the noted difficulty in order to implement the method, the comparison times with other algorithms, if the algorithm does or does not take into consideration all the possibilities (if it has or not heuristic ideas), if it can be applied in incremental databases, and some important observations. In case of the *Takes all possibilities* and *Incremental Databases* columns, if nothing is written this means that it was not pointed out by the authors about these aspects.



| Method | Difficulty | Time | Takes all possibilities | Incremental DBs | Observations |
|---|---|---|---|---|---|
| Apriori | | | Yes | | |
| Apriori All | | little faster than Apriori | Yes | | |
| GSP | | much (20 times) faster then Apriori | Yes | | |
| PSP | has a prefix tree organisation | much faster than GSP | Yes | | |
| GTC | uses time constraints | faster then PSP | Yes | | |
| FastUp | similar to GSP, improvements in candidate generation | | Yes | Yes | |
| ISE | based on GSP | 4-6 times faster than GSP | | Yes | |

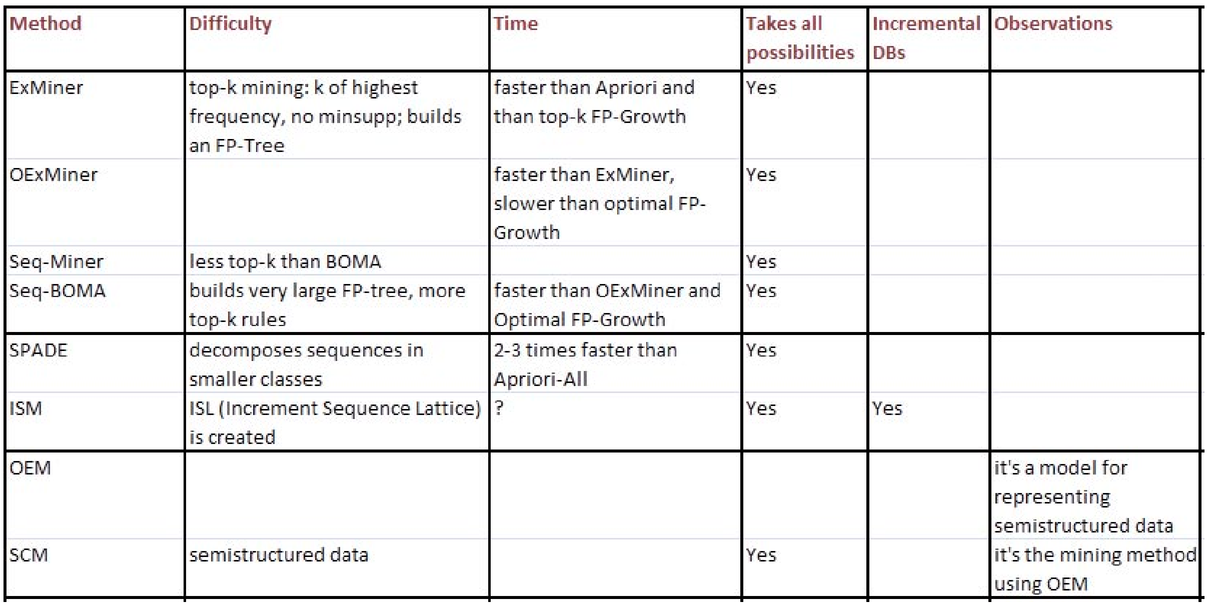
Figure 2 : First family

In Figure 6, the time comparison relative to *Apriori* algorithm is presented. You should take into consideration that the distance between methods does not represent the exact time difference (for example, if the ratio between the PSP and Apriori time seems to be 2 on the graph, it doesn't mean that it is 2 in reality, it only means that PSP is faster than Apriori).



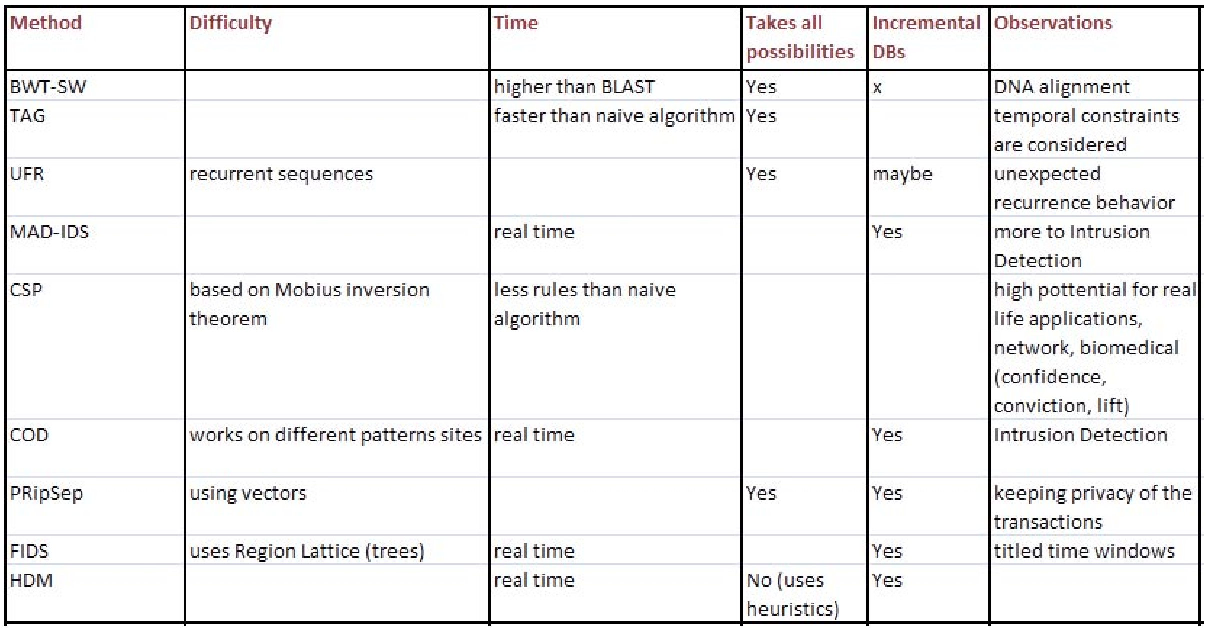
| Method | Difficulty | Time | Takes all possibilities | Incremental DBs | Observations |
|---|---|---|---|---|---|
| FP-Tree | | | | | it's a tree, not method |
| PF-Tree | takes exact periodicity into consideration | | | | it's a tree, not method |
| ITL-Tree | takes approximated periodicity into consideration | 2-5 times faster than PF-Tree | | | it's a tree, not method |
| CP-Tree | | faster than CanTrees [?] | Yes | | it's a tree, not method; one scan in comparison to FP-tree; uses pattern-growth |
| FP-Growth | less costly than apriori-like | at least 10 times faster than Apriori | Yes | | |
| FP-Stream | uses FP-Tree | ? | Yes | Yes | uses titled-time windows |
| Suffix-Tree | | no comparison w/methods for updating DB | Yes | Yes | |
| FreeSpan | | faster than GSP | Yes | | |
| PrefixSpan | | faster than GSP, FreeSpan, SPADE | Yes | | |
| DSM-FI | IsFI-forest tree is used | better than LossyCounting | Yes | | |
| LossyCounting | | 2-5 times faster than Apriori | Yes | | |
| FTP-DS | | much faster than Apriori | | Yes, real time | uses sliding-windows => trends discovery; different time granularity |
| Moment (CET) | | faster (10 times) than Charm | | | CFT is a tree |

Figure 3 : Second family



| Method | Difficulty | Time | Takes all possibilities | Incremental DBs | Observations |
|---|---|---|---|---|---|
| ExMiner | top-k mining: k of highest frequency, no minsupp; builds an FP-Tree | faster than Apriori and than top-k FP-Growth | Yes | | |
| OExMiner | | faster than ExMiner, slower than optimal FP-Growth | Yes | | |
| Seq-Miner | less top-k than BOMA | | Yes | | |
| Seq-BOMA | builds very large FP-tree, more top-k rules | faster than OExMiner and Optimal FP-Growth | Yes | | |
| SPADE | decomposes sequences in smaller classes | 2-3 times faster than Apriori-All | Yes | | |
| ISM | ISL (Increment Sequence Lattice) is created | ? | Yes | Yes | |
| OEM | | | | | it's a model for representing semistructured data |
| SCM | semistructured data | | Yes | | it's the mining method using OEM |

Figure 4 : Families 3, 4, and 5



| Method | Difficulty | Time | Takes all possibilities | Incremental DBs | Observations |
|---|---|---|---|---|---|
| BWT-SW | | higher than BLAST | Yes | x | DNA alignment |
| TAG | | faster than naive algorithm | Yes | | temporal constraints are considered |
| UFR | recurrent sequences | | Yes | maybe | unexpected recurrence behavior |
| MAD-IDS | | real time | | Yes | more to Intrusion Detection |
| CSP | based on Mobius inversion theorem | less rules than naive algorithm | | | high pottential for real life applications, network, biomedical (confidence, conviction, lift) |
| COD | works on different patterns sites | real time | | Yes | Intrusion Detection |
| PRipSep | using vectors | | Yes | Yes | keeping privacy of the transactions |
| FIDS | uses Region Lattice (trees) | real time | | Yes | titled time windows |
| HDM | | real time | No (uses heuristics) | Yes | |

Figure 5 : Family 6

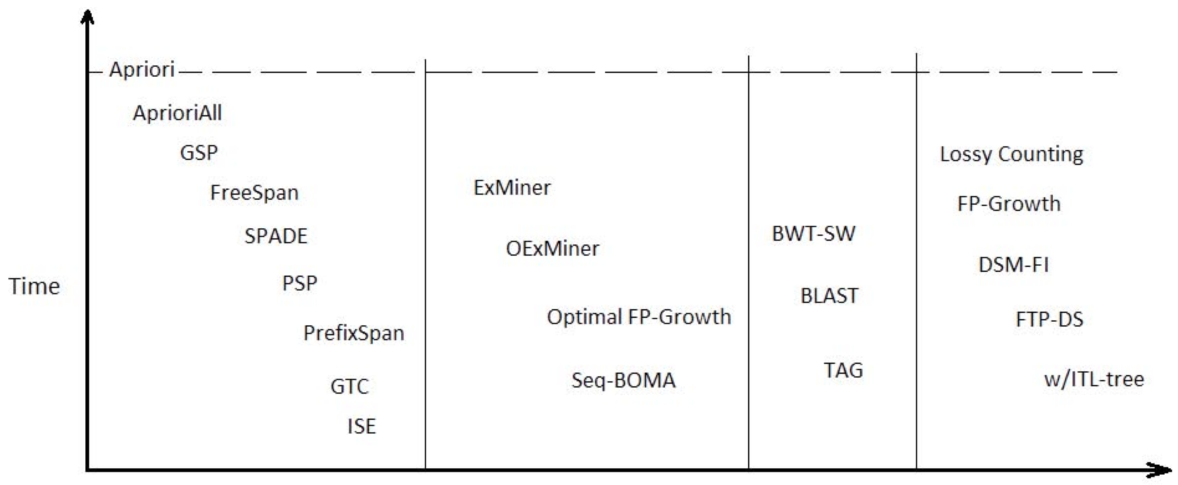


Figure 6 : Times Comparison of the algorithms. This is done relative to the *Apriori* method. The placements of the algorithms do not represent the real time ratios, they represent just that one is slower or faster than another. The times between different classes are not compared (for example, *FreeSpan* seems to be at the same level as *ExMiner* it does not mean that they have the same processing time)