Application of a weighted distance metric using simulated annealing: As an extension of the predictive approach “Event Simulator”

by

Arjen Schröder

BSc Industrial Engineering & Innovation Science – TU/e 2012
Student identity number 0634615

In partial fulfillment of the requirements for the degree of

Master of Science
In Operations Management and Logistics

Supervisors:
Dr. ir. R.J.I (Rob) Basten (TU/e)
Dr. rer. nat. G.M. (Gero) Walter (TU/e)
Ir. B. (Bob) Huisman (NedTrain B.V.)
D. (Daniel) Jaroszewski (Frankfurt Consulting Engineers GmbH)
Subject headings: Condition Based Maintenance (CBM), Event Prediction, Pattern Recognition, Similarity Sampling, Continuous Optimisation, Stochastic Heuristic, Simulated Annealing (SA)
Abstract

This master thesis is conducted on behalf of NedTrain, in cooperation with Frankfurt Consulting Engineers (FCE). In 2014, NedTrain started the Real-Time Monitoring (RTM) project in which it tries to make unplanned maintenance activities more plannable, by making use of logged data from trains. FCE is developing a method, called similarity sampling, to forecast failure events based on diagnostic events, using pattern recognition, which is incorporated in the Event Simulator tool. This method predicts failure events based on the similarity of an observed pattern of diagnostic events and historical patterns, for which it is known if it was followed by a failure event. In order to determine the similarity between patterns it evaluates the distances.

In this research we evaluated if the application of a weighted distance metric, in which diagnostic events receive relative weights, improves the prediction accuracy, compared to the Euclidean distance metric, in which all diagnostic events are treated equally. To evaluate the effects of the application of such weighted distance metric, we developed a simulated annealing (SA) methodology and applied it, to find an optimal set of weights for this distance metric. Furthermore, we evaluated how the best prediction accuracy could be obtained by varying other parameters (step size (SS), embedding window (EW) and CDF threshold).

We conclude that applying a weighted distance metric can indeed improve the prediction accuracy substantially. Although this improvement is substantial, SA turns out to be not the appropriate method to find an (near) optimal set of weights and other methods should be considered. Furthermore, the choice of the other parameters is crucial in terms of the prediction accuracy.
Preface

This report is the result of a graduation project for the completion of the master program Operations Management & Logistics at Eindhoven University of Technology. In this preface, I would like thank all the people that made this project possible.

First, I would like to thank my supervisors from the TU/e: Rob Basten, my first supervisor, for his guidance, input and feedback during this project and my second assessor, Gero Walter, for his input and feedback in finalizing this master thesis. I also thank Joachim Arts, who supervised me during the first stages of this project.

Furthermore, many thanks go to Bob Huisman from NedTrain and to Wolfgang Mergenthaler from Frankfurt Consulting Engineers (FCE) for the opportunity to conduct this research. In particular, I would also like to thank Daniel Jaroszewski from FCE for all the help during this research. I really enjoyed the visits to Hochheim, which were very informative and instructive.

Last but not least, I would like to thank my family and my girlfriend for all their support, especially in the last year with all the encountered setbacks.
Executive Summary

In 2014, NedTrain started the Real-Time Monitoring (RTM) project in which it tries to make unplanned maintenance activities more plannable, by using logged data from trains. In cooperation with Frankfurt Consulting Engineers (FCE), they analyze which of these data contain predictive value for future failures. FCE is developing a tool, called the Event Simulator, which aims to predict critical events (e.g. unplanned maintenance events or severe diagnostic error messages of the train) using pattern recognition. It compares occurrences of relevant diagnostic events (observations) with historical occurrences of diagnostic events (patterns) for which it is known whether or not it was followed by a critical event. Diagnostic events are diagnostic messages logged by the train. Patterns and observations can be described as state vectors, in which each dimension represents the number of occurrences of a certain relevant diagnostic event, counted over a time window. Based on the similarity of an observation and the patterns, a forecast is made whether or not it is likely that the critical event will occur after the observation. For this forecast the most similar patterns, called the neighborhood, are taken into account, for which it is known if a critical event will occur afterwards. Based on this neighborhood of patterns, an empirical cumulative distribution is established for the probability of the occurrence of a critical event. Together with a threshold for this distribution it is decided if a warning for a critical event is given. This method is called similarity sampling.

One important aspect in the similarity sampling method is the way in which it evaluates which patterns are similar to the observation. In order to do so, the distance between an observation state vector and all the stored pattern state vectors is calculated and the patterns with the smallest distances to the observation are taken into account in the prediction if the critical event is likely to occur. In the initial tool, the Euclidean distance metric is used to calculate the distances between an observation’s state vector and all the state vectors of patterns. This corresponds to weighting each dimension in the state vector (e.g. relevant diagnostic event) equally in the determination of the distance. However, some relevant diagnostic events might be more important than others in the possible followed critical event, since they have more predictive value. Therefore, it might be beneficial in term of the prediction accuracy to weight some dimensions in the state vector more than others in the determination of the similarity between an observation and the known patterns.

The goal of this thesis is to find out if the prediction accuracy can be improved by using such a weighted distance metric, compared to the Euclidean distance metric. Ultimately, a set of weights should be found that optimizes the prediction accuracy for the prediction of a critical event. Besides the effects of using a weighted distance metric, also other parameters within the Event Simulator are analyzed during this research to see how they affect the prediction accuracy. The parameters taken into account are: the embedding window (a time period over which occurrences are counted and stored in a pattern), the step size (the time interval for which pattern are generated) and the CDF threshold (which determines if a warning is given).

To optimize the weighted distance metric we use the simulated annealing algorithm, a local search algorithm that is known for avoiding getting trapped in local optima. It generates and assesses neighboring solutions and allows a range of worse solutions in order to find a more global optimum in the search space. The acceptance probability of worse solutions decreases during the algorithm, such that in theory it converges to a global optimum. The other parameters (embedding window, step size and CDF threshold) can be optimized by the use of grids, such that multiple combinations of parameters can be evaluated within an upper and lower bound. In order to quantify the effects of
the weighted distance metric and the other parameters, a real-life test case is used, which is provided by FCE. Based on this test case, we first evaluated to what extent the prediction accuracy can be improved by optimizing the other parameters and based on the optimal combination of these parameters, we analyzed how much the prediction accuracy can be further improved by using a weighted distance metric. Furthermore, we evaluated how well the simulated annealing algorithm can find an optimal set of weights for this specific test case.

With the optimization of the embedding window and step size parameters, the prediction accuracy can be improved substantially. The optimal combination of these parameters is surrounded with a robust optimal area in which near optimal combinations only had slightly worse prediction accuracies then the optimal combination of parameters. For combinations outside this robust area the prediction accuracy decreases drastically. Combinations closely around the robust area performed 50% worse than the optimal combination and the prediction accuracy only decreases more, by moving further away from the robust optimal area.

By applying simulated annealing to optimize the weighted distance metric on the optimal combination the best found result improves the prediction accuracy with an additional 27% compared to the optimal combination with a Euclidean distance metric. However, the algorithm of simulated annealing appeared not to be an appropriate method to always find such substantial improvements of a weighted distance metric. Due to the extensive computation time necessary, the search space could only be explored to a limited extent, which has the result that only local optimums are found. The quality of the found weighted distance metric is therefore dependent on the parts of the search space that is explored during the optimization. Although the method to find optimal weights for the distance metric did not result in the desired results in terms of reproducibility, the potential improvements that were found for the test case show the potential gain in using a weighted distance metric over the Euclidean distance metric. We conclude that the optimization of other parameters is crucial for the prediction accuracy and that the optimization of a weighted distance metric should be incorporated in the tool, however other approaches should be considered in order to found these weights.

Although the simulated annealing algorithm did not have the desired reproducible effect on the prediction accuracy, a methodology was established for using the tool for the prediction of any critical event in which this algorithm is applied. It was planned to use a case of NedTrain in order to test the methodology and to verify if the optimization of the distance metric (and the other parameters) resulted in the same order size of potential improvements in terms of prediction accuracy. However, the data of NedTrain could not be used in the Event Simulator, due to an insufficient number of occurrences and the lack of finding corresponding predecessors for critical events, which are necessary in order to make predictions using the tool. Since no predecessors were found for a wide range of critical incidents (e.g. maintenance activities or severe diagnostic error messages), this indicates that the diagnostic events of NedTrain do not contain sufficient predictive value for the forecasting of critical events. It is suggested that NedTrain can use sensor data instead of event data in order to predict critical events. With sensor data the patterns contain the average value of sensor data during an embedding window instead of the occurrences of events, to represent the state of the system. The optimization of the weighted distance metric and the other parameters evaluated during this research are also still applicable when sensor data is used.
Table of Contents

I List of Figures ......................................................................................................................... IX
II List of Tables .......................................................................................................................... X
1 Introduction ............................................................................................................................. 1
   1.1 NedTrain ............................................................................................................................ 1
   1.2 Maintenance ....................................................................................................................... 2
      1.2.1 Planned maintenance ................................................................................................. 2
      1.2.2 Unplanned Maintenance ............................................................................................ 3
   1.3 Condition Based Maintenance .......................................................................................... 3
      1.3.1 Data Acquisition ........................................................................................................ 4
      1.3.2 Data processing .......................................................................................................... 4
      1.3.3 Maintenance Decision Making ................................................................................ 4
   1.4 Real Time Monitoring (RTM) Project ................................................................................ 5
2 Problem Statement .................................................................................................................... 7
   2.1 Similarity Sampling ............................................................................................................ 7
   2.2 Distance Metric .................................................................................................................. 8
   2.3 Application of the distance metric .................................................................................... 9
   2.4 Research Objective .......................................................................................................... 10
   2.5 Research Approach .......................................................................................................... 10
3 Prediction model: FCE Event Simulator .................................................................................. 12
   3.1 Description of the initial model ...................................................................................... 12
      3.1.1 Pre-processing and set-up ......................................................................................... 12
      3.1.2 Training set .............................................................................................................. 13
      3.1.3 Monitoring Set ......................................................................................................... 13
      3.1.4 Optimization Routine ............................................................................................. 15
   3.2 Optimization of the weighted distance metric ................................................................. 16
   3.3 Other influencing parameters ......................................................................................... 16
      3.3.1 Embedding Window and Step Size ......................................................................... 17
      3.3.2 CDF Threshold ....................................................................................................... 18
      3.3.3 Selection of Input parameters ............................................................................... 18
4 Literature Review .................................................................................................................... 20
   4.1 Global minimization problem ......................................................................................... 20
   4.2 Local Search Technique .................................................................................................. 21
   4.3 Stochastic heuristics in local search .............................................................................. 22
      4.3.1 Two phased methods ............................................................................................ 22
Appendix C: Notations and definitions of the initial model description

References

Appendix A: Notations and definitions of the initial model description

Appendix B: Flowchart of the initial model description

Appendix C: Example of an application of simulated annealing: TSP
Appendix D: Implementation of simulated annealing................................................................. 59
Appendix E: Flowchart of the Simulated Annealing Optimization ............................................... 62
Appendix F: Source Code of the Simulated Annealing Optimization........................................... 67
Appendix G: Effects of the temperature level and the annealing schedule ................................. 73
Appendix H: Guidelines for Event Simulator based on gained insights .................................... 76
Appendix I: User Manual of the Event Simulator ....................................................................... 80
I List of Figures

Figure 1 Organizational chart NS group (Dutch Railways (NS), 2015) and organizational chart NedTrain (van Aspert, 2013) ................................................................................................................................. 1
Figure 2 Graphical representation of similarity sampling method ......................................................................................................................... 8
Figure 3 Graphical representation of the influence for adding weight to the distance metric ................................................................. 9
Figure 4 Research design .................................................................................................................................................................................................. 11
Figure 5 Training set, monitoring set and the moving time window (Jaroszewski, 2014) ........................................................................ 12
Figure 6 Cross correlation function of a critical event E and all other events F (Jaroszewski, 2014) ... 13
Figure 7 Forward similarity sampling (Jaroszewski, 2014) ............................................................................................................................................. 15
Figure 8 Flowchart of the FCE Event Simulator (with optimization) ........................................................................................................... 17
Figure 9 Landscape of state space in local search techniques ................................................................................................................. 21
Figure 10 Pseudo code of a threshold algorithm ......................................................................................................................................................... 24
Figure 11 Pseudo code of the simulated annealing algorithm ................................................................. 26
Figure 12 Pseudo code of the application of simulated annealing for the optimization of the distance metric ................................................................................................................................. 28
Figure 13 Jump through system space ........................................................................................................................................................................ 35
Figure 14 Movements in configuration space ................................................................................................. 35
Figure 15 Occurrences of critical event 0642 on vehicle 015 divided over the training and monitoring set .................................................................................................................................................. 39
Figure 16 Robust area of optimal parameter combination (for objective function as in (5.2-14)) ..... 40
Figure 17 Effect of step size and embedding window for objective function (5.2-13) (left) and (5.2-15) (right) ................................................................................................................................................................................... 42
Figure 18 Prediction accuracy result on objective equation (5.2 14), (5.2-13) and (5.2-15), respectively ............................................................................................................................................................................................. 43
Figure 19 Prediction accuracy results Euclidean distance (left) and weighted distance metric (right) 44
Figure 20 Graphical example of simulated annealing on local search space area ................................. 46
Figure 21 Initializing steps..................................................................................................................................................................................................... 59
Figure 22 Nested loops of reconfigurations and cooling down .......................................................................................................................... 60
Figure 23 Generator of random changes in the configuration .................................................................................................................. 61
Figure 24 Flowchart of accepting solutions (with metropolis criterion) .............................................................. 61
Figure 25 Probability distributions for a accepting decrease in objective function of scenarios in numerical example .................................................................................................................................................................................. 74
Figure 26 Acceptance probability for a decrease of 2 in the objective function for the numerical examples ........................................................................................................................................................................... 74
Figure 27 Methodology for applying the Event Simulator ......................................................................................................................... 76
II List of Tables

Table 1 Overview of discussed performance measures ................................................................. 29
Table 2 MSE example large vs. small errors .................................................................................. 29
Table 3 MAPE example of asymmetry ......................................................................................... 30
Table 4 SMAPE example of asymmetry ......................................................................................... 30
Table 5 Contingency table ............................................................................................................... 31
Table 6 Types of reconfigurations ............................................................................................... 37
Table 7 Fixed settings during evaluation of test case ................................................................. 39
Table 8 Effects on average and optimal objective value for fixed CDF thresholds based on .... 40
Table 9 Prediction performance (objective value) within and outside robust area .................... 41
Table 10 Deterioration (%) compared to optimal in prediction performance within and outside robust area ................................................................................................................................. 41
Table 11 Different objective functions and their corresponding weights together with their optimal parameter settings .................................................................................................................. 42
Table 12 Improvement of weighted distance metric compared to Euclidean distance .......... 44
Table 13 Result of multiple runs using the parameter setting in which the best improvement was found ...................................................................................................................................................... 45
Table 14 Effect of the annealing length by varying the number of temperature levels .......... 47
Table 15 Parameters of numerical examples Metropolis criterion ............................................ 73
Table 16 Resulting temperature levels of numerical example Metropolis criterion .................. 73
1 Introduction

This master thesis is related to the Real Time Monitoring (RTM) project of NedTrain and is part of the development of a forecasting method to predict failure events, in cooperation with Frankfurt Consulting Engineers (FCE). First, some general background of NedTrain is given in Section 1.1. The RTM project aims towards getting more insight in unplanned maintenance tasks and is related to the concept of condition based monitoring (CBM). The maintenance tasks performed by NedTrain are described in Section 1.2 and some general background of CBM is given in Section 1.3. Finally, the RTM project itself is further elaborated in Section 1.4, together with the role of FCE within this project.

1.1 NedTrain

NedTrain performs all maintenance activities for the Dutch Railways, “Nederlandse Spoorwegen” (NS). It is a subsidiary company to the NS business group. The history of NedTrain starts in 1938 as a part of the Dutch Railways to perform the maintenance processes on rolling stock. During the 1990’s the national railways separated formally from the national government, resulting in the independence of NedTrain. Besides the rolling stock of NS, which is the main customer of NedTrain, also vehicles from other railway companies are nowadays maintained at NedTrain, like those of Arriva and Veolia (NedTrain, 2015). NedTrain’s headquarters is located in Utrecht, the Netherlands.

NedTrain has become one of Europe’s leading rolling stock maintenance and revision companies for both passengers and freight carriers with over 3,000 employees and a turnover of about 500 million euros in 2015 (Dutch Railways (NS), 2015). NedTrain consists of four business units: Business Development, Maintenance & Service, Fleet Services and Refurbishment & Overhaul, that operate under the board of NedTrain. The current research is conducted on behalf of the department of Maintenance Development, which is part of Fleet Services. An overview of the organization of NedTrain is shown in Figure 1, together with an organizational chart of the NS group.

Figure 1 Organizational chart NS group (Dutch Railways (NS), 2015) and organizational chart NedTrain (van Aspert, 2013)
1.2 Maintenance
NedTrain aims to get the highest availability of rolling stock at the lowest possible costs. NedTrain’s maintenance activities can be distinguished by planned and unplanned maintenance. During each of these types of maintenance, both planned and unplanned jobs can be performed. First, planned maintenance is discussed in Section 1.2.1, followed by unplanned maintenance in Section 1.2.2.

Maintenance is introduced as an efficient way to assure a satisfactory level of reliability, as products deteriorate over time, because of certain stress or load in the real environment (Jardine et al., 2006). Niu et al. (2010) describes two types of maintenance: corrective maintenance and preventive maintenance.

The unplanned maintenance tasks of NedTrain correspond to corrective maintenance, which is applied after a breakdown or failure has been detected (Jardine et al., 2006). This type of maintenance is also known as reactive maintenance. In general, corrective maintenance is only suitable for non-critical parts or systems, due to the corresponding down time during failure or breakdown. According to Tsang (1995) the maintenance costs are usually high because of the high cost for restoring equipment to an operable condition under a crisis situation, the secondary damage and safety hazards inflicted by the failure and the penalty costs associated with lost production.

The planned maintenance tasks of NedTrain can be classified as preventive maintenance, in the form of usage-based maintenance or time-based maintenance. Preventive maintenance can be described as maintenance that is carried out to prescribed criteria or at predetermined intervals and has the goal to reduce the probability of failure or degradation of functioning of an item (Niu et al., 2010). When the condition of a component or a system is not monitored, then often predetermined maintenance is performed (Jardine et al., 2006). Maintenance can be scheduled at fixed dates or after a predetermined number of operating hours. Usage based maintenance is only effective when the failure rate of a part or system is an increasing function of its age or usage.

1.2.1 Planned maintenance
NedTrain’s planned maintenance activities can be classified into four categories (Snoek, 2014):

- Daily maintenance, e.g. safety checks and cleaning;
- Short-cyclic maintenance, e.g. extensive safety checks and necessary replacements of components on a three monthly basis;
- Long-cyclic maintenance;
- Overhaul, e.g. revision activities once every fifteen to twenty years.

These planned maintenance tasks are performed for over 2,800 coaches at locations spread over the Netherlands. Four different types of locations, shown in the lower section of the organizational chart of NedTrain in Figure 1, can be distinguished (van Aspert, 2013; de Jonge, 2014):

- Repair shops, consisting of a components company, “NedTrain Componenten Bedrijf” (NCB) in Tilburg, where parts are refurbished and repaired and a Refurbishment & Overhaul (R&O) workshop in Haarlem, where trains are completely revised and modernized and where maintenance on wheel sets and bogies is performed.
- Maintenance depots, “Onderhouds Bedrijf” (OB): five in total located strategically over the Netherlands where both planned and unplanned maintenance activities are performed.
• Service companies, “Service Bedrijf” (SB): 30 in total, which are located directly near the railway network, where minimal repairs and daily checks are performed, together with cleaning activities.
• Technical centres, which are service companies where also larger maintenance activities can be performed.

During a planned maintenance service two types of jobs may be performed, planned and unplanned jobs. Although a train has a scheduled arrival, a lot of unplanned jobs have to be performed, since the condition of the train and components is commonly unknown. Planned jobs during a scheduled maintenance activity mainly aim at preventive maintenance to avoid unexpected breakdowns and there exists an extensively developed maintenance schedule for this type of maintenance. Therefore, it is well known how much time it will take and which components are required. For unplanned jobs there is less insight in the time necessary and the components required to perform the maintenance tasks. This may lead to late releases of trains to NedTrain’s customer, as the planned end date may be exceeded. It is then also difficult for NedTrain to reschedule an extra visit to a maintenance depot for the unfinished job(s), because the railway tracks are intensively used. The unplanned maintenance jobs during scheduled arrivals also have impact on the inventory control, because of the unknown magnitude of the unplanned demand for spare parts. This can result in either a shortage or in an overstock of spare parts, which both have negative effects. A shortage of spare parts might lead to late deliveries, where overstock may involve extra costs due to storage costs or wear down.

1.2.2 Unplanned Maintenance
Besides planned maintenance arrivals, it can also be necessary for a train to visit a maintenance depot unscheduled. These unplanned visits are called extra arrivals, “Extra Binnenkomst” (EBK). This occurs when a train breaks down between maintenance services and the repair is urgent enough and cannot wait until the next visit to the maintenance depot. Maintenance jobs for this unexpected arrival may be planned to some extent, as it is known which critical component causing the failure during operation, has to be replaced. However, it might also be the case that other work is revealed during an EBK, resulting in unplanned jobs during an unplanned arrival. This type of maintenance can be viewed as corrective maintenance. Another phenomenon that may be classified as corrective maintenance is a stranded train, “Gestrande Treinen” (GT). This occurs when the degree of a failure or malfunction is that extreme, that the train has to stop for ten minutes or more. This leads to long delays and in the most inordinate cases, travellers have to be evacuated.

1.3 Condition Based Maintenance
As modern technology developed rapidly, products became more complex and higher quality and reliability levels were required, such that preventive maintenance techniques became more and more costly. For this reason, more efficient approaches were implemented, such as condition based maintenance (Jardine et al., 2006). With condition based maintenance (CBM), the condition of systems or parts is monitored continuously or periodically and is used to determine a dynamic preventive maintenance schedule (Peng et al. 2010). In contrast to usage based maintenance, with CBM the preventive action is only taken when it is believed that an incipient failure has been detected (Tsang, 1995). Using CBM, the proportion of necessary corrective maintenance tasks can be reduced and also less predetermined preventive maintenance tasks are required, since the focus will lie more on the condition of the system instead of performing maintenance at fixed time intervals.
According to Rao (1996), CBM can be defined as follows: “Condition based maintenance provides assessment of machinery condition, based on collecting, collating and intelligently interpreting machine data with a view to provide lead-time and required maintenance prior to failure.” It has proved to minimize the cost of maintenance, improve operational safety and reduce the quantity of in-service machine failure. This corresponds to the objective of a CBM, which is minimizing the total cost of inspection and repairs by collecting and interpreting data related to the operating condition of an asset (Knapp & Wang, 1992). Tsang (1995) states that CBM can be an appropriate approach when failure prevention is not feasible or it is not (yet) known how this can be achieved (e.g. events leading to a failure occur in a predominantly manner). It might also be a good strategy if the measureable parameters that correlate with the failure have been identified or it is at least possible to identify values of those parameters when action should be taken before full failure occurs. Grall et al. (2002) agree by stating that if the deterioration of a system can be measured directly or indirectly by parameters that are strongly related to the state of the system and a system is subject to failure if deterioration reaches a certain threshold level, it is appropriate to base maintenance decisions on a condition-based maintenance policy, instead of the age of the system.

Generally, a CBM program consist of 3 key steps: a data acquisition step (information collection), a data processing step (information handling) and a maintenance decision step (decision making). These are further discussed in the following sections.

1.3.1 Data Acquisition
Data acquisition is the process of collecting and storing relevant data for systems or components that are subject of investigation and for which a CBM program is implemented. Data collected in a CBM program can be divided into two groups: Event data and condition monitoring data. Event data contains the information of what events have occurred and often also what actions were taken after the event. Condition monitoring data is related to the health condition or state of the physical asset (Jardin et al., 2006).

1.3.2 Data processing
According to Jardin et al. (2006) data processing consists of data cleaning and data analysis. To analyse data, a number of tools, models and algorithms can be used depending on the types of data that are collected for CBM.

1.3.3 Maintenance Decision Making
The last step of a CBM program is maintenance decision making. It is crucial that there is sufficient and efficient decision support for maintenance personnel’s decisions to take maintenance actions. There are two main categories in techniques for maintenance decision support in a CBM program, known as diagnostics and prognostics. Diagnostics concerns with the detection, isolation and identification when it occurs, while prognostics deal with fault prediction before it occurs, by determining whether a fault is impending and estimate how soon and how likely a fault will occur. By definition, prognostics are superior to diagnostics, since prognostics can prevent faults or failures or at least be ready for problems in terms of spare parts and human resources. Diagnostics can be a complementary tool for providing maintenance decision support in case of unsuccessful prediction (Jardine et al., 2006). Niu et al. (2010) also distinguish between two types of CBM. The first is maintenance that is based on the current condition of the systems or components. The other is maintenance based on forecasting of the remaining equipment life. The first type belongs to the
technique for maintenance decision support using diagnostics, as it uses continuous collection of condition monitoring data to determine the maintenance schedule. The second type uses forecasting, as the maintenance schedule is based on the actual stress loading with which the remaining equipment life is forecasted and can therefore be classified as prognostics.

Diagnostics of machine faults are often used for pattern recognition. This is a procedure where information obtained in the measurement space and/or features in the feature space are mapped to machine faults in the fault space (Jardine et al., 2006). In other words, diagnostics involves identifying and quantifying damage that has already occurred (Sikorska et al., 2011). When certain limit values are exceeded, a fault condition is recognized and maintenance can be performed (Mann et al., 1995). A complete diagnostics program detects, isolates and identifies a fault or failure. Expected outcomes of a robust and reliable diagnostic program are alarms notifying the user of the systems faults and often also their severity. When applying prognostics, automated methods are not only used to detect and diagnose failures, but also to analyse and predict the degradation of a system. This is may be used to calculate the remaining useful life (RUL) in acceptable operating state before failure or unacceptable degradation of performance occurs (Peng et al., 2010). There is a strong relationship between diagnostics and prognostics, since prognostics are based on diagnostics.

1.4 Real Time Monitoring (RTM) Project
NedTrain is aiming to move towards above described condition based maintenance. Since reliability and availability are of paramount importance, getting more insight in the unscheduled maintenance tasks is receiving high priority. In order get more insight in unscheduled arrivals and unscheduled jobs, a project called Real Time Monitoring (RTM) is introduced. One of the ultimate goals of this project is to make unplanned maintenance more plannable, by predicting unforeseen maintenance activities or failures. This project aims to base the maintenance actions more on the condition of the system, by means of monitoring the state of a train or certain systems of the train in real-time using log data. Based on this condition, information could be obtained on future failures or breakdowns.

If a prediction can be made about when trains will breakdown or need additional maintenance, this maintenance can be planned better. Especially in the tight schedules at the maintenance depots this will lead to more accurate planning, which should result in better availability of the trains. If an unexpected arrival occurs, other trains might be delivered back too late to the customer and of course the train itself is also not available for operation. A better prediction of unexpected arrivals and unplanned jobs may therefore lead to better service levels with regards to the availability. Also in terms of stock management it might be beneficial to have better knowledge on when a failure might occur, such that the stock of spare parts can be better adjusted to the actual need. This may decrease overstock and understock, since more insight is available in the actual demand for spare parts.

In 2014 NedTrain started the RTM project and involved Frankfurt Consulting Engineers GmbH (FCE) as a collaboration partner. FCE is an independent engineering consulting company specialized in the development of mathematical algorithms and programs, offering its services to medium and large industrial corporations. It has a broad experience in railway companies, like Deutsche Bahn and Schweizerische Bundesbahnen, but also with other industries, like the energy sector (FCE, 2015).
NedTrain is in the possession of many gigabytes of data from sensors in the train, which were used in a first pilot to investigate the predictive opportunities of this data for CBM purposes. The cooperation is beneficial to both stakeholders. NedTrain can gain insights in how its data can help to predict if maintenance is required, while FCE has the possibility to test its algorithms and methods using real life data. (Peters & Poot-Geertman, 2014)

In total 54 double-decker-InterCity (IC) trains are tooled with a communication system that can send real time sensor data from the central computer to the maintenance organization. In the first pilot, 6 out of the 54 intercity trains, also known as VIRM’s (Verlengd InterRegio Materieel), were considered with regard to the sensor data of the traction system. Such a traction system converts electronic energy into mechanical energy and has the main functions to bring up the speed, to keep up the speed and to assist during braking. A dataset with a timespan of almost 8 months, measuring 11 signals of the traction system, was used during analysis. By means of sensor data of the traction system, FCE aimed to find anomalies in the behaviour of this system. An anomaly is detected if behaviour of signals is abnormal, compared to what one would expect on similar variables. In the method applied by FCE a certain signal is estimated, based on the values of other signals at the same point in time. If the realized value deviates too much from its expected value based on other signals at that point in time, a warning is generated. The goal was to predict the failures or malfunctions, for which diagnostic event data was used, consisting of error codes generated by the train. A primary goal of NedTrain with this pilot was to get a better understanding of the behaviour of the traction system. Ultimately, operational failures should be predicted based on these abnormalities. During the analysis of the pilot, it became clear that statistical correlation of signals were to a large extent consistent with what one would expect from a physical perspective. Based on the signal data of the traction system, the algorithm used by FCE produced warnings in the days before a diagnosis event took place. This could mean that it is possible to foresee failures of the traction system, based on the signal data. For this pilot the signal data of the traction system was used, but ultimately other systems may be included. The overall planning and the corresponding goals of the RTM project is to define (what is abnormal behaviour) in 2014, explain (which abnormal behaviour leads to malfunctions or failures) in 2014 and 2015 and ultimately to predict (how to predict a malfunction or failure using pattern recognition) in 2016 (Peters & Poot-Geertman, 2014).

An additional step in the process of RTM, was to see if it is also possible to use the above described diagnostic events to predict other events. Instead of predicting these diagnostic events based on the signal data, it might also reveal some prediction possibilities itself to other, more critical, events. In the first place, these diagnostic events may be used to predict critical events in the form of unexpected arrivals at the maintenance depot or stranded trains. Another possibility is that critical events (in the form of severe diagnostic fault events) can be predicted by other diagnostic fault events. This approach is the focus of the current research, which is conducted in cooperation with Frankfurt Consulting Engineers GmbH (FCE).

Besides the cooperation with FCE, NedTrain also organized a contest between three different companies in 2014 with the objective to find out to what extend patterns of diagnostic notifications can predict failure events. At the end of 2014, the contest terminated and the analyses of the three external parties did not result in any strong predictions. The used methods in this contest, together with their results and the analysis of these results can be found in the study of Weerts (2015).
2 Problem Statement

As described in the previous chapter, the focus of this research is on the prediction of events based on other events. The event that is predicted is called a critical event and the events that are used to make this prediction are called diagnostic events in the remainder of the report. In the case of NedTrain, critical events can consist of maintenance events like the described extra arrivals and stranded trains or severe diagnostic errors. The diagnostic events that serve as input are in the form of diagnostic fault messages logged by the trains. In terms of CBM, the condition is monitored based on the occurrences of diagnostic fault codes and are used as diagnostics. By using pattern recognition it is determined if in the nearby future a failure, malfunction or breakdown will occur (prognostics).

This chapter starts with a brief description of the prediction method called similarity sampling (Section 2.1), followed by the introduction of the concept of the distance metric used in this prediction method (Section 2.2), which is the main topic during this research. Based upon these two sections, the application of the distance metric in the prediction model is explained in Section 2.2. The objective is to optimize this metric and evaluate its effects, which is described in Section 2.4 in the form of a research question and corresponding sub questions. The research approach and the links between the research (sub) questions are described in Section 2.5.

2.1 Similarity Sampling

The focus of this research is on the forecasting technique called similarity sampling, which can be used to predict critical failure events based on diagnostic events. This prediction method is developed by FCE and incorporated in the Event Simulator Tool. This tool contains the last two steps of a CBM program: data processing, and decision support (in terms of warnings for failure in upcoming period). In this section, the similarity sampling prediction method is described globally, as a preparation for the research objective presented in Section 2.4.

In order to describe this forecasting method, a timeline should be considered in which all the occurrences of events are stated. For a certain moving time interval the numbers of occurrences of relevant diagnostic events are counted, resulting in a set of state vectors. Stored state vectors and new state vectors are respectively called patterns and observations, but are actually the same type of state vectors. A prediction is based upon the composition of an observation. Such an observation is compared to all the stored patterns and based on this comparison it is forecasted whether or not a critical event is likely to occur in the upcoming period. The goal in similarity sampling is to find stored patterns similar, called neighbors, to the observation. For each of the stored state vectors it is known if and when a critical event will occur in the upcoming period. The time until a critical event is called the forward time. Based upon the neighbors of the observation an empirical forward time distribution can be established for the probability if and when a critical event will occur for the observation. In order to find similar patterns to the observations in the so called training set, which contains all the stored patterns, the distances between an observation and all the patterns are calculated and the ones with the smallest distances are taken into account in the neighborhood and thus in the forecast of a critical event. Patterns within the neighborhood are weighted, such that the most similar patterns receive more weight in the forecast of a critical event. By also using stored patterns as observations, such that also for these state vectors the actual subsequent occurrences of critical events are known, it is possible to measure the forecasting accuracy by comparing the
forecasted critical events with the actual critical events. The set of patterns are called the training set and the set of observations are called the monitoring or validation set. The training set is thus used to learn about the patterns and their corresponding forward time to a critical event and the monitoring set is used to validate observations and measure the accuracy of the predictions.

In Figure 2 the basic principles of the similarity sampling technique are shown. It shows the occurrences of three diagnostic events A, B and C that are used to predict the critical event E over time. Here, one new observation is compared to historical patterns. Based on the most similar patterns in the training set (neighborhood), it is forecasted whether or not a critical event is likely to occur subsequent to this observation, based on the occurrence of critical events of patterns in the neighborhood. In Chapter 3 this forecasting technique is explained more mathematically and in more detail.

2.2 Distance Metric

An important and crucial step in the prediction method is the determination of the neighbourhood, since the composition of patterns in the neighbourhood eventually determines the empirical distribution for the occurrence of a critical event over time of an observation. In turn, the neighbourhood is influenced by the distance metric, e.g. how the distance is calculated between an observation in the monitoring set and all the other patterns in the training set.

A distance metric can be defined as the mapping of two vectors $x$ and $y$ onto a non-negative real number. The distance between two state vectors can be determined with a distance metric that maps vectors $x$ and $y$ on a non-negative real line

$$d(x, y) = \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$$

The most well-known distance metric is the Euclidean distance or Euclidean metric. For Euclidean vectors $x$ and $y$ in an $n$-dimensional space, the distance is (Kolman & Hill, 2008)

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$
Where $x_i$ and $y_i$ represent the $i$th dimension of vector $x$ and $y$, respectively. By defining the distance metric in Euclidean space as in Equation (2.2-2), all dimensions $i$ in the vector have the same weight in calculating the distance between vector $x$ and $y$. In order to give increasingly greater distance to points that are further apart, the standard Euclidean distance can be squared, such that the distance of Euclidean vectors $x$ and $y$ in a $n$-dimensional space becomes

$$d(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2.$$  

(2.2-3)

However, this metric in Equation (2.2-3) is actually not a real distance since it does not adhere to the triangle inequality, but it is frequently used in optimization problems in which the distances only have to be compared (Bellet, Harbrard & Sebban, 2013). Another method to vary the weight in the Euclidean distance metric is to introduce a weight term $a_i$ for each dimension $i \in n$, such that each dimension is weighted accordingly and the distance metric becomes (Pelillo, 2013)

$$d_A(a, x, y) = \sqrt{\sum_{i=1}^{n} a_i(x_i - y_i)^2}.$$  

(2.2-4)

The weights $a_i$ sum up to $n$, the dimension of the vector and can be normalized by dividing the weights by $n$. By using equal weights $a_i$ for each $i$th dimension in equation (2.2-4), the distance metric equals the Euclidean distance metric presented in Equation (2.2-2).

### 2.3 Application of the distance metric

Initially, the Event Simulator Tool uses the Euclidean distance metric to compare the observations with the patterns in the training set. However, it might be beneficial to give more weight to certain dimensions than to others, as they may have a more predictive value for a critical event to occur. Using a weighted distance metric has effect on three aspects in the similarity sampling method. First, as particular dimensions (e.g. diagnostic events) are becoming more important than others, patterns that did not belong to the neighbourhood with equal weights, might be part of the neighbourhood if different weights are applied. Secondly, the distance metric also affects the weight function that gives more similar patterns in the neighbourhood more weight in the forecast of a critical event, because the weights are based on this distance metric between patterns in the neighbourhood and the new observation. This weight function (see Equation [3.1-1]) gives higher weight to more similar patterns, e.g. smaller distances, by using the distance metric. And thirdly, since the neighbourhood and the weight function are affected by a weighted distance metric, also the establishment of the empirical distribution for the occurrence of a critical event is influenced.
Figure 3 shows the influence on the determination of the neighbourhood by adding weight to the distance metric, graphically. The blue dots represent the state vector of 2-dimensional patterns and the red dot represents a new observation. By adding weights to the different dimensions in the pattern, the neighbourhood changes from a round to a more elliptical shape, such that patterns that first did belong to the neighbourhood now do not and vice versa.

2.4 Research Objective

By means of a training set, containing stored patterns with a known forward time to a critical event and a monitoring set, consisting of observations for which critical events are forecasted and can be compared to the actual occurrence of these events, the predication accuracy of the similarity sampling can be quantified. This can be done by an objective function that measures the forecasting error or by a warning-based accuracy measure. The goal is to optimize, or at least improve, the prediction accuracy of the similarity sampling method. The question is if this is possible by the extension with a weighted distance metric. This leads to the following research question:

*Can the prediction accuracy of the predictive approach be improved by extending the distance metric with weights, compared to the Euclidean distance metric?*

In order to answer the research question, a set of sub questions is established which are answered during the remainder of this research and are grouped into four categories. These categories correspond to the steps in the research approach which are explained in Section 2.5 and are shown in Figure 4.

1. Analysis and Diagnosis:
   A. *How does the current predictive approach forecast critical events based on other events?*
   B. *What is the role of the distance metric in the current predictive approach?*
   C. *Which parameters influence the prediction accuracy in the predictive approach?*

2. Plan of Action:
   A. *Which optimization heuristics exists in literature that can deal with the optimization problem?*
   B. *Which optimization heuristics should be used to deal with the optimization problem?*
   C. *How can this optimization heuristic be applied to the current optimization problem?*

3. Evaluation:
   A. *What are the effects of other parameters on the prediction accuracy and to what extent?*
   B. *What are the effects of the optimization heuristic on the prediction accuracy?*
   C. *Which insights can be obtained from the effects of other parameters and the optimization heuristic?*
   D. *Using the gathered insights, how does the model perform on the NedTrain data?*

2.5 Research Approach

The established research sub questions, stated in Section 2.4, are answered in the remainder of this research. In order to answer these questions a research approach is developed based on a single iteration of the regulative cycle of van Strien (1997) to solve a business problem-solving project. This research design is depicted in Figure 4.
Based on the problem definition that is described in the current and previous chapter, the next steps can be performed. The first step consists of a more extended analysis and diagnosis of the current model, resulting in a detailed model description. This part is presented in Chapter 3 and gives answer to the sub questions 1A, 1B and 1C.

Based on this extended model description and the problem definition a plan of action for the optimization problem is established. This is covered in Chapter 4 and consists of the exploration and the selection of optimization heuristics. It corresponds to the sub questions 2A and 2B. The application of the selected optimization heuristic (simulated annealing) is described in Chapter 5 and deals with the sub question 2C. In order to conduct the research, FCE has provided its Event Simulator Tool which can be extended to construct the weighted distance metric, such that the prediction accuracy can be quantified and optimized. The actual implementation of the optimization heuristic in the tool, based on Chapter 5 is described in Appendix D.

The evaluation phase is covered in Chapter 6, which answers the sub question 4A and 4B and shortly addresses sub question 4D about the application of the Event Simulator on NedTrain data. First, the effects of other parameters in the similarity sampling method are analysed. These other parameters are also taking into account during the research to analyze how they influence the forecasting accuracy and how they are related to the optimization of a weighted distance metric. Thereafter, the effects of the weighted distance metric are quantified and the simulated annealing algorithm is evaluated in terms of its ability to find optimal weights for the distance metric. As a benchmark, the use of a weighted distance metric can be compared with the Euclidean distance metric with all equal weights. Using this comparison the potential improvement can be measured, in order to quantify and analyze the effects of the used weighted distance metric. For both evaluations the test case is used. The insights obtained during the evaluation of the test case (sub question 4C) are translated to guidelines which can be used to other cases and are presented in Appendix H. Finally, the conclusions, limitations and recommendations of this research are discussed in Chapter 7.

It was planned to apply the event simulator on data of NedTrain, based on insights gained from the test case. However, it became clear during the research that the data provided by NedTrain did not have enough predictive value to use it in the Event Simulator. Therefore, the gained insights obtained from the test case, could not be applied to predict a critical event in the NedTrain data. This is discussed briefly in Chapter 6.
3 Prediction model: FCE Event Simulator

In this chapter the prediction method that is used to predict critical events based on diagnostic events is explained in more detail (Section 3.1). In Section 3.2, the role of the distance metric is addressed mathematically, referring back to the optimization problem described in the research objective in Section 2.4. In Section 3.3, other parameters that might influence the prediction accuracy are discussed, together with the developments with regard to their optimization during this research. All topics described in Section 3.3 are of influence on the prediction accuracy and the distance metric and are therefore taken into account. Furthermore, the possible choices for the selection of input data are elaborated together with their relationships to the prediction accuracy.

3.1 Description of the initial model

To describe the initial model, the notations that are used are explained throughout the section, but an overview of all notations and definitions can be found in Appendix A. In Appendix B the flow chart of the prediction method is presented as it was at the beginning of this research. Here, four subsections are distinguished: pre-processing and set-up (Section 3.1.1), training set (Section 3.1.2), monitoring set (Section 3.1.3) and optimization routine (Section 3.1.4), which together lead to the predictive model. This predictive model is incorporated into the Event Simulator Tool. This section describes these sub processes within the prediction method and covers the basic principles of the similarity sampling method.

![Figure 5 Training set, monitoring set and the moving time window (Jaroszewski, 2014)](image)

3.1.1 Pre-processing and set-up

In the sub processes in this part of the model, the initial set up for the predictive approach is determined. This includes the separation of the data into a training set $T$ and a validation set $M$. As described the training set contains the occurrences of diagnostic fault codes over time and the same holds for the monitoring set, also known as the validation set. By choosing a larger period for the training set, more information can be learned about the data and the corresponding patterns. However, the period used for monitoring should also be large enough in order to validate the prediction accuracy properly. By using a moving embedding time window $l$, state vectors $x$ can be created by counting the occurrences per diagnostic event. The length of this time window is a parameter that in the initial model is chosen based on the nature of the dataset or by experimentation. In order to distinguish patterns from the training set and observations from the
monitoring set, the state vectors are indexed in the remainder of the research with \( u \) and \( t \), respectively. For the training set, each pattern \( x_u \) has a known forward time \( f_{E,u} \) to the critical event \( E \). For the validation set the (actual) forward times \( f_{E,t} \) of observations \( x_t \) are also known, but are used as comparison with the forecasted forward time \( \hat{f}_{E,t} \) based on the empirical forward time distribution, together with the CDF threshold \( c \). The training and validation set, together with the moving time window are graphically shown in Figure 5. Another parameter is the step size \( s \), which represents the time interval on which state vectors are created, with an hour as smallest the size. Since the step size values are low compared to the size of the embedding window, subsequent state vectors are overlapping. In the initial event simulator tool, the length of the embedding window, the step size and the CDF threshold can be chosen during the setup. Before all the setups can be done, the input data needs to be pre-processed to an appropriate format (csv), such that it can be used in the Event Simulator Tool.

### 3.1.2 Training Set

As described, the training set contains patterns, which are past observations in the form of state vectors \( x_u \) consisting of the number of occurrences during an embedding window, taken into account the step size and each with a known corresponding forward time (time to the critical event). In order to determine the state space, e.g. the length of the state vector, a cross validation function is used. This function is used to select the most relevant diagnostic events that are used in the prediction model and are called the predecessors in the remainder of the research. Although this determination is out of scope and assumed to be known, a short description is given. In Figure 6 an example of cross correlation functions is provided of a critical event \( E \) ("0624") and all other diagnostic events. If the correlation function \( r_{EF} \) has a high mass between \( \tau = 24 \) and \( \tau = 168 \) (the forecast horizon \( h \)), this indicates that the occurrence of diagnostic event \( F \) is an indicator that critical event \( E \) occurs in the next 24 to 168 hours. Events for which the correlation function has a high mass are taken into account as relevant indicators and are used in the state space. Eventually, the training period will consist of patterns \( x_u \) with dimensions \( x_{u,1}, \ldots, x_{u,n} \) representing relevant diagnostic events, where \( n \) is the length of the state vector, with usually \( n < 20 \). For each of the patterns \( x_u \) in the training set, the forward time of a critical event \( E \) is known and represented by \( f_{E,u} \).

![Cross Correlation Function of a Critical Event E and All Other Events F](Jaroszewski, 2014)

### 3.1.3 Monitoring Set

As for the training set, the monitoring set also contains state vectors, which are here denoted by \( x_t \). These state vectors are treated as observations and are compared to the patterns in the training set.
This monitoring set is used to validate the predictive method, since for these observations it is also known if and when a critical event \( E \) actually did occur. Simply described, in this subsection of the prediction method, actual observations \( x_t \) in the monitoring set \( M \) are compared to patterns \( x_{u_t} \) in the training set \( T \). Based on the most similar patterns in the training set for a particular observation, it is determined what the probability is that a critical event \( E \) will occur within the forecast horizon.

Now two questions remain: 1) Which patterns in the training set are considered to be similar patterns of an observation in the monitoring set and 2) How is this set of similar patterns used in order to establish an empirical distribution (or empirical forward time) for the observation.

In order to find the most similar patterns to an actual observation, the distances between the current observation and the patterns in the training set are calculated. Initially, the Euclidean distance metric is used as stated in Equation (2.2-1). Pattern \( x_{u_t} \in T \) is highly similar to an observation \( x_t \in M \) if the distance metric between \( x_{u_t} \) and \( x_t \) is very small. So, a smaller distance means higher similarity. Based on these distances a neighbourhood \( N(x_t) \) for an observation \( x_t \) can be defined, containing the most similar patterns found in the training period. After \( N(x_t) \) is established, consisting of similar patterns \( x_{u_t} \in N(x_t) \subseteq T \), a function to give weights to these similar patterns of an observation \( x_t \) can be formulated as

\[
w_{x_t} = w(d(x_t, x_{u_t})) = \frac{e^{-\frac{1}{2}d(x_t, x_{u_t})^2/\sigma^2}}{\sum_{x_{u_t} \in N(x_t)} e^{-\frac{1}{2}d(x_t, x_{u_t})^2/\sigma^2}}.
\]  

(3.1-1)

Here, \( \sigma^2 \) represents the variance of the distances of the observation with the patterns in the neighborhood \( V((d(x_t, x_{u_t})) \), where \( x_{u_t} \in N(x_t) \). With this function a weight vector is created that gives weight to each of the patterns in the neighbourhood. This weight function is used in the determination of the empirical forward time distribution stated in Equations (3.1-2) and (3.1-4). All patterns that are not similar enough will get a weight of zero and in above weight function only the patterns in the neighbourhood are used. So, \( x_{u_t} \in N(x_t) \) if \( d(x_t, x_{u_t}) < Z \), where \( Z \) is a distance threshold which is established by the Event Simulator tool. How this distance threshold is exactly determined is left out of scope in this research. Since every pattern \( x_{u_t} \in N(x_t) \) has a known forward time \( f_{E,u_t} \), an empirical forward time distribution for the occurrence of critical event \( E \) can be generated, applying above weight function to the patterns in the neighbourhood. This can be done for each observation \( x_t \in M \). This procedure is called forward similarity sampling and is graphically shown in Figure 7.

This figure represents the state space \( \Omega \) in 2 dimensions with a point cloud (in blue) that represents the patterns over time. Subsequent state vectors are connected by lines and can be seen as the time line. The green and the red dot are observations with their corresponding neighbourhood of patterns and the stars represent critical events. Since the critical events occur quickly after patterns in the neighbourhood of the red observation this observation has a fast increasing empirical forward time distribution for the critical event. Here a warning is given since the CDF threshold is reached within the forecast horizon. For the green observation no critical event occurs closely after patterns in its neighbourhood resulting in a flat empirical forward time distribution in the forecast horizon, such that no warning is given.
So, given the length of the forecast horizon $h$ and the CDF threshold $c$ and based on the empirical forward time distribution, a prediction can be made whether or not a critical event is likely to occur. If the cumulative distribution function is higher than a certain threshold during the forecast horizon, a warning is created that with a high probability the critical event will occur. The probability mass function (PMF) of the empirical forward time distribution for a given $x_t$ is given by

$$P(F_{E,t} = f) = \sum_{x_{u'} \in N(x_t)} w(d(x_t, x_{u'})) I(f_{E,u'} = f). \quad (3.1-2)$$

Here, $F_{E,t}$ is the random variable for the forward time, $f$ is a time variable and $I(f_{E,u'} = f)$ represents an indicator function, which is defined as

$$I(a = b) = \begin{cases} 1 & \text{if } b = a \\ 0 & \text{else} \end{cases}. \quad (3.1-3)$$

This PMF can be converted to a cumulative distribution function (CDF) for the empirical forward time distribution for a given $x_t$, as illustrated in Figure 7, by

$$P(F_{E,t} \leq f) = \sum_{x_{u'} \in N(x_t)} w(d(x_t, x_{u'})) I(f_{E,u'} \leq f), \quad (3.1-4)$$

where the indicator function $I(f_{E,u'} \leq f)$ is defined as

$$I(a \leq b) = \begin{cases} 1 & \text{if } a \leq b \\ 0 & \text{else} \end{cases}. \quad (3.1-5)$$

### 3.1.4 Optimization Routine

In order to optimize the prediction method, the initial routine is to play with the distance metric and the weight function in order to minimize the prediction accuracy, measured for example with the expected least square forecast error. This error is determined by comparing the actual forward time of an observation, to the expectation of the empirical forward time distribution. Also other accuracy measures, like warning success rate and event miss rate, are calculated which can be minimized in
the same manner, by varying the distance metric and weight function and see what the effects are. So, the prediction accuracy can be influenced by changing the distance metric and by changing the weight function. Also the length of the embedding window, the step size and the CDF threshold can be of influence and can therefore be varied in order to decrease the forecasting error or to optimize another accuracy measure.

3.2 Optimization of the weighted distance metric

Now that the prediction model is explained in more detail and in more mathematical terms, the role of the weighted distance metric and the optimization of its weights can also be stated in a more concrete way. As mentioned before, the distance metric is used to determine which patterns are taken into account in the neighbourhood and how much they will contribute in the empirical forward time distribution. Instead of using the Euclidean distance presented in Equation (2.2-1), the weighted distance metric of Equation (2.2-3) is used in the weight function for the neighbourhood in Equation (3.1-1) and thus also in the determination of the empirical forward time distribution in Equations (3.1-2) and (3.1-4).

As stated, the goal of the predictive approach is to give a most accurate prediction as possible for the occurrence of a critical event $E$. In order to measure the accuracy of the prediction model, the forecast based on the empirical forward time distributions of the actual observations $x_t \in M$ is compared to the actual situation. By taking the example of the expected squared forecasting error, the following accuracy measure can be established

$$V_A = \sum_{x_t \in M} (\hat{f}_{E,t} - f_{E,t})^2.$$  \hspace{1cm} (3.2-1)

Here, the actual forward time $f_{E,t}$ is compared with the predicted forward time $\hat{f}_{E,t}$, for which the expectation of the empirical forward time distribution is used. This expectation is defined as

$$\hat{f}_{E,t} = \mathbb{E}[F_{E,t}] = \sum_{f \in G_{E,t}} f P(F_{E,t} = f),$$  \hspace{1cm} (3.2-2)

with

$$G_{E,t} = \{ f_{E,u'} : x_{u'} \in N(x_t) \}.$$  \hspace{1cm} (3.2-3)

To give the most accurate prediction, the defined expected squared forecasting error in Equation (3.2-1) should be as small as possible. The smaller this forecasting error is, the more precise the method can predict future disruptions. A set of weights $a_i \in \mathbb{R}^{n^2}$ should be found that minimizes the expected squared forecasting error $V_A$. This measure can be replaced or any other suitable objective function that represents the prediction accuracy. The use of objective functions is discussed in depth in Section 5.2.

3.3 Other influencing parameters

Before the optimization approach for the distance metric is discussed, the place of this optimization within the FCE event simulator is described based on Figure 8. As the tool was under development during the research, not only the distance metric is optimized but also other parts were further developed by FCE in order to get a better forecast accuracy, compared to the initial optimization routine as described in Section 3.1. In the flowchart of Figure 8 it is shown what steps are being
performed in the prediction model and in which sequence. These other steps are of importance, as other parameters can have impact on the prediction accuracy and may influence the effect of the distance metric as well.

The other parameters of importance are the length of the embedding window \( h \), the step size \( s \) and the CDF threshold \( c \). During developments of the tool it became possible to select the option to perform optimization on these parameters. Without the optimization a fixed embedding window, step size and CDF threshold can be chosen. With optimization, a lower and upper value (integer) can be set for these parameters, together with a grid size, for which the parameters are optimized. The method is then executed for a number of values equal to the grid size equally divided over the grid between the lower and upper bound. All three parameters can be varied simultaneously, so that the number of times that the method is performed is equal to the multiplication of the grid sizes. It is chosen to incorporate the optimization of the weighted distance metric together with the other optimizations. However, it is possible to fix the other parameters by choosing a grid size of 1 and the same lower and upper bound, such that only the effect of the weighted distance metric can be analysed. By walking through the steps in the prediction model presented in Figure 8, the role of each parameter is discussed with regard to their influence on the prediction accuracy and the distance metric.

![Figure 8 Flowchart of the FCE Event Simulator (with optimization)](image)

3.3.1 Embedding Window and Step Size

The embedding window and the step size represent the time period in which occurrences are counted and on which interval this is done, respectively. This results in a set of date-timestamps with a corresponding backward-pattern and a forward-pattern. Based on these patterns the predecessors can be determined. The determination of the predecessors is part of another research and will not be in scope, but assumed to be given. The predecessors are not influenced by the distance metric itself, but only by the embedding window and the step size, as can be seen in Figure 8. However, the other way around, the distance metric is dependent on the established predecessors. Each combination results in a set of relevant events and can therefore be different when other combinations are used. Also the number of relevant diagnostic events corresponding with the length of the state vector is based on the combination of the embedding window and step size. Furthermore, the chosen combination of the embedding window and step size itself is of influence on the prediction accuracy. When the optimization of the distance metric is incorporated in the prediction method this means that for each combination of the embedding window and step size the distance metric should be optimized, separately. Each combination then has an optimal distance metric with a corresponding prediction accuracy, based on the used target function.
3.3.2 CDF Threshold
Based on the empirical forward time distribution and a CDF threshold, it is determined whether or not a warning is given for a particular pattern. A warning is given when the empirical forward time distribution reaches this threshold within the forecast horizon. Where the step size and the embedding window have influence outside (e.g. before) the similarity sampling process, the CDF threshold has impact within the similarity sampling process itself. When this CDF threshold is optimized by means of a grid, it is checked for which CDF threshold the objective function has the highest value for the combination of the step size and the embedding window. When the optimization of the distance metric is incorporated, this means that for each set of weights that is assessed for a certain combination of steps size and embedding window, it is checked which CDF threshold result in the best prediction accuracy based on the used objective function. It should be noted that the CDF threshold is only of importance if warning-based objective functions are used to measure the prediction accuracy as it plays no role in a forecasting-error based measure where forward times are compared.

3.3.3 Selection of Input parameters
The input of the prediction model is a csv file containing the diagnostic fault codes with their corresponding timestamp, their diagnostic fault code ID and type of diagnostic fault code. Furthermore, some more detailed information is available like the vehicle ID, the type of vehicle and the subsystem at which the event took place. Based on this information, input parameters can be selected that might also impact the forecast accuracy. These input parameters are discussed in the upcoming sections and consist of the selection of the vehicle (Section 3.3.3.1), the subsystem (Section 3.3.3.2), the training and monitoring period (Section 3.3.3.3) and the forecast horizon (Section 3.3.3.4).

3.3.3.1 Vehicle Selection
By selecting one vehicle, only events of that particular vehicle are taken into account during the prediction, so the prediction of the critical events of a certain vehicle is only based on diagnostic events that occurred on that same vehicle. When multiple vehicles are selected all the diagnostic events are merged together. It might also be desirable to use a part of the vehicles only in the training set to predict critical events of other vehicles in the monitoring set. If the period of events is rather small, it might be necessary to include the whole fleet or a large subset of the fleet in order to do predictions. However, it is not certain that different vehicles will have the same pattern of diagnostic events prior to a critical event, due to vehicle dependent circumstances and their failure or maintenance history. If the used fleet or set of vehicles is large enough, the data might be considered to be sufficient enough to demonstrate the prediction methods feasibility, since most types of patterns should be present sufficiently in the data set, however this should be explicitly assumed. If there are explicit differences in types of vehicles (e.g. different technical characteristics), they should be treated separately. For the case of vehicle selection, experimentation is required to see if distinctions need to be made.

3.3.3.2 Subsystem Selection
Besides the selection of vehicles, it is also possible to include or exclude subsystems in the forecasting of critical events. However, there is the possibility that other related subsystems contain predictive value. For subsystems, the safest method is therefore to include all subsystems, as the selection of predecessors is performed within the tool. It can be cross-checked afterwards with
technical experts, if the diagnostic events taken into account are logically related to the critical event that is predicted.

3.3.3.3 Training/Monitoring Period Selection
Besides the selection of the vehicles and the subsystems, the selection of the training period and the monitoring period can also affect the forecasting accuracy. Different training and monitoring periods might give different prediction results, especially with small data sets. The more data available in training set, the more patterns are available as comparison for the observation so the prediction accuracy is increased. More observations in the validation set assure that the predictions are more generalizable. Also seasonal effects should be taken into account, since they can be of great influence on the occurrence of events and corresponding critical events.

3.3.3.4 Forecast Horizon Selection
The forecast horizon is used as the time window after a pattern or observation, to check if a critical event is occurring during this horizon. This parameter can influence the prediction accuracy, but is more an input parameter that should be chosen from a practical point of view. If a forecasting window of 24 hours is chosen, but it takes two days to get a vehicle in a maintenance depot, the warning that a critical event is about to occur, is too late. However, when a very large forecasting horizon is chosen, it is still not quite clear when the critical event will occur exactly during this horizon, without checking the development over time of the empirical forward time distribution. So only a warning has not much value with large forecasting horizons.
4 Literature Review

Optimizing the prediction accuracy by means of applying weights to the distance metric cannot be done analytically and a heuristic approach should be considered. To calculate the prediction accuracy, by means of a certain objective function for one particular set of weights in the distance metric, it is necessary to walk through the whole similarity sampling process, including calculating all distances between all observations in the monitoring set and all patterns in the training set, establishing the neighbors, determining all the empirical forward times and compare the predictions with the actual occurrences of the critical event. For each time the set of weights is changed, the whole similarity sampling process needs to be repeated with the new settings, in order to obtain the objective result of the prediction accuracy. This means that evaluating different weights combinations is very time and memory consuming.

Therefore, a systematic exploration of the search space is needed, together with a good starting point, since it is impossible to check all possible set of weights. Especially for higher dimensional spaces, problems can occur due to the enormous amount of data that has to be explored then (Dekker & Aarts, 1991). In this chapter heuristic approaches are considered in order to find an optimal set of weights for the distance metric. Based on the approaches which are suitable for the optimization problem, a selection is made on which of these approaches suits this problem best.

First the optimization problem of the distance metric is formulated as a global minimization problem in Section 4.1. In Section 4.3, possible heuristics are provided that can deal with the optimization problem, which can overcome potential pitfalls of simple local search techniques as described in Section 4.2. The heuristic that is considered to be most applicable here (simulated annealing), is then further elaborated in Section 4.4, together with the argumentation why this approach was chosen above other techniques.

4.1 Global minimization problem

Approaches that are taken into consideration should be able to deal with a multi-dimensional continuous state space (vector of distance metrics) and a continuous objective function (prediction accuracy) that has to be minimized. This problem can be described as a global optimization problem. It can be formalized as a pair \((X, f)\), where \(X \subset \mathbb{R}^n\) is a bounded set on \(\mathbb{R}\) and \(f : X \rightarrow \mathbb{R}\) an \(n\)-dimensional real-valued function. The problem is to find a point \(i_{\text{min}} \in X\) such that \(f(i_{\text{min}})\) is globally minimal on \(X\). This can be formulated as

\[
\forall i \in X : f(i_{\text{min}}) \leq f(i).
\]

For relatively simple functions, where \(f\) is differentiable and the zero points of the derivative can be computed analytically, there exists satisfactory algorithms. However, with more complicated functions, as is the case in the optimization problem of the distance metric, there only exist numerical solution methods. Most of these methods do not return optimal results, but at best a value close to a global minimum is found. These numerical global optimization methods can be divided into two classes (Zabinsky, 2013):

1. Deterministic methods (algorithms in which no probabilistic information is used) and;
2. Stochastic heuristics (algorithms that use some kind of randomness, typically a random number generator).
The disadvantage of deterministic methods is that they find the global minimum only after an exhaustive search over $X$ and additional assumptions of $f$, or in a lot of cases even not at all. According to Rinnooy Kan & Timmer (1984), stochastic methods, in contrast, can almost all be proven to find a global minimum with an asymptotic convergence guarantee in probability, i.e., these methods are asymptotically successful with probability 1. Furthermore, the computational results of the stochastic methods are in general better than those of the deterministic methods.

Deterministic methods will not be taken into account as exhaustive search through the system space is not desirable in terms of computation time and the focus therefore lies on stochastic heuristics. Stochastic heuristics which are applicable to the optimization problem of the distance metric, almost all belong to the class of local search techniques. In these types of techniques neighbors of a current solution are generated and assessed in order to explore the search space to find the optimal solution.

## 4.2 Local Search Technique

The type of stochastic heuristics that are often used to solve this kind of problems can be found in the class of local search algorithms, which are also known as iterative improvement algorithms. In local search algorithms usually only one single current state has to be remembered, which is then tried to be improved by moving to neighboring states. It is therefore very memory efficient and can often find reasonable solutions in large continuous state spaces. Local search can do very well on pure optimization problems, where all states have an objective value and the goal is to find the state with the maximum or minimum objective function (Johnson, Papadimitrou & Yannakakis, 1988). Other purposes of these kinds of algorithms are to find a solution that satisfies a set of constraints.

Important in local search techniques is to recognize a local minimum and to avoid getting stuck in this local minimum. The simplest local search technique known, on which most other (stochastic) techniques are based, is the hill climbing algorithm, in which any local change that improves the current value of the objective function is accepted and terminated when no local move further improves the objective function. In some optimization problems such a local optimum might be acceptable, but when a lot of local optimums exist the solution might not be good enough. Whether or not a global optimum is found, highly depends on the initial state. Since this algorithm does not look ahead past immediate neighbors, it is not known if more optimal solutions exist. Also so called plateaus or shoulders give problems, since all neighboring states are the same as the current state, such that it get stuck and the algorithm is terminated, without finding a global or even a local optimum. The problems involved with this technique are shown in Figure 9.

![Figure 9 Landscape of state space in local search techniques](image-url)
4.3 Stochastic heuristics in local search

In order to overcome the pitfalls described in Section 4.2, a lot of stochastic heuristics have been developed as improvement on the simple hill climbing search technique. According to Dekker & Aarts (1991) there are two classes of stochastic methods that can overcome the above described difficulties in local search techniques: Two-phase methods and simulated annealing based methods. In the remainder of this section possible stochastic heuristics are briefly discussed. First examples of two-phased methods are being discussed in Section 4.3.1, followed by a more extensive elaboration on taboo search (Section 4.3.2). These latter heuristics, together with simulated annealing have showed to be the best heuristic approaches in terms of the avoidance of getting entrapped in local minima and give a near-optimal solution in continuous global optimization problems (Cvijovic & Klinowski, 1995). Since simulated annealing is used for the optimization of the weights in the distance metric, it is discussed in detail separately in Section 4.4.

According to Press et al. (1996) the simulated annealing approach can be used in optimization problems with continuous multi-dimensional control spaces and can avoid to get stuck in local optimum, it is chosen to use this technique in finding an optimal set of weights for the distance metric in order to find the best prediction accuracy. Furthermore, according to Locatelli (2000) there are many remarkable works in different applications in the field of continuous global optimization. Simulated annealing is thereby capable of finding near-optimal solutions whatever the initial conditions are (Cvijovic & Klinowski, 1995). According to Corana et al. (1987), simulated annealing also has proved to be more reliable than other algorithms, being able in almost all cases to find an optimum, or at least a point very close to it and has the advantage that is not or only slightly dependent on the starting point. For taboo search, in contrary, very few works deal with its application to the global minimization of functions depending on continuous variables (Chelouah & Siarry, 2000). For the reasons of the independency of the starting point and the large number of successful applications in a wide field of different applications in global optimization problems, simulated annealing is chosen over taboo search. Furthermore, it is also taken into account that FCE has a lot of experience using simulated annealing for all kinds of problems. However, taboo search might also be an appropriate heuristic to solve the optimization of the weighted distance metric, as Chelouah & Siarry (2000), claim to successfully adapt the original Taboo Search to continuous optimization problems.

4.3.1 Two phased methods

Two-phased methods are algorithms that search for a global minimum by doing multiple local search procedures. First, a number of starting points from $X$ are randomly sampled, then for each of these points a local minimum is determined. An example of a two-phased method is Pure Random Search (PRS). This technique is the most simple of its form and is also known as blind search. It generates a sequence of independent, identically distributed points in the feasible region $X$, typically according to a uniform sampling distribution, but any other probability distribution can be used. When a stopping criterion is met the best point of the sequence thus far is used as an approximation to the optimal solution (Zabinsky, 2011). A natural extension of the PRS is the Multi-Start method, where local search is applied to every point in the sample drawn from the uniform distribution over $X$. The local minimum with the lowest function value is considered to be the global optimum (Selman & Gomes, 2006). Other examples of this kind of stochastic methods are Controlled Random Search (CRS) and Multi-Level single linkage (Zabinsky, 2013).
4.3.2 Taboo search

Taboo search is an iterative improvement algorithm that uses a limited memory of recent past moves to help in the diversification of search procedure. It effectively guides the search procedure away from getting trapped into local optimum, by means of this diversification (Rajesh, Jayaraman & Kulkarni, 2000). It is commonly used for combinatorial optimization problems, but there are few examples of applications to global optimization problems depending on multiple continuous variables. (Hu, 1992; Cvijovic & Klinowski, 1995; Battiti & Tecchiolli, 1996; Siarry & Berthiau, 1997; Chelouah & Siarry, 2000).

The original taboo search for combinatorial optimization problems developed by Glover (1989) can be roughly described as follows (Rajesh, Jayaraman & Kulkarni, 2000; Siarry & Berthiau, 1997): It starts from an initial solution \( i \), which is randomly selected. From this starting solution a set of neighbours \( i' \) is generated by applying perturbations (e.g. moves), which have been defined beforehand. The objective function to be minimized is evaluated at each generated solution \( i' \) and the best \( i' \) becomes the new current solution even if it is worse than \( i \). By this manner it is possible to escape from local minima of the objective function. Then a new iteration is performed, starting from the new current point.

The most distinctive element of taboo search compared to other local search methods, are the taboos. These are used to prevent cycling when moving away from local optima though non-improving moves. When this situation occurs, it is prevented that the search traces back to where it came from. This is achieved by declaring taboo (e.g. disallowing) moves that reverse the effect of recent moves. Taboos are stored as a short-term memory (the taboo list) and contain a fixed number of last retained points. If a new move result in a point that is in the taboo list, it is prohibited to visit this solution. The taboo list might also be based on extra rules on prohibited solutions. Taboos are sometimes too powerful and therefore so-called aspiration criteria are often implemented. The simplest version of this criterion is to allow a move, even if it is taboo, when it results in a solution with an objective value better than that of the current best-known solution.

Typically, the algorithm is stopped after \( M \) iterations without any improvement of the objective function. Sometimes, a fixed number of iterations is used as stopping criterion or the algorithm is terminated when the objective value reaches a pre-specified threshold value.

4.4 Simulated Annealing

According to Press et al. (1996) simulated annealing (SA) is a generic, probabilistic, heuristic optimization algorithm used to find an approximation to the global optimum of a given function in a large search space. The method of simulated annealing is a technique that has attracted significant attention as suitable for optimization problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema. In simulated annealing a probabilistic mechanism is applied, that enables search procedures to escape from local minima, by accepting worse solutions by means of a Metropolis acceptance criterion.

In the remainder of this section, some background on simulated annealing is given, together with its origin (Section 4.4.1). Subsequently, the relationship between the SA algorithm and the general class of threshold algorithms is explained (Section 4.4.2). Finally in Section 4.4.3 the necessary elements of the so called metropolis procedure are discussed, such that SA can be applied to any other
problem, independent of the purpose. In Chapter 5, the application of the simulated annealing approach for the specific optimization problem of the distance metric is extensively described, based on the four elements of the Metropolis procedure.

4.4.1 Background
The term annealing refers to a technique used in the metallurgy, where metal is being heated and then slowly is cooled in order to increase the size of crystals within the material. In the field of condensed matter physics, annealing is a thermal process for obtaining low-energy states of a solid in a heat batch. This process consists of increasing the heat batch to a maximum temperature at which the solid melts and is then carefully decreased until the particles of the melted solid arrange themselves in the ground (solid) state. During the liquid phase all particles are randomly arranged, while in the ground state the parts are highly structured arranged and the energy of the system is minimal. If the temperature is lowered sufficiently slowly, the solid can reach thermal equilibrium at each temperature (Aarts, Korst & Michiels, 2005). The essence of this process is slow cooling, allowing ample time for redistribution of the atoms as they lose mobility and is essential for ensuring that a low energy state is achieved. The idea of slow cooling is implemented in the algorithm by means of exploring the solution space, using a slow decrease in the probability of accepting worse solutions (Eglese, 1990).

4.4.2 Threshold Algorithm
Simulated annealing belongs to the class of local search algorithms, known as threshold algorithms, which can be used to find solutions to combinatorial minimization or maximization problems. Consider a tuple \((X, f)\) where \(X\) represents a solution set and \(f\) is a so called cost function, which can be seen as a target function or an objective function. Here, each solution set has a corresponding cost value \(f: X \rightarrow \mathbb{R}\). Furthermore, let \(N: X \rightarrow P(X)\) be a neighbourhood function, that defines for each \(i \in X\) a set of neighbouring solutions, \(N(i) \subseteq X\). The algorithm starts with an initial solution \(i_{\text{start}} \in X\) and a neighbouring solution \(i' \in N(i)\) is then generated either randomly or using some pre-defined rule. The goal is to find optimal solution \(i^* \in X\) that minimizes the objective function of all solutions over \(X\) (Aarts, Korst & Michiels, 2005). A threshold algorithm can be described by the pseudo code in Figure 10 (Aarts, Korst & Michiels, 2005; Henderson, Jacobson & Johnson, 2003), for minimization problems. For maximization the inequality sign can be reversed.

begin
    INITIALIZE\((i_{\text{start}})\);
    \(i := i_{\text{start}}\);
    \(k := 0\);
    repeat
        \text{GENERATE}(i' \text{ from } N(i));
        \text{If } f(i') - f(i) < t_k \text{ then } i := i';
        \(k := k + 1\);
    until STOP;
end;

Figure 10 Pseudo code of a threshold algorithm

The threshold \(t_k\) can be set in different ways. First, when \(t_k\) is set to 0 for \(k = 0, 1, 2, \ldots\) only better improvements are accepted and this is a variant of the classical greedy local search in which only cost-reducing neighbours are considered. Second, by setting \(t_k = c_k\) for \(k = 0, 1, 2, \ldots\), where \(c_k \geq 0, c_k \geq c_{k+1}\) and \(\lim_{k \to \infty} c_k = 0\), a non-increasing sequence of deterministic thresholds is
established. For positive thresholds, neighbouring solutions with larger costs are accepted in a limited way by gradually lowering the threshold values to 0 in which only improvements are accepted. Third, for simulated annealing $t_k$ is a random variable with expected value $E(t_k) = c_k \in \mathbb{R}^+$ for $k = 0,1,2,...$ where the values of $t_k$ follow a probability distribution $F_{c_k}$ over $\mathbb{R}^+$. Randomized thresholds are used with values between zero and infinity and the probability of a threshold $t_k$ is being at most $y \in \mathbb{R}^+$ is given by $P_{c_k}(t_k \leq y) = F_{c_k}(y)$. Therefore each neighbouring solution can be accepted with a positive probability. The function $F_{c_k}$ is chosen in such a way that large increases in costs have a small probability of being accepted, while solutions with a small increase have a larger probability of being accepted (Aarts, Korst & Michiels, 2005). In the original simulated annealing of Kirkpatrick, Gelatt & Vecchi (1983), the negative exponential distribution function with parameter $1/c_k$ is used for $F_{c_k}$ such that the following acceptance criterion is used

$$P_{c_k}(\text{accept } i') = \begin{cases} 1 & \text{if } f(i') \leq f(i) \\ \exp\left(\frac{f(i) - f(i')}{c_k}\right) & \text{if } f(i') > f(i) \end{cases}$$  \hspace{1cm} (4.4-1)

Here, $c_k$ is used as a control parameter in the simulated annealing algorithm, at iteration $k$, which plays the role of the temperature. This simulated annealing algorithm is based on the Metropolis acceptance criterion (Metropolis et al., 1953), which models how a thermodynamic system moves from the current state $i$ to a neighbouring state $j$, in which the energy content is being minimized. Referring back to the physical annealing, $f(i)$ and $f(i')$ denote the energies (objective function values). If the temperature is reduced slowly, then the system can reach a steady state at each iteration $k$. This equilibrium follows the so called Boltzmann distribution, which describes the probability of the system being in state $i \in X$ with energy $f(i)$ at temperature $T$.

The previous pseudo-code of threshold algorithms can be extended towards simulated annealing as is shown in Figure 11 (Aarts, Korst & Michiels, 2005; Michiels, Aarts & Korst, 2007). This corresponds to the threshold algorithm of Figure 10, where $t_k$ now follows a probability distribution in the form of a negative exponential distribution function with parameter $1/c_k$ as presented in Equation (4.4-1).

begin
  INITIALIZE($i_{start}, c_0, L_0$);
  $i := i_{start}$;
  $k := 0$;
  repeat
    for $l := 0$ until $L$ do
      begin
        GENERATE($i'$ from $N(i)$);
        if $f(i') \leq f(i)$ then $i := i'$
        else
          if $\exp\left(\frac{(f(i) - f(i'))}{c_k}\right) > \text{random}[0,1)$ then $i := i'$
      end;
    $k := k + 1$;
    ADJUST ($c_k$);
  until stopcriterion
end;

Figure 11 Pseudo code of the simulated annealing algorithm
As mentioned, the typical feature of simulated annealing is that, besides accepting improvements, also deteriorations are accepted to a limited extent. Initially, for large values of $c_k$, large deteriorations are accepted. As $c_k$ decreases, only smaller deteriorations are accepted and finally as $c_k$ approaches 0, no deteriorations are accepted at all. The speed of convergence of the algorithm is determined by the choice of the parameters $L$ and $c_k$ (Osman & Kelly, 2012). The nested loop generates a neighbour of the current configuration and determines if the new configuration is accepted. This is done for $L$ times, which represents the number of iterations. Then, the control parameter $c_k$ is adjusted, e.g. decreased and the iteration starts again. This is done for a fixed amount of times or when another kind of stop criterion is reached. A particular simulated annealing algorithm is specified by choosing the stopping criterion, the generation of the next candidate point and the cooling schedule, i.e. the temperature through which the acceptance is controlled and how it is lowered (Locatelli, 2000).

4.4.3 Metropolis Procedure

According to Press et al. (1996) it is possible to use the so called metropolis procedure of simulated annealing for other purposes than thermodynamic systems, if the following elements are defined:

1. System state; a description of possible system configurations.
2. A generator of random changes in the configuration; these changes are the “options” presented to the system.
3. An objective function $TF$ (analog of energy $E$); whose minimization is the goal of the procedure.
4. A control parameter $\tau$ (analog of temperature $T$) and an annealing schedule; which tells how it is lowered from high to low values, e.g., after how many random changes in configuration is each downward step in $\tau$ taken, and how large is that step. The meaning of “high” and “low” in this context, and the assignment of a schedule, may require physical insight and/or trial-and-error experiments.

In literature, the Traveling Salesman Problem (TSP) is often used as an example for the application of simulated annealing. In Appendix C this example is shown, describing above elements. The simulated annealing algorithm is usually defined for a combinatorial minimization problem with a discrete, but very large, configuration space, like the set of possible order of cities as in the traveling salesman problem. As mentioned in the beginning of this Chapter, the basic idea of SA can also be applied to optimization problems with continuous $n$-dimensional control spaces, to find the (global) minimum of an objective function $f(i)$, in the presence of many local minima, where $i$ represents a $n$-dimensional vector. In order to apply SA to continuous minimization the four elements required by the Metropolis procedure presented above can be applied. The possible system configuration is now replaced by the $n$-dimensional vector. The principal complication in going from the discrete to the continuous application of SA lies in the neighbour generating mechanism (Ali, Törn, Viitanen, 1997).
5 Simulated Annealing Methodology

In Figure 12 the methodology is formulated that was developed during this research to optimize the distance metric using simulated annealing and is based on the pseudo code of Aarts, Korst & Michiels (2005) and Michiels, Aarts & Korst (2007) presented in in Figure 11.

![Pseudo code of the application of simulated annealing for the optimization of the distance metric](image)

This methodology makes use of the four elements of Press et al. (1996) presented in Section 4.4.3. In the remainder of this chapter the application of these four elements is explained. This involves the definition of the system state (Section 5.1), the objective function (Section 5.2), the control parameter (Section 5.3) and the generator of random changes in the configuration of the system state (Section 5.4). A description of the actual implementation can be found in Appendix D. Corresponding flowcharts and source codes are presented in Appendix E and Appendix F, respectively.

5.1 System state

The system state is represented by a weight vector $A$, where each element represents the weight $a_i$ given to a corresponding predecessor $i$, as was formulated in Equation (2.2-4). As described in Section 3.3.1, based on the embedding window and the step size a set of predecessors is determined, representing relevant events that are taken into account. The initial state is created by starting with the applying equal weights to each predecessor. This corresponds to the use of the Euclidean distance metric (Equation (2.2-2)).

5.2 Objective function

Different functions to measure the prediction accuracy might be applied over which the weighted distance metric can be minimized. When selecting an accuracy measure for reporting the results of forecasting methods and/or comparing the performance of such methods, there are tradeoffs in the
various criteria that must be considered. All accuracy measures are unique and no single measure is superior to all other measures (Makridakis, 1993).

Based on two categories, accuracy measures are outlined. First, measures based on the forecasting error are discussed in Section 5.2.1. Secondly, since the output of the prediction method is in the form of warnings, objective functions based on the correctness of these warnings can be applied and are discussed in Section 5.2.2. An overview containing all the accuracy measures that are discussed in the upcoming sections are presented in Table 1. Finally, the establishment and selection of the objective function that is used in the simulated annealing methodology is discussed in Section 5.2.3.

<table>
<thead>
<tr>
<th>Table 1 Overview of discussed performance measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section</td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Forecasting error-based</td>
</tr>
<tr>
<td>(Section 5.2.1)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Warning-based</td>
</tr>
<tr>
<td>(Section 5.2.2)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Selected Measure</td>
</tr>
</tbody>
</table>

5.2.1 Forecasting error-based

The first accuracy measure that is introduced is the Mean Squared Error (MSE), which as its name implies, provides for a quadratic loss function as it squares and subsequently averages the various errors. The MSE is defined as follows (Makridakis, 1993)

\[
MSE = \frac{\sum_{t=1}^{m} (f_t - \hat{f}_t)^2}{m} = \frac{\sum_{t=1}^{m} e_t^2}{m} \tag{5.2-1}
\]

For the purpose of evaluating the post-sample (monitoring period) accuracy of a prediction method, \( m \) is the total number of observations. The numerator represents the squared sum of the forecasting error \( e \), which is defined as the difference between the actual observation \( f_t \) (actual forward time) and the prediction \( \hat{f}_t \) (predicted forward time). Because of the square, large errors are weighted considerably more heavily than smaller errors, as can be seen in the example of Table 2. Therefore, the MSE is most relevant when a large error is much less desirable than equivalent smaller ones, like in inventory situations. A disadvantage of this accuracy measure is that it is an absolute measure and might get problematic as they are influenced highly by extreme values.

<table>
<thead>
<tr>
<th>Table 2 MSE example large vs. small errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>Scenario 1</td>
</tr>
<tr>
<td>Scenario 2</td>
</tr>
</tbody>
</table>

An accuracy measure that provides a relative measure by expressing errors as a percentage of the actual data is the Mean Absolute Percentage Error (MAPE) and is defined as follows (Makridakis, 1993)
\[ MAPE = \frac{\sum_{t=1}^{m} |f_t - \hat{f}_t|}{m} \]  

This measure is more intuitive as the extent or importance of errors can easily be judged. However, since the error is compared to the actual observation, equal errors might result in different measures, as can be seen in Table 3.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>( f )</th>
<th>( \hat{f} )</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>15</td>
<td>10</td>
<td>33%</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>10</td>
<td>15</td>
<td>50%</td>
</tr>
</tbody>
</table>

This can be overcome by using the symmetric version of MAPE, usually called the Symmetric Mean Absolute Percentage Error (SMAPE). It corrects for the problem of asymmetry and its possible influence by outliers by dividing the forecasting error \( e_t \) by the average of both \( f_t \) and \( \hat{f}_t \), such that is defined as follows (Makridakis, 1993)

\[ SMAPE = \frac{\sum_{t=1}^{m} |f_t - \hat{f}_t|}{m \left( \frac{|f_t| + |\hat{f}_t|}{2} \right)} \]  

In contrast to MAPE, it has both a lower and upper bound, as the range of above formula lies between 0% and 200%. Sometimes the division by two is left out in order to get a range of 0% and 100%. It still has some minor problems with regard to symmetry, because over- and under-forecast are not treated equally as can be seen in Table 4.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>( f )</th>
<th>( \hat{f} )</th>
<th>SMAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>10</td>
<td>11</td>
<td>9,52%</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>10</td>
<td>9</td>
<td>10,52%</td>
</tr>
</tbody>
</table>

Another limitation of this accuracy measure is that if the actual observation or the prediction is equal to 0, it will result in a SMAPE equal to the upper bound.

5.2.2 Warning-based

The output of the prediction model can be considered as a classification problem. At each time interval a warning is or is not generated (classifier), which can be compared to the actual situation (instance). This results in four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a true positive (tp); if it is classified as negative, it is counted as a false negative (fn). If the instance is negative and it is classified as negative, it is counted as a true negative (tn); if it is classified as positive, it is counted as a false positive (fp). Given the classifier for each of the instances, a two-by-two confusion matrix, also called a contingency table can be constructed representing the dispositions of the set of instances as is done in Table 5. This matrix forms the basis for many common metrics (Fawcett, 2006). Based on this contingency table rates can be established that are in relation to the actual class.
The true positive rate, also known as hit rate or recall, is calculated by dividing the correctly predicted positives \((tp)\) by the total amount of actual positives. This metric is often called sensitivity or recall and can be explained as the proportion of real positive cases that are correctly predicted positive (Powers, 2011)

\[
recall = tp rate = \frac{tp}{P} = \frac{tp}{tp + fn} \tag{5.2-4}
\]

Instances that are actually positives but are predicted as negatives \((fn)\) can be set against the total amount of actual positives, which is called the false negative error or miss rate (Powers, 2011)

\[
fn rate = \frac{fn}{P} = \frac{fn}{tp + fn} \tag{5.2-5}
\]

As the sensitivity measures the proportion of positives which are correctly identified as such, specificity measures the proportion of negatives which are correctly identified as such. This is known as the true negative rate, which is calculated by dividing the correctly predicted negatives \((tn)\) divided by the total amount of actual negatives (Powers, 2011)

\[
tn rate = \frac{tn}{N} = \frac{tn}{fp + tn} \tag{5.2-6}
\]

By dividing instances that are actually negatives but are predicted as positives \((fp)\) by the total amount of actual negatives, the false positive rate is calculated. This metric denotes the proportion of real negatives that occur as predicted positives and may also be called fallout or false alarm rate (Powers, 2011)

\[
fp rate = \frac{fp}{N} = \frac{fp}{fp + tn} \tag{5.2-7}
\]

Besides above metrics there are other often used metrics based on the contingency table. One that compares the instances to the predicted class instead of the actual class is precision (confidence) and denotes the proportion of predicted positive cases that are correctly real positives.

\[
precision = \frac{tp}{tp + fp} \tag{5.2-8}
\]

Furthermore, to determine what proportion is correctly predicted the accuracy metric can be used, simply by dividing the correctly predicted instances by the total number of instances.

\[
accuracy = \frac{tp + tn}{P + N} = \frac{tp + tn}{tp + tn + fp + fn} \tag{5.2-9}
\]
Above measures are definitions related to pattern recognition with binary classification. In statistical hypothesis testing $f_n$ and $f_p$ are sometimes referred to as Type I and Type II errors, respectively. The $f_n rate$ and $f_p rate$ are known as $\alpha$ and $\beta$, respectively, referring to falsely rejecting or accepting a hypothesis. Increasing the specificity of the test lowers the probability of type I errors, but raises the probability of type II errors, i.e. false negatives that reject the alternative hypothesis when it is true. Besides that the metrics above can serve as performance measures by itself as target functions, also combinations of these metrics can be used. One often used performance measure is the F-measure (or $F_1$ score), that combines both precision and recall into one metric (Van Rijsbergen, 1979).

$$F_1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

(5.2-10)

Here, precision and recall are both equally weighted. But in the general form the weights can be varied. Other commonly used F-measures are the $F_2$ measure, given twice more weight to recall than precision and $F_{0.5}$ measure, which puts twice more emphasis on precision than recall (Van Rijsbergen, 1979).

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{(\beta^2 \times \text{precision}) \times \text{recall}} = \frac{(1 + \beta^2) \times \text{tp}}{((1 + \beta^2) \times \text{tp}) + (\beta^2 \times f_n) + f_p}$$

(5.2-11)

Besides this often used measure, a variety of target functions can be established based on above metrics, dependent on what metric or combinations of metrics are considered as important to minimize.

5.2.3 Selection of Objective function

It is chosen to use a combination of warning based performance measures as objective function in the simulated annealing methodology. The most important criteria for choosing this objective function for the simulated annealing methodology, where interpretability and above all the degree of influence that can be applied. The level of importance of the performance measures can be varied and the impact of varying this measure can easily be seen and interpreted. The combination consists of the false negative rate ($f_n rate$), the false positive rate ($f_p rate$), the inverse of the warning success rate ($\text{precision}$) called false discovery rate and the event miss rate, each with a corresponding weight

$$TF(A) = (w_1 \times f_p rate) + (w_2 \times f_n rate) + (w_3 \times (1 - \text{precision})) + (w_4 \times \text{event miss rate})$$

(5.2-12)

The objective function is dependent on vector $A$ and a set of weights for this vector should be chosen that has the lowest value for the objection function, as it is a function where negative measures are penalized. The weights $w_1$, $w_2$, $w_3$ and $w_4$ that corresponds to the $f_p rate$, the $f_n rate$, the precision and the event miss rate respectively, can be chosen in such a manner that each of these performance measures are weighted to its relative importance.

Since railway tracks are intensively used and scheduling a visit to a maintenance depot is difficult, costly and time consuming, false positives are not desirable. Therefore it makes sense that these should be minimized, such that it will not occur often that an unnecessary visit is scheduled, where unnecessary cost and time is wasted. The relative importance can be applied in the first weight
term. However, there is usually a trade-off between the false positive rate and the false negative rate, as the false negative rate and the false positive rate are dependent: to decrease one is to increase the other. So when minimizing the false positive rate this usually results in a higher false negative rate. It is also not desirable that the false negative rate increases too much (i.e. no warnings when it should give a warning), because missed warnings may result in an (partly) inoperable system. Therefore, these two should be balanced in such a way that reflects the practical impact of both measures.

Where the \( fp \) \( rate \) denotes the percentage of incorrectly predicted positives compared to all actual negatives (the proportion of a given warning while no error relative to all no-error cases), the false discovery rate or the inverse precision, compares the incorrectly classified positives to all the positive predicted cases (the proportion of a given warning while no error relative to all warnings). This latter measure is also incorporated in the objective function such that importance of the unsuccessful warnings can be influenced. These can be balanced against the event miss rate (i.e. the proportion of actual critical events for which no warning was given in advance). In total the four weights can reflect the practical severity of each of the measures and should be related to the risk aversion of the stakeholder. As objective functions the following set of weights are used:

\[
TF(A) = (0.25 \times fp \ rate) + (0.25 \times fn \ rate) + (0.25 \times (1 - precision)) + (0.25 \times event \ miss \ rate)
\]

\[
TF(A) = (0.1 \times fp \ rate) + (0.4 \times fn \ rate) + (0.1 \times (1 - precision)) + (0.4 \times event \ miss \ rate)
\]

\[
TF(A) = (0.4 \times fp \ rate) + (0.1 \times fn \ rate) + (0.4 \times (1 - precision)) + (0.1 \times event \ miss \ rate)
\]

5.3 Control parameter

An important aspect in the implementation is the control parameter \( \tau \), equivalent to the temperature \( T \) in the original SA methodology. This control parameter is used together with the annealing schedule by which it is gradually reduced, also known as the cooling schedule. First, the starting value for the control parameter \( \tau \) is defined and should be considerably larger than the largest difference normally encountered in the target function \( \Delta TF \). When this is the case, the probability of accepting slightly worse solution is large and from here this probability can be gradually lowered. Based on (Press et al., 1996) the control parameter and the cooling schedule are established.

The initialisation of control parameter \( \tau \), is performed by multiplying the initial value of the objective function, using the Euclidean distance metric \( (A = I) \), by a cooling factor \( CFACTOR \).

\[
\tau_0 = CFACTOR \times TF(I)
\]

This initiation is performed in Step 3. A larger starting control parameter \( \tau_0 \), caused by a larger chosen \( CFACTOR \), has the consequence that in the initial stage, worse solutions are accepted more often, while this acceptance probability is more strict when using a smaller \( CFACTOR \).
The control parameter is used to accept slightly worse solutions in order to get out of local minima during movement through the system state space. This movement is performed by reconfigurations in the system state, which can be described as rearranging the weights in the state vector. This is elaborated in depth in Section 5.4. After each reconfiguration of the state vector \((A \rightarrow A')\), the result of the objective function \(TF(A')\) is assessed. If the objective is smaller than the most optimal configuration so far \(TF(A^*)\), it is set as optimal system state \(A^*\) and the \(TF(A^*)\) is also updated. Then it is compared to the result of the current configuration \(TF(A)\). If the reconfiguration results in a better (i.e. smaller) objective, this reconfiguration is kept and is used as comparison for the next reconfiguration. If not, the metropolis criterion is used whether or not to accept the reconfiguration. A random number \(\varphi\) is drawn from on the interval \([0, 1)\), such that \(\varphi\) is continuously and uniformly distributed between 0 and 1. The reconfigured system state is accepted, although it is not better than the current, when the following inequality holds

\[
\varphi < e^{-(TF(A') - TF(A))/\tau}.
\]

(5.3-2)

The optimal configuration \(A^*\) however, is of course not adjusted. If the random number is greater or equal, the model proceeds with the previous configuration and corresponding objective value. The calculation and the assessment of the reconfigured objective value are included in Step 5. The number of reconfigurations during each temperature level is defined by the parameter \(nOver\). It restricts how many times a new configuration is created and assessed. Furthermore, it is kept track how many reconfigurations were successful (\(n Succ\)). A reconfiguration is successful when either the reconfiguration is better (smaller) than the current configuration or when the difference is tolerated by the metropolis criterion. When the total number of reconfigurations is still smaller than \(nOver\), but \(n Succ\) equals the predefined limit of successful reconfigurations \(n Limit\), then the algorithm advances to the next temperature level. This is done in order to proceed faster to next levels of \(\tau\) when this control temperature parameter is too loosely defined. By setting \(n Limit = n Over\) this latter restriction is neglected.

After finishing the reconfigurations for the initial temperature level, this control parameter is lowered according a cooling schedule, also called the annealing schedule. Based on Press et al. (1996) and Aarts, Korst & Michiels (2005) it is chosen to proceed downward in multiplicative steps each amounting a certain percent decrease in \(\tau\). In other words, the temperature level \(\tau\) is reduced \((1 - \varepsilon)\tau\) after every \(n Over\) moves or earlier if \(n Limit\) is reached. For this setting \((1 - \varepsilon)\) should be determined by experiment and is denoted by \(TFACTOR\), such that

\[
\tau_i = \tau_{i-1} \ast TFACTOR,
\]

(5.3-3)

where \(i = 1, 2, \ldots, nSteps - 1\) and \(\tau_0 = CFACTOR \ast TF(I)\). According to Aarts, Korst & Michiels (2005) the value of \(TFACTOR\) should be close to, but smaller than 1 and typically lies between 0.8 and 0.99. The parameters \(nSteps\) and \(TFACTOR\) together should ensure that the algorithm stops when efforts to reduce \(TF\) further becomes sufficiently discouraging.

To show the function of the temperature level and the effects of the parameters \(CFACTOR\) and \(TFACTOR\) a few scenarios of numerical examples are considered, which can be found in Appendix G.
5.4 Generator of random changes in the configuration

So far, the way in which random changes in the configuration space are being performed was left open. It was discussed that reconfigurations are performed but not how these reconfiguration are determined. When simulated annealing is used for an optimization problem with a discrete configuration space, neighbor configurations are established from the current configuration in such a way that all possible solutions can be reached. However, since we are dealing with a continuous $n$-dimensional control space, this part is more difficult. As in a discrete optimization space, in order to move through the system state space, new configurations are being established using the current configuration ($A \rightarrow A'$). For this application it is chosen that two types of changes can be performed:

I. One dimensional: By subtracting a random amount (shift) from one randomly chosen dimension (indexSubtract) and adding this amount to another randomly chosen dimension (indexAdd);

II. Multiple-dimensional: By subtracting (or adding if subtracting is not possible) certain random amounts from all dimensions in the system space.

In both options, the random changes that are made can either be small or big. By this manner, close neighbors can be reached in one or multiple dimensions, but also system states that are further away, either in one or multiple dimensions are within reach. This can be explained as exploring a region within the configuration space by small steps in one direction or in multiple directions, while sometimes making a trip to further situated regions also in one or multiple directions. This is graphically shown in Figure 13 and Figure 14. Figure 13 shows how it is intended to find the optimal solution by exploring different regions. Figure 14 shows a simple 3-dimensional configuration space, where the small circles represent a system state (one configuration). The first and second arrow represents the exploring of the current region with small steps in one direction and multiple directions, respectively. These are the reconfigurations that will occur most frequently. The less often occurring trips to other regions are represented by the third and fourth arrow for movements in respectively one and multiple directions.
For the first kind of reconfiguration, a one-dimensional reconfiguration, the random amount (\textit{shift}), that is subtracted from one dimension, should be greater than the amount currently in that dimension, in order to prevent that this dimension becomes negative. So when this amount is smaller it is simply subtracted from one dimension (\textit{indexSubtract}) and added to another dimension (\textit{indexAdd}). However when this \textit{shift} is greater, the dimension from which the amount is subtracted is set to zero and the current value of the dimension from which the subtraction is done is added to the other dimension. By using this way of working it is always guaranteed that the dimensions remain positive and that the total sum of all dimensions remains the same during the reconfigurations. This is necessary because then the values of the dimensions can be interpreted as weights. Of course, the dimensions \textit{IndexSubtract} and \textit{IndexAdd} should not be the same in order to make an adjustment in the system space. Either a small random amount is shifted from one dimension to another dimension or a big random amount is shifted from one to another. By making this distinction between small and big random amounts, each with its own probability of occurring, one-dimensional movements in the configuration space can either be relatively close to or further away from the current configuration. This difference in the size of the random amount is incorporated by using a factor by which a random number between 0 and 1 is multiplied. For both small and big random changes a random number is drawn from a continuous uniform distribution with an interval \([0,1.0]\). By the multiplication with predefined factors, a fraction of the system space value is added or subtracted to a dimension. The size of this fraction is multiplied by the length of the state vector and dependent on the factors for both small and big movements, which are set to 0.1 and 0.5 respectively.

For the second kind of reconfiguration, a multiple-dimensional reconfiguration, all dimensions are being altered, which requires a more complex handling, as also here the restrictions hold that the values in the system space may not be negative and that the sum of values of all dimensions should remain the same during the reconfigurations. Instead of randomly selecting the dimensions to subtract and to add, changes to all the dimensions are performed for these multiple reconfigurations. This is done by generating a random number, again from the continuous uniform distribution between 0 and 1, for each of the dimensions separately. Then for each dimension, it is determined whether or not the random amount is smaller than the current value of the dimension, such that this amount can be subtracted from the current value. If not, so the random amount is greater than the current value of the dimension, the current value is subtracted from the random amount. By this manner, the values of the dimensions are always positive. However, the sum of all dimensions does not remain the same using this approach. This shortcoming can be counteracted by translating the reconfigured values in the dimension back to relative weights. It is known that the initial values in all dimensions are equal to 1, which means that the initial sum is also known, namely the length of the state vector. The reconfigured state vector can be converted to a vector with the same sum as this initial sum, by multiplying each dimension with the division of this initial sum by the sum of the new vector. As in the single reconfiguration, also for the multiple configuration a small and a big version is implemented, with the same initial factors 0.1 and 0.5. This ensures that also in the multi-dimensional direction movements can either be relatively close to or further away from the current configuration.
To summarize, four types of reconfiguration can take place, which are shown in Table 6. Each type of reconfiguration has its own probability of occurring. The goal is that regions are explored in small steps, while sometimes jumping to other regions. In the probabilities for the occurrence of the types of reconfigurations, it is therefore chosen to give higher probabilities to reconfigurations that explores the current region (i.e. low factor size) and give lower probabilities to those that causes jumps to other regions (i.e. high factor size). Both the small steps within a region and the jumping to other regions can be performed in either one or multiple directions. It is chosen to give one-dimensional reconfigurations more probability of occurring over multi-dimensional reconfigurations. Because if we look at the distance between a new and an old vector, a movement in one dimension is closer to its current reconfiguration compared to a movement in multiple dimensions. So, the highest probability (0.5) is given to a small move in one direction within the region (MoveSingleSmall). The second highest probability (0.35) is given to small moves within the region, but in multiple directions (MoveMultipleSmall). In total, 85% of the reconfigurations consist of a movement within the current region. The other 15% belongs to movements that result in jumping to other regions of the solution space. In two-third of these moves, thus a total probability of 0.10, are in one direction (MoveSingleBig) and the remaining probability (0.05) belongs to moves that causes the greatest jump, which are those that are in multiple directions (MoveMultipleBig).

<table>
<thead>
<tr>
<th>Reconfiguration</th>
<th>Dimension</th>
<th>Factor size</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoveSingleSmall</td>
<td>1-dimensional</td>
<td>0.1</td>
<td>0.50</td>
</tr>
<tr>
<td>MoveMultipleSmall</td>
<td>𝑛-dimensional</td>
<td>0.1</td>
<td>0.35</td>
</tr>
<tr>
<td>MoveSingleBig</td>
<td>1-dimensional</td>
<td>0.5</td>
<td>0.10</td>
</tr>
<tr>
<td>MoveMultipleBig</td>
<td>𝑛-dimensional</td>
<td>0.5</td>
<td>0.05</td>
</tr>
</tbody>
</table>
6 Evaluation

For the implementation of simulated annealing to optimize the weighted distance metric, a test case is used. This test case is provided by FCE and contains real-life data of train events from another maintenance company of trains. This same test case is used in the evaluation of the optimization processes in the Event Simulator. The test case and the hereby used settings in the Event Simulator during this first evaluation are briefly described in Section 6.1.

The goal of this test case evaluation is to gather insights in the optimization processes of both the weighted distance metric as the other discussed parameters (Section 3.3) that affect the prediction accuracy. Based on both evaluations, guidelines are established which can be used by FCE and NedTrain on any optimization case, using the FCE event simulator.

To evaluate the results of the optimization of the parameters and SA, the prediction accuracy is measured in terms of the objective function given as Equation (5.2-12). The evaluation is performed on the monitoring set. The prediction accuracy of the training set outperforms the prediction of the monitoring set. The goal is to find optimal parameters based on the data within the training set that are generalizable and should therefore perform as good as possible on the monitoring set. If the training set is used to evaluate the results, the parameters can be over fitted on this specific data.

First, the effects of other parameters within the Event Simulator are analyzed in order to see the influence on the prediction accuracy and to what extent these can improve the prediction accuracy. The parameters that are taken into consideration are the step size, the embedding window and the CDF threshold. Furthermore, the role of the objective function is taken into consideration. Based on these results, the effects of applying a weighted distance metric can be evaluated more appropriately in relation to the effects of these parameters on the prediction accuracy. The analysis of these optimization parameters is discussed in Section 6.2. The main result of this section is that the parameters described are of paramount importance and can improve the prediction accuracy to a very large extent.

In order to analyze the potential improvements of a weighted distance metric, the roles of the different parameters within SA are evaluated with regards to their effects on finding an optimal weighted distance metric. It is evaluated if SA can improve the prediction accuracy by means of a weighted distance metric, compared to the Euclidean distance metric. Also it is checked if these improvements are reproducible and thus the global optimum is found instead of local optima. The analysis of the simulated annealing methodology for the weighted distance metric is discussed in Section 6.3. The main findings in this section are that a weighted distance metric can substantially improve the prediction accuracy, but that simulated annealing might not be the best way to find an optimal set of weights for the distance metric within reasonable computation time, as the search space cannot be explored enough.

The insights obtained from both analyses are summarized in Appendix H. Here, these insights are translated to a methodology for getting the best prediction accuracy using the Event Simulator. Finally, in Section 6.4, it is shortly addressed why critical events available in the data set of NedTrain could not be predicted by other events using the Event Simulator, as it was initially planned to apply the established methodology to a case of NedTrain.
6.1 Test case Description

The test case consists of events from one vehicle “015” over a time period from December 2012 until July 2014. Based on all diagnostic events from this vehicle, a critical event (“0642”) from this same vehicle is predicted, without excluding any events from specific subsystems. The test case is evaluated using the fixed settings stated in Table 7.

<table>
<thead>
<tr>
<th>Table 7 Fixed settings during evaluation of test case</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General settings:</strong></td>
</tr>
<tr>
<td>Training Period</td>
</tr>
<tr>
<td>Monitoring Period</td>
</tr>
<tr>
<td>Forecast Horizon</td>
</tr>
<tr>
<td><strong>Objective function for evaluating prediction accuracy:</strong></td>
</tr>
<tr>
<td>Equation (5.2-12), for which different weights are applied (see equations (5.2-13), (5.2-14) and (5.2-15))</td>
</tr>
</tbody>
</table>

The training period covers 2/3 of the data set and the remaining 1/3 is used as the monitoring period. In this manner, the 77 occurrences of the critical event are about equally divided over these two periods, as can be seen in Figure 15. During the evaluation the forecast horizon is set fixed to $24 \times 7 = 168$ hours, thus a prediction is made whether the critical event will occur within one week.

![Figure 15 Occurrences of critical event 0642 on vehicle 015 divided over the training and monitoring set](image)

6.2 Analysis of other optimization parameters

The parameters taken into consideration are the combination of the embedding window (EW) and the step size (SS) (see Section 3.3.1) and the CDF threshold (see Section 3.3.2). When these parameters are optimized in the Event Simulator Tool, the tool uses the best combination of these parameters in terms of the objective function to finally present the prediction results, based on this best combination. However, it is not known what exactly happens behind the scenes when these parameters are varied. This section therefore analyses the effects of varying the EW, SS and CDF threshold. Besides these three parameters, also the role of the weights within the objective function (as stated in equation (5.2-12)) are taken into consideration here.

First it should be noted that an optimal combination of the three parameters highly depends on the nature of the data, e.g. how patterns develop over time towards a critical event and have therefore different optimal settings when different data is used. However, some important results can be stated based on the evaluation of the current test data set and are explained in the remainder of this section.
In Section 6.2.1 the optimal combination of parameters is compared to other combinations in terms of the objective value. The robustness of the optimal combination is discussed in Section 6.2.2, followed by the effects of using too high or too low embedding windows and step sizes (Section 6.2.3). In Section 6.2.4 the effects of the weights in the objective function are evaluated and in Section 6.2.5 these weights are related to the optimal CDF threshold.

6.2.1 Effects of optimization of EW, SS and CDF threshold

The main result of the evaluation of using different parameters is that the value of the objective function that represents the prediction accuracy is highly dependent on the choice of the parameters. The effect of optimizing the parameters EW, SS and CDF threshold simultaneously (2nd column), together with the effect of only optimizing the parameters of EW and SS using a fixed CDF threshold (column 3-11) are illustrated in Table 8. The average objective value (2nd row) represents the average over all used combinations of EW (6–144 hours) and SS (1–24 hours) and the optimal objective value (3rd row) shows the optimal value of all these combinations. The upper and lower bound of these two parameters are chosen from practical point of view. Here, the weights in the objective function as stated in equation (5.2-14) are used.

<table>
<thead>
<tr>
<th>CDF Threshold</th>
<th>Optimized</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. objective value</td>
<td>0.1741</td>
<td>0.2411</td>
<td>0.2338</td>
<td>0.2211</td>
<td>0.2142</td>
<td>0.2115</td>
<td>0.2025</td>
<td>0.2264</td>
<td>0.2630</td>
<td>0.2904</td>
</tr>
<tr>
<td>Optimal objective value</td>
<td>0.0850</td>
<td>0.1542</td>
<td>0.1350</td>
<td>0.1512</td>
<td>0.1329</td>
<td>0.1218</td>
<td>0.1097</td>
<td>0.0947</td>
<td>0.0988</td>
<td>0.0850</td>
</tr>
<tr>
<td>Improvement (min. vs avg.)</td>
<td>51.07%</td>
<td>36.03%</td>
<td>42.27%</td>
<td>31.60%</td>
<td>37.95%</td>
<td>42.42%</td>
<td>45.84%</td>
<td>58.15%</td>
<td>62.45%</td>
<td>70.74%</td>
</tr>
</tbody>
</table>

It can be stated that independently of the chosen CDF threshold, large improvements can be established by optimizing the combination of step size and embedding window. This large improvement can be explained by means of two reasons. First, the patterns should follow the nature of the data and should therefore be established using an embedding window and a step size, such that the patterns have the best predictive value. Secondly, the establishment of the predecessors is based upon the established data patterns. So, the selected predecessors based upon these patterns shall also have better predictive value. By including CDF threshold optimization, the objective values for all EW and SS combinations are improved, as for all the combinations the optimal threshold is used. The higher the CDF threshold, the more optimal the objective value becomes. However, this threshold is mostly influenced by the weights used in the objective function, as described later.

![Figure 16 Robust area of optimal parameter combination (for objective function as in (5.2-14))](image)
6.2.2 Robust optimal area of parameter combination EW and SS

The evaluation of the optimization of the parameters gives an indication that there exist a robust optimal area in terms of the EW and the SS as can be seen in Figure 16. Again, the equation of (5.2-14) is used as objective function. The robust area shows that there is a certain interval for both the EW and the SS in which the objective function always has a low (close to optimal) value. This robust area can be explained by the fact that on this interval the nature of the patterns are optimally used. It is also the case that within this area the same predecessors are used and this can also be a reason why this area has a robust lower value.

In order to show the robustness of this area, it is analyzed for the currently used objective function how the objective value increases around the optimal combination. In order to get a result of not more than a decrease of 5% in the objective function compared to the optimal objective value, it is allowed to move 4 or 5 hours above or below in the EW parameter setting. Within this range of EW, a decrease of maximal 8.5% is obtained if a step size of 1 hour less is used. If a parameter setting outside this robust area is used the prediction accuracy drops (e.g. objective function increases) substantially (around 50% worse, just as the comparison with the overall average performance of all the used combinations). In Table 9 the actual values for the objective function are shown and in Table 10 these are translated into how many percentage the objective function increases, compared to the actual optimal parameter setting. For other objective functions, these differences between the accuracy within and outside the robust area are comparable, only the sizes of these robust areas are different and are located elsewhere.

Table 9 Prediction performance (objective value) within and outside robust area

<table>
<thead>
<tr>
<th>Step Size</th>
<th>Embedding Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>EW</td>
<td>SS</td>
</tr>
<tr>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>42</td>
<td>43</td>
</tr>
<tr>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
</tr>
<tr>
<td>53</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1300</td>
</tr>
<tr>
<td>2</td>
<td>0.1215</td>
</tr>
<tr>
<td>3</td>
<td>0.1163</td>
</tr>
<tr>
<td>5</td>
<td>0.1359</td>
</tr>
<tr>
<td>6</td>
<td>0.1742</td>
</tr>
</tbody>
</table>

Table 10 Deterioration (%) compared to optimal in prediction performance within and outside robust area

<table>
<thead>
<tr>
<th>Step Size</th>
<th>Embedding Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>EW</td>
<td>SS</td>
</tr>
<tr>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>42</td>
<td>43</td>
</tr>
<tr>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
</tr>
<tr>
<td>53</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>52.9</td>
</tr>
<tr>
<td>2</td>
<td>43.0</td>
</tr>
<tr>
<td>3</td>
<td>36.9</td>
</tr>
<tr>
<td>4</td>
<td>44.5</td>
</tr>
<tr>
<td>5</td>
<td>59.9</td>
</tr>
<tr>
<td>6</td>
<td>105.0</td>
</tr>
</tbody>
</table>

Although there is a robust area in which the differences in the objective function are not that large compared to the use of parameters outside this region, it is still beneficial to find the actual most optimal parameters within this area, as these differences are still significant with neighbouring parameters as can be seen in Table 9. The computation time involved in the optimization of the three parameters in scope here is not that time consuming, in relation to the simulated annealing methodology, as is described later. An optimal combination of parameters can be found in a couple of minutes and the computation time is dependent on the size of the used grids. Obviously, the upper and lower bound should be set widely, such that the optimal combination can be found. Moreover, the grid size should be set very large, such that the optimal combination of parameters can be found, precisely.
6.2.3 Too low or too high step sizes and embedding windows

Not too low and not too high EW and SS combinations almost always give better results in terms of prediction accuracy compared to extremer values of these parameters. That the combination of lower step sizes and embedding windows give better results, can be intuitively be explained by the fact that if a combination of a too large values is used, information is lost since the occurrence of events are all put together in one pattern. Using smaller step sizes and smaller embedding windows, more information can be retrieved from the patterns. The EW and the SS should also not be too small, as too less information is stored into a pattern. Figure 16, already shows a valley, as the surface becomes higher moving towards higher step sizes and embedding windows. By applying different weights in the objective function (like those of Equation (5.2-13) and Equation (5.2-15)), this valley becomes even clearer and also the negative effect of using too small embedding windows is more explicit. This can be seen in Figure 17. Here, it is important to note that the smallest step sizes are left out to show the valley more clearly, but these also have high values for the target function.

![Figure 17 Effect of step size and embedding window for objective function (5.2-13) (left) and (5.2-15) (right)](image)

6.2.4 Impact of different weights in the objective function

Using different weights in the objective function, results not only in different values of the objective function, but also affects the optimal parameter combination. To illustrate this, the equally weighted objective function (equation (5.2-13)) is compared to the application of other weights like those in equations (5.2-13), (5.2-14) and (5.2-15), and the result on the optimal combination of parameters are shown in Table 11. All objective functions, show robust areas, but these are located elsewhere for each of the functions, as can be seen in Figure 16 and Figure 17. The choice of the objective function should correspond with the risk aversion of the stakeholder that wants to predict events.

<table>
<thead>
<tr>
<th></th>
<th>(5.2-13)</th>
<th>(5.2-14)</th>
<th>(5.2-15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FN Rate</td>
<td>0.25</td>
<td>0.1</td>
<td>0.4</td>
</tr>
<tr>
<td>FP Rate</td>
<td>0.25</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>1-Warning Success Rate</td>
<td>0.25</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>Event Miss Rate</td>
<td>0.25</td>
<td>0.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>

**Optimal Settings:**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding window</td>
<td>108</td>
<td>42</td>
<td>102</td>
</tr>
<tr>
<td>Step Size</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>CDF threshold</td>
<td>0.4</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

In order to see if the applied weights in the objective function indeed fulfill their purpose in terms of accuracy measures, the three mentioned objective functions were used to generate an optimal
solution, by optimizing the EW, SS and CDF threshold. As a recap, the weights used in the objective functions can also be found in Table 11. The individual accuracy measure results are shown in Figure 18. As can be seen, in the second objective function, the FP-rate is minimized at the expense of the FN-rate compared to the equally weighted objective function and for the third objective function vice versa. The same holds for the warning success rate and the event miss rate.

![Figure 18 Prediction accuracy result on objective equation (5.2-14), (5.2-13) and (5.2-15), respectively](image)

6.2.5 Relation between objective function and optimal CDF threshold

The optimal CDF threshold is most influenced by how the weights are divided over the individual performance measures in the objective function. In Table 11 it can be seen that in the equally weighted objective function (Equation (5.2-13)), a CDF threshold of 0.4 is optimal, while if an objective function is used where false positives are penalized more (Equation (5.2-14)), a CDF threshold of 0.9 is optimal. Here, false warnings should be avoided and therefore the CDF threshold is stricter. This can also be seen in the optimal objective values in Table 8, where the objective value becomes more optimal for higher CDF thresholds. The other way around, where false negatives are penalized more (equation (5.2-15)), the effect on the CDF threshold is reversed in the optimal setting. Here the CDF threshold is equal to 0.1, which results in the avoidance of missing critical events. This relationship between the weights in the objective function and the CDF threshold also holds for optimal CDF thresholds within the robust area. Outside the robust area this relationship is less strong.

6.3 Analysis of SA on distance metric optimization

In order to analyze the effects of the distance metric, for all the simulated annealing runs the objective function in equation (5.2-14) is used, where the focus is on preventing false positives. The main goal in this section is to evaluate if the prediction accuracy can be improved by the use of a weighted distance metric. Furthermore it is analyzed how to find the best weights for the distance metric in terms of the prediction accuracy within reasonable computation time. As is described in this section, with the SA methodology large computation time is involved. This large computation time is caused by the fact that for every set of weights assessed, the whole similarity sampling procedure must be performed entirely. Due to this problem with the computation time it was not possible to perform the necessary long runs for simulated annealing to find reproducible results and to perform simulated annealing for multiple combinations of the EW, the SS and the CDF threshold (Section 6.3.1). The improvements found using the weighted distance metric during this analysis show the positive potential effect on the prediction accuracy (Section 6.3.2). However SA cannot find reproducible results for the weights in the distance metric in terms of optimality, as is described in Section 6.3.3. In Section 6.3.4 it is stated that in theory longer annealing runs should improve this reproducibility.
6.3.1 Optimization of distance metric for optimal parameters of EW, SS and CDF threshold

Due to the limitations in computing time, the optimization of the weights within the distance metric, using simulated annealing, can only be performed on the optimal set of parameters of the EW, the SS and the CDF threshold. As described in the previous section, the differences in terms of the prediction accuracy are substantial for non-optimal combinations, especially when these are lying outside the robust region. So it makes sense to only use the optimal parameter setting. In theory, the weights within the distance metric can correct for worse EW and SS combinations, as non-optimal predecessors (based on the EW and SS combination) should receive low to none weights. However, the weights cannot correct for data patterns that do not follow the nature of the data.

6.3.2 Improvements of a weighted distance metric compared to the Euclidean distance metric.

Although there may exist weights that are more optimal, the best improvement found during numerical experimentations for the optimization of weights in the distance metric are substantially. The result of the objective function is 26.9% better than the result with the Euclidean distance metric. The most optimal found weighted distance metric has improved results in terms of all four individual performance measures within the objective function. Although it emphasizes on avoiding false positives and having a high warning success rate, most of the decrease is contributed by the improved event miss rate, since no actual event is missed (e.g. before every critical event, at least one warning is given on beforehand) in the monitoring set. Here also the most improvement was possible, but more importantly is that it does not affect the other accuracy measures. In contrary, these are all improved. The improvements on the objective function and the individual accuracy measures are stated in

Table 12 and the output of the best found weighted distance metric and the Euclidean distance metric from the Event Simulator are graphically shown in Figure 19.

<table>
<thead>
<tr>
<th>Optimal Setting parameters</th>
<th>EW</th>
<th>SS</th>
<th>CDF Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean distance metric</td>
<td></td>
<td></td>
<td>0.9</td>
</tr>
<tr>
<td>Objective Function</td>
<td>0.0850</td>
<td>0.0621</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FN-Rate</td>
<td>0.1</td>
<td>33.5%</td>
<td>32.9%</td>
</tr>
<tr>
<td>FP-Rate</td>
<td>0.4</td>
<td>2.8%</td>
<td>2.3%</td>
</tr>
<tr>
<td>1 – Warning Success rate</td>
<td>0.4</td>
<td>6.0%</td>
<td>5.0%</td>
</tr>
<tr>
<td>Event Miss Rate</td>
<td>0.1</td>
<td>16.7%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Figure 19 Prediction accuracy results Euclidean distance (left) and weighted distance metric (right)
6.3.3 Reproducibility of optimal weights for the distance metric within computation time with SA

Due to the large computation time involved with each iteration in the simulated annealing algorithm, it is not possible to use appropriate long annealing runs. The parameters that are of influence on finding reproducible results in terms of the distance metric could therefore only be increased to a limited extent. Due to this limitation only local minima are found, located in different regions of the search space and no reproducible results in terms of weights were found with the SA methodology. As is mentioned before, the best found combination of weights has a substantial positive effect on the prediction accuracy. In every other separate simulated annealing run, an improvement is found in terms of the objective function. As an example, Table 13 shows the results of the objective value together with the corresponding set of weights for the 17 predecessors, for each run performed with the parameter setting that found the most optimal set of weights during experimentation. The used parameter setting (nOver = 100, nLimit = 10, nSteps = 200, CFACTOR = 2 and TFACCTOR = 0.95) took about 6 hours and still the search space could not be explored enough in order to find some reproducible results. It can be seen that different runs result in different optima and that these optima are found for different combinations of weights within the distance metric. This makes sense since the number of reconfigurations was very low and taken into account that the dimension of the state vector has a length of 17 predecessors. Only small parts of the search space are taken into consideration during one run and the quality of the solution depends on the parts of the search space that are explored. For runs in which the number of reconfigurations within a temperature level (nOver) is increased (with decreased number of temperature levels and a faster cooling schedule, to stay within reasonable computation time), no improvements in the results were found in terms of optimality and reproducibility.

Table 13 Result of multiple runs using the parameter setting in which the best improvement was found

<table>
<thead>
<tr>
<th>Run</th>
<th>Optimal</th>
<th>0642</th>
<th>0ABC</th>
<th>0FAE</th>
<th>0AA3</th>
<th>0FAF</th>
<th>0242</th>
<th>0F91</th>
<th>0F90</th>
<th>0F9B</th>
<th>0FE2</th>
<th>0C23</th>
<th>0A9F</th>
<th>7502</th>
<th>7501</th>
<th>029A</th>
<th>0643</th>
<th>3704</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0802</td>
<td>1.702</td>
<td>1.386</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>2.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.211</td>
<td>1.298</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.423</td>
</tr>
<tr>
<td>2</td>
<td>0.0681</td>
<td>0.000</td>
<td>1.000</td>
<td>2.258</td>
<td>1.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
<td>1.382</td>
<td>1.326</td>
<td>2.618</td>
<td>1.000</td>
<td>0.445</td>
<td>1.116</td>
<td>1.875</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>3</td>
<td>0.0838</td>
<td>1.000</td>
<td>1.000</td>
<td>2.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.765</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>0.0803</td>
<td>2.889</td>
<td>1.706</td>
<td>2.638</td>
<td>0.491</td>
<td>0.576</td>
<td>0.652</td>
<td>1.511</td>
<td>0.001</td>
<td>1.370</td>
<td>3.393</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>1.233</td>
<td>0.540</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.0621</td>
<td>1.000</td>
<td>0.048</td>
<td>1.000</td>
<td>0.268</td>
<td>2.090</td>
<td>1.000</td>
<td>1.000</td>
<td>1.679</td>
<td>0.318</td>
<td>0.000</td>
<td>1.284</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.732</td>
</tr>
<tr>
<td>6</td>
<td>0.0751</td>
<td>0.348</td>
<td>0.000</td>
<td>0.092</td>
<td>1.172</td>
<td>0.247</td>
<td>0.211</td>
<td>0.211</td>
<td>0.000</td>
<td>0.374</td>
<td>0.000</td>
<td>0.000</td>
<td>1.114</td>
<td>4.769</td>
<td>0.000</td>
<td>0.000</td>
<td>2.178</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.0847</td>
<td>2.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.0803</td>
<td>0.780</td>
<td>1.091</td>
<td>0.577</td>
<td>0.711</td>
<td>0.447</td>
<td>1.423</td>
<td>0.622</td>
<td>1.139</td>
<td>0.428</td>
<td>1.756</td>
<td>0.695</td>
<td>0.398</td>
<td>3.024</td>
<td>0.923</td>
<td>0.928</td>
<td>1.362</td>
<td>0.697</td>
</tr>
</tbody>
</table>

In order to find more reproducible results, the search space was decreased, by decreasing the number of predecessors. However, this had a very negative effect on the prediction accuracy, since important predecessors are taken out of the prediction. Also increasing the step size, in order to reduce the computation time had a very negative effect on the prediction accuracy, as is in line with earlier analysis of this parameter. To visit more regions in the search space, the probability of performing bigger jumps to other regions within the search space (MoveSingleBig and MoveMultipleBig) were increased and the probability for smaller jumps within these regions (MoveSingleSmall and MoveMultipleSmall) were decreased. However, this makes the algorithm more like random search instead of a simulated annealing methodology and certainly did not have the effect that it converges to an optimal solution.

Based on all the performed SA runs, with varied parameter settings that are applicable within a reasonable computation time, no clear statements can be made on which predecessors are important, as for all of the found (local) optimums the weights differ and there are no predecessors that always receive small or high weights. Furthermore, it cannot be concluded that above
parameter setting always results in the best possible prediction accuracy, but of all total runs with various parameter settings, this setting provided the most optimal prediction accuracy. For all possible parameter settings, multiple runs are performed in order to see the effect of the setting. Despite the lack of reproducible results, it can be concluded that there are minima for which the prediction accuracy is improved significantly. These are however too hard to find with the applied simulated annealing technique within the used computation time.

![Simulated Annealing example on local search space area](image)

Although the quality of the solution highly depends on which regions are explored during the search, there are examples that show the advantages of using simulated annealing as a local search technique. The simulated annealing run in Figure 20 shows that for the limited annealing length it is able to escape from local minimums in order to find a better minimum in a small region in the search space. Although it is on a very low scale, this example gives an indication that if longer annealing runs are performed in which larger parts of the search space are explored, the used SA methodology might be capable of finding more global optima.

6.3.4 Longer runs to find more optimal and reproducible prediction accuracy

In theory, to have the highest probability of finding the best results with the simulated annealing algorithm, the four parameters $nOver$, $nSteps$, $TFACTOR$ and $CFATOR$ should be as large as possible. According to Johnson et al. (1989), long annealing runs must be allowed and one should be aware of the interactions between the parameters in SA. To allow for long annealing runs, two possibilities are stated: 1) slower cooling ($TFACTOR$) and more temperature levels ($nSteps$) and 2) use more reconfigurations within a temperature level ($nOver$). Furthermore, one should take into account the interaction with the initial temperature level ($CFATOR$). However, due to the computation time restrictions, if one of these parameters is increased, another parameter should be decreased and therefore the lengths of the runs are always limited and only the parameters could be varied reciprocally. Johnson et al. (1989) also states that even with long runs there can still be a large variation in the quality of solutions found by different runs. However, it is claimed that it seems better to perform one long run than to take the best of a collection of shorter runs. During this evaluation better results were found for the first possibility in which more temperature levels are used with less reconfigurations within the temperature levels than for the first option with more reconfigurations within the temperature levels (and less temperature levels and quicker cooling). In order to show that longer annealing lengths improve the found solution in terms of optimality, Table 14 shows the optimal results of the SA runs with the settings of those of Table 13, but with varied...
As can be seen is that increasing this number of temperature levels indeed give more optimal values for the objective functions and also on average the objective values are higher.

Table 14 Effect of the annealing length by varying the number of temperature levels

<table>
<thead>
<tr>
<th>nSteps</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Average</th>
<th>Average (top 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.0646</td>
<td>0.0681</td>
<td>0.0752</td>
<td>0.0802</td>
<td>0.0802</td>
<td>0.0803</td>
<td>0.0839</td>
<td>0.0848</td>
<td>0.0772</td>
<td>0.0693</td>
</tr>
<tr>
<td>100</td>
<td>0.0718</td>
<td>0.0788</td>
<td>0.0793</td>
<td>0.0808</td>
<td>0.0831</td>
<td>0.0831</td>
<td>0.0836</td>
<td>0.085</td>
<td>0.0807</td>
<td>0.0766</td>
</tr>
<tr>
<td>50</td>
<td>0.0743</td>
<td>0.0796</td>
<td>0.0824</td>
<td>0.0827</td>
<td>0.0828</td>
<td>0.0829</td>
<td>0.085</td>
<td>0.085</td>
<td>0.0818</td>
<td>0.0787</td>
</tr>
</tbody>
</table>

According to Henderson, Jacobson & Johnson (2003) the initial temperature is crucial within simulated annealing. If it is too low at the beginning, it can get trapped in an inferior solution. If it is too high, it can waste too many iterations (computing time) by accepting too many hill-climbing moves. This is supported by Johnson et al. (1989), as it may not be necessary to spend too much time at very high temperature levels. During the numerical experimentations, the final optimal solution for a run was almost always found during first temperature levels and during later temperature levels, this solution was not improved anymore. It might be that the initial temperature is too low, such that it gets trapped in an inferior solution. However, the lack of finding (reproducible) near-optimal results could be attributed more to the limited annealing length (and only exploring small parts of the search space) than to the initial temperature. Varying the initial temperature did not show any improvements and the temperatures used did have high enough acceptance probabilities of accepting worse solution in early stages of the algorithm.

### 6.4 Application on NedTrain Test case

A logical next step would be to apply the gained insights from the test case to the VIRM data provided by NedTrain, so that the predictive value of events on critical events can be evaluated by applying the optimization processes on the parameters and the distance metric. Here, the possible critical events that are considered are in the form of EBK’s, stranded trains or severe diagnostic events. To apply the gained insights on this data set, a methodology is established which is presented in Appendix H.

Unfortunately, critical events in the form of EBKs or stranded trains were not possible to forecast by means of the Event Simulator, because there are too few individual events of one type, to properly train the patterns of relevant events, so it is certainly not possible to thereby divide the data into a training set and a monitoring set to evaluate the prediction accuracy. It makes no sense to make predictions about EBK’s or stranded trains in general, as each of these events have different causes. These events were therefore not taken into consideration for forecasting purposes. As the number of occurrences for several severe diagnostic events was sufficiently available, these were taken into account in order to predict these events by means of the Event Simulator. However, to use the Event simulator to predict a critical event, it should have some predecessors and there were no severe diagnostic events found that had any corresponding predecessors. Since the data of NedTrain became available in a later stage of the research this could not be seen beforehand.

Diagnostic events of all subsystems were taken into consideration and the severity of the events were based upon the class of the diagnostic fault code (codes of type A and B, corresponding to safety and performance messages), that had a sufficient amount of occurrences. Due to this last restriction already a large part of the events was disregarded. For events with a sufficient number of
occurrences no predecessors were found. These findings were in line with the conclusions drawn by FCE, who performed an analysis on sever diagnostic events of the traction subsystem. They also did not find events that could be forecasted based on other diagnostic event patterns. For another dataset, with events of another type of train (SLT-series), also no predecessors were found for sever critical events that could be used in the Event Simulator. Here, only the error codes ATB0409 (collective code for multiple automatic train control disruptions) and HSP01FF and HSP02FF (messages that respectively HBU1 and HBU2 are inoperable, who convert high voltage into low voltage) were taken into account.

In the case of NedTrain, based on the analysis performed, it can be concluded that sever diagnostic events are not preceded by other diagnostic events, since no corresponding predecessors were found by the method used in the Event Simulator. The Event Simulator has shown that it was able to predict critical events in the test case data, using the current determination of predecessors. For NedTrain, it seems that the diagnostic events do not contain predictive value for other, more critical events.
7 Conclusion and Discussion

The main goal of this thesis, as formulated in Chapter 2, was to see if the prediction accuracy of the Event Simulator could be improved by applying a weighted distance metric instead of the Euclidean distance metric. Based on the evaluation of the test case in Chapter 6, it is concluded that applying a weighted distance metric can indeed improve the prediction accuracy substantially.

To evaluate the effects of the application of a weighted distance metric, simulated annealing (SA) was applied to find an optimal set of weights for this distance metric. These weights give relative importance to diagnostic events used to predict the occurrence of a critical event (e.g. failure, malfunction or breakdown). As a benchmark, the Euclidean distance metric was used, which corresponds with equal weights of diagnostic events. In order to apply the SA heuristic to the optimization problem, a methodology was established (see Chapter 5), by defining the four elements of Press et al. (1996) for simulated annealing. This methodology was implemented as an extension in the Event Simulator tool. Besides the weighted distance metric, also the influences of other parameters that affect the prediction accuracy were taken into account. It was evaluated how the best prediction accuracy could be obtained by varying the parameters step size (SS), embedding window (EW) and CDF threshold.

The main findings of this research are elaborated in Section 7.1. The limitations on the main findings of this research are briefly discussed in Section 7.2, followed by recommendations for further research in Section 7.3. For practical purposes, a set of guidelines (steps) is presented in Appendix H, which can be used to obtain the best results for the prediction of any particular critical event.

7.1 Conclusions

(1) A weighted distance metric can substantially improve the prediction accuracy

Improvements in the prediction accuracy were found for a weighted distance metric up to 27% compared to the Euclidean distance metric, where all diagnostic events receive the same weight in the prediction of a critical event. Although the results were not reproducible, as is discussed in conclusion (2), it still shows the potential gain that can be achieved by using weights in the distance metric. Furthermore, in almost every performed simulated annealing run the prediction accuracy was improved to some extent, regardless of the used settings.

(2) SA is not able to find optimal and reproducible solutions within reasonable computation time

The quality of the weights found in terms of the prediction accuracy varied to a large extent and this quality is dependent on which regions in the search space are explored. Due to the limited annealing lengths, caused by the large computation time involved, large improvements in the prediction accuracy were only found sporadically. Even with computation times over eight hours no clear optimal and reproducible result was found. These findings are in line with literature on finding optimal solutions using simulated annealing. According to Ali (1997), SA can be seen as a randomization device, which by means of an acceptance/rejection criterion allows some ascent steps during the optimization. It is therefore quite possible that the procedure visits close neighbors of the optimal solution, but due to the acceptance/rejection mechanism it leaves this region and proceeds with worse solutions. The role of the computation time is also mentioned by Dekker & Aarts (1991), who state that there is no guarantee that the SA algorithm will eventually succeed in finding a global optimum, since it is dependent on the number of temperature levels and the carefulness by which the control parameter is decreased. A high number of temperature levels and
slow cooling of the control parameter affects the efficiency and therefore a compromise has to be made between effectiveness and efficiency (Dekker & Aarts, 1991). When the Event Simulator is used in a real-time CBM setting, the role of computation time is less crucial, as the optimization then only takes place during initialization and possibly on a timely basis to verify if the settings are still optimal. Therefore, it is not necessary to have optimization heuristics that find solutions within a couple of minutes, but it should be able to find optimal and reproducible results within the eight hours computation time used in this research for the SA algorithm.

3) The choice of the EW, SS and CDF threshold is crucial in terms of the prediction accuracy
Besides the influence of the distance metric, also other parameters were varied in order to evaluate their impact on the prediction accuracy. For the used test case, the results of optimizing the other parameters EW and SS showed that the quality of forecasting a critical event is highly dependent on the set of parameters used. The optimal set of parameters performed on average 50% better than combinations of parameters outside a so called robust optimal region. There exists a robust optimal region around the optimal combination in which the prediction accuracy is near optimal. Within this area the prediction performance is comparable to the optimal combination of parameters, while outside this area this performance decreases quickly. Furthermore, extreme high or low values for both EW and SS should be avoided. The optimal CDF threshold is foremost dependent on the focus of the objective function in terms of false positives and false negatives, but also has large impact on the prediction accuracy.

7.2 Limitations
Due to the lack of predictive value of diagnostic events to critical events, the gained insights could not be tested on the data of NedTrain. Furthermore, the improved results found in the test case with optimized parameters and a weighted distance metric could not be compared to improvements found for another case, to see if the results were comparable. Therefore, the evaluation of the parameters and the distance metric are only based on one single case and although it gives strong indications about the possible improvements, it should be tested on other cases to make any strong conclusions about the role of the distance metric and the other parameters. Although the tool could not make prediction for critical events within the data of NedTrain, the tool have shown to have high prediction accuracy if the events do have predictive value.

7.3 Recommendations
Since substantial improvements were found for a weighted distance metric compared to the Euclidean distance metric, the role of the distance metric cannot be neglected and further research should be performed on how this metric can be optimized better. Other heuristics might be tested in order to find more reproducible results. For example, taboo search might be considered. Chelouah & Siarry (2000) state that this algorithm outperforms simulated annealing in terms of computation times. However, these results were obtained for lower dimension problems \( n < 10 \), while in the case of the similarity sampling the number of predecessors that corresponds with the number of dimensions are often higher (equal to 17 in the test case). Another possible heuristic that might be applied, but was not yet addressed in this thesis is the gradient descent algorithm. For this heuristic, only absolute functions that are differentiable can be used, such that the objective function to measure the prediction accuracy should be a forecasting error based-measure, like the sum of squared errors. With this heuristic less computation time is involved and is it able to work in spaces
of any dimension. Since this algorithm requires a forecasting error based-measure it was not taken into account in this research less interpretable than warning-based functions.

The Event Simulator as described makes use of diagnostic events to predict critical events. For NedTrain, earlier analysis within the RTM project revealed that sensor data can contain predictive value for upcoming critical events. The Event Simulator with the similarity sampling method can also be used for sensor data in broadly the same manner as for event data, but in the version used this was not incorporated yet. Instead of a state vector containing number of occurrences of relevant events, the state vector represents the (average) value of relevant sensors. It is also possible to use both event data and sensor data as input. The role of the distance metric and the other parameters remains the same. Sensor data is variable data, while the occurrences of events is discrete data and according to Jardine et al. (2006) variable data usually outperforms discrete data in a CBM program to represent the actual state of a system. In newer versions of the tool, the use of sensor data is incorporated and first analysis on NedTrain data by FCE showed potential forecast possibilities by using sensor data on sever diagnostic events. In further research the use of sensor data as a predictor of critical events can be analyzed for NedTrain data. Here, the weighted distance metric should also be taken into account, together with the optimization of the other parameters.
References


De Jonge, T. M. (2014). The facility location problem applied to a train maintenance environment.


Appendix A: Notations and definitions of the initial model description

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>Critical event</td>
</tr>
<tr>
<td>$E$ is defined as an unexpected arrival or stranded train or a severe diagnostic fault code itself.</td>
<td></td>
</tr>
<tr>
<td>$h$</td>
<td>Forecast Horizon</td>
</tr>
<tr>
<td>The forecast horizon represents the period for which the prediction is made if a critical event is likely to occur.</td>
<td></td>
</tr>
<tr>
<td>$l$</td>
<td>Embedding time window length</td>
</tr>
<tr>
<td>The window length is a moving time window in which the occurrences for each diagnostic fault code are counted.</td>
<td></td>
</tr>
<tr>
<td>$s$</td>
<td>Step Size</td>
</tr>
<tr>
<td>The step size determines the time interval at which state vectors are created.</td>
<td></td>
</tr>
<tr>
<td>$u$</td>
<td>Time index in training period</td>
</tr>
<tr>
<td>The index in the training period is denoted with $u$ to distinguish state vectors from this period to the monitoring period.</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>Time index in monitoring period</td>
</tr>
<tr>
<td>The index in the monitoring period is denoted with $t$ to distinguish state vectors from this period to the training period.</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>Training Period</td>
</tr>
<tr>
<td>The training period is defined as the time span in which observations (state vectors) have a known forward time.</td>
<td></td>
</tr>
<tr>
<td>$M$</td>
<td>Monitoring Period</td>
</tr>
<tr>
<td>The monitoring period is defined as the time span in which actual observations (state vectors) are used to forecast the forward time for these observations and to compare this forecast with the actual situation.</td>
<td></td>
</tr>
<tr>
<td>$x_u$</td>
<td>State vector at time $u \in T$</td>
</tr>
<tr>
<td>The state vectors of the training period contain the number of occurrences per relevant diagnostic fault code (predecessor) at a certain time $u$, given a window length of $l$ and have a known forward time.</td>
<td></td>
</tr>
<tr>
<td>$x_t$</td>
<td>State vector at time $t \in M$</td>
</tr>
<tr>
<td>The state vectors in the monitoring period contain the number of occurrences per relevant diagnostic fault code (predecessor) at a certain time $t$, for which a forward time distribution is empirically forecasted.</td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td>Length of state vector</td>
</tr>
<tr>
<td>The length of the state vector is determined using the cross correlation function between a critical event and all other events, such that only relevant events are taken into in the state space.</td>
<td></td>
</tr>
<tr>
<td>$f_{E,u}$</td>
<td>Forward time at time $u \in T$</td>
</tr>
<tr>
<td>Each state vector from the training period has a known forward time. This is the time until the critical event.</td>
<td></td>
</tr>
<tr>
<td>$f_{E,t}$</td>
<td>Forward time at time $t \in M$</td>
</tr>
<tr>
<td>Each state vector from the monitoring period has a known forward time. This is the time until the critical event and can be compared to the expectation of the empirical forward time.</td>
<td></td>
</tr>
<tr>
<td>$\hat{f}_{E,t}$</td>
<td>Empirical forward time at time $t \in M$</td>
</tr>
<tr>
<td>For an actual observation (state vector) in the monitoring period an empirical forward time distribution is established based on the forward times of state vectors in the neighbourhood in the training period. Together with a threshold is it determined if a warning is given (a critical event is expected within the forecast horizon)</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>CDF threshold</td>
</tr>
<tr>
<td>When the above described empirical distribution (CDF) exceeds this threshold within the forecast horizon, the critical event is expected within the forecast horizon.</td>
<td></td>
</tr>
</tbody>
</table>
The neighbourhood of a state vector in the monitoring period consist of similar state vectors from the training period, such that $N(t) \subseteq T$.

**$r_{EF} := \text{Cross correlation function of critical event } E \text{ and arbitrary event } F$**

The cross correlation function is used to determine relevant dimensions, e.g. diagnostic fault codes, and therefore determines the length of the state vector, e.g. the state space. A high mass of the function $r_{EF}$ indicates that event $E$ is likely to occur after event $F$. The relevant dimensions are called the predecessors, but this part is out of scope in the further research.
Appendix B: Flowchart of the initial model description

Event Simulator

Preprocessing and set up
- Determine length of embedding window and step size
- Determine training period and monitoring period
- Preprocess input data set

Training Set
- Create Cross-Correlation functions
- Find relevant events based on cross correlation functions
- Create state vectors based on relevant events
- Determine forward time for each state vector in training period

Monitoring Set
- Create monitoring set, in which each actual observation is a state vector with relevant events
- Find similar set of observations in training period based on the distance between state vector in monitoring period and state vector in training period using a distance measure
- Give weight to similar observations using a weight function
- Determine empirical forward time distribution of actual observation based on forward times of similar points

Optimization Routine
- Adjust input parameters (embedding window and step size), distance measure and weight function to minimize expected squared forecasting error

Predictive Model
- Determine expected squared forecasting error
- Accept?
  - Yes
  - No
Appendix C: Example of an application of simulated annealing: TSP

In order to show the use of simulated annealing, its application for optimizing the Traveling Salesman Problem is provided, which is often referred to in literature, as it is very a suitable application. The Traveling Salesman Problem (TSP) reflects the problem in which the traveller needs to visit \( n \) cities, where the distance between each pair of cities is known. The goal is to find the shortest route that visits all the cities at once. This can be classified as a NP-hard problem, as when the number of cities increases linearly, the time to solve this problem will increase exponentially. For \( n = 80 \) it will take millions of years to solve this problem exhaustively. It also belongs to a class of minimization problems for which the objective function will have many local minima and SA is therefore very suitable. Above four elements can be applied on the TSP problem in the following way (Press et al., 1996):

1. **Configuration:** The configuration in the TSP is in the form of a permutation \( 1 \ldots N \), which can be interpreted as the order in which the cities are visited.

2. **Changes in configuration:** Neighbours of a current configuration can be established by doing (random) rearrangements to the current sequence in which cities are visited, by:
   a. Removing a section of the path and replace it with the same cities in the opposite order.
   b. Removing a section of the path and replace it between two cities on another randomly chosen part of the path.

3. **Objective Function:** The function that is minimized can be the total length of the route, by making use of the known distance between each pair of cities or its coordinates. However, this might be extended by extra penalty costs, for example when a river is crossed. The method is therefore very flexible as there are no restrictions in establishing the target function.

4. **Annealing schedule:** First a starting value for the temperature parameter should be chosen, which should be much larger than the largest difference in the objective function normally encountered when doing rearrangements. Then this value should be slowly decreased (which can be done in multiple ways), such that the Metropolis criterion becomes more strictly in accepting worse solution over time.

When this approach is compared to the more basic local search techniques, getting stuck in local optimum is avoided by making use of the metropolis criterion to accept inferior solutions. When an appropriate annealing schedule is chosen, the probability is high that the simulated annealing technique will converge to an optimal solution.
Appendix D: Implementation of simulated annealing

All of the elements for the simulated annealing methodology were defined in Chapter 5, using the Metropolis procedure for continuous optimization problems which was explained in 4.4.3. In order to implement the methodology within the FCE Event Simulator Tool a flowchart is established (See Appendix E). This flowchart contains all the elements and the sequence of how these elements are preceded. Furthermore it shows all the appropriate links between these elements and their corresponding loops. C# language is used in order to explain the process flows and the corresponding loops, as well as the calculations within the steps. The total flowchart can be found in Appendix E and is discussed in three steps (initialization, nested loops and establishment and assessment of reconfigurations) in which parts of the flowchart are being discussed in more detail. Figure 8 is used, together with the steps in the developed SA methodology pseudo code in Chapter 5. The source code can be found in Appendix F.

Initialization

When the predecessors are established, the length of the weight vector $A$ is known. This weight vector is set to equal weights and is therewith set as the optimal weight vector so far. This initial vector is then used to calculate the initial value for the target function. By looking back to Figure 8 it can be seen that for calculating the value of the target function first the complete similarity sampling procedure is followed wherein also the CDF threshold may be optimized. For the purpose of simulated annealing it is now only necessary to know that when the function $OptimizationCDF(i)$ is being called with the current embedding window, step size, weight vector and predecessors the value of the target function with these parameters is returned. Now the initial temperature value can be calculated using the $CFactor$. These steps are shown together in Figure 21 and comprise the first three steps of the proposed SA methodology pseudo code that were presented in Section 5.3.

Nested Loops

After the initialization steps described above the actual simulated annealing can start. The annealing can be modelled with a loop within a loop, a nested loop, where the first loop regulates the temperature level, while the second loop organizes the reconfigurations that are performed within each temperature level. The second loop is executed $nSteps$ times and after each execution the temperature is lowered using the $TFactor$. During each execution $nOver$ reconfigurations are done, or when $nSucc$ equals $nLimit$, this second loop is interrupted prematurely. The content of the second loop is here shown as a black box, as the goal here is to present the double loop, but is discussed in more detail hereafter. During the reconfigurations, the best weight vector $A^*$ so far,
which has the lowest value for the target function \(TF(A^*)\) is saved and when both loops are completely passed, the optimal weight vector can be returned. This nested loop of reconfigurations within the loop of the cooling down process is shown in Figure 22 and represents step 4 until step 8 (with step 5 as black box) of the proposed SA methodology pseudo code that was presented in Chapter 5.

![Figure 22 Nested loops of reconfigurations and cooling down](image)

**Establishment and assessment of reconfigurations**

The loop that executes the reconfigurations (step 5 of the SA methodology), can be differentiated into two separate parts. The first part contains the establishment of random changes in the reconfiguration space as described in Section 5.4 and the second part covers the assessment of each reconfiguration using the metropolis criterion. In order to determine which type of reconfiguration is executed, a random number \(dec\) is drawn from between 0 and 1 and based on this number one of the four movements is executed. If the value is smaller or equal than 0.5 a small one dimensional reconfiguration is performed, if the value is greater than 0.5 but smaller than 0.85, a small multi-dimensional reconfiguration is executed. Big reconfigurations in one dimension are done if the value of the random number \(dec\) is greater than 0.85 but smaller than 0.95. For values above 0.95 big, reconfigurations in multiple dimensions is conducted. Using this way of working the described occurrence probabilities are implemented. As can be seen in Figure 23, each of the four movements is incorporated within a separate function, that all return the random reconfigured weight vector \(A'\) based on the type of movement. For this new vector the function \(OptimizationCDF()\) can be invoked in order to return the result of the objective function. The functions \(MoveSingleSmall()\) and \(MoveSingleBig()\) are identical functions except for the factor that is used in establishing the shift, which are 0.1 and 0.5 respectively. This also applies for the functions \(MoveMultipleSmall()\) and \(MoveMultipleBig()\).
After the establishment of the random change, it is assessed. This assessment is shown in Figure 24. After each made reconfiguration $A'$, the result of the objective function is compared to the current configuration $A$. If the weight vector $A'$ has a lower result than the current weight vector, then the current $A$ is being set to $A'$. If the result is higher, but the metropolis criterion is fulfilled, the solution also is accepted, which means that for the assessment of the next reconfiguration this $A'$ is used. So $A$ is set to $A'$ and $TF(A)$ to $TF(A')$. $A^*$ should always contain the optimal solution, so this solution is only updated if the new reconfiguration is better than the optimal solution so far. This means that it is only updated if $TF(A') < TF(A^*)$. So important to note is that vector $A$ to which a reconfigured vector $A'$ is compared, is not necessarily the optimal vector $A^*$. It may also be a less optimal vector that was accepted by the metropolis criterion. This is one of the applications of the simulated annealing algorithm, in order to get out of local minima by jumping through the state space even if locally the optimization problem seems to be optimal. The number of reconfigurations during a temperature level is determined by the definition of the parameter $nOver$. However, it might also be possible that the reconfigurations can be stopped prematurely if the number of successful reconfigurations $nSucc$ exceeds a certain predefined limit $nLimit$. A successful reconfiguration is defined as a reconfiguration that results in a lower objective value than the current configuration in use, or if it is accepted by the metropolis criterion. If the assessment results in one of these two results, the parameter $nSucc$ is increased by one, so if its limit is reached it can break the current temperature level.

Figure 23 Generator of random changes in the configuration

Figure 24 Flowchart of accepting solutions (with metropolis criterion)
Appendix E: Flowchart of the Simulated Annealing Optimization

1. Set \( A = I \)
2. Calculate \( T(F(A)) \)
3. Calculate \( \tau : \tau = \text{CFACTOR} \times T(F(A)) \)
4. For \( (i = 0; i < \text{nSteps}; i++) \)
   - \( nSucc = 0 \)
   - For \( (k = 0; k < \text{nOver}; k++) \)
     - Generate random number: \( \text{dec} = \text{rand}(0,1) \)
     - \( \text{dec} \leq 0.5? \)
       - \( \text{true} \)
       - \( \text{MoveSingleSmall} : \text{Make small change to one dimension} \)
       - \( \text{false} \)
       - \( \text{MoveMultipleSmall} : \text{Make small change to all dimensions} \)
     - \( \text{false} \)
     - \( \text{MoveSingleBig} : \text{Make large change to one dimension} \)
     - \( \text{true} \)
     - \( \text{MoveMultipleBig} : \text{Make large change to all dimensions} \)
   - \( \text{true} \)
     - \( \text{Set} : \text{nSucc} = \text{nSucc} + 1 \)
     - \( \text{Set} : \text{ans} = 1 \)
     - \( \text{Set} : \text{TF}(A) = \text{TF}(A') \)
     - \( \text{Set} : A = A' \)
   - \( \text{false} \)
     - \( \text{Set} : \text{ans} = 0 \)
     - \( \text{Set} : \text{TF}(A) = \text{TF}(A) \)
     - \( \text{Set} : A = A \)
   - \( \text{true} \)
     - \( \text{Set} : \text{ans} = 1 \)
     - \( \text{Set} : \text{TF}(A) = \text{TF}(A') \)
     - \( \text{Set} : A = A' \)
5. \( \text{true} \)
   - Change \( \tau : \tau = \text{TFACTOR} \times \tau \)
   - \( \text{false} \)
6. \( \text{Return} A^{*} \)
Flowchart of sub process: “Move Single Small”

1. Calculate \( z = \text{CFAC} \times T \).

2. Generate random number: \( \text{dec} = \text{rand}(1,0) \)

   - if \( \text{dec} \leq 0.57 \)
     - \( \text{MoveSingleSmall}: \text{Make small change to 1 dimension} \)
   - if \( \text{dec} \leq 0.85 \)
     - \( \text{MoveMultipleSmall}: \text{Make small change to all dimensions} \)
   - if \( \text{dec} \leq 0.95 \)
     - \( \text{MoveSingleBig}: \text{Make large change to 1 dimension} \)

3. \( A' = A + \Delta A \)

4. Generate random dim. to add:
   - \( \text{indexAdd} = \text{rand}(0,A.\text{Length}) \)

5. \( \text{Generate random dim. to subtract:} \)
   - \( \text{indexSubtract} = \text{rand}(0,A.\text{Length}) \)
   - \( \text{indexAdd} == \text{indexSubtract} \)

6. \( j = 0; j < A.\text{length}; j++ \)

7. \( A'[\text{indexSubtract}] = 0.0 \)
   - \( A'[\text{indexAdd}] == A[\text{indexSubtract}] \)
   - \( A'[\text{indexSubtract}] = \text{shift} \)
   - \( A'[\text{indexAdd}] == \text{shift} \)

8. Return \( A'[\text{}\]
Flowchart of sub process: “Move Multiple Small”

Generate random number:
\[ \text{dec} = \text{rand}(1,0) \]

- \( \text{dec} \leq 0.5 \) true
  - \( \text{MoveSingleSmall: Make small change to 1 dimension} \)
  - \( \text{MoveMultipleSmall: Make small change to all dimensions} \)
- \( \text{dec} \leq 0.85 \) true
  - \( \text{MoveMultipleSmall: Make small change to all dimensions} \)
- \( \text{dec} \leq 0.95 \) true
  - \( \text{MoveSingleSmall: Make small change to 1 dimension} \)
- \( \text{dec} \leq 0.5 \) false
  - \( \text{MoveMultipleSmall: Make small change to all dimensions} \)
  - \( \text{MoveMultipleBig: Make large change to all dimension} \)

\[ j = 0; \quad j < A.\text{length}; \quad j++ \]

\[ A'[i] = A[i] \]

\[ j = 0; \quad j < A.\text{length}; \quad j++ \]

\[ \text{Generate random shift: } \text{shift} = \text{rand}(1,0) \times A.\text{length} * 0.1 \]

\[ A[i] = \text{shift} < 0 \]

\[ \text{true} \rightarrow A'[i] = \text{shift} - A[i] \]
\[ \text{false} \rightarrow A'[i] = \text{shift} \]

\[ j = 0; \quad j < A.\text{length}; \quad j++ \]

\[ \text{Get sum of all dimensions: } \text{norm} \leftarrow A'[j] \]

\[ j = 0; \quad j < A.\text{length}; \quad j++ \]

\[ \text{Give relative weight: } A'[i] = A'[j] \times A.\text{length} / \text{norm} \]

\[ \text{Return } A'[\]
Flowchart of sub process: “Move Single Big”

- **MoveSingleSmall**: Make small change to 1 dimension
- **MoveMultipleSmall**: Make small change to all dimensions
- **MoveSingleBig**: Make large change to 1 dimension
- **MoveMultipleBig**: Make large change to all dimensions

Generate random number: $\text{dec} = \text{rand}(1,0)$

- $\text{dec} \leq 0.57$: MoveSingleSmall
  - $\text{dec} \leq 0.85$: MoveMultipleSmall
  - $\text{dec} \leq 0.95$: MoveSingleBig
- $\text{dec} > 0.57$: MoveMultipleBig

Generate random number to add: $\text{indexAdd} = \text{rand}(0, \text{A.Length})$

Generate random number to subtract: $\text{indexSubtract} = \text{rand}(0, \text{A.Length})$

- $\text{IndexAdd} = \text{IndexSubtract}$: true
  - $\text{indexAdd} = \text{indexSubtract}$
  - $\text{indexAdd} = \text{indexSubtract}$

$\text{A'}[\text{indexAdd}] += \text{shift}$

$\text{A'}[\text{indexSubtract}] -= \text{shift}$

Return $\text{A'}$
Flowchart of sub process: “Move Multiple Big”

Generate random number:
dec = rand(1,0)

dec ≤ 0.5?
true
false

dec ≤ 0.85?
true
false

dec ≤ 0.95?
true
false

MoveSingleSmall:
Make small change to 1 dimension

MoveMultipleSmall:
Make small change to all dimensions

MoveSingleBig:
Make large change to 1 dimension

MoveMultipleBig:
Make large change to all dimensions

j = 0; j < A.length; j++

A'[j] = A[j]

Generate random number:
shift = rand(1,0) * A.length * 0.5

A[j] - shift < 0?
true
false

A'[j] = shift * A[j]

j = 0; j < A.length; j++

Get sum of all dimensions:
norm += A'[j]

j = 0; j < A.length; j++

Give relative weight:
A'[j] = A'[j] * A.length / norm

return A'[]
Appendix F: Source Code of the Simulated Annealing Optimization

```csharp
public double[] SimulatedAnnealingOptimization(Tuple<string, string> evTuple, int embeddingWindow, int stepSize, bool isTraining, TrainingPreferences preferences)
{
    int nover = 100;
    int nlimit = 10;
    int nSteps = 10;
    int nsucc;
    int ans;
    double objective;
    double objectiveP = 0.0;
    double objectiveOptimal = 0.0;
    double CFACTOR = 0.05;
    double TFACTOR = 0.95;
    double tau = 0.0;
    double dec = 0.0;
    double[] Z = new double[nSteps];

    var ponderation = new double[_regression.DataPatterns.First().BwPattern.Matrix.GetLength(1)];
    for (int j = 0; j < ponderation.Length; j++)
    {
        ponderation[j] = 1.0;
    }

    objective = OptimizationCDF(evTuple, isTraining, embeddingWindow, stepSize, ponderation, preferences, false);
    ponderationOptimal = ponderation
    objectiveOptimal = objective;
    tau = CFACTOR * objective;
    for (int i = 0; i < nSteps; i++)
    {
        nsucc = 0;
        for (int k = 0; k < nover; k++)
        {
            dec = _randObj.NextDouble();
        }
    }
```

66
double[] ponderationNew;
if (dec <= 0.5)
    ponderationNew = MoveSingleSmall(ponderation);
else if (dec <= 0.85)
    ponderationNew = MoveMultipleSmall(ponderation);
else if (dec <= 0.95)
    ponderationNew = MoveSingleBig(ponderation);
else
    ponderationNew = MoveMultipleBig(ponderation);

objectiveP = OptimizationCDF(evTuple, isTraining, embeddingWindow, stepSize, ponderationNew, preferences, false);
ans = 0;
if (objectiveP < objectiveOptimal)
{
    objectiveOptimal = objectiveP;
    ponderationOptimal = ponderationNew;
}
if (objectiveP < objective || _randObj.NextDouble() < Math.Exp(-(objectiveP - objective) / tau))
    ans = 1;
if (ans == 1)
{
    nsucc++;
    objective = objectiveP;
    ponderation = ponderationNew;
}
if (nsucc == nlimit)
    break;

Z[i] = objectiveOptimal;
tau *= TFACTOR;
    OnProgressUpdate(nSteps, i, "Optimization", Z);
}
return ponderationOptimal;
Source code of sub method: "Move Single Small"

```csharp
private double[] MoveSingleSmall(double[] ponderation)
{
    int indexAdd = _randObj.Next(ponderation.Length);
    int indexSubstract = _randObj.Next(ponderation.Length);

    while (indexAdd == indexSubstract)
    {
        indexSubstract = _randObj.Next(ponderation.Length);
    }

    var ponderationNew = new double[ponderation.Length];
    for (int j = 0; j < ponderation.Length; j++)
        ponderationNew[j] = ponderation[j];

    double shift = (_randObj.NextDouble()) * ponderation.Length * 0.1;

    if (ponderation[indexSubstract] - shift < 0)
    {
        ponderationNew[indexSubstract] = 0.0;
        ponderationNew[indexAdd] += ponderation[indexSubstract];
    }
    else
    {
        ponderationNew[indexSubstract] -= shift;
        ponderationNew[indexAdd] += shift;
    }

    return ponderationNew;
}
```
Source code of sub method: “Move Multiple Small”

```csharp
private double[] MoveMultipleSmall(double[] ponderation)
{
    double[] ponderationNew = new double[ponderation.Length];
    for (int j = 0; j < ponderation.Length; j++)
    {
        ponderationNew[j] = ponderation[j];
    }

    for (int j = 0; j < ponderation.Length; j++)
    {
        double shift = (_randObj.NextDouble()) * ponderation.Length * 0.1;

        if (ponderation[j] - shift < 0)
            ponderationNew[j] = shift - ponderation[j];
        else
            ponderationNew[j] -= shift;
    }

    double norm = 0.0;
    for (int j = 0; j < ponderation.Length; j++)
    {
        norm += ponderationNew[j];
    }

    for (int j = 0; j < ponderation.Length; j++)
    {
    }

    return ponderationNew;
}
```
Source code of sub method: “Move Single Big”

```csharp
private double[] MoveSingleBig(double[] ponderation)
{
    int indexAdd = _randObj.Next(ponderation.Length);
    int indexSubstract = _randObj.Next(ponderation.Length);

    while (indexAdd == indexSubstract)
    {
        indexSubstract = _randObj.Next(ponderation.Length);
    }

    var ponderationNew = new double[ponderation.Length];
    for (int j = 0; j < ponderation.Length; j++)
    {
        ponderationNew[j] = ponderation[j];
    }

    double shift = (_randObj.NextDouble()) * ponderation.Length * 0.5;

    if (ponderation[indexSubstract] - shift < 0)
    {
        ponderationNew[indexSubstract] = 0.0;
        ponderationNew[indexAdd] += ponderation[indexSubstract];
    }
    else
    {
        ponderationNew[indexSubstract] -= shift;
        ponderationNew[indexAdd] += shift;
    }

    return ponderationNew;
}
```
Source code of sub method: “Move Multiple Big”

```csharp
private double[] MoveMultipleBig(double[] ponderation)
{
    double[] ponderationNew = new double[ponderation.Length];
    for (int j = 0; j < ponderation.Length; j++)
        ponderationNew[j] = ponderation[j];

    for (int j = 0; j < ponderation.Length; j++)
    {
        double shift = (_randObj.NextDouble()) * ponderation.Length * 0.5;

        if (ponderation[j] - shift < 0)
            ponderationNew[j] = shift - ponderation[j];
        else
            ponderationNew[j] -= shift;
    }

    double norm = 0.0;
    for (int j = 0; j < ponderation.Length; j++)
        norm += ponderationNew[j];

    for (int j = 0; j < ponderation.Length; j++)

    return ponderationNew;
}
```
Appendix G: Effects of the temperature level and the annealing schedule

To show the function of the temperature level and the effects of the parameters in the Metropolis criterion a few scenarios of numerical examples are considered for which the used parameters are shown in Table 15.

Table 15 Parameters of numerical examples Metropolis criterion

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( TF(i) )</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>( CFACTOR )</td>
<td>0.05</td>
<td>0.10</td>
<td>0.05</td>
<td>0.10</td>
</tr>
<tr>
<td>( TFACTOR )</td>
<td>0.8</td>
<td>0.8</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>( nSteps )</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

In these examples, the initial objective value is assumed to be equal to 100. In the first scenario a \( CFACTOR \) of 5\% results in an initial temperature level \( \tau_0 \) of 5. This level is reduced according to a \( TFACTOR \) by 20\%. In the second scenario the \( CFACTOR \) is increased to 10\% such that the initial temperature level \( \tau_0 \) becomes 10. In the third scenario the \( TFACTOR \) has changed to 0.90 in comparison to the 0.8 in the first scenario. In the last scenario both changes to \( CFACTOR \) and \( TFACTOR \) are adopted.

By using the discussed method of initiating the temperature level and using the provided annealing schedule, the temperature levels for each of the scenarios are calculated, which are presented in Table 16.

Table 16 Resulting temperature levels of numerical example Metropolis criterion

<table>
<thead>
<tr>
<th>Scenario</th>
<th>( \tau_1 )</th>
<th>( \tau_0 )</th>
<th>( \tau_2 )</th>
<th>( \tau_3 )</th>
<th>( \tau_4 )</th>
<th>( \tau_5 )</th>
<th>( \tau_6 )</th>
<th>( \tau_7 )</th>
<th>( \tau_8 )</th>
<th>( \tau_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>5.00</td>
<td>4.00</td>
<td>3.20</td>
<td>2.56</td>
<td>2.05</td>
<td>1.64</td>
<td>1.31</td>
<td>1.05</td>
<td>0.84</td>
<td>0.67</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>10.00</td>
<td>8.00</td>
<td>6.40</td>
<td>5.12</td>
<td>4.10</td>
<td>3.28</td>
<td>2.62</td>
<td>2.10</td>
<td>1.68</td>
<td>1.34</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>5.00</td>
<td>4.50</td>
<td>4.05</td>
<td>3.65</td>
<td>3.28</td>
<td>2.95</td>
<td>2.66</td>
<td>2.39</td>
<td>2.15</td>
<td>1.94</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>10.00</td>
<td>9.00</td>
<td>8.10</td>
<td>7.29</td>
<td>6.56</td>
<td>5.90</td>
<td>5.31</td>
<td>4.78</td>
<td>4.30</td>
<td>3.87</td>
</tr>
</tbody>
</table>

These temperature levels each have their own probability distribution for accepting a decrease in the target function and are shown together per scenario in Figure 25. For each individual scenario, the development of the acceptance probability distribution can be obtained, when the temperature level \( \tau \) is decreased. Of course, for each scenario the acceptance probability distributions decrease as the temperature levels become smaller. The initial probability distribution for the initial temperature level \( \tau_0 \) is decreased. Of course, for each scenario the acceptance probability distributions decrease as the temperature levels become smaller. The initial probability distribution for the initial temperature level depends on the \( CFACTOR \). Scenario 1 and 3 have the same starting distributions as do Scenario 2 and 4 since the \( CFACTOR \) is the same for these scenarios. The way in which these initial distributions develop over the temperature levels are dependent on the value of the \( TFACTOR \). By choosing a larger for \( TFACTOR \) value (i.e. a smaller percentage decrease \( \varepsilon \) after each temperature level), the distribution becomes less quick angular, such that higher decreases in the objective function have a greater probability of being accepted during the next steps.
Figure 25 Probability distributions for a accepting decrease in objective function of scenarios in numerical example

![Graphs showing probability distributions for different scenarios.](image)

**Figure 25** Probability distributions for a accepting decrease in objective function of scenarios in numerical example

**Acceptance probability: decrease of 2 in objective function**

![Graph showing acceptance probability for different scenarios.](image)

**Figure 26** Acceptance probability for a decrease of 2 in the objective function for the numerical examples

If the acceptance probability of a certain decrease in the objective function (for example a decrease of 2) is plotted over the temperature level \( \tau \), it can be seen how these probabilities develop over the temperature levels for each of the above introduced scenarios (Figure 26). Comparing the scenarios during the temperature level \( \tau_9 \), shows that about 36% of the cases in which a decrease of 2 is found in the objective function is still accepted in Scenario 3, while this is only about 5% in Scenario 1, while the initial probability was the same. This is due to the fact that the temperature in Scenario 3 is cooled down slower than in Scenario 1. Although the initial acceptance probability for Scenario 2
was higher than for Scenario 3, at level $\tau_9$ Scenario 2 has a lower probability of accepting a decrease of 2 in the objective function, as the temperature is faster lowered. Between $\tau_5$ and $\tau_6$, the lines intersect, so from this point the acceptance probability is lower for Scenario 2. The acceptance probability of Scenario 4 will always be higher than Scenario 2, since they have the same starting temperature level and Scenario 4 is cooled down slower. Eventually all the probabilities will go to approximately zero, but how fast is dependent on the starting probability and the level of annealing after each level. Of course, the acceptance probabilities will converge faster to zero for higher decreases in the objective function.

The speed of converging to a solution and the quality of this solution is thus dependent on the $\text{CFACTOR}$, $\text{TFACTOR}$, $\text{nSteps}$ and the number of reconfigurations within each temperature level ($\text{nOver}$). The quality of the solutions is thereby dependent on how the reconfigurations are established, the generator of random changes in the configurations, which is discussed Section 5.4. Accepting higher reductions in the objective function more often (slow decreasing acceptance probability), results in more often (and higher) uphill climbing, such that the probability of finding more global minima increases. However, this means that it will take more time, since more temperature levels are necessary to converge to a solution for which no better result of the objective function is accepted.
Appendix H: Guidelines for Event Simulator based on gained insights

Based on the evaluation of the test case, guidelines are established for using the Event Simulator for the prediction of any particular critical event. Each individual critical event that is forecasted using the Event Simulator should be treated individually as the use of optimal parameters is dependent on the event that is predicted. In Figure 27 the steps of the methodology containing the guidelines are summarized. The guidelines are divided into four steps: consisting of practical decisions that should be considered in advance, pre-analysis of the critical event, the optimization of parameters and finally, the verification of the prediction results. When the tool is used in a real-time setting an extra step might be necessary in order to recalculate parameters over time. In this section these subsequent steps are addressed. In Appendix I a user manual for the Event Simulator is presented which can be used together with this methodology.

**Figure 27 Methodology for applying the Event Simulator**

**Step 1: Practical decisions for initial settings**
In advance of the actual optimization steps, decisions have to be made about initial settings of the Event Simulator. These decisions involve how the prediction accuracy is calculated by means of the objective function and which forecast horizon should be used for the prediction. They should be based on practicality and are of paramount importance. When in later stages (e.g. after optimization of other parameters) it is decided that the prediction accuracy should be determined differently or another forecast horizon should be used, the whole optimization process has to be redone from start. Therefore, it is important to define these settings in advance.

*Definition of objective function*
Before the Event Simulator is used, one should first define the objective function. As described, this function consists of four individual prediction accuracy measures, which can be weighted in terms of the risk aversion of the stakeholder. The most important decision to make is whether false positives or false negatives should be penalized more. For critical systems, where high costs are involved with down time when the system is inoperable, it should be avoided that events are missed. For less critical systems and certainly for systems for which it is quite costly to take it out of operating mode, false warnings are not desirable. False warnings also affect the credibility of the prediction model.
The definition of the objective function is crucial in the optimization steps, as the optimization is based upon this function.

**Definition of forecast horizon**

Besides the objective function, also the forecast horizon should be defined in advanced. As this parameter determines on which upcoming time interval the warning is based, the stakeholder should decide how long in advance the potential warning should be given. Here, a decision should be made between a short and a long horizon. Using a short horizon, the warning is given just before the possible failure, but the time interval in which the potential failure might occur is more precise. For longer horizons, the failure is known in earlier stages, but less information is known about when exactly the failure will occur. For systems on which maintenance can be applied on short notice, short forecast horizons are beneficial, but if this is not the case and maintenance actions should be planned in advance it might be necessary to use a longer horizon.

**Step 2: Pre-analysis of critical event data**

There are some necessities regarding the input data, such that a critical event can be predicted using the Event Simulator. For the used test case there were no problems on this matter, but as is described later, for NedTrain it caused that no predictions could be made.

**Check number of occurrences**

The number of occurrences should be substantial, such that a training set and a monitoring set can be established. The training set should contain enough occurrences so that the model can be trained properly and also the monitoring set should be have enough data points, so that the learned patterns can be validated. If more events are available over a longer period of time, this will improve the prediction accuracy and furthermore, better corresponding predecessors can be found.

**Available corresponding predecessors**

Although the establishment of the predecessors is left out of scope during the current research, it has a crucial role in the Event Simulator. In order to predict a certain critical event, it should have some corresponding predecessors on which the prediction of the occurrence of a critical event is based. If the tool cannot find any correlating events, it is not possible to make any prediction at all. As the predecessors are determined by the combination of the EW and SS, these should already be varied to some extent. Some combinations might reveal predecessors that are not found using other combinations.

**Step 3: Optimization of parameters**

If there are no issues with regard to the number of occurrences of a critical event and the tool can find corresponding predecessors, the optimization steps can be performed in order to find the most optimal set of parameters that result in the best prediction accuracy. It is recommended to first optimize the parameters of the EW, the SS and the CDF threshold and subsequently perform the optimization of the distance metric on this optimal set of parameters.

**Optimization of embedding window, step size and CDF threshold**

In order to get the best prediction results the grids used for the optimization of the three parameters should be as large as possible. This involves using wide upper and lower boundaries and large grid sizes. As computation time necessary to optimize these parameters is not extensively and the results in terms of the prediction accuracy are enormous. However, if the computation time is limited one should decrease the computation time by decreasing the grid for the CDF threshold and
not of the EW and the SS. It has been shown that the optimal CDF threshold is high for objective functions in which false positives are avoided and low for functions with high penalties for false negatives. Therefore, some parts of the grid can be neglected in the optimization and the computation time can be halved by only taking into account high or low CDF thresholds, with still a high probability of finding the optimal combination of parameters.

Check robustness of optimal combination of parameters
When the most optimal combination is found, it should be checked if this combination is part of a robust optimal area. If so, the probability is high that for other observations the found optimal combination is also optimal. If not, it might be that for the particular used training and monitoring set, the combination found is an coincidental optimal and only applicable to the current data set. With a combination within a robust optimal area the parameter setting is more generalizable and avoids that the setting is over fitted to the used training and monitoring set. Therefore, an optimal combination should be overruled if it is not part of a robust area, by selecting a near-optimal solution that fulfills the requirement of robustness. In order to check the robustness, a surface area graph or a table can be applied as was used in Section 6.2.

Perform optimization of the distance metric on optimal parameter combination
For the found optimal parameters of the EW, SS and CDF threshold the optimization of the weighted distance metric should be performed subsequently. When for this purpose the simulated annealing methodology is used, the distance metric is always improved; however with limited annealing lengths there are large differences in the quality of the found weighted distance metrics. In order to set the parameters in the SA algorithm experimentation is required, but the following two general guidelines can be taken into account:

- It should be avoided that a too low initial temperature is used. The initial temperature level can better be set too high, although this will waste much computation time. By using a value for $nLimit$ that is 10% of the number of reconfigurations in a temperature level ($nOver$), it can be prevented that too much time is wasted in temperature levels that have a too high acceptance probability.
- Within limited computation time not all parameters can be set high, such that in theory the best result can be obtained. Two main options can be selected to tune the parameters. The first option is too use a high number of reconfigurations within a temperature level, but then the number of temperature levels should be set lower and the temperature should be decreased quicker. The other option is to use fewer reconfigurations within a temperature level and to increase the number of temperature levels and to decrease the temperature more gradually. This second option is recommended, since it gave the best results during the test case.

Step 4: Verification of results on prediction accuracy
The optimal weight found by the SA algorithm is based on the objective function and the corresponding results can be checked subsequently to evaluate the effects on the individual accuracy measures. If the result is satisfying, the weights can be applied, if not one can execute the SA algorithm with the same parameters or can refine the parameters trying to obtain better results. One can perform multiple runs with a same parameter setting trying to find the best weighted distance metric in terms of the prediction accuracy.
Step 5: time-based assessment on optimal parameter combination
When in the future the Event Simulator is used in real-time setting as a CBM program, it might be necessary to reevaluate the parameter setting on a timely basis. The previous optimization steps can then be repeated to verify if the current parameter setting is still optimal. If it is decided that initial parameters has to be changed, the whole process should be executed again.
Appendix I: User Manual of the Event Simulator

1. Selection of data set which is used to perform the prediction
2. Selection of methodology and type of data that is taken into account (here, we only focus on event data in supervised learning environment)
3. Assign vehicles to training/monitoring set, with the possibility to let some vehicle(s) only be used for training or vice versa.
4. Selection of subsystems to include in the prediction
5. Selection of training/monitoring period by selection of start date of training (upper date field) and start date of monitoring (lower date field)
6. Graphical representation of training/monitoring period, where green box represent the training period
7. Open additional project settings (via File > Edit Settings). Here also a new session can be started if necessary (via File > New Session)
8. Adjust the project settings, for example the forecast horizon
9. Start computation and go to next interface
1. Selection of critical event(s) to predict
2. Graphical representation of chosen critical event(s) classified in training/monitoring period
3. Selection of the optimization function, such that optimization of parameters is performed (instead of taken fixed values for parameters)
4. Selection of grid size, the lower value and upper value for which parameters are optimized (embedding window, CDF threshold and step size)
5. Selection of similarity algorithm (settings Measure, Epsilon and Neighbours are not related to the similarity sampling algorithm)
6. Start analysis and go to next interface
1. Predecessors used with the corresponding weights of vector $A$ within the distance metric
2. Interactive graph containing events and predicted warnings together with actual critical events (possibility to zoom in and select individual timestamps)
3. Selection of vehicles and training/monitoring results
4. Selection of critical event
5. Performance accuracy measures of the most optimal settings found