MASTER

Finding redundant data sources in smart environments

El Harchi, M.

Award date:
2016

Link to publication
Finding Redundant Data Sources in Smart Environments

Master Thesis Report

By

Mohamed El Harchi
Student ID: 0646831

Graduation supervisor: Tanir Ozcelebi
Graduation tutor: Ehsan Ullah Warriach

January 26, 2016
Abstract

A smart environment is a physical space (e.g. home, office, hospital and airport) containing various smart devices that perform various tasks for users of this physical space. These smart devices can support several applications (e.g. lighting control application and air-conditioning control application) that gather, process and control data from a given environment. Nowadays, smart devices in a smart environment are rapidly increasing. These smart devices mostly use various sensors (e.g. light sensors, temperatures sensors and humidity sensors) to gather data from a given environment. Therefore, sensors have a crucial role. Currently, when a sensor fails, the application that uses this sensor will fail or will not work properly. It stays in this condition until a maintenance personnel replaces the failed sensor. Manual maintenance of a failed sensor decreases the availability and the reliability of the application that uses this sensor.

The key challenge is to replace a failed sensor in a smart environment with another sensor that can provide the same or highly correlated data from available sensors in a given environment without a manual intervention. In this thesis, we call this sensor a redundant data source. This allows the application to continue without interruption. As a result, maintenance costs will be reduced, and the reliability and the availability of the application will be increased.

In this thesis, the state of the art similarity and correlation measures have been studied for finding similarity and correlation between the output data of sensors, respectively. Based on this study, an approach has been proposed to find redundant data sources. The approach consists of two parts. The first part uses Shape Based Distance as a similarity measure in combination with Pearson Correlation measure to find redundant data sources that are similar. The second part uses Distance Correlation as a correlation measure to find redundant data sources that are highly correlated. Pearson Correlation and Distance Correlation use 0.8 and 0.9 as a threshold, respectively.

Further, a system architecture has been proposed that collects data from the given environment and replaces the failed sensor with the redundant data source. The proposed architecture was implemented on a wireless network platform, called MyriaModem.
Preface

I would like to thank my supervisor assistant Prof. Dr. Tanir Ozcelebi for offering me the possibility to do this master thesis. I am thankful to him for the time and effort that he put into advising me during this thesis. I would also like to thank Dr. Ehsan Warriach for his constant support, feedbacks, cooperation, and assistance on this thesis. Furthermore, I would like to thank MSc Sila Ozen for taking the time to review my thesis report and giving feedbacks. I would like to thank the secretary of the System Architecture and Networking (SAN) group for offering me a working place. Finally, I want to thank assistant Prof. Dr. Sander Stuijk for taking part in my examination committee.

This work has been supported by the ProHeal project funded by the Information Technology for European Advancement (ITEA2).
# Contents

1 Introduction ................................................. 5  
1.1 The problem statement ........................................ 5  
1.2 The research questions ......................................... 6  
1.3 Definitions and formal description of the problem statement ........................................ 6  
1.4 The proposed approach ......................................... 7  
1.5 Organization of the thesis ...................................... 8  

2 Literature Review ............................................. 9  
2.1 Similarity measures for time series ............................... 9  
2.1.1 Euclidean Distance ......................................... 9  
2.1.2 Dynamic Time Warping ...................................... 9  
2.1.3 Complexity Invariant Distance for ED and DTW ............... 10  
2.1.4 Shape Based Distance ...................................... 11  
2.1.5 Extended Jaccard ........................................... 11  
2.2 Correlation measures for time series ........................... 12  
2.2.1 Pearson Correlation ......................................... 12  
2.2.2 Distance Correlation ........................................ 12  
2.3 Conclusion ................................................... 13  

3 The System Architecture ....................................... 14  
3.1 MyriaModem .................................................. 14  
3.1.1 Communication of MyriaModem ............................... 15  
3.1.2 Sensors on MyriaModem ...................................... 16  
3.2 The proposed architecture ...................................... 17  
3.2.1 Deployment diagram ........................................ 17  
3.2.2 Activity diagram ............................................ 18  
3.2.3 Class diagram .............................................. 19  
3.2.4 Data format for redundant data sources .................... 22  

4 The Synthetic Data Model .................................... 24  
4.1 Synthetic data generation ...................................... 24  
4.2 Distortion generation ......................................... 25  
4.2.1 Uniform amplitude shifting .................................. 25  
4.2.2 Out of phase .............................................. 26  
4.2.3 Uniform amplitude scaling .................................. 26  
4.2.4 Noise .................................................... 27  
4.2.5 The combination of the distortions ......................... 28  

5 Performance Evaluation using Synthetic Data .................. 29  
5.1 Performance evaluation ....................................... 29  
5.1.1 One Nearest Neighbour (1-NN) classifier ................... 29  
5.2 Similarity measures .......................................... 30  
5.2.1 Results of the experiment ................................ 31  
5.2.2 Statistical analysis ........................................ 33  
5.3 Correlation measures ......................................... 35  
5.3.1 Results of the experiment ................................ 36  
5.4 Conclusion ................................................ 37  

Finding Redundant Data Sources  
in Smart Environments
Chapter 1

Introduction

1.1 The problem statement

A smart environment (SE) is a physical space containing various smart devices that perform various tasks for users of this physical space. In [1], SE has been defined as ‘a small world where different kinds of smart devices are continuously working to make this small world more comfortable to live in’. A smart device can support applications (e.g. lighting control application and air-conditioning control application) that gather, process and control data from a given environment. Smart devices are able to communicate with each other in a given environment. Figure 1.1 shows an example of an SE. It shows two rooms that are equipped with various sensors to monitor these rooms. A smart air-conditioning control application can use a temperature sensor to automatically adjust the room temperature, and a PIR sensor to turn off completely when a person is not present in the room. Usually, an SE is equipped with various sensors (e.g. presence sensors, temperature sensors, humidity sensors and light sensors) to monitor the given environment and to support applications. Therefore, sensors in an SE have a crucial role.

![Figure 1.1: A smart room](image)

Mostly, when a sensor fails, it leads to an application failure or performance degradation in the best case. In other words, the application that uses this sensor will not satisfy its application requirements or perform worse. This application stays in this failure state until the user reports it and a maintenance person investigates the failure state of the application and replaces the failed sensor with a new sensor.

Nevertheless, the manual replacement of the failed sensor has the following drawbacks:

1. The reliability and the availability of the application decreases due to maintenance delay.
2. The cost of ownership increases due to maintenance costs.
3. The users get frustrated (e.g. air-conditioning control application does not work properly).

A solution to avoid these drawbacks is to replace the failed sensor automatically for a temporary period until the maintenance personnel arrives and fixes the problem. The replacing of failed sensor refers to using the data of another working sensor from the given environment that can provide the same or highly correlated data. We call such a sensor a redundant data source. This allows the application to continue without interruption. As a result, the maintenance cost will
be reduced, the user of the given environment does not get frustrated, and the reliability and availability of the application will be increased.

1.2 The research questions

A redundant data source of a failed sensor is a data source (DS) from the environment (e.g. a temperature sensor), whose output data is similar or highly correlated to the output data of a failed sensor. The type of the redundant data source may be different from the type of a failed sensor. For example, if the output data of a temperature sensor is correlated with the output data of a humidity sensor, we may replace the temperature sensor with the humidity sensor. Data sources and sensors are used interchangeably in this thesis.

The definition of the redundant data source raises the following question:

- When do we consider an output data of a DS\(_1\) similar or highly correlated to an output data of DS\(_2\)?

We consider an output data of a DS\(_1\) similar to an output data of DS\(_2\) if the following holds:

- \(\text{similarity}\_\text{measure}(\text{DS}_1,\text{DS}_2) \leq \text{threshold}\_\text{similarity}\)

We consider an output data of a DS\(_1\) highly correlated to an output data of DS\(_2\) if the following holds:

- \(\text{correlation}\_\text{measure}(\text{DS}_1,\text{DS}_2) \geq \text{threshold}\_\text{correlation}\)

Similarity and correlation functions are defined by the corresponding similarity and correlation measures, respectively. A similarity measure calculates the similarity between two data sources. A correlation measure calculates the correlation between two data sources. The \(\text{threshold}\_\text{similarity}\) and \(\text{threshold}\_\text{correlation}\) determine when two data sources can be considered similar or highly correlated, respectively.

In this thesis, the following questions are investigated:

1. Which similarity measure is the most suitable to find similarity between two data sources?
2. Which correlation measure is the most suitable to find correlation between two data sources?
3. What value of \(\text{threshold}\_\text{similarity}\) is an appropriate threshold?
4. What value of \(\text{threshold}\_\text{correlation}\) is an appropriate threshold?

1.3 Definitions and formal description of the problem statement

Time series \(T=\{t_1,\ldots,t_M\}\) is the output data of a data source. It is a sequence of \(M\) real valued data points, which are observed at certain equidistant points in time. Time series and signals are used interchangeably in this thesis. Table 1.1 summarizes the notations that are used in this thesis.

An SE can be defined as a quadruple \((\text{App}\_\text{Set}, \text{DS}\_\text{Set}, \text{State}\_\text{App}, \text{State}\_\text{DS})\), which contains:

- a set of applications (e.g. air-conditioning application) \(\text{App}\_\text{Set} = \{\text{App}_1,\ldots,\text{App}_L\}\),
- a set of data sources (e.g. light sensors and temperatures sensors) \(\text{DS}\_\text{Set} = \{\text{DS}_1,\ldots,\text{DS}_N\}\), where \(\text{DS}_i\) and \(\text{DS}_j\) may have the same type or different types, \(i \neq j\) and \(1 \leq i,j \leq N\),
- an application state function \(\text{State}\_\text{App}(\text{App}_l) \in \{\text{TApp}, \text{FApp}\}\), where \((\text{TApp} : \text{True}\_\text{App}, \text{FApp} : \text{False}\_\text{App})\). \(\text{State}\_\text{App}(\text{App}_l) = \text{TApp}\) means that \(\text{App}_l\) works correctly, and \(\text{State}\_\text{App}(\text{App}_l) = \text{FApp}\) means \(\text{App}_l\) does not work or does not work properly.
CHAPTER 1. INTRODUCTION

- and a data source state function \( \text{State}_{DS}(DS_i) \in \{T_{DS}, F_{DS}\} \), where \( T_{DS} : \text{True}DS, F_{DS} : \text{False}DS \). \( \text{State}_{DS}(DS_i) = T_{DS} \) means that \( DS_i \) works correctly, and \( \text{State}_{DS}(DS_i) = F_{DS} \) means that \( DS_i \) does not work.

In this thesis, we want to replace the failed sensor \( DS_i, \text{State}_{DS}(DS_i) = F_{DS} \), with another sensor \( DS_j, \text{State}_{DS}(DS_j) = T_{DS}, i \neq j \), from the available sensors \( DS_{Set} \) in the given SE, such that:

- \( \text{similarity\_measure}(DS_i, DS_j) \leq \text{threshold\_similarity} \)
- \( \text{correlation\_measure}(DS_i, DS_j) \geq \text{threshold\_correlation} \).

The \( \text{threshold\_similarity} \) and \( \text{threshold\_correlation} \) determine when two sensors can be considered similar or highly correlated, respectively.

The work of this thesis focuses on identifying redundant data sources and the \( \text{State}_{DS} \) and \( \text{State}_{App} \) functions will be investigated in the future work.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS</td>
<td>Data Source</td>
</tr>
<tr>
<td>T</td>
<td>Time series</td>
</tr>
<tr>
<td>( t_m )</td>
<td>m-th element of T</td>
</tr>
<tr>
<td>( M )</td>
<td>the length of the time series</td>
</tr>
<tr>
<td>( N )</td>
<td>the number of data sources</td>
</tr>
<tr>
<td>( T_{DS,n} )</td>
<td>the output data of the data source n</td>
</tr>
<tr>
<td>( t_{m,n} )</td>
<td>m-th element of ( T_{DS,n} )</td>
</tr>
</tbody>
</table>

Table 1.1: List of notations

1.4 The proposed approach

In order to find redundant data sources in a SE, our proposed approach consists of the following steps:

- Study the literature about the state of the art similarity and correlation measures,
- Generate synthetic data sets that represent the data in the real world,
- Collect real data from different sensors deployed at two different locations using MyriaNed wireless network platform,
- Compare the state of the art similarity measures using the synthetic and real data sets,
- Compare the state of the art correlation measures using the synthetic and real data sets,
- Conduct an experiment to determine \( \text{threshold\_correlation} \) using the real data.
1.5 Organization of the thesis

The organization of the thesis is as follows:
In Chapter 2, we provide an overview of the state of the art similarity and correlation measures. Afterwards, we describe the proposed system architecture that collects data from the given environment and replaces a failed sensor in Chapter 3. The synthetic data that are used to evaluate the performances of the similarity and correlation measures are presented in Chapter 4. Furthermore, the performances of the similarity and correlation measures are evaluated in Chapter 5. Next, in Chapter 6 we explain steps for finding redundant data sources and finding appropriate thresholds. Then, in Chapter 7 we show with a scenario how redundant data sources affect the availability and the reliability of a lighting control system application. Finally, in Chapter 8 we conclude this thesis and provide suggestions for future work.
Chapter 2

Literature Review

2.1 Similarity measures for time series

Several similarity measures have been proposed in the literature. In the following section, we present the similarity measures that are most commonly used in the literature.

2.1.1 Euclidean Distance

The Euclidean Distance (ED) is the most popular similarity measure used in the literature \[2, 3, 4\]. It is used due to its simplicity and its time complexity \(O(M)\) \[10\], where \(M\) is the length of the time series or the number of samples. ED is defined as follows:

\[
ED(T^{DS,1}, T^{DS,2}) = \sqrt{\sum_{i=1}^{M} (t^{DS,1}_i - t^{DS,2}_i)^2}
\]

However, ED has a drawback. It gives a poor result when two time series are out of phase or when there is noise in the data \[5, 6, 7, 8\].

2.1.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is another similarity measure to compare two time series. It was first proposed in \[9\] for speech recognition. In contrast to ED, DTW is able to find similarity between two time series even if they are out of phase and do not have the same length \[6, 8\]. DTW used in \[10\] to find similarity in databases.

DTW works as follows:

A \(M_{T1}\) by \(M_{T2}\) matrix \(D\) is constructed as shown in Figure 2.1, where \(M_{T1}\) and \(M_{T2}\) are the length of the time series \(T^{DS,1}\) and \(T^{DS,2}\), respectively. The element of the matrix \(D\) at the \(i\)-th row and the \(j\)-th column is \(ED(t^{DS,1}_i, t^{DS,2}_j)\), where \(1 \leq i \leq M_{T1}\) and \(1 \leq j \leq M_{T2}\). Further, we take a path \(W\) from \(i = j = 1\) until \(i = M_{T1}\) and \(j = M_{T2}\). The path \(W\) is a set of elements, \(W = \{w_1, w_2, ... w_k\}\), where \(1 \leq k \leq K\), \(\max(M_{T1}, M_{T2}) \leq K < M_{T1} + M_{T2} - 1\) and \(w_k\) is an element of the matrix \(D\). The path \(W\) should satisfy the following constraints:

- **Boundary conditions:**
  \(w_1 = D(1, 1)\) and \(w_K = D(M_{T1}, M_{T2})\),

- **Continuity:**
  Given \(w_k = D(i, j)\) then \(w_{k-1} = D(i', j')\) where \(i - i' \leq 1\) and \(j - j' \leq 1\),

- **Monotonicity:**
  Given \(w_k = D(i, j)\) then \(w_{k-1} = D(i', j')\) where \(i - i' \geq 0\) and \(j - j' \geq 0\).

However, there are many paths for \(W\) to satisfy the above constraints. We take only the path that minimizes the following:

\[
DTW(T^{DS,1}, T^{DS,2}) = \min_W \left( \sqrt{\sum_{k=1}^{K} (w_k)} \right)
\]
This path is also computed using the following approach:
After constructing the matrix $D(i,j) = ED(t_{DS,1}^i, t_{DS,2}^j)$, where $1 \leq i \leq M_{T1}$ and $1 \leq j \leq M_{T2}$. Another $M_{T1} \times M_{T2}$ matrix $Store\_Matrix$ is constructed and the elements of this matrix are as follows computed:

- $Store\_Matrix(1,1) = D(1,1)$.
- $Store\_Matrix(i, 1) = D(i, 1) + Store\_Matrix(i-1,1)$, where $2 \leq i \leq M_{T1}$.
- $Store\_Matrix(1, j) = D(1, j) + Store\_Matrix(1,j-1)$, where $2 \leq j \leq M_{T2}$.
- $Store\_Matrix(i, j) = D(i,j) + \min(Store\_Matrix(i-1,j-1), Store\_Matrix(i-1,j), Store\_Matrix(i,j-1))$, where $2 \leq i \leq M_{T1}$ and $2 \leq j \leq M_{T2}$.

This approach is called dynamic programming. Then $DTW(T_{DS,1}^{T_{DS,2}}) = Store\_Matrix(M_{T1}, M_{T2})$. However, large time series may decrease the performance of DTW, the time complexity of DTW is $O(M_{T1} \times M_{T2})$. It has been shown that for large time series, DTW is not significantly better than ED [7].

2.1.3 Complexity Invariant Distance for ED and DTW

Authors in [11] claimed that if two time series have different complexities and seem very similar to the human eye, these two time series tend to be not similar with the current similarity measures. In [11], the authors have proposed a novel approach to detect similarity between time series even if the time series have different complexities. The approach is called Complexity Invariant Distance (CID). CID uses information about complexity differences between two time series as a correction factor for ED and DTW. CID for ED and DTW is defined as follows:

- $CID_{ED}(T_{DS,1}^{T_{DS,2}}) = ED(T_{DS,1}^{T_{DS,2}}) \times CF(T_{DS,1}^{T_{DS,2}})$
- $CID_{DTW}(T_{DS,1}^{T_{DS,2}}) = DTW(T_{DS,1}^{T_{DS,2}}) \times CF(T_{DS,1}^{T_{DS,2}})$,
where CF is a complexity correction factor is defined as:

$$CF(T_{DS1}, T_{DS2}) = \frac{\max(CE(T_{DS1}), CE(T_{DS2}))}{\min(CE(T_{DS1}), CE(T_{DS2}))}$$

CE estimates the complexity of a time series and is defined as:

$$CE(T_{DS1}) = \sqrt{M - 1 \sum_{i=1}^{M-1} (t_{DS1,i} - t_{DS1,i+1})^2}$$

### 2.1.4 Shape Based Distance

Recently, a novel clustering algorithm, named k-shape, has been proposed in [12]. Similarity measures have a crucial role in clustering algorithms. As a similarity measure for k-shape clustering, in [12], they used a normalized version of the Cross-Correlation measure (CC). They named a Shape Based Distance (SBD) similarity measure.

CC is defined as follows:

$$CC(T_{DS1}, T_{DS2}) = CC_{w-M}(T_{DS1}, T_{DS2})$$

where \(w \in \{1, 2, ..., 2M - 1\}\) and \(CC_{w-M}(T_{DS1}, T_{DS2})\) defined as:

$$CC_{w-M}(T_{DS1}, T_{DS2}) = \begin{cases} 
\sum_{i=1}^{M-(w-M)} (t_{DS1,i+M} \times t_{DS2,i}), & \text{if } (w - M) \geq 0 \\
CC_{-(w-M)}(T_{DS2}, T_{DS1}), & \text{if } (w - M) < 0 
\end{cases}$$

SBD defined as:

$$SBD(T_{DS1}, T_{DS2}) = 1 - \max_w \left( \frac{CC_w(T_{DS1}, T_{DS2})}{\sqrt{CC_0(T_{DS1}, T_{DS1}) \times CC_0(T_{DS2}, T_{DS2})}} \right)$$

Further in [12], the authors have showed that SBD is as competitive as DTW and ED. Further, they have showed that SBD is significantly faster than DTW.

### 2.1.5 Extended Jaccard

Extended Jaccard (ExtJaccard) \([13, 14, 15]\) is another similarity measure and is defined as follows:

$$ExtJaccard(T_{DS1}, T_{DS2}) = \frac{\sum_{i=1}^{M} (t_{DS1,i} \times t_{DS2,i})}{\sum_{i=1}^{M} (t_{DS1,i}^2) + \sum_{i=1}^{M} (t_{DS2,i}^2) - \sum_{i=1}^{M} (t_{DS1,i} \times t_{DS2,i})}$$

The range of ExtJaccard is between 0 and 1. The value 0 means that the time series \(T_{DS1}\) and \(T_{DS2}\) are not similar and the value 1 means that the time series \(T_{DS1}\) and \(T_{DS2}\) are similar.
2.2 Correlation measures for time series

In this section, we review the correlation measures for time series.

2.2.1 Pearson Correlation

The Pearson Correlation (PCor) measure is the most commonly used in the literature for time series [16, 17, 18, 19, 20] to find correlation between time series. The PCor measures the strength and direction of the linear relationship between two time series. The PCor between two time series $T^{DS,1}$ and $T^{DS,2}$ is computed by:

$$PCor(T^{DS,1}, T^{DS,2}) = \frac{\sum_{i=1}^{M} (t^{DS,1}_{i} - \bar{T}^{DS,1})(t^{DS,2}_{i} - \bar{T}^{DS,2})}{\sqrt{\sum_{i=1}^{M} (t^{DS,1}_{i} - \bar{T}^{DS,1})^2} \times \sqrt{\sum_{i=1}^{M} (t^{DS,2}_{i} - \bar{T}^{DS,2})^2}}$$

where

$$\bar{T}^{DS,1} = \frac{\sum_{i=1}^{M} t^{DS,1}_{i}}{M} \quad \text{and} \quad \bar{T}^{DS,2} = \frac{\sum_{i=1}^{M} t^{DS,2}_{i}}{M}$$

As shown in Figure 2.2, the range of PCor is between -1 and 1. Positive values indicate a positive linear correlation and negative values indicate a negative linear correlation. A value of 0 indicates that there is no linear relation between the time series $T^{DS,1}$ and $T^{DS,2}$.

However, PCor has a drawback. It cannot detect a non-linear relation between time series [21, 22]. A value 0 does not imply no relation between time series. If the relation between two time series has a wave pattern or a diamond pattern as shown in Figure 2.2, PCor is not able to detect these kinds of relation.

![Figure 2.2: Pearson Correlation: relation between two time series](https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient)

2.2.2 Distance Correlation

A new correlation measure was introduced in [21, 23, 24, 25] to overcome the drawback of the PCor. The new correlation method is called Distance Correlation (DCor) and is defined by:

$$DCor(T^{DS,1}, T^{DS,2}) = \frac{\sum_{i=1}^{M} ||t^{DS,1}_{i} - \bar{T}^{DS,1}|| ||t^{DS,2}_{i} - \bar{T}^{DS,2}||}{\sum_{i=1}^{M} ||t^{DS,1}_{i} - \bar{T}^{DS,1}||^2 \times \sum_{i=1}^{M} ||t^{DS,2}_{i} - \bar{T}^{DS,2}||^2}$$

where

$$||t^{DS,1}_{i} - \bar{T}^{DS,1}|| = \sqrt{\sum_{i=1}^{M} (t^{DS,1}_{i} - \bar{T}^{DS,1})^2} \quad \text{and} \quad ||t^{DS,2}_{i} - \bar{T}^{DS,2}|| = \sqrt{\sum_{i=1}^{M} (t^{DS,2}_{i} - \bar{T}^{DS,2})^2}$$

As shown in Figure 2.2, the range of DCor is between -1 and 1. Positive values indicate a positive non-linear correlation and negative values indicate a negative non-linear correlation. A value of 0 indicates that there is no non-linear relation between the time series $T^{DS,1}$ and $T^{DS,2}$.

However, DCor is not able to detect these kinds of relation.

\[
DCor(T^{DS,1}, T^{DS,2}) = \frac{\text{Dis}_Cov(T^{DS,1}, T^{DS,2})}{\sqrt{\text{Dis}_\text{Var}(T^{DS,1}) \times \text{Dis}_\text{Var}(T^{DS,2})}}
\]

where

\[
\text{Dis}_Cov(T^{DS,1}, T^{DS,2}) = \frac{1}{M} \sqrt{\sum_{k,l=1}^{M} A_{kl} \times B_{kl}}
\]

is the distance covariance between \(T^{DS,1}\) and \(T^{DS,2}\), and

\[
\text{Dis}_\text{Var}(T^{DS,1}) = \frac{1}{M} \sqrt{\sum_{k,l=1}^{M} A_{kl}^2}, \quad \text{Dis}_\text{Var}(T^{DS,2}) = \frac{1}{M} \sqrt{\sum_{k,l=1}^{M} B_{kl}^2}
\]

is distance variance for \(T^{DS,1}\) and \(T^{DS,2}\), respectively. \(A\) and \(B\) are matrices for \(T^{DS,1}\) and \(T^{DS,2}\), respectively. The element at the \(k\)-th row and the \(l\)-th column of these matrices is computed as follows:

\[
a_{kl} = a_{kl} - \bar{a}_k - \bar{a}_l + \bar{a},
\]

\[
b_{kl} = b_{kl} - \bar{b}_k - \bar{b}_l + \bar{b}.
\]

The DCor measures the linear and non-linear relation between two time series. As shown in Figure 2.3, the range of DCor is between 0 and 1. A value 0 means that there is no linear and non-linear relation between the time series \(T^{DS,1}\) and \(T^{DS,2}\). This is an advantage over the PCor. The DCor gives more information about the relation between two time series.

![Figure 2.3: Distance Correlation: relation between two time series](https://en.wikipedia.org/wiki/Distance_correlation)

\[\text{Figure 2.3: Distance Correlation: relation between two time series}^{2}\]

### 2.3 Conclusion

The choice of a similarity measure depends on the application domain and on the characteristics of the time series [7]. There is no similarity measure suitable for every application [7].

Regarding the correlation measures, DCor gives more information about the relation between two time series than PCor. However, DCor does not give information about the direction of the correlation.

---

Chapter 3

The System Architecture

This chapter introduces sensors and a wireless network platform node that are used in collecting and processing data from a given environment. Further, it also introduces the proposed system architecture that collects data from a given environment, finds redundant data sources and replaces a failed sensor.

3.1 MyriaModem

MyriaModem\(^3\) is used as a wireless network platform node. It used because of its following features:

- Low power,
- Ad hoc,
- Highly scalable.

In the following section, these features are explained in more details. MyriaModem is developed by DevLab in the Netherlands and has an open hardware platform. This means it is possible to implement different software architectures on it. As shown in Figure 3.1, MyriaModem consists of the following:

- low power embedded microcontroller,
- low power 2.4 GHz radio chip,
- battery holder and battery ON-OFF switch,
- 3 user LEDs in the colors red, orange and green,
- 2 user push buttons A and B,
- an expansion connector of 17 pins.

![Figure 3.1: MyriaModem\(^3\)](image)
3.1.1 Communication of MyriaModem

To communicate with other MyriaModem, a MyriaNed stack as a communication protocol is used. The key features of this protocol are:

- **Low power**
  In order to consume very little energy, MyriaNed protocol, as shown in Figure 3.2, puts the MyriaModem in an active and in a sleep mode repeatedly. The interval between the starts of two consecutive active modes is called a round. In the start of each round or in the active mode, MyriaModem starts the radio communication and talks with its neighbours. With its neighbours, we mean other MyriaModem in the given environment.

- **Ad hoc**
  To communicate correctly with the neighbouring nodes, the nodes should wake up and sleep at the same time. The start of each round of all neighbours must be synchronized. Otherwise, it is not possible to talk to each other. To do that, as shown in Figure 3.3, the MyriaModem nodes send randomly a join message during the sleep mode. After receiving the join message, the MyriaModem nodes will align their timing and be synchronized.

- **Highly scalable**
  The communication range of the MyriaModem is 20 meters. A direct communication is not possible with MyriaModem nodes that are further than 20 meters. To communicate with these nodes, a mechanism called gossip is invented. The gossip mechanism makes sure that all nodes pass the received messages to each other. For example in Figure 3.4, the node C in round 1 can only send a message to nodes A, B and D. In round 2 and with using the gossip mechanism, the node E receives the message of node C through node D.

Further, MyriaNed consists of layers. Figure 3.5 shows these layers:

- radio layer is responsible for sending data in the air,
- neighbor layer is responsible for communication with in range neighbours,
gossip layer is responsible for processing the received data and passes the received message to other nodes,

• application layer is responsible for performing applications on the MyriaModem,

• startup and main loop layer is responsible for scheduling between layers.

3.1.2 Sensors on MyriaModem

Sensors are crucial for SE. With sensors, we can monitor the given environment. MyriaModem can support a large number of sensors. In this thesis, the following sensors are used:

• PIR sensor:
  – detects the presence of a person in the given environment.

• light sensor:
  – measures the light intensity in the given environment.

• humidity sensor:
  – measures the humidity in the given environment.

• temperature sensor:
  – measures the temperature in the given environment.

Table 3.1 gives the abbreviation of the sensors. Figure 3.6 shows these sensors.
CHAPTER 3. THE SYSTEM ARCHITECTURE

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>TMP</td>
</tr>
<tr>
<td>Humidity</td>
<td>HUM</td>
</tr>
<tr>
<td>Light passive infrared sensor</td>
<td>Light</td>
</tr>
<tr>
<td></td>
<td>PIR</td>
</tr>
</tbody>
</table>

Table 3.1: List of sensors and their abbreviation

![The sensors](image)

Figure 3.6: The sensors

3.2 The proposed architecture

In this section, we talk about the proposed system architecture that collects data from a given environment, finds redundant data sources and replaces a failed sensor. The proposed system architecture is presented as a deployment diagram, an activity diagram and a class diagram.

3.2.1 Deployment diagram

Figure 3.7 depicts a deployment diagram of the proposed architecture. The deployment diagram shows the hardware aspect of the proposed architecture. As shown in Figure 3.7, it consists of three component types:

1. redundant data source computation component,
2. RxTx node component,
3. several sensor sensing node components.

**Redundant data source computation server component:**

This component is responsible for computing the redundant data sources of each sensor in the given environment. This component runs, in this thesis, on a personal computer and is implemented in MATLAB. Before it computes the redundant data sources, it gathers data of each sensor from the RxTx node through a serial interface. Data represents the output of a sensor for a certain period of time. This period of time is user defined. In our case, the user is the designer of the application for the given SE. After finding the redundant data sources of each sensor node, the component sends these redundant data sources to the RxTx node through a serial interface.
CHAPTER 3. THE SYSTEM ARCHITECTURE

RxTx node component:
This component is implemented on the MyriaModem and it is responsible for collecting and sending data from and to the sensor sensing nodes and the redundant data sources computation component. Before it sends data to the redundant data source computation component, it pre-processes the data. This data pre-processing is needed because the sensor sensing nodes in the given environment send their sensor value to RxTx node at the same time. Therefore, a collision may occur and the time stamps of the sensor values may not match. To solve this problem, we sum the received sensors’ values and divide by the number of these values. Further, RxTx node receives redundant data sources from the redundant data source computation component through a serial interface and passes them to sensor sensing nodes component through a wireless communication.

Sensor sensing node component:
This component is implemented on the MyriaModem and is responsible for the following:
- Monitoring a given environment using sensors (e.g. light sensors),
- Sending the value of the sensor to its neighbours sensor node and to RxTx Node,
- Receiving data from its neighbours nodes and RxTx node,
- Running an application (e.g. lighting application that turns light ON and OFF).

![Deployment diagram](image)

Figure 3.7: Deployment diagram

3.2.2 Activity diagram

In this section, more details about the above components are given. Figure 3.8 shows the activity diagram of these components. This diagram shows dynamic aspects of the proposed architecture.
CHAPTER 3. THE SYSTEM ARCHITECTURE

Redundant data source computation server component:
First of all, the redundant data source computation component waits until \( n \) samples of all sensors are received. The sample size is defined by the designer. Then, the computation of finding redundant data sources is started. The computation is done in three steps:

1. Z-normalization.
2. similarity measure in combination with PCor to find redundant data sources that are similar.
3. correlation measure to find redundant data sources that are correlated.

After these steps, redundant data sources of every sensor in the given environment is computed and are ready to be sent to RxTx node through a serial interface.

RxTx node component:
First of all, RxTx node component initializes MyriaModem to make it ready for sending and receiving data. Then, it waits for data from both the sensor sensing nodes and the redundant data source computation. Data from the redundant data source computation is received through a serial interface and immediately send to all sensor sensing nodes. As shown in Figure 3.9, RxTx node repeatedly receives sample values from the sensor sensing nodes. After a TimeInterval, the RxTx node takes an average of these sample value for each sensor. Taken the average is necessary because the time stamps of the sample value does not match with each other. These differences in the time stamps makes the comparison between sensors useless. Figure 3.9 shows how the average calculation is computed. The sample values from each sensor are summed and divided by the count of these sample values within each TimeInterval. After taking the average, the average sample values are sent to redundant data source computation components.

Sensor sensing nodes component:
First of all, sensor node component initializes MyriaModem. Then, it waits for data from botch the RxTx node and the neighbouring nodes. If no data are received, the sensor sensing node sends its value to both RxTX node and the neighbouring sensor sensing node. Further, the node uses this value to support the applications (e.g. lighting control system application) that runs on it. The node is repeatedly receiving data from both the RxTx node and the neighbouring nodes. When the node receives data from the RxTx node, it checks if the data was intended for its own use. Otherwise, it ignores it. This data contains the ID number of its redundant sensor sensing node. Further, the node uses the data of the neighbouring sensor node only if the ID number of this neighbouring sensor matches with the ID number of the redundant sensor.

3.2.3 Class diagram
Figure 3.10 shows the class diagram of the proposed architecture. It shows the above components as objects along with their attributes and methods. It also shows the collaboration among the objects.

Redundant data source computation server component:
As shown in Figure 3.10, the RDS computation, Redundant Data Source (RDS), object uses Find_RDS() method to compute the redundant sources of every sensor in the given SE. To do that, Number_Of_Samples of sensors should be defined. It is user defined, and the data of every sensor is stored in Samples_Storage. This Samples_Storage is a txt file and is retrieved from the SerialInterface object. Further, the RDS computation object uses the ZNormalization object, the SimilarityMeasure object and the CorrelationMeasure object to find the redundant data sources. After finding the redundant data sources, the ID number of every sensor is stored along with the ID number of the redundant sources in ID_RDS_Storage. The ID_RDS_Storage is a txt file. The SerialInterface retrieves the content of the ID_RDS_Storage and sends this content to RxTx node or object.
CHAPTER 3. THE SYSTEM ARCHITECTURE

Finding Redundant Data Sources in Smart Environments

Figure 3.8: Activity diagram

Figure 3.9: Data aggregation (average)
CHAPTER 3. THE SYSTEM ARCHITECTURE

RxTx node component
As shown in Figure 3.10, the RxTx object has two attributes NodeID and IntervalTime. This object is implemented on the MyriaModem. Every MyriaModem has a unique NodeID number. IntervalTime is user defined. This is the interval time between sample values of a sensor. Every IntervalTime the RxTx object sends a sample value of the sensors in the given SE to redundant data sources computation component. Further, it has the following methods:

- **InitializeRxTxNode()**:
  - This method initializes MyriaModem (e.g. MyriaNed Protocol, serial interface).

- **Receive_SampleValue_Sensors()**:
  - This method receives sample values of the sensors node. It stores every sample values along with the ID number of the sensor node.

- **DataAggregation()**:
  - Every IntervalTime, this method computes an average sample for every sensor node.

- **Send_SampleValue_SerialInterface()**:
  - This method sends sample values of the sensor nodes component to the redundant data source computation component through the serial interface.

- **Receive_ID_RDS()**:
  - This method receives the ID numbers of the sensors in the given environment along with the ID number of their redundant data sources.

- **Send_ID_RDS()**:
  - This method sends the ID numbers of the sensors in the given environment along with the ID number of their redundant data sources to sensor nodes component.

Sensor sensing nodes component
As shown in Figure 3.10, the SensorSensing object has two attributes, namely, NodeID and IntervalTime. This object is implemented on the MyriaModem. This MyriaModem with the SensorSensing object contains one sensor or more than one. As shown in Figure 3.10, the Sensor object represents this sensor. It has three attributes. The Type attribute represents the type of the sensor (e.g. temperature sensor), the SensorValue gives the value of the sensor and the SamplesTime attribute gives the sample time of the sensor.

The SensorSensing object has the following methods:

- **InitializeSensorSensingNode()**:
  - This method initializes MyriaModem (e.g. MyriaNed Protocol, serial interface).

- **Receive_ID_RDS()**:
  - This method receives ID number of the redundant data source.

- **Receive_SampleValue_Neighbors()**:
  - This method receives the data from the neighboring sensor node only if the ID number of this neighboring sensor node matches with the ID number of the redundant data source.

- **SensorSensing()**:
- This method reads the value of the sensor.

- **Send\_Sample\_Value()**:
  
  - After reading the value of the sensor, the `Send\_Sample\_Value()` sends this value to the RxTx node component.

### 3.2.4 Data format for redundant data sources

MyriaModem uses MyriaNed protocol to send messages to other MyriaModem in the given SE. Figure 3.11 shows the structure of the message of the MyriaModem which is sent or received. The application payload is the message which we want to send. Table 3.2 shows the data format of this message. This message contains the sensor ID number (e.g. 12FP) and its redundant data source ID number (e.g. 11FF). When the sensor sensing node receives this message, it checks if this was intended for its own use by checking the node ID number with data source ID. Otherwise, it ignores it.

<table>
<thead>
<tr>
<th>Data source ID (4 Bytes)</th>
<th>Redundant data source ID (4 Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12FP</td>
<td>11FF</td>
</tr>
</tbody>
</table>

Table 3.2: Data format for redundancy

---

http://myriamodem.vanmierlo.com/
Figure 3.11: Packet definition of the MyriaNed protocol
Chapter 4

The Synthetic Data Model

4.1 Synthetic data generation

In this chapter, we generate synthetic data sets which are used to evaluate the performance of the similarity and correlation measures from Chapter 2. The synthetic data sets should capture the properties of the data in the real world. In the real world, data is distorted (e.g. delay in the communication channel or noise in the data). The following types of distortion may occur:

- uniform amplitude shifting,
- out of phase,
- uniform amplitude scaling,
- noise,
- the combination of the above.

Therefore, the similarity and correlation measures need to take the above distortion into account. Eleven synthetic data sets are generated, where each data set contains the following signals with various frequencies ($f$):

- Sine wave signal: $SineSignal(t) = \sin(f \times 2 \times \pi \times t)$,
- Square wave signal: $SquareSignal(t) = \text{square}(f \times 2 \times \pi \times t)$,
- sawtooth wave signal: $SawtoothSignal(t) = \text{sawtooth}(f \times 2 \times \pi \times t)$,
- The distortion of these above signals:
  In the next section, the types of distortion are explained in more details.

Table 4.1 and Table 4.2 summarize the parameters settings for these data sets.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform amplitude shifting</td>
<td>$C_{U\text{AmpShifting}}$</td>
<td>0, 1.5, 2 and 3.5</td>
</tr>
<tr>
<td>Uniform amplitude scaling</td>
<td>$C_{U\text{AmpScaling}}$</td>
<td>0, 1.5, 2 and 3.5</td>
</tr>
<tr>
<td>Out of phase</td>
<td>$C_{\text{OutPhase}}$</td>
<td>0, 2, 5 and 10</td>
</tr>
<tr>
<td>Signal to noise ration</td>
<td>snr</td>
<td>10</td>
</tr>
<tr>
<td>Length of signal</td>
<td>M</td>
<td>100</td>
</tr>
<tr>
<td>The size of the data set</td>
<td>N</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4.1: Parameters setting for synthetic data sets
### 4.2 Distortion generation

In this section, the meaning of every distortion is explained and generated. The distortions are generated using MATLAB.

#### 4.2.1 Uniform amplitude shifting

Uniform amplitude shifting distortion appears when two time series are uniformly shifted from each other in the amplitude level as shown in Figure 4.1. Uniform means that the shifting is the same for every sample of the time series. In other words, The two time series have the same shape while the amplitude level of every time series is different. To generate this distortion for a sine wave signal, a square wave signal and a sawtooth wave signal, a distortion constant, $C_{UAmpShifting}$, is added to the original signals. The following formulas generate the uniform amplitude shifting distortion for the following signals:

- \[ \text{SineSignal}(t) = \sin(f \times 2 \pi \times t) + C_{UAmpShifting} \]
- \[ \text{SquareSignal}(t) = \text{square}(f \times 2 \pi \times t) + C_{UAmpShifting} \]
- \[ \text{SawtoothSignal}(t) = \text{sawtooth}(f \times 2 \pi \times t) + C_{UAmpShifting} \]

When the $C_{UAmpShifting}$ increases, the distortion increases. Figure 4.1 shows a uniform amplitude shifting distortion for the sawtooth wave signal with $C_{UAmpShifting} = 1.5$. 

### Table 4.2: Synthetic data set

<table>
<thead>
<tr>
<th>Time series</th>
<th>N</th>
<th>M</th>
<th>5</th>
<th>10</th>
<th>9</th>
<th>7</th>
<th>15</th>
<th>30</th>
<th>33</th>
<th>40</th>
<th>47</th>
<th>50</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sawtooth signal original</td>
<td>79</td>
<td>100</td>
<td>5</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>15</td>
<td>30</td>
<td>33</td>
<td>40</td>
<td>47</td>
<td>50</td>
<td>13</td>
</tr>
<tr>
<td>Sawtooth signal (UAmpShifting)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sawtooth signal (Out phase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sawtooth signal (noise)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sawtooth signal (combination)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Square signal original</td>
<td>79</td>
<td>100</td>
<td>8</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>16</td>
<td>20</td>
<td>22</td>
<td>44</td>
<td>50</td>
<td>55</td>
<td>11</td>
</tr>
<tr>
<td>Square signal (UAmpShifting)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Square signal (Out phase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Square signal (noise)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Square signal (combination)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine signal original</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine signal (UAmpShifting)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine signal (Out phase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine signal (noise)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine signal (combination)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Finding Redundant Data Sources in Smart Environments
4.2.2 Out of phase

In Figure 4.2 an out of phase distortion is shown. Two time series are out of phase, if they are shifted from each other in the time axis while they have the same shape. To generate this kind of distortion, we add a distortion constant, $C_{OutPhase}$, to the original signal. The following formulas generate the out of phase distortion:

- $\text{SineSignal}(t) = \sin(f \times 2 \times \pi \times t + C_{OutPhase})$
- $\text{SquareSignal}(t) = \text{square}(f \times 2 \times \pi \times t + C_{OutPhase})$
- $\text{SquareSignal}(t) = \text{sawtooth}(f \times 2 \times \pi \times t + C_{OutPhase})$

When the $C_{OutPhase}$ increases, the distortion increases. Figure 4.2 shows an out of phase distortion for sawtooth wave signal with $C_{OutPhase} = 2$.

4.2.3 Uniform amplitude scaling

Uniform amplitude scaling distortion appeared when two time series are uniformly scaled from each other in the amplitude level as shown in Figure 4.3. They have the same shape while the
amplitude level of every time series is different. To generate this distortion for a sine wave signal, a square wave signal and a sawtooth wave signal, a distortion constant, $C_{\text{UAmpScaling}}$, has been added. The following formulas generate the uniform amplitude scaling distortion:

- **SineSignal**$(t) = \sin(f \times 2 \times \pi \times t) \times C_{\text{UAmpScaling}}$
- **SquareSignal**$(t) = \text{square}(f \times 2 \times \pi \times t) \times C_{\text{UAmpScaling}}$
- **SawtoothSignal**$(t) = \text{sawtooth}(f \times 2 \times \pi \times t) \times C_{\text{UAmpScaling}}$

When the $C_{\text{UAmpScaling}}$ increases, the distortion increases. Figure 4.3 shows an uniform amplitude scaling for sawtooth wave signal with $C_{\text{UAmpScaling}} = 1.5$.

4.2.4 Noise

As shown in Figure 4.4, the noise distortion means that two time series have the same shape while one is distorted due to the noise. To generate this distortion for a sine wave signal, a square wave signal and a sawtooth wave signal, the MATLAB function `awgn` is used. The syntax of `awgn` is the following:

- **Sine wave signal**
  $\text{SineSignal}(t) = \text{awgn}(\sin(f \times 2 \times \pi \times t), 10, \text{‘measured’})$
- **Square wave signal**
  $\text{SquareSignal}(t) = \text{awgn}(\text{square}(f \times 2 \times \pi \times t), 10, \text{‘measured’})$
- **Sawtooth wave signal**
  $\text{SawtoothSignal}(t) = \text{awgn}(\text{sawtooth}(f \times 2 \times \pi \times t), 10, \text{‘measured’})$
4.2.5 The combination of the distortions

Figure 4.5 shows the combination of the distortions. To generate this type of distortion, we add a distortion constant, $C_{UAmpShifting}$, $C_{OutPhase}$, $C_{UAmpScaling}$ and the noise to the original signal. To generate the combination of the distortion for a sine wave signal, a square wave signal and a sawtooth wave signal, distortion constants, $C_{UAmpShifting}$, $C_{OutPhase}$, $C_{UAmpScaling}$ and the noise have been added. The following formulas generate the combination of distortions:

- $\text{SineSignal}(t) = \text{awgn}((\sin(f \times 2 \times \pi \times t + C_{OutPhase}) \times C_{UAmpScaling} + C_{UAmpShifting}), 10, \text{'measured'})$
- $\text{SquareSignal}(t) = \text{awgn}((\sin(f \times 2 \times \pi \times t + C_{OutPhase}) \times C_{UAmpScaling} + C_{UAmpShifting}), 10, \text{'measured'})$
- $\text{SawtoothSignal}(t) = \text{awgn}((\sin(f \times 2 \times \pi \times t + C_{OutPhase}) \times C_{UAmpScaling} + C_{UAmpShifting}), 10, \text{'measured'})$
Chapter 5

Performance Evaluation using Synthetic Data

5.1 Performance evaluation

In this chapter, the performance of the similarity and correlation measures of Chapter 2 is evaluated and compared to each other using the synthetic data model of the previous chapter. The performance is evaluated in the terms of accuracy and computation time. The accuracy determines which similarity and correlation measure gives the best results even if the time series have the following distortions:

- uniform amplitude shifting distortion,
- out of phase distortion,
- uniform amplitude scaling distortion,
- noise distortion,
- the combination of the above.

To measure the accuracy, One Nearest Neighbour (1-NN) classifier is used. Further, the similarity and the correlation measures are implemented in the MATLAB. To measure the computation time, we use the `tic` and `toc` functions of the MATLAB:

```matlab
% MATLAB code to compute the computation time of similarity or correlation measures
tic;
.
.
similarity_measures(); % or correlation_measure()
.
computation_time = toc;
```

5.1.1 One Nearest Neighbour (1-NN) classifier

The 1-NN classifier is mostly used in the literature [6, 8, 12, 26] to evaluate the performance of the similarity and the correlation measures. It is used because of the following:

- accuracy of the 1-NN depends on the underlying similarity and correlation measures,
- easy to implement.

Before using 1-NN, the time series are labeled as shown in Table 5.1. Similar time series are getting the same label. Further, the data set is divided into two parts as shown in Table 5.1. One part contains 3 original signals (a sine wave signal, a square wave signal and a sawtooth wave signal) and the other part contains 237 distorted signals of the original signals. Every original signal has 79 similar signals, but these signals are distorted. 1-NN determines which similarity and correlation measure is able to detect the most signals (79 distorted signals) for each original signal.
1-NN works as follows:
As shown in Algorithm 1, 1-NN takes three parameters as inputs, an original data set, a distorted data set and a similarity measure or a correlation measure. The original data set contains the original signals and the distorted data set contains the distorted signals. From line 3 to line 8, the 1-NN algorithm takes each time a \( T^{DS,i} \) from the original data set and finds a \( T^{DS,j} \) in the distorted data set that minimizes the given similarity measure \( (T^{DS,i}, T^{DS,j}) \) or, in the case of the correlation measure, maximizes the given correlation measure \( (T^{DS,i}, T^{DS,j}) \). After finding the \( T^{DS,j} \), the label of this \( T^{DS,j} \) is checked. If the label matches with the label of \( T^{DS,i} \), the \( nr\_labels\_matches \) variable is increasing with one. As an output, 1-NN delivers the accuracy of the similarity or the correlation measure. The accuracy is computed on line 9.

Algorithm 1 1NN for similarity measures

1: function 1NN(Distortion\_DataSet, Original\_DataSet, similarity\_measure)  
2: \( nr\_labels\_matches \leftarrow 0 \)  
3: for each \( T^{DS,i} \) from Original\_DataSet do  
4: \( T^{DS,j} \) = Find \( T^{DS,j} \) from Distortion\_DataSet that minimizes similarity\_measure \( (T^{DS,i}, T^{DS,j}) \)  
5: if the label of \( T^{DS,i} \) and \( T^{DS,j} \) are the same then  
6: \( nr\_labels\_matches \leftarrow nr\_labels\_matches + 1 \)  
7: end if  
8: end for  
9: return Accuracy \( \leftarrow nr\_labels\_matches / |\text{Distortion\_DataSet}| \)

Table 5.1: Time series are labeled and synthetic data set is divided into two parts.

<table>
<thead>
<tr>
<th>Label</th>
<th>Original dataset</th>
<th>Size of dataset</th>
<th>Length of time series</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sawtooth signal original</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>Sawtooth signal (LampShifting)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>Sawtooth signal (LampSizing)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>Sawtooth signal (Out phase)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>Sawtooth signal (Combination)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Square signal original</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Square signal (LampShifting)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Square signal (LampSizing)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Square signal (Out phase)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Square signal (Combination)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>Sine signal original</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>Sine signal (LampShifting)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>Sine signal (LampSizing)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>Sine signal (Out phase)</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>Sine signal (Combination)</td>
<td>79</td>
<td>100</td>
</tr>
</tbody>
</table>

5.2 Similarity measures
In this section, the accuracy of the similarity measures is evaluated and compared to each other using eleven synthetic data sets. The following similarity measures are evaluated:

- ED,
CHAPTER 5. PERFORMANCE EVALUATION USING SYNTHETIC DATA

- CIDED,
- DTW,
- CIDDTW,
- SBD,
- ExtJaccard.

Further, the computation time of each similarity measure is measured and compared to each other.

5.2.1 Results of the experiment

Table 5.2 and Table 5.3 show the accuracy and the computation time of similarity measures for every data set and the average accuracy of similarity measures, respectively.

<table>
<thead>
<tr>
<th>Synthetic data set</th>
<th>ED</th>
<th>CIDED</th>
<th>DTW</th>
<th>CIDDTW</th>
<th>SBD</th>
<th>ExtJaccard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set1</td>
<td>0.359</td>
<td>0.397</td>
<td>0.346</td>
<td>0.388</td>
<td>1.000</td>
<td>0.426</td>
</tr>
<tr>
<td>Data Set2</td>
<td>0.350</td>
<td>0.354</td>
<td>0.329</td>
<td>0.342</td>
<td>0.633</td>
<td>0.451</td>
</tr>
<tr>
<td>Data Set3</td>
<td>0.346</td>
<td>0.384</td>
<td>0.329</td>
<td>0.354</td>
<td>0.992</td>
<td>0.376</td>
</tr>
<tr>
<td>Data Set4</td>
<td>0.338</td>
<td>0.350</td>
<td>0.333</td>
<td>0.338</td>
<td>1.000</td>
<td>0.392</td>
</tr>
<tr>
<td>Data Set5</td>
<td>0.350</td>
<td>0.414</td>
<td>0.342</td>
<td>0.422</td>
<td>1.000</td>
<td>0.397</td>
</tr>
<tr>
<td>Data Set6</td>
<td>0.346</td>
<td>0.350</td>
<td>0.333</td>
<td>0.342</td>
<td>0.996</td>
<td>0.392</td>
</tr>
<tr>
<td>Data Set7</td>
<td>0.350</td>
<td>0.354</td>
<td>0.329</td>
<td>0.346</td>
<td>1.000</td>
<td>0.409</td>
</tr>
<tr>
<td>Data Set8</td>
<td>0.346</td>
<td>0.342</td>
<td>0.342</td>
<td>0.338</td>
<td>0.996</td>
<td>0.354</td>
</tr>
<tr>
<td>Data Set9</td>
<td>0.346</td>
<td>0.405</td>
<td>0.354</td>
<td>0.405</td>
<td>0.983</td>
<td>0.371</td>
</tr>
<tr>
<td>Data Set10</td>
<td>0.359</td>
<td>0.376</td>
<td>0.350</td>
<td>0.354</td>
<td>1.000</td>
<td>0.397</td>
</tr>
<tr>
<td>Data Set11</td>
<td>0.350</td>
<td>0.392</td>
<td>0.342</td>
<td>0.388</td>
<td>1.000</td>
<td>0.409</td>
</tr>
<tr>
<td>Average accuracy</td>
<td>0.349</td>
<td>0.374</td>
<td>0.339</td>
<td>0.365</td>
<td>0.964</td>
<td>0.398</td>
</tr>
<tr>
<td>Computation time</td>
<td>8.649</td>
<td>15.094</td>
<td>15377</td>
<td>12439</td>
<td>204.09</td>
<td>37.745</td>
</tr>
</tbody>
</table>

Table 5.2: The accuracy and the computation time for the similarity measures

![average accuracy](image)

Figure 5.1: The average accuracy for the similarity measures

As shown in Table 5.2 and in Figure 5.1, SBD gives the best accuracy. The rest of the similarity measures give a bad accuracy. However, For data set 2, SBD gives a bad accuracy, this is because the time series in this data set have the same frequencies as shown in Figure 5.2 and the comparison of the SBD is based on the shape of the time series. Therefore, SBD may not be able to make difference between these time series. Since the comparison of the SBD based on the shape
of the time series, the distortions have less effect on SBD. The comparisons of other similarity measures are based on the difference between the time series. Therefore, the distortions (uniform amplitude shifting, out of phase, uniform amplitude scaling, noise and combinations of these distortions) have effects on the accuracies of these similarity measures. Regarding the computation time, ED has the best computation time, while DTW and CIDDTW have the worst computation time. The time complexity of ED is $O(M)$ and DTW is $O(MT_1 \times MT_2)$.

**Normalization:**
It is possible to eliminate some of these distortions in the time series, by using Z-normalization. As shown in Figure 5.3 and in Figure 5.4, the Z-normalization is able to eliminate the uniform amplitude shifting and the uniform amplitude scaling distortions, respectively. The Z-normalization transforms the time series into new time series with a mean of 0 and a standard deviation of 1.

**Z-normalisation is defined as follow:**

\[
\hat{t}_{i}^{DS,1} = \frac{t_{i}^{DS,1} - T^{DS,1}}{\sigma_{T^{DS,1}}}
\]

where $\hat{t}_{i}^{DS,1}$ is a Z-normalized sample and
CHAPTER 5. PERFORMANCE EVALUATION USING SYNTHETIC DATA

Figure 5.4: Eliminate the uniform amplitude scaling distortion

\[ T^{DS,1} = \frac{\sum_{i=1}^{M} t_{i}^{DS,1}}{M}, \text{ and } \sigma_{T^{DS,1}} = \sqrt{\frac{\sum_{i=1}^{M} (t_{i}^{DS,1} - T^{DS,1})^2}{M}} \]

the mean and the standard deviation, respectively and M is the length of the time series.

Table 5.3 and Figure 5.5 show the results after using the Z-normalization. The accuracies are improved for all similarity measures. However, on average, SBD gives the best accuracy.

<table>
<thead>
<tr>
<th>Synthetic data set</th>
<th>ED</th>
<th>CITED</th>
<th>DTW</th>
<th>CIDDTW</th>
<th>SBD</th>
<th>Ext.Jaccard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set1</td>
<td>0.684</td>
<td>0.641</td>
<td>0.954</td>
<td>0.827</td>
<td>1.000</td>
<td>0.684</td>
</tr>
<tr>
<td>Data Set2</td>
<td>0.489</td>
<td>0.519</td>
<td>0.675</td>
<td>0.662</td>
<td>0.620</td>
<td>0.489</td>
</tr>
<tr>
<td>Data Set3</td>
<td>0.595</td>
<td>0.650</td>
<td>0.899</td>
<td>0.789</td>
<td>1.000</td>
<td>0.595</td>
</tr>
<tr>
<td>Data Set4</td>
<td>0.654</td>
<td>0.532</td>
<td>0.916</td>
<td>0.764</td>
<td>1.000</td>
<td>0.654</td>
</tr>
<tr>
<td>Data Set5</td>
<td>0.662</td>
<td>0.633</td>
<td>0.857</td>
<td>0.827</td>
<td>1.000</td>
<td>0.662</td>
</tr>
<tr>
<td>Data Set6</td>
<td>0.688</td>
<td>0.781</td>
<td>0.688</td>
<td>0.726</td>
<td>1.000</td>
<td>0.688</td>
</tr>
<tr>
<td>Data Set7</td>
<td>0.705</td>
<td>0.700</td>
<td>0.895</td>
<td>0.857</td>
<td>1.000</td>
<td>0.705</td>
</tr>
<tr>
<td>Data Set8</td>
<td>0.473</td>
<td>0.435</td>
<td>0.688</td>
<td>0.629</td>
<td>0.996</td>
<td>0.473</td>
</tr>
<tr>
<td>Data Set9</td>
<td>0.578</td>
<td>0.624</td>
<td>0.802</td>
<td>0.755</td>
<td>0.992</td>
<td>0.578</td>
</tr>
<tr>
<td>Data Set10</td>
<td>0.662</td>
<td>0.709</td>
<td>0.696</td>
<td>0.679</td>
<td>1.000</td>
<td>0.662</td>
</tr>
<tr>
<td>Data Set11</td>
<td>0.679</td>
<td>0.637</td>
<td>0.958</td>
<td>0.814</td>
<td>1.000</td>
<td>0.679</td>
</tr>
<tr>
<td>Average accuracy</td>
<td>0.624</td>
<td>0.624</td>
<td>0.820</td>
<td>0.757</td>
<td>0.964</td>
<td>0.624</td>
</tr>
</tbody>
</table>

Table 5.3: The accuracy and the computation time for the similarity measures (Z-normalization)

5.2.2 Statistical analysis

In order to validate the experimental results, we need to analyze the results statistically. Therefore, we used Friedman test [27] as a statistical test. Friedman test compares the performance of the similarity measures on multiple data sets and determines which similarity or correlation measure is statistically significant.

The Friedman test works as follow:

1. It ranks the similarity measures for each data set. The similarity measure with the best accuracy gets the highest rank.
2. If similarity measures have the same accuracy, the average of the ranks is taken. The similarity measures get the same average rank. For example, ED and ExtJaccard have the same accuracy:

<table>
<thead>
<tr>
<th>Synthetic data set</th>
<th>ED</th>
<th>CIDED</th>
<th>DTW</th>
<th>CIDDTW</th>
<th>SBD</th>
<th>ExtJaccard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set1</td>
<td>0.684(4.5)</td>
<td>0.641(6.0)</td>
<td>0.954(2.0)</td>
<td>0.827(3.0)</td>
<td>1.000(1.0)</td>
<td>0.684(4.5)</td>
</tr>
</tbody>
</table>

Because rank 1, rank 2 and rank 3 are used for SBD, DTW and CIDDTW, respectively. The similarity measure ED and similarity ExtJaccard measure get rank 4 and 5, respectively or 5 and 4, respectively. But because they have the same accuracy they get \( \frac{4+5}{2} = 4.5 \) as rank.

3. Next, we make the null hypothesis, which states that all similarity measures are equivalent.

4. The null hypothesis will be rejected if

\[
Friedman_{statistic} \geq F_{distribution}, \text{ where:}
\]

\[
Friedman_{statistic} = \frac{(N-1)\sum_{j}^{k} \frac{R_j^2}{N} - \frac{k(k+1)^2}{4}}{N(k-1) - \frac{k(k+1)^2}{4}}
\]

\( k \) is the number of the similarity measures, \( N \) is the number of the data set and \( R_j = \frac{1}{N} \sum_{i}^{N} r_{ij} \) is the average ranks of a similarity measure \( j \), \( r_{ij} \) is the rank of the \( j \)-th of \( k \) similarity measures on the \( i \)-th of \( N \) data sets. \( F_{distribution} \) is a critical value. This value can be calculated using three parameters:

(a) Degree of freedom 1: (k-1),
(b) Degree of freedom 2: (k-1)*(N-1),
(c) Probability level: 0.05.

5. If the null hypothesis is rejected, we proceed with the Nemenyi test. The Nemenyi test compares the similarity measures to each other. The accuracy of the similarity measures is significantly different if their average rank differ by at least the critical difference:

\[
CD = q_{0.05} \sqrt{\frac{k(k+1)}{6N}}
\]

where \( q_{0.05} \) is a critical value for the two-tailed Nemenyi test. I used a Perl script which is provided in [12] to analyse the results statistically.
**CHAPTER 5. PERFORMANCE EVALUATION USING SYNTHETIC DATA**

**Friedman test:**
To evaluate the significance of their differences in accuracy, we rank the accuracy using Friedman test. Table 5.4 shows the ranks, the ranks are between the brackets. On average SBD has the highest rank, it means that SBD gives better accuracy in the majority of the data sets.

<table>
<thead>
<tr>
<th>Synthetic data set</th>
<th>ED</th>
<th>CIDED</th>
<th>DTW</th>
<th>CIDDTW</th>
<th>SBD</th>
<th>ExtJaccard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set1</td>
<td>0.684(4.5)</td>
<td>0.641(6.0)</td>
<td>0.954(2.0)</td>
<td>0.827(3.0)</td>
<td>1.000(1.0)</td>
<td>0.684(4.5)</td>
</tr>
<tr>
<td>Data Set2</td>
<td>0.489(5.5)</td>
<td>0.519(4.0)</td>
<td>0.675(1.0)</td>
<td>0.662(2.0)</td>
<td>0.620(3.0)</td>
<td>0.489(5.5)</td>
</tr>
<tr>
<td>Data Set3</td>
<td>0.595(5.5)</td>
<td>0.650(4.0)</td>
<td>0.899(2.0)</td>
<td>0.789(3.0)</td>
<td>1.000(1.0)</td>
<td>0.595(5.5)</td>
</tr>
<tr>
<td>Data Set4</td>
<td>0.654(4.5)</td>
<td>0.532(6.0)</td>
<td>0.916(2.0)</td>
<td>0.764(3.0)</td>
<td>1.000(1.0)</td>
<td>0.654(4.5)</td>
</tr>
<tr>
<td>Data Set5</td>
<td>0.662(4.5)</td>
<td>0.633(6.0)</td>
<td>0.857(2.0)</td>
<td>0.827(3.0)</td>
<td>1.000(1.0)</td>
<td>0.662(4.5)</td>
</tr>
<tr>
<td>Data Set6</td>
<td>0.688(5.0)</td>
<td>0.781(2.0)</td>
<td>0.688(5.0)</td>
<td>0.726(3.0)</td>
<td>1.000(1.0)</td>
<td>0.688(5.0)</td>
</tr>
<tr>
<td>Data Set7</td>
<td>0.705(4.5)</td>
<td>0.700(6.0)</td>
<td>0.895(2.0)</td>
<td>0.857(3.0)</td>
<td>1.000(1.0)</td>
<td>0.705(4.5)</td>
</tr>
<tr>
<td>Data Set8</td>
<td>0.473(4.5)</td>
<td>0.435(6.0)</td>
<td>0.688(2.0)</td>
<td>0.629(3.0)</td>
<td>0.996(1.0)</td>
<td>0.473(4.5)</td>
</tr>
<tr>
<td>Data Set9</td>
<td>0.578(5.5)</td>
<td>0.624(4.0)</td>
<td>0.802(2.0)</td>
<td>0.755(3.0)</td>
<td>0.992(1.0)</td>
<td>0.578(5.5)</td>
</tr>
<tr>
<td>Data Set10</td>
<td>0.662(5.5)</td>
<td>0.709(2.0)</td>
<td>0.696(3.0)</td>
<td>0.679(4.0)</td>
<td>1.000(1.0)</td>
<td>0.662(5.5)</td>
</tr>
<tr>
<td>Data Set11</td>
<td>0.679(4.5)</td>
<td>0.637(6.0)</td>
<td>0.958(2.0)</td>
<td>0.814(3.0)</td>
<td>1.000(1.0)</td>
<td>0.679(4.5)</td>
</tr>
<tr>
<td>Average ranks</td>
<td>4.909</td>
<td>4.727</td>
<td>2.273</td>
<td>3</td>
<td>1.182</td>
<td>4.909</td>
</tr>
</tbody>
</table>

Table 5.4: Friedman test results for the similarity measures

**The null hypothesis:**
\[
Friedman_{statistic} = 25.77 \quad \text{and} \quad F_{distribution} \quad \text{with 5 and 50 degrees of freedom and at the probability level of 0.05 is 11.07. Since} \quad Friedman_{statistic} > F_{distribution}, \quad \text{The null hypothesis is rejected. It means that the similarity measures don’t have the same behaviour.}
\]

**Nemenyi test:**
When the null hypothesis is rejected, we proceed with Nemenyi test to evaluate the significance of the differences in the ranks. Table 5.5 depicts the result of the Nemenyi test. SBD and DTW are significantly better than ED, CIDED and ExtJaccard. There is no significant difference between CIDDTW, CIDED, ED and Extend Jaccard. There is also no significant difference between SBD, DTW and CIDDTW.

<table>
<thead>
<tr>
<th></th>
<th>SBD</th>
<th>DTW</th>
<th>CIDDTW</th>
<th>CIDED</th>
<th>ExtJaccard</th>
<th>ED</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBD</td>
<td></td>
<td>No evidence</td>
<td>No evidence</td>
<td>Better</td>
<td>Better</td>
<td></td>
</tr>
<tr>
<td>DTW</td>
<td>No evidence</td>
<td></td>
<td>No evidence</td>
<td>Better</td>
<td>Better</td>
<td></td>
</tr>
<tr>
<td>CIDDTW</td>
<td>No evidence</td>
<td>No evidence</td>
<td>No evidence</td>
<td>No evidence</td>
<td>No evidence</td>
<td></td>
</tr>
<tr>
<td>CIDED</td>
<td>Worse</td>
<td>Worse</td>
<td>No evidence</td>
<td>No evidence</td>
<td>No evidence</td>
<td></td>
</tr>
<tr>
<td>ExtJaccard</td>
<td>Worse</td>
<td>Worse</td>
<td>No evidence</td>
<td>No evidence</td>
<td>No evidence</td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>Worse</td>
<td>Worse</td>
<td>No evidence</td>
<td>No evidence</td>
<td>No evidence</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Nemenyi test for similarity measures

**5.3 Correlation measures**
In this section, we will evaluate and compare the accuracy of PCor and DCor correlation measures to each other using eleven synthetic data sets. The output range of PCor is between -1 and 1 and the output range of DCor is between 0 and 1. PCor gives information about the direction of the correlation. In order to compare these correlation measures to each other, we make the
output range of PCor also between 0 and 1 by taken the absolute value of the output $\text{abs}(\text{PCor})$. Therefore, the output range of $\text{abs}(\text{PCor})$ is between 0 and 1. At this moment, we are interested in the performance of the correlation measures rather than in the direction of the correlation.

5.3.1 Results of the experiment

As shown in Table 5.6 and Figure 5.6, DCor gives the best accuracy. However, for data set 2 both correlation measures give bad accuracy. This is because the time series as shown in Figure 5.2 (SineSignal, SquareSignal and SawtoothSignal) in data set 2 have the same frequency. The time series have the same behavior. The results in Table 5.6 uses no normalization. As regards the computation time, PCor gives the best computation time.

Normalization:

Table 5.7 and Figure 5.7 show the results using the Z-normalization. Using the Z-normalization has few effects on the accuracy. The distortion such as uniform amplitude shifting and the uniform amplitude scaling has no effect on the correlation measures. DCor gives the best results.

<table>
<thead>
<tr>
<th>Synthetic data set</th>
<th>PCor</th>
<th>DCor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set1</td>
<td>0.827</td>
<td>0.911</td>
</tr>
<tr>
<td>Data Set2</td>
<td>0.300</td>
<td>0.342</td>
</tr>
<tr>
<td>Data Set3</td>
<td>0.772</td>
<td>0.895</td>
</tr>
<tr>
<td>Data Set4</td>
<td>0.895</td>
<td>0.902</td>
</tr>
<tr>
<td>Data Set5</td>
<td>0.970</td>
<td>0.992</td>
</tr>
<tr>
<td>Data Set6</td>
<td>0.662</td>
<td>0.772</td>
</tr>
<tr>
<td>Data Set7</td>
<td>0.848</td>
<td>0.907</td>
</tr>
<tr>
<td>Data Set8</td>
<td>0.911</td>
<td>0.903</td>
</tr>
<tr>
<td>Data Set9</td>
<td>0.895</td>
<td>0.987</td>
</tr>
<tr>
<td>Data Set10</td>
<td>0.992</td>
<td>0.996</td>
</tr>
<tr>
<td>Data Set11</td>
<td>0.823</td>
<td>0.916</td>
</tr>
<tr>
<td>Average accuracy</td>
<td>0.809</td>
<td>0.871</td>
</tr>
<tr>
<td>Computation time</td>
<td>16.862</td>
<td>5082.8</td>
</tr>
</tbody>
</table>

Table 5.6: The accuracy and the computation time for the correlation measures

Figure 5.6: The average accuracy for the correlation measures
Table 5.7: The accuracy and the computation time for the correlation measures (Z-normalization).

<table>
<thead>
<tr>
<th>Synthetic data set</th>
<th>PCor</th>
<th>DCor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set1</td>
<td>0.806</td>
<td>0.895</td>
</tr>
<tr>
<td>Data Set2</td>
<td>0.300</td>
<td>0.346</td>
</tr>
<tr>
<td>Data Set3</td>
<td>0.781</td>
<td>0.903</td>
</tr>
<tr>
<td>Data Set4</td>
<td>0.869</td>
<td>0.941</td>
</tr>
<tr>
<td>Data Set5</td>
<td>0.970</td>
<td>0.996</td>
</tr>
<tr>
<td>Data Set6</td>
<td>0.684</td>
<td>0.789</td>
</tr>
<tr>
<td>Data Set7</td>
<td>0.819</td>
<td>0.899</td>
</tr>
<tr>
<td>Data Set8</td>
<td>0.907</td>
<td>0.920</td>
</tr>
<tr>
<td>Data Set9</td>
<td>0.886</td>
<td>0.979</td>
</tr>
<tr>
<td>Data Set10</td>
<td>0.987</td>
<td>0.992</td>
</tr>
<tr>
<td>Data Set11</td>
<td>0.827</td>
<td>0.911</td>
</tr>
<tr>
<td>Average accuracy</td>
<td>0.863</td>
<td>0.870</td>
</tr>
</tbody>
</table>

Figure 5.7: The average accuracy for the correlation measures (Z-normalization).

5.4 Conclusion

Similarity measures:
From the experiment, we conclude that SBD gives the best accuracy. However, there is no significant difference between SBD, DTW and CIDDTW. Further, SBD and DTW outperform ED, CIDED and ExtJaccard with statistical significance. Using normalization, we can eliminate the uniform amplitude shifting and the uniform amplitude scaling distortions.

Correlation measures:
From the experiment, DCor gives better accuracy than PCor. However, PCor runs faster than DCor.

On the need for a threshold:
Given a set of data sources $DS_{Set} = \{DS_1, ..., DS_N\}$ and a failed sensor $DS_1$, $State_{DS} (DS_1) = F_{DS}$, in a given environment. With the SBD, DTW and CIDDTW, we can find the most similar data source $DS_i$ among the set of data sources, $DS_{Set} = \{DS_2, ..., DS_N\}$ to $DS_1$ that minimizes SBD, DTW and CIDDTW($DS_1, DS_i$). However, this does not imply that $DS_i$ is similar to $DS_1$. We need a $threshold_{similarity}$. This $threshold_{similarity}$ determines when two time series are really similar. Unfortunately, for similarity measures, it is hard to determine the $threshold_{similarity}$. Since it depends on the length and the shape of the data.
In contrast, for correlation measures, it is possible to determine the threshold of correlation. The output of PCor is between -1 and 1, we can take for example 0.9 as value for the threshold correlation. This means that, $\text{PCor}(T^{DS,1}, T^{DS,2}) \geq 0.9$, $T^{DS,1}$ and $T^{DS,2}$ highly correlated and at least 90% similar.

Based on the above conclusions, we propose the following:
To find redundant data sources that are similar, we use a similarity measure in combination with a PCor and an appropriate threshold correlation. We rank data sources from most similar to least similar using similarity measures then we apply PCor with an appropriate threshold correlation to filter data sources that are not similar. We use PCor instead of DCor because PCor gives information about the direction of the correlation. The direction of the correlation is important. Positive direction gives more information about the similarity between two time series.
Chapter 6

Finding Redundant Data Sources

6.1 Steps to find redundant data sources

In this chapter, the proposed approach for finding redundant data sources is explained. Further, two experiments are performed on the real data. One experiment uses similarity measures in combination with PCor measure to find redundant data sources that are similar. The other experiment uses correlation measures to find redundant sources that are correlated. These experiments determine the appropriate threshold correlation.

6.1.1 Similarity measures

In this section, we explain the proposed approach for finding redundant data sources that are similar using a similarity measure in combination with PCor and an appropriate threshold correlation. This approach consists of three steps:

1. data sources normalization step,
2. data sources ranking step,
3. data sources filter step.

6.1.1.1 Data sources normalization

Mostly, an SE is equipped with various sensors. In order to compare two time series from different sensor types, we need to eliminate the measurement units (e.g. temperature sensors (°C) and humidity sensors (%)). The Z-normalization, beside eliminating distortions, can also eliminate the measurement units as shown in Figure 6.1.

6.1.1.2 Data sources ranking

The data sources from a given environment are compared to each other after eliminating the measurement units. In step 2, as shown in Table 6.1, a similarity measure is applied to compare and rank the data sources from most similar to least similar. In other words, for every data source $DS_i$ in a given environment, we find the data sources that are most similar and least similar to $DS_i$. As shown in Table 6.1, for data sources $DS_1$, $DS_2$ is the most similar to $DS_1$ and $DS_{20}$ is the least similar to $DS_1$.

<table>
<thead>
<tr>
<th>Data sources</th>
<th>Most similar to least similar</th>
</tr>
</thead>
<tbody>
<tr>
<td>$DS_1$</td>
<td>$DS_2$</td>
</tr>
<tr>
<td>$DS_2$</td>
<td>$DS_1$</td>
</tr>
<tr>
<td>$DS_3$</td>
<td>$DS_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$DS_{20}$</td>
<td>$DS_{17}$</td>
</tr>
</tbody>
</table>

Table 6.1: Most similar to least similar
6.1.1.3 Data sources filter using PCor

After applying step 2, the data sources in the given environment are ranked from most similar to least similar for every sensor in the given environment as shown in Table 6.1. In step 3, correlation measure PCor is applied with an appropriate threshold correlation to discard data sources that are not similar as shown in Table 6.2. We use only the range between 0 and 1, the positive correlation.

Table 6.2: After applying PCor with an appropriate threshold correlation

<table>
<thead>
<tr>
<th>Data sources</th>
<th>Similar data sources (with PCor ≥ threshold correlation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS₁</td>
<td>DS₂  DS₃  DS₄  0  0  ...  0  0  0</td>
</tr>
<tr>
<td>DS₂</td>
<td>DS₁  DS₃  DS₄  ...  0  0  0  0</td>
</tr>
<tr>
<td>DS₃</td>
<td>DS₂  DS₁  DS₄  0  ...  0  0  0</td>
</tr>
<tr>
<td>...</td>
<td>...  ...  ...  ...  ...  ...  ...  ...</td>
</tr>
<tr>
<td>DS₂₀</td>
<td>DS₁₇ DS₁₅ DS₁₃ 0  ...  0  0  0</td>
</tr>
</tbody>
</table>

6.1.2 Correlation measures

In this section, we explain the proposed approach for finding redundant data sources that are correlated using a correlation measure with an appropriate threshold correlation. This approach consists of two steps:

1. data sources normalization step,
2. data sources filter step using a correlation measure with an appropriate threshold correlation.
6.1.2.1 Data sources normalization

As for similarity measure, we use the Z-normalization. This step is optional for correlation measures, because the experiment in the previous chapter has shown that Z-normalization has no effect on the results of the correlation measures. However, in order to reasonably measure the correlation between two time, authors in [28, 29] have suggested to use normalization.

6.1.2.2 Data sources filter using correlation measure

As shown in Table 6.3, the correlation between data sources is computed. If the correlation value is $\leq$ threshold correlation the corresponded data source is discarded.

<table>
<thead>
<tr>
<th>Data sources</th>
<th>Correlated data sources with an threshold correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>DS2</td>
</tr>
<tr>
<td>DS2</td>
<td>DS1</td>
</tr>
<tr>
<td>DS3</td>
<td>DS2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>DS20</td>
<td>DS17</td>
</tr>
</tbody>
</table>

Table 6.3: After applying correlation measure with an appropriate threshold correlation

6.2 Finding the threshold for real data

In this section, two experiments are performed on the real data. One experiment uses similarity measures in combination with PCor measure to find redundant data sources that are similar. The other experiment uses correlation measures to find redundant sources that are correlated. These experiments determine the appropriate threshold correlation.

6.2.1 Real data from the environment

The real data on which we perform the experiments are collected from two rooms in two different locations. One room was at the TU/e in Metaforum room MF 6.111. The other room was at my home in Utrecht. By doing this, we know for sure data from these rooms are different. Data is collected with the following sensors:

- 4 PIR sensors,
- 8 TMP sensors,
- 4 HUM sensors,
- 4 light sensors.

These sensors are deployed on the MyraInModem and deployed in the two rooms as shown in Figure 6.2. The system architecture that is described in Chapter 3 is used to collect the data from the environment.

6.2.2 Expected redundant data sources

After collecting data from these rooms, the following scenario is made in order to determine the threshold correlation:

The PIR1,1, TMP1,3, HUM1,7, Light1,9, PIR2,11, TMP2,13, HUM2,17 and Light2,19 sensors are broken, we want to replace them with their redundant data sources. Table 6.4 and Table 6.5 show the expected redundant data sources for these broken sensors that are similar and correlated,
Figure 6.2: Case study: two rooms equipped with different sensor.

Table 6.4: The expected data sources that are similar

<table>
<thead>
<tr>
<th>Failure</th>
<th>The expected data sources that are similar</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1_1</td>
<td>0</td>
</tr>
<tr>
<td>TMP1_3</td>
<td>0</td>
</tr>
<tr>
<td>HUM1_7</td>
<td>0</td>
</tr>
<tr>
<td>Light1_9</td>
<td>0</td>
</tr>
<tr>
<td>PIR2_11</td>
<td>0</td>
</tr>
<tr>
<td>TMP2_13</td>
<td>0</td>
</tr>
<tr>
<td>HUM2_17</td>
<td>0</td>
</tr>
<tr>
<td>Light1_9</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.5: The expected data sources that are correlated

<table>
<thead>
<tr>
<th>Failure</th>
<th>The expected data sources that are correlated</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1_1</td>
<td>0</td>
</tr>
<tr>
<td>TMP1_3</td>
<td>0</td>
</tr>
<tr>
<td>HUM1_7</td>
<td>0</td>
</tr>
<tr>
<td>Light1_9</td>
<td>0</td>
</tr>
<tr>
<td>PIR2_11</td>
<td>0</td>
</tr>
<tr>
<td>TMP2_13</td>
<td>0</td>
</tr>
<tr>
<td>HUM2_17</td>
<td>0</td>
</tr>
<tr>
<td>Light1_9</td>
<td>0</td>
</tr>
</tbody>
</table>

respectively.

**Room 1:**
- PIR1_1 and PIR1_2 are able to detect the presence of persons in room1. The detection regions of these sensors do not intersect with each other. PIR1_1 and PIR1_2 detect the presence of a person not at the same time. PIR1_1 gives more information about the presence of a person in the room than PIR1_2, because PIR1_1 is next to the door. As a result, the PIR1_1 and the PIR1_2 will be low correlated and not similar in the most of the time.
- Light1_10 and Light1_9 detect light intensity in the room. Light intensity is not uniformly spread in the room and Light1_10 get more light than Light1_9. Because Light1_10 is next to the window. As a result Light1_10 and Light1_9 are low correlated and not similar.
• The temperature is everywhere the same in the room, TMP1.3, TMP1.4, TMP1.5 and TMP1.6 will be highly correlated and in the most of the time similar.

• We expect that the humidity is everywhere the same in the room, HUM1.7 and HUM1.8 will be highly correlated and in the most of the time similar.

• HUM1.7 and HUM1.8 in some how will be correlated with TMP1.3, TMP1.4, TMP1.5 and TMP1.6.

**Room 2:**

• PIR2.11 and PIR2.12 are next to each other. The detection regions of these sensors do intersect with each other. PIR2.11 and PIR2.12 detect the presence of a person at the same time. As a result, the PIR2.11 and the PIR2.12 will be highly correlated and in the most of the time similar.

• Light intensity is not uniform spread in the room and Light2.19 get more light than Light2.20. Because Light2.19 is next to the window. As a result Light2.19 and Light2.20 are low correlated and not similar.

• The temperature is everywhere the same in the room, TMP2.13, TMP2.14, TMP2.15 and TMP2.16 will be highly correlated and in the most of the time similar.

• The humidity is everywhere the same in the room, HUM2.17 and HUM2.18 will be highly correlated and in the most of the time similar.

• HUM2.17 and HUM2.18 in some how will be correlated with TMP2.13, TMP2.14, TMP2.15 and TMP2.16.

### 6.2.3 Performance metrics

To compare the expected redundant data sources with the redundant data sources from the experiment we use the following performance metrics:

- **Accuracy:**
  - Defined as \( \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Negative} + \text{False Positive} + \text{True Negative}} \).
  - The ratio of how close the results from the experiment to the expectation.

- **Precision:**
  - Defined as \( \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \).
  - The ratio of a collected data from an environment is a redundant data.

- **Recall:**
  - Defined as \( \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \).
  - The ratio of redundant data in a collected data from an environment.

- **F1 score:**
  - Defined as \( \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \).
  - Combines precision and recall.
  - Trade off between precision and recall.

We define True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) as follows:
• TP defined as:
  Found redundant data, is a true redundant data.

• FP defined as:
  Found redundant data, is a false redundant data.

• TN defined as:
  No redundant data is found, is true.

• FN defined as:
  No redundant data is found, is false.

6.2.4 Varying the threshold\textit{correlation} value
In this section, we vary the threshold value between 0 and 1. We choose the threshold value that give the highest value in terms of the accuracy, precision, recall and F1 score.

6.2.4.1 Similarity measures
Figure 6.3 shows the accuracy, precision, recall and F1 score for the combination of similarity measures and PCor for a threshold between 0 and 1.

![Graphs showing impact of threshold on accuracy, precision, recall, and F1 score](image.png)

Figure 6.3: Impact of threshold on the real data.
CHAPTER 6. FINDING REDUNDANT DATA SOURCES

As the value of the threshold increases:

- **Accuracy:**
  is increasing for all similarity measures in combination with PCor until a threshold of 0.8, then the accuracy decreases. A value one for accuracy means that the expected redundant data sources are the same as from the experimental results.

- **Precision:**
  is increasing for all similarity measures in combination with PCor until a threshold of 0.8, then the precision stays constant. The increase in the precision results in fewer false redundant data sources. Precision one means we found redundant data sources that we have expected. However, it does not mean we have all redundant data sources. we may miss some redundant data source.

- **Recall:**
  is decreasing for all similarity measures in combination with PCor. This reduction decreases the number of the redundant data sources.

- **F1 score:**
  is increasing for all similarity measures in combination with PCor until a threshold of 0.8, then the accuracy decreases. The increase in the F1 score results in a good balance between recall and precision.

From the graph ED, SBD and ExtJaccard in combination with PCor and a threshold of 0.8 give the highest value in terms of the accuracy, precision, recall and F1 score.

6.2.4.2 Correlation measures

Figure 6.4 shows the accuracy, precision, recall and F1 score for the correlation measures for a threshold between 0 and 1.

As the value of the threshold increases:

- **Accuracy:**
  is increasing for all correlation measures until a threshold of 0.8, then the accuracy decreases.

- **Precision:**
  is increasing for all correlation measures until a threshold of 0.9, then the precision stays constant.

- **Recall:**
  is decreasing for all similarity measures and correlation measures.

- **F1 score:**
  is increasing for all similarity measures and correlation measures until a threshold of 0.8, then the accuracy decreases.

From the graph, a threshold of 0.9 is a good threshold for all correlation measures. At this threshold value, the precision is one. It means that the redundant data sources are found. But not all the redundant data sources are found. Because recall is smaller than one.
6.2.5 Expected redundant data sources versus results

As shown in Table 6.6, the results of the experiment using ED, SBD or ExtJaccard in combination with PCor and a threshold of 0.8 are the same as shown in Table 6.4. Table 6.7 shows the results of the experiment using correlation measures with a threshold of 0.9. These results are not the same as shown in Table 6.5. Table 6.8 and Table 6.9 give the correlation measures between PIRs in room1 and between HUM sensors and TMP sensors in both room.

<table>
<thead>
<tr>
<th>Failure</th>
<th>The experimental results: the redundant data sources that are similar</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1_1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>TMP1_3</td>
<td>TMP1_4  TMP1_5  TMP1_6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HUM1_7</td>
<td>HUM1_8  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Light1_9</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>PIR2_11</td>
<td>PIR2_12  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>TMP2_13</td>
<td>TMP2_14  TMP2_15  TMP2_16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HUM2_17</td>
<td>HUM2_18  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Light1_19</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Table 6.6: The redundant data sources using SBD and PCor with a threshold of 0.8.

Figure 6.4: Impact of threshold on the real data.
CHAPTER 6. FINDING REDUNDANT DATA SOURCES

Table 6.8 and Table 6.9 show that there is correlation between PIRs in room1 and correlation between HUM and TMP in both rooms, but the correlation is lower than 0.9.

<table>
<thead>
<tr>
<th>Failure</th>
<th>The experimental results: the redundant data sources that are correlated</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1_1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>TMP1_3</td>
<td>TMP1_4 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HUM1_7</td>
<td>HUM1_8 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Light1_9</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>PIR2_11</td>
<td>PIR2_12 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>TMP2_13</td>
<td>TMP2_14 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HUM2_17</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Light1_19</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Table 6.7: The redundant data sources using correlation measures with a threshold of 0.9.

<table>
<thead>
<tr>
<th>PCor</th>
<th>PIR1_2</th>
<th>TMP1_4</th>
<th>TMP1_5</th>
<th>TMP1_6</th>
<th>HUM1_8</th>
<th>TMP2_14</th>
<th>TMP2_15</th>
<th>TMP2_16</th>
<th>HUM2_18</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1_1</td>
<td>0.732</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMP1_3</td>
<td></td>
<td>0.291</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HUM1_7</td>
<td>0.035</td>
<td>0.273</td>
<td>0.122</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMP2_13</td>
<td>0.403</td>
<td>0.457</td>
<td>0.392</td>
<td>0.871</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HUM2_17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.8: Correlation using PCor.

<table>
<thead>
<tr>
<th>DCor</th>
<th>PIR1_2</th>
<th>TMP1_4</th>
<th>TMP1_5</th>
<th>TMP1_6</th>
<th>HUM1_8</th>
<th>TMP2_14</th>
<th>TMP2_15</th>
<th>TMP2_16</th>
<th>HUM2_18</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR1_1</td>
<td>0.745</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMP1_3</td>
<td></td>
<td>0.317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HUM1_7</td>
<td>0.319</td>
<td>0.439</td>
<td>0.339</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMP2_13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.565</td>
</tr>
<tr>
<td>HUM2_17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.562</td>
<td>0.576</td>
</tr>
</tbody>
</table>

Table 6.9: Correlation using DCor.

6.3 Conclusion

From the above experiments, we conclude the followings:

- ED, SBD and ExtJaccd in combination with PCor and a threshold of 0.8 a good candidate to find redundant data sources that are similar.
- DCor and PCor with a threshold of 0.9 a good candidate to find redundant data sources that are highly correlated.
- There is a correlation between TMP sensor and HUM sensor, but the correlation is low.
- Distance between sensors has effect on the similarity and correlation.
Chapter 7

Performance Evaluation of The Smart Lighting Application

7.1 Performance evaluation of the smart lighting application

In this chapter, the performance of an application is evaluated in terms of availability and reliability. The availability and the reliability performance metrics are defined as follows:

- **Availability** is defined as:
  \[
  \text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \times 100\% ,
  \]
  where
  - MTTR is Mean Time To Recover from a failure.
  - MTBF is Mean Time Between Failures.

- **Reliability** is defined as:
  \[
  \text{Reliability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}
  \]

Figure 7.1: Case study: smart room

Further, we make the following scenario:
A SE is equipped with light sensors, temperature sensors, humidity sensors and PIR sensors as shown in Figure 7.1. The scenario makes use of a Smart Lighting Application (SLA). Based on the output data of the PIR sensor, SLA turns light ON or OFF. In the current situation, when PIR sensor fails, SLA stops working. SLA stays on this condition until the user reports it and a maintenance personnel investigates the failure state of SLA and replaces the failed PIR sensor with a new PIR sensor. This takes at least 1 day to solve this problem. The SLA application is not available for at least 1 day.

7.1.1 Performance of the application without redundant data sources

Figure 7.2 shows the time line for the working duration of the SLA. The unit of the time line is in hours. As shown in Figure 7.2, during 168 hours \((t_3 - t_1)\) the SLA is 144 hours available.
and 24 hours not available due to failure. Further, it takes 24 hours \((t_2 - t_1)\) to recover from the failure. Therefore, the availability and the reliability of the SLA without redundant data sources is \((\text{MTBF} = t_3 - t_2 = 168 - 24 = 144 \text{ hours and } \text{MTTR} = 24 \text{ hours})\) \(\frac{144}{144+24} \times 100 = 85.71\%\) and \(168 - 24 = 144 \text{ hours, respectively.}\)

![Figure 7.2: Time line of the working of the SLA without redundant data sources.](image)

### 7.1.2 Performance of the application with redundant data sources

As shown in Figure 7.3, time to recover \((t_2 - t_1)\) is shorter than without redundant data sources. If we ignore the time to collect data. It takes 0.74117 seconds \((0.0002 \text{ hours})\) if we use SBD together with PCor and a threshold correlation of 0.8 to recover from failure. Data collecting is done before a failure occurs. Therefore, the availability and the reliability of the SLA with redundant data sources is \((\text{MTBF} = t_3 - t_2 = 168 - 0.0002 = 167.9998 \text{ hours and } \text{MTTR} = 0.0002 \text{ hours})\) \(\frac{167.9998}{167.9998+0.0002} \times 100 = 99.9998\%\) and \(168 - 0.0002 = 167.9998 \text{ hours, respectively.}\)

![Figure 7.3: Time line of the working of the SLA with redundant data sources.](image)

### 7.1.3 Conclusion

We can conclude that replacing a failed sensor with a redundant data source has a huge effect on the availability and the reliability. The reliability and availability of the application are increased. Further, the maintenance cost reduces.
Chapter 8

Conclusions and Future Work

This thesis is about finding redundant data sources for a failed sensor in a smart environment. In order to find a redundant data source, the state of the art similarity and correlation measures have been studied. These similarity and correlation measures are able to find similarity and correlation between two time series, respectively. To choose a suitable similarity and correlation measure, an experiment is performed on synthetic data sets. These synthetic data sets represent the data in the real world. Data in the real world are in some way distorted. Two time series have the same shape, but one of this time series can have the following distortions:

- uniform amplitude shifting distortion,
- out of phase distortion,
- uniform amplitude scaling distortion,
- noise distortion,
- the combination of the above distortion.

The experiment shows that SBD and DCor measure give good results when the time series are distorted. Further, the experiment shows that Z-normalization can eliminate the uniform amplitude shifting distortion, the uniform amplitude scaling distortion and the measurement units. However, thresholds are still needed for similarity and correlation measures. Therefore, experiments are performed on the real data to determine these thresholds. The real data are collected using various sensors from two different rooms in two different locations to make sure that the data from these two rooms are different. The sensors are deployed on MyriaModem.

From the experiment, the following conclusions are made:

1. ED, SBD and ExtJaccd in combination with PCor and a threshold of 0.8 are a good measures to find redundant data sources that are similar.
2. DCor and PCor with a threshold of 0.9 are a good measures to find redundant data sources that are highly correlated.
3. There is a correlation between TMP sensor and HUM sensor, but the correlation is lower than 0.9.
4. Distance between sensors has effect on the similarity and correlation.

Based on the experiments on the synthetic data sets and on the real data set, the following conclusions are made:

1. SBD in combination with PCor and a threshold of 0.8 is a better measures to find redundant data sources that are similar even if the data sources are distorted.
2. DCor with a threshold of 0.9 is a better measure to find redundant data sources that are highly correlated even if the data sources are distorted.
CHAPTER 8. CONCLUSIONS AND FUTURE WORK

8.1 Contribution

1. A comparison between the state of the art similarity measures.
2. A comparison between the state of the art correlation measures.
3. A system architecture that collects data from the environment and replaces a failing sensor. This architecture is implemented on a wireless network platform called MyriaModem.
4. A solution for finding redundant data sources.
5. An appropriate threshold for correlation measure.

8.2 Future work

- Find an appropriate threshold for correlation measure that takes into account the distance between sensors in the given environment and the types of sensors.
- Find a conversion function that converts the output data of a redundant data source (e.g. temperature sensor) to the output data of the failed sensor (e.g. humidity sensor).
- Determine an application state function $State_{App}(App\_Set) \in \{\text{TApp}, \text{FApp}\}$ that gives the status of an application.
- Determine a data source state function $State_{DS}(DS\_Set) \in \{\text{TDS}, \text{FDS}\}$ that gives the status of a data source (sensor).
Bibliography


[22] MICHAEL CLARK. A comparison of correlation measures.


Appendices
Appendix A

The experimental results

Figure A.1: ED similarity measure in combination with PCor.
APPENDIX A. THE EXPERIMENTAL RESULTS

Figure A.2: CIDED similarity measure in combination with PCor.

Figure A.3: DTW similarity measure in combination with PCor.
APPENDIX A. THE EXPERIMENTAL RESULTS

Figure A.4: CIDDTW similarity measure in combination with PCor.

Figure A.5: SBD similarity measure in combination with PCor.
APPENDIX A. THE EXPERIMENTAL RESULTS

Figure A.6: ExtJaccard similarity measure in combination with PCor.

Figure A.7: PCor correlation measure.
Figure A.8: DCor correlation measure.