TI3800 Bachelor Project

Analyser for technical analysis indicators using historical intraday values

Final Report

Authors
F. C. A. Abcouwer
M. Janssen
X. Savenberg
R. S. Verboon

Thesis Committee
Dr. M.M. de Weerdt
Dr. Ir. J.H.M. Anderluh
Dr. phil. H.-G. Gross
Dr. M.A. Larson
Summary

Technical analysis is widely used among traders, financial professionals and financial institution as a tool to identify trading opportunities. However, the academic world does not share this point of view, since they claim the evidence to promote the usability of technical analysis is sparse and inconsistent.

Noticing this impasse we decided that developing software to analyse this issue would be a way to facilitate research on the subject, and might provide a way to make an educated decision on the usability of technical analysis. The goal is to build an application that runs a strategy and shows the results of this strategy to the user.

Over the course of this project, the project group has developed such a benchmark system, with development proceeding according to an incremental, agile development technique called Scrum. The largest challenge we faced was storing and processing our data. After investigating the options, we decided to use MapReduce with an implementation of Apache Hadoop and reduced both file storage size and processing time by implementing Apache Avro. The final system can be divided into three main sections:

**Server** On the server side, the Hadoop framework was used to facilitate the implementation of a scalable MapReduce approach to the problem of going through a large amount of stock market data and producing trades using an implementation of a technical analysis-inspired strategy.

**Client** The client side manages the visualisation aspects of the project, with the GUI having been built using the JavaFX framework.

**Metrics** A calculation layer which processes the trades that the simulation generated, and calculates all the metrics shown in the user interface.

While working on the project, we worked hard to ensure high code quality and high levels of maintainability and extensibility of the system using a diverse set of tools. Testing was also an important aspect of development: using JUnit for Java, we wrote unit tests for various parts of our application. In addition, we wrote integration tests to test the proper functioning of larger parts of the system all at once.

The application delivered in the end implemented all of the features listed as ‘must have’, ‘should have’, except for one ‘should have’ feature: algorithmic optimisation of strategies. The reason for this is that implementing this feature would have taken too much time that it would have detracted greatly from the overall quality of the final application, and we therefore decided not to include it.

At the end of this project, we have developed an application accepts a strategy along with parameters defined by the users, processes it in a few minutes while showing its progress to the user, and displays customisable information about the results of the simulation in an intuitive user interface.
Preface

This document is the final report for a project undertaken as part of the TI3800 Bachelor Project course at the Delft University of Technology. Over the course of the project, we developed a system for benchmarking and analysing automated trading strategies while working at the faculty of Electrical Engineering, Mathematics and Computer Science of the TU Delft. The report contains information on our assignment, process, product and results.

We would like to thank the following people for their contributions to our project:

- Mathijs de Weerdt, for being our mentor and giving us valuable advice and support throughout the project.
- Michael Gassman, for providing us the required data and helping us to understand the content of this set.
- Jasper Anderluh, for sharing his domain expertise with us, providing valuable advice and taking place on our thesis committee.
- Hamid Belmekki, for providing us with their domain expertise, advice from the perspective of our prospective users and testing our application.
- Gunter Fluyt and Jan De Bondt, for providing domain expertise.
- Gerd Gross and Martha Larson for coordinating the Bachelor Project course, and taking place on our thesis committee.

Friso Abcouwer, Mark Janssen, Xander Savenberg, Ruben Verboon

June 2013
## Contents

1. **Introduction**

2. **Assignment**
   - 2.1 Assignment Description .......................................................... 6
   - 2.2 List of features .............................................................................. 9

3. **Process**
   - 3.1 Methodology ................................................................................ 11
   - 3.2 Project management method ......................................................... 11
   - 3.3 Roles ............................................................................................... 11
   - 3.4 Planning ......................................................................................... 12
   - 3.5 Planning tools ................................................................................ 13
   - 3.6 Development tools ........................................................................ 13

4. **Background Information**
   - 4.1 Principles of Trade Execution ......................................................... 16
   - 4.2 Moving Average Strategies ............................................................. 17

5. **System Design and Implementation**
   - 5.1 Global System Design ................................................................... 19
   - 5.2 Data Processing ............................................................................. 19
   - 5.3 Strategy Simulation ....................................................................... 25
   - 5.4 Used frameworks and technologies ............................................. 28

6. **Quality Assurance and Testing**
   - 6.1 Methodology ................................................................................. 32
   - 6.2 Process ........................................................................................... 33
   - 6.3 Results of code quality and testing .............................................. 34

7. **Final Product**
   - 7.1 Strategy upload ............................................................................ 37
   - 7.2 Setup .............................................................................................. 37
   - 7.3 Benchmark launch ........................................................................ 38
   - 7.4 Strategy result overview ............................................................... 41

8. **Conclusions**
   - 8.1 Final Product ................................................................................. 47
   - 8.2 Future Work ................................................................................... 47

9. **Evaluation**
   - 9.1 Preparation ................................................................................... 49
   - 9.2 During the Project ........................................................................ 49
   - 9.3 Lessons Learned ............................................................................ 50

A. **Dataset information**

B. **Interviews**
   - B.1 Gunter Fluyt & Jan de Bondt ...................................................... 54
   - B.2 Hamid Belmekki ........................................................................... 54
   - B.3 Jasper Anderluh ............................................................................ 55
   - B.4 Hamid Belmekki ............................................................................ 56

C. **Plan of Approach**


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1 Introduction</td>
<td>58</td>
</tr>
<tr>
<td>C.2 Assignment Description</td>
<td>59</td>
</tr>
<tr>
<td>C.3 Approach</td>
<td>62</td>
</tr>
<tr>
<td>C.4 Project Approach</td>
<td>63</td>
</tr>
<tr>
<td>C.5 Quality Assurance</td>
<td>64</td>
</tr>
<tr>
<td>D Feedback Software Improvement Group [Dutch]</td>
<td>66</td>
</tr>
<tr>
<td>E Bachelor seminarium</td>
<td>67</td>
</tr>
</tbody>
</table>
1 Introduction

This document forms the final report of our bachelor project, which took place over the course of the fourth term of the 2012-2013 academic year. The goal of the project was to construct an application that can be used to backtest strategies for stock market trading on historical market data, as a means of evaluating a (new) strategy’s effectiveness.

Technical analysis is a method to forecast prices of a stock, on the other hand there is fundamental analysis. Technical analysis uses trends in stock market in volume, price and other available data, whereas fundamental analysis takes a closer look at the company at hand to predict its value, it uses the income and profit statements.

For the Bachelor Seminar course, three of our team members had already investigated the algorithmic optimisation of trading strategies, which allowed us to gain a lot of knowledge on the field of automated trading. The paper in question has been attached to this report and can be found in Appendix E.

Even though technical analysis is widely used among traders and financial professionals, especially active day traders, it is still considered to be pseudo-science by many academics. Academics such as Eugene Fama say the evidence for technical analysis is sparse and is inconsistent with the weak form of the efficient-market hypothesis. Users hold that even if technical analysis cannot predict the future, it helps to identify trading opportunities.

Seeing the inconsistency of the conclusions of both parties, writing software to analyse technical analytic strategies in a proper metric-based manner might provide a way to make a more educated decision about the usability of these strategies. It is this pattern of thinking that led to our project assignment. The problem with existing systems is mainly the time it takes to test such a strategy.

The goal of the application developed during the course of this project is to provide the user with a clear and accurate statistical review of the trading strategy used as input. It should be used as a tool to facilitate the search for a more efficient technical analysis-inspired strategy, or to find out if a consistent strategy of that type even exists within today’s market parameters.

In this report, we first describe the assignment in Section 2, providing a description and feature list for the application. In Section 3, we explain the methodology we used for our project, and elaborate on the tools we used. Section 4 contains background information on financial trading and trading strategies, and Sections 5 and 6 go into our application’s design and our approach to quality assurance and testing, respectively. Section 7 gives an overview of the final product and its features, and Section 8 outlines our conclusions and suggestions for future work and research. Finally, Section 9 contains our evaluation of the project and lessons learned.
2 Assignment

In this chapter, we give an overview of our assignment as it was set up at the start of the project. We first describe the problem that inspired the assignment, our goals and our prospective clients. Then, we discuss the assignment in more detail and list the deliverables, risks and challenges for the project. Finally, we provide a list of features expected of the final product, arranged according to level of importance using the MoSCoW model.

2.1 Assignment Description

In this section an elaboration on the assignment can be found: the problem space and goals of the project are clearly defined, the profile of a potential client is provided, deliverables outlined and potential risks are summarised and acknowledged.

2.1.1 Introduction

The modernisation of the stock market and the development of new technologies have cleared the way for automated trading. A specific kind of automated trading that has gained a lot of popularity recently is High Frequency Trading, or HFT. Its growing popularity can be attributed to the new opportunities created by split-second trading, and the affordability of the required computing power.

Although these relatively new ways of trading have gained popularity within the financial markets, it is frowned upon by many. While proponents claim these strategies provide liquidity in the market, opponents believe them to be a danger to the intrinsic value of stock markets, because of the artificially created chain of supply and demand. The opponents also fear the limitations of HFT, of which flash-crashes are the perfect example. They believe the usage of HFT should be regulated, or even abolished.

2.1.2 Problem Description

The development and testing of HFT strategies is a hard and complex task in the financial industry. Due to the continuous race to zero delay, HFT systems have to work in the most competitive surroundings, while performing at the highest speeds available. In order to have a faster or more reliable system running before the competitors, they do not just have to be developed quickly, but also need to be tested as quickly and accurately as possible. This leads to the problem to be solved by this assignment: traders need a system that can accurately analyse results generated by running a strategy over large amounts of data, while only taking little time to generate these results.

2.1.3 Goal

By providing quantitative analysts, also known as quants, with a tool that thoroughly benchmarks and tests a strategy, a financial institution can not only profit from the time gained by speeding up the testing process, but also from early error discovery and an increase in behaviour predictability, among other benefits. This provides the institution with higher quality strategies, and facilitates the search for better and faster strategies. Because of the higher predictability of a strategy's behaviour, malfunctions would be less likely, which should also improve the general image of HFT.

Before we started work, we agreed that we would decide whether or not to include optimisation features on May 24th. This turned out to be a good choice, as we soon realised that we would not be able to create a tool capable of performing both benchmarking as well as optimisation at an
acceptable level of quality in the limited time available. Therefore, we made the choice to focus on creating a benchmarking application with no automated optimisation features.

2.1.4 Client

The idea for the assignment originally came from one of our team members. Dr. Ir. J.H.M. Anderluh (Assistant Professor in Financial Mathematics, DIAM Probability and Statistics, and Director of Fund Management company HiQ Invest) advised us on how the project can become a usable tool for the financial industry. Because of his domain expertise, Mr. Anderluh can provide us with a deeper understanding of the field, but he did not act as the client.

Plausible clients for this application are financial institutions that deal with high-frequency trading on a day to day basis. These institutions include, but are not limited to, hedge funds, proprietary trading firms and investment banks.

Our target users are quantitative analysts who design, implement, review and improve high frequency trading strategies. Hamid Belmekki, Managing Director and Global Head of Flow Quantitative Group for Fixed Income, Credit, FX and Equity for Société Générale, also provided us with feedback on our implementation, and in developing the application we considered Mr. Belmekki and his colleagues to be our ‘model users’.

To learn more about the domain we would be working in and the demands of our users, we interviewed several people working in the trading industry. A summary of these interviews can be found in Appendix B. By performing these interviews, we were able to get a better understanding of both the practices and standards used in the trading business today, as well as the features our prospective clients would expect to find in our application.

2.1.5 Assignment Formulation

During the course of this project we have developed a back-testing system for high frequency trading, in which a user can define their own trading strategy, and let the system analyse the performance this strategy on a number of different measures. The assignment has two main areas of focus:

Simulation The system analyses user-defined intraday trading strategies based on technical analysis using historical values. Part of the research involved in this project is finding out what statistical measures can be used determine the validity and profitability of a strategy.

Strategy Another important aspect of the application is the handling of trading strategies. Apart from simulating a year of trading, the system should be able to run user-defined strategies on the available data.

The application should provide users with the option to visualise these results for our full data range (a year of the DAX 30), by Global Industry Classification Standard, or stock by stock individually. This option is crucial, because it allows checking whether a strategy performs better in certain sectors of the market. If the user is only interested in certain parts of the market, they should be able to only view results for those sections.

In short, the list of features of the application is as follows. The application should be able to:

- Graphically represent historical data and the results of simulations
- Have users define and simulate their own strategies
- Provide users with an overview of how their strategy performed, based on performance measures such as risk, number of trader or profit
- Provide users with suggestions on how to improve their strategy (optional feature)
2.1 Assignment Description

2.1.6 Deliverables

By the end of the project, our goal was to have completed the following:

- A plan of approach
- An orientation report
- A software package that can complete the tasks as described in the assignment description
- A final report conforming to the Bachelor Project course guidelines
- A final presentation

The plan of approach can be found in Appendix C on page 58. This document, which also contains most our findings from the orientation report, serves as the final report. Our code has been sent to the Software Improvement Group for evaluation: further information on this can be found in Appendix D on page 66. Our final presentation will be on Wednesday, July 3rd, 2013, at 14:00.

2.1.7 Risks

When we started, we expected the two biggest risks of our project to be as follows:

1. **Data acquisition proves too difficult.** The data needed for this project is financial market data. In a commercial project, licenses for tick-data can exceed several thousands of dollars per month. Our challenge is to acquire data which is representable as HFT data, but does not cost us anything.

2. **We underestimate the tasks ahead, or overestimate our knowledge.** In other words: the assignment turns out to be too complex. This is a common pitfall in many software development projects, and there is no reason to simply assume there is no possibility of us experiencing this as well.

As we had anticipated acquiring the data would prove difficult, we had already put effort into this before we officially started work on the project. Because of this, we managed to acquire the data we needed before we started the project. This leaves us with the option that the project would prove too challenging to complete on time. To mitigate this risk, we chose to postpone deciding on whether or not to include strategy optimisation until after a first period of initial development: this proved to be a good choice, as we realised that including it would not be feasible given our time constraints. After we had made the decision not to include optimisation features, we were able to focus our work on the other parts of the application, in the end creating a system much more useful and robust than one that would have resulted from trying obsessively to include every possible feature.

2.1.8 Challenges

Apart from ‘external’ risks such as data acquisition, carrying out the project itself also posed some challenges. We identified these as follows:

- **Data Storage and Processing.** The full year of data we acquired consisted of over 370 million data points formatted in a CSV file. It soon became clear that an efficient way to store, access and handle this data would be required; otherwise, our application could turn out to be sluggish and inefficient, thus useless to our prospective users. When we interviewed Mr. Belmekki, we were told that simulating a strategy at Société Générale is done by the trader leaving their own computer powered on overnight, and coming back the next morning to see the results. Mr. Anderluh, on the other hand, said that at his company, running a strategy over a few months’ worth of data took around five minutes on his laptop. Though these are extreme examples,
we realised that it would be important to reach a level of simulation speed our prospective users could be satisfied with. In short, this meant that making sure our data was stored and processed efficiently and swiftly would be one of our most important challenges.

- **Users.** Finally, as with nearly every software development project, taking into account the needs and wants of the user was a challenge in and of itself. Regardless of how well we stored our data and how fast we could process it, if we could not present it to the user in a way that is useful and understandable to them, we would have built our application for nothing. In our initial interviews, our prospective users all agreed that a well-designed interface would be a necessity for an application like ours.

In the following chapters, we will explain in detail how the system we have designed handles these challenges.

### 2.2 List of features

Below is a list of features for our application, arranged according to the MoSCoW model. The requirements and their ordering were arrived at through interviewing prospective users on what features they would like to see in our application, and how important they considered each feature.

#### 2.2.1 Must have

‘Must-haves’ are required for an application with the most basic level of performance. These features were chosen because they represent the most critical functionality the program should possess.

- **Simulation** The application should simulate a given strategy over the provided dataset.
- **Speed** The time needed to process a strategy over the full dataset should not exceed 10 minutes.
- **GUI** The GUI should provide the user with a clear overview of the results of the simulation.
- **Strategy input** A user should be able to input their own strategy in a straightforward, simple way.
- **Results** The following information must be presented to the user as a result of the simulation:
  - Total net profit - This is the most basic indicator of a strategy’s quality.
  - Total number of trades made - If there are not enough trades, the results can not be considered statistically valid.
  - Equity high and low - The range of equity values is important to the trader to see if the strategy is not overly aggressive.
  - Exposure - Exposure is a basic risk measure.

#### 2.2.2 Should have

Implementing all ‘should-haves’ along with the ‘must-haves’ will make for an application that, along with its most important core features, also contains all features that can also be considered important, but not critical.

- **Algorithmic Optimisation** Automated strategy optimisation. This can be divided into two aspects:
  - Automated parametrical optimisation - The program should be able to suggest parameter values that would provide improved results over the ones the user initially selected.
Automated strategy improvement - Apart from tweaking a strategy’s parameter values, the program should also be able to suggest improvements to the strategy itself.

**Speed** The time needed to process a strategy over the full dataset should not exceed 7 minutes.

**Parametrical Tweaking** The user should be able to personally alter their parameters, then re-run the strategy.

**Market Information** The application should visualise and summarise information on the simulation’s market.

**Trade Overview** The application should display a full overview of all trades executed during the latest simulation.

**Results** The following information should be presented to the user as a result of the simulation:

- Equity curve - An equity curve represents most of the behaviour of a strategy.
- Equity standard deviation and variance - These are basic indicators of risk, though using them on their own can be unreliable.
- Winning/Losing trade information - The ratio of winning to losing trades is also important to traders.
- Periodical overviews (weekly/monthly) - Traders want to be able to check the results for a specific time frame, for example in a week where the market was very volatile.

**2.2.3 Could have**

‘Could-haves’ are noncritical features that can be characterised as ‘nice to have’. While not terribly important to the functioning of the program, implementing these features would improve the user experience significantly.

**Speed** The time needed to process a strategy over the full dataset should not exceed 5 minutes.

**Analysis Customisation** The user should be able to select which stocks and what timeframe to use during the simulation.

**Progress Feedback** The user should be able to see the progress of the current simulation.

**Connection Status** The user should be able to see the status of the simulation service. This could be a local process or a remote execution platform.

**Strategy Examples** Examples of different trading strategies, and how to use them as input.

**Results** The following information could be presented to the user as a result of the simulation:

- List of all trades - Though listing all trades that were carried out by the strategy can be an overload of information, it can be useful in some cases to have access to this set of data, e.g. to thoroughly check the behaviour of the strategy in the event of a flash crash.

**2.2.4 Would like**

As the name implies, these are features that we ‘would like’ to implement, but that we feel are neither of critical importance to the application nor realistically attainable in the time span of this project.

**Integrated Strategy Designer** An IDE-like subsystem in which the user can define strategies.

**Report Generator** Summarises the results of the latest simulation and exports them to specified format.
3 Process

In this section, the methodology, planning and tools used in our project are discussed in detail.

3.1 Methodology

The entire process from initial research and orientation up to developing the system took place in room HB07.230 on the seventh floor of the Electrical Engineering, Mathematics and Computer Science building of the Delft University of Technology. Working together in a single room, as well as using a proper process management model significantly eased cooperation and communication.

3.2 Project management method

In order to stimulate cooperation within the group, the need for a proper project management method was soon acknowledged. We identified two modern agile project management approaches that seemed to match our vision on project management: Scrum\(^1\) and Kanban\(^2\). For more information please refer to the manuals found on the given websites.

We opted for the Scrum project management approach since we have more experience using this approach, and since Scrum seemed to be more appropriate for guiding a software development project using a small team than Kanban.

When using the Scrum approach, certain roles, such as the product owner and the scrum master need to be assigned to team members. The assignment of these (and some other) roles went very naturally and did not even need a real discussion.

Every week started with a weekly sprint planning meeting on Monday morning, to discuss what work would be done that week. Each morning after that, there is a daily standup, where every team member updates the others on his progress, announces his plans for that day, and where the team can discuss potential issues. Every Friday afternoon, there was a sprint review meeting, usually with Mr. De Weerdt, reviewing that week’s progress. After that, there was a sprint retrospective meeting with the team, which was always held as informal as possible, keeping the goal of reflecting on that week’s sprint and finding ways to optimise our process the following week(s) in mind.

3.3 Roles

We identified five main areas of interest for our group, each of them crucial to our success. These were the product itself, software development practices, quality assurance and testing and reports. We believed making a team member responsible for every one of these areas would help us succeed, and our mentors would also each carry out a specific role in the project. Based on this, we divided roles amongst the team members and mentors as follows.

\(^1\)http://www.scrum.org/
\(^2\)http://www.kanbanblog.com/
### Table of Roles and Responsibilities

<table>
<thead>
<tr>
<th>Name</th>
<th>Roles</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friso Abcouwer</td>
<td>Scrum and Reports</td>
<td>Responsible for the Scrum process and editing and finalising reports.</td>
</tr>
<tr>
<td>Mark Janssen</td>
<td>Lead Development</td>
<td>In charge of the software development process.</td>
</tr>
<tr>
<td>Xander Savenberg</td>
<td>Project Manager</td>
<td>Manages product requirements and the project in general.</td>
</tr>
<tr>
<td>Ruben Verboon</td>
<td>Quality Assurance</td>
<td>In charge of ensuring high code quality and proper testing procedures.</td>
</tr>
<tr>
<td>Mathijs de Weerdt</td>
<td>Group Mentor</td>
<td>Supervises the group during the project.</td>
</tr>
<tr>
<td>Jasper Anderluh</td>
<td>Group Mentor</td>
<td>Domain expert. Will provide feedback from the perspective of the user.</td>
</tr>
<tr>
<td>Gerd Gross</td>
<td>Coordinator</td>
<td>Coordinator of the Bachelor Project course.</td>
</tr>
</tbody>
</table>

### 3.4 Planning

During the project, we worked in sprints with a length of 5 working days. Our initial week-by-week planning for the project was as follows, with the dates below indicating what our goal will be for the end of each week.

- **28 April**: End of Sprint One: Drafts of Plan of Approach and Orientation Report, setup of the development environment.
- **8 May**: End of Sprint Three: Basic trading engine that can ‘stream’ price ticks, and basic visualisation.
- **17 May**: End of Sprint Four.
- **24 May**: End of Sprint Five: Definitive decision on whether to include optimisation or not.
- **31 May**: End of Sprint Six.
- **7 June**: End of Sprint Seven: First draft versions of a majority of the final report.
- **14 June**: End of Sprint Eight: Feature-complete version of application finished, code sent to SIG for review.
- **21 June**: End of Sprint Nine: Last improvements, implementation of SIG feedback, final report and presentation nearing completion.
- **28 June**: End of Sprint Ten, end of development: code sent to SIG, finished final report and presentation.
- **3 July**: Final presentation.

Because of several days in sprints two and three where the faculty was closed, we combined these two sprints into a single sprint of six working days. Because we had not properly taken this account, we had to move some work forward, which delayed some things we would have preferred to finish earlier. Towards the end of the project, however, we fully caught up on our backlog and were back on schedule for the last three and a half weeks.
3.5 Planning tools

Trello

At the start of the project, we created accounts on Trello\(^3\) to keep track of todo’s online. However, we soon found out that using Trello as well as a physical scrum board meant we would be tracking tasks twice. This led us to use the scrum board for most of our todos, and to only use Trello for logistical tasks, such as tracking whose job it was to reserve a room for our final presentation or buy office supplies. After most of those tasks had been completed, we abandoned the use of Trello and tracked the few remaining logistical tasks on the scrum board. While Trello can be a useful tool in some situations, in the end it was not suited to our way of working on a project.

PangoScrum

In addition to the physical Scrum board, we used PangoScrum\(^4\) to keep track of our backlog online. Though PangoScrum is still in beta, it offered some features that we found useful for managing our Scrum process, including tools for mapping out products and sprints and planning scrum meetings such as the sprint planning and review meetings. In this way, we were able to separate our general planning from our day-to-day planning, using PangoScrum and our physical board for each of these, respectively.

3.6 Development tools

IntelliJ IDEA

IntelliJ IDEA will be our Integrated Development Environment (IDE) of choice, facilitating development and testing of our code. IntelliJ IDEA comes with a wide set of code inspections and test coverage analysis that we will be using to guarantee high levels of code quality.

GitHub

GitHub will be used to host our code using the Git version control system, and to review code. We use the concept of feature branches combined with pull request, which we elaborate on in Section 6.1.

Maven

Apache Maven\(^5\) is a project management tool that can be installed as a plugin in IntelliJ and ensures that the Java application source code is well structured according to a defined standard. Using a Project Object Model (POM file), maven can manage a project’s dependencies, and build, test, report and document the project.

\(^3\)https://trello.com/
\(^4\)http://pangoscrum.com/
\(^5\)http://maven.apache.org/
3.6 Development tools

**Jenkins**

To ensure tests are checked frequently enough, we included the use of Jenkins in our project. Jenkins is a free open-source continuous integration service for software development, which also reports on test-suite successes. Since this continuous integration service is hooked to the project’s GitHub repository, the full project is rebuilt and retested on every commit. This way, every commit is guaranteed to pass all tests before it can be merged into the master branch.

**JUnit**

Another important plugin is JUnit. This plugin allows for automated unit testing of the application. In Section 6.1 we will go into more detail on the actual unit testing. By using JUnit in combination with Jenkins, reports were generated that display the amount of tests, and in particular the amount of tests that were unsuccessful.

Furthermore, IntelliJ provided us with rich JUnit test integration. Tests can be run directly from the IDE and test results are presented in an easily understandable way. When tests fail, IntelliJ shows what test assertion failed along with a stack trace to assist further debugging.

**Spring ReflectionUtils**

The Spring ReflectionUtils and ReflectionTestUtils packages can be used for working with the Java Reflection API. This API can be used to access classes, fields, methods and constructors during runtime, even if they are protected or private. This offers programmers better opportunities for debugging and testing, as it allows them to access fields that would otherwise be inaccessible. An example of using ReflectionTestUtils to access a private field can be found in Listing 1 through this setup method, the periodSelector combobox can be accessed freely during the test, even though it is actually a private field of the MarketController.

```
@Before
public void setUp() {
    marketController = new MarketController(null);
    periodSelector = new ComboBox<String>();
    ReflectionTestUtils.getField(marketController, "periodSelector", periodSelector);
}
```

**Mockito**

We also used Mockito for testing. Mockito is a framework that allows ‘mock’ objects, which can emulate aspects of the behaviour of real objects. This is useful in testing for several reasons: it reduces dependance on external classes that are not the object of the test in question, and it makes testing easier because mocked objects are very predictable.

[http://junit.org](http://junit.org)
[http://static.springsource.org/spring-framework/docs/3.2.3.RELEASE/javadoc-api/org/springframework/util/ReflectionUtils.html](http://static.springsource.org/spring-framework/docs/3.2.3.RELEASE/javadoc-api/org/springframework/util/ReflectionUtils.html)
3.6 Development tools

Listing 2: Example of Mockito

```java
@Before
public void setUp() throws Exception {
    exposureController = new ExposureController(null);

    marketListing = Mockito.mock(MarketListing.class);
    asset1 = Mockito.mock(DAX30Asset.class);
    asset2 = Mockito.mock(DAX30Asset.class);

    ReflectionTestUtils.setField(exposureController, "marketListing", marketListing);
    HashMap<String, DAX30Asset> testMap = new HashMap<String, DAX30Asset>();
    Mockito.when(asset1.getISIN()).thenReturn("Test1");
    testMap.put("Test1", asset1);
    Mockito.when(asset2.getISIN()).thenReturn("Test2");
    testMap.put("Test2", asset2);
    Mockito.when(marketListing.getStocks()).thenReturn(testMap);
}
```

In Listing 2, an example of the use of Mockito is given. The MarketListing and DAX30Asset classes are mocked, and told what to return when specific methods are called. If we had wanted to attain this behaviour with 'real' objects, it would have required a lot of extra code to be written, and introduced room for errors on the part of those two classes.

IntelliJ Code Coverage

To allow for good writing of automated unit tests, it is important to assess the code coverage that is realised by those tests. Important classes require high coverage (with exception of simple methods like getters and setters) to ensure the correctness of their functionality. Using IntelliJ’s built-in code coverage functionality, test coverage of our project could be assessed. With these results, additional tests could be written in case e.g. branch coverage was too low, and redundant tests can optionally be removed. In section 6.1, more details about this project’s code coverage are provided.

Checkstyle

Code aesthetics may not matter much for the functionality of the system, but they make a great difference when it comes to maintainability of the code. It is important to ensure that code is readable, consistent and easy to understand. By combining Maven and Checkstyle, we could run the `mvn site` command, which produced a report on code style violations, allowing us to eliminate these.
4 Background Information

In this section we will outline some background information relating to our problem domain. We will first describe the basic principles of (high frequency) trading, and then give some examples of the concepts behind several popular trading strategies.

4.1 Principles of Trade Execution

The process of a security transaction is described elegantly by Gehrke et al. in their 2004 paper [8]: the following section is based on their explanation. The example they use is especially relevant to our system, as they describe the process as it takes place at the Deutsche Börse. Gehrke et al. describe a security transaction as consisting of the following steps:

1. Information and advice
2. Registration of the order and its routing
3. Matching and price determination
4. Clearing and settlement

Information and Advice

The first phase consists of investors investigating the market and deciding what securities to invest in. In order to do this, they need information on current events as well as information on current prices, or price quotations. These can be obtained directly from the stock exchange or from financial service providers like . An automated trading system cannot take into account complex market conditions such as news reports, however, allowing us to ignore this step.

Registration of the Order

Xetra®, Clients, Xetra® being the trading system used by the Deutsche Börse, transfer investors’ buy and sell orders to the electronic order book kept by the stock exchange. However, since our project does not include automated trade execution, it will not need any order registration or recording.

Matching and Price Determination

In this phase, bid and ask orders are compared, in order to find possible ‘partners’ who are willing to buy or sell at the prices they have each indicated. The bid price is the price a buyer would offer to purchase a security, and, accordingly, the ask price is the price a seller would be willing to sell their securities at. Because the level of detail in the data we have received does not allow for full order book reconstruction, our system does not simulate a full order book, and therefore only handles the most basic type of trades: (“Market orders”. As described by Gehrke et al.:

Market Orders are unlimited buying and selling instructions that will be carried out at the next price determination.

In short, this means that a market participant places an order to buy or sell, regardless of the price at the time of execution. After all, because there is some time (usually a few milliseconds) between placing an order and its execution, the price is almost never the same at these two points in time.
Because the price difference is usually not very big in the small amount of time it takes to execute an order, this is an acceptable simplification.

A day’s trading begins with an opening auction, wherein all outstanding orders from the previous day of trading and all orders that have been submitted during the pre-trading phase are processed. Prices are determined through the principle of most execution: that is, the price chosen is the price at which the largest amount of people want to buy and sell.

### Clearing and Settlement

In this phase, the Xetra® system carries out the trade, overseeing the transfer of the security being traded to the buying party and the transfer of the monetary compensation to the selling party, and sends a receipt to both parties. Our project focuses on mimicking the behaviour of the “Matching and Price Determination” phase. Since our goal is to backtest given strategies, the “Information and Advice” phase is not needed because of the idea of automated trading strategies, and “Matching and Price Determination” and “Clearing and Settlement” can be skipped since automated trade execution (on a real stock exchange) is not needed in order to backtest a system.

### 4.2 Moving Average Strategies

Technical analysis is based upon predicting the stock price with past stock data. One of the most useful techniques in Technical analysis is the double moving average (DMA).

#### 4.2.1 Double Moving Average Crossover Strategy

Using moving averages is one of the most popular methods of technical analysis. Many traders use them to spot trend reversals, especially in a moving average crossover, where two moving averages of different lengths are placed on a chart. Points where the moving averages intersect can signify buying or selling opportunities (see Figure 1).

**How to compute a moving average**

A simple form of a double moving average is calculated by the mean of the past \( n \) time points. For example, to calculate a basic 5-point moving average you would take the arithmetic mean of the past five data points. In Table 1, the sum of the past 5 points is divided by the number of points to arrive at the 5-point average.

A \( n \)-point moving average can be achieved by taking the arithmetic mean of the past \( n \) data points. This value shows the evolution of the stock.

The method is called a *moving* average, because at each new datapoint, the average is calculated again. The new average is calculated on the past \( n \) points. This can quickly be achieved by subtracting the last value and adding the new value.
4.2 Moving Average Strategies

| 5 7 6 8 9 | 35/5 = 7 |
| 7 6 8 9 7 | 37/5 = 7.2 |
| 5 7 6 8 9 10 | 40/5 = 8 |
| 5 7 6 8 9 10 12 | 46/5 = 9.2 |

**Table 1:** An example of the calculation of a 5-point moving average.

**The double moving average as a trading strategy**

We can turn this mathematical concept into a simple strategy by combining two moving averages, i.e. a double moving average (DMA). By picking a short-term moving average (for example 50 points) and a long-term moving average (150 points) and plotting these, we can identify moments when trading would be advantageous. In this case, this means buying when the short-term moving average crosses over the long-term moving average, and selling when the long-term moving average crosses over the short-term moving average. This is illustrated in Figure 1.

![Figure 1](chart.png)

**Figure 1:** This chart of the DAX30 illustrates the buying or selling opportunities when the method is used in a crossover strategy. Notice how the DMA crossover announces the beginning and the end of the down trend of the chart.
5 System Design and Implementation

5.1 Global System Design

As a result of the research and orientation iterations we decided on the system architecture illustrated in Figure 2. In this design a server-client separation is clearly visible. We have decided to use the DAS4 Hadoop Cluster as the environment to run this server component on, in order to obtain sufficient processing power, our reasoning for which is explained later in this section.

The server component has several responsibilities:

**Hadoop and HDFS** are components already available on the DAS4 cluster.

**Hadoop Client** The Hadoop client will act as supervisor for Hadoop jobs. It will submit new jobs to the Hadoop job tracker and watch their status.

**Service** The service component forms the interface to the client subsystem. This component is responsible for running new strategies using the Hadoop Client and returning simulation data to the client.

The client component receives actions from the user, and consists of several subcomponents:

**GUI** The graphical user interface is the JavaFX system that the user sees on-screen.

**Client Core** The client core is the component which communicates with all other subcomponents to form a fully operational system.

**Visualisation processing** The visualisation processing component that translates the results into something suitable for display on screen.

**Strategy Design** The strategy design component allows the user to select a strategy and customise its parameters.

5.2 Data Processing

5.2.1 Available options

One of the main challenges we identified at the start was processing the large quantities of data required to generate a result. Even on the most recent consumer-grade desktop or laptop computers,
simply reading our data would take several minutes at the least, excluding any time added by further processing. Extrapolating the result from figure 3 to the full data set of 13GB, it would take about 35 minutes to turn the raw data into Java objects, not accounting for time needed for generating, analysing and presenting a strategy’s results. This led us to conclude that it would be better to process the data on a more powerful system.

The problem at hand can be classified as a big data problem. Brown et al. state [2] that we are in ‘the era of big data’, and it is certainly not difficult to imagine why parties active in financial markets would be interested in this field, considering the vast amounts of data they work with every day. Having realised that processing all of the data serially would take a long time, even on a very fast computer, we turned our attention to parallel data processing. After all, it seems obvious that an execution platform with not only a lot of computing power, but also the ability to divide the problem up into pieces and simultaneously process those pieces, instead of serially going through the data, could, if implemented properly, provide a result in less time than a more basic, serial approach would.

While serially testing a strategy is relatively easy, parallel processing provides a somewhat bigger challenge: after all, strategies make decisions based on previously attained results within the same simulation, so one can not simply divide the data up into pieces and process everything separately. This led us to the MapReduce [14] approach to parallel programming. This solution has gained a lot of popularity the past years in the field of data processing: today, a wide variety of applications use MapReduce [3]. Some examples that could be considered to be related to this work are the use of MapReduce for machine learning by Chu et al. as described in their 2007 paper [4], or the data mining performed by Grossman et al. in 2008 [5]. We will elaborate on the workings of MapReduce in Section 5.2.2.

MapReduce in itself is just an algorithm, of which implementation and execution can be done using several different platforms and technologies. A popular open source MapReduce implementation is Apache Hadoop [6]. Furthermore, several proprietary implementations are available, such as Google BigQuery. The options we consider for execution are local execution with Hadoop, Amazon Web Services’ Elastic MapReduce [7], Google BigQuery [8], and the DAS4 grid which has its own Hadoop cluster [9]. In table 2 a comparison is made between the four different execution platforms. We have chosen the DAS4 grid, since it has sufficient power and is available to us at no cost.

5.2.2 About MapReduce

MapReduce offers a model in which it is relatively easy to program in parallel [14]. MapReduce features two phases: a mapping and a reducing phase. MapReduce starts off by taking a chunk of data, distributing it among several mappers. The mappers then map data to a key-value structure. An example: each mapper receives one input line of a file and counts the words. The corresponding key values could then be <hello, 1> <world, 1>. This is illustrated in Figure 4.

<table>
<thead>
<tr>
<th></th>
<th>scalability</th>
<th>cost</th>
<th>power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Local Hadoop Execution</td>
<td>–</td>
<td>no</td>
<td>–</td>
</tr>
<tr>
<td>2) Amazon Elastic MapReduce</td>
<td>++</td>
<td>yes</td>
<td>++</td>
</tr>
<tr>
<td>3) Google BigQuery</td>
<td>++</td>
<td>yes</td>
<td>++</td>
</tr>
<tr>
<td>4) DAS4 with Hadoop</td>
<td>o</td>
<td>no</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 2: MapReduce implementation comparison
These key-value pairs are distributed among the reducers. Each reducer receives all pairs with the same key. The reducer can then aggregate the key value pairs.

![Diagram](image_url)

**Figure 3:** The MapReduce model in our system is composed of financial data (top). Mappers generate key-value pairs, and send these to the reducers. Each reducer gets all pairs corresponding to the same key. The reducers can then process the pairs.

To provide more context of how MapReduce works, we provide some pseudocode of a simple program that runs a method on a set of trades.

Using sequential execution, this would look something like this:

**Listing 3:** Example of a simple sequential trader

```python
def sequential(list):
    trades = new list
    for each tick in list:
        trade = runStrategy(tick)
        if (trade != null):
            trades.add(trade)
    output.write(trades)
```

In this algorithm, the strategy handles each tick sequentially. If we, however, take advantage of the fact that we do not need last week's ticks to properly run a strategy on this week, we can split up our data set per week. Furthermore, if we have a single-stock strategy, we can also split the dataset per stock (per ISIN). After splitting the dataset, the strategy can be executed on each subset in parallel.

The following example provides a simple MapReduce algorithm that does exactly this:
Listing 4: Example of a simple MapReduce trader

```python
function mapper(input):
    output = new list
    for each tick in input:
        if (currentTick.ISIN == previousTick.ISIN &&
            currentTick.YearWeek == previousTick.YearWeek):
            output.add(currentTick)
        else:
            execute_asynchronous(reducer(output))
    output = new list
    previousTick = currentTick

function reducer(input):
    trades = new list
    for each tick in input:
        trade = runStrategy(tick)
        if (trade != null):
            trades.add(trade)
    output.write(trades)
```

In the mapping phase, trades are grouped per ISIN by week, so that the reducer method gets trades for a single stock (ISIN) in exactly one week. The reducer method simply runs the strategy just like the sequential algorithm does. However, the key here is that multiple reducers run in parallel. This means that once a complete subset of data is found, the first reducer can start running the strategy on this subset. Splitting data then continues while reducers asynchronously (i.e. on another thread, in another process or on an entirely different machine) complete the strategy on their subset of data.

### 5.2.3 Data Format

For data storage, Hadoop uses the Hadoop Distributed File System (HDFS). The system works using multiple file servers, with HDFS managing multiplication and distribution of data. The advantages of this system are durability and accessibility: data is safely stored on several servers so that no data is lost in case of a single server failure. Due to the distribution of data, it is possible to access the data quickly, which is especially important with data-intensive processing tasks.

In Figure 4, the HDFS architecture is shown. Each server packs a part of the HDFS, while also providing a part of the cluster’s processing power.
The data we acquired was originally stored in a CSV file, meaning all data, even numerical values, was stored as text. We realised that it would save our application a lot of time if the data was stored in a more compact format. After all, regardless of whether we stored our data using Hadoop, it would obviously make sense to look for a way of storing our data that both takes up little space and can be processed swiftly.

When looking for ways to serialise our data, we came across several options. Google Protocol Buffers\(^1\) and Apache Thrift\(^2\) are two examples. Twitter has released an open source library, Elephant Bird\(^3\), for using Protocol Buffers or Thrift with Hadoop. Another option we found was MessagePack\(^4\). We felt that the documentation on the MessagePack website was rather unclear, and that combining it with Hadoop would also require us to add another technology, Hive, to our implementation. This led us to quickly dismiss MessagePack for our project.

Finally, there is the Avro\(^5\) framework, supported by the Apache Software Foundation. Avro can generate data structures and associated classes, and stores data in a binary format. It is also easy to use with Hadoop. The fact that it fit so well with our choices to use Java and Hadoop made us decide to implement Avro in our application.

Avro has several advantages compared to Protocol Buffers and Thrift. For example Avro has a feature called projection, which we also used in our implementation. Projection allows for parsing only a subset of fields of a schema without selecting a whole set. This reduces processing time and also saves developers from having to program additional object types.

For example, consider an object consisting of 10 data fields. Imagine only 2 data fields are needed and it is exactly known which data fields exist. Without projection, the full 10 data fields are read and parsed. With projection enabled only 2 data fields are read, reducing the amount of data fields which are read. This will impose a reduction in the time needed for reading.

We benchmarked Avro’s performance using two data files, each representing the same part of our complete dataset. One file was a regular CSV formatted file, the other used the Avro serialised data.

---

\(^1\)https://developers.google.com/protocol-buffers/
\(^2\)http://thrift.apache.org/
\(^3\)https://github.com/kevinweil/elephant-bird
\(^4\)http://msgpack.org/
\(^5\)http://avro.apache.org/
5.2 Data Processing

The size comparison can be seen in Figure 5. From this image, we can conclude that Avro provides a significant reduction in file size; over 40%. The time needed to read all data from file and build Java objects has an even bigger reduction in Avro. Figure 6 shows a reduction in time of over 70% can be achieved.

A reduction in space is achieved by storing information more efficiently than CSV (plain text). In figure 7, inspired by M. Kleppman, a storage block is showed that stores the text 'trading' and the number '1337' in one Avro record, in just 11 bytes. Compared to CSV, the number 1337 would be stored as text, so taking 4 bytes in stead of the three here.

Figure 7: An example of storing the word trading and the number 1337 in Avro.

Table 3: Comparison of several techniques to store data.

<table>
<thead>
<tr>
<th></th>
<th>CSV</th>
<th>Avro</th>
<th>Thrift</th>
<th>Protocol Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storing numbers efficiently</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Hadoop support</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Projection feature</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

5.3 Strategy Simulation

5.3.1 Strategy Definition

Several commercial vendors in financial software have created their own domain-specific language (DSL) for definition of trading strategies. One of the most well-known languages is the commercial EasyLanguage DSL, created by TradeStation.

Such a DSL allows the end-user to define a strategy without extensive programming knowledge. We will look at several financial DSLs that are currently in use and model our definition format accordingly, e.g. by using industry common terms to describe the strategy as to make the user feel at ease. However, because of the limited scope of our project, we will not be including a fully fledged DSL in our project.

Instead, strategies are defined as a Java class implementing a specific strategy interface (see Listing 5). When placed in the correct location, this class is automatically discovered and instantiated by the application. One of the advantages of this approach is that regular Java libraries and tools can be used to write (possibly very complex) strategies.

### Listing 5: The strategy interface.

```java
public interface Strategy {
    @NotNull
    LinkedHashMap<String, Float> parameterDescriptionsAndDefaults();

    void initialise(float... parameters);

    @Nullable
    Trade process(SpreadTick tick);
}
```

In order to make an implementation of a strategy a valid strategy, the following methods have to be implemented.

- **parameterDescriptionsAndDefaults** - This method returns a map which contains the parameters needed for the strategy. The entries in the map are \(<\text{description}, \text{default}>\) pairs, in which the description is the descriptive string the user sees when launching the strategy, and the default field is a float used as a default parameter, which the user can tweak when launching a new simulation.

### Listing 6: Example of the parameterDescriptionsAndDefaults method for a double moving average strategy.

```java
@NotNull
@Override
public LinkedHashMap<String, Float> parameterDescriptionsAndDefaults() {
    final LinkedHashMap<String, Float> descriptions = new LinkedHashMap<String, Float>(2);
    descriptions.put("Small window size", DEFAULT_SMALL);
    descriptions.put("Big window size", DEFAULT_BIG);
    return descriptions;
}
```

5.3 Strategy Simulation

- **initialise** - As the name suggests, this method initialises the objects needed in order to run the strategy, while taking the tweaked parameters into account. For an example, please refer to the implementation of the initialisation of double moving average strategy, in which the two parameters defined in `parameterDescriptionAndDefaults` are used to initialise the size of the buffers.

  **Listing 7**: Example of the `initialise` method for a double moving average strategy.

  ```java
  @Override
  public void initialise(final float... params) {
    super.initialise(params);
    size1 = (int) params[0];
    size2 = (int) params[1];
    assert size1 < size2;
    buffer1 = new LongBuffer(size1);
    buffer2 = new LongBuffer(size2);
  }
  ```

- **process** - Since this method contains all the trading logic, it is the core of the strategy. It receives all the spread ticks in the dataset sequentially, and produces a buy or sell signal based on the implemented trade logic. In the following Listing, the process method as designed for the double moving average strategy is provided as an example.
5.3 Strategy Simulation

Listing 8: Example of the process method for a double moving average strategy.

```java
@Nullable
@Override
public Trade process(@NotNull final SpreadTick tick) {
    // Add average price to the buffers
    buffer1.add(tick.getBestAskPrice() + tick.getBestBidPrice());
    buffer2.add(tick.getBestAskPrice() + tick.getBestBidPrice());

    // Get the average price of the buffers.
    final long avg1 = buffer1.getSum() / size1;
    final long avg2 = buffer2.getSum() / size2;

    return goldenCrossCheck(tick, avg1, avg2);
}

@Nullable
private Trade goldenCrossCheck(final SpreadTick tick, final long avg1, final long avg2) {
    if (avg1 < avg2) {
        if (goldenCross) {
            goldenCross = false;
            return new Trade(SpreadTick.newBuilder(tick).build(), TradeType.BUY, 1);
        }
    } else if (avg1 > avg2) {
        if (!goldenCross) {
            goldenCross = true;
            return new Trade(SpreadTick.newBuilder(tick).build(), TradeType.SELL, 1);
        }
    }
    return null;
}
```

5.3.2 Market Simulation Accuracy

A trading strategy benchmark package should simulate real market behaviour as realistically as possible, in order to represent probable strategy behaviour in the real market. In this section we will elaborate on choices made to obtain an acceptable level of market accuracy, and the implications these choices include.

- **Order book reconstruction** - The perfect trade strategy benchmark system would continuously reconstruct the order book based on incoming data and trade. However, due to the limitations of our dataset, which contains level 1 order book information (see Section A), this option quickly became impossible.

  Not having a reconstructed order book implies that strategies can only fire market orders, since limit orders, market-to-limit orders, etc. would rely on the order book for correct execution.

  For further reading on the subject of order book reconstruction, we suggest “LOBSTER: Limit Order Book Reconstruction System” by R. Huang and T. Polak [10].

- **Immediate trade execution** - A more plausible solution would be to trade on the currently
quoted price, meaning the strategy would be able to buy the amount of shares it wants to buy for the price it last saw. However, this would imply that this strategy, or the company owning the strategy, would have won the race to zero. Since this is practically never the case in reality, this option implies a skewed representation of market behaviour, which should be avoided at all times. This method had at first been implemented, since the team did not realise the effects of the assumption of immediate trade execution until Mr. Belmekki pointed out the issue (see Section 5 on page 54).

- **Data slippage** - As suggested by Mr. Belmekki, adding a form of data slippage would reduce the gap between a real market’s behaviour and that of our simulation. By adding data slippage, a strategy no longer receives the shares at the price quoted in the tick, but at the price of one of the following ticks. By randomly selecting the amount of slippage from within a range specified by the user, more realistic market behaviour is obtained, since the race to zero is no longer ignored. The pseudocode in Listing 9 outlines how we have implemented data slippage in our system. In short, every trade that is generated from tick A also receives a randomized slippage number N, and then has to wait until the system processes the Nth tick after tick A, at which point it is executed.

### Listing 9: Pseudocode representation of data slippage

```java
class Slippage
    list<Trade,counter>

    function processTick(tick):
        for <trade,counter> : list do
            if counter == 1 :
                trade.tick = tick
                context.write(trade)
            else :
                counter--

    function processTrade(trade):
        slippage = (int) expectation + sigma*RandomNumber
        list += <trade, slippage>
```

### 5.4 Used frameworks and technologies

#### 5.4.1 Spring Framework

As the basis of our application, we wanted to use a framework that speeds up development by providing implementations of often used software design principles and patterns, such as dependency injection. The use of proper design patterns is a requirement to ensure a high level of code quality, achieved through loose coupling and separation of concerns (qualified by SIG in the module coupling and unit interfacing measures).

We decided to use the Spring Framework[^1], which is an open source application framework and inversion of control container for Java. By using inversion of control (IoC), it moves application instantiation to a separate component that uses an external object graph instead of relying on statically assigned object relations within the application code. In this way, binding objects is achieved through dependency injection, a common software design pattern that allows removing hard-coded dependencies. This pattern leads to decoupling of objects within the application and allows easier testing and better separation of concerns [13].

An example of this is shown in Listing 10, where an instance of TradingbenchmarkService is injected into the class’ field using Spring, by marking it with an @Autowired annotation. If TradingbenchmarkService (i.e. the type of the field) were to be an interface rather than an concrete class, Spring will find a suitable implementation of the interface to inject. This means that code using the functionality of the interface (ServiceConsumer in this example) will never be coupled to the implementation, but only tied to the contract as specified in the interface.

**Listing 10:** Example of Spring dependency injection

```java
public class ServiceConsumer {
    @Autowired
    TradingbenchmarkService tradingbenchmarkService;

    public void calculate() {
        tradingbenchmarkService.someCalculation();
    }
}
```

IoC is not our only reason for choosing the Spring Framework, however. It also provides us with several useful extensions, such as the Spring Integration extension which provides support for Enterprise Integration Patterns (EIPs). While we will not directly be using EIPs because of the strong link between our client and server components, Spring Integration can still help us tie together these components through its remoting components, more specifically its Java Remote Method Invocation (RMI) service exporter and proxy components. With these components, linking the server and client using a common service interface is just a matter of a few lines of configuration.

Furthermore, we are using the Spring for Apache Hadoop extension, which simplifies our development process regarding Hadoop. By providing a unified configuration model and related APIs, we can create and manage Hadoop jobs using the same software paradigms utilised in the rest of the application.

Finally, we use the Spring Batch framework which, combined with functionality from the Spring for Apache Hadoop extension, allows us to execute several stages of a task sequentially without requiring substantial amounts of code. We make use of this to prepare the Hadoop environment by removing any existing output folders before automatically continuing with execution of the Hadoop job.

### 5.4.2 Remote Method Invocation

For the client–server communication, Java Remote Method Invocation (RMI) is used. This communication protocol has the ability to directly get Java objects from a remote location. In Figure 8, a graphical illustration is given of the RMI principle. The client uses a stub of the object to receive, whereas the server uses the actual implementation of the object. The protocol uses the lower network layers to handle the transport. In our implementation the connection is routed over a SSH tunnel, due to the restricted (firewalled) access of the DAS4.

### 5.4.3 JavaFX

When it came to picking a GUI framework, we had several options. Because the GUI only displays the results from the simulation, and does not have any other complex features, we knew we needed to find a solution to build our GUI in little time, while producing a user interface that is intuitive, responsive

http://www.springsource.org/spring-batch/
and aesthetically pleasing. Things that could cut down in a language we are already experienced in or that takes little effort to learn

Having worked with Java Swing and AWT before, we knew we would be much better served by finding a way to build a GUI that was more intuitive, offered more features and was simply less frustrating than either of them. In our initial interviews with our prospective users (see Section B), we were also asked to prioritise constructing a GUI that looked better than a 'standard' Swing or AWT application. One option would have been using a web-based interface, using (for example) HTML5, CSS and JavaScript.

A comparison of these methods can be found in Table 4. It is clear that a web-based or JavaFX interface would be a better choice than standard AWT or Swing, but the differences between those first two do not, at first glance, appear to be all that large. Though JavaFX does not necessarily provide a better-looking result than HTML and JavaScript, we chose to use JavaFX based on the following reasons:

- **Easy Integration** - Because JavaFX is included with Java 7, it is easy to integrate it into most Java applications. We also found several tutorials for using JavaFX combined with Spring dependency injection online, which made implementing this more straightforward.

- **Ease of Use** - Because most group members have more Java than web development experience, we expected JavaFX to be easier to learn. JavaFX also comes with a useful ‘Scene Builder’ tool that allows developers to construct GUI pages by simply dragging and dropping GUI components.

- **Features and Customisability** - The Scene Builder generates FXML files, which can then be customised freely. JavaFX also offers expansive charting features, which we knew we would need to show our results. Furthermore, GUI components (including charts) can also be styled using CSS.

<table>
<thead>
<tr>
<th>Ease of Integration with Java</th>
<th>Swing/AWT</th>
<th>HTML/JavaScript</th>
<th>JavaFX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiarity with Language</td>
<td>high</td>
<td>medium</td>
<td>high</td>
</tr>
<tr>
<td>Programming Effort/Difficulty</td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>“Looks Good”</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 4: Comparison of several GUI-building options.
JavaFX offers a straightforward way of creating a GUI and linking it to application logic. After making an FXML file, be it by hand or using the Scene Builder, the components can be linked to the application by way of fx:ids. This might look as follows:

**Listing 11: Example of an FXML file**

```xml
<fx:root type="javafx.scene.layout.VBox" xmlns:fx="http://javafx.com/fxml">
  <TextField fx:id="textField"/>
  <Button text="Click Me" onAction="#doSomething"/>
</fx:root>
```

Referencing these fx:ids is done by giving the component in the Java code the same name as the fx:ids, as well as using the @FXML annotation.

Linking the textField component and doSomething function would look as follows:

**Listing 12: Example of linking FXML and Java**

```java
public class CustomControl extends VBox {
  @FXML private TextField textField;

  // Constructor, getters and setters here

  @FXML
  protected void doSomething() {
    System.out.println("The button was clicked!");
  }
}
```

We found using JavaFX to build our GUI to be very intuitive, and did not encounter any significant obstacles in this area during development.
6 Quality Assurance and Testing

This section discusses the different types of testing methodologies that were used for testing both the client and server modules. The tests are globally divided into three categories: **automated tests**, **manual tests** and **user acceptance tests**. In automated tests individual methods, multiple methods and classes are tested, while manual testing tests the system as a whole, which includes, but is not limited to GUI-consistency checks, spelling accuracy checks, etcetera. In user acceptance tests, potential users get a short demo of the project, get to experiment with the software, and can provide feedback on the software. In this section, we will also describe the third party testing tools used during the project.

Finally, we will discuss the code analysis results from SIG (Software Improvement Group), and how we took their feedback and suggestions into account.

6.1 Methodology

Code quality can be improved greatly by programmers not only writing decent code, but also reviewing the code of others. The choice for Github as our version control system was easily made, since it provides the feature of making pull requests before merging into the master branch.

The process of coding is visualised in figure 9. The first step for implementing a new feature is forking the repository into a personal repository (1), checking it out locally (2) and making a new branch (3). The programmer can then write code (4). When his feature is complete he can rebase the branch (5), push the feature branch to his fork (6) and make a pull request (7). These pull-requests give the developers the option to increase code quality by commenting on new code before it is added to the ‘final’ product. This way, at least one other person other than the programmer has seen each line of code produced, which reduces the chance of poor quality code ending up in the final product. If a pull request is not of adequate quality, steps 4-7 are repeated until the pull request is considered satisfactory.

We have used the following testing constructs to test our system to a level of proper adequacy:

- **Unit Testing**: This is the form of testing we used most during the project. It focuses on testing small, individual parts of the system with predictable behaviour, to ensure they are functioning correctly.

- **Scenario-Based Testing**: Testing the entire system as a based on use scenarios. In our case, we have tested scenarios using both the client and the server modules, and the "double-moving-average" strategy as described in Appendix 4.2.

![Figure 9: The workflow of programming](image-url)
● **Integration Testing:** Testing the software components with other software components (e.g. third party software). In our case, we tested the interaction between the client and server modules.

● **Performance Testing:** We tested the change in performance of our system by gradually increasing the size of the number of input ticks, and keeping track of the difference in processing time for the resulting trades.

● **Usability Testing:** Throughout the project, we made sure to personally use the system and also asked our prospective users to go for a ‘test run’ of the system, to make sure the interface was useful and understandable.

Unit testing for the application was performed using JUnit, which is the standard framework for unit testing in Java. Especially data classes and exact methods which have a clear and predetermined input-output relation, such as the GlobalResult or MarketDataService, were fit for unit testing. However GUI component classes like the FXMDialog class have proven to be untestable using unit testing constructs, therefore these classes had to be tested manually. This is done by running the application, and checking if all the components are rendered correctly. An example of a JUnit test case is displayed in Listing 13.

In developing unit tests for the application, the built-in IntelliJ features for measuring and reporting Java code coverage were used to assess the coverage achieved through automated testing. Though the goal is to achieve ‘high’ coverage coverage, there must be a balance between effort and achieved coverage, as stated on the official JUnit webpage:

“While it is usually better to test more, there is a definite curve of diminishing returns on test effort versus code coverage.”

Listing 13: Example of a JUnit test

```java
@Test
public void testSetListViewValues() throws Exception {
    ListView<String> listView = new ListView<String>();
    ReflectionTestUtils.setField(exposureController, "listView", listView);
    ArrayList<String> values = new ArrayList<String>();
    values.add(0, "Test1");
    values.add(1, "Test2");
    exposureController.setListViewValues(values);
    Assert.assertEquals(compare, listView.getItems());
}
```

The above is an example of a JUnit test that checks if the values in a ListView in the Exposure tab of the GUI are set properly. A simple list of values is made, then added to the ListView. Using reflection, the exposure tab’s ListView attribute can be accessed from this test, even though it is a private field. To determine whether the test was successful, JUnit checks whether the ListView matches the `compare` array, which was defined in the test class’ setup method.

### 6.2 Process

During development of our application, we noticed that GUI components would be difficult to test as long as the back-end of the feature was not complete. Keeping this in mind, we decided that until most of our features were functional, we would only test GUI-elements manually by running
the application and checking if all components were in place. The back-end of the application (the 'calculation layer'), on the other hand, was a perfect fit for unit testing.

Whenever a feature was added to the calculation layer, unit tests were written to ensure correctness of this layer. Mathematical mistakes are undesirable in any project, but should be tested for even more thoroughly in applications like the one developed during this project, where an incorrect calculation in the application could have serious repercussions in reality. The lead tester ensured that these tests were added before a pull request of this feature could be made. These pull requests were a great help to improve code quality and testing frequency, because they have to:

- Be checked by at least one other member of the team - an extra pair of eyes can spot mistakes or suggest improvements.
- Pass all developed tests - this ensures new functionality does not 'break' older functionality.

On 12 June 2013, our code was sent to the Software Improvement Group for a first review. The code sent for review was feature complete, but did not include integration-tests, or tests of GUI-elements yet. The code was sent to SIG three days before the official deadline, because we assumed this would lead to receiving feedback sooner, ensuring more time to process the suggestions made. However, this assumption was a mistake on our behalf.

While waiting for the feedback, we used the extra time to add integration and scenario tests to the project, since we thought this would surely be a point of criticism in the review. It was a surprise to see that this remark was absent when we received the feedback on 21 June 2013: we were in fact given a compliment for performing unit testing in the first place.

After we had received the SIG review, we processed the feedback they provided by adopting their suggestions and fixing the faults the review pointed out. There were two main points of criticism in the review:

1. **Component Balance** - The amount of code in the 'client', 'server' and 'shared' packages was noted to be imbalanced: the review noted there was a disproportionally large amount of code in the 'client' package. We agreed with this feedback, and moved all functionality not related to client behaviour, which mostly comprised the 'calculation layer' of the application, out of 'client' and into its own package.

2. **Unit Interfacing** - A number of functions was found to have a large amount of parameters. We easily identified these functions using IntelliJ, and then changed these functions accordingly to reduce their complexity.

Since the points of improvement suggested by this review took relatively little time to fix, we used the remaining time to continue building tests for the application in order to improve code coverage, which was still low in certain areas of the application.

### 6.3 Results of code quality and testing

The continuous integration platform Jenkins tested our program on a regular basis over the course of the project. Figure 14 shows the number of passing tests (green). Figure 1 shows the number of tests passing (green) and failing (red) for each pull request. The number of tests varies due to the fact that some pull requests contain a fewer number of tests.

Our global line coverage for the entire project is 68%. Figure 12 shows the coverage percentage of the tests. The coverage in the shared package is relatively low, due to the fact that the shared package contains a substantial amount of auto-generated code.
6.3 Results of code quality and testing

Figure 10: Number of tests in master

Figure 11: Number of tests in pull request. Failing tests (red) are almost always intermediate builds, which are not merged to master until the errors have been fixed.
6.3 Results of code quality and testing

Test coverage

Class coverage
Method coverage
Line coverage

percentage

client metrics server shared

Figure 12: Coverage of tests among different packages
7 Final Product

7.1 Strategy upload

The user is able to upload their strategy, by following the steps below.

1. Implement a strategy by extending our strategy interface.
2. Package the strategy in "strategies.jar".
3. Place the jar in the folder where the application runs.
4. Run the 'start server' button in the next section. This will automatically upload and use the provided jar file.

7.2 Setup

A setup dialog appears when running the application, as shown in Figure 13. Here, a connection to the DAS4 server can be made by pressing the “Start server ...” button, which will open an SSH tunnel to the DAS4 cluster. If the user has provided a jar file, the system will automatically upload and add it to the list of available strategies.

The area outlined in blue displays progress messages after the connection has been opened. If a connection is already present, this step can be skipped using the “Skip” button.
7.3 Benchmark launch

![Image of a user interface for selecting a strategy and slippage policy]

**Figure 14:** Selecting the strategy and slippage policy.
7.3 Benchmark launch

**Figure 15:** Selecting the strategy from the drop-down box.

**Figure 16:** Selecting the slippage policy from the drop-down box.
Before running a simulation, a strategy and slippage policy have to be selected, and parameters have to be set for both.

The user is able to select their strategy in the strategy selection dialog, as seen in Figure 14. The top drop-down box shows the available strategies, as shown in Figure 15. The second drop-down box shows the available slippage policies, which can be seen in Figure 16.

When both the strategy and the slippage policy have been set, the parameters for the strategy and the slippage policy can be entered. Figure 17 shows an example of entering parameters. Finally, after the parameters have been entered, the simulation can be started with the "Run Strategy" button.
7.4 Strategy result overview

Summary tab

The first thing the user sees after running a strategy is the summary tab, shown in Figure 18. This tab displays a detailed summary of the other tabs. This tab displays results concerning equity, exposure, and trades.

From this tab, the user can navigate to the other tabs, using the buttons at the top of the window.

Figure 18: The summary tab.
Market tab

Figure 19: The market tab.

The market tab initially shows all data on the server for the DAX30 Performance Index, comprising the 30 stocks of the DAX30.

The tab can also be used to generate more specific charts: the user can select the DAX30 index or a stock individually and plot the price path of that asset for a specified time interval.
7.4 Strategy result overview

**Equity tab**

![Image of equity tab]

**Figure 20:** The equity tab.

The equity tab in Figure 20 displays the amount of equity during the runtime of the strategy. A chart displays the equity in Euros over a course of time. A table on the right side of the screen displays the standard deviation, low, high, start, average and variance of the equity.
7.4 Strategy result overview

Exposure tab

Figure 21: The exposure tab.

The exposure tab in Figure 21 shows a chart with the exposure of all assets. A list on the left enables the user to select a different asset and show the exposure of that asset. On the right side, a table displays the high, average, and low values of the exposure.

Exposure is displayed as a fraction of the value in stocks over the total cash.
7.4 Strategy result overview

Weeks tab/Months tab

Figure 22: The months tab.

The months and weeks tabs display the strategy’s performance over a period of time. Figure 22 shows an example of the results for March 2012. Per month or week, the tab displays a table and a chart, the chart displaying the course of the total equity, and the table containing a selection of metrics derived from the data of that month or week.

The initial equity denotes the amount of equity at the beginning of that month or week, whereas the final equity denotes the amount of equity at the end of that month or week. Losers are defined as trades which don’t turn a profit, as opposed to winning trades, which do generate profit.
7.4 Strategy result overview

All trades tab

The all trades tab, shown in Figure 23, displays all trades that have been generated by the strategy. The first column displays the point in time at which the trade was made, and the second column shows which asset was traded. Whether the stock was bought or sold is then denoted by a "SELL" or "BUY" flag, and the volume of the order is listed in the size column. The price denotes the best ask or bid price for respectively a buy or sell order.

![Figure 23: The all trades tab.](image)
8 Conclusions

Over the course of the past ten weeks, we have constructed a Java application that accepts user-defined strategies, tests their performance over a year’s worth of DAX30 data, and graphically presents the results of the simulation to the user. In this chapter, we present our final remarks on how we feel our product has turned out, and offer suggestions on what direction, in our opinion, future development on this or related software packages should take.

8.1 Final Product

Looking back on the list of requirements we put together on at the start of the project, as outlined in Section [2.2], we can conclude that all of the ‘must-have’ requirements, all but one of the ‘should-have’ requirements, and all but two of the ‘would-have’ requirements have been met. The final application accepts a strategy along with parameters defined by the users, processes it in a few minutes while showing its progress to the user, and displays metric-based information about the results of the simulation in an intuitive user interface.

As stated above, there is one ‘Should-have’ feature that did not make the cut: we did not implement algorithmic optimisation of the simulation result. The reason for this is simple: in order to build a robust, useful application in such a short amount of time, we could not afford to include this extensive feature. We had expected from the start of the project that this feature would be difficult to implement, which is why we mapped out a development schedule which included this feature as well as one that did not. After work on the project had started, we quickly realised that wanting to include this feature had indeed been too ambitious, and that we would not able to implement it in such a short amount of time. Not including optimisation has, in our opinion, allowed us to build an application that is more reliable and user-friendly than it otherwise would have been, and gave us time to focus on two often neglected aspects of software development: code quality and testing. In short, we believe this choice has greatly improved the application’s overall quality potential for further development.

There also were two ‘Could-have’ features which have never been implemented, one of them being the feature which enables the user to customise the timeframe, and set of stocks to trade on in order to reduce the time needed for a simulation to finish. When approaching the end of the project, we did not believe this feature still to be valid, as having a calculation speed of a few minutes diminished the need for customising the simulation. The other ‘Could-have’ feature that was not implemented was connection status visibility. Since the connection with DAS4 is only used while simulating, and never re-used, this feature did not turn out to provide a useful addition to the system’s functionality.

8.2 Future Work

Though the application as delivered is fully functional, it could always be expanded with more features. Our suggestions for features to add to the application in the future are as follows:

- **Automated Algorithmic Optimisation** - A user should be able to receive suggestions on how to achieve better results with their strategy from the application. We think this feature would add a lot to the user experience if it were included.

- **‘Would like’ Features** - The features listed as ‘would like’ in our original feature list, that is, an integrated strategy designer and the option to generate a report explaining the results of the simulation in detail.

- **More Result Measures** - Depending on what potential users are interested in, additional information about the result could be added to the application’s GUI. Examples would be VaR (Value at Risk) or showing weekly/monthly results per ISIN.
8.2 Future Work

- **Calculation Optimisation** - The program we delivered certainly is by no means slow, but could be made faster and more memory efficient.

- **Better Strategy Framework** - Users currently have to program and package their own strategies, and are only able to use Java. Providing support for other programming languages and making strategy uploading a more streamlined experience would be

- **Multi-Stock Strategies** - The application currently only allows strategies that trade single stocks. Expanding this to strategies trade multiple stocks would greatly enhance the application’s usability.

It should not be too difficult to extend the application as is to include most of these features, but some should prove quite challenging. Especially interesting is ‘multi-stock strategies’: parallelising a single stock strategy is doable, as we have shown, but a strategy that cross-references past behaviour of multiple stocks would be a far greater challenge to implement. Therefore, we recommend the further investigation of parallelisation as it relates to multiple cross-referential processes, which would undoubtedly also prove useful in many areas other than finance.
9 Evaluation

This bachelor project has allowed us to experience a full-fledged software development process from beginning to end, from product design to final product evaluation. In this chapter, we will reflect on our own performance before, and during this project, and analyse what could be improved in any future projects.

9.1 Preparation

The preparation for our project started rather early, since we wrote our own proposal to begin with. We also completed a large literature study early on in the form of the bachelor seminar, which was helpful to explore the state of the art in trading optimisation and the complexities involved. We also acknowledged the challenge in acquiring sufficient market data early on. Without market data, the product would have to be based on sample data, without the ability to properly test the system. However, a large amount of market data was finally delivered just a few days before the project, for which we thank Deutsche Börse.

9.2 During the Project

In the following sections we reflect on our own performance during this project, and analyse what could be improved in any future projects. We discuss both the project flow and the workflow as a team, respectively.

9.2.1 Working on the Product

From the start of the execution phase of the project, we worked full-time in our own room. Working in a single room allowed us to communicate effectively and also keep track of what anyone was working on at any time. This is a huge support to the Scrum workflow that we followed. It also meant that daily meetings and weekly reviews were all part of our regular workflow.

While day-to-day planning went well in general because of our constant interaction, we (in hindsight) did not always work in a clear path towards the final product. Many times, pieces of code were written while they were not the most important part at that moment. This meant that delays were introduced when the part that was not yet being worked on was actually a dependency for more major work. Had we re-prioritised work earlier, these stalls could have been avoided. In the future, there should be one person in the team appointed to keep an eye on what is actually important in each stage of the project.

Furthermore, the lack of clear milestones meant that important features were not finished when they were not a dependency for other parts. This also left out a clear motivator in many projects: deadlines. Without deadlines, some pieces of code were finished far later than than they could and should have been.

Finally, we think the project would have benefited from more frequent interaction with (prospective) end users in the form of interviews with product evaluations. In that way, there would have been the potential of re-prioritising the feature set and better feedback on the product’s features (such as the user interface) as development progressed. While this would have meant changes in the planning, our Scrum methodology would have made it easy to adapt to this.
9.2.2 Working as a Team

It proved challenging at the start of the project to balance every team member’s skills in various areas, but we managed to adapt our planning and workflow to suit the differences. For example, the level of programming experience of the team member varied. By making sure every team member got up to speed with the choice of frameworks, we made sure that every team member was up to most programming tasks. However, we could have sped this up even more by assuring that each team member read appropriate introductory guides before diving into the actual codebase and learning by doing.

Furthermore, unequal experience in writing in English meant that sections of reports as well as comments in code were of varying quality. By making sure that work that was expected to contain more errors more thoroughly checked by other group members, we avoided introducing large differences in the quality of the English used in our code and final report, though this did of course take up more time.

To solve any problems along the way, we utilised the principle of the Scrum sprint retrospective: a small meeting that held after each sprint review. In these meetings, which we held every Friday afternoon, each team member was given the chance to say what, in his opinion, were notable positive and negative developments to have occurred that week, and was able to comment on the performance of the other team members. Listing a ‘negative’ was always followed by a discussion on how that situation could be improved. The approach of discussing annoyances and other (minor) complaints openly every week allowed us to solve these issues before they turned into serious problems.

9.3 Lessons Learned

To summarise what we would expect to improve upon in future projects, we provide a short list:

- Roles should be clarified to make sure there is a leader for each project aspect at all times.
- Clearly defined milestones (with deadlines) throughout the entire course of the project should expedite development.
- Frequent interviews with prospective users assure that the final product meets their expectations, even though this might cause changes in planning.
- Better preparation by, for example, reading more introductions to frameworks and tutorials before starting programming work.

Though there is always room for improvement, as described above, we feel that overall the project went well. We are proud of our results, and we are grateful for the opportunity to have been able to work together as a group, which was a challenging, educational, and, most of all, very enjoyable experience.
A Dataset information

The dataset used for this project consists of one year of orderbook level 1 data as it was published by CEF®Core over Xetra Classic®, starting 29 March 2012. Order book level 1 or top of the book datasets represents the highest bid and lowest ask in the order book at a given time. These are interesting because they signal the prevalent market and the bid and ask price needed to get an order fulfilled. The difference between the highest bid and lowest ask is called the spread.

Please refer to Section A on page 53 for a sample of the raw, used data.

Covered Securities

<table>
<thead>
<tr>
<th>Number</th>
<th>ISIN</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DE000A1EWWW0</td>
<td>ADIDAS AG</td>
</tr>
<tr>
<td>2</td>
<td>DE0008404005</td>
<td>ALLIANZ SA</td>
</tr>
<tr>
<td>3</td>
<td>DE000BASF111</td>
<td>BASF SE</td>
</tr>
<tr>
<td>4</td>
<td>DE000BAY0017</td>
<td>BAYER AG</td>
</tr>
<tr>
<td>5</td>
<td>DE0005200000</td>
<td>BEIERSDORF AG</td>
</tr>
<tr>
<td>6</td>
<td>DE0005190003</td>
<td>BMW AG</td>
</tr>
<tr>
<td>7</td>
<td>DE0008032004</td>
<td>COMMERZBANK AG</td>
</tr>
<tr>
<td>8</td>
<td>DE0005439004</td>
<td>CONTINENTAL AG</td>
</tr>
<tr>
<td>9</td>
<td>DE0007100000</td>
<td>DAIMLER AG</td>
</tr>
<tr>
<td>10</td>
<td>DE0005140008</td>
<td>DEUTSCHE BANK AG</td>
</tr>
<tr>
<td>11</td>
<td>DE0005810055</td>
<td>DEUTSCHE BOERSE AG</td>
</tr>
<tr>
<td>12</td>
<td>DE0005552004</td>
<td>DEUTSCHE POST AG</td>
</tr>
<tr>
<td>13</td>
<td>DE0005557508</td>
<td>DEUTSCHE TELEKOM AG</td>
</tr>
<tr>
<td>14</td>
<td>DE0005785802</td>
<td>FRESENIUS MC</td>
</tr>
<tr>
<td>15</td>
<td>DE0006047004</td>
<td>HEIDELBERGCEMENT</td>
</tr>
<tr>
<td>16</td>
<td>DE0006048432</td>
<td>HENKEL</td>
</tr>
<tr>
<td>17</td>
<td>DE0008027707</td>
<td>HYPO REAL ESTATE</td>
</tr>
<tr>
<td>18</td>
<td>DE0006231004</td>
<td>INFINEON</td>
</tr>
<tr>
<td>19</td>
<td>DE0007162000</td>
<td>K+S GROUP</td>
</tr>
<tr>
<td>20</td>
<td>DE0006483001</td>
<td>LINDE</td>
</tr>
<tr>
<td>21</td>
<td>DE0008232125</td>
<td>LUFTHansa</td>
</tr>
<tr>
<td>22</td>
<td>DE0005937007</td>
<td>MAN AG</td>
</tr>
<tr>
<td>23</td>
<td>DE0006599995</td>
<td>MERCK KGAA</td>
</tr>
<tr>
<td>24</td>
<td>DE0007257503</td>
<td>METRO</td>
</tr>
<tr>
<td>25</td>
<td>DE0008430026</td>
<td>MUNCHENER RUE</td>
</tr>
<tr>
<td>26</td>
<td>DE0007037129</td>
<td>RWE AG</td>
</tr>
<tr>
<td>27</td>
<td>DE0006202005</td>
<td>SALZGITTER</td>
</tr>
<tr>
<td>28</td>
<td>DE0007164600</td>
<td>SAP AG</td>
</tr>
<tr>
<td>29</td>
<td>DE0007236101</td>
<td>SIEMENS</td>
</tr>
<tr>
<td>30</td>
<td>DE0007500001</td>
<td>THYSSENKRUPP</td>
</tr>
</tbody>
</table>

Table 5: Securities covered in dataset
Data flags

As explained in section 4.1 different trading flags are needed in order to indicate the start of a new phase in the trading day.

ATF or Auction trade flag

A Intraday Auction  
B Between Auction  
C Continuous Trading  
E End-of-day Auction  
F Closing Auction  
H Halt  
N No Trading  
O Opening Auction  
P Pre Trading  
Q Quote Driven  
R Post Trading  
S Suspended  
T Call Message  
V Volatility interruption  
X XPREC or XFREEZE-phase for instruments with TM08  
Z End of trading

PT or Price Type

A new best ask price  
B new best bid price  
S New spread (new best bid price and new best ask price)  
P Trade Price

trade_type

' ' Normal trade  
X Best trade  
O Over the counter (OTC) trade
<table>
<thead>
<tr>
<th>DATE</th>
<th>ISIN</th>
<th>TIMESTAMP</th>
<th>ATF</th>
<th>PT</th>
<th>CURR</th>
<th>PRICE</th>
<th>SIZE</th>
<th>BEST BID</th>
<th>BID SIZE</th>
<th>BEST ASK</th>
<th>ASK SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:30.00</td>
<td>O</td>
<td>P</td>
<td>EUR</td>
<td>38,11</td>
<td>48131</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:30.30</td>
<td>C</td>
<td>S</td>
<td>EUR</td>
<td>0</td>
<td>5221</td>
<td>38,1</td>
<td>300</td>
<td>38,2</td>
<td>300</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:30.40</td>
<td>C</td>
<td>P</td>
<td>EUR</td>
<td>38,1</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:30.70</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>873</td>
<td>38,105</td>
<td>300</td>
<td>38,2</td>
<td>300</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:31.19</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>919</td>
<td>38,105</td>
<td>300</td>
<td>38,2</td>
<td>300</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:31.59</td>
<td>C</td>
<td>S</td>
<td>EUR</td>
<td>0</td>
<td>800</td>
<td>38,11</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:33.59</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>600</td>
<td>38,115</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:34.83</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>693</td>
<td>38,115</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:34.87</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>800</td>
<td>38,12</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:36.42</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>600</td>
<td>38,125</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:36.43</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>1146</td>
<td>38,125</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:40.17</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>434</td>
<td>38,135</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:42.86</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>354</td>
<td>38,135</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:42.86</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>38,135</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:43.08</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>354</td>
<td>38,135</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:44.97</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>970</td>
<td>38,14</td>
<td>400</td>
<td>38,2</td>
<td>400</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:48.22</td>
<td>C</td>
<td>A</td>
<td>EUR</td>
<td>0</td>
<td>970</td>
<td>38,145</td>
<td>180</td>
<td>38,195</td>
<td>180</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:48.48</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>800</td>
<td>38,155</td>
<td>180</td>
<td>38,195</td>
<td>180</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:49.13</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>46</td>
<td>38,16</td>
<td>180</td>
<td>38,195</td>
<td>180</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:58.48</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>270</td>
<td>38,165</td>
<td>180</td>
<td>38,195</td>
<td>180</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:00:59.46</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>270</td>
<td>38,165</td>
<td>180</td>
<td>38,195</td>
<td>180</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:00.08</td>
<td>C</td>
<td>P</td>
<td>EUR</td>
<td>38,195</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:00.28</td>
<td>C</td>
<td>P</td>
<td>EUR</td>
<td>38,19</td>
<td>445</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:00.37</td>
<td>C</td>
<td>S</td>
<td>EUR</td>
<td>0</td>
<td>456</td>
<td>38,165</td>
<td>279</td>
<td>38,195</td>
<td>279</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:02.46</td>
<td>C</td>
<td>B</td>
<td>EUR</td>
<td>0</td>
<td>800</td>
<td>38,17</td>
<td>279</td>
<td>38,195</td>
<td>279</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:03.27</td>
<td>C</td>
<td>A</td>
<td>EUR</td>
<td>0</td>
<td>800</td>
<td>38,17</td>
<td>2578</td>
<td>38,19</td>
<td>2578</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:03.35</td>
<td>C</td>
<td>A</td>
<td>EUR</td>
<td>0</td>
<td>800</td>
<td>38,17</td>
<td>2757</td>
<td>38,19</td>
<td>2757</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:04.13</td>
<td>C</td>
<td>P</td>
<td>EUR</td>
<td>38,17</td>
<td>179</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2012-03-29</td>
<td>DE0005140008</td>
<td>2012-03-29 09:01:04.15</td>
<td>C</td>
<td>P</td>
<td>EUR</td>
<td>38,17</td>
<td>621</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6: Sample of the raw dataset
B Interviews

B.1 Gunter Fluyt & Jan de Bondt

In the early stages of the project, we introduced the idea of applying backtesting to high frequency trading strategies using a high resolution dataset to Mr. Fluyt. As Global Head of Rates, Credits and Equity Derivatives for Developed Markets at ING, we consider Mr. Fluyt to be an expert in the field of trading. He also introduced us to Jan de Bondt, Managing Director Fixed Income Currencies and Commodities Sales at Société Générale.

Mr. Fluyt started by mentioning that he believes "the high-frequency game" is still too immature and that current testing strategies are insufficient. HFT is still moving at a higher pace than it can actually handle: therefore, trying to develop an improved testing and benchmarking system is a great step in order to push these systems to a higher level of maturity. However, in order to make "the quants" (quantitative analysts) actually want to use the system, it will need to be fast, since they don't have enough patience to let their ideas grow.

Mr. De Bondt thinks Mr. Fluyt is too pessimistic about high frequency trading, and the people developing these systems. He states that while more accurate testing is always a goal to be desired, testing is currently already being performed at a high rate. However, applied testing strategies are more often than not kept in-house, and therefore developing a transparent testing system should be encouraged, especially if it is developed independently: as an academic pursuit, for instance. Mr. Fluyt agrees and states that we should talk to some of their "tech guys". Mr. De Bondt proposes to schedule a meeting with Hamid Belmekki from Société Générale.

B.2 Hamid Belmekki

During the meeting with Mr. De Bondt and Mr. Fluyt, Mr. De Bondt suggested to talk to Hamid Belmekki, Managing Director Global Head of Flow Quantitative Group for Fixed Income, Credit, FX and Equity at Société Générale.

Before we met with Mr. Belmekki, he had already spoken to several of his quantitative researchers focused on High-Frequency Trading concerning our project. It should be noted that he told us in advance that there was not that much he would be able to tell us, since a great amount of information is subject to a non-disclosure agreement.

First, we discussed the usability of several trade measures, why they mattered and others don't, and how to interpret them. The results from this discussion are reflected in the feature list in Section 2.2. This relatively short discussion was quickly followed by an elaboration on strategy testing and approval protocols and systems. Société Générale’s testing protocol roughly corresponds to the following steps:

1. Development of the strategy
2. Strategy testing using an in-house developed benchmark system which uses the data of 100 securities going back 1 month.
3. Once the strategy can be considered feasible (he could not tell us what they consider feasible), it goes into “paper trading” for 3-5 months, which means the strategy is live, but can not execute any trades.
4. If this last session is consistent with previous test results, and still profitable, the strategy can go live once test results are approved by management.

The entire process to develop and approve a strategy takes approximately five to seven months: one month of development, followed by four to six months of testing and concluding with management’s
approval to go live. Mr. Belmekki stated that he could not explain much more without violating the
non-disclosure agreement, so he changed the subject by providing us with some facts about their
current testing system.

- The system is an in-house developed project
- It is not an integrated system
- It backtests 100 securities going back 1 month
- A test-session of a complex strategy takes about 45 minutes if it runs over the full data-set
- It is a possibility to test shorter periods and/or less securities

This last feature is his personal favourite, since this enables the quants to tweak a strategy under
development and quickly see preliminary results. When we ask about what feature would be a big
improvement and make his life easier he doubts and replies: “A testing system should be more like
an Italian sports car: really fast but also sexy”. He claims that within the industry of financial software,
everyone always forgets to have a properly designed system, which would make working with these
complex systems on a day-to-day basis more comfortable.

B.3 Jasper Anderluh

We prepared a number of questions for the interview with Mr. Anderluh:

- What kind of tools does HiQ Invest use in order to test the feasibility of strategies?
  
  We use a number of tools. We start by using statistical packages like R and Excel, in
  which we design and tweak our strategies. Once we are pleased with the outcome
  of these relatively small tests, we expose them to an in-house developed testing tool,
  in which we have replicated a real-time stock market. In this tool, the strategy can
  fire orders, which are processed by the “stock market” before receiving a callback
  from the replicated orderbook.

- What measures produced by your systems are, in your opinion, the most important?
  
  - Stability of the PNL
  - Profitability compared to the number of trades
  - Number of trades executed (more important than how many ticks where processed)
  - Whether the strategy helps diversify the portfolio and complements other strategies already
    used by the company

- What time-frame can be considered acceptable for backtesting?
  
  Different companies use different strategies, so different time-frames can be con-
  sidered acceptable, however in HiQ invest we have a system running that processes
  1 year of ticks in approximately 5 minutes.

- Would you consider 1 year of DAX30 individual ticks a well selected data set?
  
  Yes, whoever gave you that set gave you quite the gift. Such a set should help you
  simulate practically all common scenarios.

After the prepared questions, a more open conversation started, in which Mr. Anderluh gave us the
following main tips:

- A strategy does not have to close its positions every day or weekend, but real high-frequency
  systems probably will.
Don’t worry about the way the markets react to your trades, there is virtually no way of testing this interaction.

A system in which a strategy is simulated against a live orderbook which provides a callback should produce more accurate results.

Fun idea: The possibility of allowing multiple strategies to trade against a single orderbook thread.

B.4 Hamid Belmekki

Right after we sent the code to SIG (Software Improvement Group), we contacted some of the field experts that we had interviewed at the start of the project. By going back to back to Societe Generale’s Managing Director, Global Head of Flow Quantitative Group for Fixed Income, Credit, FX and Equity, Hamid Belmekki, we were able to get feedback on our code quality, product usability and overall design.

At first we gave Mr. Belmekki a short introduction and demo of the product, after which we let him discover the rest of the application on his own. Since we could not use the DAS4-setup on location, we used one month of data to process strategies locally.

After a few minutes, Mr. Belmekki made his first remark, stating that he’s happy to see that somebody finally took the effort of not making a standard Swing/AWT Java application, but instead something using more fancy graphic libraries. When we tell him it’s designed using JavaFX and styled using CSS, he’s amazed by the possibilities this combination generates, and the discussion veers slightly off topic, while he is discovering the rest of the application.

After Mr. Belmekki had used the application for a while, he summarized the improvements to be made:

- **STD and Variance** - These two measures are too important to be missing in a trading benchmark application regarding equity information, as they can show the user how volatile the user’s equity behaved during the simulation.

- **Single Stock Strategies Only** - Though the application is useful for single stock strategy benchmarking, it would be disappointing not to extend the application to other strategies (Mr. Anderluh made a similar remark).

- **Week/Month Overview Need ISIN-Specific Information** - The week and month overview panes are a great start for periodical strategy behaviour overview, but they need ISIN-specific results. For example, if a user sees something strange happened in August on stock X, they will want to know how the strategy behaved during that specific period, but they also want to be able to see how the strategy handled this specific stock during this event, which is currently impossible.

- **Graph Zoom** - For the same reasons stated above, the equity chart needs a zoom function. In such a way, the user can check equity behaviour during specific events on specific time-frames.

- **Customisation** - Once again, for similar reasons the user might want to simulate a specific time-frame and stock-set. However, implementing the features mentioned above (week/month overview with isin-specific information and graph zoom) would provide exactly the same results. In case both features are implemented, I would only use the custom simulation because I don’t like to wait for a simulation to complete, and adding less stocks and less time would reduce the time needed to produce results. However, if I can believe you only need four to five minutes to process DAX30 for a full year, I think you can force the user to go get a coffee, instead of sitting behind his desk impatiently.
• **Data slippage** - If a strategy wants to make a trade, there is no guarantee that it will be able to make the trade on that specific price. Instead, traders know how many ticks after they decide to buy they will usually be able to actually perform their trade. By adding data slippage, the results of the benchmark will not be exactly the same every time, and will more accurately represent a realistic situation.

After telling us what could be improved, Mr. Belmekki also listed the features he appreciated.

• **Calculation speed** - If it is true you only need a 4-5 minute time-frame needed to process Dax30 individually for a full year, that is amazing, and it would be a great improvement on our current software.

• **Interface** - The interface looks nice, is very comprehensible, and it is also easy to learn.

• **Equity curve** - The equity curve still is the most important feature. Even though quants keep saying they need to have a good risk analysis, and need to see what a strategy might do in the case of unfortunate events and the like, in the end, if the strategy does not make a profit, it will not be used. Therefore, you can’t go without a clear equity curve.

• **Measures** - The chosen measures are good, if the suggestions are implemented. Though they are pretty basic, for a first version of the application this is not bad at all.

• **Rerun strategy** - Even though the implementation of this feature is not 100% done yet, this is a big first step towards automated optimisation, which really should be the future vision of automated trading.

In closing, Mr. Belmekki’s expressed his opinion that what had been produced so far went far beyond what he would expect from a ten week project done by four undergraduate students.
C Plan of Approach

This appendix contains the original Plan of Approach as written in April 2013.

C.1 Introduction

C.1.1 Plan of Approach

In this document, we will outline the plan of approach for our bachelor project. First, we will give an outline of the background and contents of our assignment. Then, we will discuss how we mean to carry out our project, that is, what our general approach to working in a team will be, what methods and techniques we will be using, and how we will carry out quality assurance.

C.1.2 Company Outline

The Delft Institute of Applied Mathematics describes itself as follows on its website:

Mathematics and mathematical research play an indispensable role in every technical discipline. This may range from new numerical methods for fluid dynamics computations, real time forecasting systems for weather, water and transport of hazardous materials, stochastic models of large networks, financial mathematics, statistical models of production and transport systems, and risk and decision support modelling in conjunction with large technical systems. Innovative mathematical research addresses the ever changing needs of society directly in the form of contract research, and it also contributes indirectly by supporting advanced research in a variety of other technical disciplines.

In addition several commercial software systems are made available to clients and colleagues in areas including forecasting water levels, inverse modelling, and risk analysis.

The mission of the DIAM is the mathematical modelling of physical, technical and societal phenomena using advanced mathematical techniques and methods in an applications driven manner.

C.1.3 Background and Motivation of the Assignment

Throughout the entire history of the stock market, investors have always sought to get ahead of their competitors in some way, be it through conventional or creative, risk-free or risky, legal or even illegal means. Since the advent of the twenty-first century, financial markets have increasingly become populated with traders who rely on the potential of computing to improve their results. And while researchers currently disagree on whether this is a good or bad thing, there does seem to be a consensus regarding the fact that so far, the research done has only barely scratched the surface of the possibilities this field has to offer.

On the modern stock market, a large portion of trading has become automated. Investment banks, trading firms, pension funds and many other institutional traders use automated trading in the hopes of minimising risk and maximising profits. A specific kind of automated trading that has gained a lot of popularity recently is High Frequency Trading (HFT), a manner of trading that rapidly trades securities, trading in and out of positions in the blink of an eye. With computing power becoming ever cheaper and more readily available, competition among traders is no longer just about who has

http://math.ewi.tudelft.nl/
the best trading strategies: it is also a matter of who is the fastest in deciding when and how to trade.

Obviously, a human investor is not able to simultaneously make large quantities of decisions in the blink of an eye: for this reason, companies build automated trading systems that take instructions from humans and then perform ‘split-second trades’ according to pre-defined strategies. This way of trading is not without risks, however: take for example the 2010 ‘Flash Crash’, where the Dow Jones suddenly fell about 1000 points (approximately 9%), only to recover within a few minutes, which was attributed by many to the limitations of automated trading. Malfunctioning systems have also been known to cause traders massive losses. Therefore, it is crucial to not only build a system that is not only robust, but one that can devise strategies and predict the course of the market as effectively as is possible.

In order to accurately analyse the risk and predict the behaviour and profitability of a strategy we feel there is a growing need for systems to be able to benchmark these systems in a fast and reliable way.

### C.2 Assignment Description

#### C.2.1 Introduction

As mentioned in the introduction, the modernisation of the stock market and the development of new technologies has cleared the way for automated trading. A specific kind of automated trading that has gained a lot of popularity recently is High Frequency Trading, or HFT. The growing popularity of this ‘new kid on the block’ can be attributed to the new opportunities created by split-second trading, and the affordability of the required computing power.

Although these relatively new ways of trading have gained popularity within the financial markets, it is frowned upon by many. Where the proponents claim these systems provide liquidity in the market, the opponents claim these systems to be a danger to the intrinsic value of stock markets, because of the artificially created chain of supply and demand. The opponents also fear the limitations of HFT, of which flash-crashes are the perfect example. They believe the usage of HFT should be regulated, or even abolished.

#### C.2.2 Client

The assignment was written by one of our team members, which makes us, in a way, our own clients. We have asked Dr. Ir. J.H.M. Anderluh (Assistant Professor in Financial Mathematics, DIAM Probability and Statistics, and Director of Fund Management company HiQ Invest) to advise us on how the project can become a usable tool for the financial industry. Because of his domain expertise, Mr. Anderluh can provide us a deeper understanding of the field, but he will not act as the client.

Plausible clients are financial institutions that deal with high-frequency trading on a day to day basis. These institutions include, but are not limited to, hedge funds, proprietary trading firms and investment banks.

The target users are quantitative analysts who design, implement, improve and review high frequency trading strategies. Therefore we have asked Damien Petitjean (Lead quant for Societe Generale) to provide feedback on our implementation.
C.2 Assignment Description

C.2.3 Contact Information

Team

Friso Abcouwer
F.C.A.Abcouwer@student.tudelft.nl

Mark Janssen
Mark@ch.tudelft.nl

Xander Savenberg
X.Savenberg@student.tudelft.nl

Ruben Verboon
R.S.Verboon@student.tudelft.nl

Project supervisors

Dr. M. M. de Weerdt
M.M.deWeerdt@tudelft.nl

Dr. Ir. J.H.M. Anderluh
j.h.m.anderluh@tudelft.nl

Dr. phil. H.G. Gross
H.G.Gross@tudelft.nl

Mentor
Domain expert
Project supervisor

C.2.4 Problem Description

The development and testing of HFT strategies is a hard and complex task in the financial industry. Due to the continuous race to zero delay, these systems have to work in the most competitive surroundings, while performing at the highest speeds available. In order to have a faster or more reliable system running before the competitors, these systems do not just have to be developed quickly, but also need to be tested as fast and accurately as possible.

C.2.5 Goal

By providing quantitative analysts, also known as *quants*, with a tool that thoroughly benchmarks and tests a strategy, a financial institution can not only profit from the time gained by speeding up the testing process, but also from early error discovery and an increase in behaviour predictability, among other benefits. This provides the institution with higher quality strategies, and facilitates the search for better and faster strategies. Because of the higher predictability of a strategy's behaviour, malfunctions would be less likely, which should also improve the general image of HFT.

As described in section C.3.4, the decision as to whether the optimisation features will be implemented or not will be made on May 24th. If the implementation of these features is considered feasible, the improvements made by the optimisation process should be measured by the increase in performance as described below. Since the possibility of improving a strategy depends on the input, we feel that at this point we cannot give an estimation of how big the improvements have to be in order to consider this feature as accomplished yet, aside from the obvious fact that performance decreases are not acceptable in any case.

C.2.6 Assignment Formulation

During the course of this project we will develop a back-testing system for high frequency trading, in which a user can define a technical analysis-based trading strategy, and let the system analyse and optimise this strategy on a number of different measures. To ensure the resulting program becomes a usable tool to improve strategy quality and facilitate the search for better and faster strategies, the assignment has two main areas of focus:
Simulation The system statistically analyses user-defined intraday trading strategies based on technical analysis using historical values. Part of the research involved in this project is finding out what statistical measures can be used determine the validity and profitability of a strategy.

Optimisation Another important aspect of the assignment is optimisation. The system should be able to learn using multi-objective machine learning techniques by looking at historical values. The machine learning technique has to be multi-objective, since the profitability of a strategy is not the only measure that makes a strategy successful: researching what other measures there are and how they interact will be a part of our research as well.

We want to give the option to visualise these results for our full data range (which encompasses a year of the DAX 30), by Global Industry Classification Standard, or stock by stock individually. This option is crucial, because it allows checking whether a strategy works better in certain sectors of the market.

If the user is only interested in certain parts of the market, they should be able to disable the computation of the don’t care markets. Reducing the number of calculations in this way should allow the user to see their results more quickly.

In short, the list of features of the application is as follows. The application should be able to:

- Graphically represent historical data and the results of simulations
- Have users define and simulate their own strategies
- Provide users with an overview of how their strategy performed, based on performance measures such as risk, number of trader or profit
- Provide users with suggestions on how to improve their strategy (optional feature)

C.2.7 Deliverables

By the end of the project, we will deliver a report conforming to the Bachelor Project course guidelines, and a prototype software package that can complete the tasks as described in project assignment. There are multiple reasons why we will develop a prototype, rather than a fully developed and distribution-ready system:

- The time frame of ten to twelve weeks is too short to fully develop a system of this complexity.
- Bigger data sets are too expensive or too difficult to acquire.

C.2.8 Preconditions

We are building a statistical analyser for trading strategies. Accordingly, we will use the statistical axiom that states that the larger the number of exposure units independently exposed to loss becomes, the greater the probability that actual loss experience will equal expected loss experience will be. This is more commonly known as the law of large numbers, and states the credibility of the analysis increases with the size of the data pool under consideration.

C.2.9 Risks

The biggest risks of this project are as follows:

1. **Data acquisition proves too difficult.** The data needed for this project is financial market data. In a commercial project, licenses for tick-data can exceed several thousands of dollars
C.3 Approach

62

per month. Our challenge is to acquire data which is representable as HFT data, but does not cost us anything.

2. We underestimate the tasks ahead, or overestimate our knowledge. In other words: the assignment turns out to be too complex. This is a common pitfall in many software development projects, and there is no reason to simply assume there is no possibility of us experiencing this as well.

Luckily, we have already obtained the data we need. This leaves us with the option that the project will prove too challenging to complete on time. To mitigate this risk, we will decide on one of two ways to proceed after a first period of initial development: if we are optimistic about the project at that point, we will continue with our original plan; otherwise, we will limit our program to a still functional, but less ambitious design.

C.3 Approach

C.3.1 Introduction

In this section, we will outline the methods, techniques and tools we will use to develop our system. We will also give an indication on how development will typically proceed during the project, and give a general overview of our planning.

C.3.2 Methods and Techniques

We will be developing our application in Java 7 using an approach combining the Scrum development methodology and Test Driven Development practices. On top of Java we will be using the popular Spring application development framework for high performing, easily testable and reusable code.

Our toolset will consist of the following applications and technologies:

IntelliJ IDEA will be our Integrated Development Environment (IDE) facilitating development and testing of our code. IntelliJ IDEA comes with a wide set of code inspections and test coverage analysis that we will be using.

GitHub will be used to host our code using the Git version control system, and to review code.

Maven is the build automation tool that we will be using to manage our builds, their dependencies and (integration) testing of the project.

JUnit 4 is the testing framework of choice, which integrates seamlessly with IntelliJ IDEA and Maven.

Checkstyle shall be used to help adhere to coding standards.

Jenkins hosted on Cloudbees is a continuous integration framework guaranteeing successful builds and test execution.

Trello will be used for keeping track of our todo lists.

PangoScrum backs the physical Scrum board using overviews and an online backlog.

C.3.3 Proceedings

We will have a weekly sprint planning meeting every Monday morning to discuss what work will be done that week. Every morning after that we will have a daily standup, where every team member updates the others on his progress, announces his plans for that day, and where the team can discuss
potential problems or stumbling blocks. Every Friday afternoon we will have a sprint review meeting with Mr. De Weerdt, reviewing that week’s progress. After that, we will have a sprint retrospective meeting with the team, reflecting on that week’s sprint and finding ways to improve our process the following week.

C.3.4 Planning

We will be working in sprints with a length of 5 working days. Our week-by-week planning for the project is as follows, with the dates below indicating what our goal will be for the end of each week.

28 April End of Sprint One: Drafts of Plan of Approach and Orientation Report, setup of the development environment.


8 May End of Sprint Three: Basic trading engine that can ‘stream’ price ticks, and basic visualisation.

17 May End of Sprint Four.

24 May End of Sprint Five: Definitive decision on whether to include optimisation or not.

31 May End of Sprint Six.

7 June End of Sprint Seven: First draft versions of a majority of the final report.

14 June End of Sprint Eight: Feature-complete version of application finished, code sent to SIG for review.

21 June End of Sprint Nine: Last improvements, implementation of SIG feedback, final report and presentation nearing completion.

28 June End of Sprint Ten, end of development: code sent to SIG, finished final report and presentation.

3 July Final presentation.

Due to national holidays and TU Delft collective free days, Sprints Two and Three will only consist of three working days, and Sprint Five will only consist of four days. We will take this into account for our sprint planning, and believe that this will not pose much of a problem: because Sprints Two and Three are early on in the project, we will be able to compensate for any unfinished work in the weeks following them.

After week three, we will be deciding whether we want to consider taking on the path of benchmarking and optimisation, or if we will be pursuing a more extensive benchmarking application without any optimisation features. This choice will be decisive for the planning of the rest of the project.

C.4 Project Approach

C.4.1 Introduction

In this section, we will explain our approach to this project and how it relates to the stakeholders involved. For every stakeholder, we will outline the role(s) and responsibilities they will take on over the course of the project. We will also elaborate on how we will obtain and manage the information we need, and which facilities we will require.
C.4.2 Stakeholders

The table below contains an overview of the people involved and the roles they will take on during the project. In addition to being expected to work on reports, program code and tests, every group member has one or two specific responsibilities.

<table>
<thead>
<tr>
<th>Name</th>
<th>Roles</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friso Abcouwer</td>
<td>Scrum and Reports</td>
<td>Responsible for the Scrum process and editing and finalising reports.</td>
</tr>
<tr>
<td>Mark Janssen</td>
<td>Lead Development</td>
<td>In charge of the software development process.</td>
</tr>
<tr>
<td>Xander Savenberg</td>
<td>Project Manager</td>
<td>Manages product requirements and the project in general.</td>
</tr>
<tr>
<td>Ruben Verboon</td>
<td>Quality Assurance</td>
<td>In charge of ensuring high code quality and proper testing procedures.</td>
</tr>
<tr>
<td>Mathijs de Weerdt</td>
<td>Group Mentor</td>
<td>Supervises the group during the project.</td>
</tr>
<tr>
<td>Jasper Anderluh</td>
<td>Group Mentor</td>
<td>Domain expert. Will provide feedback from the perspective of the user.</td>
</tr>
<tr>
<td>Gerd Gross</td>
<td>Coordinator</td>
<td>Coordinator of the Bachelor Project course.</td>
</tr>
</tbody>
</table>

C.4.3 Information

Our primary sources of information on the problem domain are Jasper Anderluh and Xander Savenberg. Xander will also be the Product Owner in our Scrum team. Communication with Xander will occur mainly in person, as he is a member of the project group, and communication with Mr. Anderluh will take place mainly through e-mail. We will also have face-to-face meetings with him several times to get feedback on our product during the development process.

C.4.4 Facilities

We will be working in room HB07.230 of the Faculty of EEMCS at the TU Delft. Furthermore we have been granted access to the DAS-4 supercomputer, which we aim to use to process our data.

C.5 Quality Assurance

C.5.1 Introduction

In this section, we will outline our approach to quality assurance during this project. It is mostly inspired by industry standard approaches.

C.5.2 Quality

All team members will work on their own branch, and when they send a request to merge their finished work into the master branch, the other team members will receive a notification. The merge will not take place before at least one team member has reviewed the work and all issues have been resolved.

We believe that this measure will improve code quality and also allow us to work more efficiently, as everyone will have a better understanding of code written by others.
• **Documentation** - The system will be documented on code level and on system level. Code is documented to support maintainability and robustness, and the system is documented so that the user will have a point of reference when using it.

At the code level, we will be using Javadoc. We will document the purpose of each class and all non-trivial methods.

For the user, we will document our system in the form of a wiki specifying the workings of the system: this wiki will function as a manual of sorts. In the final sprint, a manual for new users will be composed from parts of the wiki.

• **Version Control** - For version control we will use Git, a system that is widely used throughout the industry. We will use host our repository on GitHub, a website that offers users space for Git repositories. Git has the advantage of being decentralised and providing seamless branching and merging, which offers tremendous benefits for collaboration.

• **Evaluation** - Code will be pushed when it is considered to be functioning properly, or at the very least not causing any failures in the system. Before making a commit to Git, the developer will test the part and submits the tests and the piece of code.

For implementing new features, we will use the branching feature of Git. A developer makes a new *feature branch*, i.e. a new branch exclusively for that feature. When the feature is done, the branch is published on Github. After review by one or more peer developers the feature branch will be merged into the master branch.

• **Pilots** - Pilots are made throughout the project, mainly in the latter part of the project. The pilots feature a functioning system. This system is reviewed by quantitative researchers. This feedback helps the program to be compliant to the demands of the users. Also this feedback helps us to prioritise unimplemented features, so that the most important features are implemented at the end of the project.
D Feedback Software Improvement Group [Dutch]

De code van het systeem scoort 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Component Balance en Unit Interfacing.

De codebase is opgedeeld in duidelijke componenten, met een indeling (client/server/shared) die eenvoudig te begrijpen is. Daarnaast is het goed om te zien dat jullie Maven gebruiken, dit maakt het makkelijker om externe tools op jullie code los te laten.

Bij Component Balance wordt er naar de verdeling van het codevolume over de componenten gekeken. Alle code in een heel groot component is probleematisch, maar duizend kleine componenten zonder duidelijke verantwoordelijkheden zijn dat ook. Jullie score voor deze meting valt wat lager uit omdat het merendeel van het codevolume in de client zit. Dit is niet direct "fout", maar het valt wel op dat er aardig wat logica in de client zit die je misschien eerder op de server zou verwachten (zie bijvoorbeeld ISINResult.java en MarketListing.java). Daarnaast bevat het shared-component nu erg weinig code, terwijl je zou verwachten dat het datamodel tussen client en server gedeeld wordt. We verwachten niet dat jullie met code gaan schuiven puur om deze score te verhogen, maar denk eens kritisch na welke functionaliteit waar hoort.

Voor Unit Interfacing wordt er gekeken naar het percentage code in units met een bovengemiddeld aantal parameters. Doorgaans duidt een bovengemiddeld aantal parameters op een gebrek aan abstractie. Daarnaast leidt een groot aantal parameters nogal eens tot verwarring in het aanroepen van de methode en in de meeste gevallen ook tot langere en complexere methoden. Een voorbeeld van waar je met minder parameters zou kunnen werken is GlobalResult.dataCompactor(). Je zou de eerste twee parameters kunnen vervangen door een class Range.

Maar over het algemeen scoort de code dus bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase.

Tot slot is het positief om unit tests in de code aan te trekken. Hopelijk lukt het jullie om de verhouding productiecode:testcode gelijk te houden als er functionaliteit wordt toegevoegd.
Multi-Objective Machine Learning Strategies
in Investment Portfolio Optimisation

F.C.A. Abcouwer  R.S. Verboon
f.c.a.abcouwer@student.tudelft.nl  r.s.verboon@student.tudelft.nl
M. Janssen
mark@ch.tudelft.nl

Abstract
In this paper, we examine various machine learning techniques and their applications on the stock market. We review the general concepts of support vector machines, artificial neural networks and genetic algorithms. We provide an introduction to technical analysis and the problem of portfolio optimisation, and provide arguments in favour of using multi-objective machine learning techniques to help solve this problem, as well as some examples of indicators that could be used for this purpose. We discuss the applications of multi-objective support vector machines, artificial neural networks and genetic algorithms on the stock market. Furthermore, we give recommendations on the direction future research in this field should take, in particular research on the usage of advanced machine learning and distributed computing techniques. Finally, we see opportunities in a more thorough analysis on intraday data, which is a field that provides new challenges in need of a different approach.

1 Introduction
Throughout the entire history of the stock market, investors have always sought to get ahead of their competitors in some way, be it through conventional or creative, risk-free or risky, legal or even illegal means. Since the advent of the twenty-first century, financial markets have increasingly become populated with traders who rely on the potential of computing to improve their results. And while researchers currently disagree on whether this is a good or bad thing, there does seem to be a consensus regarding the fact that so far, the research done has only barely scratched the surface of the possibilities this field has to offer.

On the modern stock market, a large portion of trading has become automated. Investment banks, trading firms, pension funds and many other institutional traders use automated trading in the hopes of minimising risk and maximising profits. A specific kind of automated trading that has gained a lot of popularity recently is High Frequency Trading (HFT), a manner of trading that rapidly trades securities, trading in and out of positions in the blink of an eye. With computing power becoming ever cheaper and more readily available, competition among traders is no longer just about who has the best trading strategies: it is also a matter of who is the fastest in deciding when and how to trade.

Obviously, a human investor is not able to simultaneously make large quantities of decisions in the blink of an eye: for this reason, companies build automated trading systems that take instructions from humans and then perform ‘split-second trades’ according to pre-defined strategies. This way of trading is not without risks, however: take for example the 2010 ‘Flash Crash’, where the Dow Jones suddenly fell about 1000 points (approximately 9%), only to recover within a few minutes, which was attributed by many to the limitations of automated trading. Malfunctioning systems have also been known to cause traders massive losses. Therefore, it is crucial to not build a system that is robust and can also devise strategies and predict the course of the market as effectively as is possible.
One way of devising trading strategies, which has gained both popularity and credibility in recent years, is Technical Analysis (TA). Through the study of past performance of stocks and other indicators, technical analysts seek to find patterns in the performance of the market and attempt to use these patterns to their advantage. This way of analysing the market has also drawn a lot of criticism, especially from proponents of the Efficient Market Hypothesis (EMH), who believe that financial markets are ‘informationally efficient’ and, as a result of this, cannot be consistently predicted. A more thorough explanation of TA and the EMH will be given in Section 4.

Many researchers have already looked into stock market prediction and portfolio optimisation. Though the terms and techniques used may vary, in the end most papers we cite have the same goal: trying to teach a computer to predict the stock market or help make investment decisions as accurately as possible.

Besides the approaches discussed in this paper, there are of course other machine learning methods with their own pros and cons. Huang and Tsai [19] used a self-organising feature map (SOFM) combined with support vector regression (SVR) to predict the Taiwanese futures market, and Afolabi and Olude [2] used a hybrid Kohonen self-organising map (SOM).

There have also been many reviews of various techniques. An example is work by Yoo, Kim and Jan [37], who compared classical statistical techniques, neural networks, support vector machines, and case-based reasoning.

In this paper, we will explore using machine learning techniques for stock market prediction and portfolio optimisation, combining knowledge from the fields of finance and computer science. We will examine how existing Machine Learning techniques such as Neural Networks (NN), Support Vector Machines (SVM) and Single- or Multi-Objective Genetic Algorithms (GA) can be used to assist traders in devising and optimising strategies. In Section 2, we explain the basic concepts of several popular Machine Learning techniques. In Section 3, we introduce the financial concepts to be used in the rest of the paper. In Section 4, we argue that multi-objective techniques are superior for the purposes of stock market prediction and portfolio optimisation for a variety of reasons. In Section 5, we take several multi-objective machine learning techniques and show how they can be used for stock trading purposes. Finally, in Section 6 we review our findings, give our suggestions regarding what areas future research should target, and provide some advice on how it should be carried out.

2 Machine learning

In this section we describe three well-known machine learning techniques: Support Vector Machines, Artificial Neural Networks, and Genetic Algorithms. First we introduce Support Vector Machines (SVM) in Section 2.1. In Section 2.2 we give a brief overview of the Artificial Neural Networks technology. Finally, in Section 2.3 we introduce Genetic Algorithms.

2.1 Support Vector Machines

Support Vector Machines (SVM) find their origin in pattern recognition [13]. The technique is designed as a two-class classifier, which means that a trained SVM is able to distinguish between two classes of input. A common example is the optical recognition of numbers.

SVMs are operated in a two phases. First, the SVM is trained to supplied data which is already classified. Second, the SVM is able to classify other objects according to its model.

SVM is a so-called white-box model: it is possible to see the inner workings of the system after training. This white-box model is opposed to a black-box model, in which the user is not able to see the underlying parameters of a trained model.

SVM in general

The first phase, the training, is based upon a regression of a line. If we consider a simple example, a dataset with points in a two dimensional plane with classification. This dataset is represented
Figure 1: This is a simple example of a 2-dimensional classification problem. The dotted line is the line that separates the two classes of nodes. The Support vectors are indicated by a continuous line.

as two groups of dots, as shown in Figure 1. SVM uses a maximisation problem to draw a line between the two groups of points. The distance from the line to each class of points has to be maximised. This maximisation is done using Lagrange multipliers. The distance from the line to each of the two classes of dots has to be maximised, subject to that the line has to be in between the two group of dots.

From this optimisation, two support vectors follow. These two vectors indicate the decision line between the two groups of dots. These two vectors are as far apart from each other as possible, due to the optimisation. These vectors are tangents of the two classes of dots.

The input is given in a feature space. Several features can determine the place of an object in the space. The features are characteristics of objects. For some feature spaces the objects are not in such way that an easy classification can be given. The solution to this problem is to map the objects from the feature space to a higher order feature space. This is done using a kernel function. In this higher order feature space a SVM classification can be made. A new object can be classified by mapping it to the higher order from the normal feature space and than its class can be estimated.

For some problems it is not possible to give a perfect classification based upon the training set. The solution to this is to allow some objects to be out of the classification line. This means that some objects are not in the correct corresponding side of their class. This is named the soft margin [13]. The soft margin determines how many objects are 'allowed' to lay outside of the classification.

**SVM and Time Series**

In finance several examples exist of the usage of SVM for time-series forecasting. Tay [31] used a five day Relative Price Difference (5-RDP) to do forecasting. The method used here is to train the SVM on 5-RDP values from the past. Kim [22] did research on the number of dimensions the feature space should have. A higher order feature space gives more possibilities to do accurate classification on. The problem of over fitting can be overcome to choose an order of dimension which is not too high [31].

**2.2 Artificial Neural Networks (ANNs)**

Artificial Neural Networks (ANNs) are a model of computing that performs pattern recognition tasks in a manner that is inspired by the structure and performance of a biological neural network,
such as the one found in the human brain. ANNs are extensively described in books by Duda et al. [17] and Yegnanarayana [36].

Attractive features of a biological neural network include robustness and fault tolerance, flexibility, the ability to deal with a variety of data situations and collective (parallel) computation.

The performance of ANNs will probably never come close to that of a biological neural network, if not only for that we do not fully understand the operation of biological neurons. Furthermore the number of interconnections and asynchronous nature make it impossible to simulate; the human brain, for example, has about 85 billion neurons with $10^{14}$ to $10^{15}$ interconnections.

However, we can utilise some features of a biological neural network by constructing a network consisting of basic interconnected processing units. The general model of a processing unit consists of a function part followed by an output part. The function part receives $N$ input values and applies an activation function to derive the so-called the activation value. For example, a very simple activation function could sum all inputs, making the activation value the sum of all inputs. The output part produces a signal from this activation value.

Several processing units are interconnected according to a predefined topology to accomplish a specific task. The inputs to a processing unit may come from outputs of other processing units and/or from external sources. The amount of output of one unit received by another unit is influenced by the strength of the connection, reflected in a weight value associated to the link. Each of the $N$ units in a given ANN has, at any given instance of time, a unique activation value and a unique output value. The set of $N$ activation values of the networks defines the activation state at that instant. A schematic diagram of a simple neural network can be found in Figure 2.

![A simple neural network](image)

**Figure 2:** Schematic diagram of a simple neural network. [35]

To update the output states of all units, multiple strategies are possible: the output could be updated asynchronously, changing all activation values at the same time, or sequentially, taking the output state of the network into account each time. For each unit, the output state can be determined from the activation value either deterministically or stochastically. ANN models use more complex equations to govern activation dynamics according to task to be handled.

**ANNs and Time Series**

ANNs are normally used for pattern recognition tasks. To make them suitable for time series prediction, a multilayer perceptron (MLP) or radial basis function (RBF) topology is used, with a set of $N$-tuples as inputs and a single output as the target value of the network. These techniques are extensively described in papers by Connor [12] and Frank et al. [18].

A MLP topology consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Each node (except for input nodes) is a neuron with a nonlinear
activation function. MLP utilises backpropagation to train the network, i.e. for a desired output, the network learns from many possible inputs.

In a RBF network, radial basis functions are used as activation functions. Radial basis functions are a means to approximate multivariate functions by linear combinations of terms based on a single univariate function. A full introduction to radial basis functions can be found in a book by Buhmann [8]. Radial basis functions of inputs and neuron parameters are combined into a linear combination which then forms the output of the network. RBF networks are typically trained by a two-step algorithm, with backpropagation as an optional third step to fine-tune the network.

2.3 Genetic Algorithms (GA)

A Genetic Algorithm (GA) is a search heuristic that arrives at its solution in a way derived from natural selection and evolution, or ‘survival of the fittest’ - the ‘fittest’ in this case being the most optimal solutions to the problem the algorithm is trying to solve. Based on these principles, GAs use principles of evolution such as population, selection, reproduction and mutation. GAs evolve a sample of candidate solutions to a problem over ‘generations’, improving upon the best solutions so far while still leaving room for deviation from the norm. To illustrate the workings of GAs, we will take you through the process step by step.

Step 1: Express the solution domain genetically

Solutions to the problem should be encoded in a way that allows for the GA to easily perform necessary operations such as crossover and mutation. Usually, this is done by encoding the solution as an array of bits, though other options are also available, such as using arrays of other types or data structures such as trees.

Step 2: Define a fitness function

A fitness function is a function that, when provided with an encoded solution, expresses how ‘fit’ that solution is to solve the problem in a fitness score. A higher score indicates a better solution.

Step 3: Initial population

Having found a way to encode solutions, an initial population $A$ of $n$ possible solutions is randomly generated.

Step 4: Calculate fitness of initial population

Using the fitness function, the fitness of every member of $A$ is calculated.

Step 5: Make selection from current population

Now that we know the fitness of every member of $A$, we want to create a new generation from the fittest individuals the current population has to offer. A way to select suitable individuals is through roulette wheel selection: we randomly select individuals, with individuals with a higher fitness score having a higher chance to make it into the selection: This can be compared to a roulette wheel where some slots are bigger than others, hence the name.

Step 6: Crossover and mutation

With this selection, crossover and mutation are used to randomly generate the ‘offspring’ of these solutions. When using bitstrings to represent solutions, a crossover operation would combine parts of two solutions from the selection into a new bitstring, and a mutation would take an individual from the selection and randomly alter one or more of its bits. Depending on how much the population should change between generations, the odds of crossover and mutation occurring can be increased or decreased.

Step 7: Insert offspring into new population

The resulting offspring of the previous step, along with some individuals from the selection, is added to a new population $B$ for the next generation of solutions.
Step 8: Repeat Steps 5-7 until both populations have the same size
Keep generating offspring until the new population is ready to replace our old one.

Step 9: Replace old population by the new one
Having replaced $A$ with $B$, there is a new set of individuals whose fitness can be evaluated.

Step 10: Repeat steps 4 through 9 until stopping conditions are reached
The algorithm typically terminates after a certain amount of generations, or after the population has reached a certain level of fitness.

Because a solution with a higher fitness score has a higher chance to ‘survive’, the average fitness over the population will increase over time, and because of the differentiating effect of crossover and mutation, solutions are (somewhat) protected against clustering around a suboptimal local peak. In this way, Genetic Algorithms, when given enough time and having variables such as crossover rate, mutation rate and population assigned appropriately, will be able to arrive at a solution to a wide variety of problems, and they are indeed employed today in many fields, ranging from bioinformatics and chemistry to economics and finance.

A specialisation of GAs that is often used in stock market prediction is Genetic Programming (GP). The distinguishing feature of GP is that its population is built up of computer programs. By measuring each individual program’s performance, GP tries to find a program that is best suited to performing a certain computational task and providing the desired output. In Section 5 we will elaborate further on how this approach, or GA in general, can be used in predicting the stock market and making investment decisions.

GAs and Time Series
Using GAs with time series can be done in several ways. Two examples are predicting future stock price or trying to find a profitable strategy for a market, given historical data. A way to encode solutions in these two situations is in the form of polynomial functions, which allows for the easy application of mutation and crossover operations. These could be built up as a tree structure, with mutation changing branches and crossover switching branches around between trees. Fitness can be measured in the first case by seeing how close the solution gets to correctly predicting stock price or market trend (up or down), or, in the second case, by measuring how much profit (or loss) a generated strategy would have made over the given time period. A good example of this approach was given by Neely et al. in their 1997 paper. [28]

3 Financial Background

In this section, we will give a brief overview of some concepts used throughout the paper. In section 3.1 an introduction to Technical Analysis is given. This principle relates to the forecasting of stocks. In section 3.2 multiple indicators are shown, which can be used in Multi-Objective Learning for Finance, this directly relates to the next section, section 4. These indicators are indicators an investor would directly use.

3.1 Technical Analysis (TA)

Technical analysis has been around for quite some time now, but this part of the financial analysis is still often referred as pseudo-science, and its relation to finance is considered by some to be the same as alchemy is to chemistry. The goal of technical analysis is to forecast the value of assets, without any knowledge about the company itself. Instead, TA bases its predictions on the historic price path of the asset. The three main assumptions of technical analysis are as follows:

- **The market discounts everything**: TA assumes that, at any given time, a stock’s price reflects all aspects affecting the company, and as such there is no need to account for fundamental factors, broad economic factors or market psychology in a strategy. It follows that
the investor then only needs to consider the supply and demand of a stock in the market, which can be derived from the movement of the stock’s price.

- **Prices move in trends:** The second of the basic assumptions of TA is that prices trend directionally: up, down, or sideways or some combination. If a trend can be found in a stock’s past performance, the analyst can use this to predict the stock’s future behaviour, as it is likely to keep following its current trend.

- **History tends to repeat itself:** In terms of price movement, history is assumed to repeat itself. This is believed to be caused mainly by market psychology: because investor behaviour repeats itself, recognisable and thus predictable price patterns should emerge over time.

The counterpart to TA is the more traditional method of fundamental analysis. Fundamental analysis is based on the efficient market hypothesis (EMH), which states that the attributes of a company, such as profit, size, etc., are immediately reflected in the price of the asset. The efficient market hypothesis exists in several forms. The weak efficient market hypothesis states that all available knowledge from the past is already reflected in the price. The semi-strong efficient market hypothesis holds that prices reflect all publicly available information and change instantly when this information changes. The strong efficient market hypothesis states that inside knowledge is also directly reflected in the price and that therefore inside knowledge does not have any value.

The goal of TA is to forecast the price paths of assets. This has been done by optical pattern recognition - the most basic example being holding a ruler over a printed graph and extrapolating along the ruler. There are a lot of theories on how price paths behave. As an example we consider the work of Osler [29]. Osler considers two observations about trends: first, that trends tend to reverse course at predictable support and resistance levels, and second, that trends tend to be unusually rapid after rates cross such levels. These two observations about trends are just some of the many aspects considered in TA.

### 3.2 Indicators in Multi-Objective Learning for Finance

The usage of multi-objective machine learning in finance is based upon indicators. These indicators are measures of a stock’s various attributes, and can be weighed by an algorithm to decide on a trading strategy.

A classification can be made upon the indicators for machine learning. The most fundamental one is the historic price path. Others are the fundamental indices and the macroeconomic indices as proposed by Tsai [33]. Fundamental indices include elements used in fundamental analysis e.g. net income growth, cash flow ratio, and return on assets. Macroeconomic indices include for example the import and export values. In this paper we will not consider these macroeconomic indices.

When making investment decisions, the trader is confronted by a very large pool of available indicators to use in devising their strategy. An overview of indicators is given in Table 1.

### 4 Multi-Objective Optimisation

In this section, we will give an introduction to technical analysis and the basic principles of multi-objective optimisation. We will explain why an investor would want to take a multi-objective approach to portfolio optimisation.

#### 4.1 Basic Principles of Multi-Objective Optimisation

The basic concepts of multi-objective optimisation are elegantly described by Castillo-Tapia et al. [10], as follows.
%K Stochastic %K. It compares where a security’s price closed relative to its price range over a given time period.

\[ \frac{HH_t - LL_t}{HH_t - LL_{t-n}} \times 100 \]
where \( LL_t \) and \( HH_t \) mean lowest low and highest high in the last \( t \) respectively.

%D Stochastic %D. Moving average of %K.

\[ \frac{\sum_{i=0}^{n-1} %K_{t-i}}{n} \]

Slow %D Stochastic %D. Moving average of %K.

\[ \frac{\sum_{i=0}^{n-1} %D_{t-i}}{n} \]

Momentum It measures the amount that a security's price has changed over a given time span.

\[ C_t - C_{t-4} \]

ROC Price rate-of-change. It displays the difference between the current price and the price \( n \) days ago.

\[ \frac{C_t - C_{t-n}}{C_t - C_{t-n}} \times 100 \]

Williams’ %R Larry William’s %R. It is a momentum indicator that measures overbought/oversold levels.

\[ \frac{HH_t - C_t}{HH_t - LL_t} \times 100 \]

A/D Oscillator Accumulation/distribution oscillator. It is a momentum indicator that associates changes in price.

\[ \frac{HH_t - C_{t-1}}{HH_t - LL_t} \]

Disparity5 5-day disparity. It means the distance of current price and the moving average of 5 days.

\[ \frac{C_t}{MA_5} \times 100 \]

Disparity10 10-day disparity.

\[ \frac{C_t}{MA_{10}} \times 100 \]

OSCP Price oscillator. It displays the difference between two moving averages of a security's price.

\[ \frac{MA_t - MA_{10}}{MA_5} \]

CCI Commodity channel index. It measures the variation of a security’s price from its statistical mean.

\[ \frac{(M_t - SM_t)}{(0.0153D_t)} \]
where \( M_t = (H_t + L_t + C_t)/3 \), \( SM_t = \sum_{i=1}^{n} M_{t-i} \), and \( D_t = \sum_{i=1}^{n} |M_{t-i} - SM_t| \).

RSI Relative strength index. It is a price following an oscillator that ranges from 0 to 100.

\[ 100 - \frac{100}{1 + \left( \frac{\sum_{i=1}^{n} Up_{t-i}}{\sum_{i=1}^{n} Dw_{t-i}} \right) / \left( \sum_{i=1}^{n} Up_{t-i}/n \right) / \left( \sum_{i=1}^{n} Dw_{t-i}/n \right) } \]
where \( Up_t \) means upward-price-change and \( Dw_t \) means downward-price-change at \( t \).

MACD Moving average convergence/divergence. It shows the relationship between two moving averages of prices.

\[ MACD(n)_{t-1} + n/2 + 1 \times (DIFF_t - MACD(n)_{t-1}), \]
\[ DIFF_t = EMA(12)_t - EMA(26)_t, \]

\( C_t \) is the closing price at time \( t \), \( L_t \) the low price at time \( t \), \( H_t \) the high price at time \( t \), \( MA_t \) the moving average of \( t \) days, and \( EMA \) the exponential moving average.

Table 1: Indicators
Without loss of generality, we will assume only minimisation problems and are interested in solving problems of the type:

\[
\text{minimise } \vec{f} (\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{1}
\]

subject to:

\[
g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}
\]

\[
h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}
\]

where \( \vec{x} = [x_1, x_2, \ldots, x_n]^T \) is the vector of decision variables, \( f_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, k \) are the objective functions and \( g_i, h_j : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m, j = 1, \ldots, p \) are the constraint functions of the problem.

We are interested in the concept of multi-objective optimality, which can be more formally described through the following definitions:

**Definition 1.** Given two vectors \( \vec{x}, \vec{y} \in \mathbb{R}^k \), we say that \( \vec{x} \leq \vec{y} \) if \( x_i \leq y_i \) for \( i = 1, \ldots, k \), and that \( \vec{x} \) dominates \( \vec{y} \) (denoted by \( \vec{x} \prec \vec{y} \)) if \( \vec{x} \leq \vec{y} \) and \( \vec{x} \neq \vec{y} \).

**Definition 2.** We say that a vector of decision variables \( \vec{x} \in X \subset \mathbb{R}^n \) is non-dominated with respect to \( X \), if there does not exist another \( \vec{x}' \in X \) such that \( \vec{f}(\vec{x}') \prec \vec{f}(\vec{x}) \).

**Definition 3.** We say that a vector of decision variables \( \vec{x}^* \in F \subset \mathbb{R}^n \) (\( F \) is the feasible region) is Pareto-optimal if it is non-dominated with respect to \( F \).

**Definition 4.** The Pareto Optimal Set \( P^* \) is defined by:

\[
P^* = \{ \vec{x} \in F \mid \vec{x} \text{ is Pareto-optimal} \}
\]

**Definition 5.** The Pareto-front \( PF^* \) is defined by:

\[
PF^* = \{ \vec{f}(\vec{x}) \in \mathbb{R}^k \mid \vec{x} \in P^* \}
\]

The image of the Pareto optimal set forms the Pareto front which is normally displayed in a graphical form in order to determine the most desirable combination of objectives. We thus wish to determine the Pareto optimal set from the set \( F \) of all the decision variable vectors that satisfy equations 2 and 3. When dealing with real-world problems, in practice, not the entire Pareto optimal set is normally desirable (e.g. it may not be desirable to have different solutions that map to the same values in objective function space) or achievable.

### 4.2 Reasons for Multi-Objective Optimisation

In the financial world, the performance of traders is not only evaluated based on the profit they make. Profit is certainly an important aspect of performance, but metrics related to risk are also important indicators of how a trader is doing. A trader who implements a trading rule which maximises profit regardless of the risks involved is, so to speak, ‘asking for trouble’.

The general idea of portfolio management is to have a portfolio which maximises profit under as little risk as possible. Modern portfolio theory is based on theories formulated by Markowitz [26]. Markowitz believes that as an investor, an optimal portfolio does not exist, because there is an inherent trade-off between risk and profit.

This trade-off is a key argument to justify using multi-objective machine learning. A trader does not just want to maximise profit, but wants to find an optimal point between profit and risk. In technical analysis, the objectives are composed of multiple fitness functions.

Closer to our research question is the work of Sagi et al. [7], who proposed four objectives. The objectives are: maximising profit, minimisation of transaction cost, minimisation of risk trend, and minimisation of risk correlated to an index.
5 Techniques for multi-objective optimisation

In the Section 2, we examined several Machine Learning techniques and explained why multi-objective machine learning is a good solution to the problem of predicting the behaviour of the stock market and assisting traders in devising effective trading strategies. We now turn our attention to the application of support vector machines, artificial neural networks, and finally multi-objective genetic algorithms to this problem.

5.1 Support Vector Machines (SVM)

Earlier work on SVM mainly focused on pattern recognition of images. The application to finance was made later. A study by Mukherjee [27] applied SVM to a chaotic time series. The experiments showed that SVM is able to forecast a time series better than other techniques as neural networks. Time series can be used to represent financial data, but the time series used here was generated by the researchers themselves. Therefore, it is uncertain whether their conclusions can actually be applied to ‘real’ financial time series. This is due to the fact that in this work time series were considered of a different origin. It is possible that time series of finance behave different in these models.

Work by Kim [22] concluded that it is indeed possible to forecast financial time series. The time series were taken from futures prices on the Chicago Mercantile Market: the results were that SVM outperformed back-propagation neural networks. By employing technical analysis indicators, Kim was able to successfully optimise his strategy using SVM.

Most studies we found only used a small input size for their SVM or other machine learning techniques, and only a small amount of features. Some studies, however, consider a large number of input features [9]. The twenty input features used in this study is more than in most studies. The features used in this study are again technical analysis indicators, such as the RSI and the MASD. Experiments were performed on the S&P 500 index, attempting to predict the closing price of the next day. The usage of the twenty features on the whole index led to a forecasting result with an accuracy of 86%.

Another study has shown that a trading system using SVMs can outperform other trading
strategies [24]. This study compared a trading strategy using SVMs to a Technical Analysis Trader, a Random Trader and a few other traders. They have shown that the SVM trader outperformed the other traders.

**SVM in a hybrid setting**

In general, SVM are not very effective when they are used in multi-objective settings. It is possible to approximate multi-objective behaviour with a good fitness function that optimises several parameters, but, as explained previously, this is not a very effective solution.

SVM can be used, however, in a multi-objective GA setting, as shown by Ayhin, Karakose and Akin [5], by using SVM as an intermediate step. SVM is applied to each objective, so multiple SVMs exist per iteration. This technique has been described as ‘parallel learning’ [6]. Although it has been proven to work for only two other cases, the results so far seem promising. The experiments in this work were performed on a three-phase engine, but we believe that this technique of having multiple objectives, optimising them individually, and then using GA to combine them can also be applied to finance.

This same kind of optimisation has also been performed in other fields, as shown in work by Igel [20]. Experiments were performed using a common medical benchmark dataset, and the results were modelled on a Pareto frontier. Their results showed that SVMs can help achieve better results in multi-objective optimisation.

A work more related to the traditional time series analysis combined ARIMA with SVMs [30]. The ARIMA (Autoregressive integrated moving average) analysis has been widely used in time-series analysis. The ARIMA model is an extension of the moving average model; in ARIMA the volatility also moves over time. This work has shown evidence that SVMs can help better forecast time series. The forecasting errors were reduced. Optimal parameter selection is not achieved – this is still left for future research. Although ARIMA is not part of the ‘technical analysis school’, it can be used to forecast stocks.

**5.2 Artificial Neural Networks**

Artificial Neural Networks are not directly suited for multi-objective optimisation problems. However, they have been successfully applied for financial optimisation problems in a single-objective setup. Additionally ANNs can be combined with other techniques to form hybrid machine learning strategies. We will review several papers that researched ANNs in this area.

Chen et al. [11] model and predict the direction of return on the market index of the Taiwan Stock Exchange. A probabilistic neural network (PNN) is used to forecast the direction of index return after it is trained by historical data. These forecasts are compared with generalised methods of moments (GMM) with a Kalman filter. In addition, forecasts are applied to various index trading strategies, and performances are compared with those generated by a ‘buy-and-hold’ strategy: that is, simply buying stocks and holding on to them for a long period of time before selling, as opposed to trading frequently. Performance is also compared to investment strategies guided by forecasts estimated by the random walk model and parametric GMM models. Empirical results of their study show that the PNN-based investment strategies obtain higher returns than other examined investment strategies.

Lam [23] investigated the usage of neural networks, specifically the backpropagation algorithm, to integrate fundamental and technical analysis for financial performance prediction. She compares the performance of neural networks with a minimum and maximum benchmark. As the maximum benchmark the average return from the top one-third returns in the market that approximates the return from perfect information is used. The minimum benchmark is the overall market average return that approximates the return from highly diversified portfolios. This study’s experimental results indicate that neural networks using one year’s or multiple years’ financial data consistently and significantly outperform the minimum benchmark, but not the maximum benchmark. Neural networks with both financial and macroeconomic predictors do not outperform the minimum nor maximum benchmark in this study. Moreover, the study has a demonstration of rule extraction
as a post-processing technique to improve prediction accuracy using expert knowledge and for
explaining prediction logic to financial decision makers.

Tsai et al. [32] used ANNs in combination with a decision tree (DT) to form a hybrid machine
learning technique for stock price forecasting. Their experimental result shows that the combined
model has a 77% accuracy, which is higher than single ANN and DT models. However, they do
not compare against other algorithms such as GA or SVN. They also indicate that a GA could be
used to pre-process the data for better performance.

In a newer paper, Tsai et al.[33] propose a hybrid algorithm which is based upon several machine
learning techniques to select the right indicators. The basic technique to select good indicators
is Principle Component Analysis or PCA, a procedure that transforms a set of observations of
possibly correlated variables into a set of values of linearly uncorrelated variables (called principal
components). Other methods include machine learning techniques such as a back-propagation
neural network, genetic algorithms, and decision trees (CART). Experimental results show that
the intersection between PCA and GA and the multi-intersection of PCA, GA, and CART perform
the best. Their experiments show that the 85 original variables could be reduced to the 14 and
17 most important features respectively.

5.3 Genetic Algorithms (GA)

Early explorations into portfolio optimisation using GAs yielded varying, but mostly positive
results. Only a single, simple objective played a role: making as much profit as possible. This
was usually measured by the excess return the algorithm’s strategy would have provided over a
standard buy-and-hold strategy.

However, genetic algorithms can also be used in a multi-objective setup as introduced in the
previous section. In this use case they are usually referred to as ‘multi-objective evolutionary
algorithms’ or MOEAs.

Anione et al. [4] were (one of) the first to use MOEAs in the area of investment portfolio
optimisation. They adopt a GA with a weighted linear aggregating function as the basis of their
research. As an objective they use a minimisation of risk and a maximisation of (expected)
return, introducing a trade-off coefficient between the two factors. To achieve this using GA
several different populations are to be evolved for a number of trade-off coefficient values. These
different populations form different portions of the Pareto frontier. Of course, the higher the
number of populations, the more granular the Pareto frontier will be. Loraschi et al. [25] pursued
this research and found that a distributed GA approach outperformed the sequential version of
the algorithm.

Neely et al. [28] used Genetic Programming (a certain ‘flavour’ of GA) to find technical trad-
ing rules for several Foreign Exchange (FX) markets. The rules had a tree structure, which made
them suitable for crossover and mutation operations. Trees were made up of arithmetic oper-
ations, Boolean operations, conditional operations, numerical constants and Boolean constants. An
example trading rule generated by this approach could be: \( \text{avg}(15) > \text{avg}(250) \), which means: go
long if the 15-day moving average exceeds the 250-day moving average. Neely et al. were very
positive about their results, even though they had not put any effort into optimising the genetic
process.

Allen and Karjalainen [3] used GA’s to construct trading rules in a way similar to the work
by Neely et al., but they were not as optimistic about the results: they concluded that their
simulation did not earn significant excess returns over a simple buy-and-hold strategy, and that
their work provided little evidence for the existence of economically significant technical trading
rules.

Furthermore, Dempster and Jones [14], using a similar genetic programming strategy and
building on their own previous work, concluded that though the majority of indicator-based rules
were loss-making individually, it was possible to profit from technical trading rules when they are
constructed by combining several different indicators instead of using individual ones: in this case,
they found it was possible to make a moderate, but significant profit.
Finally, Dempster et al. [15] conducted further research to compare (combinations of) several machine learning techniques in the area of intraday FX trading. While most research focuses on optimisation using closing prices, i.e. one data point per day, many trading firms instead leverage intraday high-frequency data to be able to trade during the day. The aim here is to end the day with a net open position of zero, instead of using much longer horizons as is customary when trading using closing prices. In their research they consider several strategies that use a collection of technical indicators as input and seek a profitable trading rule defined in terms of them. The strategies are approached using reinforcement learning and a GA, and compared to an appropriate Markov decision problem and a simple heuristic. They find the GA approach to deliver superior results at nonzero transaction costs, although they do mention that none of the evaluated algorithms produce 'significant profits' at more realistic transaction costs.

A study from Bodas-Sagi [7] performs experiments using MOEAs. This system uses the simple trading role to buy and hold an asset for a few days. The forecasts are given for a day, a week and a month. Algorithms are rated as a fraction of the forecast they have correctly forecasted. An algorithm only forecasts if the price will be lower or higher as it is now. The system reviews several combination of systems. The objectives which are maximised to are RSI and MACD, these terms are explained in table 1. The best result was achieved using a MIX of objectives. This means combining all objectives together yields the best results. Other methods that were compared to were single-objective optimisations and multi-objective optimisations based upon specific components. Experiments were performed using the S&P 500 index. This paper shows evidence that multi-objective machine learning can play a role in forecasting asset price paths.

Yet another study (Becker et al. [6]) performed experiments in the MOEA field. The researchers used three fitness functions that are measures of the fitness of a stock in a portfolio. The question at hand was which stocks should be selected in a portfolio. Experiments were performed using the S&P 500, excluding some companies. A test was performed to compare three evolutionary techniques, based on the fitness functions for optimisation: a parallel algorithm, a sequential algorithm and a constrained fitness function. The parallel algorithm did outperform their benchmark and the constrained fitness function produced results close to the parallel algorithm.

6 Conclusions and Future Research

In this paper, we examined the application of various machine learning techniques on the area of investment portfolio optimisation. After investigating the advantages and disadvantages of artificial neural networks, support vector machines and genetic algorithms, we came to the conclusion that genetic algorithms are most suited for a multi-objective optimisation approach. We examined some of the theory behind multi-objective investment and technical analysis, and the arguments for and against the latter. Furthermore, we also described in detail the application of multi-objective genetic algorithms to portfolio optimisation.

Though we have found evidence in favour of theory, we have found that technical analysis is a field that is still very much lacking in recognition amongst researchers, mostly because there is a lack of evidence to support the theory. In many of the studies we came across in writing this paper, the research scope was often limited, or experiments were carried out in such a way that the results were either vague or not reliable enough. Often, little to no effort was made to optimise their machine learning process at all: perhaps this was due to a lack of expertise in the relevant computer science areas. The papers we found commonly suggest that further research could or should explore the effects of optimisation. The case could be made that in the past, researchers might not have had access to sufficient computing power or the raw data required to perform thorough research, but it seems apparent that these issues have become less relevant over the years.

In order to tackle the problem of inadequate amounts of computing power, we feel that more research could be done in the area of utilising distributed computing strategies in order to more effectively solve machine learning problems. In the case of genetic algorithms, distributed computing could, for example, help achieve higher population sizes. In the more general case, the
use of larger or more granular training data sets and an increased number of indicators (decision variables) used in a multi-objective approach could produce markedly different results. Even if no significant increase in accuracy is found this way, a distributed computing approach with a noteworthy improvement in speed (run time) could still open the door to more real-world applications of machine learning in finance. Rapid evaluation and improvement of strategies could change the way that trading is performed. Trading is extremely time-sensitive: deploying a strategy without taking the time to properly test and evaluate it carries with it a risk of large losses, but taking too long carries with it the risk of missing out on opportunities for profit. Improved computing speed would allow strategies to be more properly tested and optimised before they are put into use in a real-world trading platform.

Moreover, we feel more thorough studies could definitely help decide the debate about technical analysis. At the moment, many studies have been published, but few to none have the scale required to help change the general opinion on technical analysis in the financial sector. We propose that a large-scale study be carried out, comparing a large number of technical indicators implemented in many different strategies, incorporating the previous suggestion of utilising distributed computing power, will certainly shed new light on technical analysis. Whether these results can provide a definitive breakthrough on the application of technical analysis is to be seen, considering the fair amount of skepticism in this area.

An area of research with relatively little work is usage of intraday data, even though its widespread use in the financial world in the form of high-frequency trading. We think that the application of machine learning techniques to intraday data certainly forms an interesting research direction. Most current experiments are only performed on closing prices of data, probably due to the common availability of this data. Intraday data, especially tick data, is very valuable for more elaborate strategies. A system to evaluate and improve strategies based on technical analysis using machine learning could help a trader make even better decisions in a system using intraday data.

We feel that the use of advanced machine learning techniques could be an enormous asset to research in the general financial domain, and the pursuit of future research will certainly contribute to shaping real-world applications in this area.

Acknowledgements

We would like to express our great appreciation to our supervisor, Mathijs de Weerdt, for his guidance and feedback in writing this paper. We would also like to extend our thanks to Xander Savenberg, whose knowledge of the domain and feedback also proved very useful.

References


References


