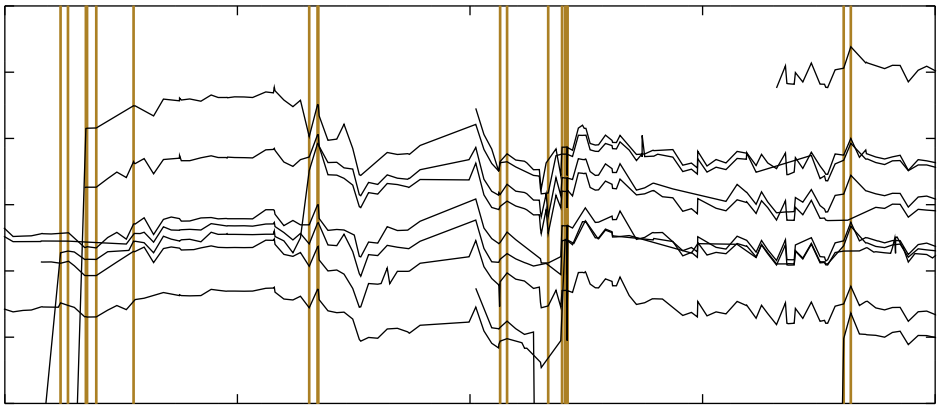


Advanced Algorithms to Identify Performance Degradation

Master's thesis in Computer Science: Algorithms, Language and Logic

ANNIKA JOHANSSON

MARKUS OTTERBERG

Department of Computer Science & Engineering
Machine Learning & Algorithms
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Advanced Algorithms to Identify Performance Degradation
ANNIKA JOHANSSON
MARKUS OTTERBERG

Supervisor: Prasanth Kolachina, Department of Computer Science and Engineering
Supervisor: Mattias Runge, Ericsson
Examiner: Peter Damaschke, Department of Computer Science and Engineering

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Figure of performance degrading jobs noted by the algorithm developed

Typeset in LaTeX
Gothenburg, Sweden 2016

Abstract

For the purpose of analysing performance of a system, data measuring the resources consumed are gathered. The common goal, independent of what is measured, is to draw conclusions on the performance and see if an update has improved or degraded it. Performance analysis in computer science becomes increasingly important as software controls more and more complex processes and requires more and more accuracy in both precision and timing.

As new data are rapidly generated, automation of the analysis both saves money and achieves a more reliable result compared to manual inspection. Machine learning is common for automated data analysis. In this thesis methods from the machine learning field are applied to performance data with the aim of identifying performance degradation. Both the data and aggregated points are analysed with k-means and k-medoids clustering algorithms and the results show points leading to degraded performance.

The performance measurements analysed are the load and memory usage of the hardware, generated during testing of the actual hardware and software in a simulated environment. It is generated from a number of different tests running different scenarios, which gives the data a large internal spread in covariance. Due to this large spread a threshold method is not exact enough to determine performance of a single update.

In order to analyse changes in the data, aggregated adaptations consisting of the change from one point in time to another are generated. The changes are clustered for each kind of measurement and the clustering is quantitatively and qualitatively evaluated in order to determine its success.

By using two stage hierarchical clustering, where the first layer is used to remove outliers, most of the points leading to performance degradation within the dataset are singled out. At each stage of the clustering different distance metrics are evaluated and the optimal k and the corresponding weights are algorithmically found for each metric. After evaluating each way of clustering the top performing ones are chosen based on quantitative and qualitative measures, such as V-measure and adjusted Rand index.

The centroids of the chosen clustering method are labelled and all points are labelled, each point according to the centroid of it's respective cluster. The points labelled as performance degrading are used to locate updates which led to degraded performance. Finally, the method designed is compared to what is generally required of a fault detection system to determine if it can be used as such.

Acknowledgements

This thesis is the result of discussing cell phone coverage during Håkan Hellström concerts, glancing at the boats in Göta älv while running slow scripts and using up more and more whiteboard space. For everything beyond that, we owe many people great thanks.

We would like to thank our supervisors, Prasanth Kolachina and Mattias Runge, for support, inspiration and great ideas and our examiner Peter Damaschke for clear and constructive feedback. Thanks to Ericsson for setting up this project, Cecilia Ekerstig, Johan Mellgren and Rickard Thomasson for shown interest and many clarifications. Thanks to Magnus Larsson and team Bees for practical help and cheers. Thanks to David Grankvist, Jonatan Kilhamn, Jennifer Panditha and our opponents, Sean Pavlov and Simon Almgren, for reading and challenging. And also, thanks to our families and friends for pep, interest and putting up with our absent-mindedness.

Annika and Markus, Göteborg June 20, 2016

Contents

1

Introduction

The work in this thesis consists of using machine learning algorithms to construct a fault detection system that clusters performance data in order to identify performance degradation. The performance data derives from testing a signal processing software belonging to the telecommunications company Ericsson. After aggregating the data and creating appropriate datasets, these datasets are clustered using k-means and k-medoids algorithms and a number of different distance measures. The clustering is then evaluated and the result utilised for identifying performance degradation in the software.

This chapter is briefly describing the method, starting from the context of the problem to a description of the problem itself. An overview of the field is performed as well as an analysis of societal aspects. Lastly, the outline of the rest of the thesis is stated.

1.1 Context

Ericsson is a large global company that provides mobile communication infrastructure (more information on Ericsson is found in appendix A.1). One of Ericsson's products is a base station for mobile communication—a combined transmitter, receiver and antenna—that communicates with other units such as cell phones. The communication is done

through sending signals over a number of channels. The coding and decoding of these signals is performed by software in the base station. That software is the targeted software of our study.

The software runs on a multiprocessor architecture. It has a limited amount of memory and cycles to perform its functions due to the real world situation. Hence, this software is subject to several real-time constraints.

The importance of reliable performance from all units in the mobile communication system is easily grasped. Ensuring good performance is one of the main challenges of the software industry today. The software is tested frequently, providing daily measurements points of total processor load, memory usage and latency for each of several processes.

These performance data are collected in real time and stored in a database. In this age of information, software is updated often and it is of utmost importance to know the effects of a certain update in order to be able to follow the development of the software. The aim of this thesis is to use the performance data to automatically identify performance degradation caused by an update.

1.2 Previous work

The use of machine learning for analysing performance degradation is a field that grows as machine learning techniques become increasingly sophisticated and data on software performance becomes more easily accessible. One common application is the management of a server cluster. The unequal demand for performance between different users of the server cluster and the ability to transfer a user between servers opens up for the possibility to analyse the performance of each server and allocate users automatically based on the result. In [1] a threshold algorithm is used to analyse the load data. However [2] outperforms a threshold algorithm by applying a CUSUM algorithm that finds which servers are overloaded. CUSUM is a stochastic model that identifies abrupt increases from a target value (which is the mean of prior measurements). [3] and [4] describe how the complexity of events on servers leads to threshold methods becoming unreliable.

Another potential application of machine learning in this area is to calculate the expected time frame of software degradation. The problem would then be to predict the time left until software ageing causes the software to crash. [5] lists a number of papers where machine learning is used in order to predict such software crashes. They state that the most used algorithms to analyse performance data are decision trees, neural networks, support vector machines and Naïve Bayes.

Most of the cited results have access to pre-labelled data, for example measurements

about server overloading. Such a data point can be labelled as an example of bad performance, and the researchers can then apply supervised learning techniques. When similar points are found, they can say "these are also bad-performance-clustered points" without the need for human inspection. In the problem at hand, no labelled data are available, so unsupervised techniques are applied. Since the data in this thesis consists of performance measurements and the goal is to identify performance degradation the setting is very similar to the situations described above. We therefore consider our research to be within the same field.

One example of the similarities in datasets is that not only the current state but also earlier data points are important in order to correctly label the data. Therefore, aggregated data points such as the derivative in each data point will be used, in similarity to [6]. In that paper, the time for software crash is identified by applying Lasso regularisation on both load data and aggregated data points such as the average slope of each parameter.

1.3 Problem description

The background to the current problem is that the software in use has a restricted amount of hardware. To avoid overusing those resources the software is monitored. Questions then arise: How should the data gathered be used? How can one say if the software is being improved or degraded from a certain update?

Ericsson currently applies a threshold method to analyse the data, i.e. only the absolute value of the usage of each resource is measured. When this value reaches a pre-set limit, the product owner is warned and the latest update, that caused the breach, is examined and reworked. This is a simple and straightforward solution but it has disadvantages. For example it might be the case that the last update did not increase the measured value very much, compared to other updates—it was just unfortunate enough to be the one to cross the threshold. Also, considering only the current absolute value might not warn the product owner about ongoing trends that might cause a breach in the future.

Ericsson is aware of the drawbacks of the threshold method and is therefore also applying manual inspection of the data. The data are presented to the project owner in graphs corresponding to each test case run on the software. This method is however both time consuming and arbitrary since no distinct method is used to identify updates that caused performance degradation.

1.4 Purpose

The goal of this thesis is to extract valuable information from the recorded performance data in order to assess whether a certain update has degraded or improved the performance of the system. In order to establish the direction of progress, the data are analysed and classified. The analysis will provide a way for Ericsson to supervise their future software updates. The goal of this project is to answer these research questions:

- What is considered to be a performance degradation of the targeted software?
- Is it possible to measure dangerous trends in the software performance, and warn the product owner them?
- Is it possible to find out which update is most likely to have caused performance degradation?
- Could this be automated so that a monitoring software could warn the product owner automatically?

1.5 Scope

The project will be kept within the scope of the software under test, and will not cover interactions with other software within the base station.The actual usage of the software and the technicalities thereof will not be of relevance, except insofar as needed to provide the background of the project. Neither will the commercial use of telecommunication be within the scope of the project.

We will not consider improving, monitoring or analysing any other part, software or data than the one directly linked to the this thesis. The work will not include any research on the hardware that the software is running on, except for its performance. Regarding the software under test, no suggestion for or implementations of improvements to the software will be made within the scope of the project. The project aims to identify updates that have affected the performance of the software but will not answer why they affected the performance or how they had this effect.

1.6 Societal aspects

The thesis work is situated within two fields with important societal aspects. One is automatic fault detection and one is wireless communication. Automatic fault detection is a growing field as more automation is incorporated in society. Determining which update started a negative trend of degraded performance is very beneficial since the developers can focus on improving that update, saving both time and money. In many industries automation of processes means loss of jobs. It is however not probable that that is the case in our application since the fault detection is only a small portion of the project owners job.

Judging whether an update has improved or degraded the software could be a sensitive thing since it essentially is judging the work of an employee. Concluding that someone has not done a good job is often not well received, especially when the data are ambiguous and the opposite conclusion might also be within reach. The human mind tends to make decisions that benefits the individual or the group to which the individual belongs [7]. This could result in not rating a certain update as degrading the performance, in order to keep a good work atmosphere. However, automatic identification aims to control this issue by presenting information that is unbiased and correct. This may lead to a better work environment and faster technical progress.

In our society, connecting to each other using our phones has become a vital part of many peoples' lives. Telecommunication is considered a part of the infrastructure and carries information of high societal importance. Imagine, for instance, an emergency call to the fire department that cannot get through because of telecommunications system overload. Developing the system to handle higher load and more users is therefore of great importance to the society. It is also part of this thesis.

1.7 Thesis outline

Following this introductory chapter, a background on the telecommunications domain is provided in chapter 2. This aims to put the problem statement and the data in their correct context, and to this end briefly explain the telecommunications system as well as the software whose performance data are analysed. The testing that gives rise to the data and the meaning of the measurements are explained.In chapter 3 the non-domain specific theory used in the thesis is stated, including an overview of the research on fault detection systems as well as machine learning. This chapter aims to give a background that motivates the methods applied in the work.

The entire process, going from data to results, is described in chapter 4. The process consists of obtaining aggregated points from the raw data, placing them in different datasets and performing a hierarchical clustering in two steps. The first step aims to remove outliers in the data and the second step identifies performance degradation. The clusterings are evaluated and the best clusterings are used to identify updates that have caused performance degradation.

The results from performing these methods are presented in chapter 5 as both tables of evaluations and plots of clusterings. Resulting plots and tables are also presented in appendices C and D. Although chapter 5 contains some discussion of the results, further discussion on the system as a fault detection system is provided in chapter 6. There also a number of areas for future work are presented. The thesis is concluded in chapter 7 with a short summary of the results.

2

Background

The domain of the data that are analysed is the telecommunications network of wireless communication. In this section the theory of that domain is explained in order to give a background to the data as well as to the relevance of the subsequent analysis.

The data are generated from the running of a software called Baseband Layer 1 (BBL1), located inside a mobile tower. To provide an understanding of the task of this software the mobile network with components and usage is described here. The specific technology of the software is explained in greater depth in appendix A, so that the data and domain can be thoroughly understood.

2.1 Cellular networks

Most of us understand a cellular network from a users point of view. When someone uses their cell phone, it connects to a cell tower and they can talk, send text messages and check Facebook. How this is achieved technically is largely unknown.

A cell phone or other device connected to the network is called a user equipment (UE). The UE will always connect to its nearest cell tower, or more precisely to the base station

situated in that cell tower. Base station or radio base station (RBS) is a collection name for all different computers that handle connections from UE into the network [8, ch. 9.3]. The connection from UE to RBS is called up-link (UL) and the reverse is called down-link (DL).

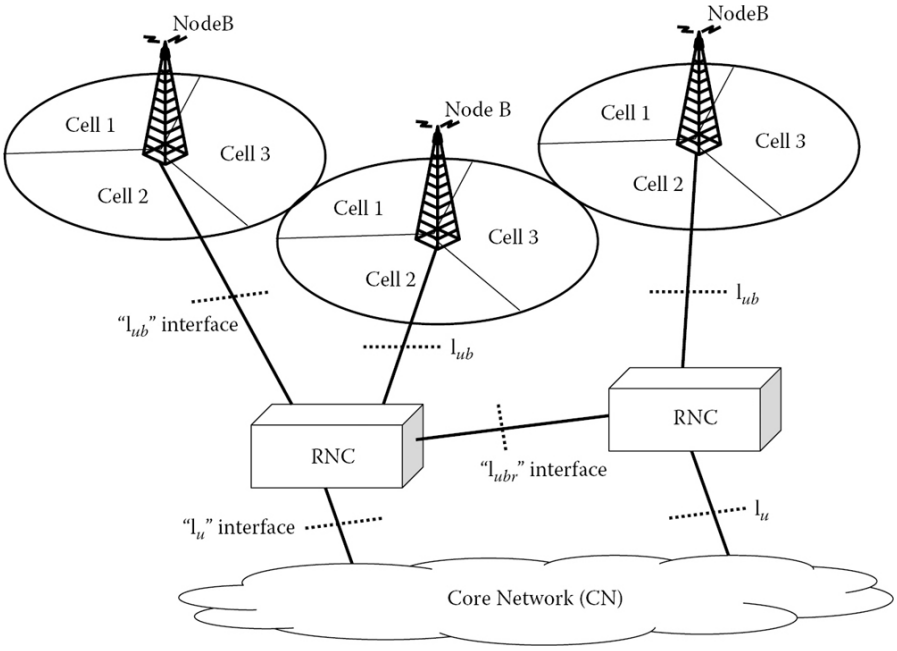


Figure 2.1: Illustration of a cellular network. Figure taken from [9, ch. 1].

In figure 2.1 the 3G network is depicted, in which each RBS is called a NodeB. The connection to a NodeB is made through signalling one of the antennas on the cell tower. Every NodeB has at least 3 antennae and each of their coverage areas is called a cell [9]. Depending on which cell the UE is located in, it will reach the NodeB through that antenna. To reach further out on the network the UE's data are transferred with the so-called I_{ub} protocol to a radio network controller, (RNC). The RNC has one or more NodeBs and is also connected to other RNCs. Together they build up the cellular network. The tasks of the RNC include keeping track of which UE are in its connected cells, routing information and sending data from other RNCs to the NodeBs in its system. The cellular network is in turn connected to the *core network*, which is a common term for all land-line phones and the Internet.

The wireless connection from UE to NodeB consists of signals travelling through the air. The frequency of the signals lie within the radio region of the electromagnetic spectrum. This radio spectrum lies between 3 Hz and 3 THz and is used for all radio communication today, from submarines to radio astronomy [10, ch. 1]. A *band* is an interval in the spectrum; one such band is the ultra high frequency band, from 300 to 3000 MHz, used for mobile phone communication [11]. The physical boundaries of this

band limits how many communications can be established in the same place at the same time. The reasons for this, and solutions to it, are described in appendix A.2.

2.2 Base station

As seen in figure 2.1, the base station a key component of the communications systems of today. An RBS is used to connect mobile phones to the wider telephone network. The RBS consists of one or more antennas and a base signal subsystem [12, ch. 2]. The base signal subsystem in its turn consists of a number of different hardware-modules (computers) that work together to handle everything from cell phone signal input to outputting signals and forwarding connections to the RNC. The implementation of the base signal subsystem conforms to the layer 1 protocol described in appendix A.3.1 and the tasks include channel handling, appendix A.4.1, signal processing, appendix A.4.2, and to implement multiple access, appendix A.2.

2.3 BBL1

BBL1 is a multi-threaded software that runs within the RBS (NodeB). Its main responsibility is to:

- Decode signals from UE to RNC
- Encode signals from RNC to UE
- "Listen" to the common radio base frequency

The *common radio base frequency* is the channel where all UE send a signal if they want to connect to the network. When a signal is sent there the RBS forwards it to the RNC. The RNC then signals the RBS back to set up a connection to that UE. Depending on what the UE wants to transfer—real-time speech or data—different kinds of channels are used. There are dedicated and common channels. Common channels are also called shared channels and are used by a number of UE at the same time.

Data is usually transferred on common channels since the timing is of less relevance than for speech. Speech requires a dedicated channel to avoid disturbance. The details on the implementation of the channels is presented in appendix A.4.1.

The connection between a UE and an RBS consists of a unique code that the UE sends and receives its signals with. The signals are encoded in order to separate them from

the other UE talking to the RBS on the same frequency within the same cell. The signals are encoded by the UE before they are sent, and then decoded by the BBL1, see appendix A.2.4. Both the encoding and decoding is done on hardware specialised in signal processing, called DSP, see section 2.3.1.

The information from the RNC that is to be sent to the UE has to be processed before being encoded. The process aims to build up the signal that the information is transferred in. This signal has to have the right frequency, wavelength and so on to be able to travel directly to the UE through the air (see signal processing, appendix A.4.2). After being packaged into a signal, the data are encoded with the right UE code and sent to the UE. The sending is done by the antenna, which is connected to the BBL1 through via Ethernet.

2.3.1 Hardware layout

BBL1 is run on a multi-core hardware platform located within the RBS. For signal processing purposes BBL1 uses a number of DSP units.

A DSP is a special kind of microprocessor. A microprocessor is a collection of registers and logic elements on a microprocessor chip, with an internal data bus where data are transferred between the registers [13]. Microprocessors can be programmed and have data memory connected to them through their external buses. A bus is a connection that can be used to connect something to the processor.

A DSP is designed for fast signal processing. The procedure of converting, processing and then reconstituting data it requires several electronic stages. Nowadays the calculations could be performed by an ordinary multiprocessor, but the performance would be worse. This is caused by a number of things, for example the implementation of multiplication not being optimised, and the code structure necessitating inefficient memory access. DSPs can be designed in many different ways, but share some basic architecture. This includes a 24-bit data word processor split into ten functional blocks. It also has one external bus for program and two for memory. Overall this architecture is perfect for small and complex signal processing [14].

2.4 Testing of BBL1

BBL1 is continuously tested as part of continuous integration, see section 3.1. Tests written by developers implement regression testing on both functionality and capacity. During development, the developers can run the tests they find suitable to ensure that

the feature they are building is functional. Some of the tests are also placed in one or more of the three types of test-suites that are run regularly:

Commit-gate Every commit goes through a test-suite before being incorporated in the master branch.

Hourly Every hour a test-suite is run with more tests than the commit-gate suite. Does not run while the daily suite is running.

Daily Every night a large suite is run, which includes the hourly tests as well as many longer tests.

The tests are grouped into jobs, where the daily jobs comprise the largest collection of tests. In practice, each job is run on a simulated node situated within Ericsson's facilities. The testing consists of placing the new version of BBL1 on the node, and then run the tests. Each test simulates traffic from UE to the node and back. The response of the node is matched against the expected response. During some tests the load on the hardware is measured, giving rise to the data analysed in this thesis.

There are a couple of inconsistencies in the testing environment worth mentioning, since they might affect the data gathered. The operating system of the node is upgraded at uneven intervals, sometimes affecting the efficiency of the BBL1. The test cases can be changed by the developers even though they have already been run for a while, potentially changing the measurements in comparison to earlier data points.

2.4.1 Performance data

During testing, different performance data are gathered. The data are generated by taking measurements on both hardware use and system response time. The data can be separated into two categories: load data and latency data.

Load data

The load data consist of two different measurements: DSP load and memory load. Both of them are measured in percent of total capacity. Memory load is the largest measured amount of memory used by the software throughout the run of a test case. DSP load is the average load on all DSPs in the hardware. The load of each DSP is the average processing power used by that DSP throughout the run of a single test case. Due to the multi-threaded nature of the software the total DSP load will vary with the efficiency of the corresponding code update.

Latency data

The tests described above are, as mentioned, simulations of connections with UE. There are a number of demands on these connections in order to achieve the connection speed needed to supply the UE with the data it requests. The connection to and from each UE is implemented by the channels described in appendix A.4.1. In order to check the demands, the time needed to complete the calculations used for the connections is measured. The time is measured in chips, see appendix A.2.4. The measurements are taken either by adding Logic Analyser Tool (LAT) events in the code that captures the calculations, or by taking timestamps with the hardware that runs the tests, the DPAD card. The measurements are then saved as the latency data for each respective channel. The latency measurements taken with DPAD may be affected by upgrades on the software that is run on DPAD. It is not known how large the fluctuations in the data possibly caused by this might be. The latency measurements taken with LAT events are known to be stable.

There are 12 different latency measurements; their name, measurement-type and meanings are listed in appendix A.4.3. Each latency measurement is taken from a number of different test cases; however, not all test cases have all types of latency data.

3

Theory

In this chapter, the theory needed in order to understand the research of this thesis is presented. Expressions are defined and earlier work clarified. This includes an outline of the field of fault detection as well as of the field of machine learning, both with emphasis on the methods used in this report.

3.1 Continuous integration

BBL1 is developed using continuous integration, which is part of the development practice of extreme programming [15], but which is used here as a self-standing practice. It is a software development practice that aims to keep updates small and regular and all code is kept in a version handling repository—at Ericsson, Git is used. Furthermore, every time a new version of the code is uploaded to the repository, a test suite is automatically run.

The results of the tests shows if the new version is as stable as the earlier version. A successful version is called a "green build", and "keeping it green" is a common goal among the developers. The goal is that new versions should be added every day, so that the changes become small and well integrated. This gives great possibility of fault tracking and bug fixes, due to both the modest size of code additions, and the fast feedback to the

developers. The developers will remember what they changed and can faster see what might have caused the red light. By updating the main project often large branches are also avoided, which saves time from resolving conflicts.

3.2 Fault detection system

This thesis aims to construct a fault detection system for software based on output from its testing. In order to build a fault detection system of good quality with easily accessible documentation it is of high importance to study the latest work within fault detection. Here a number of important definitions, including the definition for a good fault detection system are presented.

Fault detection is taken to mean identifying faults in a process or system. Fault is another word for problem. A problem does not necessarily imply complete failure, but rather non-optimal performance [16]. A symptom is a variable value needed to detect faults. If the symptom is obtained by active testing it is called a test result.

Fault detection is the process of recognising that a problem has occurred. This is different from, but often confused with, fault diagnostics, which is identifying the root causes of the problem. Fault diagnostics is not within the topic of this thesis. Automated fault detection depends on input from sensors or derived measures of performance. Sensor malfunction is a common problem so distinguishing between that and actual faults is a major issue in the field.

A fault detections system should as the name implies identify faults. While this might sound simple, there are a number of demands such a system should fulfil, they are listed in figure 3.1 [17]. A system based upon our work will aim to fulfil these demands, with the exception that no diagnosis of the system will be performed. A fault detection system aims to transform data from unclassified information to a classification of the state, into faulty or normal behaviour. In general, this is performed in four steps, as shown in figure 3.2.

The measurement space consists of measurements x_1,$x_2, \ldots$,x_n with no a priori knowledge of the problem that can relate these measurements. This corresponds to the raw data presented in section 4.1. The feature space is built of points $y = (y_1,..,y_i)$ where every y_i is a feature obtained as a function of measurements, using a priori problem knowledge. This knowledge is stated in chapter 2 and leads to the aggregated points listed in section 4.3 which builds up the feature space. In order to create the decision space, the features are combined to extract useful information about the process behaviour. Here the aggregated points are combined to create different datasets for clustering, listed in section 4.4. The clustering gives rise to the final step of the fault

1. Early detection
2. Isolability - discriminate between different failures
3. Robustness to noise and uncertainties
4. Novelty identifiability - new or known malfunction
5. Multiple fault identifiability
6. Explanation facility - justify results
7. Adaptability
8. Reasonable storage and computational requirements

Figure 3.1: List of demands on a fault diagnosis system.

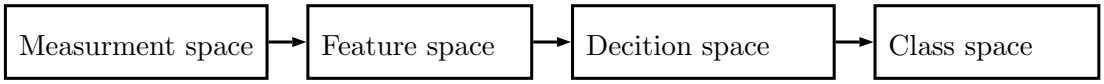


Figure 3.2: Illustration of the transformations in a fault detection system (figure inspiration taken from [18, ch. 3]).

detection system, the class space. The class space consists of a categorical indication of which failure class, including normal behaviour, a given measurement belongs to.

3.2.1 Constructing a fault detection system

There are a number of ways to construct a system that has all the characteristics listed in figure 3.1. The three general categories of such systems are quantitative model-based methods, qualitative model-based methods and process history based methods. A quantitative model-based method creates a model of the process built up on a priori known mathematical relationships between the inputs and outputs of the system. In a qualitative model-based method, these mathematical relationships are known for different units in the process.

Due to the nature of the problem at hand, a process history based method will be appropriate. In fact, all the data available in this problem are history data and no knowledge of mathematical relations is available. Therefore the most common history based methods for constructing a fault detection system are described here as an explanation for the choice of method made.

Process history based methods

The main distinction of history based methods are qualitative and quantitative systems [19]. Within qualitative methods expert systems and qualitative trend analysis (QTA) are found. An expert system is a very specialised system that solves problems in a narrow domain of expertise. It is basically an *if-then-else* system built on a knowledge base. In the problem at hand not enough detailed knowledge about the causes of the data is available to construct such a system. QTA is an abstraction of trend information and the usual application is to filter the data to extract information. As the goal of our method is to identify peaks in the data, filtering is inappropriate. However, some inspiration from QTA is taken, as presented in the section on trend-analysis, 3.3.

Quantitative methods aim to perform pattern recognition on the data in order to classify it. Within quantitative methods, distinction is made between statistical methods, neural networks and self-organising maps. Statistical methods use knowledge of a priori class distribution to perform classification. Since no such knowledge is available to us, statistical methods are not usable. However, the discussion on Gaussian Mixture Models, section 3.4.2, might be a possible way of incorporating statistical hypotheses on the data.

Excluding statistical methods, there are neural networks and self-organising maps left. Neural network methods can be performed both supervised and unsupervised. As no classified data is available only unsupervised methods are considered. They are called self-organising neural networks and the structure is adaptively determined based on the input to the network. In general, the usage of neural networks in fault detection systems is a widespread and an interesting research direction. However it seems that the greatest advantage of neural networks is their capability for fault diagnostics, which is not within the scope of this thesis. We therefore continue to explore a more simple approach that aims only to label the data as faulty or not: the self-organising map.

A self-organising map is described as grouping together the data according to some similarity measure. The similarity is usually referred to as a distance metric, determining the distance between two data points. The self-organising map should be applicable on the input data available, as the hypothesis is that similar patterns are found in all the updates that lead to degraded performance.

There are a number of algorithms for solving the self-organising map problem. The most common ones are the clustering algorithms and according to an overview performed in [19] the most popular clustering algorithm is the k-means algorithm. We explore it further in section 3.4.1. When choosing clustering algorithm it is also important to study the works of others who performed fault detection using clustering. Two of those cases are described below.

Data-driven fault diagnostics for an automobile suspension system by using a clustering based method. In [20] Wang and Yin present a fault diagnostics system that builds on a clustering method. This method is purely data driven and builds on a limited a priori knowledge of fault features. The problem setting is that of identifying faults in the suspension system of a car by analysing sensor data from the corners of the car. These data are clustered using possibilistic c-means clustering (PCM). PCM is a fuzzy clustering, which means that it assigns each data point probabilities corresponding to several different classes, in contrast to regular clustering which will either assign or not assign each point to a given cluster.

It is known that different problems in the suspension system generates different patterns in the four-dimensional data. The proposed method therefore aims to group these fault patterns in different clusters and to be able not only to identify faults but also distinguish between different categories of faults. The algorithm starts by clustering known normal data in one cluster. When a new data point is added it is determined if it belongs to the normal cluster or forms a new cluster. A new cluster indicates a fault, and future data that are also placed in this cluster will also be classified as faults. This creates a fault line in the data that is more adaptable than a threshold.

In our method the final resulting clustering is aimed to be used as the clustering in Wang and Yin's method. A new measurement that is placed in an cluster earlier determined to represent faults will also be classified as faulty. Wang and Yin's successful experiments point towards interesting results of this approach.

Layered clustering multi-fault diagnosis for hydraulic piston pump. In [21], Du, Wang and Zhang present a fault diagnostics system that uses layered clustering. This is interesting, since our method applies a two-layered clustering algorithm to obtain a fault detection system. The problem setting is that of a hydraulic piston pump in an airplane, subjected to high level of stress. Over its lifetime it is likely that faults arise. The pump is monitored in real-time by a prognostic and health management system. The system has earlier been able to identify faults but not to diagnose the type of the fault.

Due to the nature of the airplane usage it is common that multiple faults occur simultaneously. This is unfortunate in a system that can distinguish only one problem at a time. Du, Wang and Zhang introduce layered data analysis using clustering. Each layer can distinguish between a number of combined problems, and by applying the next layer the algorithm can distinguish within those faults, and so on up the layers. This use of hierarchical clustering is interesting also in the problem at hand, since some data are known to be faulty measurements and some represents the actual faults to be identified. Section 4.6.1 describes how this method is implemented.

3.3 Trend analysis

In our method, part of the fault detection system builds upon trend analysis. Trend analysis is a broad and active field with many applications, the most common being stock market analysis. There, trend analysis is used to predicting the opportune moment to buy or sell a certain stock. Trend analysis is also present in the field of chemical engineering, where it is used to identify states of a chemical system.

One common approach to using trend analysis in a stock-market setting is the sliding window, used for example in [22]. In the sliding window approach, the data are organised into several ordered subsets of a pre-specified fixed length, where each subset is shifted one time step compared to the previous one; i.e. if the first interval ranges from time index 1 to 10, the next window will range from 2 to 11. Each of these windows is then split into two parts. The first part becomes the characteristics of the trend and the second, smaller part becomes the outcome. All characteristics values are labelled as either "UP" or "DOWN" depending on the direction of the slope of the outcome. This process creates the training data. When new values are recorded, a new sliding window is created of the same length as the ones in the training data and clustered into one of the two clusters "UP" or "DOWN", where the labels now represent the predicted trend for this stock. Our method includes an trend analysis inspired by the sliding window.

Closer to the fault detection application is the use of trend analysis in chemical engineering. Here, data on processes are analysed in order to identify the state of the chemical system. In [23], Cheung and Straphanopoulos identify 4 routine tasks carried out to analyse the state of the chemical process:

1. Distinguish normal from abnormal conditions
2. Anticipate future operational states
3. Identify causes for process trends
4. Plan and schedule sequences of operating steps

Faulty assessment of data can lead to erroneous decisions in these four areas. Cheung and Straphanopoulos therefore propose a formal representation of process trends called triangulation, to help process operators' correctly read data. However, triangulation leads to a form of smoothing of the data. Since the application of this thesis aims to identify peaks in the data, triangulation cannot be considered helpful here. It is nevertheless interesting that the aim of [23] is the same as ours, even though trend analysis is a completely different field than fault detection. In fact using trend analysis in fault detection is a widely known and used concept in fault diagnostic, as can be seen in the overview of the field in [24]. It is clear that our approach of using trend

analysis is one of many attempts within the field to incorporate historical knowledge in the decision-making process.

3.4 Machine learning

The field of Machine learning was coined in 1980 with the workshop and book *Machine Learning: An Artificial Intelligence Approach*, edited by Ryszard Michalski, Jaime Carbonell and Tom Mitchell [25]. However, the field was foreseen much earlier, for example by Alan Turing in 1950:

> Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's?
> If this were then subjected to an appropriate course of education one would obtain the adult brain. [26]

To produce a program that simulates an adult brain would be to produce a program that solves a problem in a known way, i.e. to implement a solution in code. To simulate a child's brain however, and teach it, is to implement an algorithm that learns, and then give it something to learn. In Turing's time self learning algorithms where just ideas, but now they exist. The algorithms used in our method for learning from the data are presented below.

There are two general classes of machine learning: supervised and unsupervised learning. In supervised learning, the dataset contains not only data but also meta-data. This meta-data is usually some kind of labelling of the data, for example the data "red" could be labelled "COLOUR" and "car" could be labelled "VEHICLE". The goal is to create an algorithm that learns from these labelled data, in order to then classify new, unlabelled data: for example determining whether "GREEN" is a colour or a vehicle [27].

In unsupervised learning, no prior meta-data is provided. Instead, the algorithm identifies significant patterns or features in the data. One common method within unsupervised learning is clustering, where points close to each other are placed in the same cluster [28].

3.4.1 k-means and k-medoids

k-means, or Lloyd's algorithm [29], is an algorithm used for clustering data. The k in the name of the algorithm comes from the fact that k-means clusters its data into k

clusters, where k is a pre-specified positive integer [30, ch. 2]. The algorithm works by assigning each data point to a cluster based on which of the k centre points, or *centroids*, it is closest to. Starting from some initial centroid distribution, each centroid position is recalculated every time a new data point is added to its cluster. This adjustment aims to minimise the sum of all distances within that cluster, i.e. the new centroid is given by the new cluster mean. Then, the data points are reassigned based on the closest amongst the new centroids and the whole process is iterated until none of the reassignments assign a data point to a new cluster.

k-medoids is very similar to k-means. However, instead of calculating centroids that are separate from the rest of the data, k-medoids uses actual data points as centre points, called medoids [31, ch. 14]. When computing the new centre points of a cluster, k-medoids tries all other points in that cluster as a potential medoid. A point is chosen as the new medoid if the change would reduce the sum of within-cluster distances. The process is iterated until the optimal medoid has been found.

Assessing which centre point is the closest is not as trivial as one could believe at first. There is a myriad of different ways to calculate the distance between two points.

3.4.2 Gaussian mixture models

The k-means algorithm assumes clusters that are well separated. If the clusters overlap, k-means will have problem distinguishing them from each other. A Gaussian mixture model is based on the assumption that the data can be modelled as a number of normal distributions mixed together [32]. They can therefore be separated by letting an algorithm similar to k-means, called the EM algorithm (expectationmaximisation algorithm), identify not only a mean but also a variance for each cluster (distribution). If the identification is correct, i.e. the clustering is correct, it is possible to not only identify distributions in the data, but also label each data point with a probability distribution over all the clusters. This solves the overlapping cluster problem by allowing an area where the data are more or less equally likely to belong to either of the overlapping clusters.

In the problem at hand it is probable that the data indicating faults and no faults are overlapping due to the large amount of data in the same interval. Therefore a Gaussian mixture model might be a more correct model of the data. It is however not priorly known if the clusters of faults have some kind of normal distribution, so that the Gaussian mixture model would gain much better result than k-means. Due to this uncertain outcome, Gaussian mixture models are not experimentally explored in this thesis but are relevant for future work.

3.4.3 Distance measures

The distance measure used in k-means and k-medoids will affect which points are assigned to which cluster. In order to find the best clustering, the distance measure used has to correspond well to the problem at hand. This thesis uses three different distance measures: the squared Euclidean distance, the city block distance and the Mahalanobis distance. In the squared Euclidean distance and the city block distance, all features of the data contribute equally to the distance measure. This means that a point (x,y) with $x >> y$ will be equally far from origo as a point (y,x). This is useful if the data are spread equally in all directions. However, if the data use different scales for different dimensions, so that for example one dimension is always much larger than another, the distance measure will mostly capture the distance in the larger dimension. This can lead to information loss and with it poor clustering. In order to compensate for this, weighted distances can be used, giving the option of scaling up the influence of one or more dimensions on the distance measure.

Squared Euclidean distance

The Euclidean distance is the most commonly used distance. In two or three dimensions, it refers to the everyday notion of distance: the length of the shortest path between two points [33]. In the general case of n dimensions, the Euclidean distance between two vectors x and y are calculated as:

$$\|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.1}$$

Since the square root is computationally complex it is useful to avoid it when clustering. To solve this, the squared Euclidean distance, i.e. removing the square root, is commonly used instead, it is defined as:

$$\|x - y\| = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{3.2}$$

The squared Euclidean distance is not a metric as it does not satisfy the triangle inequality, however, in k-means the calculated distances only have to be compared to each other and therefore the squared Euclidean distance is usable anyway. The weighted squared Euclidean distance is:

$$\|x - y\| = \sum_{i=1}^{n} w_i (x_i - y_i)^2 \tag{3.3}$$

City block distance

City block distance is also called Manhattan distance or taxicab distance. The names refer to the distance a car would drive in a city like Manhattan, i.e. going in straight lines only intersecting at right angels. This is the L_1 or 1-norm distance between two points [25].

The distance between two vectors x and y is calculated as:

$$\|x - y\| = \sum_{i=1}^{n} |x_i - y_i| \tag{3.4}$$

The weighted city block distance is defined as:

$$\|x - y\| = \sum_{i=1}^{n} w_i |x_i - y_i| \tag{3.5}$$

Mahalanobis distance

Both the above distances assume that there is no covariance within the dataset. If there is, the data will in two dimensions be placed in an oval shape instead of a circular one. Then a point outside the oval would be just as much a outlier as a point outside a circle. To compensate for this, the Mahalanobis distance transforms the data according to its covariance (expressed as a covariance matrix C), and then calculates the squared Euclidean distance in the resulting data set [34]. In order to do this, the points to find the distance between must be in the dataset, which means that the Mahalanobis distance can be used for the k-medoids method, but not for k-means.

The distance between two vectors x and y is calculated as:

$$\|x - y\| = \sqrt{(x - y)C^{-1}(x - y)'} \tag{3.6}$$

As can be seen in equation 3.6, the Mahalanobis distance is already weighted by the covariance matrix of the data. It is possible to further weight the Mahalanobis distance, for example as in [35], but this is not used in our method.

3.4.4 Evaluation of clustering

When performing clustering with an algorithm, it is important to be able to evaluate how good the result is depending on what is deemed to be important, i.e. what the clustering should show. One way to decide what the clustering should show is to make a model clustering in advance. It will then be compared to the resulting clusters from any clustering algorithm. This model clustering could be performed easily in the case of clustering labelled data, by simply choosing each label as a cluster. If such data are not present, a *gold assignment* [36] can be used. A gold assignment is an a priori labelling of the data according to domain knowledge. The evaluation should be able to provide information on how good the new clustering is compared to the model one without manual inspection of the clustering result. For this there exists a number of different evaluation and scoring techniques.

Partitions and contingency table

When comparing two different clusterings, the goal is to identify if the elements are placed in the same cluster in the different clusterings. In other word, if the partitioning of data is the same. The following evaluation techniques need two partitions of the data. The model clustering will be referenced as the *class* partition, while the assignment resulting from the algorithm will be referenced as the *clustering* partition.

There is no way of knowing in advance in what order the clustering algorithm will cluster points. Therefore, for example, cluster 1 and cluster 3 do not necessarily correspond to class 1 and class 3 and therefore a statement such as "If all points belonging to class 1 ends up in cluster 1, this algorithm is good" would be insufficient. In order to compare clusterings amongst each other and evaluate their performance anyway, a structure must be created to cover this loss of information. We construct a contingency table in table 3.1 where each row corresponds to a class, $C = \{u_i | i = 1,...,c\}$, and each column to a cluster, $K = \{v_j | j = 1,...,k\}$.

Class\Cluster	v_1	v_2	...	v_k	Sums
u_1	n_{11}	n_{12}	...	n_{1k}	$n_{1\cdot}$
u_2	n_{21}	n_{22}	...	n_{2k}	$n_{2\cdot}$
...	...	...	...	...	...
u_c	n_{c1}	n_{c2}	...	n_{ck}	$n_{c\cdot}$
Sums	$n_{\cdot 1}$	$n_{\cdot 2}$	...	$n_{\cdot k}$	$n = n_{\cdot\cdot}$

Table 3.1: Contingency table, where n_{ij} is the number of data points that are members of both class i and cluster j.

Element n_{ij}, is the number of data points belonging to class i that got assigned to cluster j by the algorithm. There are a number of different evaluation techniques to apply to this contingency table. Two of them are the V-measure and adjusted Rand index, which will be covered in the next sections.

V-measure

One method for evaluating the performance of a clustering algorithm is called V-measure [37], proposed by Andrew Rosenberg and Julia Hirschberg. V-measure is based on two criteria; homogeneity and completeness. V-measure combines these criteria into a single score that can be weighted in order to favour either homogeneity or completeness, or keep them evenly balanced.

Homogeneity is the measure most commonly used in different techniques for evaluating the performance of a clustering algorithm. It measures how accurately the clustering algorithm grouped data points from only one class to a given cluster. The downside of only measuring homogeneity is that the score can be inflated by simply increasing the number of clusters to generate. This is possible because the trivial case for achieving full homogeneity is when a clustering algorithm assigns each data point to its own cluster. That way only data points from a single class is present in any cluster. The following formula is used for calculating the homogeneity h in V-measure:

$$h = \begin{cases} 1 & \text{if } H(C) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases} \tag{3.7}$$

where

$$H(C|K) = -\sum_{j=1}^{|K|}\sum_{i=1}^{|C|}\frac{n_{ij}}{n}\log\left(\frac{n_{ij}}{n_{\cdot j}}\right)$$

$$H(C) = -\sum_{i=1}^{|C|}\frac{n_{i\cdot}}{n}\log\left(\frac{n_{i\cdot}}{n}\right)$$

Completeness, on the other hand, is less commonly incorporated into clustering measures. It measures how accurately an algorithm grouped all data points belonging to a single class into a single cluster. The trivial case for achieving full completeness is when an algorithm clusters all data points into a single cluster. That way all data points from any one class will belong to the same cluster. The following formula is used for calculating completeness c in V-measure:

$$c = \begin{cases} 1 & \text{if } H(K) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases} \tag{3.8}$$

where

$$H(K|C) = -\sum_{j=1}^{|K|}\sum_{i=1}^{|C|}\frac{n_{ij}}{n}\log\left(\frac{n_{ij}}{n_{i.}}\right)$$

$$H(K) = -\sum_{j=1}^{|K|}\frac{n_{.j}}{n}\log\left(\frac{n_{.j}}{n}\right)$$

As can be seen from the trivial examples of each criterion, they counteract each other to some extent. Thus combining both of them into a single value, called a V-measure, creates a balanced score that can be used to measure the overall performance of the clustering algorithm:

$$V_\beta = \frac{(1+\beta) * h * c}{(\beta * h) + c} \tag{3.9}$$

where $\beta = 1$ will put equal emphasis on homogeneity and completeness, while a $\beta > 1$ will favour completeness and $\beta < 1$ will favour homogeneity.

In [38], a comparison of external evaluation measures is conducted. It shows that the V-measure is a good estimate of the quality of clustering. However, it has the drawback that as the number of clusters increase, the V-measure is likely to increase as well. This is caused by an increase in the homogeneity that does not seem to be compensated by a decrease in completeness.

Adjusted Rand index

Another method for evaluating the result of a clustering algorithm is the adjusted Rand index method. It builds upon the Rand index method suggested by Rand in 1971 [39]. Rand analyses the partition by checking where a pair of two points have been placed in the two partitions. He defines four cases:

A) the points in the pair are placed in the same class and the same cluster

B) the points in the pair are placed in different classes and different clusters

C) the points in the pair are placed in different classes and in the same cluster

D) the points in the pair are placed in the same class and in different clusters

Then cases A and B will correspond to agreements between the two clusterings and C and D will give disagreements between the two. The total number of points in A, B, C and D will be $\binom{n}{2}$.

Let the lower-case letters a, b, c and d be the number of data points corresponding to Rand's cases A, B, C and D, respectively. These can easily be calculated from the contingency table 3.1 as follows:

$$a = \frac{1}{2}\sum_{i=1}^{|C|}\sum_{j=1}^{|K|} n_{ij}(n_{ij}-1) \tag{3.10}$$

$$b = \frac{1}{2}\left(n^2 + \sum_{i=1}^{|C|}\sum_{j=1}^{|K|} n_{ij}^2 - \left(\sum_{i=1}^{|C|} {n_{i.}}^2 + \sum_{j=1}^{|K|} n_{.j}^2\right)\right) \tag{3.11}$$

$$c = \frac{1}{2}\left(\sum_{j=1}^{|K|} n_{.j}^2 - \sum_{i=1}^{|C|}\sum_{j=1}^{|K|} n_{ij}^2\right) \tag{3.12}$$

$$d = \frac{1}{2}\left(\sum_{i=1}^{|C|} {n_{i.}}^2 - \sum_{i=1}^{|C|}\sum_{j=1}^{|K|} n_{ij}^2\right) \tag{3.13}$$

The Rand index R is defined as:

$$R = \frac{a+b}{\binom{n}{2}} \tag{3.14}$$

In [40], Hubert and Arabie express as criticism against the Rand index that it does not correct its value to shield against randomness, i.e. the index does not take on a constant value when the partitions have been chosen completely randomly (under a null model). It is also difficult to compare the Rand index to other measures, since it is not normalised. As a counter-suggestion, Hubert and Arabie present the adjusted Rand index, ARI, defined as:

$$\text{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}/\binom{n}{2}}{\frac{1}{2}\left(\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right) - \sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}/\binom{n}{2}} \tag{3.15}$$

Equation 3.15 can be interpreted as:

$$\text{ARI} = \frac{\text{index} - \text{expected index}}{\text{maximum index} - \text{expected index}} \tag{3.16}$$

The adjusted Rand index is bounded by 0 and 1 inclusive, and becomes 0 when the partitions are completely random.

4

Method

This chapter contains information on the workflow and method. Initially research of the problem domain was performed in order to understand the reasons behind the problem at hand and the importance of its solution. In chapter 2 the results of the initial study is shown as an overview from the real life applications of the software that generates the data. Building on that understanding of the usage, the technical specifications of the performance of the software were analysed. Then the software development method and its testing were understood. After that the data were extracted from the database and plotted in order to visualise it and more clearly understand how the measurements from different tests differ, see section 4.1. Then the data generated by testing the software could be understood and a goal on what to find in the data was constructed, see section 4.2. After the goal was constructed, the work to identify an algorithm that would find the data points searched for could begin. A study of the field and common solutions to similar problems was conducted, see chapter 3. Based on that study, datasets where created, as presented in section 4.4. The result of applying the algorithms from chapter 3 on these datasets is presented in chapter 5, and their fulfilment of the goal, from section 4.2, is presented in chapters 6 and 7.

4.1 Raw data

The data in this thesis are collected from running the tests described in section 2.4. The measurements are collected from 19 different tests. The tests subject the BBL1 to different amounts of load from UE on the node, and also tests different functionality of the BBL1. The operating system on the node is only updated once in a while so several tests are run on the same version. However, at each point in time a different version of BBL1 is likely to run, since developers integrate their code continuously, see section 3.1. All tests scheduled to run at the same time are grouped together as one job and assigned a serial number. Since most of the job-runs are scheduled, either hourly or daily, approximately the same amount of jobs are processed between each point of measurement acquired. Hence, the job numbers can be used to build a timeline over the data.

Only the tests which were successful will generate data, thus there are jobs where not all the tests that ran have a data point. Also, not all jobs generate data of the type analysed in this thesis, resulting in many job numbers which are not associated with any data point. For this reason, th job numbers span a range of about 50,000, but there are only about 300 data points for each test.

4.1.1 Load

During the running of tests, a number of measurements are taken on the hardware of the system. The memory usage is measured and the highest usage throughout the test is saved as memory usage. The load of the DSP is also measured and the average amount during the test is saved as DSP load.

These data can be visualised in graphs where different tests are plotted as separate lines; see figure B.1 for DSP load and figure B.2 for memory usage, in appendix B. The y-axis in these graphs is the DSP load/memory usage respectively, while the x-axis can be seen as indexed time, based on the job numbers as mentioned before.

Upon visual inspection, the data seems to have a correlation between the different tests. Our hypothesis is that this can be used to indicate the change in performance of the underlying software, which is of interest. In order to prove or disprove this hypothesis, a single set of data points to represent data from all the tests is created.

One way to create such a set, called estimate, would be to simply take the mean of all tests for each point where they have measured values. However, doing this would create a very imbalanced dataset, since at some jobs, only very few test yielded any data. This could lead to the mean changing drastically between two jobs, even if nothing in the

data indicates a change.

To calculate an estimate that is representative for all the tests and does not falter over missing data points, it is designed to measure the change made for each test between two existing data points in that test and then divide that change by the number of jobs between the two points, i.e. normalising the data. Then this new data are added together and expanded in order to calculate the change our hand-crafted estimate will have between two jobs. The formula for those calculations is shown in table 4.1 and equations 4.1 to 4.4. In figure 4.1 the estimate, black coloured broader line, is added to a figure containing all the DSP load data from the different tests in different coloured lines, and respectively in figure 4.2 for the memory usage.

	In the following equations this notation is used:
v	value of DSP load or memory usage, taken from raw data
j_n	job number n
h	test number—there are 19 different tests
e	combined estimate for all tests, function from job number to value
s	starting value for the estimate
v_{h,j_n}	value of test h at job number j_n
v'_{h,j_n}	derivative of test h in j_n from last measured value
H_k	the set of all tests h for which there exists a value for v_{h,j_k}
e_{j_t}	estimate at job j_t

Table 4.1: List of variables defined for calculation of estimate and aggregated points

Since there does not exist values for all tests at all job numbers the previous job number has to be defined as something else than $k-1$. The previous job number is denoted as k_p. For every test h and every job number k, let k_p be the first previous job number that has a value for test h.

The derivatives can then be calculated as:

$$v'_{h,j_k} = \frac{v_{h,j_k} - v_{h,j_{k_p}}}{j_k - j_{k_p}} \tag{4.1}$$

$$v'_{j_n} = \sum_{H_{(k=n)}} \frac{v'_{h,j_k}}{|H_n|} \tag{4.2}$$

The starting value is set to:

$$s = \frac{\sum_{n=1}^{N} \sum_{h \in H_n} v_{h,j_n}}{\sum_{n=1}^{N} |H_n|} \tag{4.3}$$

This leads to the following definition of the estimate for all test data:

$$e_{j_t} = \begin{cases} s & \text{if } t = 0 \\ e_{j_{t-1}} + v'_{j_t}(j_t - j_{t-1}) & \text{if } t \neq 0 \end{cases} \tag{4.4}$$

Note that this leads to the value of the derivative in one time point being used only in the interval closest before the measurement time. In other words, values raging over longer time than one time interval will not affect the estimate value in the earlier time intervals. This effect could be avoided by first interpolating all tests for all jobs, so that v_{h,j_n} would exist for all j_n. However, the interpolation would lead to creation of data values that cannot be trusted, since the data values are clearly not linear over time. For this reason, using the derivative value close to the measurement point gives a more accurate estimate than an interpolation for all points.

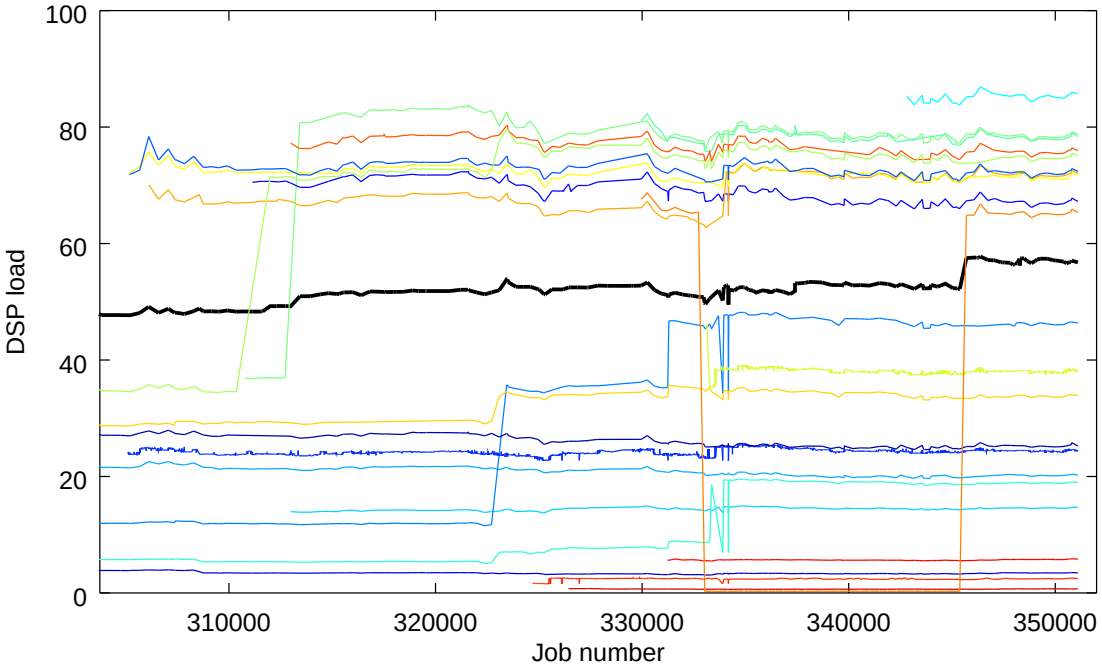


Figure 4.1: Plotted data measuring DSP load over time, with estimated mean.

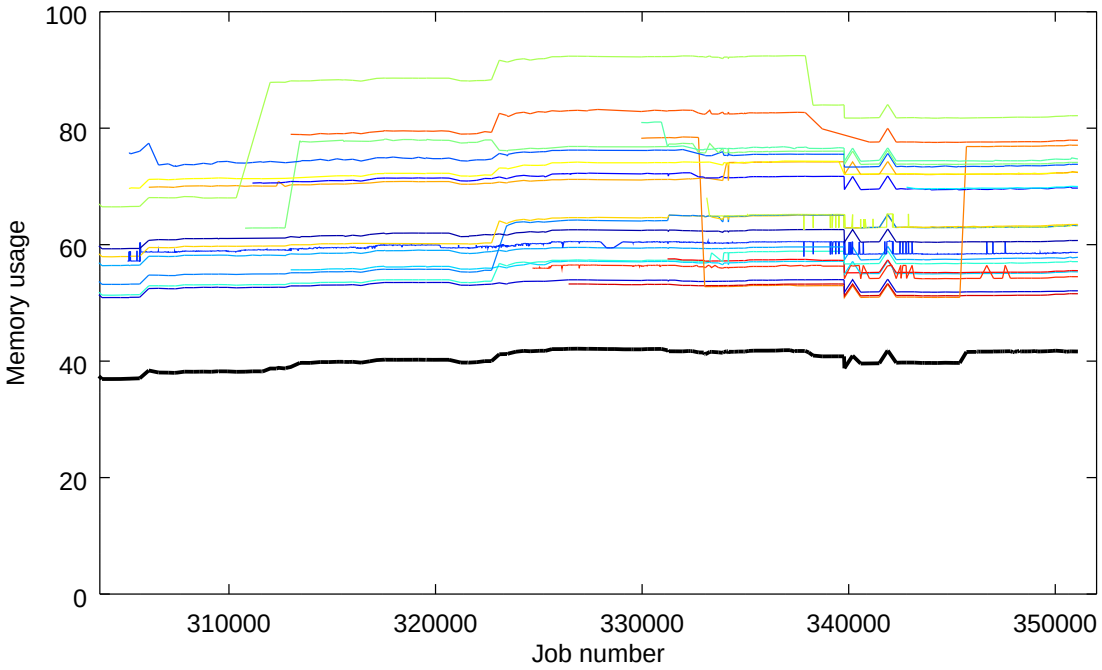


Figure 4.2: Plotted data measuring memory usage over time, with estimated mean.

The estimation is inserted as a black line in figures 4.1 and 4.2. Observing these figures, the estimate gives the impression to follow the same trends that is observed in in the raw data, even picking up the smaller changes. This indicates that the estimate is a way to represent all the tests simultaneously, providing a way to condense the amount of data to a single set representing all tests at once.

4.1.2 Latency

The latency data describes how long a task, e.g. setting up a channel, takes. The unit of time is called chip (section A.2.4). More about the measurement process can be read in section 2.4.1 and a list of the latencies are found in appendix A.4.3. Since the latency measurement consists of data from different test cases with different load, it has a heterogeneous data spread. Plots over the raw latency data can be found in figures B.6 through B.18, in appendix B.2.

For the latency, visual inspection shows common trends amongst the different test cases within each latency dataset, similar to the DSP load and memory usage cases. In order to illustrate these trends, the same method for calculating an estimate as in section 4.1.1 is used. Figure B.19 shows one example of how the estimate follows the test case measurement.

Plotting the estimates for all the different latencies into one graph yields the figure 4.3. It is obvious that the fluctuations in the data are very small due to the nature of the measurements—the time intervals are extremely short. In order to be able to analyse the data it needs to be transformed in some way. The two most common transformations for this kind of problem were tried: the Gaussian distribution transformation and the logarithmic transformation of normalised data.

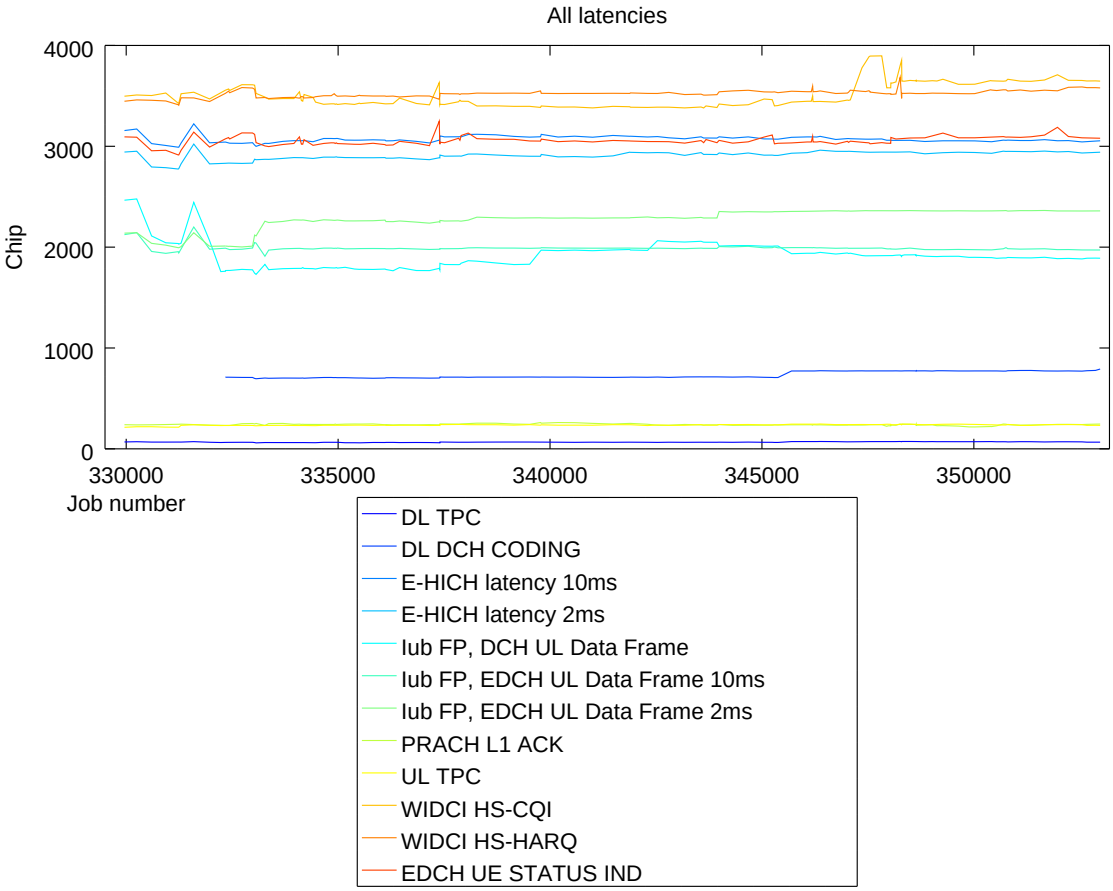


Figure 4.3: Estimate for all latencies.

In the first approach each estimate is transformed by a Gaussian distribution. This is done by first transforming the data to uniform distribution and then transform it with a Gaussian distribution with variance 1 and mean 0. The Gaussian transformation separates the data well and makes it much easier to analyse. However, the spreading makes the data fill up the entire spectrum, which makes correlation improbable. In fact there does not appear to exist any correlations between the different latencies, which unfortunately makes them uninteresting for the problem at hand. The transformed data can be viewed in the appendix, figure B.20.

In the second approach each estimate is placed just above zero by subtracting the minimum of that estimate from all elements. The resulting data are plotted using a logarithmic scale on the y-axis. The idea is to give a clearer picture of the fluctuations since the logarithmic scale emphasises small changes close to zero. The result is shown in the appendix, figure B.21. Notice that the procedure removes one data point for each estimate since the minimum will become zero and therefore undefined on the logarithmic scale. More importantly the lack of an even covariance in the data makes the normalisation very difficult—not even the normalised data are centred. This, in turn, makes the logarithmic scale unhelpful since only some of the data are scaled. Overall the picture becomes more unclear than the Gaussian transformation.

Since no correlation could be found after transforming the data, we conclude that the change in latencies will not be useful for detecting a change in performance of this software. Thus, the latency changes approach is not pursued any further.

4.2 Goal

The overall goal is to be able to evaluate the performance of the software based on memory usage and DSP load. The first goal is finding the spikes in the data and determining whether these spikes are the cause of a faulty measurement, or if the data point is valid. The second step is to classify the valid points that show signs of performance degradation in the software. The third step is to evaluate that classification in relation to the quality of clustering and Ericsson's demand.

4.3 Aggregated data

The raw data consist of values on the DSP load and memory usage for each job taken from some test. In order to analyse connections between jobs and trends in the data a number of different aggregated points are created. These are all described below, using the notation established in section 4.1.1. In the next section the aggregated points are combined into different datasets that will be clustered with machine learning algorithms.

4.3.1 Normalised value

The value of the DSP load and memory usage is given in percent, naturally ranging from 0 to 100. The aggregated points listed below are much smaller, since they stipulate different kinds of differences between values. In order to get a good spread when combining

aggregated points into two-dimensional data the raw data values are normalised, labelled v_n. This is done by dividing each value with the maximum possible value, i.e. 100 percent.

$$v_{nh,j_n} = \frac{v_{h,j_n}}{100} \tag{4.5}$$

4.3.2 Difference in value

The goal stipulates that the interesting points are the ones that have a large increase in either DSP and memory load. In order to analyse this, aggregated points showing the increase or decrease are generated. They are henceforward called d, difference value.

For each test and for each job, the difference value is defined as the value of that test at that job subtracted by the value of the same test at the previous job where it was measured. Formally:

$$d_{h,j_k} = v_{h,j_k} - v_{h,j_{k_p}} \tag{4.6}$$

4.3.3 Trend value

In order to capture something of the longer perspective in the data, trend analysis is attempted. Since long trends are hard to find, see section 3.3, a short perspective is chosen. When analysing a short perspective of trends, a number of previous points are chosen and the difference between the lowest or highest value among them is subtracted from the value of the active point. Due to the risk of arbitrariness when choosing a fixed window for this analysis we have chosen to include all earlier points until the sign of the trend is changed, i.e. until the first change in derivative. This gives a short but informative history for each point in the data, d_t.

Let $k_{p'}$ be the previous job number where:

$$v'_{h,j_{k_{p'}}} = 0 \tag{4.7}$$

$$d_{th,j_k} = v_{h,j_k} - v_{h,j_{k_{p'}}} \tag{4.8}$$

4.3.4 Relative value

For the raw measurements there is a maximum value of 100 percent that should naturally not be reached. Because of this, an increase in value is more serious if it appears when the value is high than when the value is low. Therefore two relative measures are created: d_r and $d_{t,r}$, which aims to connect the difference or trend with the value. The relative measure is created by taking the logarithm of the value for the point and multiplying it with the difference or trend in that point. This leads to a scaling of difference and trend according to the value in that point. Note that to avoid negative numbers from the logarithm, the values and not the normalised values are used.

$$d_r = d \log v \tag{4.9}$$

$$d_{t,r} = d_t \log v \tag{4.10}$$

4.4 Experiments

The aggregated data are to be analysed by machine learning using clustering. In clustering it is important that the datasets which are analysed are relevant to the problem. It is also beneficial if the dataset is easily comprehensible, since that makes the resulting clustering more easily interpretable. From these considerations, the goal and our understanding of the domain, a number of datasets, described below, were constructed to perform clustering on. Each of these datasets can be run either on one test at a time or on all the data points at once. Of course, both methods are tested.

4.4.1 Difference values in one dimension

The difference values show the increase or decrease between each sequential pair of points. It is the most concrete figure available for the impact of an update on the performance. It will therefore be relevant for the problem to perform clustering based on these points and find those that have similar difference values.

4.4.2 Difference values from DSP load and memory usage

One hypothesis of the project is that the performance degradation shows in both memory usage and DSP load values and that they therefore are correlated. In order to analyse the dependency between these two, combined points c are aggregated according to equation 4.11. Figure 4.4 shows the combined points plotted.

$$c_{h,j_n} = \left(d_{\text{dsp}_{h,j_n}}, d_{\text{memory}_{h,j_n}}\right) \tag{4.11}$$

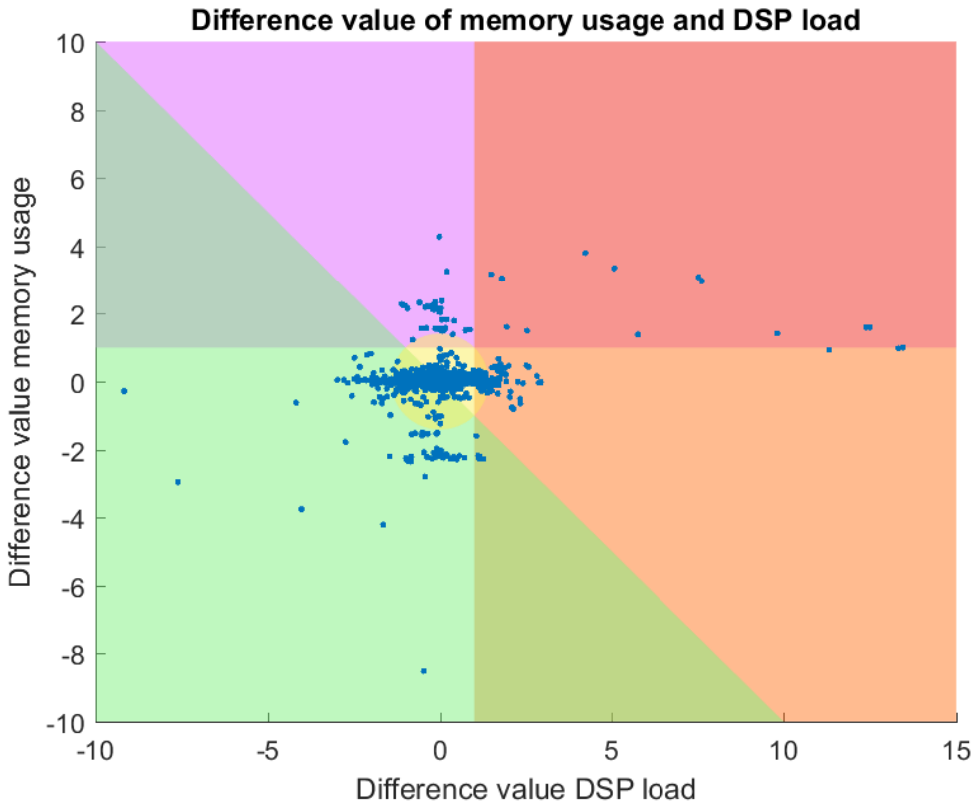


Figure 4.4: Scatter-plot for combined difference values of memory usage and DSP load.

In figure 4.4 a number of fields are shown, representing areas of interest for the difference values plotted. The yellow circle represent values that have a low change in difference value for both memory and DSP, i.e. points that show no significant performance degradation. The green area show points where performance has increased in either DSP, memory or both. These points are not of interest due to the nature of the problem. In the pink area, those points are found that show signs of performance degradation in memory. These are of high interest to the problem. Correspondingly, in the orange

area points that show signs of performance degradation in DSP are found, also of high interest to the problem. In the red area, those points are found that are degraded in both DSP and memory. A clean correlation would be identified as points on a 45° tilted axis through the first quadrant. Only a few points reside on this axis.

During discussion with the developers at Ericsson it became clear that identifying the points in the red area is not sufficient since there are points with increase in either memory usage or DSP load that are also of high relevance, i.e. the pink and orange areas. To identify these points based only on memory usage or DSP load is beneficial for the developers since the traceability increases if they know what has increased. Because of this the combination of memory usage and DSP load will not be pursued further.

The following datasets will be analysed for both memory usage and DSP load separately. This will make it easier to identify points in the pink and orange areas, and points in the red area will be identified as degrading in both DSP and memory.

4.4.3 Difference and trend values

Upgrades that have increased the value, i.e. have a high difference, and also are part of an increasing trend may in themselves be problematic or be part of a problematic trend. Therefore it is interesting to combine the difference value and trend value into a two-dimensional dataset and cluster it. The points correspond to (d,d_t).

4.4.4 Difference, trend and normalised values

The "difference and trend value" dataset contains no information of the actual value. This is problematic since negative changes are considered more serious when performed at a high value. One way to take this into account is to simply add the normalised value as a third dimension, generating points (d,d_t,v_n).

4.4.5 Relative difference and relative trend values

A three-dimensional clustering is not always optimal for combining effects of different features in the data. This is because the scale of one feature might make it stick out in the dataset, and therefore the clustering will be highly dependent on that feature. To avoid this effect a two-dimensional dataset is created, with the same aim as the three-dimensional above. Instead of adding the normalised value as a third dimension the other values are scaled with it, creating relative difference and trend values as ($d_r, d_{t,r}$).

4.5 Evaluation of clustering

An important part of clustering is to evaluate the result of the clustering and use this evaluation to compare different clusterings. In order to evaluate a clustering a goal on what the clustering should achieve is needed. In this thesis a gold assignment approach, as explained in section 3.4.4, is chosen.

In the first clustering, which aims to remove outliers, the gold assignment represents placing all difference values below -10 as negative outliers, values in the interval $[-10,10]$ as valid values and values over 10 as positive outliers. These outlier values are unreasonable within the problem domain.

The gold assignment for the second clustering is based on the information about what Ericsson wants from the clustering. The project owner has provided limits for acceptable difference and trend values. The first is a limit which no value may surpass while the other limit only concerns points that have a high actual value to begin with. The limit for a trend was put as the same as the limit for a difference. This means that a number of updates are not allowed to do what one update is not allowed to do.

The limits provided by the project owner state that the sensitive area is the range of values higher than 80% ($v > 80$). For lower values, an increase in difference value or trend value of above 3% is unacceptable ($(d|d_t) > 3$). In the sensitive area, above 80%, an increase in difference value or trend value is not allowed to go over 1% ($(d|d_t) > 1$).

These limits let us classify all data as good or bad points. Any clustering that separates the bad and good points is a good clustering, no matter how many different clusters the points are placed in. It is important to remember, though, that a huge amount of clusters would grant problems when interpreting the clustering and would therefore not be useful. Furthermore, a difference between the number of clusters in the gold assignment and the clustering will cause a difference in the V-measure that needs to be compensated for.

The V-measure is a combination of homogeneity and completeness, see section 3.4.4. When the number of clusters in the gold assignment is fewer than the number of clusters in the clustering, the more important measure becomes the homogeneity, since it is more important that the points assigned the gold-assignment label "bad" are not mixed with "good" points in different clusters, than them being placed together in one great "bad" cluster. The opposite is true for when the clustering has fewer clusters than the gold assignment. In order to compensate for this behaviour β will be set to weigh either toward completeness, when $\beta > 1$, or toward homogeneity, when $\beta < 1$. Of course $\beta = 1$ represents equal emphasis, as in the standard V-measure.

Since the number of clusters differs a lot from the clustering compared to the gold assignment, this needs to be compensated. In general, letting the number of clusters from the gold assignment be g, then $\frac{g}{k} > 1$ if there are more clusters in the gold assignment than in the clustering; $\frac{g}{k} = 1$ if there are equally many clusters; and $\frac{g}{k} < 1$ if, as in our case, there are fewer clusters in the gold assignment than in the clustering. So by setting

$$\beta = \frac{g}{k} \tag{4.12}$$

compensation for this difference is attempted. Visual inspection of the data shows that 5 clusters are reasonable, for both the first and second clustering. In the first clustering $g = 3$ leading to $\beta = \frac{3}{5} = 0.6$ and in the second clustering $g = 2$ leading to $\beta = \frac{2}{5} = 0.4$. Hence these values of β will be used to evaluate the clusterings.

4.6 Clustering methodology

The method for clustering the datasets, which are described in section 4.4, is described here. The final goal is to identify points corresponding to the goal set in section 4.2. In order to achieve this, hierarchical clustering is performed in two steps. The goal of the first step is to remove outliers. The valid points found in step one are clustered in the second clustering. The clusters that result from the second clustering are labelled according to their centroids. The points belonging to a cluster that is labelled as performance degrading are considered to be points corresponding to the goal set in section 4.2. Details on each step of the methodology is presented below.

To achieve a good clustering it is important to choose a distance measure that captures the similarities and differentiations in the data that the clustering is supposed to find. We have chosen to try three different distance measures and evaluate the result of clustering with each of them to determine which best suits the data and gold assignment. The three distances are chosen to extract different features in the data. Squared Euclidean distance aims to show the change in value equal in all dimensions. In city block distance the different dimensions are of greater importance. Mahalanobis takes into account the covariance of the data which makes large changes in a dimension that has great spread less important. Due to the great difference in covariance in the data, this distance metric is of high interest. For the sake of consistency the same measures are used in the first and second clustering, except for weighted Mahalanobis distance which is not covered.

4.6.1 First clustering

According to the developers, the raw data contains values that are results of faulty measurements. Some of the values almost double between jobs and given the background of the data that is not a probable case. An update that makes the performance so much worse should not pass other tests in the suite. Therefore it is clear that those very high differences come from erroneous testing.

Most of the data have very small difference values. This is apparent when performing a histogram plot of the data, see appendix figure B.5. The values of interest to achieving the goal are located close to the peak in the histogram—the values that have slightly higher difference but are not faulty measurements. In order to find these values a way to remove the faulty measurements is needed. The dataset containing only difference values for each measure point, memory usage and DSP load, described in section 4.4.1, is considered to be the best to cluster on for this problem. This is because the outliers are identified based on their exceptional difference values.

As previously stated it is possible to either cluster each test on its own and remove points which are regarded as outliers in that test or cluster on all data points at once and choose outliers based on that clustering. In order to asses the performance of per-test clustering, each test-clustering was evaluated using the evaluation strategy described in section 4.5. Table 4.2 shows an excerpt from table C.2, which contain these results, it is apparent that different tests yield very different V-measure. Also the clustering for memory usage follows the same principles and can therefore be treated the same way without being considered separately. For reference, the memory usage table is available in the appendix as table C.1.

Some tests receive a higher V-measure, as is the case for test 5, while many of them receive a score of zero, e.g. test 15. Figure 4.5 shows the difference values for two different tests with clusters marked in different colours and the gold assignment marked with red lines. A quick glance at this figure shows that some tests follow the overall structure of all of the data pretty well, while some might have all their values grouped up in one gold-assignment section, for example around zero. Consequently, the tests containing only points belonging to a single class will, due to the nature of the measurements, get a score of zero. This is a reasonable result since points which are not that extreme in the bigger picture, when considering all data, might be marked as outliers within their own tests. Thus, clustering on a per test basis will not provide a solid and good foundation to build the algorithm on. Hence, this direction is explored no further.

The clustering of one-dimensional data is performed with three different distance measures: squared Euclidean, city block and Mahalanobis distance. For each distance the number of clusters $k = 3,5,7,10$ are used. This gives a substantial set of clusterings. In order to differentiate the clustering that provides the best partition of the data, the eval-

test number	h	c	$v_{\beta=1}$	$v_{\beta=0.6}$	ARI
5	1.000	0.192	0.322	0.388	0.206
15	0.000	0.000	0.000	0.000	0.000

Table 4.2: Clustering per test for one dimensional difference data of DSP. Part of table C.2.

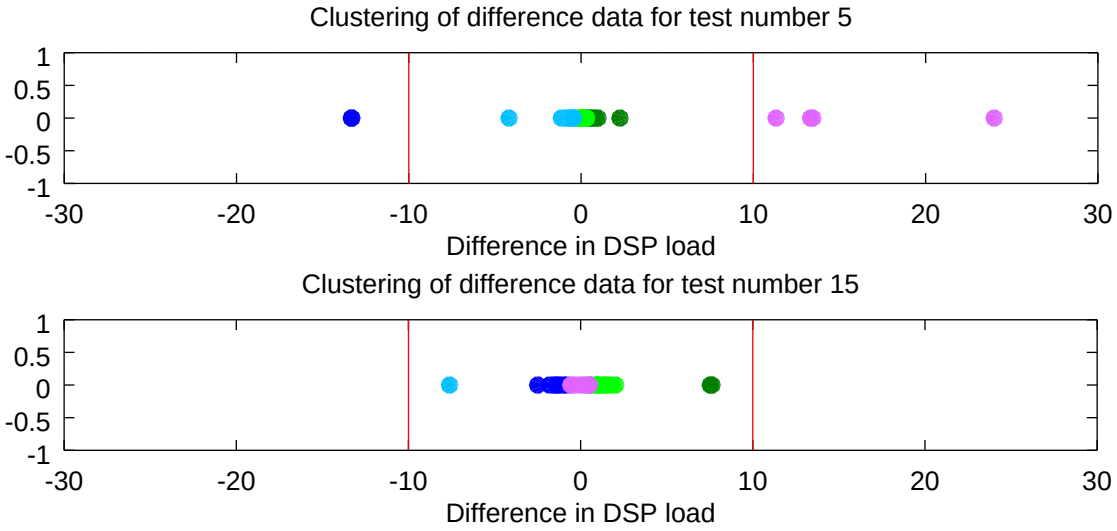


Figure 4.5: Scatter-plots for clustering in 5 clusters with city block distance for difference values of DSP load. One good and one bad matching test is shown.

uation technique described in section 4.5 is used. For the first clustering this means that the clustering with the highest $v_{\beta=0.6}$ value is the best clustering. Using that clustering it becomes necessary to choose which clusters are to be considered as containing valid points and which clusters should be determined as containing outliers.

The differentiation between valid and invalid clusters is made using the centroids of the clusters. The goal is to capture the clusters that are within the intervals set by the gold assignment as interesting clusters. Therefore if the centroid is within the gold assignment limit then the cluster will be included as valid points. The limits for centroid labelling in the first clustering is therefore set to $[-10,10]$ for both memory and DSP.

4.6.2 Second clustering

The datasets for the second clustering are constructed using only the valid points found in the first clustering, i.e. the dataset without outliers. The two- and three-dimensional datasets described in section 4.4 are used for the second clustering. That gives three different kinds of datasets for clustering.

Before clustering, the distances need to be adjusted for two- and three-dimensional datasets. This is due to the scale of the features of each data point being quite broad. As mentioned in section 3.4.3 it is appropriate to use weighted distances in order to compensate for this. City block and squared Euclidean distance can be weighted according to section 3.4.3. It is possible to weight Mahalanobis distance but since it already is a weighted squared Euclidean distance in itself the result would be equal to some weighting of squared Euclidean. Therefore a high number of possible weights for squared Euclidean and city block distance and not move forward with the Mahalanobis distance was decided to try.

The parameters that determine the result of clustering one dataset are distance, number of clusters and weights. For each experiment those three parameters need to be optimised in order to obtain the best possible clustering of that dataset. The procedure for finding the best possible clustering of each dataset is:

1. Set distance to squared Euclidean or city block
2. Choose $k = 3,5,7,10$ that gives the highest $v_{\beta=0.4}$ value
3. Choose $w = w_1,w_2,w_3$ that gives the highest $v_{\beta=0.4}$ value

This procedure yields the best clustering for each dataset for both distances. The best possible clustering of every dataset is the one among those that has the highest $v_{\beta=0.4}$ value. Accordingly the best possible clustering for solving the problem at hand is the dataset clustering that has the highest $v_{\beta=0.4}$ value. This is because the $v_{\beta=0.4}$ value is calculated on Ericsson's gold assignment. The best clusterings should be the ones that also show performance degradation the best. To distinguish between clusterings with similar v_β, a detailed analysis of which points that are assigned to which cluster will also be performed. In order to compare this to the gold assignment by Ericsson it becomes necessary to first decide which clusters contain points with performance degradation.

The clusters found should be connected to the goal described in section 4.2 by containing points that do or do not show performance degradation. Therefore, to be able to classify the points, the clusters themselves need to be labelled. The clusters are represented by their centroids so labelling the centroids will mean labelling the cluster. We choose to

label the centroids using the gold assignment provided by Ericsson, this gives a labelling on each centroid depending on its difference, trend and value.

When the points, that are identified as performance degrading by our system, are found they are compare to the points marked as performance degrading by Ericsson. The points found by our system should be the same points—not more or less. This analysis, together with the evaluation measures, contributes to deciding which clustering is the best for the problem and how well it corresponds to the goal in section 4.2.

5

Results

PLOTS AND TABLES showing results obtained by carrying out the methods explained in the previous chapter are shown here. The chapter also contains explanations of the results and possible reasons for our observations.

5.1 First clustering

It is clear that some of the data result from faulty measurements, see section 4.6.1. The aim of the first clustering is to identify clusters that contain outliers and remove those points from further investigation.

The data that show the difference in values between two points in a test is described in section 4.4.1. It is divided into sets of difference in DSP value and difference in memory value in accordance with equation 4.6. The clustering algorithms k-means and k-medoids are run on both the datasets.

The clustering is run with different settings for k and distance. Note that k-medoids is only used with Mahalanobis distance since that distance measure is not theoretically applicable for k-means; otherwise the k-means algorithm is used. A clustering algorithm will assign a clustering label to each data point. This partition of the data is to be

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.6}$	ARI
squared Euclidean	3	0.608	0.911	0.729	0.695	0.768
city block	3	0.608	0.911	0.729	0.695	0.768
Mahalanobis	3	0.273	0.766	0.403	0.360	0.474
squared Euclidean	5	0.946	0.100	0.181	0.227	0.100
city block	5	0.946	0.100	0.181	0.227	0.100
Mahalanobis	5	0.497	0.024	0.047	0.060	0.031
squared Euclidean	7	0.909	0.054	0.102	0.132	0.052
city block	7	0.909	0.054	0.102	0.132	0.052
Mahalanobis	7	0.555	0.016	0.031	0.040	0.013
squared Euclidean	10	0.963	0.034	0.066	0.085	0.025
city block	10	0.963	0.034	0.066	0.085	0.025
Mahalanobis	10	0.575	0.012	0.023	0.031	0.007

Table 5.1: Clusterings of one-dimensional data, (d), of DSP load before outlier removal. Compared to outlier gold assignment.

compared to the gold assignment described in section 4.5. The comparison aims to place a value on how similar the partitions are. In this thesis the V-measure, section 3.4.4, and the ARI, section 3.4.4, are chosen to analyse the results of clustering. In V-measure, h stands for the homogeneity of the clustering and c for the completeness of the clustering. Note that both the weighted, with $\beta = \frac{3}{5}$, and standard V-measure are presented. Tables 5.1 and 5.2 show the resulting evaluation values for the different runs.

In tables 5.1 and 5.2 it is clear that the V-measure depends more on the number of clusters than the chosen distance metric. As mentioned in section 3.4.4, this is one of the known drawbacks with V-measure. It is however possible to examine the differences between distances for the same number of clusters. It is notable that the squared Euclidean and the city block distances have the same values for all clustering and also the same values for homogeneity and completeness. This is related to the placement of data points leading to the measures being equal in one dimension.

The best clustering is primarily chosen to be one that has the highest value for $v_{\beta=0.6}$ since that measure is best adapted to the data and the gold assignment. However, the measures follow each other so well that the same distance and number of clusters are chosen regardless of which measure chosen to look at. In table 5.2, note that using Mahalanobis distance for creating 3 clusters yields an evaluation measure of 1 for all different measures. This suggests that the clustering follows the gold assignment perfectly. Mahalanobis for 3 clusters is therefore chosen as algorithm for first clustering of memory

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.6}$	ARI
squared Euclidean	3	0.868	0.255	0.394	0.456	0.368
city block	3	0.868	0.255	0.394	0.456	0.368
Mahalanobis	3	1.000	1.000	1.000	1.000	1.000
squared Euclidean	5	1.000	0.111	0.200	0.250	0.138
city block	5	1.000	0.111	0.200	0.250	0.138
Mahalanobis	5	1.000	0.111	0.199	0.249	0.137
squared Euclidean	7	1.000	0.040	0.077	0.100	0.036
city block	7	1.000	0.040	0.077	0.100	0.036
Mahalanobis	7	1.000	0.023	0.045	0.059	0.012
squared Euclidean	10	1.000	0.026	0.050	0.065	0.018
city block	10	1.000	0.023	0.045	0.059	0.015
Mahalanobis	10	1.000	0.015	0.030	0.040	0.006

Table 5.2: Clusterings of one-dimensional data, (d), of memory usage before outlier removal. Compared to outlier gold assignment.

data. For DSP load no clustering is perfect, however, city block distance gives very high results on all measures using 3 clusters. Since the $v_{\beta=0.6}$ is the highest for city block, $k = 3$, it is the chosen distance measure for first clustering of DSP load data.

The clustering of all data that got the best result in tables 5.1 and 5.2 is used to remove the outliers. The data that should be kept for further experiments are the ones considered as valid points in the gold assignment. Therefore the clusters that have a centroid within the interval described in section 4.6.1 will be described as valid points. In figures C.1 and C.2 the points that are removed are red and the valid points blue.

5.2 Second clustering

The second step towards achieving a clustering capable of separating points causing performance degradation from points that do not, is to cluster the data without outliers. As stated in section 4.4 it is important to have a dataset that corresponds well to the problem at hand. We try three different datasets and evaluate which is the best representation of the data by calculating evaluation measures for a number of different clusterings on the datasets. This shows which dataset and clustering method corresponds best to Ericsson's gold assignment.

Dataset	Distance	k	w_1	w_2	w_3	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
(d,d_t)	squared Euclidean	5	0.5	0.5	–	0.480	0.078	0.134	0.194	0.080
(d,d_t)	city block	5	0.5	0.5	–	0.502	0.080	0.138	0.201	0.082
$(d_r,d_{t,r})$	squared Euclidean	5	0.1	0.9	–	0.441	0.076	0.130	0.186	0.086
$(d_r,d_{t,r})$	city block	5	0.1	0.9	–	0.441	0.076	0.130	0.186	0.086
(d,d_t,v_n)	squared Euclidean	5	0.2	0.3	0.5	0.479	0.079	0.136	0.196	0.082
(d,d_t,v_n)	city block	5	0.4	0.4	0.2	0.501	0.082	0.141	0.204	0.085

Table 5.3: Best weights and k for all dataset-distance combinations, DSP load data.

Dataset	Distance	k	w_1	w_2	w_3	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
(d,d_t)	squared Euclidean	3	0.1	0.9	–	0.678	0.199	0.308	0.402	0.390
(d,d_t)	city block	3	0.2	0.8	–	0.710	0.233	0.351	0.448	0.438
$(d_r,d_{t,r})$	squared Euclidean	3	0.5	0.5	–	0.702	0.221	0.336	0.433	0.421
$(d_r,d_{t,r})$	city block	3	0.3	0.7	–	0.702	0.221	0.336	0.433	0.421
(d,d_t,v_n)	squared Euclidean	3	0.1	0.2	0.7	0.710	0.233	0.351	0.448	0.438
(d,d_t,v_n)	city block	3	0.1	0.4	0.5	0.710	0.233	0.351	0.448	0.438

Table 5.4: Best weights and k for all dataset-distance combinations, memory usage data.

The clustering on the datasets is performed with both weighted and un-weighted distance measures. Since the weighted distances give better results they are shown here. The optimal weight and k for each dataset is listed in table 5.3 for DSP load and in table 5.4 for memory usage. In appendix D.1 the results for clustering with un-weighted distances are shown for comparison.

The best clustering for each dataset is the one with the highest $v_{\beta=0.4}$ value. It is apparent that when the distances give different results, city block distance is more successful. Therefore city block distance is chosen for all clustering, creating 5 clusters in the DSP load data and 3 clusters in memory usage data. The weights used are the corresponding ones in tables 5.3 and 5.4. These clusterings are plotted, with colours representing each cluster, in appendix figures D.1 to D.6. After performing the clustering, the clusters are analysed and in order to decide which clustering method gives the most accurate depiction of the data. The v_β evaluation itself cannot do this since the values are so similar for all clusterings. Therefore, the points that the clustering would mark as points showing sign of performance degradation are retained and compared against the gold assignment from Ericsson. As stated in chapter 4 the clustering's centroids are labelled according to the gold assignment and all points belonging to a certain centroid share the labelling of that centroid.

For each of the three different clusters, for DSP load and memory usage respectively, a plot of all data points as they are represented in that model is created. The points are then colour coded according to the gold assignment: green points are considered *performance harmless* while red points are considered *performance harmful* by Ericsson. The points labelled as *performance degrading* by the corresponding clustering is then circled with a blue circle. The goal of this visual inspection is to get a feel for which points are labelled what and why certain points are clustered in a certain cluster. In a perfect clustering all the red points would be circled while none of the green points are. An example of these plots can be seen in figure 5.1; the rest of these figures reside in appendix D.3, figures D.10 to D.8.

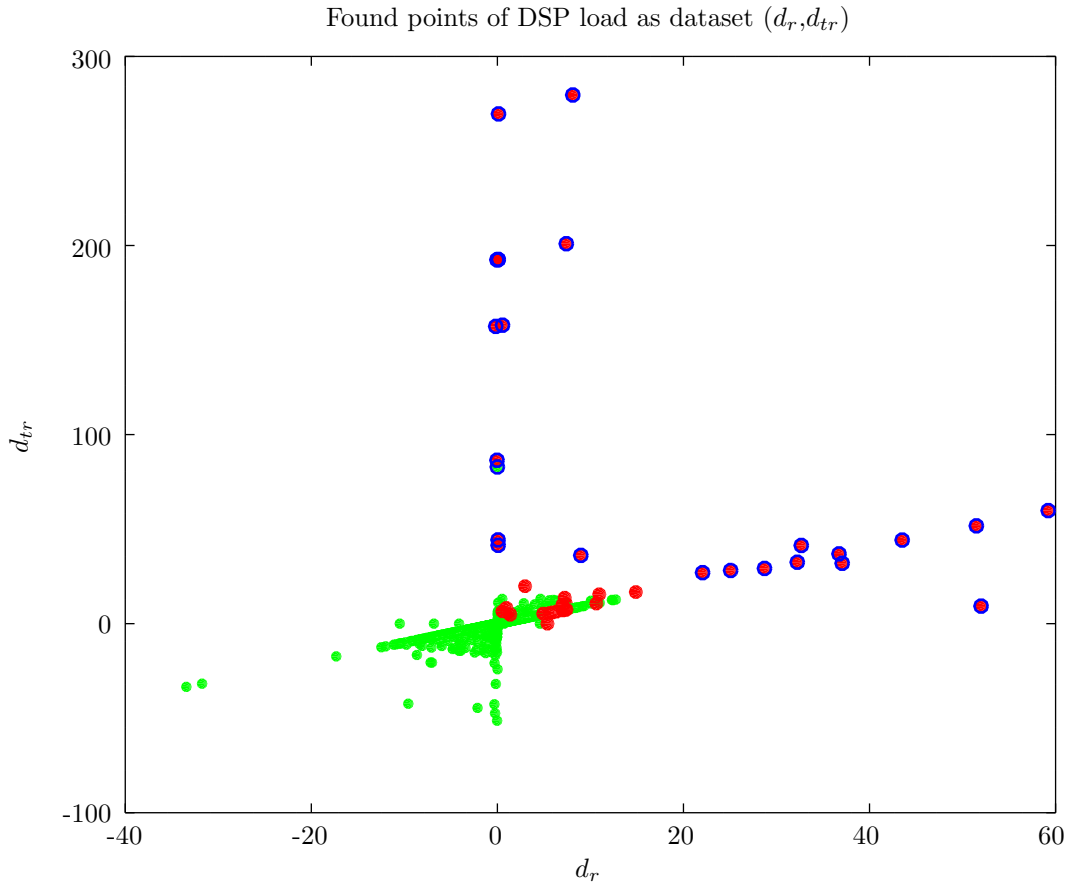


Figure 5.1: According to Ericsson's gold assignment, green points are categorised as performance-harmless points and red as performance-harmful points. The circled points are found by the fault detection system in this thesis.

All of these figures are also condensed into two tables, 5.5 and 5.6, showing the number of points labelled as performance degrading found and how many of those that are actually

considered performance harmful. This yields a quick overview of whether the clusterings are labelling performance harmless points as performance degrading, or failing to label performance harmful points as performance degrading. The points stated as missing in the tables are the total amount of points labelled by Ericsson minus the found points that are correct. The found points that are not correct are stated as false positives in the tables.

Dataset	Found	Correct	Missing	False positives
(d,d_t)	24	24	19	0
(d,d_t,v_n)	24	24	19	0
$(d_r,d_{t,r})$	23	22	21	1

Table 5.5: For DSP load data. Table over number of points found (labelled as performance degrading) by the best clustering for each dataset. Correct is in relation to the gold assignment from Ericsson. In total there are 43 points to be found for DSP load.

Dataset	Found	Correct	Missing	False positives
(d,d_t)	48	17	0	31
(d,d_t,v_n)	48	17	0	31
$(d_r,d_{t,r})$	50	17	0	33

Table 5.6: For memory usage data. Table over number of points found (labelled as performance degrading) by the best clustering for each dataset. Correct is in relation to the gold assignment from Ericsson. In total there are 17 points to be found for memory usage.

We can see that for memory usage, table 5.6, all datasets are labelling more points as performance degrading than what is correct, however, the relative dataset $(d_r,d_{t,r})$ includes two more false positives than the others and can thus be considered worse. In addition, with regards to DSP load the relative dataset labels one less point as performance degrading in comparison to the others, and even labels one incorrectly. This can also be seen in figure 5.1 where one green point is circled, around the coordinates (0,90).

The tables 5.5 and 5.6 also show that the datasets (d,d_t) and (d,d_t,v_n) identify the same amount of points. A more detailed comparison shows that the sets identify exactly the same points, which explains the evaluation measures being very similar between the two datasets. This result shows that the points that are distinguished by value, i.e. the points in the sensitive section, are not identified by any of the datasets. A number of steps are taken to correct this. Constructing the relative dataset is one step in trying to achieve a better clustering for points in the sensitive area, but as seen above it is unsuccessful. Since one part of the problem is the different scales on d, d_t and v_n experiments where also made with $(d,d_t,v * s)$ with s as different constant scaling of v. The resulting evaluation using V-measure and ARI on these experiments are worse then

the experiments already presented in this report and are therefore not included. Further discussion on how to improve the clustering with respect to the sensitive area are found in section 6.2.

Based on the values in tables 5.5 and 5.6, and the V-measures in tables 5.3 and 5.4 the overall best clustering method was chosen for DSP load and memory usage respectively. It is concluded that (d_r,$d_{t,r}$) is worse than (d,d_t) and (d,d_t,v_n). (d,d_t) and (d,d_t,v_n) perform equally well but since two dimensions are easier to visualise (d,d_t) is chosen as the best and most useful dataset. Thus, the clustering used for identifying performance degradation is shown in figures D.1 and D.4.

6

Discussion

Discussion regarding the results is conducted here, both on details in the clustering algorithm and in the broader context of the field of fault detection. Future work related to the project as well as possible implementations are discussed.

6.1 Finding performance degrading updates

Once the centroids, and with them the clusters, are labelled the points that show performance degradation can be distinguished. These points correspond to one test case of a job that consists of a number of test cases. Our hypothesis is that if one test case shows performance degradation the job should be considered bad. One could also consider using the amount of test cases for each job that shows performance degradation as a probability for it being a bad job.

Each such job is marked with golden lines in figure 6.1 (DSP load) and figure 6.2 (memory usage). These are the jobs that a fault detection system based upon our work would warn for, if used in production. Note that figures 6.1 and 6.2 are zoomed-in versions of the total result shown in appendix figures D.13 and D.14. This can be seen as an alternative way to visualize the result of our fault detection system, since these are the updates it would give a warning for if it was used in production.

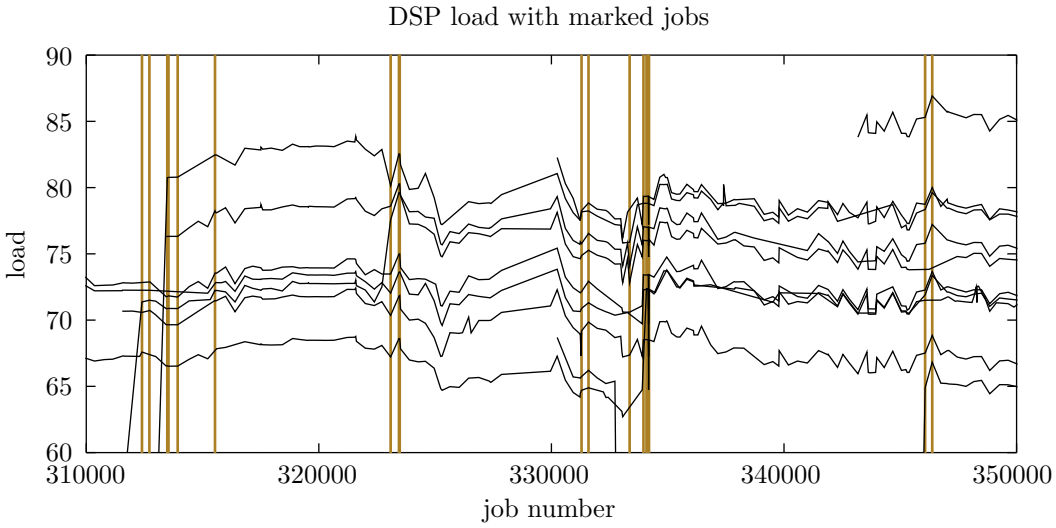


Figure 6.1: Graph of all load data for DSP load with marks for jobs showing signs of performance degradation. Found with (d,d_t), city block distance, 5 clusters, $w_d = 0.5$, $w_{d_t} = 0.5$.

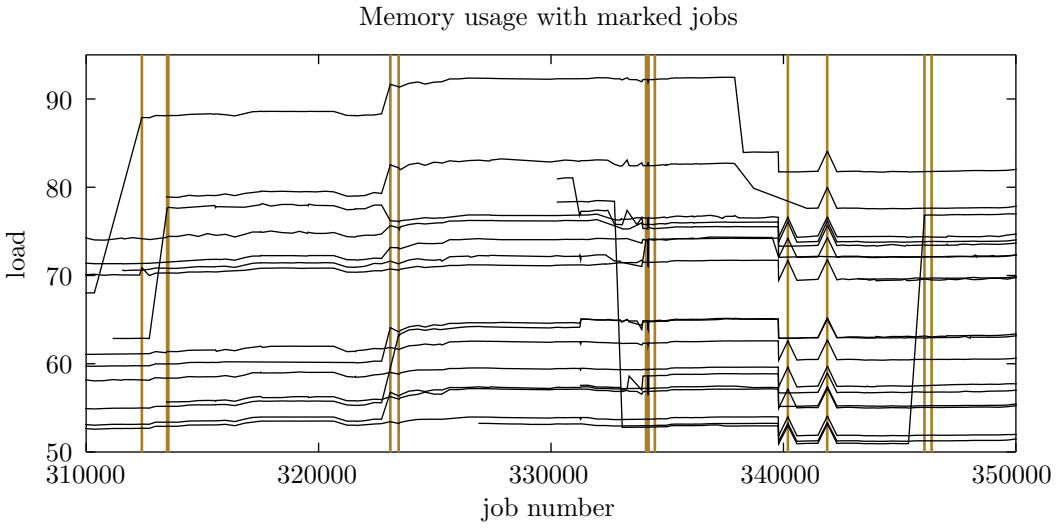


Figure 6.2: Graph of all load data for memory with marks for jobs showing signs of performance degradation. Found with (d,d_t), city block distance, 3 clusters, $w_d = 0.2$, $w_{d_t} = 0.8$.

6.2 Actual value

As can be seen in figure 6.3 some points which are considered performance-harmful, drawn in red, appears to reside in the middle of a swarm of performance-harmless points, drawn in green. This is because Ericsson has a more strict limit for tolerance in increase when it comes to points which already have a high actual value, however, this is not captured in the current models. An attempt to capture this feature has been made with the (d, d_t, v_n) and $(d_r, d_{t,r})$ clusterings.

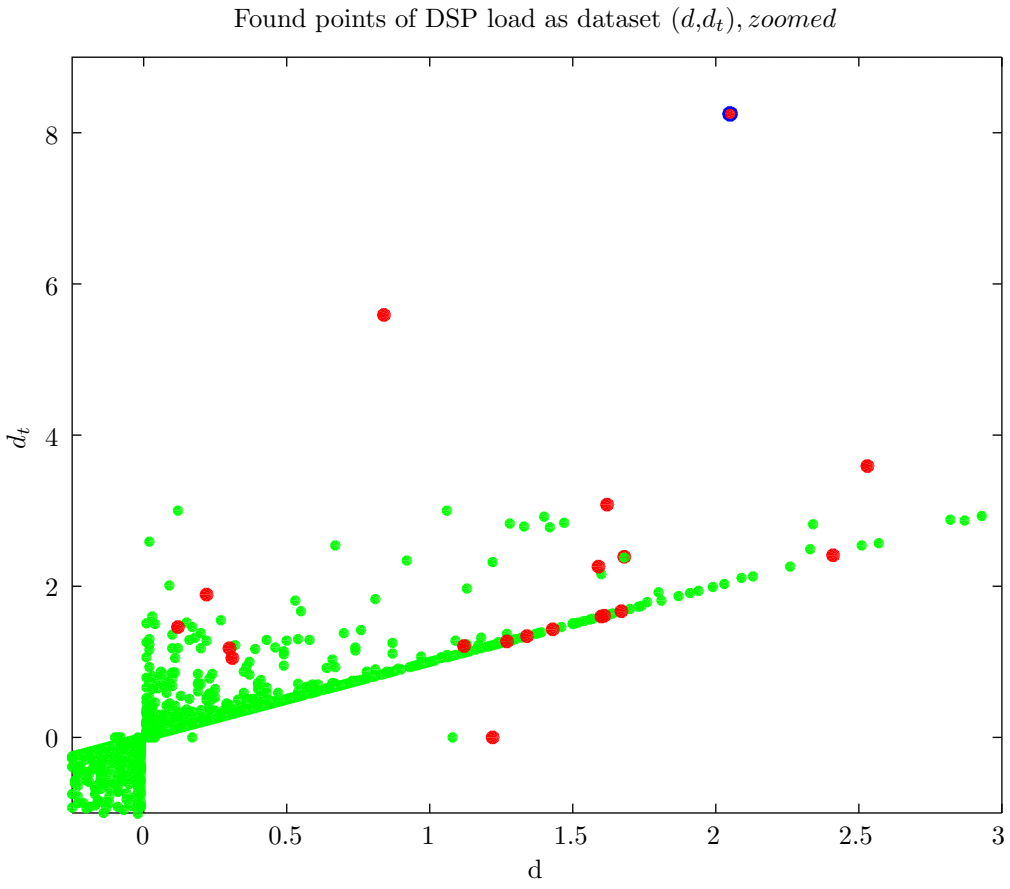


Figure 6.3: Zoomed in version of figure D.7.

The three-dimensional clustering, (d, d_t, v_n), tries to resolve this issue by incorporating the actual value normalised, i.e. divided by 100. This, however, does not change the clustering since the distance is not affected a lot when using such small values. On the other hand using the actual value without normalising it would yield a clustering that is heavily focused on the actual values, which is not that interesting since it still has to

be a performance harmful point and not just have a high value.

The relative clustering, $(d_r, d_{t,r})$, addresses this problem by incorporating the actual value into the difference values using the log function. However, as seen in figure 5.1, this instead causes one of the performance-harmless points to be relocated to the coordinates (0,90) and thus being clustered as performance degrading. This could be tolerated if it had also provided a major improvement in finding the performance harmful points hidden in the swarm of harmless points, however, this is not the case and as such this variation is not optimal.

6.3 Fault detection system

In section 3.2 the field definition of a fault detection system is reviewed and the system constructed in this thesis aims to achieve that standard. In that section a list, figure 3.1, is presented, that states a number of demands on a fault diagnostics system and a subset of those requirements are applicable on a fault detection system. In order to perform a qualitative evaluation of how well the fault detection system developed achieves the standard of a fault detection system, this section provides a concise report on how well the thesis system fulfils the appropriate requirements from the list.

Early detection. By utilising the data classification provided by our method it is possible to detect performance degradation as soon as the data are generated. In this setting it means that the problematic update can be stopped before affecting the software.

Multiple fault identifiability and isolability. The faults detected by the system in this thesis correspond to either performance degradation leading to increased memory usage or increased use of DSP cycles, increasing the load. It will be clear for the users of the system which kind of fault is identified. Whether or not there actually is a fault in the code update performed or if the changes in load and usage data depend on something else, for example an update in the testing system or in the operating system, will not be determined by the fault detection system developed. Therefore the isolability of the system is lacking, although due to exterior effects.

Robustness. Since the fault detection system implements hierarchical clustering to remove outliers it should be robust to faulty data. If however the dataset changes, to contain for example a great amount of points with difference value higher than zero, the clustering will be affected and the values from the evaluation of clustering might change. That could imply that a metric other than the chosen distance and number of clusters

would be optimal. Since the result of clustering with the different distance metrics is not greatly different this should probably not affect the result of the fault detection system, but there is of course that risk associated with choosing a distance measure based on present data and not planning to update the choice.

Adaptability. It is in the nature of a machine learning algorithm to adapt to new data. The clustering will change with new input which will make the labelling of points as performance degrading improve as more data are added. This is one of the greatest advantages of this fault detection system compared to the threshold system previously available. A threshold has to be set and changed by the developers, but the clustering adapts itself instead. One interesting feature of this is that a point that passed the system a number of jobs ago could be labelled as bad if run again after new data were added. This is caused by the build in comparison to other data. It might seem illogical that a point update is not always good or bad, but it can also be considered an advantage that the system evolves.

Reasonable storage and computational requirements. The data and aggregated points require a large amount of storage, however, not more than the standard of a normal computer and therefore is considered reasonable. Noted should be that the dataset is growing every day, which means that it might reach an unreasonable size. It is possible to remove around half the dataset since it corresponds to tests that generate a low load and therefore does not contribute to the fault detection. This is shown in section 4.6.1. It is not within the scope of this thesis to optimise the number of tests but it is easily implemented. Although the dataset is growing, it is not big enough to obtain any substantial differences in computational requirements. Still, measures have been taken to lower the computational cost, for example the squared Euclidean distance is used instead of the Euclidean. The clustering algorithm, though complex, is among the simpler solutions for the problem. Another reasonable approach would be to implement a neural network for the problem. It is likely that such an approach would have substantially bigger computational requirements than the solution proposed.

Overall the system fulfils the requirements of a fault detection system and can therefore be implemented as such.

6.4 Future work

There are some ideas which we have come up with that could advance this area and further delve into the topic of this thesis. Firstly the fault detection could be improved by creating consistent clustering and incorporating the actual value into the result. Secondly

a monitoring system could be built to survey the performance of the type of system discussed. Lastly the decision process, now utilising clustering, could be represented with the use of an Artificial Neural Network (ANN) instead.

6.4.1 Consistent clustering

The ability to distinguish between normal and fault is in section 6.3 refereed to as isolability and is an important part of any fault detection system. As can be seen in section 5.2 the system in this thesis warns for memory degradation in more points than the ones deemed to be performance harmful by the gold assignment. On the other hand, using DSP data it warns for fewer points than those classified as performance harmful. This big difference in behaviour can become problematic when the fault detection system is used in practice. If it is better to warn more times or only when the warning is more certain depends on the usage of the system. When utilised in a critical environment a more generous warning system will be safer, on the other hand in less critical setting an over-ambitious warning system can be the cause of irritation and risk being shut down or ignored. In order for the developers to get the choice between the two variations the system has to take on one of the characteristics, not both of them as is the situation today. Streamlining this behaviour is a possible improvement to the system.

6.4.2 Incorporating actual value

As discussed in section 6.2 the current fault detection system does not incorporate the actual value in a good way. This is something which would be useful to successfully perform. Also finding a general way do this, within this problem domain, would be useful since other problems also could benefit from such a distinction.

6.4.3 Monitoring system

The work presented in this thesis could be used to create a monitoring system. After the initial clustering is made on all existing points, two approaches are available. The first is to label each new data point generated according to which centroid is closest. The second is to completely rerun the clustering algorithm each time a set number of new points becomes available. The first of those strategies has the advantage of being quick to perform, while the second can adapt according to new normal conditions.

After classifying the new points according to one of the strategies above, the update corresponding to each classified point can be determined to be regarded as an outlier, conceived during an error in measurement; a "bad value" that should generate a warning for the product owner or developer; or a "good value" that needs no further attention.

6.4.4 Artificial neural networks

An ANN can be used to model the outcome of a function given a set of input data, even if the function itself is unknown. This approach is easier when dealing with labelled data since these labels can be used to validate the output of the network in order for it to learn and make a better assessment of the next point during training.

Since the ANN will need training before labelling new points, the work of this thesis could give a way to label the data in order to allow that training. The ANN could then be used to classify the new points as they arrive.

7

Conclusions

The goal of this project is to use clustering to develop a fault detection system capable of identifying performance degradation. The raw data consist of performance measurements of memory usage and DSP load. From these, aggregated points on trend and difference between measurements are calculated. Together with the raw data values they are placed in three different datasets which aim to correspond to the problem of identifying points that have a high increase in difference or trend and points that have a smaller increase but a high starting value. These datasets are then analysed by using a two-step hierarchical clustering.

The first clustering removes outliers. Experiments here include using k-means and k-medoids algorithms with different k values and distance measures (city block, squared Euclidean and Mahalanobis). The clustering is evaluated using V-measure and adjusted Rand index (ARI) against a gold assignment for outlier removal.

The second clustering identifies points that show signs of performance degradation. To aid in evaluating this, a gold assignment was provided by Ericsson. The clustering experiments are performed using k-means with different k and both weighted city block distance and weighed squared Euclidean distance for a number of weights. The evaluation also includes a weighted V-measure in order to compensate for the small amount of clusters in the gold assignment.

In the second clustering, centroid labelling is performed to identify clusters which contain points that show performance degradation. The identified points are compared to the points labelled as performance harmful in the gold assignment provided by Ericsson. The comparison shows that the method successfully identifies a number of the points that Ericsson says show signs of performance degradation. The points identified by the method can be used to filter out some jobs that the developers should take a closer look at.

Bibliography

[1] G. Khanna *et al.*, "Application performance management in virtualized server environments," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*. IEEE, 2006, pp. 373–381.

[2] M. Andreolini *et al.*, "Dynamic load management of virtual machines," in *Cloud Computing: First International Conference, CloudComp 2009, Munich, Germany, October 19-21, 2009, Revised Selected Papers*, vol. 34. Springer, 2011, p. 201.

[3] S. Ohta and T. Hirota, "Machine learning approach to the power management of server clusters," in *Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 571–578.

[4] T. Hayashi and S. Ohta, "Performance degradation detection of virtual machines via passive measurement and machine learning," *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, vol. 5, no. 2, pp. 40–56, 2014.

[5] M. Monisha *et al.*, "A survey on the application of machine learning algorithms to predict software aging," *International Journal on Advanced Computer Theory and Engineering (IJACTE)*, vol. 2, no. 5, 2013.

[6] D. Simeonov and D. Avresky, "Proactive software rejuvenation based on machine learning techniques," in *Cloud Computing*. Springer, 2009, pp. 186–200.

[7] W. S. Cooper, "Decision theory as a branch of evolutionary theory: A biological derivation of the savage axioms." *Psychological Review*, vol. 94, no. 4, p. 395, 1987.

[8] L. Ahlin *et al.*, *Principles of Wireless Communications*. Studentlitteratur, 2006.

[9] L. Song and J. Shen, *Evolved Cellular Network Planning and Optimization for UMTS and LTE*. Taylor & Francis, 2010.

[10] J. Hagen, *Radio-Frequency Electronics: Circuits and Applications*. Cambridge University Press, 1996.

[11] L. Ahlin *et al.*, *Mobil radiokommunikation.* Studentlitteratur, 2001.

[12] K. Terplan and P. Morreale, *The Telecommunications Handbook.* Springer Berlin Heidelberg, 2000.

[13] B. Holdsworth, "4 - small computer architecture," in *Microprocessor Engineering*, B. Holdsworth, Ed. Butterworth-Heinemann, 1987, pp. 88 – 126.

[14] S. Heath, "5 - digital signal processors," in *Microprocessor Architectures (Second Edition)*, 2nd ed., S. Heath, Ed. Oxford: Newnes, 1995, pp. 148 – 171.

[15] M. Meyer, "Continuous integration and its tools," *Software, IEEE*, vol. 31, no. 3, pp. 14–16, May 2014.

[16] G. Stanley, "A guide to fault detection and diagnosis," 2013, from Operations Management Automation, Accessed: 2016-05-05. [Online]. Available: http://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/faultdiagnosis.htm

[17] S. Dash and V. Venkatasubramanian, "Challenges in the industrial applications of fault diagnostic systems," *Computers & chemical engineering*, vol. 24, no. 2, pp. 785–791, 2000.

[18] V. Venkatasubramanian *et al.*, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[19] V. Venkatasubramanian *et al.*, "A review of process fault detection and diagnosis: Part iii: Process history based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[20] G. Wang and S. Yin, "Data-driven fault diagnosis for an automobile suspension system by using a clustering based method," *Journal of the Franklin Institute*, vol. 351, no. 6, pp. 3231–3244, 2014.

[21] J. Du *et al.*, "Layered clustering multi-fault diagnosis for hydraulic piston pump," *Mechanical Systems and Signal Processing*, vol. 36, no. 2, pp. 487–504, 2013.

[22] H. He *et al.*, "Stock trend analysis and trading strategy," *Joint Conference on Information Sciences, Advances in Intelligent Systems Research*, 2006.

[23] J.-Y. Cheung and G. Stephanopoulos, "Representation of process trendspart i. a formal representation framework," *Computers & Chemical Engineering*, vol. 14, no. 4, pp. 495–510, 1990.

[24] M. R. Maurya *et al.*, "Fault diagnosis using dynamic trend analysis: A review and recent developments," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 2, pp. 133 – 146, 2007, special Issue on Applications of Artificial Intelligence in Process Systems Engineering.

[25] C. Sammut and G. Webb, *Encyclopedia of Machine Learning*, ser. Encyclopedia of Machine Learning. Springer US, 2011.

[26] A. M. TURING, "I.computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, 1950.

[27] I. Muhammad and Z. Yan, "Supervised machine learning approaches: A survey." *ICTACT Journal on Soft Computing*, vol. 5, no. 3, 2015.

[28] M. Kyan *et al.*, *Unsupervised Learning*. John Wiley & Sons, Inc., 2014.

[29] S. P. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[30] J. Kogan, *Introduction to Clustering Large and High-Dimensional Data*. Cambridge University Press, 2006.

[31] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[32] S. Theodoridis, "Chapter 12 - bayesian learning: Inference and the {EM} algorithm," in *Machine Learning*, S. Theodoridis, Ed. Oxford: Academic Press, 2015, pp. 585 – 638.

[33] E. W. Weisstein, "Distance," 2016, from MathWorld–A Wolfram Web Resource, Accessed: 2016-04-11. [Online]. Available: http://mathworld.wolfram.com/Distance.html

[34] A. Orlov, "Mahalanobis distance," 2011, from Encyclopedia of Mathematics, Accessed: 2016-04-11. [Online]. Available: http://www.encyclopediaofmath.org/index.php?title=Mahalanobis_distance&oldid=17720

[35] M. Wolfel and H. K. Ekenel, "Feature weighted mahalanobis distance: improved robustness for gaussian classifiers," in *Signal Processing Conference, 2005 13th European*. IEEE, 2005, pp. 1–4.

[36] C. D. Manning *et al.*, *Introduction to Information Retrieval*, 1st ed. GB: Cambridge University Press - M.U.A, 2008.

[37] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure." in *EMNLP-CoNLL*, vol. 7, 2007, pp. 410–420.

[38] K. Kazuaki, "Empirical comparison of external evaluation measures for document clustering by using synthetic data," *IFAT*, vol. 2014, no. 1, pp. 1–7, 2014.

[39] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[40] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[41] L. Åsgård and C. Ellgren, *Ericsson, Historien om ett svenskt företag.* Norstedts, 2000.

[42] S. Ericsson, SE-164 83 Stockholm, "This is ericsson," 2015, accessed: 2016-02-01. [Online]. Available: http://www.ericsson.com/res/thecompany/docs/this-is-ericsson.pdf

[43] O. Aftab *et al.*, "Information theory and the digital age," 6.933 - Final Paper, The Structure of Engineering Revolutions, Massachusetts Institute of Technology, Cambridge, Tech. Rep., 2001.

[44] A. Richardson, *WCDMA Design Handbook.* Cambridge University Press, 2005.

[45] R. Horak, *Telecommunications and data communications handbook.* Wiley-Interscience, 2007.

[46] International Telecommunication Union, "Reference model of open systems interconnection for ccitt applications," ITU-T Recommendation X.200, 1988.

[47] S. Haykin and M. Moher, *Modern Wireless Communications.* Pearson/Prentice Hall, 2005.

[48] 3GPP, "Release 1999," 1999, accessed: 2016-02-16. [Online]. Available: http://www.3gpp.org/specifications/releases/77-release-1999

[49] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications.* Wiley, 2000.

[50] H. Holma and A. Toskala, *HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications.* Wiley, 2007.

[51] W. Stallings, *Wireless Communications and Networks.* Pearson Prentice Hall, 2005.

A

Background

All detailed information about the setting of the software of interest for this thesis is presented here. For example its explained what place it has in our society and what job it conducts when operational. Some general background, e.g. the parts presented in chapter 2, is needed to understand this chapter.

A.1 Company background

Ericsson is a large company within the phone communication industry. Today they design and build both hardware and software used for everything from cell phone calls to connecting cars and electrical meters to the Internet.

Ericsson was founded in 1876 as a telegraph repair workshop by Lars Magnus Ericsson in Stockholm. 1879 LM Ericsson started to produce its own telephones. Ericsson's telephones were improvements of Siemens' telephones, which Lars had disassembled to figure out how they worked [41]. Around the time for WWII Ericsson decided that the future of Ericsson would lie in the transmission systems rather than building phones. Exchanging the old mechanical switches with the faster crossbar switches during 1940s granted Ericsson an even greater market abroad.

In 1981 Ericsson was involved in creating and launching the first fully automatic cellular phone system (NMT, 1G) in Saudi Arabia and, one month later, in Sweden. In 2009 Ericsson launched the world's first 4G Long-Term Evolution (LTE) network and, together with Verizon, transmitted the first data call on a 4G network. All along in the history of Ericsson, the company has constantly been pushing the boarders of telephony, from big bulky wall-mounted telephones to launching the first cellular network to use with mobile telephones. Later pushing their bailiwick to adopt to new 3G, and later 4G LTE, standards, where Ericsson also took part in the development of those standards.

With the market for telecommunication reaching almost all the people on the planet, Ericsson is taking the step to connecting devices. Their vision for 2020 is 50 billion connected devices, providing a wide variety of services [42]. From coffee ready when you wake up to unlocking your house when the carpenter comes around, the possibilities are endless, as well the potentials for the software industry.

A.2 Multiple access

Within a cell there might be multiple UE that want to communicate with the RBS. If every UE just sent its message to the RBS on the same frequency, they would all interfere with each other's signals and no one would be able to get through.

Imagine a room full of people who all want to talk to a coordinator. If everyone just speak their piece the coordinator will have no chance in differentiating who said what, unless someone talks stronger than the rest and drowns their voices. However, if only the strongest voice gets through everyone will start to yell to get their information delivered and that is an unsustainable solution.

There are a few different ways to solve the problem of granting Multiple Access to the same station. Some techniques, such as TDMA, section A.2.2, created for an earlier generation can still be used for some channels or functions in newer generation protocols.

A.2.1 FDMA

The first solution, which was implemented and used in 1G-networks, is called Frequency Division for Multiple Access (FDMA). This solution is based on the idea that each UE gets its own pair of frequencies, one to use for sending data, UL, and one used for receiving data, DL.

In the analogy with the room full of people, we can think of FDMA as if we were to force

all people to use different pitches, then the overall volume could be kept at a reasonable level while the multitasking coordinator would be able to differentiate between different people's information.

However, the amount of available frequencies are a limited resource and as the number of UE increases dramatically, the available frequencies has to be split amongst these UE.

A.2.2 TDMA

Another way to solve the issue of Multiple Access is to assign a unique time slot to each UE, during which only that UE is allowed to transmit data. This technique is called Time Division for Multiple Access and is commonly used in 2G-technologies. It requires the UE to packet big pieces of information in order to fit them into a single time slot.

This technique is easy to imagine, we just assign each person in the room with a time slot. If there are ten people in the room then we might assign each person a six second period where only that person is allowed to talk to the coordinator. This keeps the communication clean. However, the person can only talk during those six seconds then s-/he has to wait for nearly a minute until it is her/his turn again.

A.2.3 CDMA

A third solution is to let all UE talk at the same time on all frequencies but encoded. This is also a technology developed for 2G-networks, known as Code Division for Multiple Access. The trick is to mask the signal of the UE with a code of a pre-specified length, see section A.2.4. Masking each bit with a sequence of zeros and ones using XOR gives the possibility of using XOR between the coded sequence and the code once more in order to recover the original transmission. In order to not interfere with the transmission of other UE, the codes used must be orthogonal to one another. This will make it possible to extract a single original transmission from the composite wave that is being transmitted between UE and the RBS.

Another way for the coordinator to distinguish between information from different people is by letting them speak different languages. The multitasking coordinator can then just collect all the communication and afterwards sort out who said what, based on what languages the information has. This way every one can speak at the same time, use a reasonable sound level and still get their message delivered without interference or interfering with anyone else's transmission.

A.2.4 WCDMA

WCDMA, Wideband CDMA, is based on the same fundamental idea as CDMA. It divides different UE by giving them separate orthogonal codes. However, WCDMA offers a much faster speed for data transmission by, as the name suggests, using wider bandwidth with frequency bands 5MHz wide instead of 1.25MHz which regular CDMA uses. Since all UE within a cell are sharing the same frequencies, they are not as limited as in FDMA, instead spreading a signal over several frequencies can even save battery for the UE [43], as well as allow for faster transmission.

Chips and spreading

To be able to utilise a broader spectrum of frequencies, spreading of the signal is required. The length of the codes decides both how many UE may be connected at any one point in time and also how high data transmission rate is available. The code, known as spreading code, basically multiplies each symbol sent a set number of times equal to the predefined spreading factor (SF), see figure A.1. The size of each individual encoded copy is one *chip*. The spreading factor is 2^m where m is between 2 and 8 for the up-link and between 2 and 9 for the down-link. Thus, the SF lies between 4 and 256 chips for the UL. During 10 ms, 38 400 chips can be transmitted so using a shorter SF will provide a higher data transmission rate, while a longer SF means more codes will be orthogonal to one another, which in turn lets more UE transmit or receive data at the same time [44, ch. 3.3]. When a signal is received, the spreading process can just be applied again to recover the original data signal sent. This procedure works because XOR is an injective operator.

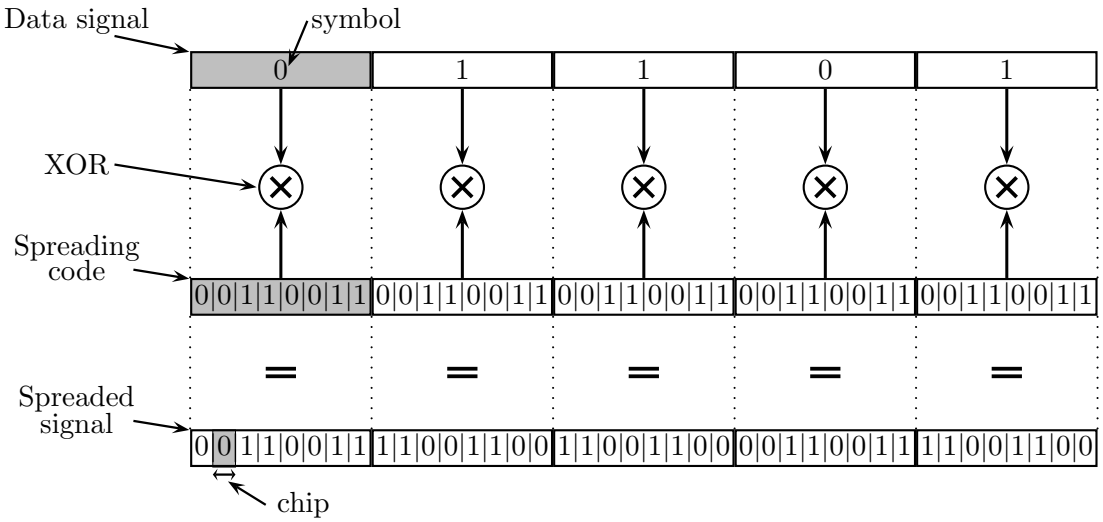


Figure A.1: Illustration of the spreading and encoding of a signal using a SF of 8, figure inspiration taken from [44, ch. 1].

A.3 Layers

A modern communications system is divided into seven layers as illustrated in figure A.2. This follows the Open Systems Interconnection, OSI, reference model x200 which is fostered by the International Organisation for Standardization ISO [45]. The standardisation aims to have a network architecture where products from different manufacturers can cooperate with each other and all kinds of users. BBL1 operates within with layer 1, further described in section A.3.1. The responsibility for each layer is both extensive and irrelevant for the problem solved in this thesis. An overview of the layers gives however also an overview of how information is handled and transported further within the network. Therefore the data handled by each layer and a short description is stated in figure A.2.

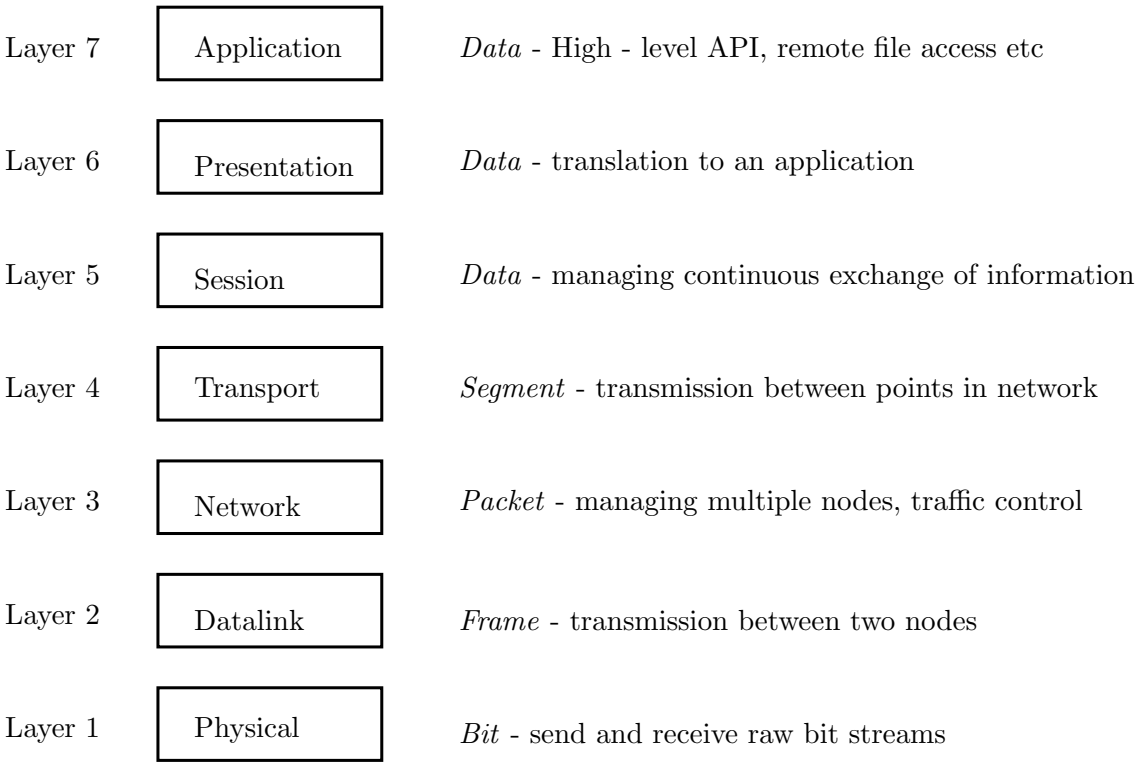


Figure A.2: Illustration of all layers in a network [46].

A.3.1 Physical layer

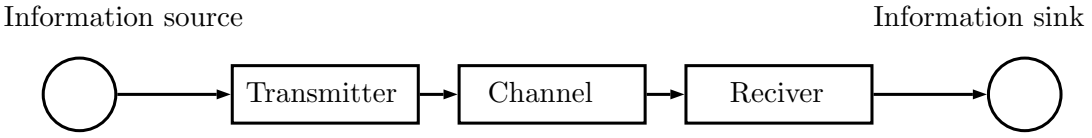


Figure A.3: Illustration of physical layer [47, ch. 1.3].

The physical layer is the communications pipe for information between the source and the destination [47, ch. 1.3]. It has three components:

Transmitter Takes information bearing signal and modifies it to a form suitable for transmission over the channel. Suitable means shaped to pass reliably while efficiently using the radio spectrum and minimising interference.

Channel Is the physical means for transporting the signal to the receiver. When the channel is a propagation path through air, it runs risks of channel distortion (interference

from other channels) and receiver noise / channel noise (caused by electronic devices between receiver and transmitter). The channel is also time-varying in its nature, due to the changes in surroundings and the movement of UE.

Receiver Processes the received signal in order to produce an estimate of the original signal. For this the receiver has to estimate the time-varying nature of the channel so that they can be compensated for and implement error correction to improve the reliability of the wireless channel.

The components are piped according to figure A.3.

A.4 BBL1

A general overview of the BBL1 software is given in section 2.3. The following sections give a more in-depth picture of the work performed by the BBL1.

A.4.1 Channel handling

In section A.3.1 the function of channels are described as part of the components forming the physical layer. There are a number of channels of different kind going to and from the RBS. All of them are handled by the BBL1. All of the channels except High speed packet access, see below, are defined in the 3GPP Release '99 (WCDMA R99) [48] which is the first phase WCDMA standard.

Transport channel

The data generated by the higher layers is carried over the air with transport channels. In the physical layer each transport channel is mapped to one or more physical channels. Depending on the purpose of the physical channel different kinds of signal processing is performed by BBL1 on the data to be transferred on that channel. The signal processing will always involve data to or from IQ signal, section A.4.2, but can also involve coding and spreading, section A.2.4, for common channels. The exact procedure for each physical channel is out of relevance for the thesis, but the processing time for each channel will be analysed. The higher level purpose of the channel can therefore be of interest to the reader.

Over the channel the data are transferred in frames. Every frame takes one Transmission Time Interval (TTI) to transfer. The TTI of WCDMA R99 is 10ms [44, ch.4.3]. Large

data will be transferred in sequential frames, but every frame is decoded separately. When one whole frame is delivered the decoding can begin.

WCDMA R99 has two different kind of transport channels, dedicated and common [49]. There is only one dedicated channel (DCH). DCH carries all the information coming from higher layers that is intended for the dedicated UE. This information contains both data for the actual service and higher layer control information. The information is however only transferred, so the physical layer cannot distinguish between control data and data to be transferred. DCH maps to two different physical channels: Dedicated physical data channel (DPDCH) and Dedicated physical control channel (DPCCH).

There are six different common channels in WCDMA R99. A common channel is a resource divided between all or a group of UE. The division is handled by code spreading as described in section A.2.4. Figure A.4 shows the six common transport channels and the physical channels to which they map. Note that the task of the physical channels is the one of its corresponding transport channel.

There are also a number of physical channels that carry information only relevant for the physical layer. These are:

1. **CPICH** Common pilot channel
2. **SCH** Synchronisation channel
3. **AICH** Acquisition indication channel
4. **PICH** Paging indication channel
5. **CSICH** CPCH status indication channel
6. **CD** Collision detection or **CA-ICH** Channel assignment indicator channel

The channels 1-3 are mandatory for a base station to transmit in order for the system to function. The others are used if CPCH is in use [49, ch. 6.2.3].

High speed packet access

High speed packet access, HSPA, consists of down-link (HSDPA) and up-link (HSUPA). It is an upgrade to the WCDMA standard that increases packet data transfer through layer 1 [49, ch. 11]. BBL1 handles encoding for HSDPA which includes a number of different channels: HS-SCCH, HS-PDSCH, E-HICH, E-AGCH, E-RGCH. It also decodes HSUPA which consists of handling the channels: HS-DPCCH, E-DPCCH, E-DPDCH.

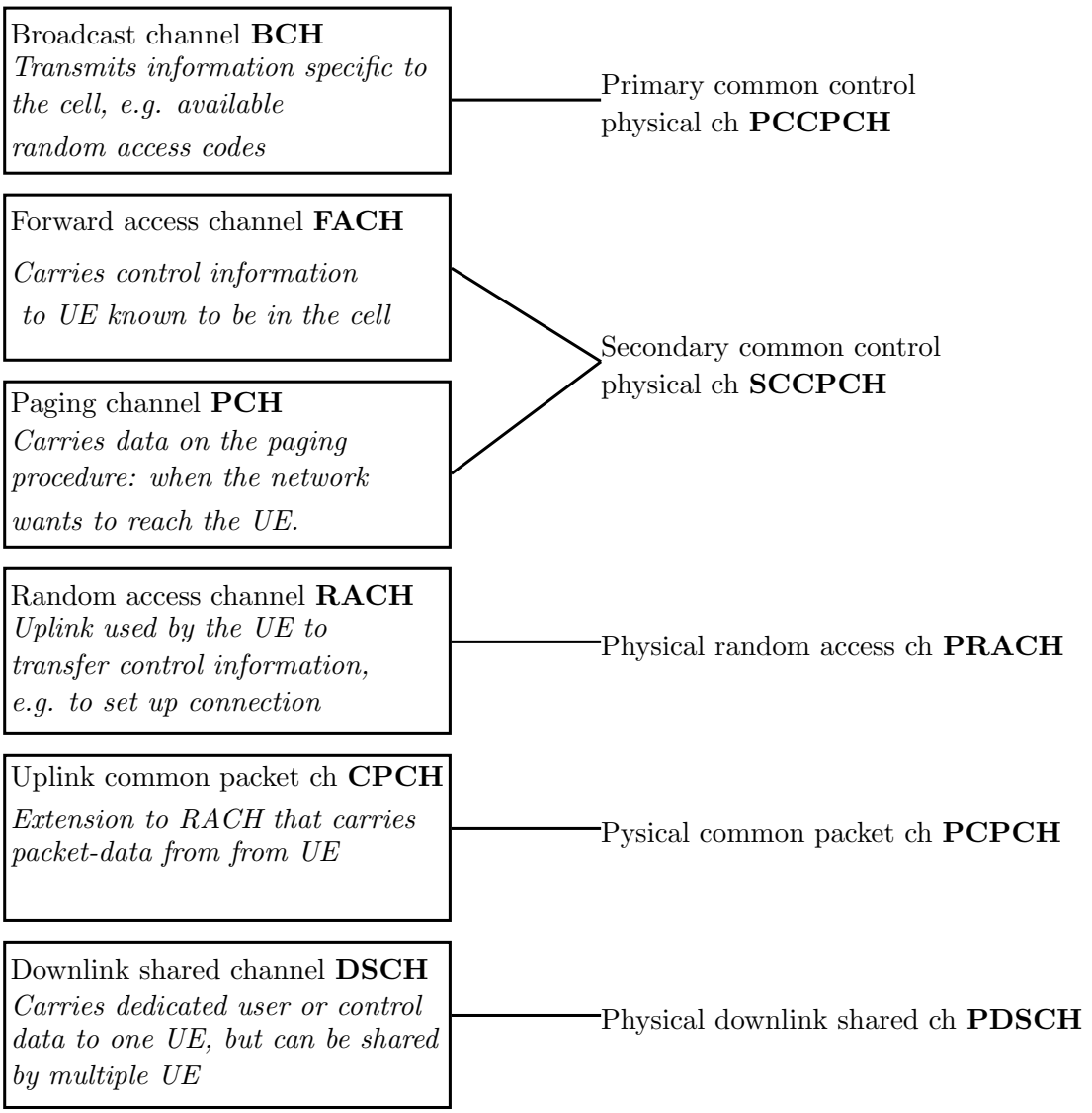


Figure A.4: Illustration of transport channels and their physical channels.

Both UL and DL for HSPA are common channels separated with code spreading as described in section A.2.4. In the HSPA channels however, the TTI is only 2ms, and it is possible to send to one UE on all codes of the channel [50, ch. 4.2.1]. This gives ability to send a lot of data to one user and then dynamically change to an other user. The order in which data is sent is decided by two schedulers, one for the up-link and one for the down-link. HS schedules all data transfer on the down-link, and Enhanced up-link (EUL) on the up-link. HS and EUL are external software located on the same hardware-card as BBL1.

The scheduling is adjusted after which UE that has the best reception at the moment. This increases the total rate of data sent to all UE while decreasing the total amount of bandwidth used for all the transfers. This is possible since the frequencies are allocated only when needed by the UE and not statical throughout the connection as in normal WCDMA R99.

Power control

In wireless systems, the key resource is the radio spectrum [47, ch. 7.6]. All efforts that aim to enable more connections on a smaller spectrum are made. In the cellular industry there is also an other ever present problem, battery time. Aside from building better batteries, efforts are also made in the network to make the battery last longer. One ingredient in network design contribution to the solution of both of these problems is to Transmit Power Control (TPC).

In order to achieve effective communication between the base station and the UE, the received power must be sufficiently above the background noise [51, ch. 10]. This sets the minimum level on the received signal to interference ratio (SIR_{est}). If the UE had unlimited battery it could constantly transmit on its highest power and be heard load and clear by the base station. However, if the UE did that, an other UE in the system might be overshadowed. Due to the power propagation loss in air, if two UE transfer with the same power but on different lengths from the base station, the furthest away will always be overshadowed by the closer one. This means that to keep the system running with many UE, the base station has to control the power from each of them to make sure that they do not overshadow each other. By keeping the UE on the lowest acceptable power the base station also makes sure that the UE use as little of their battery power as possible.

The communication regarding power control is performed on the DCH of the UE. The base station has a pre-set target signal to interference ratio, (SIR_{target}) [44, ch. 2.3.5]. Every time the base station receives a transmission from a UE it estimates the SIR_{est}. If $SIR_{est} > SIR_{target}$ the base station sends '0' on the DCH, telling the UE to lower its power. If $SIR_{est} < SIR_{target}$ a '1' is sent instead. This is called a up-link DCH closed loop or UL-TPC. There is also a down-link DCH closed loop, or DL-TPC, which works in exactly the same way but with opposite roles, i.e. the UE is sending instruction bits to the base station in order for its signals to the UE to be optimal.

A.4.2 Signal processing

As mentioned in section A.3 BBL1 handles all the tasks of the physical layer. This includes communication to UE or RNC via air. In order to send data through air it is encoded into a signal in the ultra high frequency band, the one used for radio communication as described in section 2.1.

After the signal is created it is also spread according to section A.2.4 since BBL1 sends 3G signals. In order to send more information faster the signal sent is a combination of two signals. This is called and IQ signal [8]. The details on how this works is described in section A.4.2. The signal processing described here is carried out on hardware specially built for signal processing purposes, see section 2.3.1.

IQ signals

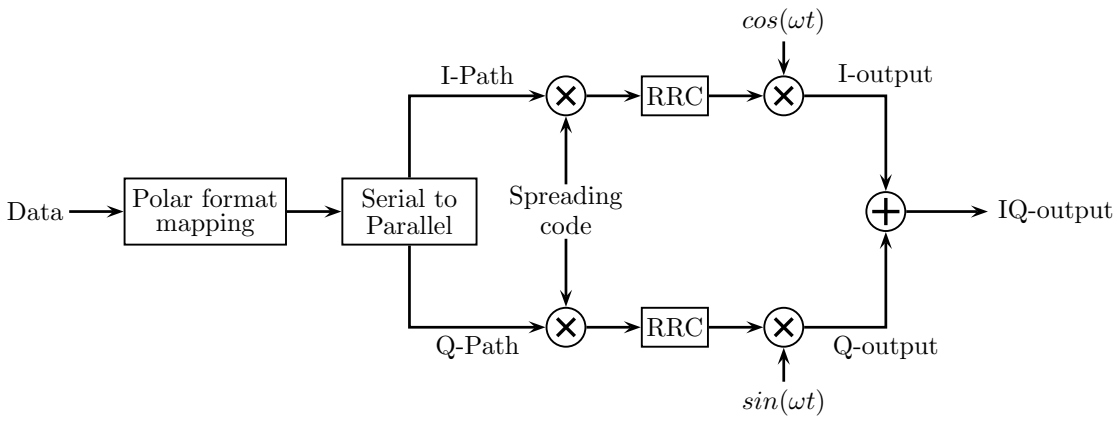
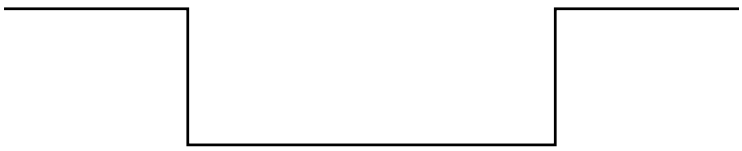


Figure A.5: Illustration of IQ signal processing. [44]

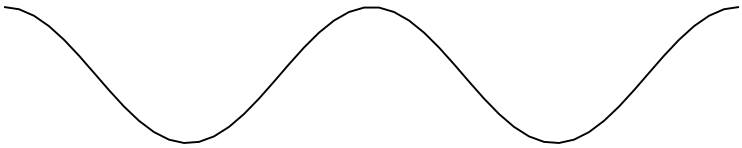
The data to be sent consists of binary information, i.e. a sequence of '0's and '1's. In figure A.5 it shows that the first process on the data is to map it into polar form [44]. This is done by mapping '0' to a $+1$ polar signal and '1' to a -1 step-function signal, in accordance with the WCDMA R99. See example in figure A.6. After that the signal is divided it into an I part and an Q part. This is done by an serial to parallel (S/P) converter that basically sends every other bit on the I-path and the other on Q-path. After the separation, each signal is encoded with the UE spreading code according to section A.2.4. This creates a signal that is longer than the original, since each step in the old signal now is represented by a number of chips.

The I and Q signals are passed through a root-raised-cosine filter (RRC) in order to remove high frequency variations. The two signals are, however, still step-signals. By

Step function (I-Path, figure A.5):



Cosine wave:



Composite wave (I-output, figure A.5):

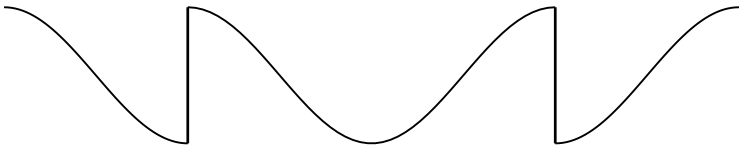


Figure A.6: Illustration of IQ signal processing. [44]

multiplying them whit trigonometric functions sine and cosine respectively, see figure A.6, the signals take the common smooth continuous shape. The I signal will now have a phase shift of 180°in the interval where the step-signal was '−1'. Since the Q signal is multiplied with a sine signal it will have a phase-shift of 90°in the interval where the step-signal was '−1', and a phase-shift of −90°where the step-signal was '+1'.

The last procedure is to add signal I and Q. That results in a signal that can have 4 different phase-shifts, 45°, 135°, −45°and −135°. These values therefore correspond to the step-signals of I and Q according to table A.1. Because of this, when I and Q are combined their sum is a signal containing all the original data.

The IQ signal is then sent through the air to the receiving part, for example an UE. In the UE the I and Q signals are decoupled by using the connection in table A.1. The UE specific spreading code is used to decode the I and Q signals. Then they are combined by simply zipping them. A mapping from polar signals back to binary bits will result in the original data.

I	Q	Output
+1	+1	45°
−1	+1	135°
−1	−1	−135°
+1	−1	−45°

Table A.1: Value of combined signal from I and Q signals

If there are multiple user data they can still be combined into the same IQ signal. This works by adding a step in figure A.5 between RRC and trigonometric signal where the I or Q paths of all users are summarised. Meaning that for each user the steps are followed until after RRC. Then the results of all Q paths are summarised, and the same for all I paths. The result of that is processed as earlier, outputting an IQ signal whit 2^n possible phase-shifts, where n is the number of users. Note that the amplitude of the combined IQ signal also will wary, due to the signals canceling each other. Using the same procedure as before with its spreading code the UE will be able to retrieve the data encoded for it.

Due to disturbance in the channel, i.e. the air that the signal travels to, the phase-shift of the received signal is not always easy to determine. It can also have changed due to movement of the receiver. To compensate for this the signal processing system includes a number of error calculations, including adding a probability to each received symbol. The details of this procedure lies however not within the scope of this thesis.

A.4.3 Latencies

DL TPC *(DPAD)* is counted from when the complete DL-TPCTPC field has been received until the same DL-TPC command is written to the hardware register.

DL DCH CODING *(LAT event)* counts the total time for coding user data to the UE.

E-HICH latency 10ms and **E-HICH latency 2ms** *(LAT event)* is counted from when the last bit of the transport channel block has been received until the same information is written to the command queue.

Iub FP, DCH UL Data Frame *(DPAD)* is counted from when the last bit of the transport channel TTI has been received until the first bit of the same TTI has been sent with I_{ub}.

Iub FP, EDCH UL Data Frame 10ms and **Iub FP, EDCH UL Data Frame 2ms** *(DPAD)* is counted from when the last bit of the transport channel block has

been received until the first bit of the same transport block has been sent with I_{ub}.

PRACH L1 ACK *(LAT event)* is counted from when the complete preamble has been received until the information is written to the command queue.

UL TPC *(LAT event)* is counted from when the complete UL-TPC field has been received until the same UL-TPC command is written to the hardware register.

WIDCI HS-CQI *(DPAD)* is counted from when the last bit of the HS-DPCCH CQI field has been received until the first bit of the same message has been sent.

WIDCI HS-HARQ *(DPAD)* is counted from when the last bit of the HS-DPCCH HARQ field has been received until the first bit of the same message has been sent.

EDCH UE STATUS IND *(DPAD)* is counted from when the last bit of the UE Rate Request has been received until the first bit of the same rate request has been sent.

B

Raw Data

Raw data, that was gathered from Ericsson's database is presented here. Each colour represents a different test case and the broader black lines represent our hand-crafted estimate.

B.1 DSP load and memory usage

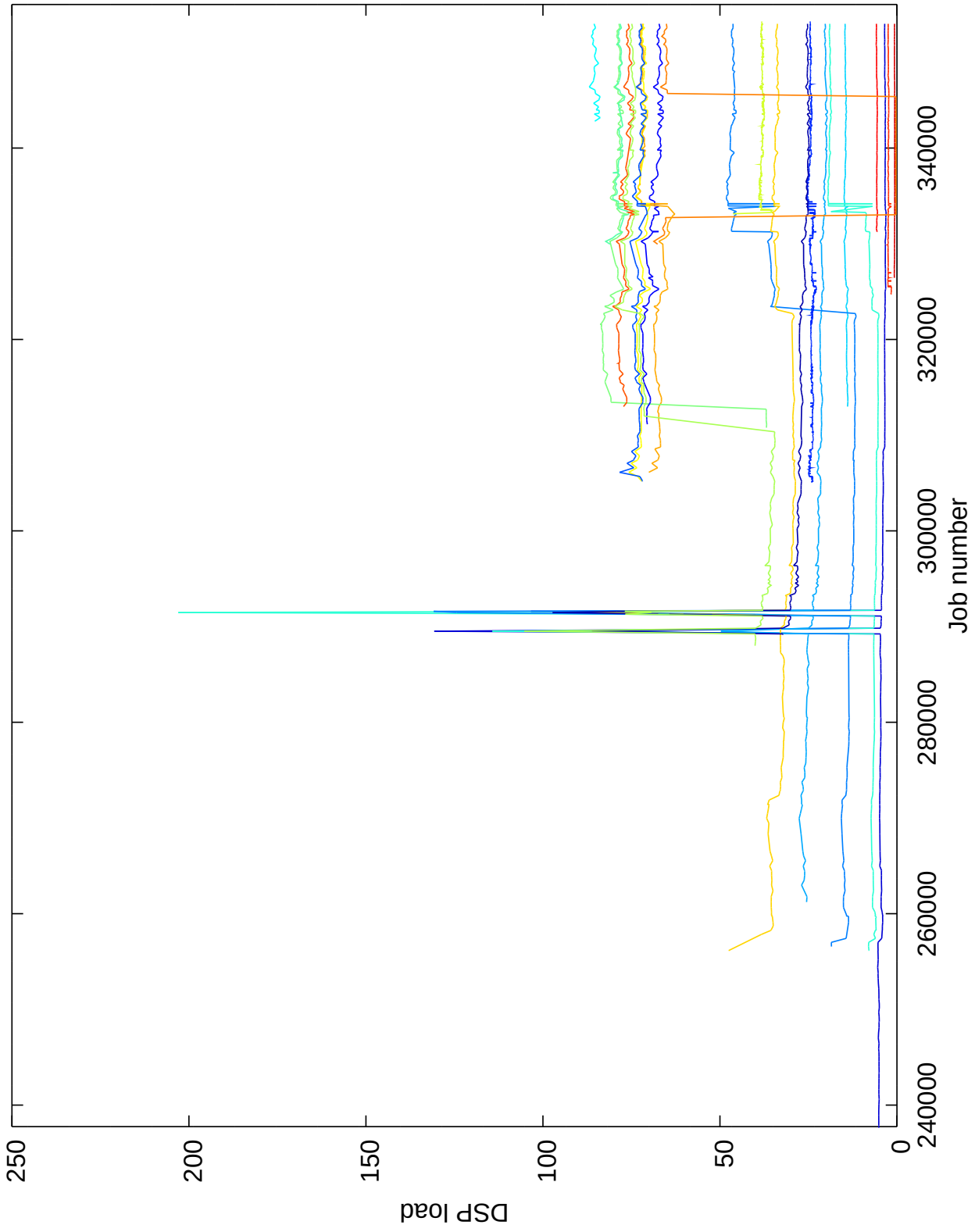


Figure B.1: Plotted data measuring DSP load over time.

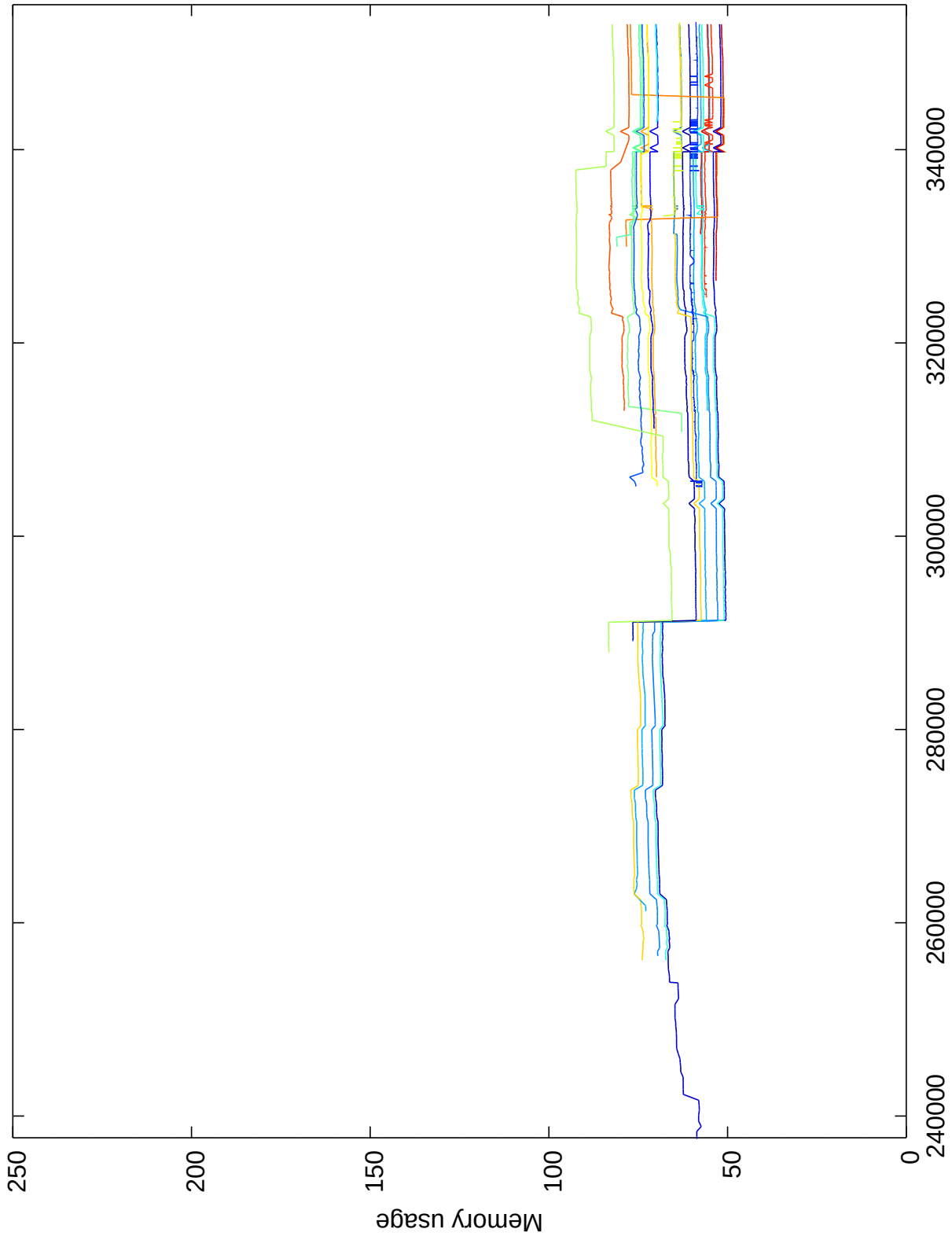


Figure B.2: Plotted data measuring memory load over time.

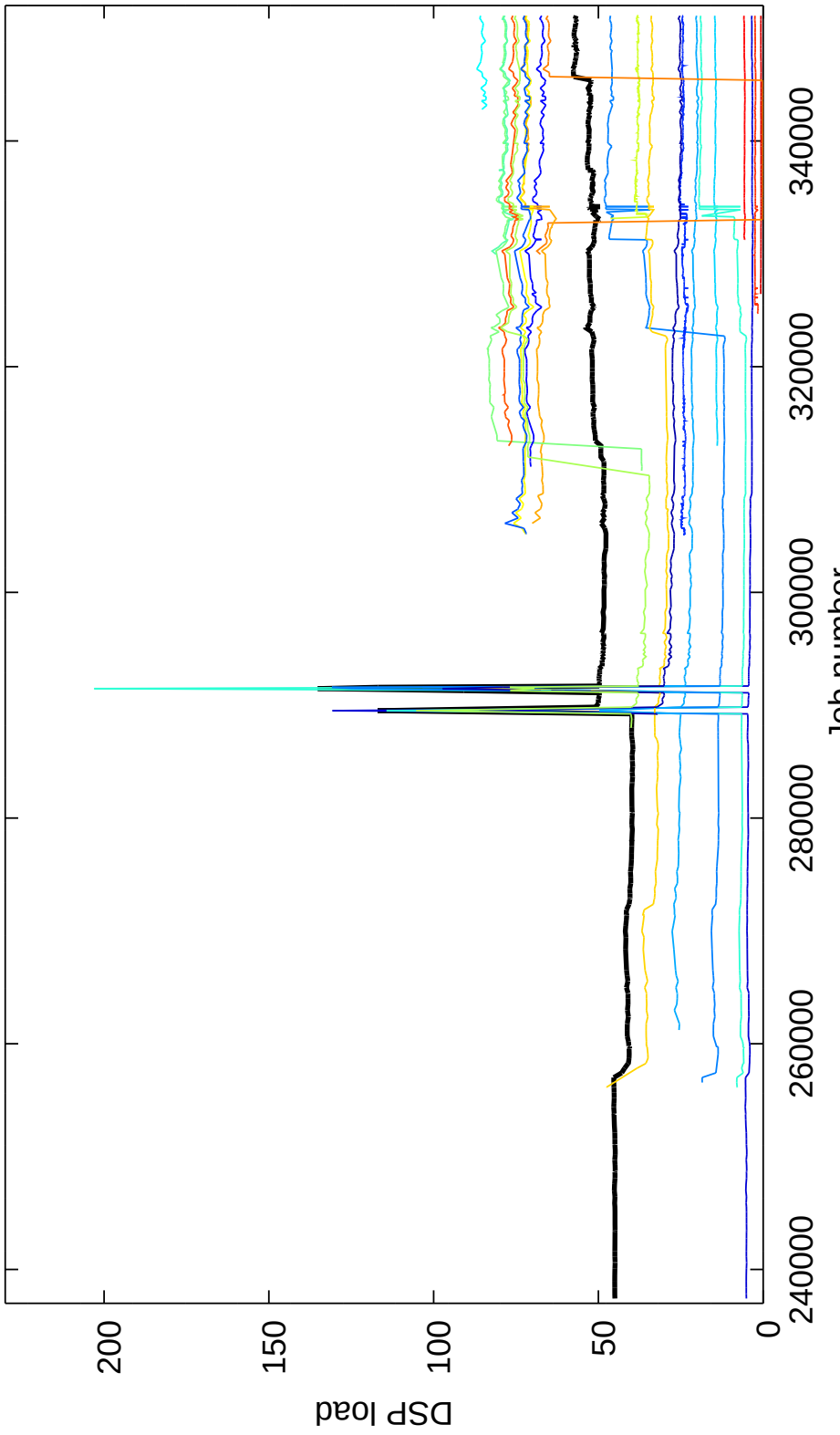


Figure B.3: Plotted data measuring DSP load over time with calculated mean.

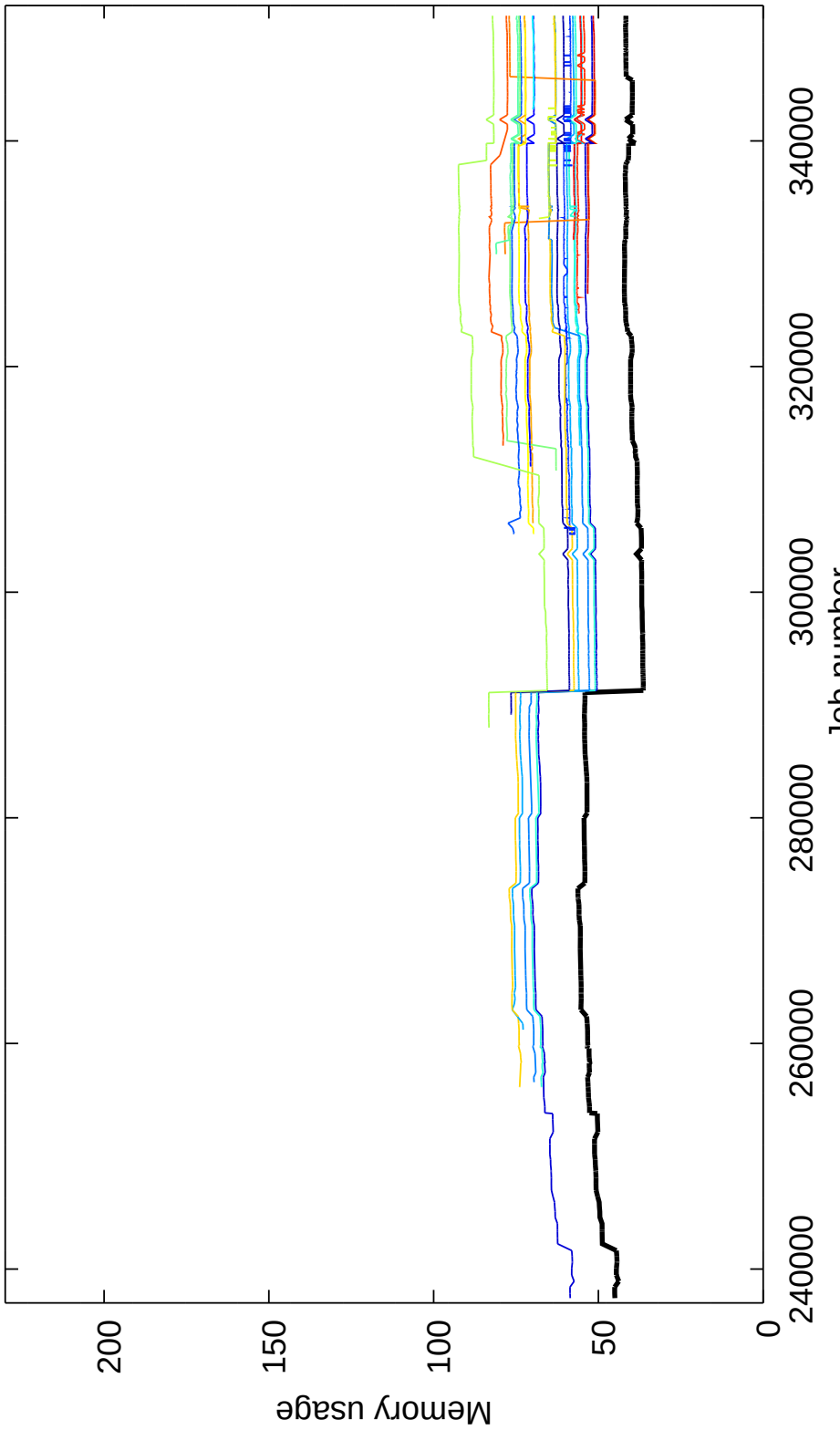


Figure B.4: Plotted data measuring memory load over time with calculated mean.

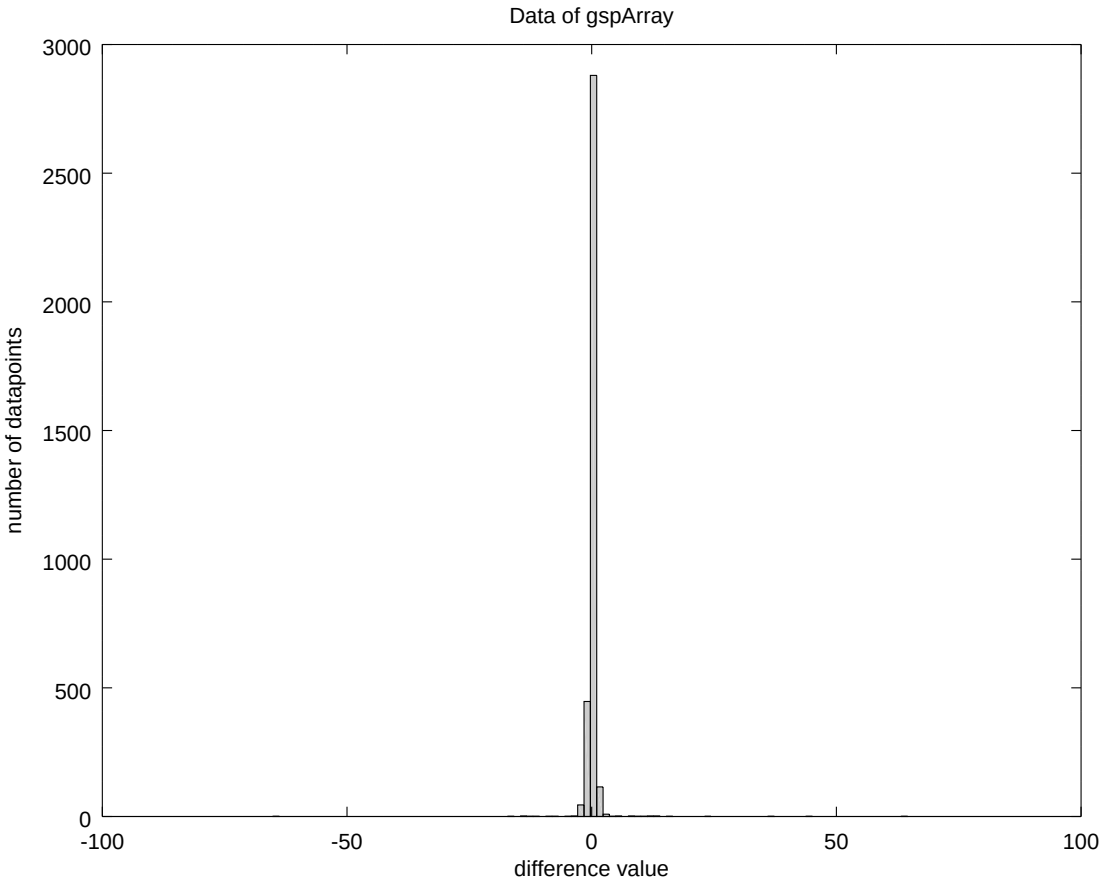


Figure B.5: Histogram over all difference values for DSP load data.

B.2 Latencies

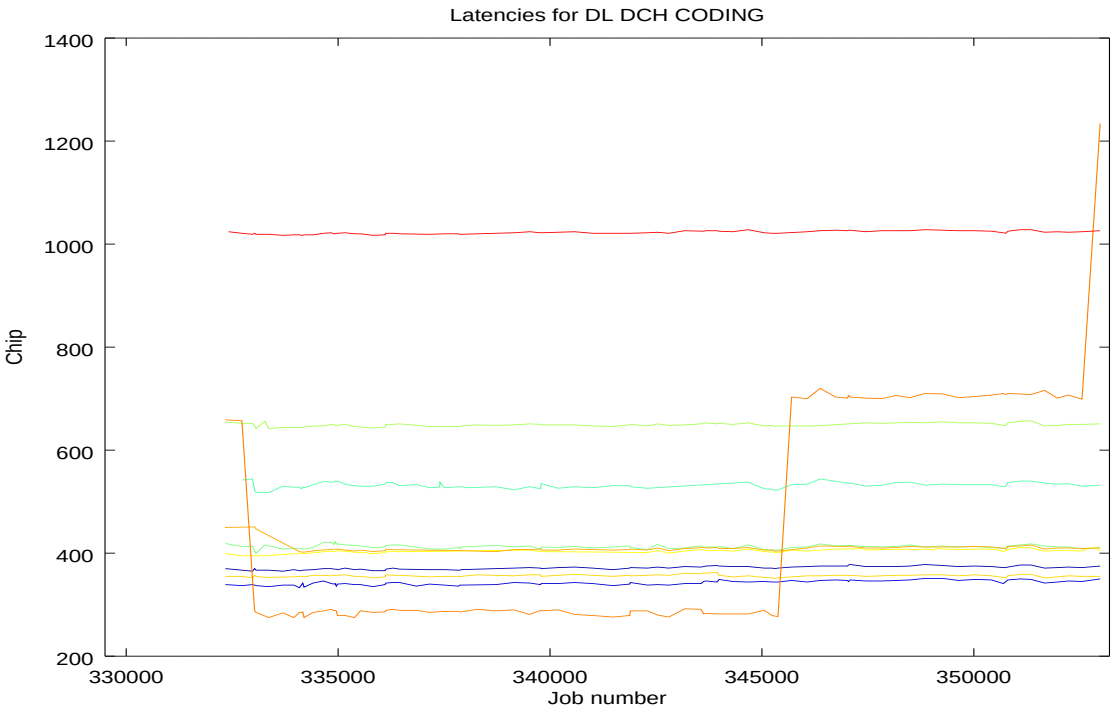


Figure B.6: Raw data for one latency from all test cases

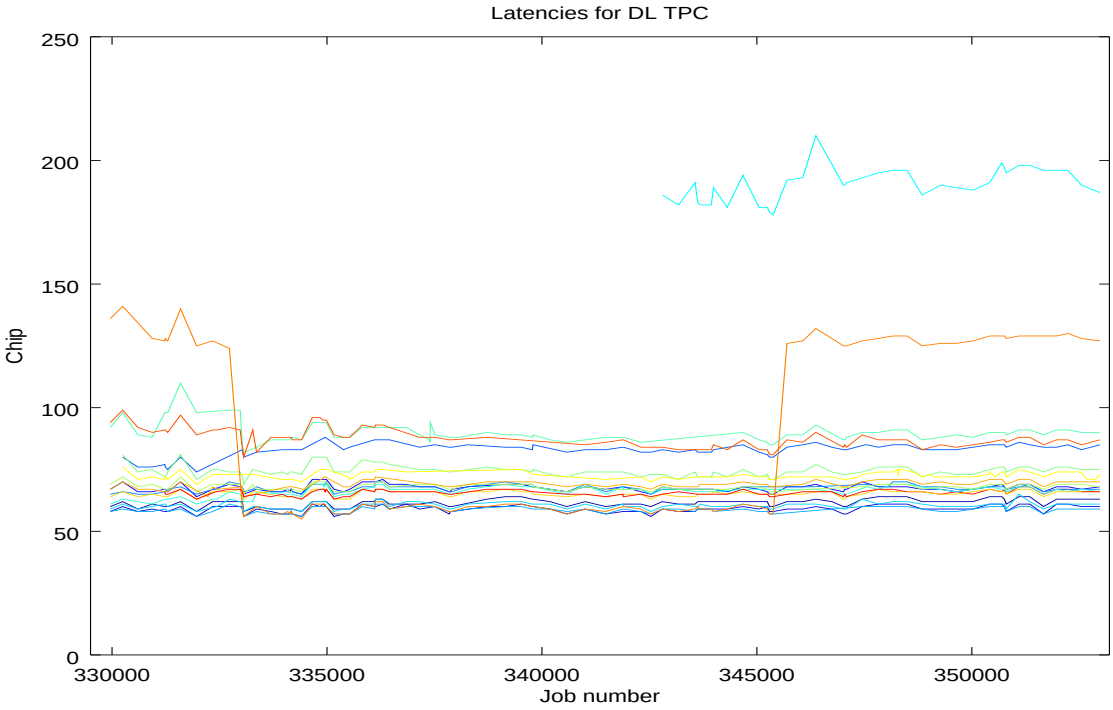


Figure B.7: Raw data for one latency from all test cases

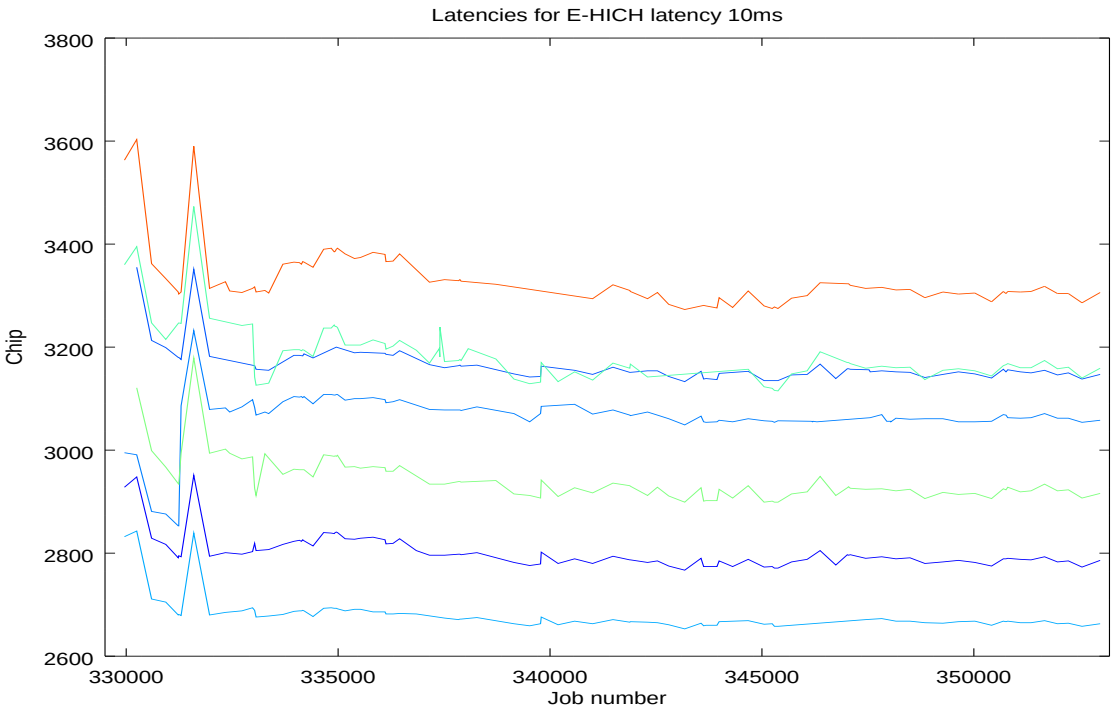


Figure B.8: Raw data for one latency from all test cases

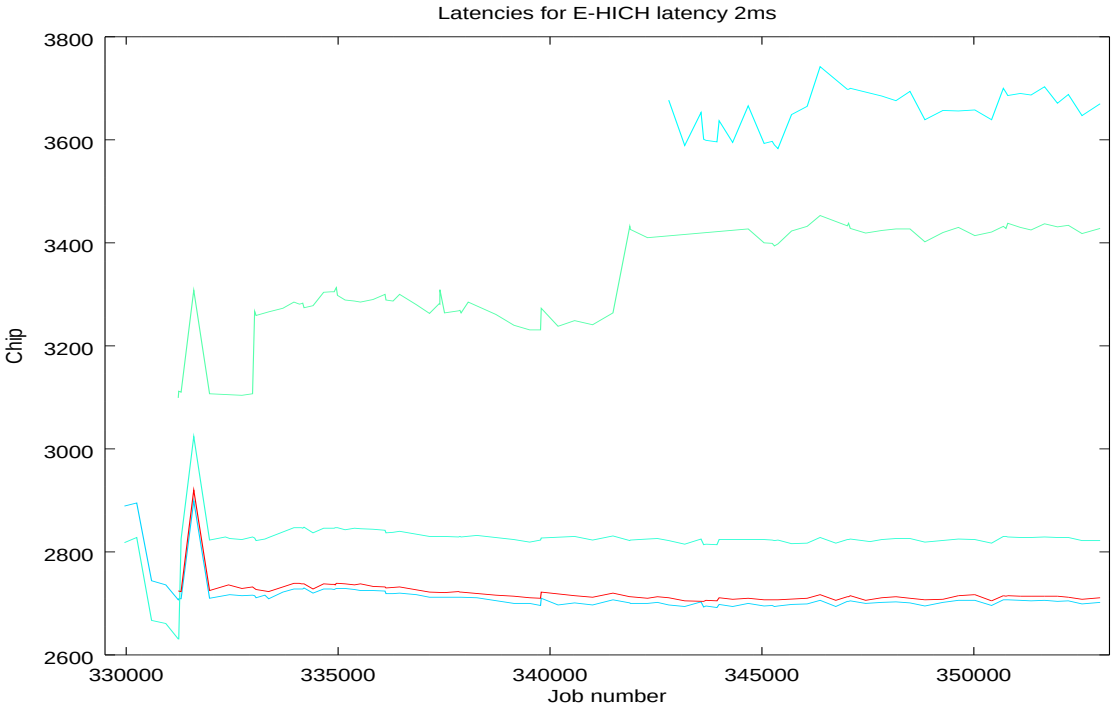


Figure B.9: Raw data for one latency from all test cases

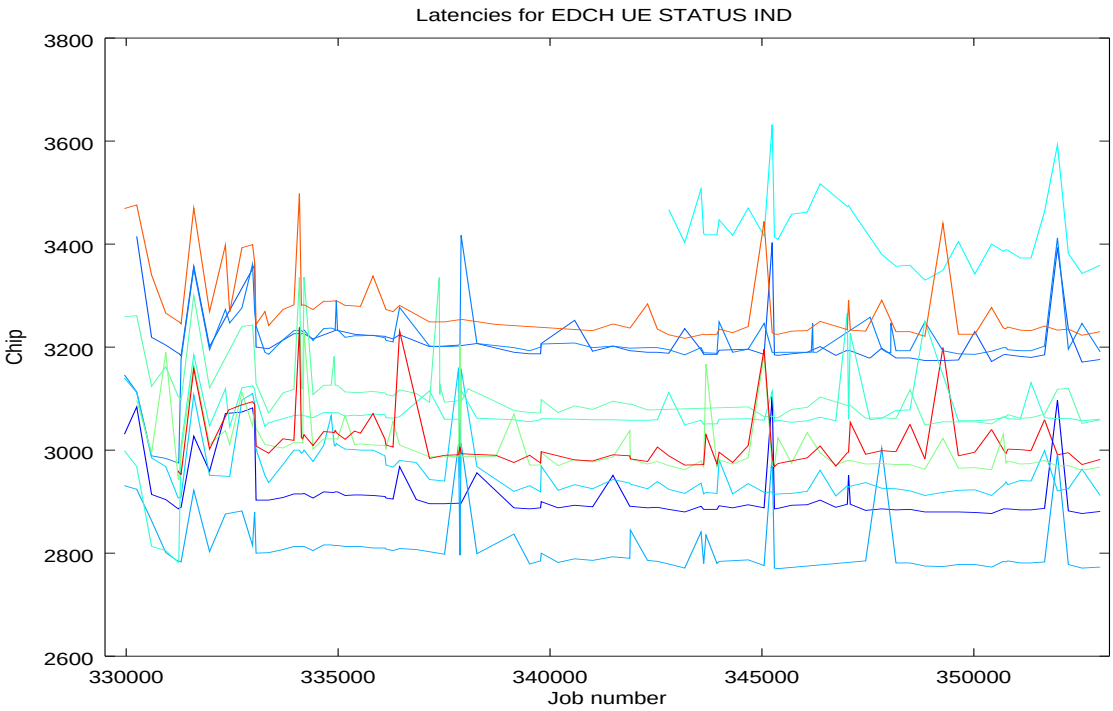


Figure B.10: Raw data for one latency from all test cases

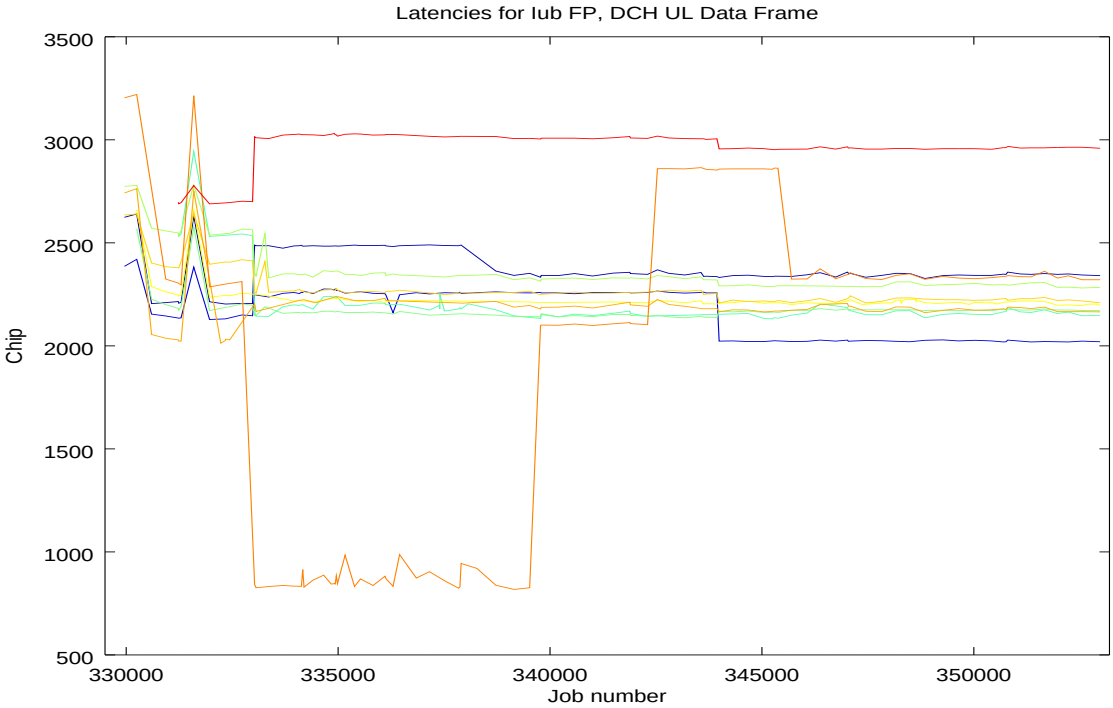


Figure B.11: Raw data for one latency from all test cases

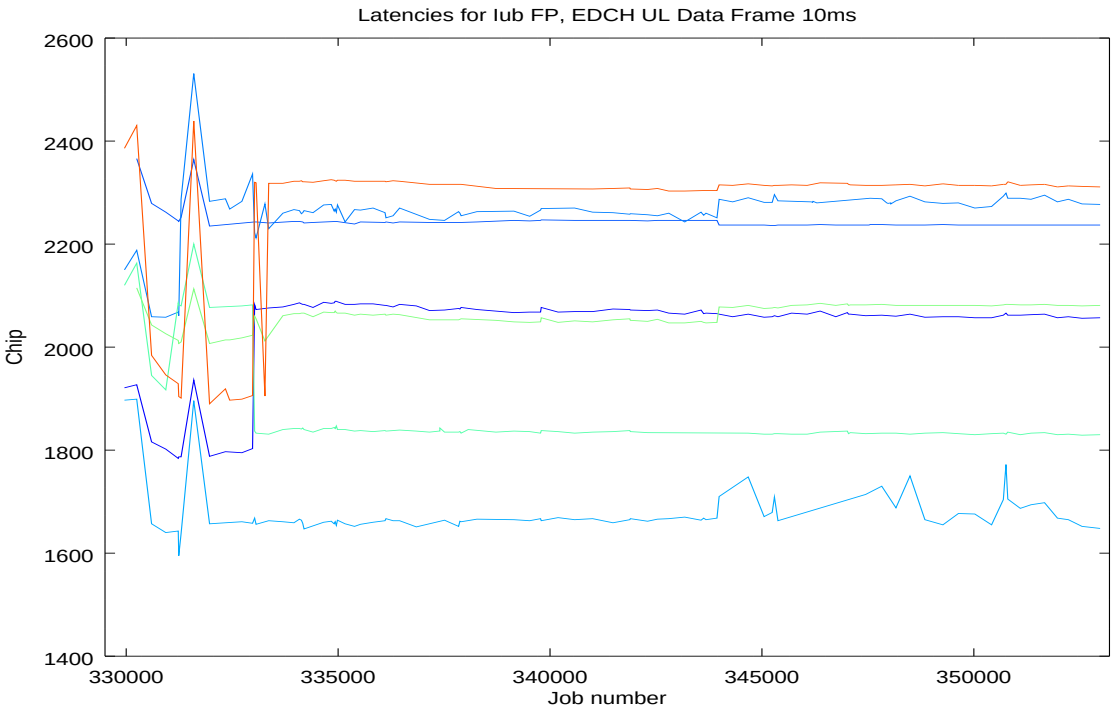


Figure B.12: Raw data for one latency from all test cases

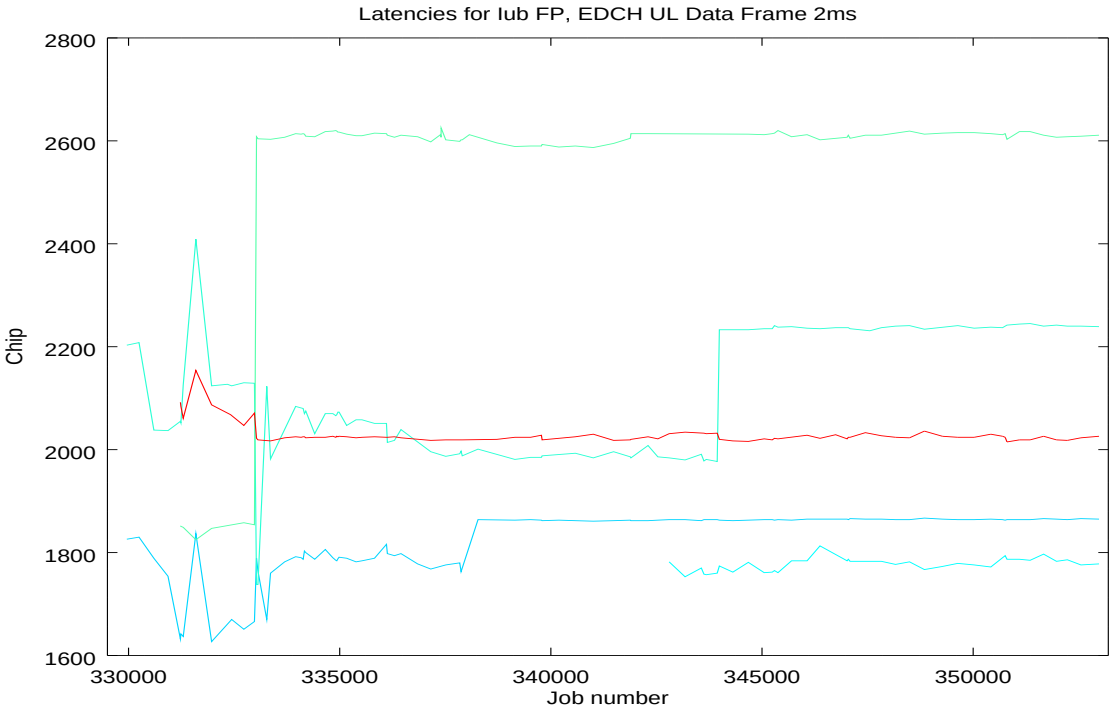


Figure B.13: Raw data for one latency from all test cases

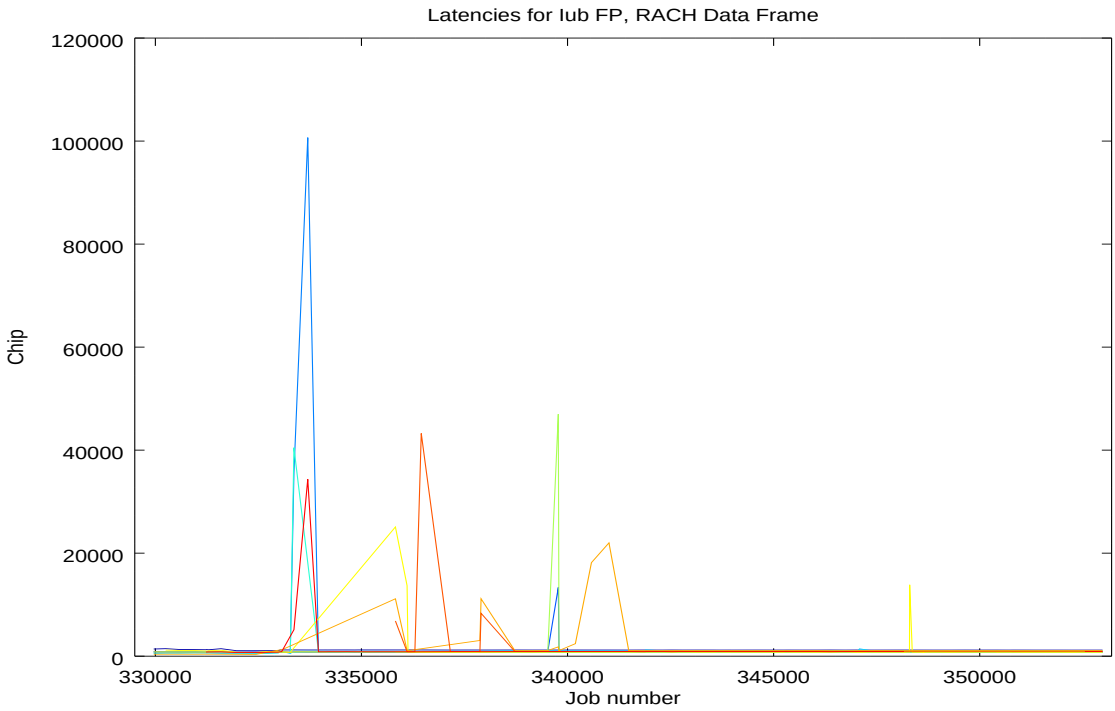


Figure B.14: Raw data for one latency from all test cases

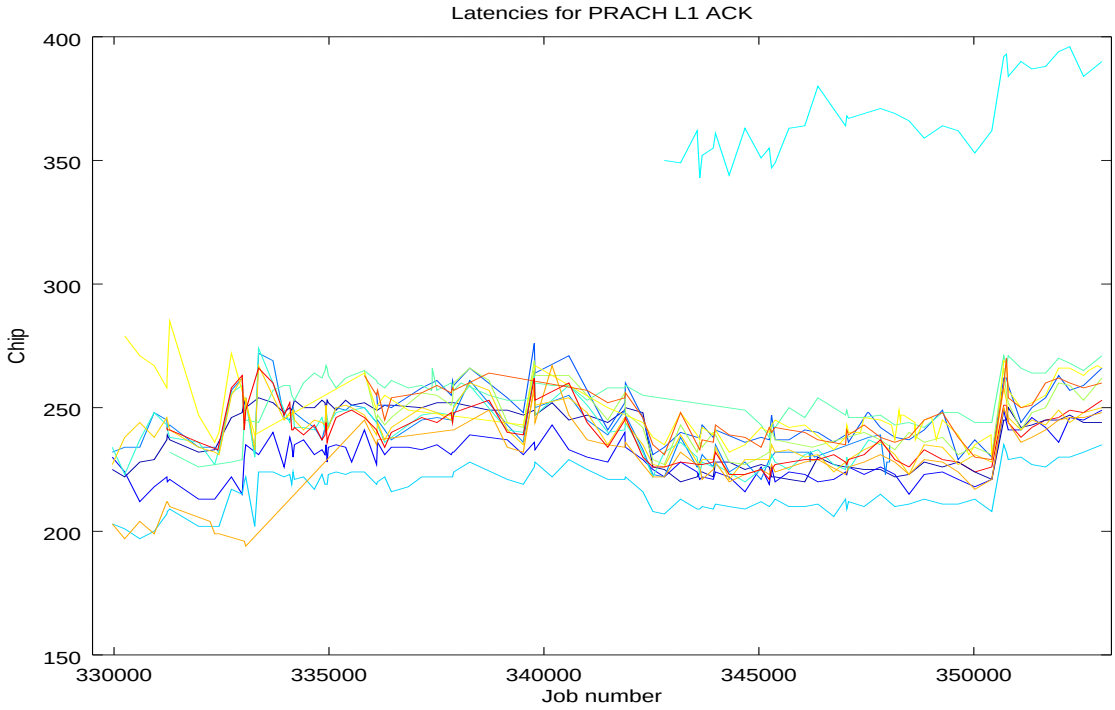


Figure B.15: Raw data for one latency from all test cases

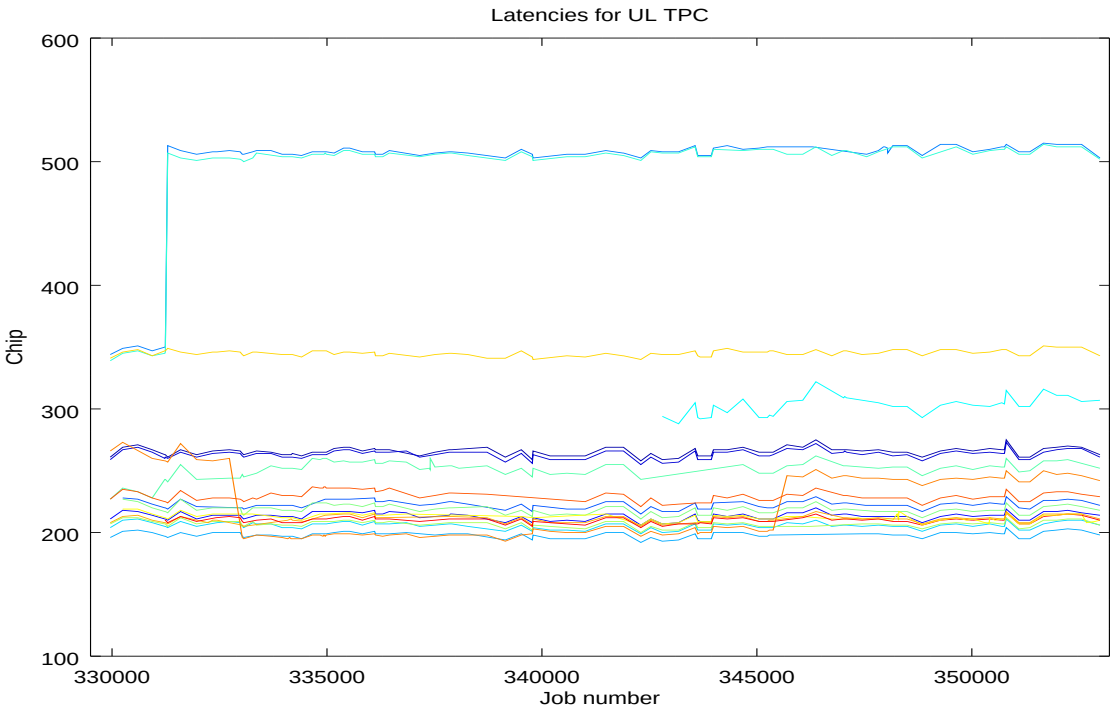


Figure B.16: Raw data for one latency from all test cases

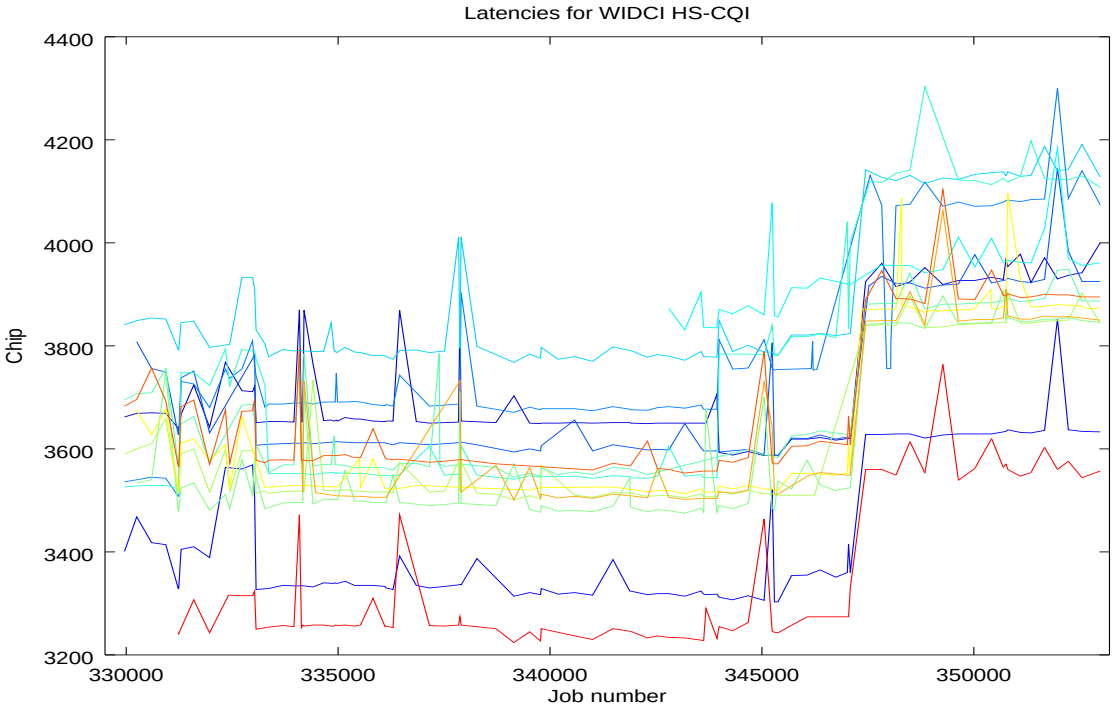


Figure B.17: Raw data for one latency from all test cases

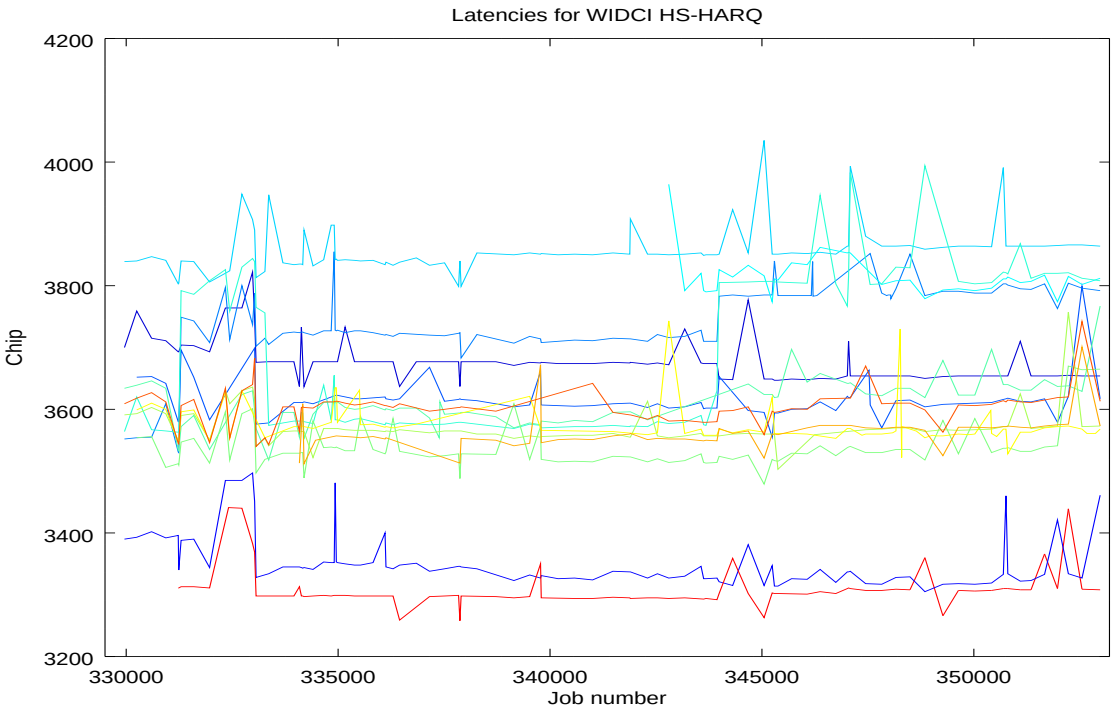


Figure B.18: Raw data for one latency from all test cases

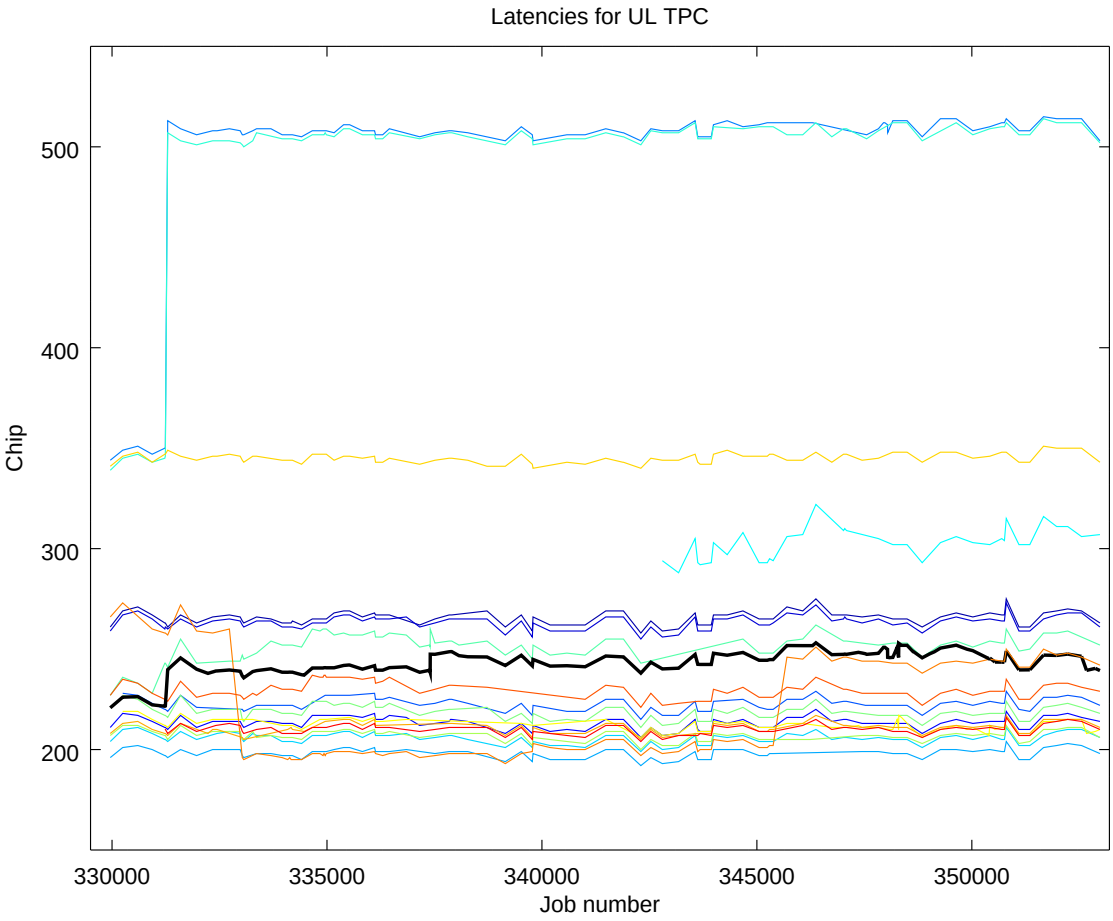


Figure B.19: Data on one latency for different test cases. The estimated mean is shown as a black line.

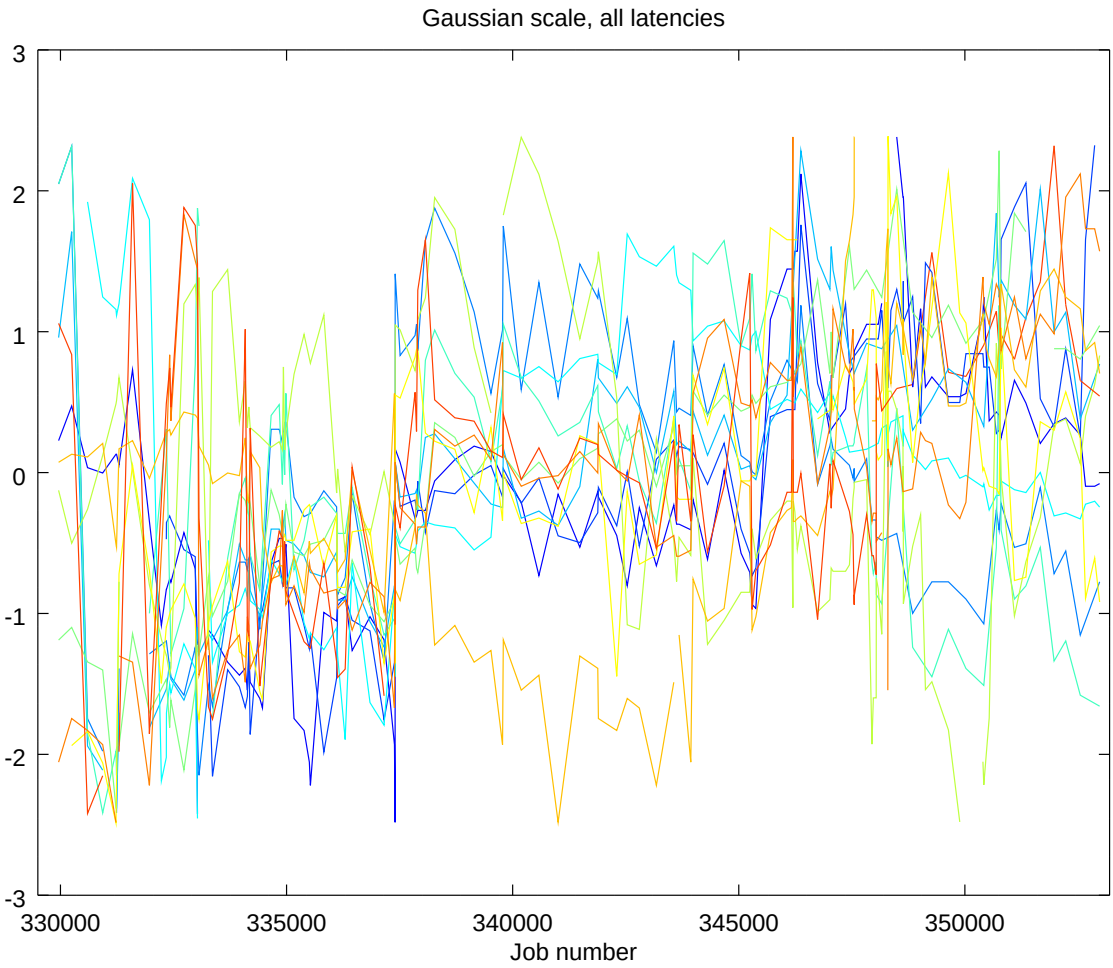


Figure B.20: Latency data with Gaussian scale.

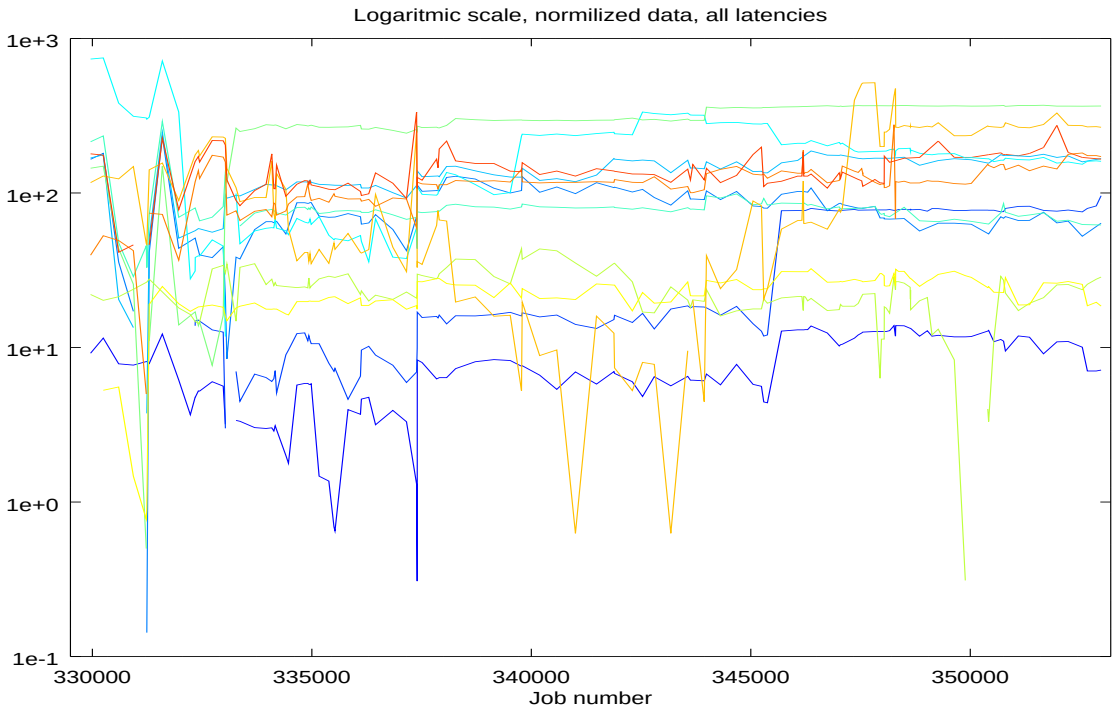


Figure B.21: Normalised latency data with logarithmic scale.

C

First Clustering

DATA GATHERED for making the first clustering in which outliers that should not be included in our final clustering are removed. Here all results of the one-dimensional clustering can be found.

C.1 Clustering per test

test number	h	c	$v_{\beta=1}$	$v_{\beta=0.6}$	ARI
1	1.000	0.046	0.088	0.114	0.038
2	1.000	0.033	0.063	0.083	0.028
3	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000
5	1.000	0.077	0.143	0.182	0.098
6	1.000	0.030	0.058	0.076	0.018
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	1.000	0.034	0.065	0.085	0.027
10	0.000	0.000	0.000	0.000	0.000
11	1.000	0.085	0.157	0.199	0.103
12	0.892	0.213	0.344	0.406	0.319
13	0.000	0.000	0.000	0.000	0.000
14	1.000	0.045	0.086	0.111	0.043
15	0.000	0.000	0.000	0.000	0.000
16	1.000	0.327	0.493	0.564	0.418
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000
19	0.000	0.000	0.000	0.000	0.000

Table C.1: Clustering per test for one dimensional difference data of memory

test number	h	c	$v_{\beta=1}$	$v_{\beta=0.6}$	ARI
1	0.000	0.000	0.000	0.000	0.000
2	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000
5	1.000	0.192	0.322	0.388	0.206
6	1.000	0.051	0.098	0.126	0.029
7	0.000	0.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000
9	0.921	0.162	0.276	0.334	0.194
10	0.000	0.000	0.000	0.000	0.000
11	1.000	0.037	0.072	0.094	0.021
12	1.000	0.028	0.055	0.072	0.016
13	0.000	0.000	0.000	0.000	0.000
14	0.000	0.000	0.000	0.000	0.000
15	0.000	0.000	0.000	0.000	0.000
16	1.000	0.237	0.383	0.453	0.273
17	0.000	0.000	0.000	0.000	0.000
18	0.000	0.000	0.000	0.000	0.000
19	0.000	0.000	0.000	0.000	0.000

Table C.2: Clustering per test for one dimensional difference data of DSP

C.2 Clustered data

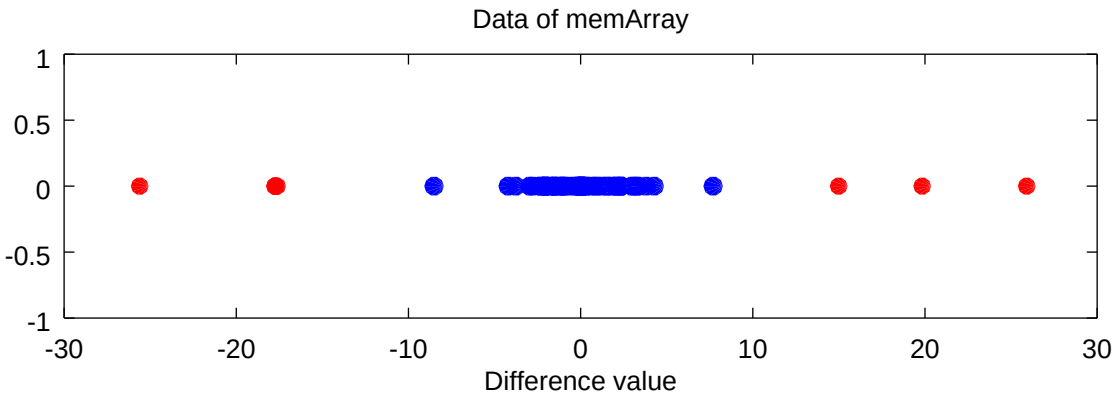


Figure C.1: Scatterplot for the difference values of memory usage clustered into 3 clusters with k-medoids using Mahalanobis distance. Points considered valid are blue and faulty measures are red.

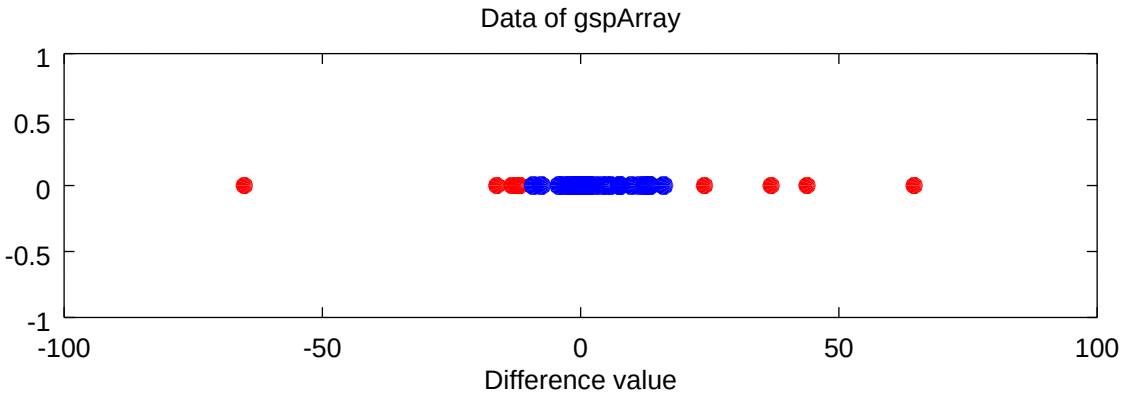


Figure C.2: Scatterplot for the difference values of DSP load clustered into 3 clusters with k-means using city block distance. Points considered valid are blue and faulty measures are red.

D

Second Clustering

DATA GATHERED after the initial clustering and the extreme outliers has been removed. Tables for clustering of this data with unweighed distances are shown first. After that the clustering with weighted distances are shown as plots of the clustering with colours representing the different clusters. Lastly the resulting marking of points and jobs are presented.

D.1 Unweighed distance

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
squared Euclidean	3	0.188	0.041	0.067	0.092	0.033
city block	3	0.190	0.038	0.064	0.089	0.028
squared Euclidean	5	0.480	0.078	0.134	0.194	0.080
city block	5	0.502	0.080	0.138	0.201	0.082
squared Euclidean	7	0.663	0.054	0.100	0.157	0.066
city block	7	0.671	0.052	0.096	0.152	0.062
squared Euclidean	10	0.712	0.047	0.088	0.142	0.052
city block	10	0.674	0.043	0.081	0.129	0.046

Table D.1: Difference and trend values, (d,d_t), of DSP load after outlier removal. Compared to Ericsson gold assignment

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
squared Euclidean	3	0.620	0.147	0.238	0.323	0.306
city block	3	0.642	0.103	0.178	0.258	0.210
squared Euclidean	5	0.688	0.082	0.146	0.220	0.156
city block	5	0.688	0.082	0.146	0.221	0.157
squared Euclidean	7	0.711	0.041	0.077	0.124	0.065
city block	7	0.694	0.033	0.062	0.102	0.044
squared Euclidean	10	0.682	0.026	0.050	0.083	0.034
city block	10	0.699	0.018	0.035	0.060	0.015

Table D.2: Difference and trend values, (d,d_t), of memory load after outlier removal. Compared to Ericsson gold assignment

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
squared Euclidean	3	0.189	0.039	0.065	0.090	0.030
city block	3	0.190	0.038	0.064	0.089	0.028
squared Euclidean	5	0.480	0.078	0.134	0.193	0.079
city block	5	0.671	0.069	0.125	0.193	0.092
squared Euclidean	7	0.690	0.041	0.078	0.126	0.033
city block	7	0.679	0.052	0.097	0.153	0.060
squared Euclidean	10	0.701	0.033	0.064	0.105	0.025
city block	10	0.679	0.033	0.064	0.104	0.026

Table D.3: data: (d,d_t, v_n) DSP load

Table D.4: Difference, trend and normalised values, (d,d_t, v_n), of DSP load after outlier removal. Compared to Ericsson gold assignment

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
squared Euclidean	3	0.620	0.147	0.238	0.323	0.306
city block	3	0.642	0.103	0.178	0.258	0.210
squared Euclidean	5	0.688	0.082	0.146	0.220	0.156
city block	5	0.688	0.082	0.146	0.221	0.157
squared Euclidean	7	0.711	0.040	0.076	0.124	0.064
city block	7	0.694	0.031	0.059	0.096	0.039
squared Euclidean	10	0.682	0.014	0.028	0.048	0.008
city block	10	0.697	0.024	0.046	0.076	0.027

Table D.5: Difference, trend and normalised values, (d,d_t, v_n), of memory load after outlier removal. Compared to Ericsson gold assignment

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
squared Euclidean	3	0.170	0.035	0.057	0.080	0.023
city block	3	0.453	0.662	0.538	0.498	0.679
squared Euclidean	5	0.462	0.087	0.146	0.206	0.099
city block	5	0.645	0.073	0.131	0.200	0.108
squared Euclidean	7	0.684	0.054	0.101	0.159	0.069
city block	7	0.659	0.070	0.127	0.194	0.101
squared Euclidean	10	0.700	0.044	0.083	0.134	0.055
city block	10	0.673	0.043	0.082	0.131	0.050

Table D.6: Relative difference and relative trend values, $(d_r, d_{t,r})$, of DSP load after outlier removal. Compared to Ericsson gold assignment

distance	k	h	c	$v_{\beta=1}$	$v_{\beta=0.4}$	ARI
squared Euclidean	3	0.702	0.221	0.336	0.433	0.421
city block	3	0.645	0.104	0.179	0.260	0.212
squared Euclidean	5	0.688	0.082	0.146	0.220	0.156
city block	5	0.689	0.082	0.147	0.222	0.157
squared Euclidean	7	0.711	0.043	0.081	0.131	0.064
city block	7	0.708	0.036	0.068	0.111	0.051
squared Euclidean	10	0.698	0.019	0.038	0.063	0.017
city block	10	0.682	0.018	0.036	0.060	0.016

Table D.7: Relative difference and relative trend values, $(d_r, d_{t,r})$, of memory load after outlier removal. Compared to Ericsson gold assignment

D.2 Clustered data plots

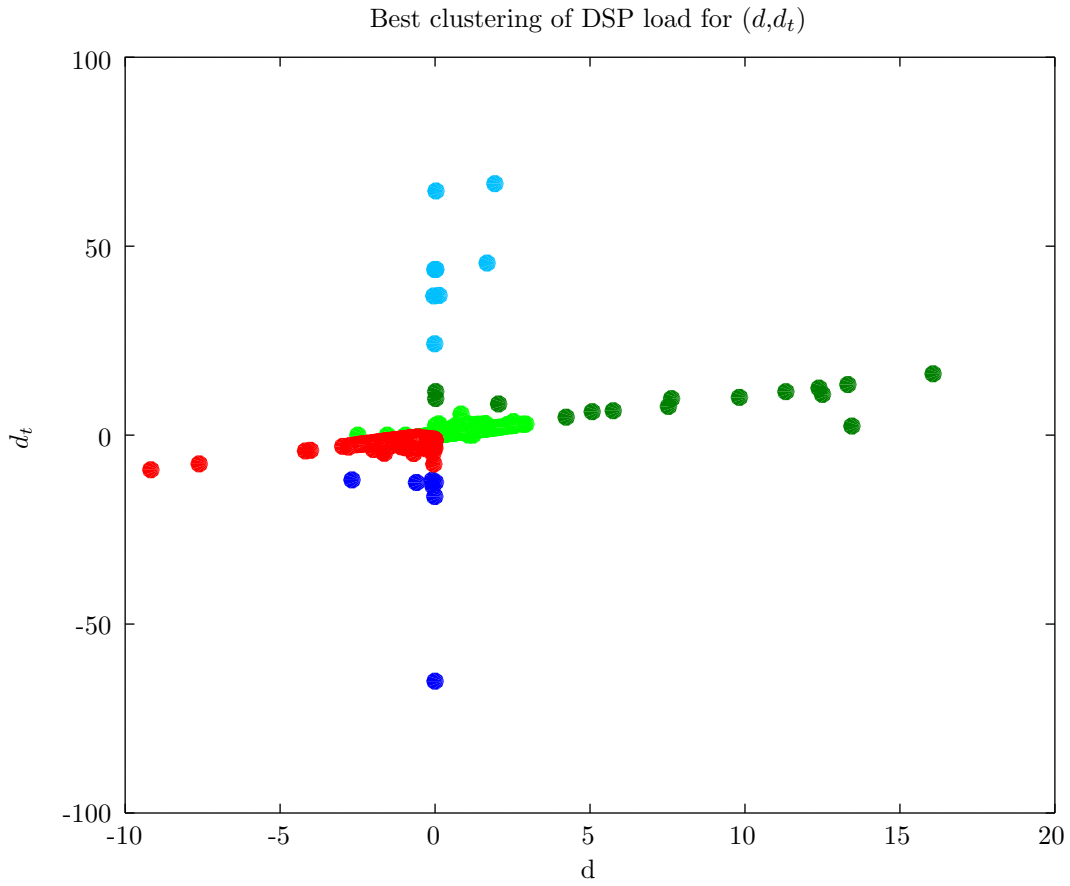


Figure D.1: Clustering with clusters shown as colours

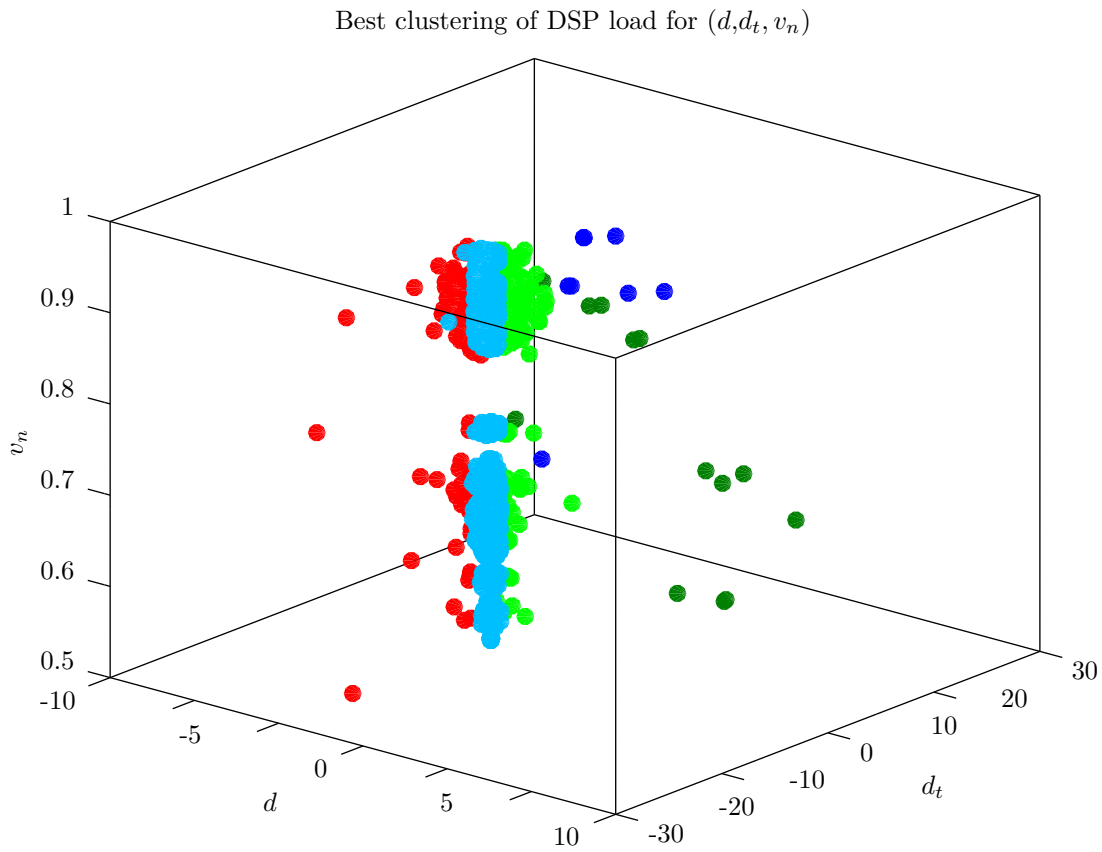


Figure D.2: Clustering with clusters shown as colours

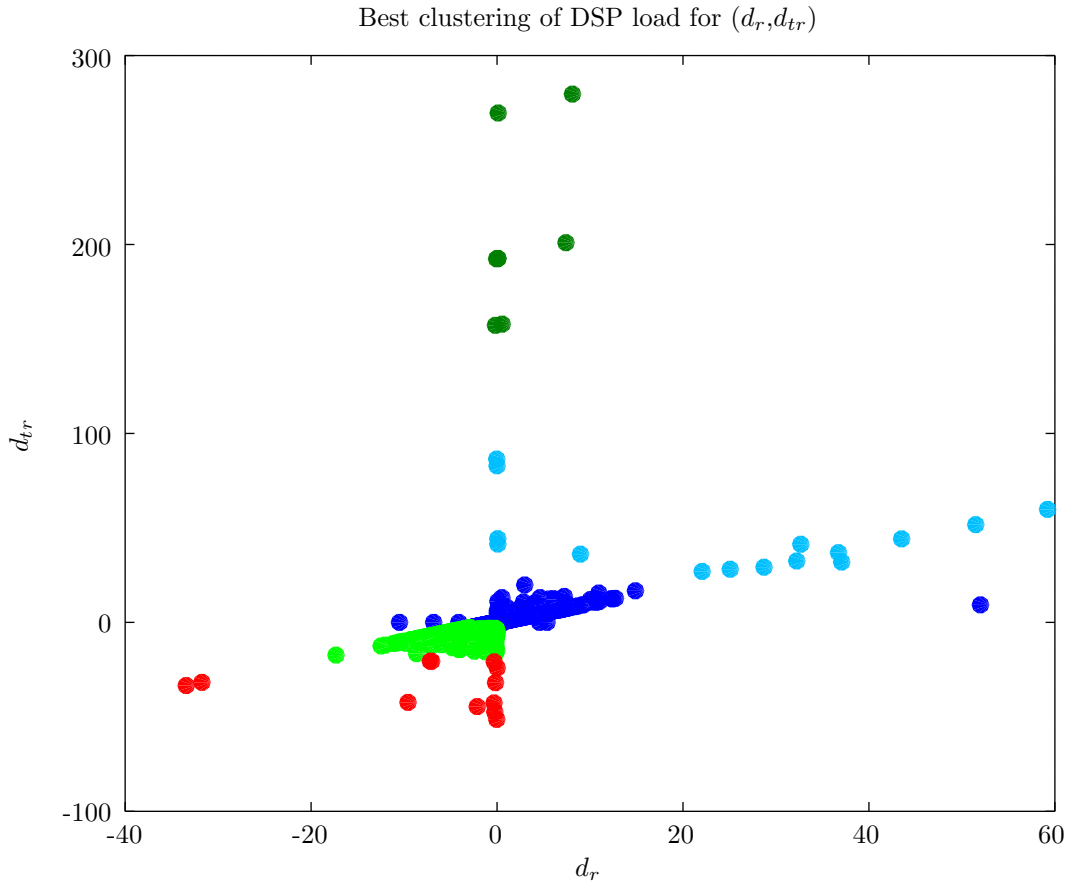


Figure D.3: Clustering with clusters shown as colours

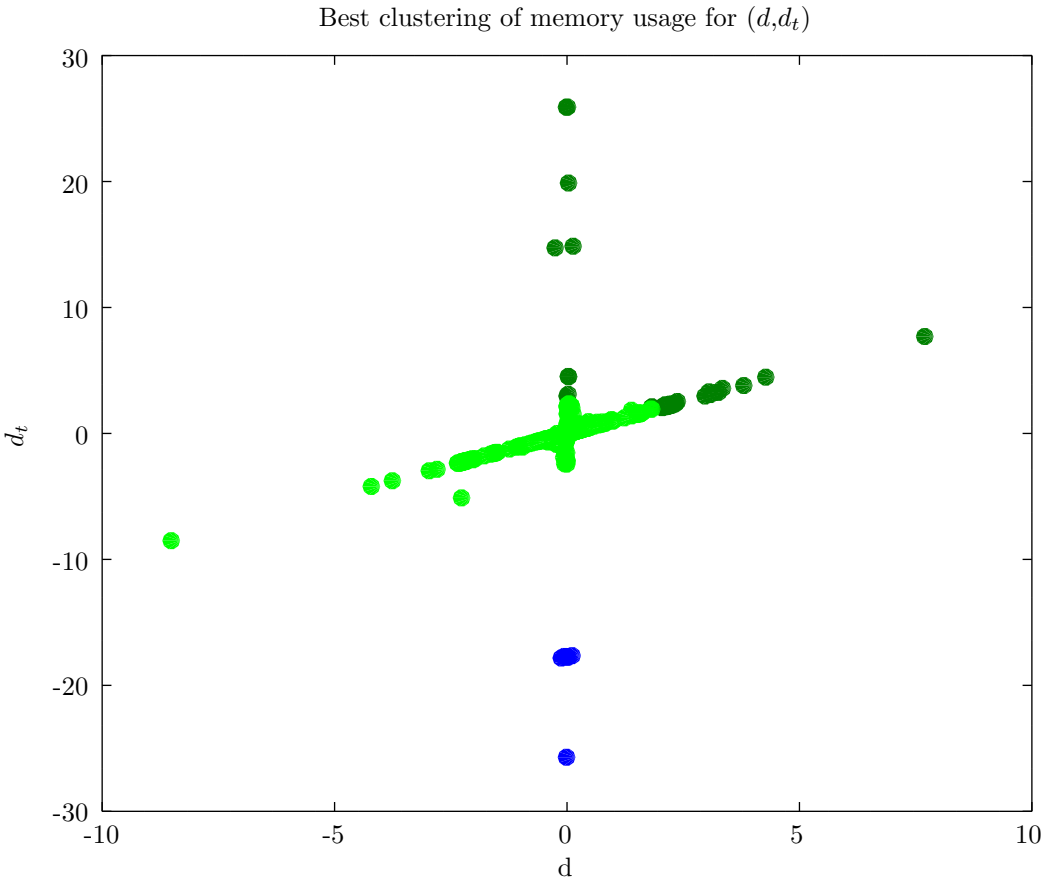


Figure D.4: Clustering with clusters shown as colours

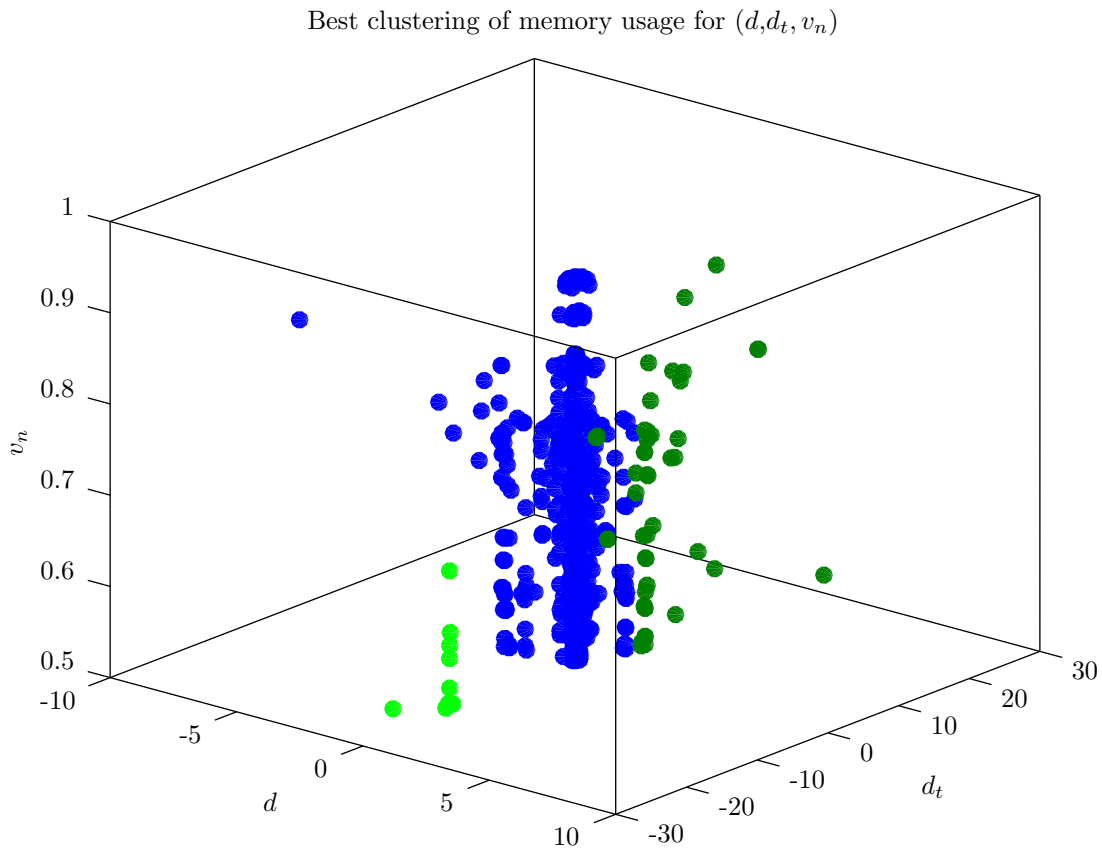


Figure D.5: Clustering with clusters shown as colours

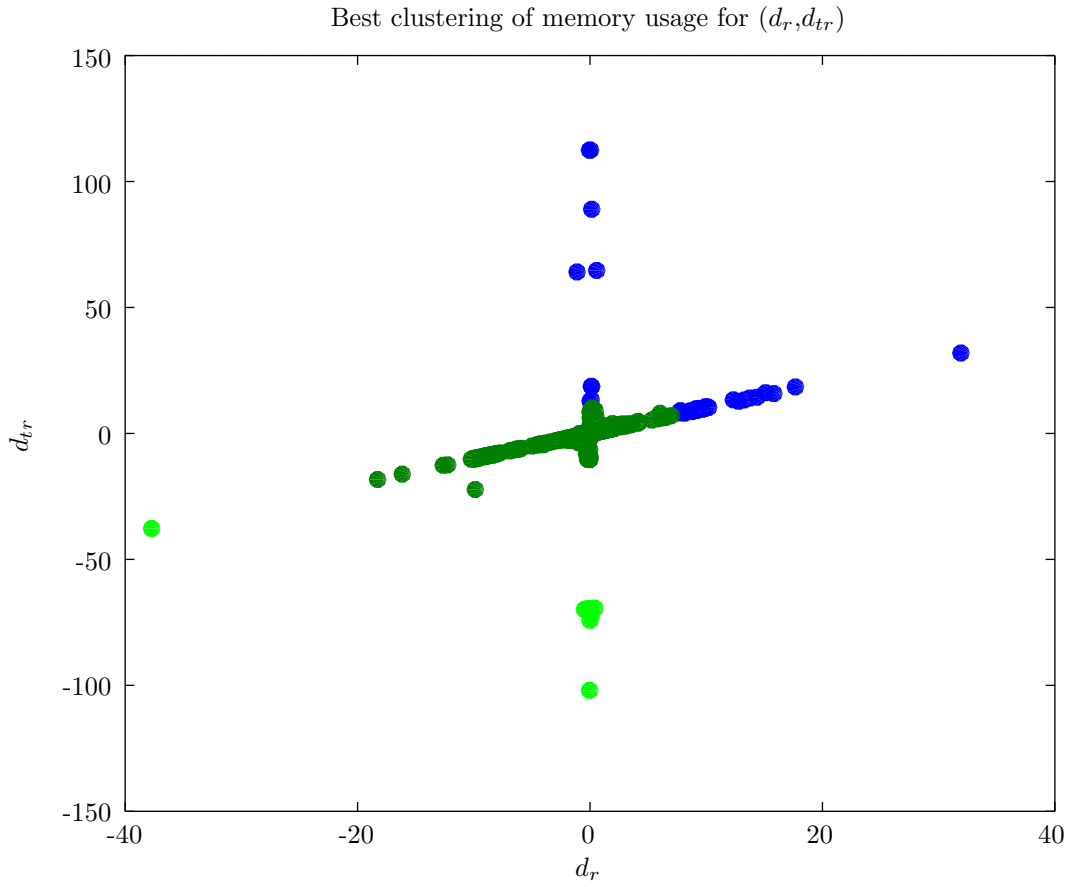


Figure D.6: Clustering with clusters shown as colours

D.3 Resulting points found

Here a number of plots that show which points from gold assignment that are taken out through centroid labelling as points that show sign of performance degradation.

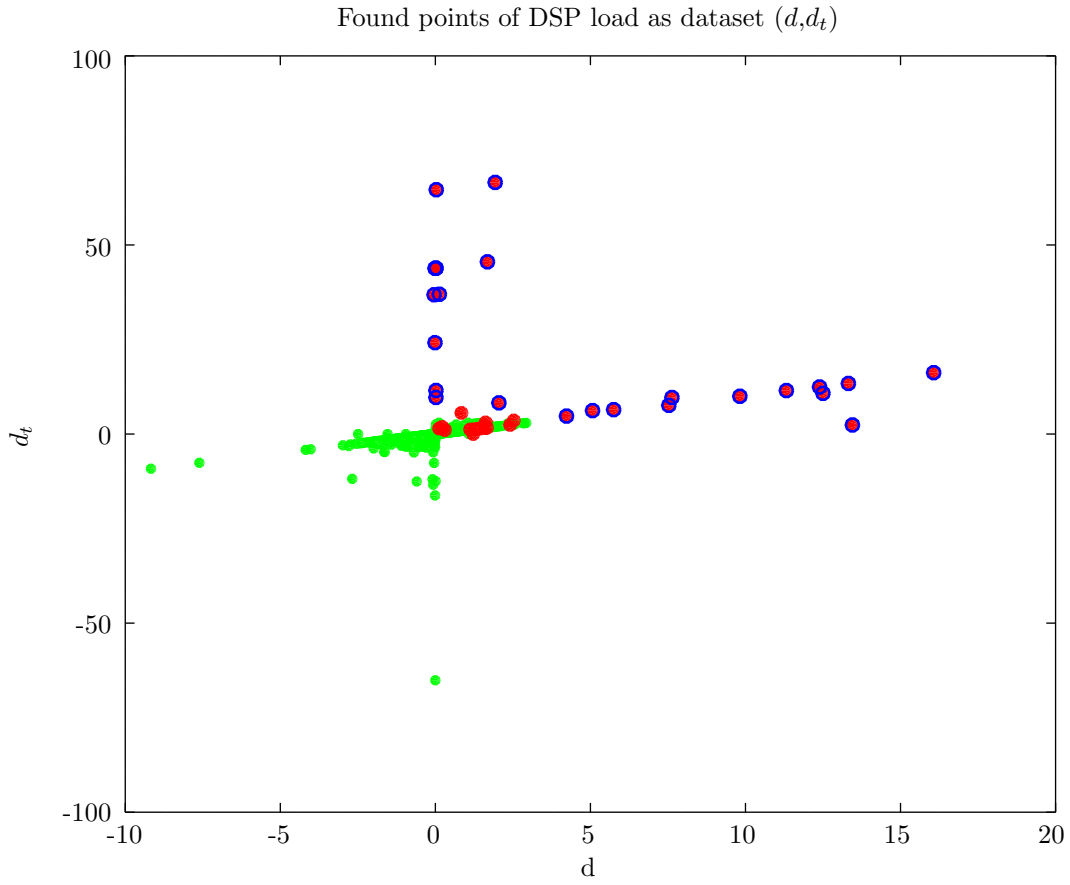


Figure D.7: According to Ericsson's gold assignment green points are categorised as normal points and red as performance degradation. The circled points are found by the fault detection system in this thesis.

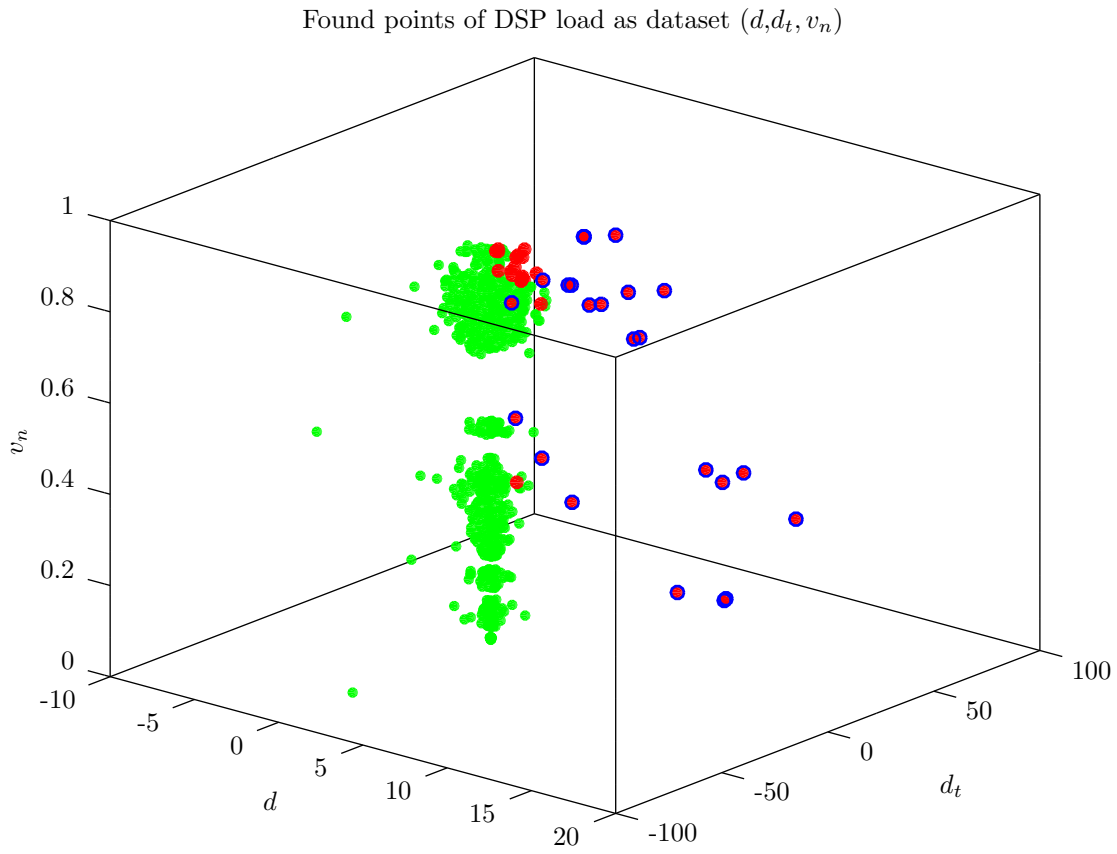


Figure D.8: According to Ericsson's gold assignment green points are categorised as normal points and red as performance degradation. The circled points are found by the fault detection system in this thesis.

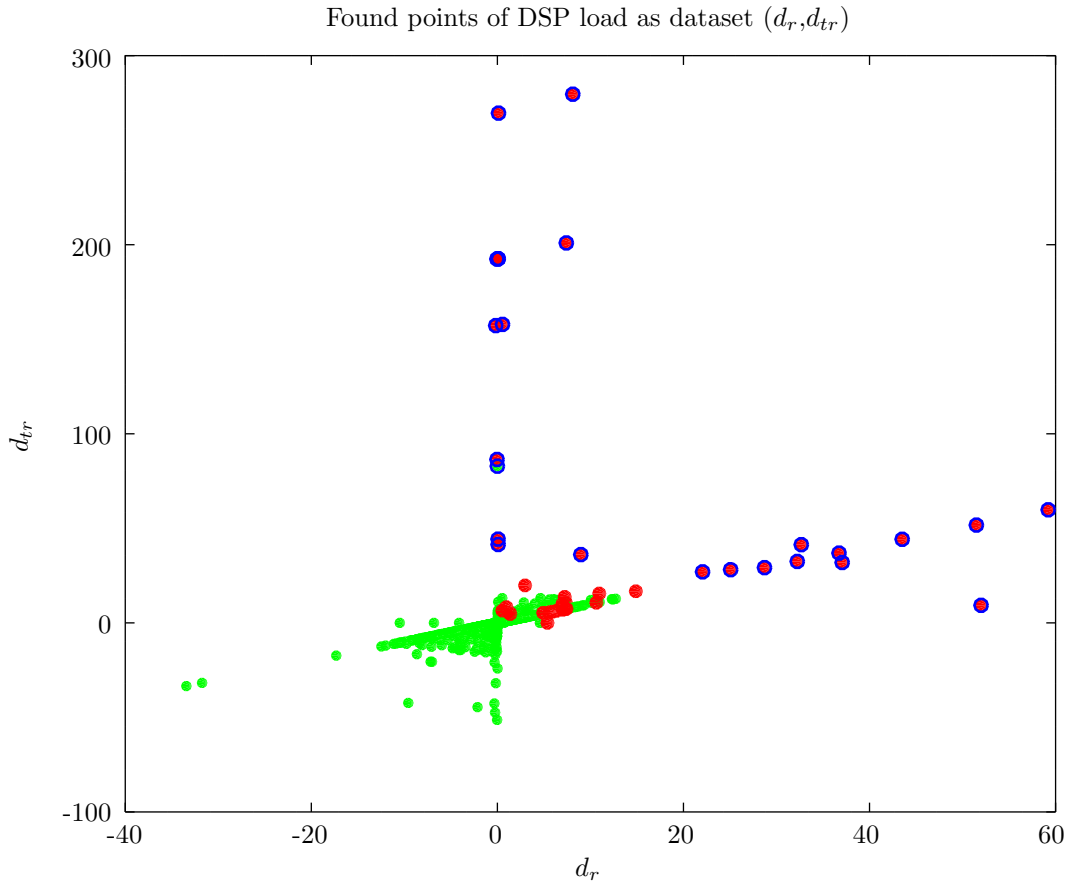


Figure D.9: According to Ericsson's gold assignment green points are categorised as normal points and red as performance degradation. The circled points are found by the fault detection system in this thesis.

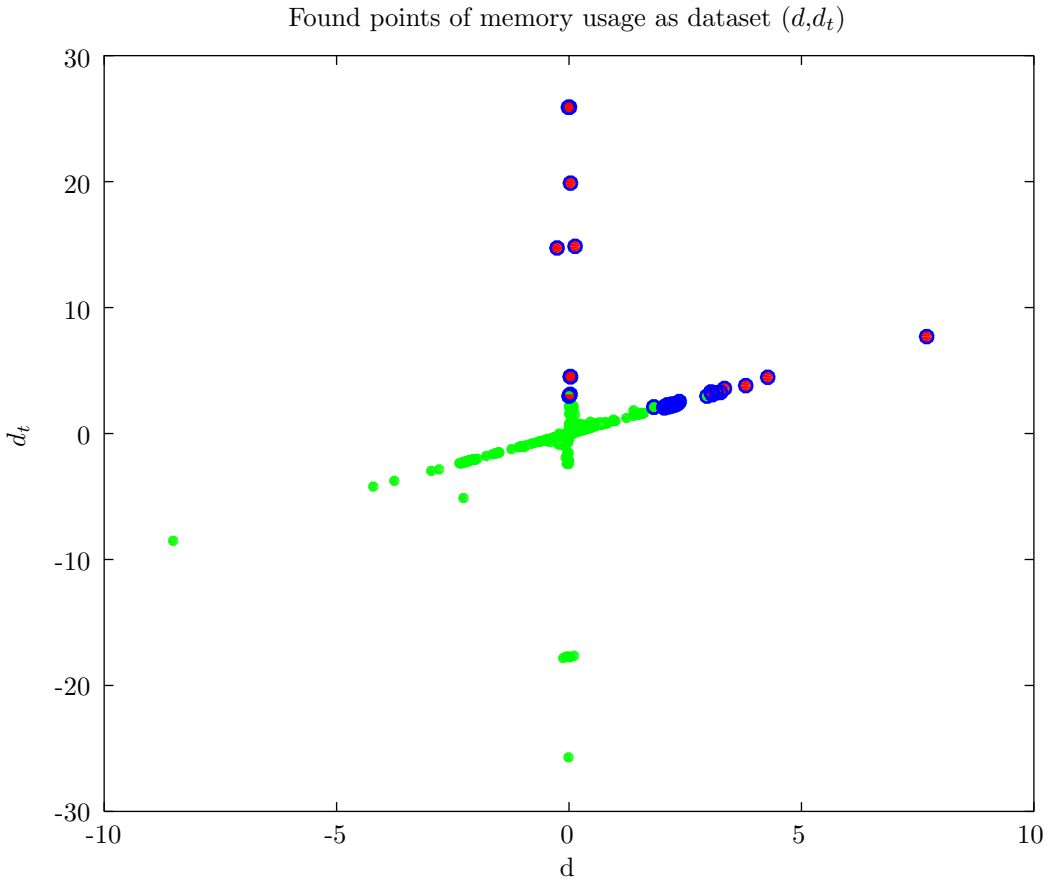


Figure D.10: According to Ericsson's gold assignment green points are categorised as normal points and red as performance degradation. The circled points are found by the fault detection system in this thesis.

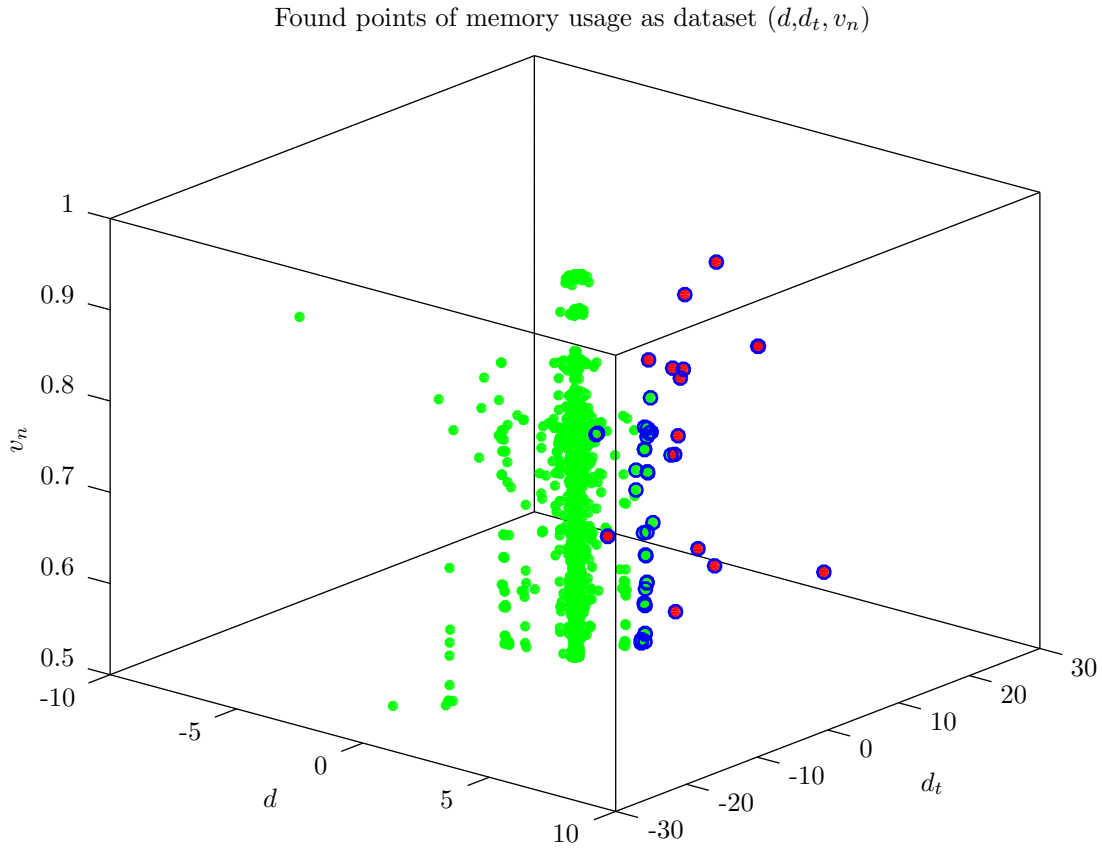


Figure D.11: According to Ericsson's gold assignment green points are categorised as normal points and red as performance degradation. The circled points are found by the fault detection system in this thesis.

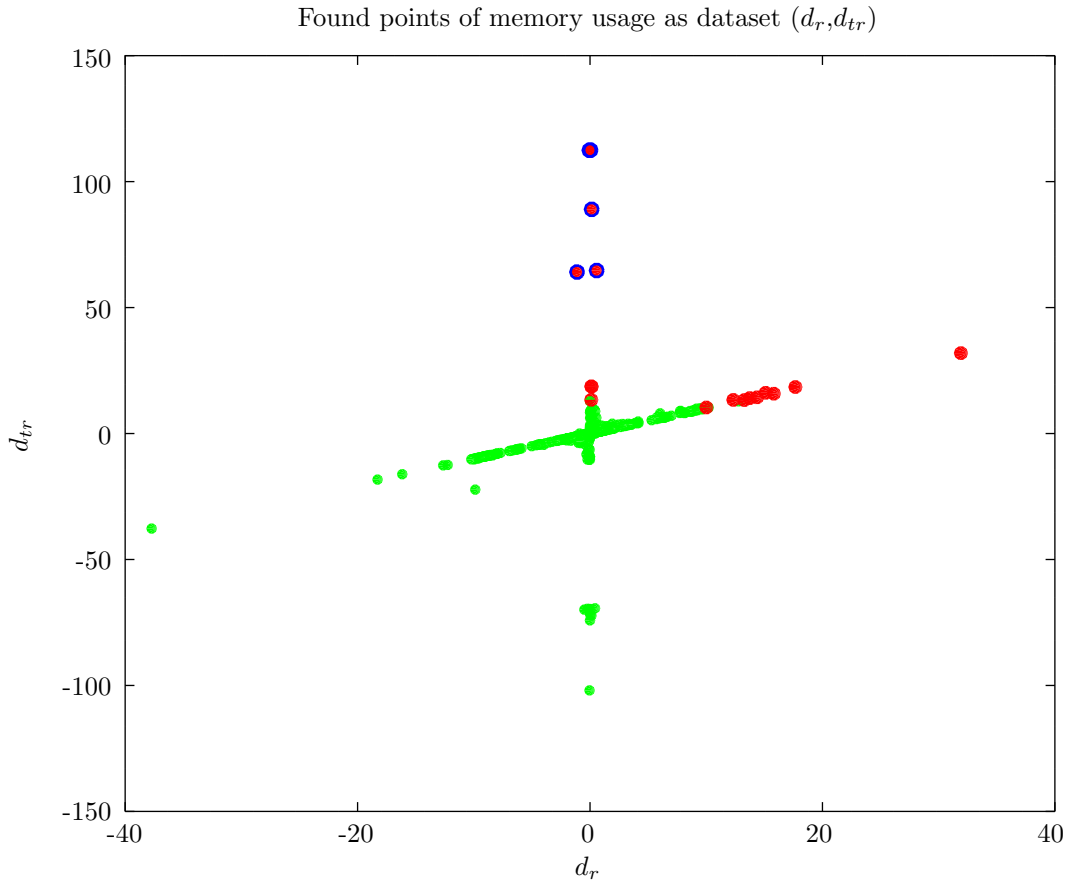


Figure D.12: According to Ericsson's gold assignment green points are categorised as normal points and red as performance degradation. The circled points are found by the fault detection system in this thesis.

D.4 Resulting marking of performance degrading jobs

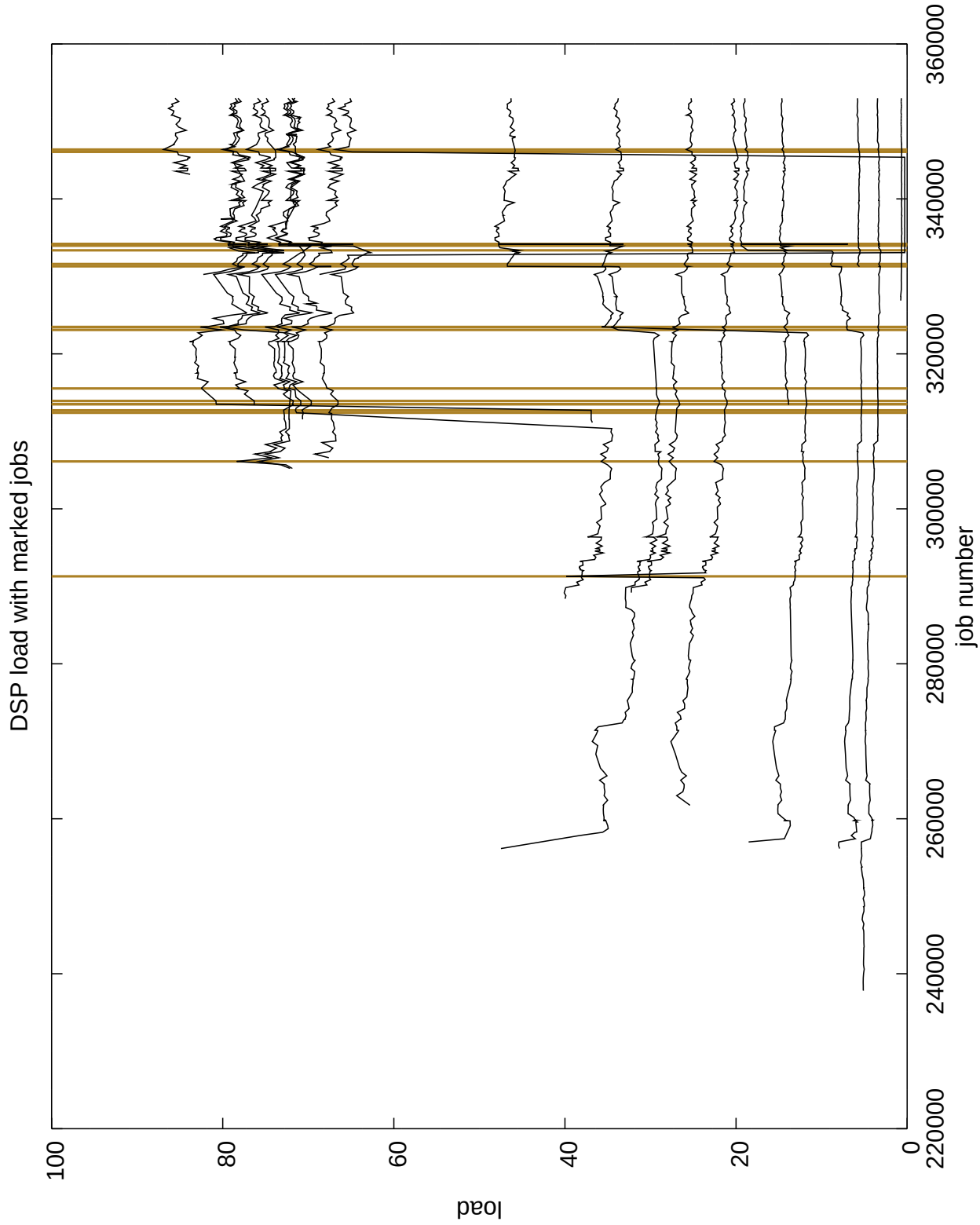


Figure D.13: Graph of all load data for DSP load with marks for jobs showing signs of performance degradation. Found with (d,d_t), city block distance, 5 clusters, $w_d = 0.5$, $w_{d_t} = 0.5$

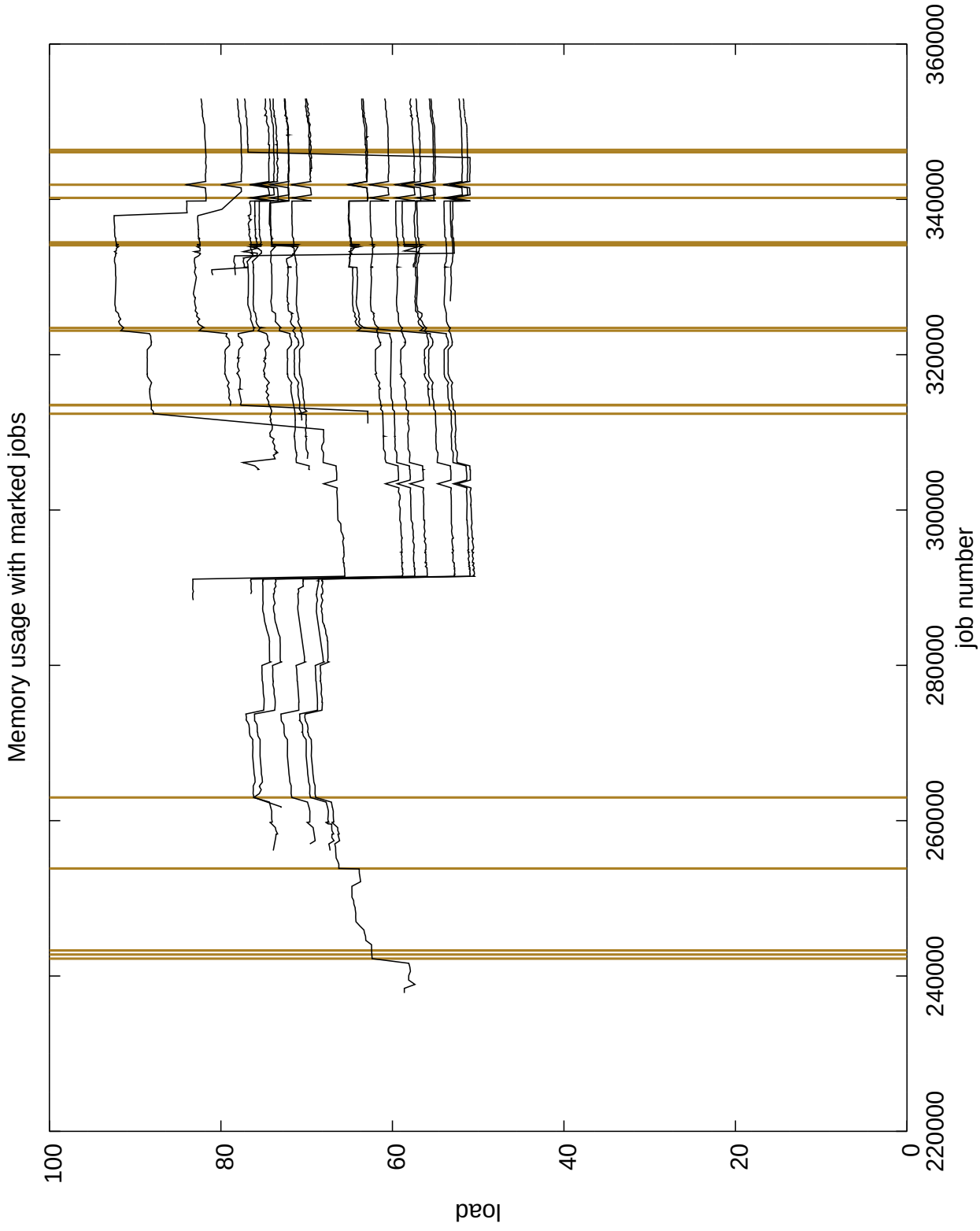


Figure D.14: Graph of all load data for memory with marks for jobs showing signs of performance degradation. Found with (d,d_t), city block distance, 3 clusters, $w_d = 0.2$, $w_{d_t} = 0.8$