VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV EKONOMIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF ECONOMICS

# THE USE OF ARTIFICIAL INTELLIGENCE ON STOCK MARKET

VYUŽITÍ UMĚLÉ INTELIGENCE NA KAPITÁLOVÝCH TRZÍCH

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. MARTIN KUNA
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. PETR DOSTÁL, CSc.
SUPERVISOR

BRNO 2010

Brno University of Technology
Faculty of Business and Management

Academic year: 2009/10
Institute of Economics

# Diploma Thesis Assignment

**Bc. Martin Kuna**

European Business and Finance (6208T150)

Pursuant to Act. No. 111/1998 Coll., on Higher Education Institutions, and in accordance with the Rules for Studies and Examinations of the Brno University of Technology an Dean's Directive on Realization of Bachelor and Master Degree Programs, the director of the Institute of is submitting you a diploma thesis of the following title:

**The Use of Artificial Intelligence on Stock Market**

In the Czech language:

**Využití umělé inteligence na kapitálových trzích**

Instructions:

Introduction
Executive summary
Theoretical basis of the work
Problem analysis and current situation
Proposals and contribution of suggested solutions
Conclusions
References
Apendices

Literature / Sources:

DOSTÁL, P. Advanced Economic Analyses. 1. vyd. Brno: CERM, s.r.o., 2008, 80s. ISBN 978-80-214-3564-3.
DOSTÁL, P. Pokročilé metody analýz a modelování v podnikatelství a veřejné správě. 1. vyd. Brno: CERM, s.r.o., 2008. 340s. ISBN 978-80-7204-605-8.
GALUSHKIN, A. Neural networks theory. New York : Springer, c2007. 396 s. ISBN 978-3-540-48124-9.
MUNAKATA, T. Fundamentals of the new artificial intelligence : neural, evolutionary, fuzzy and more. London : Springer, c2008. 255 s. ISBN 978-1-84628-838-8.
REJNUŠ, O. Finanční trhy. Ostrava: Key Publishing, 2008. 559 s. ISBN 978-80-87071-87-8.
The MathWorks, Inc.. Documentation for MathWorks Products [online]. [2009] [cit. 2009-07-05]. Dostupný z WWW: <http://www.mathworks.com/access/helpdesk/help/helpdesk.html.

The supervisor of diploma thesis: doc. Ing. Petr Dostál, CSc.

The deadline for submission for the diploma thesis is given by the Schedule of the Academic year 2009/10.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta podnikatelská
L.S.

Ing. Tomáš Meluzín, Ph.D.
Director of the Institute

doc. RNDr. Anna Putnová, Ph.D., MBA
Dean of the Faculty

Brno, 31.3.2010

## Abstract

This Master's thesis is focused on application of selected artificial intelligence methods on stock markets. Specifically, it is concerned with utilisation of artificial neural networks for trend prediction, and with the possibility to apply genetic algorithms to the investment portfolio optimisation problem. It contains suggestions on how these problems might be solved in practice. These suggestions are framed as models created in the Matlab development environment.

## Abstrakt

Diplomová práce se zabývá aplikací vybraných metod umělé inteligence v prostředí kapitálových, potažmo akciových, trhů. Konkrétně se zaměřuje na využití umělých neuronových sítí pro predikci trendu a na možnost aplikace genetických algoritmů k řešení problému optimalizace investičního portfolia. Obsahuje návrh řešení uvedených problémů v praxi. Návrhy jsou koncipovány ve formě modelů zpracovaných ve vývojovém prostředí Matlab.



## Keywords

Artificial Intelligence, Neural Networks, Genetic Algorithms, Prediction, Portfolio Optimisation, Capital Market

## Klíčová slova

Umělá intelligence, neuronové sítě, genetické algoritmy, predikce, optimalizace portfolio, kapitálový trh

## Bibliographic Citation

KUNA, M. *The Use of Artificial Intelligence Methods on Stock Market.* Brno: Brno University of Technology, Faculty of Business and Management, 2010. 74 p. Supervisor doc. Ing. Petr Dostál, CSc.

# Declaration of Originality

Hereby, I declare that this Master's thesis is original and was created under the supervision of doc. Ing. Petr Dostál, CSc. The citation of sources is complete and I did not violate the copyright law (pursuant to Act. No. 121/2000 Coll., on Copyright Law).

Brno, 31 August 2010

.....

Bc. Martin Kuna

## Acknowledgements

The author of this work would like to thank doc. Ing. Petr Dostál, CSc. for his valuable advice and support related to this Master's thesis and the underlying research.

# CONTENTS

# INTRODUCTION

The stock markets are a vital part of the global financial system and can potentially generate significant earnings. This fact attracts many individuals and enterprises to participate on these markets in order to invest their capital with the vision of future yields. But only those market participants who make the right decisions can be successful – the others are, in contrary, losing their money.

Since the capital and, in particular, stock markets, are very complex systems, the corresponding decisions an investor or a speculator has to make are subsequently also extremely complicated. Furthermore, the behaviour of the markets is influenced by a significant range of factors and processes, which are constantly changing and are often difficult to assess. Some of the most common questions are: "Should I buy or sell? Will the price of a particular security increase, or decrease in the future?", and: "Which equities should I include in my portfolio to maximise return and minimise risk?" In order to facilitate such complicated problems, many theories, methods and tools were developed, helping to analyse market behaviour and to provide answers to such difficult questions.

One of the tools capable of addressing such complex issues is the artificial intelligence. It is a relatively new area including methods for solving complex tasks such as price prediction or portfolio optimisation. Given this fact, examining possible ways of practical utilisation of these methods is an extremely interesting topic, because it can help to make 'the right decisions' and therefore lead to the success on the market.

The purpose of this Master's thesis is to present the underlying research that will examine and analyse possible utilisation of two artificial intelligence methods, namely artificial neural networks and genetic algorithms, for stock market indices forecasting and portfolio optimisation respectively.

Thanks to the spread of personal computers and availability of software providing artificial intelligence functionality, the application of the results of the research covered in this thesis is widely possible also for small individual investors and is not limited only to universities, research centres or large organisations, as until recently.

# 1 EXECUTIVE SUMMARY

In this chapter, the main goals of the research, which are derived from the basic problem setting, are described. Following, the methods used during the study are outlined, alongside fundamental information about the data and equipment needed to solve the problem and use the results of the research.

## *1.1 Problem Definition*

Two main problems that are covered in this thesis are arising from the request of BELTREX, spol. s r.o., a small enterprise which wants to start trading their free funds on the stock market in order to increase their value. One part of the free cash-flow would be invested for a mid-term period, minimising the risk by suitable diversification. The rest could be used for short-term speculative trading on the daily basis.

Given these requirements, there are two fundamental questions:

- How should the investor construct their portfolio in order to minimise risk and, at the same time, maximise return from their investment? The trickiness of this issue stems from the fact that there is virtually infinite number of possible stock portfolio combinations.
- For a short term trading, should the company speculate on rise, or fall of the price of a particular equity? If the investor wants to earn profits from their trading, the ability to predict trends will be one of the vital parts of their strategy. The problem here is the accuracy and robustness of the prediction.

## *1.2 Goals*

The goal of this research is to examine the theory of neural networks and genetic algorithms, and consequently develop two models that can be employed in real price trend prediction and portfolio optimisation tasks.

The first model is based upon an artificial neural network. It is to be used to predict the trend direction for a specific equity for the following trading day. The goal is to

construct a model that will have both high success probability, little or no large draw-downs (i.e. the equity curve will rise steadily), and high returns.

The second model will solve a portfolio optimisation task. Using the genetic algorithms, it will create an optimal portfolio as a subset of securities from a large pool of stock market indices. This new portfolio should outperform the original large pool of securities in both terms of return and risk, and having close-to-optimal return-to-risk ratio.

Both models should be suitable for practical application in BELTREX, spol. s r.o., a small individual corporate investor. They need to be easy to use and well automated because the final users will probably not be skilled programmers and financial experts.

## *1.3 Methods*

In the theoretical part of this work, a secondary research was used in order to collect, analyse and summarise the most important information related to the application of artificial neural networks and genetic algorithms for prediction and optimisation tasks for the stock market.

On the other hand, the fundamental part of the practical section is based on primary research, which was conducted during the development and testing of the computer models on historical data. In this part, two artificial intelligence methods, neural networks and genetic algorithms, are applied because they are suitable for solving complex tasks outlined hereinbefore. Specific methods used in this section are experiment and comparison. Experimenting was utilised to construct and optimise the computer models, whereas comparison was conducted later to evaluate the results of the research in broader perspective.

Several prediction models with various parameters and inputs are compared to each other in terms of their simulated trading results. The assessment criteria included the success probability, the total profits of the models and the respective smoothness of the equity curve for the testing period. As little large draw-downs as possible, high total return, and high success probability are preferred to frequent large draw-downs, low

returns and low success probability. The best performing model was selected as the final solution, which can be applied in real conditions.

Comparison was chosen as the best assessment method also in case of portfolio optimisation model. In this case, the benchmark to which the performance of the model was compared was the S&P 500 index from which the original pool of indices was generated. The compared criteria were profit and risk as they are well describing the basic and, from the investor's point of view, most important features of a portfolio. I.e. the portfolio with the same demanded profit as of the benchmark index should bear a lower risk than the index itself. The second benchmark was the efficient frontier of the original pool of equities. The new portfolio suggested by the model should be close to the efficient frontier curve, i.e. the return-to-risk ratio will be close to optimum.

## 1.4 Data and Technical Equipment Used

All stock market data used as inputs to the models was acquired from Yahoo Finance (36), which provides historical daily data free of charge. Since this data is considered to be of a good quality and in a required extent, there is no need to use paid sources.

Both models are constructed in a form of computer programs developed on the Matlab platform because it provides advanced built-in customisable artificial intelligence functionality ready-to-use in specific applications. Matlab is also one of the leading software products for highly demanding computing tasks, e.g. from the financial field.

The final programs can be, in case of need, compiled as a stand-alone executable applications, or standard libraries re-usable on platforms such as Java or Microsoft .NET.

Specifically, the following Matlab toolboxes were primarily used:

- Neural Network Toolbox
- Financial Toolbox
- Data Feed Toolbox
- Genetic Algorithm/Optimisation Toolbox

For the purpose of this research the Matlab development environment available at the Faculty of Business and Management of the Brno University of Technology was used.

# 2 THEORETICAL BASIS OF THE WORK

This chapter serves as an introduction to the common issues of artificial intelligence and capital markets (with focus on stock market). Firstly, definitions of artificial intelligence (AI), and two fundamental AI methods used in this work, are provided. Secondly, fundamental investment factors are discussed since they are important for evaluation of the research. Finally, several most popular theories related to the financial markets are presented with focus on their impact on the methods used in this thesis.

## *2.1 Definitions*

In this section, the two examined AI methods, the neural networks and the genetic algorithms, are briefly discussed. The aim is to provide basic overview, focused on practical features of their behaviour and implementation, rather than on underlying theory. However, further detailed information can be found in literature (e.g. in sources listed in the References chapter).

### 2.1.1 Artificial Intelligence

The field of Artificial Intelligence, or AI, is not easy to describe comprehensively. Various academics present different views and approaches towards this term.

According to Bellman's broad definition from 1978, AI is "the automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning, ..." (cited in 21, p. 563). Later in 1990, Kurzweil defined AI as "the art of creating machines that perform functions that require intelligence when performed by people" (cited in 21, p. 563). There are plenty of other ways how Artificial Intelligence was defined but these two were found to be probably the most suitable ones in relation to this work. Generally, we can understand this term as summed up by Munakata: "[Artificial Intelligence is] the study of making computers do things that the human needs intelligence to do" (18, p. 7). This extended definition includes the fact that AI is mimicking the human thought processes, and that the intelligent tasks are performed by computer technology.

On the other hand, it might be useful do define what is not AI. This category covers pure numeric computations, including engineering problems such as solving linear equations, numeric differentiation, and integration, statistical analysis, etc. Specific tasks such as pure data recording and information retrieval, including non-intelligent databases, simple word processing, or most business data and file processing are not AI (18).

The AI is able to solve problems in fields of reasoning with complete or incomplete information, various forms of perception and learning, or inference based on knowledge. Consequently, it can be applied to problems such as control, classification, and, finally, prediction and optimisation.

The work on Artificial Intelligence started in the early 1950s (17, 21, 18) and, as it is apparent from the definitions given above, it covers wide range of techniques and methods. However, it is a field that is still undergoing constant development. Wider practical commercial and industrial applications of these methods have grown especially since 1990, when they were firstly utilised in Japan on a larger scale (18). The purpose of this thesis is to cover several of the most recent methods of AI, the neural networks and the genetic algorithms. These are often presented alongside other relatively new methods including fuzzy systems, rough set theory, and chaos theory.

#### *2.1.1.1 Artificial Neural Networks*

The artificial neural networks (ANNs), or just neural networks (NNs), same as genetic algorithms, are AI methods strongly inspired by nature. Basically, a neural network is a simplified abstract computer model of the human brain. Same as the brain, the NNs are formed by neurons, the basic units, organised in layers and interconnected by links (18).

There is always an input layer, which is used to 'feed' the network with data, and an output layer, which provides output data. Depending on the type of network used, there can be one or more hidden layers (10). Definitions of these layers, the number of neurons in each layer, the interconnections of these neurons (their respective transfer functions), the way of the training mechanism and the way of acquiring output, are the fundamental parts of neural network architecture (6).

The input layer is formed by input neurons, each neuron for each type of input. These neurons themselves do not manipulate the inputs in any way. On the other hand, the output layer is formed by output neurons (again, one neuron per one type of output), which provide the final output. In the hidden layer(s) custom number(s) of neurons are located. It is apparent, that the basic building block of each network is the neuron. On the following figure, there are inputs $x_1$, $x_2$, ..., $x_m$, coming into the neuron. Each input $x_i$ is multiplied by its corresponding weight $w_i$, and then the product $x_i w_i$ is fed into the body of the neuron. Finally, the neuron computes its output y as a certain function of $y = f(x_1w_1 + x_2w_2 + ... + x_mw_m)$. This function is called the activation, or transfer, function. As a neuron is an abstract model of a brain neuron, the activation function is an abstract model of electrochemical signals received and transmitted by the natural neuron. A threshold also shifts a critical point of the value of $x_1w_1 + x_2w_2 + ... + x_mw_m$ for excitation of the neuron (18).

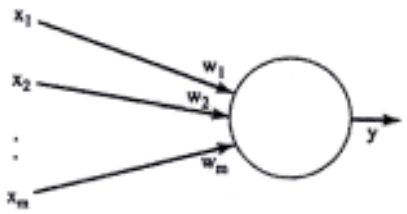


**Fig. 1 - model of a neuron; source: (18)**

The most common activation functions are hardlim, purelim, logsig, and tansig (10).

**Fig. 2 - Activation functions hardlim, purelim, logsig, tansig; source: (7)**

However, the analogy to the human brain is not apparent only in the construction of the neural network but also in the way how it works. Dostál (6) describes two basic phases of the neural network. In the first phase, the NN acts as a 'curious pupil' - it learns from the data provided and adjusts its internal parameters. In the second phase, the network

acts as an ‘expert’ – it produces outputs based on knowledge learned during the first phase (6).

One of the important features of neural networks is that they work as a so called black box. This means that it is impossible to describe the internal structure of the system in full detail. It is possible to set only some parameters of the inner structure of the network (5). When defining its architecture, we can use various types of networks, such as the basic perceptron, or more advanced Hopfield network. The most popular model is the backpropagation network, which can be used, besides others, for prediction tasks (18, 10).

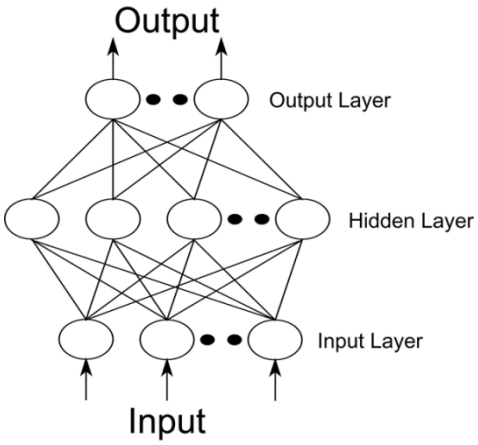


**Fig. 3 - A simple example of backpropagation architecture; adapted from (21)**

The main classes of NNs utilisation cover problems “where the influences on searched phenomena are random, and deterministic relations are very complicated, so that we are unable to separate them and indentify them analytically” (6, p. 48). Neural networks “are suitable for simulation of complicated and often irreversible strategic decision-making” (6, p. 48). Specifically, the most common commercial and industrial applications of NNs are classification, prediction, control (10, 18, 7). Munakata (18), Dostál (5, 6, 7) and Galushkin (10) presented several practical applications, such as pattern recognition of hand-written characters, solving the n-Queen Problem, the Travelling Salesman Problem, clustering cancer findings, or prediction of stock market prices and trends.

#### *2.1.1.2 Genetic Algorithms*

Genetic algorithm (GA), also called Evolutionary Algorithm, is a computer model based on genetics and evolution in biology. The pioneer of natural genetics was a 19$^{th}$ century scientist J. G. Mendel and his findings were further used in Darwin's evolutionary theory. The computer application of genetic algorithms was developed in 1970s by J. Holland and D. Goldberg (7).

The computer genetic algorithms are based on the theory that during the evolution only the best individuals survive whereas the weak ones die-out. The basic elements of GAs are: *selection* of solutions (individuals) based on their goodness, *reproduction* for crossover of genes, and *mutation* for random change of genes. Through these processes, genetic algorithms find better and better solutions to a problem "just as species adapt to their environments" (18, p. 85).

It is useful to briefly review basic terms of genetics in real life, these are chromosome, parents, offspring, and processes of selection, crossover and mutation. Chromosomes consist of millions of genes. One gene from the mother and one gene from the father form a gene-pair for the child. Each pair, or pairs, of genes contribute to specific characteristics of a child, such as eye colour or the blood type. From the chromosomes of parents we obtain new chromosome of their offspring by the processes of genetics. The first process is called selection and represents the choice of chromosomes of the parents. The second process is crossover breeding, when a new chromosome, i.e. a new combination of genes, is formed from chromosomes of both parents. Occasionally, however, some gene values change, perhaps because of some unusual physical, chemical, or biological effects (18).

In a computer model of GAs, the chromosomes are usually represented as a binary string, e.g. 01100110. Each bit in this string represents a gene, i.e. it has a certain value (0 or 1) and position in the chromosome (0 to 7 in the example above). The binary strings mostly represent decimal numbers, which are often used as a parameter for assessing the fitness of the solution (7, 18). The selection means that the chromosomes with the best fitness value (e.g. the highest decimal value) are chosen for further reproduction. These chromosomes then undergo the process of crossover breeding,

when a part of two or more parents' chromosomes are exchanged. This generates a new offspring chromosome with, ideally, a higher fitness value. Mutation in a form of random change of bits from 0 to 1, or from 1 to 0 can seldom occur. Then the whole process of reproduction repeats until a certain condition is fulfilled. Each repetition, or epoch of population, represents one generation in the biological evolution process. The terminal condition is usually a given fitness value of the chromosome, or a specific number of epochs completed (7, 4).

A GA usually operates on a population of initially randomly generated chromosomes. The population advances towards better solutions by applying the above mentioned genetic operators. In each generation, favourable solutions generate offspring that replace the inferior individuals. Crossover hybridises the genes of two parent chromosomes in order to exploit the search space, and constitutes the main genetic operator in GAs. The purpose of mutation is to maintain the diversity of the gene pool. An evaluation of fitness function plays the role of the environment to distinguish between good and bad solutions (4).

According to Dostál (7), the reproduction process of the computer GAs consists of the following exact steps:

1. Random generation of initial population of *n* chromosomes.
2. Quality evaluation of all chromosomes of the given population via the fitness function.
3. Creation of new chromosomes by applying the operators of selection, crossover, and mutation.
4. Eliminating the old chromosomes from the original population in order to create space for new chromosomes.
5. Including the new chromosomes into the population and its concurrent evaluation.
6. If the terminal condition is reached, the best chromosome becomes the solution of the problem – otherwise the reproduction process continues.

These steps are presented on the following diagram.

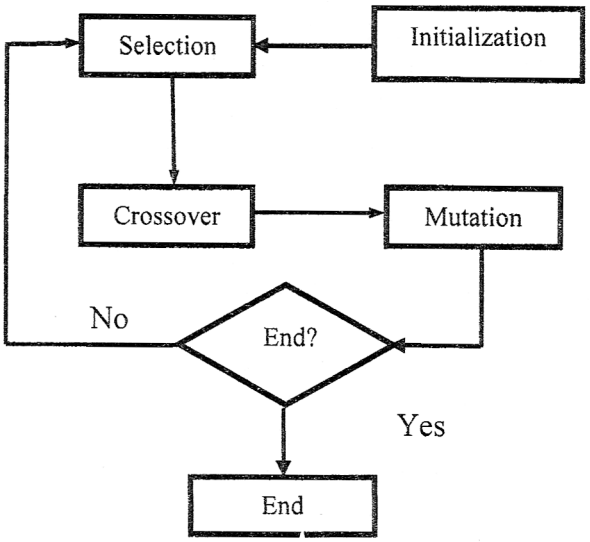


**Fig. 4 - Schema of a GA reproduction process; source (5)**

The genetic algorithms can be used principally for optimisation tasks, e.g. to solve the Travelling Salesman Problem, and other similar problems where finding the exact solution by a 'brute force' would take almost an infinitely long time. The goal of GAs utilisation is not to find the exact solution, but a solution that is close the optimal one (4, 7, 18).

## *2.2 Fundamental Investment Factors*

Investor evaluating possible opportunities always needs to consider carefully three fundamental factors: return, risk, and liquidity (20). The purpose of this section is to briefly describe these factors and explain their significance for investment assessment. Their definition is vital because they are later used to assess the results of the computer models developed during this research.

### 2.2.1 Return

One of the main factors for financial investment evaluation is its return. It represents the rate of appreciation of invested financial funds. These are usually invested in a particular security, or into a portfolio of securities, for a specific time period (20).

There are two main classes of methods used for return calculation, static methods and dynamic methods. Static methods do not take the factor of time into account. They are suitable for evaluation of investments, where time is not very important for the calculation. Specifically, these are usually short-term investments when the calculations are used for estimative evaluation of possible investment opportunities. The dynamic methods, on the other hand, are used for evaluation in cases where the factor of time and consequently the interest rates play an important role. They are particularly suitable for long-term investments, and in times of high interest rates (20).

Another important classification of returns is based on the fact whether they are presented before or after tax. The calculation of net returns, i.e. after tax, might be especially complicated due to the fact that the taxation depends on legislation of specific countries, on the type of income, on the type of investor (natural person or corporate body), international taxation agreements, etc. (20).

Because this work is focused mainly on speculative short-term investments, the static methods will be used for evaluation of results. For the sake of simplicity, all the returns will be also calculated before tax, and only capital yield will be considered. However, the before-tax static methods of evaluation are considered as suitable for the purpose of research results presentation and approximate investment opportunities comparison. The calculations will be limited only to the capital yield to ensure comparability of results. Due to the fact that dividends are not always paid at one certain point in time and that the historical dividend data is not available for longer term, the comparability of results of this research would be radically hindered.

### 2.2.2 Risk

Risk can be regarded as the degree of uncertainty that attaches to the future returns on investment in terms of both capital gains and losses, and dividend and capital payments (22). There are various possible types of risk related to the investment on the capital market. The most common ones mentioned by Rejnuš (20) are the following:

- Interest rate risk.
- Inflation risk.

- Risk of insolvency or bankruptcy of the issuer.
- Risk of the loss of liquidity.
- Currency risk.
- And many others.

All the above mentioned risks often include several specific sub-risk categories and are generally difficult to assess objectively, since the risk evaluation in these categories often depends on personal judgement (14). Hence, these types of risk will be excluded from the evaluation of the research. Instead, the following universal, well comparable ratio will be used.

Probably the most popular way to assess the risk of an investment into a certain security, or a portfolio of securities, is standard deviation. The use of this statistical tool is suitable when there is enough suitable data available (20, 22). Rejnuš (20, p. 183) explains the role of standard deviation in the following way: "… [we need to calculate not only] the expected return of the investment but we also need to evaluate how the real returns might differ from this expected return." Practically, the higher the value of standard deviation, the higher is the risk of the particular investment. The following formula is used to calculate the standard deviation from set of historical time-series data (28):

$$\sigma = \sqrt{\frac{\sum_{k=1}^{N}(x_i - \bar{x})^2}{N-1}}$$ , where

$\sigma$ is Standard Deviation,
x is Historical Price,
$\bar{x}$ is Mean Arithmetic Return of the Time-series,
N is the Number of Samples.

The standard deviation determines the variation in returns on the asset or portfolio and gives investors a mathematical basis for investment decisions. The overall concept of risk is that as it increases, the expected return on the asset will increase as a result of the risk premium earned – in other words, investors should expect a higher return on an investment when said investment carries a higher level of risk, or uncertainty of that

return. When evaluating investments, investors should estimate both the expected return and the uncertainty of future returns. Standard deviation provides a quantified estimate of the uncertainty of future returns (28).

In most theories, the investor is regarded as risk-averse. This means that when two investment opportunities with exactly the same expected returns are given, the investor will choose the one with lower risk. Likewise, if two alternatives have two identical levels of uncertainty attached to them, the one that offers greater average level of return will be the one that is preferred (22, 20). This assumption is reflected in the modern portfolio theory described in one of the following sections.

### 2.2.3 Liquidity

The third fundamental factor of the investment strategy is liquidity. It represents how quickly it is possible to 'convert' an asset into cash. There are many factors influencing the liquidity, such as aggregate demand. The importance of this factor comes from the general fact that with decreasing liquidity, the value of particular assets also decreases (20). An important resume is that assets with higher liquidity should be preferred over those with lower liquidity (using the 'ceteris paribus' principle).

## *2.3 Fundamental Stock Market Theories*

In this section, two dominant theories related to the problems examined in this research are briefly discussed. The Efficient Market Theory is related to the predictability of the markets, whereas the Modern Portfolio Theory is a highly important concept used in portfolio management.

### 2.3.1 Efficient Market Theory

The Efficient Market Hypothesis (EMH) states that nobody can 'beat the market' because it is very close to the economic ideal of perfectly competitive market. Information can be acquired very quickly and cheap, insider trading is strongly prohibited, and the amounts traded on the markets do not allow anyone to take full control over the prices. Furthermore, the barriers for entry and exit are low since the

transaction costs are also low. Generally, the EMH asserts that the prices fully reflect all available information, and that all participants are price takers. Therefore, it is impossible to win substantial excess profits over a longer period of time (22, 15).

EMH exists in three basic forms, strong, semi-strong, and weak. The weak form claims that prices of traded securities reflect all past publicly available information. The semi-strong form adds that prices instantly change, reflecting all new public information. The strong form adds another assumption - prices reflect all hidden or insider information (22, 15).

#### *2.3.1.1 Random Walk Theory*

The Random Walk Theory (RWT) asserts that it is impossible to predict future price movements on the financial markets, because the prices follow a 'random walk'. This hypothesis is in agreement with EMH and states that when there is no new information that can be absorbed by the market, the share prices fluctuate randomly around their intrinsic value (22, 19).

For investor, the most important result of the EMH and RWT would be that the best market strategy is "buy and hold" as opposed to any attempt to "beat the market" (19).

The importance of EMH and RWT for this research is based on the fact that if all past information is already included in the price, and if the prices are following the random walk, it is impossible to predict future price based on historical data.

However, there is evidence, that this might not always be valid. Ryan (22) presents several examples, such as the January effect, day-of-the-week effect, etc. These illustrate, in contrary to both theories, that some trends can be identified and that history can repeat itself.

It is also important to emphasize that the RWT, originally from 1953, is based purely on statistical methods. Since that time, many new theories and methods were developed which challenge both RWT and EMH. Commonly cited is the Behavioural Theory (9). More technical approaches include also several methods of Artificial Intelligence. Lawrence (15) presents two of them, the chaos theory, and the artificial neural networks.

“Contrary to the EMH, several researchers claim the stock market and other complex systems exhibit chaos. Chaos is a nonlinear deterministic process which only appears random because it cannot be easily expressed,” (15, p. 10). Application of this theory revealed that it is possible to identify assets that do not behave purely randomly, but contain a trend factor.

Another method of AI that can be used against EMT and RWT are the artificial neural networks. Unlike statistics, neural networks are capable to learn non-linear, chaotic systems (18). Therefore, in cases when the prices are not purely random, NNs might perform in stock price prediction better than chance. This assumption led to the selection of the neural networks as a method that will be used to solve the prediction problem in this research.

There has been an ongoing debate about the validity of the EMH, and some researchers attempted to use neural networks to validate their claims (23, 26, 12, 15). There has been no consensus on the EMH’s validity, but many market observers tend to believe in its weaker forms. That means it is possible, to a certain extent, to predict the behaviour of the market, and win excess profit.

In this work it is assumed, that, in line with above mentioned criticism, the Efficient Market Theory can be invalid, at least in some cases. Hence, it might be possible to use the historical data to predict the future development.

### 2.3.2 Modern Portfolio Theory

It is useful to briefly discuss Markowitz’s modern portfolio theory since its concept is closely related to one of the principal problems examined in this thesis – the portfolio optimisation.

Early portfolio theory using a statistical concept was first developed in the 1950s by Harry Markowitz, who was later in 1990 awarded a Nobel Prize (22).

Markowitz's great insight was that the relevant information about securities can be summarized by three measures: the mean return (taken as the arithmetic mean), the standard deviation of the returns and the correlation with other assets' returns. The mean

and the standard deviation can be used to plot the relative risk and return of any selection of securities (27) as it is represented on the following figure:

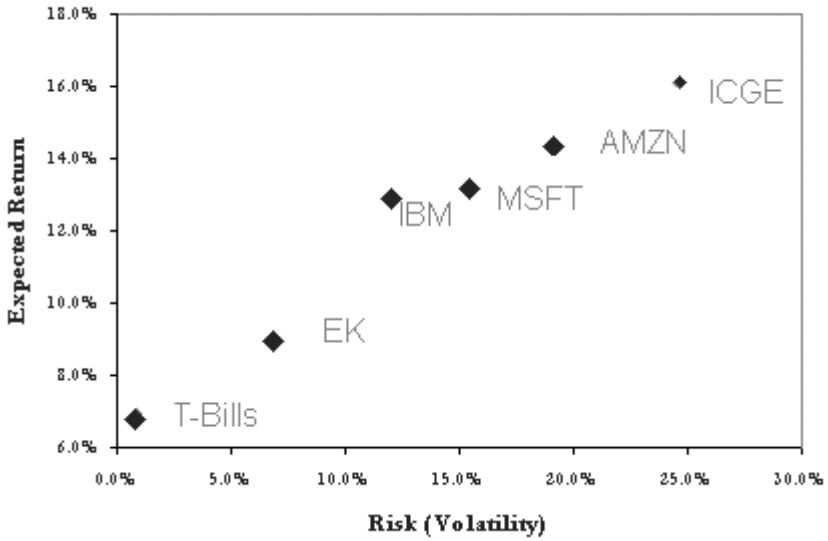


**Fig. 5 - Example of Risk and Return; adapted from (29)**

The common question of an investor is which securities they should invest in. Investor who prefers a low risk strategy would choose T-Bills from the previous figure, while investor who does not care about risk would choose Internet Capital Group (ICGE). There is no one security that is best for ALL investors. Typically, the answer to the investment problem is not the selection of one asset above all others, but the construction of a portfolio of assets, i.e. diversification across a number of different securities (27). According to Elton & Gruber (8), the importance of diversification as of a tool to minimize risk is considered as the major outcome of this theory (8).

Modern Portfolio Theory is a mathematical formulation of the concept of diversification in investing, with the aim of selecting a collection of investment assets that has collectively lower risk than any individual asset. An investor can reduce portfolio risk simply by holding combinations of instruments which are not perfectly positively correlated. In other words, investors can reduce their exposure to individual asset risk by holding a diversified portfolio of assets. Diversification may allow for the same portfolio expected return with reduced risk (30).

It is possible to construct various portfolios formed by every conceivable mixture of selected securities. If we calculate the standard deviation (risk) and the arithmetic mean (return) for each of the possible portfolios, we can identify a set of portfolios which provide the lowest level of risk for each level of return, and the highest level of return for each level of risk. By considering all combinations of assets, a special set of portfolios stand out - this set is called the efficient frontier. For any level of risk, the efficient frontier identifies a point that is the highest returning portfolio in its risk class. Likewise, for any level of return, the frontier identifies the lowest risk portfolio in that return class (27).

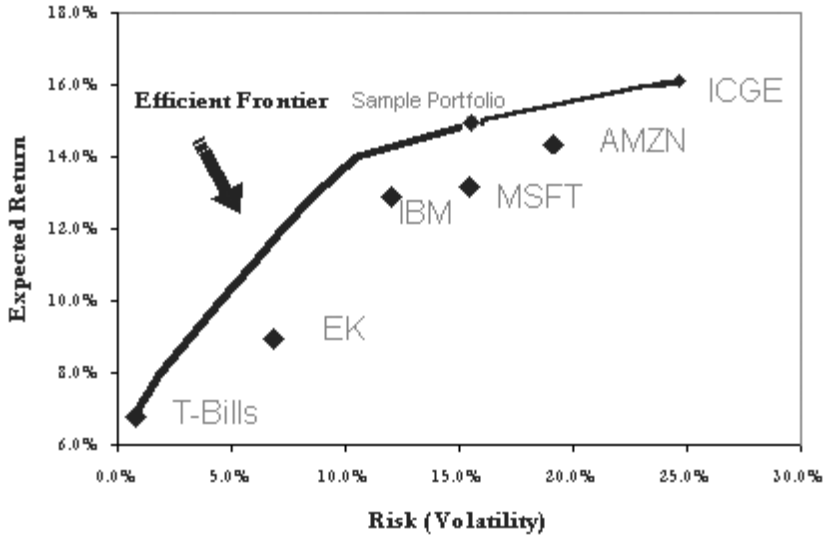


**Fig. 6 - Example of Efficient Frontier; source: (29)**

## *2.1 Stock Market Analysis Methods*

The following stock market analysis methods are the most commonly used tools to analyse and, in particular, forecast the market development. It is useful to provide an overview of these concepts because some of their results might serve as a benchmark or input for stock market applications of artificial intelligence methods. It is interesting to note that these methods, since they are used to predict the development on the markets, stand in opposition to the EMH and RWT.

### 2.1.1 Fundamental Analysis

Fundamental analysis is the most complex stock market analysis method. It involves an in-depth analysis of company performance in order to determine intrinsic value of its shares. This method assumes that current and future price of a share depends on its intrinsic value, which is calculated after a detailed analysis of the company, its industry, and overall economic conditions. If the intrinsic value of a share is higher than the market price, the share is undervalued. Likewise, if the intrinsic value exceeds the market price the share is overvalued (20).

Main advantage of fundamental analysis is its systematic approach and its ability to predict changes before they show up on the charts (15). The disadvantage, on the other hand, is that this method is relatively subjective. Each analyst has different knowledge and experience and can interpret various facts differently (20). Due to this fact, it might be difficult to formalise this knowledge for purposes of automation. This is also a reason why this analytical method was not widely used in the practical part of this research.

Fundamental analysis is often presented as a method with good results for long-term prediction, whereas it has none or limited significance for short-term analyses (19, 20). Hence, its utilisation for the short-term prediction problem in this research is very limited. On the other hand, it might be, theoretically, used as one of the methods to provide inputs for the portfolio optimisation task. However, it was finally not applied due to the fact that it is difficult to find a high quality public source of fundamental analysis data.

### 2.1.2 Technical Analysis

Technical Analysis is a method, which is based on belief that ‘history repeats itself’. It is used by approximately 90 % of major stock traders, mostly for future price prediction (23). This method is, unlike the previous one, considered to be a tool for short-term prediction (20).

Technical analysts try, among others, to identify trends and patterns in historical data. These patterns can then be used to predict future changes in prices of securities, and to

determine best timing for their trading. This tool uses not only basic analysis of plain historical data, such as prices and volumes, but also encompasses construction of various technical indicators. These indicators and their combinations might then be used to indicate trading signals (20, 15).

Main advantage of technical analysis is that it can be highly automated, especially in case of technical indicators. On the other hand, some of the technical analysis methods (e.g. the Elliot Wave Theory) do not provide universal results because each analyst might identify different patterns in the same chart (20).

Due to the possibility of automation, the results and tools of technical analysis, especially the technical indicators, which are considered as universal, objective ratio, commonly serve as input data for NN-based forecasting models (23, 12, 13).

### 2.1.3 Other Analytical Methods

Number of other methods that can be used to analyse the development on the markets was described in literature. From psychological analysis theories (20), or traditional statistical approach (12, 31), to sophisticated technological methods that were developed thanks to the advancement of computer technology. Various forms of artificial intelligence, especially the neural networks, are amongst the most widely cited ones (15, 3, 13). Several applications, described by e.g. Van Eyden (23), Yoon (26), or Gately (11), revealed that these new methods can be very successful in stock market prediction and are therefore suitable to solve this kind of tasks.

# 3 PROBLEM ANALYSIS AND CURRENT SITUATION

The purpose of this chapter is to analyse the problem of stock market prediction and portfolio optimisation, both in general context and with focus on NNs and GAs. Firstly, the company which plans to utilise the results of this research is introduced along with their investment plans. Secondly, the problems of stock market prediction and portfolio construction are analysed. This analysis covers not only the challenges, but also current findings on how these challenges were solved by other researchers.

## *3.1 Company overview*

BELTREX, spol. s r.o., is a small enterprise which was established in 1993. It is a family business based in Brno, Czech Republic, and wholly owned by two Czech citizens. During its existence, its activities ranged from construction machinery wholesale at the early stage to IT services and web application development in recent years.

The management of the firm used to keep all their free cash-flow on the current account, or in a form of term deposit. However, during the recession in recent years the interest rates of Czech banks fell rapidly and this strategy became unsatisfactory. For instance, the interest rate on BELTREX business account is currently only 0.01 % p.a. Consequently, the management decided that part of their free cash could be used for speculative purposes in order to increase its value.

The owners of the business decided that the free funds should be placed on capital markets because they offer interesting possible returns in comparison to deposit banking products. One of the requests is that all trading activities will be performed and managed by the firm itself. This is because one of the owners has a negative experience with externally managed investing.

The amount which the firm wishes to invest is 50,000 USD. This should be split in two halves. 20 % of this amount will be used for possibly high risk – high return short-term daily speculation on stock markets. The remaining 40,000 USD will be used for a

longer-term portfolio investment. The investor is rather dynamic, willing to undergo a higher risk in exchange for high return.

In relation to the plan outlined above, the author of this research was asked to recommend and develop a model for short-term stock market prediction, and a model for optimal portfolio construction.

## 3.2 Stock Market Prediction

In this section, the issue of markets predictability is firstly discussed. Secondly, currently used prediction methods are presented, and, finally, specific problems related to NN prediction models development are examined.

### 3.2.1 Predictability of the Markets

Predictability of the stock markets is an issue that was examined by numerous scientists but a consensus still was not found. The supporters of Efficient Markets Theory (especially in its strong form) and the Random Walk Theory believe that the development of the markets is unpredictable. On the other hand, for instance the Behavioural Theory of finance says that this is possible because people do not always act rationally (9), i.e. this negates one of the Efficient Market Hypothesis' basic assumptions.

One way or another, it is definitely easier to prove that it is possible to predict the markets than to prove that this is not possible. Murphy (19) presented one interesting argument which can be reformulated this way: 'the fact that many people are unable to forecast the markets does not mean that they are not predictable'.

Apart from the theories, there were numerous practical attempts to forecast the stock markets, which proved that it is, to lesser or greater extent, possible (23, 12, 13, 15). In accordance with these experiments, in this research it was anticipated that the EMT and RWT are invalid.

On the other hand, during the literature review no experiment with absolute reliability was identified. The respective scientists rather presented numerous problems and limitations of their findings. The most often cited constraints include the following:

- No prediction method proved to be 100% reliable. There were models with less than 50 % (11) to over 90 % (23) success rate in market prices prediction, but none with absolute reliability over longer period.
- No prediction method proved to work for all assets. Certain forecasting model can work for one stock but can provide unsatisfactory results for another stock, or even for a different type of financial security (26).

These facts can be explained using a time-series description framework presented by Dostál (7). According to this framework, the time-series has two parts, a deterministic one, which can be predicted, and a stochastic one, which is completely random and therefore unpredictable. Hence, when a research is carried on an asset with strong stochastic part, the prediction results will be comparatively worse than those of a forecast done for a stock with weaker stochastic part.

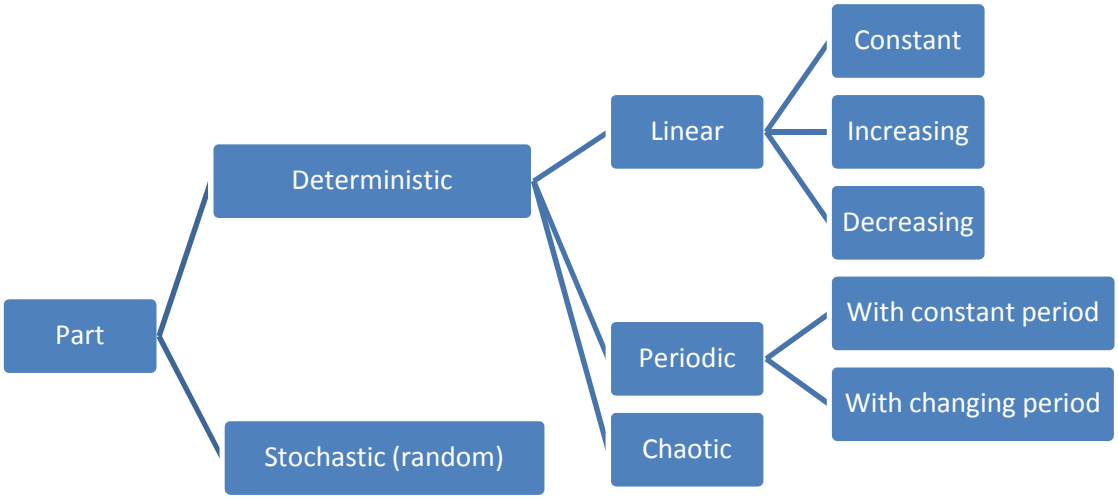


**Fig. 7 - Parts of a time-series; adapted from (7)**

To conclude these findings, it will be anticipated that markets are predictable and a method to forecast the future might be found and applied in reality. On the other hand, it is realistic to expect results which will have certain constraints. It is also highly important to stress that there is no guarantee that such method will work perfectly for an

infinitely long time period in the future, especially because the economic conditions are constantly changing. Hence, at some point it might be necessary to adjust or completely rebuild the model.

### 3.2.2 Current Situation in the Field of Market Prediction

Although it is virtually impossible to determine what are the currently most popular methods for prediction, it is estimated that fundamental and technical analysis belong to the most widely used ones (15). From author's personal experience, technical analysis is the most popular method used by investment analysts for short-term predictions, whereas fundamental analysis is used for a longer scope. In investment recommendation and analysis papers of the largest world banks such as Citi, UBS, Industrial Bank of China, and others, a combination of these two methods is very common (19, 15, 36).

But are the other methods mentioned in this thesis also used in real world applications? And if yes, to what extent? No satisfactory answer based on empirical research was found in literature. However, we can suppose that the other methods are either not widely used, or that nobody wants to reveal that they are using them. Since a sound prediction method can result in exceptional profit, we can expect that no company would like to share such tool, which might be, in fact, a source of their competitive advantage.

### 3.2.3 The Use of Neural Networks for Stock Market Prediction

Even though it is difficult to assess if and how much the Neural Networks are used for real-world trading purposes, it might be useful to present the results of various researches. These can be used to evaluate whether NNs are of any use for the real market prediction tasks.

As it was mentioned in previous chapters, several authors, such as Galushkin (10) or Munakata (18), present the neural networks as a tool capable to predict the development on stock markets. In addition to those general statements, many researchers attempted to test this ability in practice.

Mizuno et al. used NNs to predict the Tokyo stock exchange prices with overall prediction success of 63 % (cited in 15). Phua et al. applied a hybrid model of NNs and GAs to predict price movements on Singapore stock exchange and predicted the direction of price change with 81 % accuracy (cited in 15). Probably the most successful model identified during the literature review was a widely cited JSE-system by Van Eyden (23). This model was able to predict the market movement trend correctly in 92 %. The latter two examples are, however, rather extremes. Most NN prediction models described in literature had an overall success rate in a range of approximately 58 % to 67 % (12, 13, 15). The results of most of these models were also compared to the results of other investment methods in simulated trading environment. Usually, the NNs models outperformed statistical methods, buy-and-hold strategies, and random trading (34, 23, 12).

The common finding of the above mentioned researchers is that the success of the model depends on many variables. The fundamental ones are presented in the following sub-section.

#### *3.2.3.1 Key Challenges of Neural Networks Application*

The first problem that needs to be solved when trying to construct a NN prediction model is to find a suitable architecture of the neural network. There are various types of networks with many possible topology combinations. According to Gately (11), basic features which need to be specified are the following:

1. **Network type and learning function**

   Not all network types are suitable for all types of tasks. Generally, the most widely recommended for the prediction purposes is the backpropagation network (18, 7, 11).

2. **Number of hidden layers**

   Although the backpropagation network can contain any number of hidden layers, Stuart & Norvig (21, p. 593) claim that "with one (sufficiently large) [hidden layer], it is possible to represent any continuous function of the inputs; with two layers, any function at all can

be represented." On the other hand, for instance, Tseng (34) constructed a working prediction model with ten hidden layers.

3. **Number of neurons in hidden layers**

   Sufficient number of neurons is necessary for the training of the network. But how much 'sufficient' is, is difficult to determine. Generally, the more the input data is used, the more neurons are needed. On the other hand, too many neurons can lead to overfitting[1] and rapid increase in computation time (21).

4. **Transfer function**

   Transfer functions that seem to work best for stock market prediction are sigmoid and tangent functions (18). Note that this is valid for back-propagation networks, with other network types, other functions might also give good results.

Another problem is to indentify appropriate input data. Some authors used just historical closing prices, such as Dostál in one of his models (7). Others added opening, maximum and minimum daily prices, and daily volumes (25). And finally, some even used tens of technical and fundamental analysis indicators for given security, prices of most important commodities, prominent financial indexes, or data of fundamental foreign exchange currency pairs. For instance, Van Eyden (23) used more than sixty various inputs in his model for prediction of Johannesburg stock exchange index development. His model was able to predict the trend correctly in 92 % of cases. Although it seems that it is not possible to generalise the relationship between the number of inputs and the results of the model, Van Eyden's research proved that selecting good inputs is a fundamental condition that needs to be met in order to obtain very good results. However, to determine which inputs are 'good' is a very difficult task (21).

Related to the question of input data selection, it is also necessary to determine the number of input samples for training. Some researchers developed valid models which

[1] Overfitting is a state when the network is able to memorize all the examples by forming a large lookup table, but will not generalize well to inputs that have not been seen before (21).

used data of last 500 or even 1000 days (25), whereas others used data of only past three months (34). Those who suggest using long time periods claim that longer time-series contain higher number of occurrences of various patterns (15). Hence, the network can identify all these patterns and identify them during the prediction. In contrary, researchers who used shorter period of data suggest that these data do not contain patterns which are not valid any more (7). Either way or another, longer time periods and higher number of data types result in increased demand for computation time.

The particular asset on which the prediction is carried might be also an important factor for the success of the experiment. Most of the above cited researchers determined for what asset they want to predict right at the beginning of their examination. This is because certain models can work for one security but do not have to give satisfactory results for another one.

To conclude, there are many possibilities how the neural network prediction models can be constructed because there are many variables to be determined. Based on the research results in this field, it is difficult to make any generalisations since the findings and approaches of various scientists were often contradictory. As several of them stated, the determination of 'right' combination of inputs, network architecture, and predicted asset, is usually result of trial-and-error experimentation (18, 21).

## 3.3 Portfolio Optimisation

At the beginning of this section, the problem of diversification and optimisation is briefly outlined. Secondly, key questions an investor has to address when diversifying/optimising their portfolio are presented. Finally, the use of Genetic Algorithms for such task is discussed.

The portfolio optimisation task arises from the request for diversification of various assets. The need of diversification is explained in the theoretical part of this research and is based on Harry Markowitz's Modern Portfolio Theory.

The goal of the portfolio optimisation task is to choose the best portfolio out of all possible combinations of assets. Since there are only two factors describing these

possible portfolios, return and risk, it seems to be a trivial task. However, we have to bear in mind that from *n* assets it is possible to construct an infinite number of various portfolios – the weights of each item of *n* can range from 0 % to 100 %. Therefore, it is virtually impossible to simply calculate all these portfolios, and finally pick the best one. In such case, a method capable of solving such problems without exploring all the possible combinations needs to be applied.

### 3.3.1 Key Issues of Practical Portfolio Diversification and Optimisation

When trying to construct a new portfolio, the investor needs to address two fundamental questions (8):

1. **Into which securities do we want to invest? From pool of which assets shall we select the securities that will form the final portfolio?**

   Since it would be tremendously difficult to assess all securities that are currently traded on world stock exchanges, the investor should streamline their selection and define a pool of all assets that might be considered when searching for the optimal portfolio.

   Various criteria might be used in order to generate such pool, e.g.:

   - the general type of asset (stocks, bonds, options, ETFs, etc., or mixture of two or more of these),
   - geographic location (US assets, securities traded on emerging markets, etc.),
   - liquidity (in line with the corresponding section in theoretical part of this thesis, investor should prefer more liquid assets over less liquid ones to minimise additional risk),
   - associated transaction costs (investor should naturally prefer lower transaction costs, which depend mainly on the type of asset, stock exchange where it is traded, currency rates, and the broker used to execute the trade).

2. **What profit do we want to earn, and what risk are we able to undertake?**

   According to Wang (35), the investor firstly needs to decide if they want to:

   - minimize risk for a given expected return (this strategy is suitable for more conservative, risk averse investor), or
   - maximize expected return for a given risk (this strategy is suitable for more dynamic, return-oriented investor).

   In agreement with that decision, they should consequently quantify what risk they are willing to undertake, or, alternatively, what is their target return. This also strongly depends on the type of assets in the initial pool.

After discussing these questions with the owners of the company, they decided to invest into most liquid US blue chips. The liquidity in this sector is sufficient and the selection generally reflects investor's dynamic profile. They already have an investment US dollar account at InteractiveBrokers, a broker company well known for their low commissions. However, if possible, the model should be 'asset-independent', i.e. it should be possible to adjust it to work with any type of assets.

The investor also decided that they will set a fixed target return for the portfolio. Therefore, the task of the optimisation model will be to generate a portfolio with minimal risk for that given return.

Since the conditions on the markets are changing rapidly, the management of the company wants to use the portfolio optimisation tool to update their portfolio on a regular basis in order to reflect these changes.

The last request was that the investor should be able to define the number of assets in the final portfolio. This is to keep the portfolio rather smaller in order to allow for simple administration and management of the portfolio, and to keep the transaction costs low.

### 3.3.2 The Use of Genetic Algorithms for Portfolio Optimisation

Since Genetic Algorithms are suitable for solving optimisation tasks efficiently, it is possible to employ this method also for complicated portfolio optimisation problems. For instance Dostál (7) uses GAs to construct optimal portfolio, which will be strongly correlated with NASDAQ index. For this task, the portfolio was constructed from only eight stocks. Several authors such as Lin & Gen (16), or Vedarajan et al. (24) also carried their GA-based researches on similar portfolio optimization problems to the one described in this research. The results of their models verified the possibility of practical utilization of Genetic Algorithms for portfolio selection problem.

#### *3.3.2.1 Fundamental Questions*

The portfolio selection problem basically means that we need to determine:

1. **Which securities from the initial pool of assets will form the final portfolio, and which will be excluded?**

   This is a combination problem known as 'N choose K', where K items can be selected from the pool of N items. K represents number of assets in the portfolio set by the user. To illustrate the nature of the problem on a simple example, if we want to choose five stocks out of twenty, we have more than 15,000 possible combinations how this selection can be done. If we wanted to employ traditional 'brute force' methods, we would need to calculate variance matrices and optimal weights for all these combinations. With higher N and K, the number of combinations grows rapidly and would be highly computation demanding.

   Since this problem can be represented by a binary number (1 = include the particular stock in the portfolio, 0 = exclude it), GAs seem to be a suitable solving method.

2. **What will be the weights of the respective securities in the final portfolio?**

   If we determine which stocks might be included in the portfolio, the calculation of their appropriate weights can be efficiently performed by utilisation of traditional methods such as quadratic programming. Such traditional methods are useful once we have selected which equities will be included in the portfolio (32).

During the implementation in Matlab, it will be necessary to set several parameters of the GA model, most importantly we will need to define the size of the population, the fitness function, how the initial population will be generated, what functions will be used for selection, crossover and mutation, etc.

Furthermore, as in case of the NN prediction, we need to define the input data for the optimisation model. To calculate risk and return, closing prices are sufficient. Since the request of the investor is to make the model flexible, it should be able to work with data of a reasonable number of assets of any type. Consequently, it is also important to set the time period of the input data. For example, Lin & Gen (16) used a period of past three months, similarly Bradová (2) used 2-month past data to construct a portfolio updated on a monthly basis. Based on these examples, the relation of the input data length and the period of portfolio updates can be observed. Generally, for portfolios updated monthly the input time window was about 2 - 3 months, for portfolios updated on a quarterly basis data for past 9 – 12 months were usually used.

To conclude this subsection, we will try to construct a GA-based model, whose task will be to find minimal risk portfolio of K assets. This portfolio will be selected from a larger universe of N securities, initially US blue chips. The number of securities included in the monthly-updated portfolio will be set by the user and the whole model should be asset-type independent.

# 4 PROPOSALS AND CONTRIBUTION OF SUGGESTED SOLUTIONS

In this chapter, the solution of the problem outlined at the beginning of this thesis is presented, as well as specific contribution of this solution. It is organised into two subchapters, in the first one, the stock market prediction model is proposed. In the second one, the portfolio optimisation model is presented. Both models were created in Matlab development environment using the official documentation (33).

## *4.1 Stock Market Prediction*

For the stock market prediction task, a computer NN-based model was developed in Matlab. As described in the Goals, it is intended to be utilised for one-day-ahead prediction.

In this section, the solution is firstly generally described to provide an overview of its general concept. Consequently, the most important parts of the source code of the Matlab application are commented. Finally, actual results of the best performing model are presented, followed by important general remarks and limitations of the solution.

### 4.1.1 General Description of the Solution

Before the actual source code of the final model is presented, it is useful to describe its overall concept and the development process in more general way.

The development process itself included several fundamental stages as outlined in the previous chapter. Specifically, it was necessary to:

1. Determine what we want to predict
2. Choose an asset for which the prediction will be performed
3. Define input data types
4. Define the length of input time-series
5. Determine how to measure and test the results of the prediction
6. Find the optimal architecture of the neural network

These steps were performed sequentially. Especially the third and sixth point was then performed in several iterations. The description of these stages follows.

#### *4.1.1.1 What We Want to Predict*

Following the problem definition, the solution was designed to predict the closing price of the following day. Comparing particular day's actual opening price and the predicted closing price, we can determine the predicted trend, and speculate on rise or fall of the security.

#### *4.1.1.2 Asset Selection*

According to the general finding extracted from the literature, it is useful to initially choose an asset for which the prediction will be done. Since a NN model 'tuned' for a particular equity does not have to provide good results for another one, we will stick to only one selected asset. The model was optimised and tested for this particular equity, exactly as it was done in other researches (11, 23, 25, 26).

S&P 500 index was selected because it reflects the development of 500 stocks of US blue chips. It is therefore relatively tolerant to potential strong unpredictable fluctuations of one particular stock. The choice of a large index was also partly inspired by Van Eyden's research (23) which was successfully carried on an index as well. Since it is not possible to trade the index itself, a derivative reflecting the development of the index was used in the model. Specifically, SPDR S&P 500 ETF (ticker SPY) traded on the New York Stock Exchange was used.

#### *4.1.1.3 Input Data Definition*

Due to the requirement that the model should be maximally automated and easy to use, the input data needs to be easy to acquire from publicly available sources, without any additional input from the user.

In agreement with other researchers in this field (23, 25), historical data, namely open, high, low and close prices (OHLC), and daily volumes, of the selected asset was used. This data was obtained from Yahoo! Finance (36) via the Matlab Datafeed Toolbox. In addition, several indicators of technical analysis were employed because several researchers utilised these input data in the past with success (34, 23), so that they proved

to be suitable for this task. Since Matlab Financial Toolbox has standard methods to calculate many common TA indicators it is very easy to obtain them. Namely, several experiments during the development were carried with simple and exponential moving averages (SMA and EMA), relative strength index (RSI), moving average convergence divergence (MACD), stochastic oscillator, and Bollinger bands, along the volume, open, high, low, and close prices of the selected asset. For exact definitions of these indicators, please refer to Rejnuš (20).

Several combinations utilising the input data types described above were tested. Some combinations included various subsets of all the data types described above, whereas other combinations used all of them. Moving Averages, RSI, and Stochastic Oscillator were tested with various periods used for their calculations, e.g. moving averages with periods from 5 to 20 days were utilised.

| Input data | Success Rate (%) | Absolute Profit/Loss ($) | Equity curve development |
|---|---|---|---|
| **OHLC, Volume, SMA (10), EMA (10), RSI (14), Stochastic Osc. (5), MACD, Bollinger B. (20)** | 74 | 27.14 | Relatively steady with 2 larger draw-downs |
| **OHLC, Volume, SMA (10), EMA (10), RSI (14), Stochastic Osc. (5), MACD** | 64 | 18.14 | Growing, but very unstable |
| **OHLC, Volume, SMA (5), EMA (5), RSI (7), Stochastic Osc. (5), MACD, Bollinger B. (20)** | 67 | 18.6 | Growing, but very unstable |
| **OHLC, Volume, SMA (15), EMA (15), RSI (14), Stochastic Osc. (5), MACD, Bollinger B. (20)** | 72 | 27.94 | Relatively steady with several draw-downs |

**Fig. 8 - Sample results of tests to identify best input data**

Number of tests was performed in order to identify the best input data set. The best overall results were obtained using OHLC, daily volumes, SMA and EMA with period of 10 days, RSI with period 14 days, Stochastic Oscillator with standard 5 days period, Moving Average Convergence Divergence, and Bollinger Bands with standard period

of 20 days. Several result samples from tests performed on a feed-forward backpropagation network with ten hidden layers, and 100 neurons in each layer, are presented in the previous figure.

#### *4.1.1.4 Input Time-Series Length Definition*

On how many samples should the NN be trained? Since opinions of various researchers differ, there is no best answer to this question. Some prefer short time-series of about two to three months (34), some prefer hundreds of days (25).

Due to the fact that the computation demand rises drastically with the length of the input time-series, shorter periods were preferred in this research. This is also because the final user does not own any exceptionally powerful computing machine that could be utilised for the practical application. As suggested by Tseng (34), historical data of past three months were used for experimentation. To be more precise, historical data together with related indicators for last 60 days were used as input for the neural network.

#### *4.1.1.5 Definition of Measurement Indicators and the Testing Process*

Because the final model should be usable for real trading purposes, the measurement of its results in the testing phase needs to be understandable to any ordinary trader. Furthermore, the results need to be comparable to other prediction and/or trading models and strategies.

For the sake of simplicity, transactions costs, taxes, and other additional fees were excluded during the development and testing of the model, similarly as in most other cases identified in literature (11, 12, 23, 34).

Following Tseng's (34) approach, the model was constructed on the basis of moving time window. I.e. every day a new neural network is constructed and trained on data of past 60 days. When testing the model, it was simply run $n$-times on data moving forward by one day. The output that was afterwards evaluated was formed by $n$ subsequent predicted closing prices. This method ensures that the network is always trained on 'fresh' data of the same length.

The evaluation of predicted results was designed to simulate real trading. Depending on the predicted closing price and the actual day's opening price, either long or short position is taken. This position is then closed at the closing price of that particular day. When such trading is simulated on *n* subsequent days, the success rate of the prediction is calculated, as well as the profit and loss (P&L) of each day and of the testing period as a whole. To visualise the P&L of the whole period, a chart was used, allowing identification of draw-downs that occurred during the testing.

To compare various versions of the model during the development, i.e. various combinations of NN architectures and input types, a test on each version was run, each time exactly *n*-times. Finally, the best model configuration was selected, depending on the overall success rate, the absolute profit/loss of the period, and the steadiness of the equity curve. The Success Rate indicates in how many per cent the trade position taken was correct.

Due to the lack of computing power, all model testing was done with *n* parameter equal to 50. I.e. all prediction tests were performed for 50 consecutive days, ending July 30$^{th}$ 2010.

#### *4.1.1.6 Optimal Neural Network Architecture Identification*

As suggested by various authors (18, 21), the trial and error experimentation during the design of the NN was employed. This is mainly because the literature review revealed very little generally valid information that could be used when designing such specific neural network.

1. **Network type and learning function**

   Feed-forward backpropagation network configuration was selected as it is most widely recommended network type for prediction tasks.

2. **Number of hidden layers and number of neurons in each layer**

   Numerous tests were run with different numbers of hidden layers and neurons in these layers. The best performing model for the input data selection presented in 4.1.1.3 Input Data Definition was a network with 10 hidden layers and 100 neurons in each layer. Various network

configurations were tested, mostly with 1, 2, 5, and 10 layers, and 5, 10, 20, 50, 100 neurons in each layer. It is necessary to emphasize that there is tremendous number of possibilities how the layers can be configured, since the numbers of neurons do not have to be the same in all layers. Due to the lack of computing capacity, only several tests with arbitrage-defined numbers of layers and neurons were performed. In the tests, each layer had the same number of neurons in it. On the following example, the test results for a NN with 10 hidden layers are presented:

| Neurons per layer | Success rate (%) | Absolute P/L ($) | Equity curve development |
|---|---|---|---|
| **5** | 49 | -2.08 | Extremely volatile |
| **10** | 56 | 23.14 | Very unstable with big drop-downs |
| **20** | 68 | 19.04 | Relatively stable, with several draw-downs |
| **50** | 68 | 16.24 | Unstable with big drop-downs |
| **100** | 74 | 27.14 | Relatively stable, with some minor draw-downs |

**Fig. 9 – Test results for a neural network with 10 hidden layers**

### 3. Transfer function

Following several recommendations in literature, mainly tangent and sigmoid functions were tested as transfer functions. Specifically, *tansig* and *logsig* functions from Neural Network Toolbox were tested. Both provided similar results. Since the *logsig* function did slightly better, it was chosen as a transfer function for the final model.

#### *4.1.1.7 The Final Solution in Practice*

Since the best combination of inputs and architecture was identified, the final model has been constructed. In this subsection, an overview of this model is generally presented on a case study.

When the predictNextDayClose.m file is run, the user is asked to enter the date for which the prediction should be performed. Since the model is constructed to provide prediction for the next day, it is not anticipated that dates further into the future will be provided. The program works correctly with history dates for back testing, and on the other side of the range it is able to provide prediction for tomorrow. Please note, that the prediction for 'tomorrow' can be performed after the 'today's' close of NYSE since the data from 'today' will be used in the model. In this case study, we intended to predict the closing price for 30$^{th}$ June 2010.

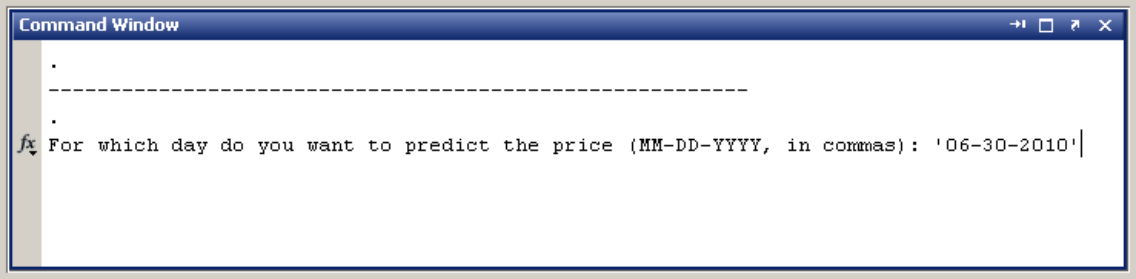


**Fig. 10 - The prediction date dialog which appears after the program is launched**

When the date is entered, historical data are automatically downloaded from Yahoo! Finance server by the *yahooDataFetcher* function. Then, technical analysis indicators are calculated, complete input data is prepared, and the network options are set. Finally, the neural network is trained. During the training, the user can see the progress of the process. In our example, the training terminated after some 153 iterations as it is visible on the following figure.

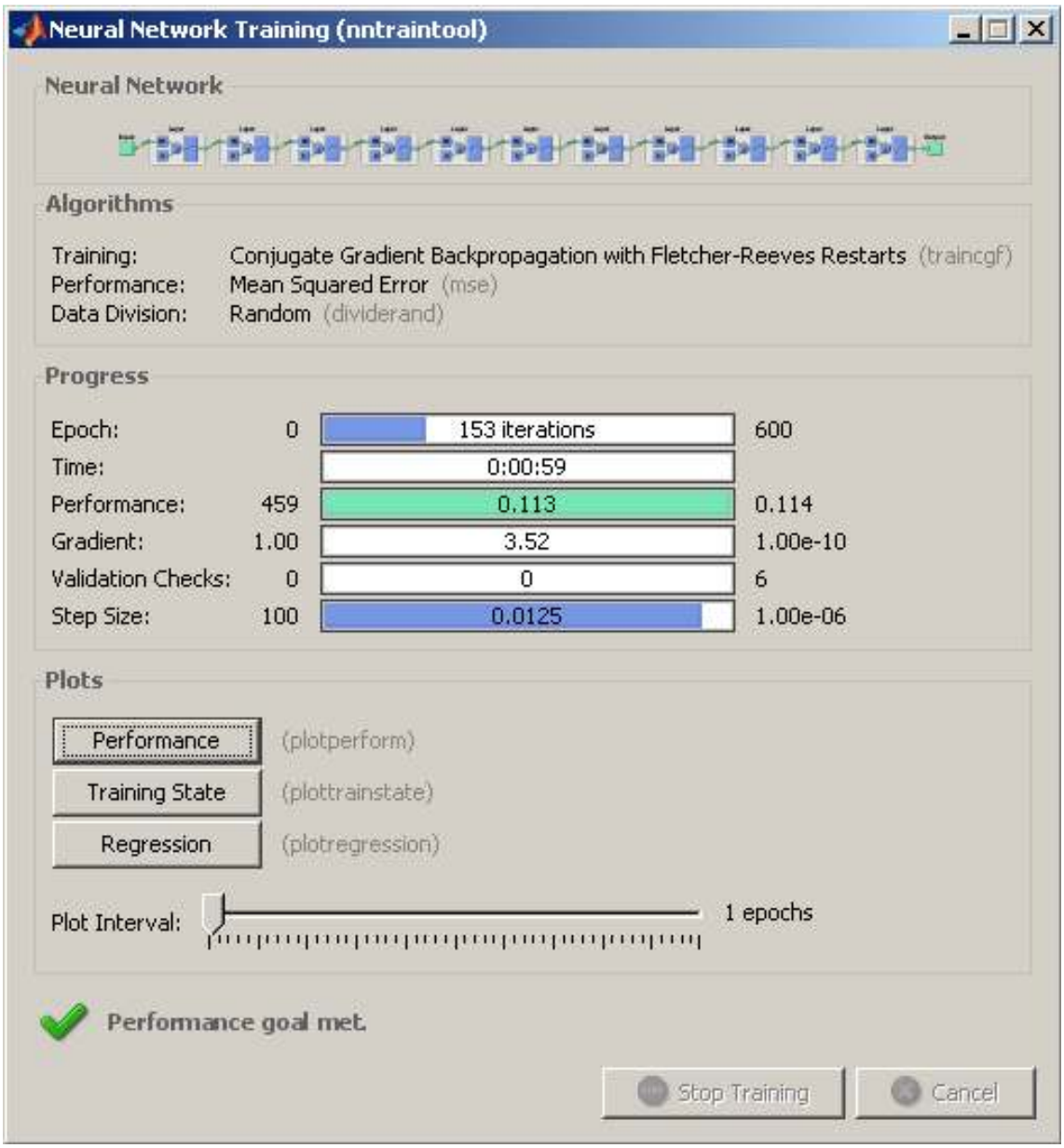


**Fig. 11 - The Neural Network Training Visualisation**

After the training, a simulation is performed by the trained neural network. The output of the simulation, i.e. the predicted prices, is presented along the real historical data for respective dates. Both time-series are visualised on the following figure. The very last point is the predicted price for the next day.

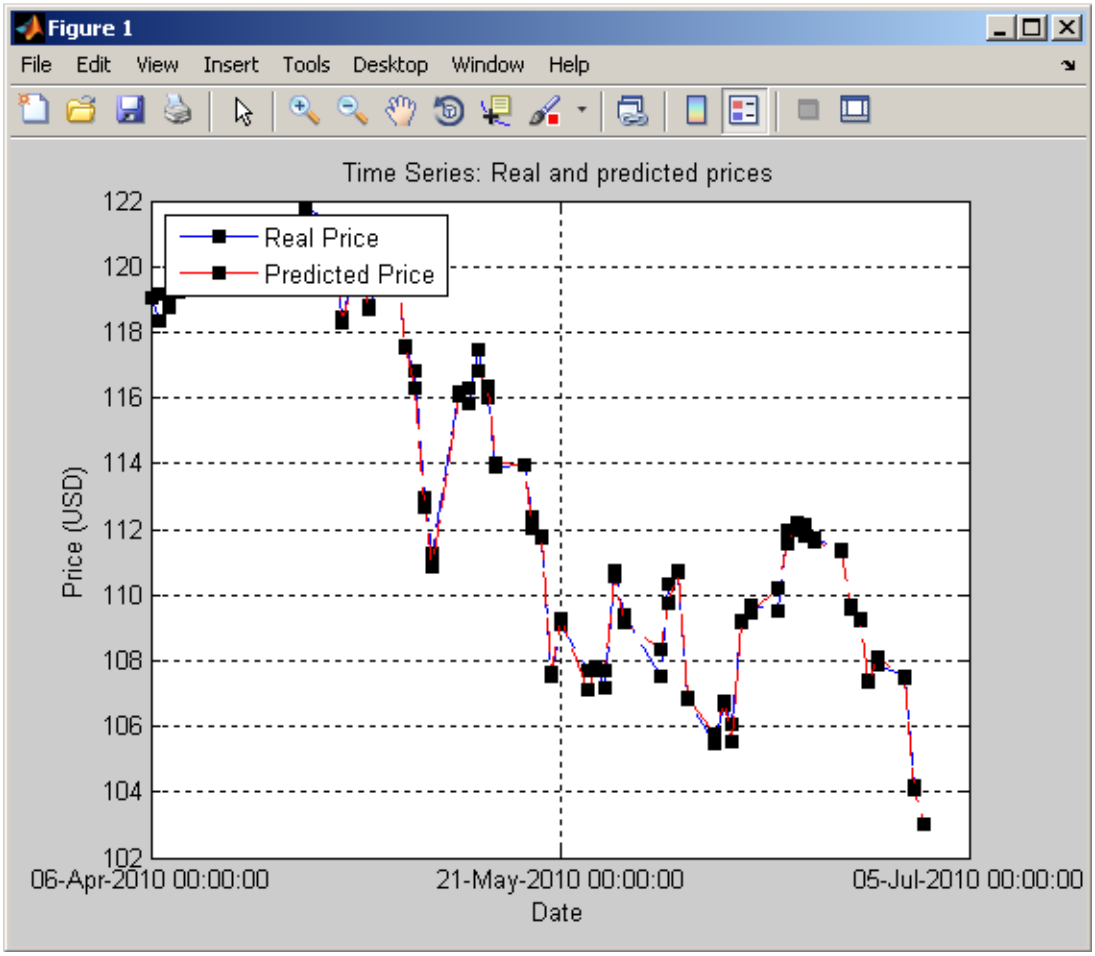


**Fig. 12 - Predicted and real historical prices**

Finally, the desired prediction for the required date is printed into the command window in Matlab. The user is also informed if the following day's closing price is expected to fall or rise.

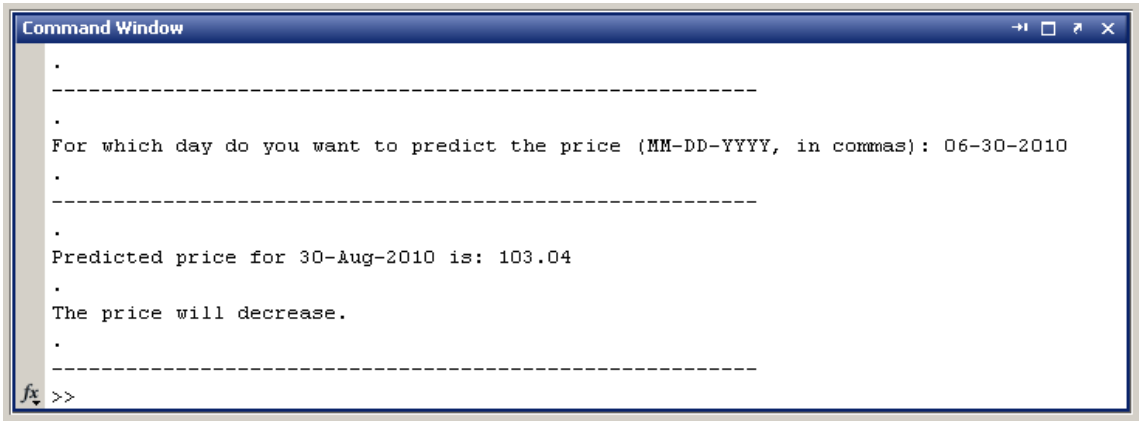


**Fig. 13 - Presentation of the predicted price**

### 4.1.2 The Final Solution in Detail

In this subsection the final solution is presented in detail - key parts of source code are presented and commented. For full source code with comments, please refer to Appendix A.

The program for the NN prediction performs the following fundamental steps:

1. Set basic variables
2. Download data from Yahoo
3. Prepare input data for the network (indicators, truncation to 60 days)
4. Set the configuration of the NN
5. Train the neural network
6. Perform simulation using the trained network
7. Visualise the accuracy of the simulation on a chart
8. Output the predicted closing price for tomorrow

The application itself consists of two files, *predictNextDayClose.m* and *yahooDataFetcher.m*. The first one is the master program, the latter one is a function used to download historical data into the model. The prediction program can be started by executing the *predictNextDayClose* m-file within the Matlab development environment. All user input/output is presented in the command window and in independent visualisation windows.

After the user-defined and configuration variables are set, the basic historical data for SPY ticker are downloaded. From this data, several indicators of financial analysis are consequently computed as it is presented in the following code. Several built-in methods from the Financial Toolbox are used. Specifically, simple and exponential averages with period 5 and 15 are constructed, as well as moving average convergence divergence, 14-day relative strength index, stochastic oscillators, and Bollinger bands.

```
SMA10 = tsmovavg(Close, 's', 15);
EMA10 = tsmovavg(Close, 'e', 15);
[MACD NINEPERMA] = macd(Close);
RSI14 = rsindex(data(1:end,5),14)';
STOSCkd = stochosc(data(1:end,3), data(1:end,4), data(1:end,5), 10, 3,
'e');
STOSCk = STOSCkd(1:end,1)';
```

```
STOSCd = STOSCkd(1:end,2)';
netInputsComplete(end+1,:) = CloseBolling(1,:);
netInputsComplete(end+1,:) = CloseBolling(2,:);
netInputsComplete(end+1,:) = CloseBolling(3,:);
```

Then input and simulation data is prepared, as well as the target vector for the training. Closing prices of the following trading day are used as target values, i.e. the closing prices excluding the first one are used as the target values:

```
netTargetsCycle = netInputs(4, 2:end);
```

Finally, the NN options are set and the network is trained and simulated on data converted into the sequential format as follows.

```
net = newff(netInputsCycleSeq, netTargetsCycleSeq, numHiddenNeurons,
{'logsig' 'logsig' 'logsig' 'logsig' 'logsig' 'logsig' 'logsig'
'logsig' 'logsig' 'logsig' 'purelin'},'traincgf');

trset = 60;
net.divideParam.trainRatio = trset;
net.divideParam.valRatio = (100 - trset) / 2;
net.divideParam.testRatio = (100 - trset) / 2;
net.trainParam.epochs = 600;
net.trainParam.goal = mean(netTargetsCycle) * 0.001;

[net,tr] = train(net, netInputsCycleSeq, netTargetsCycleSeq);

…

Y = sim(net, simInputsCycleSeq);
```

At the very end, the rounded predicted price is printed into the command window:

```
predictedPrice = sprintf('%0.2f',netOutputs(1,end));
…
display(['Predicted price for '
datestr(simDatesForPredictedPrices(:,end),1) ' is: ' predictedPrice
]);
```

### 4.1.3 Results

The best performing model identified during this research was able to predict the trend in 74 % of cases. In the final evaluation of this model, long or short position was opened on SPY ticker for amount of USD 10,000 at particular day's opening price, and closed at the same day's closing price. With this equal daily investment, the model was able to earn USD 2,559, i.e. the return was 25.59 %. In comparison, when the position type

(long or short) was randomly selected, returns oscillated around zero. Specifically, the Random-Walk-based strategy generated loss of USD 80, i.e. its return was -0.8 %. The following figure presents the development of cumulated daily profits/losses. All this data are based on a test performed for the testing period specified in section 4.1.1.5.

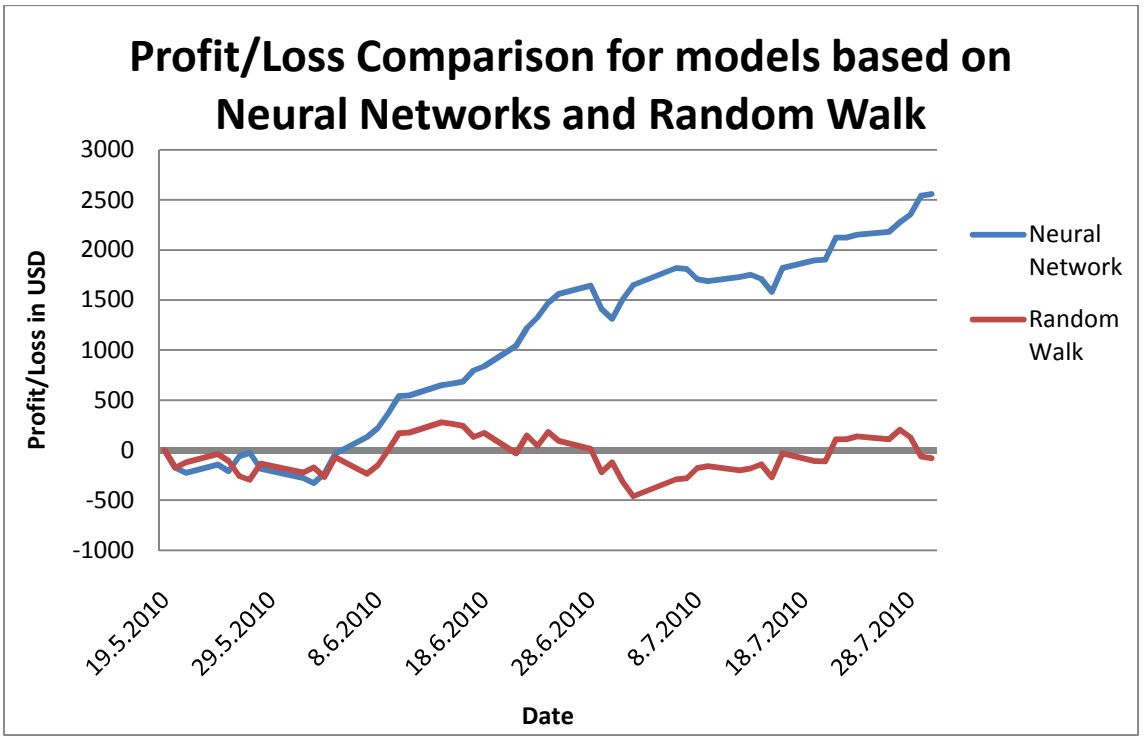


**Fig. 14 - Comparison of Profits/Losses of Neural Network model and Random Walk trading, calculated for USD 10,000 daily investment**

### 4.1.4 General Notes and Limitations

It is necessary to present several important notes and limitations of the solution.

Firstly, it is not possible to guarantee that the model will deliver similar success ratio and profits obtained in the tests for an infinite time. Although the model was tested on historical data, it might provide less satisfactory results in the future due to the dynamic nature of financial markets. They are constantly evolving, new regulations of the markets are planned, and there is number of random factors influencing the development. E.g. the model is not capable to predict the prices when they are influenced by random events, such as terrorist attacks or natural disasters. Hence, the

user needs to observe the performance of the prediction carefully and stop using it when such exceptional influences bias the markets.

Secondly, the solution cannot be considered to be robust enough for use with other securities than S&P 500 ETF. Although it is rather a feature of the model, some can perceive it as a limitation.

Thirdly, the model is able to predict the trend for the following day only. I.e. it cannot be used for longer-term analysis of the markets.

For the sake of simplicity, commissions, taxes and other fees were excluded from the testing and evaluation. This is to allow for better general comparability because the trading commissions might differ at various brokers in various countries and depending on investment amounts.

The model is now ready to be used and further customized in the Matlab environment. This means that the user needs to have the Matlab software at their disposal. However, the application can be compiled into standalone executable file, or a library further re-usable on .NET, or Java development platform, making it suitable for almost any device.

It is also important to bear in mind that the output of the model is only a prediction of the following day's closing price, rather than a strategy. It needs to be incorporated into a complex trading strategy, considering all transaction costs, reactions to exceptional events on the markets, selection of broker and appropriate trading platform, utilisation of stop-loss market orders in order to minimise possible losses, etc. Before one starts to trade 'live', such strategy should be thoroughly back-tested. To back-test on a long historical period is strongly advisable because the model was tested on equal periods of 50 consecutive days only.

Finally, to buy exactly at particular day's opening price and sell at the closing price is not always possible. However, the trading strategy can be fully developed by the user of the model to suit their needs. The only output of the model is the prediction of tomorrow's closing price. How will be this information incorporated in the final trading strategy depends only on the trader.

## 4.2 Portfolio Optimisation

In this section, the solution of the portfolio optimisation problem is presented. Firstly, it is generally described, partly on a case study. Secondly, a more detailed description including the source code is provided. Finally, results, important notes and limitations on the use of the model are presented.

### 4.2.1 General Description of the Solution

Before the actual source code of the application is presented, the concept of the model and the approach towards the development process are presented.

The goal of this program is to construct minimum risk portfolio formed by a subset of a larger universe of assets. The number of securities in the portfolio and the expected return are defined by the user at the beginning of the optimisation task itself. Specifically, we need to select K assets out of N assets. The main requirement is that these K assets need to have minimum risk for the given expected return. Similar problem was already covered by number of authors (1, 16, 24, 32) who used various ways to solve it. Following Rosen's approach (32) outlined in the previous chapter, Genetic Algorithms are used to solve this N choose K problem efficiently, whereas quadratic programming is consequently employed to calculate the weights of particular assets in the portfolio.

The initial pool is formed by N = 100 assets. Top 100 most liquid stocks forming the S&P 500 index were selected for the testing purposes of this research. These equities were chosen manually using MS Excel, the liquidity data for year 2009 was extracted from Yahoo! Finance (36). However, the model is flexible and can work with indices of any kind, including bonds, derivatives, etc. The presented selection is based on high liquidity to minimise additional risk of the portfolio. For the demonstration of the solution, the stocks from S&P 500 were selected so that we can later use the index as a benchmark which can be compared to the recommended portfolio. The figure 100 was selected after some testing in Matlab. An error occurred in Matlab when more than 130 indices were used in the model, probably due to memory constraints. To ensure smooth operation on various computers, the final number was arbitrarily cut down to 100.

The application is intended to run from the Matlab environment. However, same as for the NN prediction model, it can be easily compiled for further use on other platforms as well.

Generally, during the execution of the model implemented in Matlab the following fundamental steps are performed:

1. The user enters user-defined variables
2. Source data are downloaded from Yahoo! Finance Server
3. Basic statistics is calculated from the downloaded data
4. The portfolio optimisation routine based on GA is performed in order to identify the lowest risk portfolio for user-defined target return
5. The final portfolio, which is the result of the optimisation, is presented

In the following subsection, the program is generally described on an example portfolio optimisation task. For detailed information about the technical realisation, please see The Solution in Detail section and Appendix A containing commented source code of the application.

#### *4.2.1.1 The Final Solution in Practice*

When the START.m file is run, the user has to enter the boundaries of the calculation period. For this period, historical statistics used in the model are later computed. The period has to be exactly one year long, e.g. from 1st April 2009 to 31st March 2010. Then, the user specifies the number of equities in portfolio. It should be reasonably lower than the number of assets in the pool from which the portfolio will be constructed. In the example presented on the following figure, the final portfolio will be formed by ten equities. Finally, the required return of the portfolio needs to be specified in per cent.

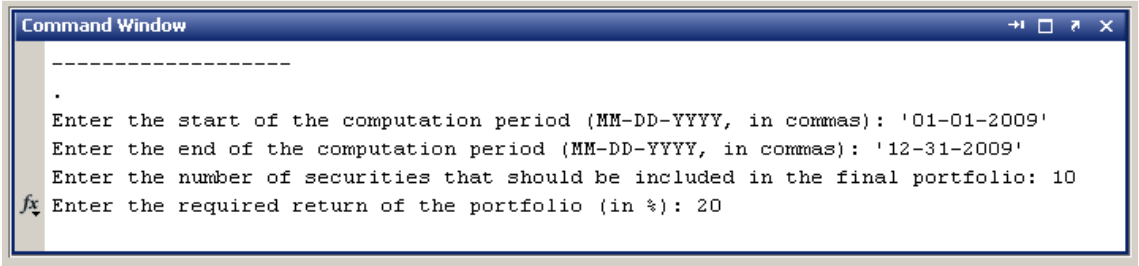


**Fig. 15 - The user variables dialog which appears after the program is launched**

The goal of the solution is to give the final user maximum control over:

- The period of source data.
- The number of equities in the final portfolio.
- The required return.
- The benchmark used for comparison and the contents of the initial pool of assets (please note that for this, a minor update to the code and stocksTickers.mat source variable need to be done).

When the required variables are specified by the user, the program downloads the closing prices for a benchmark (e.g. S&P 500 index; ticker ^GSPC) and all indices specified in *stocksTickers.mat*. In the future, any other benchmark than ^GSPC can be used and the *stocksTickers* vector of initial pool of assets can be customised. It can contain any indices for which it is possible to download historical data from the Yahoo! Finance server.

The downloaded historical data are then used to calculate annual risk and return, and the covariance matrix for all equities. Consequently, these are used as an input for the GA-based optimisation routine. This routine aims to identify a subset of the initial pool of securities. Via GA, various subsets are generated during the evolution process. The subset with the best fitness function (minimum risk) is then selected as the final solution. The number of assets in the final portfolio and the required return are user-defined constraints of the genetic algorithm.

The graphical representation of the evolution process is presented on the following picture. In the upper part, the development of the fitness value during the evolution is visualised. In this case, some 53 generations were needed to reach the best fitness value of 0.13457. In the lower part of the figure, the equities selected for the portfolio are represented. The second, fourth, sixth, etc. stocks were selected, in their respective order as in the *stocksTickers.mat* file.

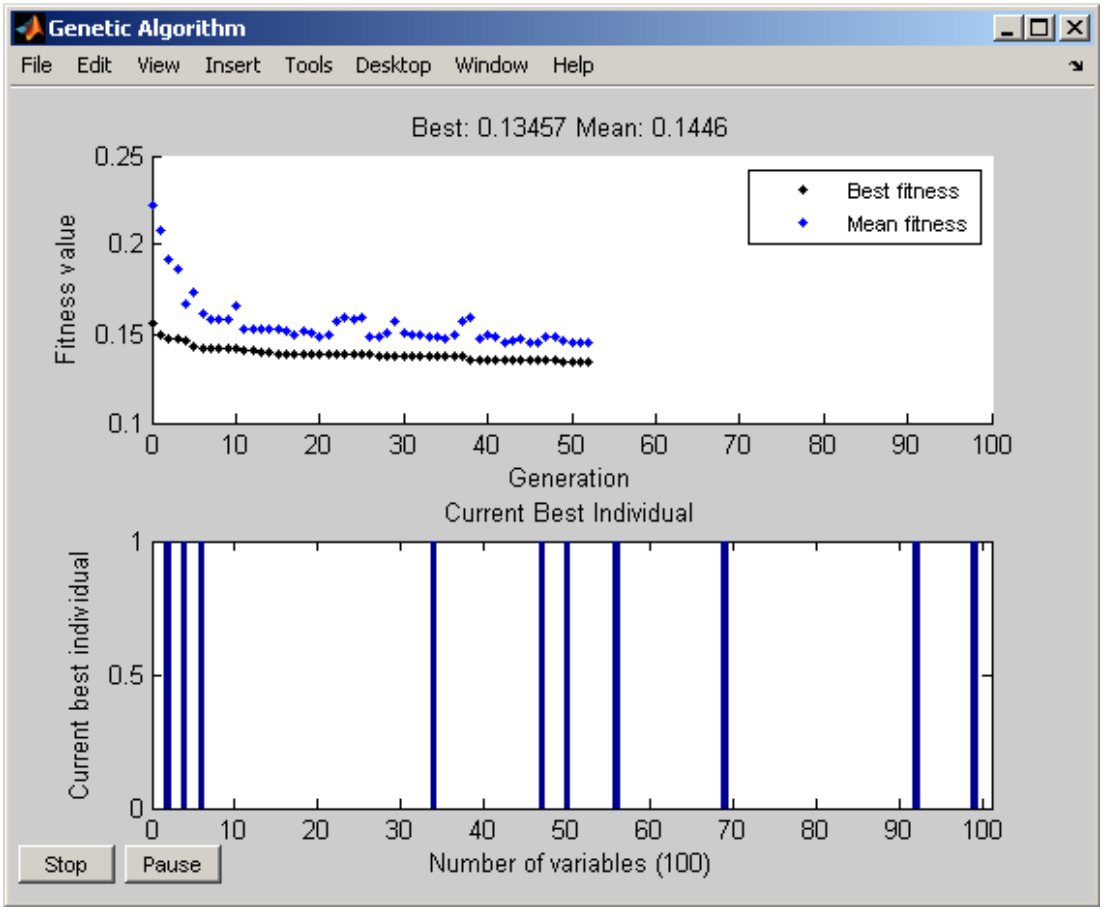


**Fig. 16 - Visualisation of the GA evolution process**

After the termination of the optimisation routine, the results are printed into the command window. The recommended portfolio structure is presented in detail, along the overall statistics for the portfolio as a whole and the selected benchmark. An example of the results screen is shown on the following picture. The portfolio is formed by ten equities as demanded, has an overall return of 20 %, and standard deviation of nearly 13.5 %. It is clearly visible that the recommended portfolio has significantly better performance than the benchmark (S&P 500) in terms of both return and risk.

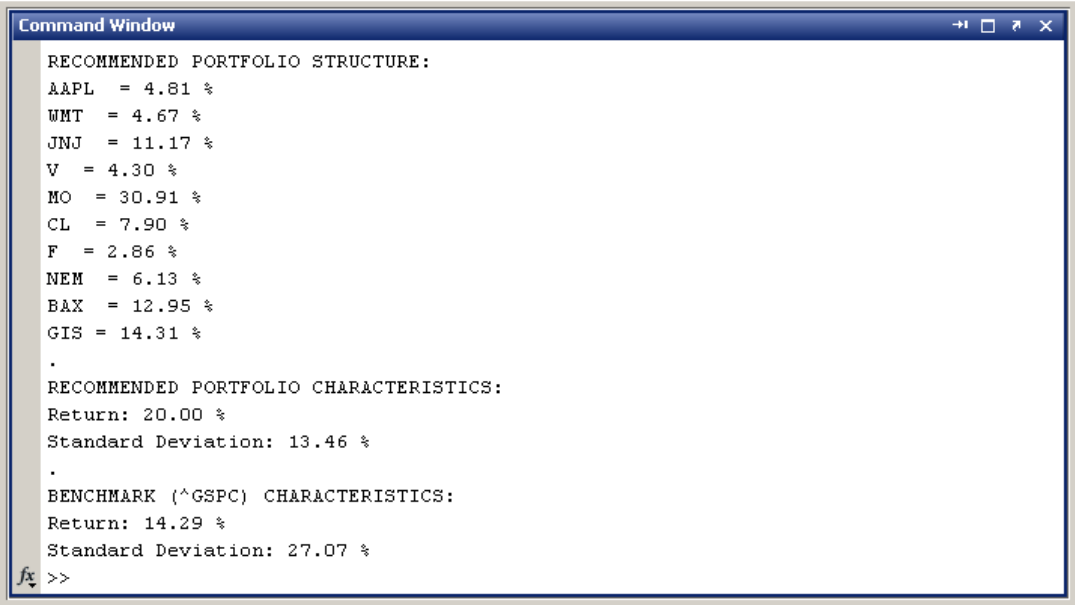


**Fig. 17 - Presentation of the results of the optimisation routine**

Finally, the recommended portfolio is visualised on a graph so that the user can quickly asses its return-to-risk ratio, and compare it to the efficient frontier. On the following example, the recommended portfolio (the red dot) is located on the efficient frontier, i.e. the solution is optimal (or close to optimum) also from the MPT point of view.

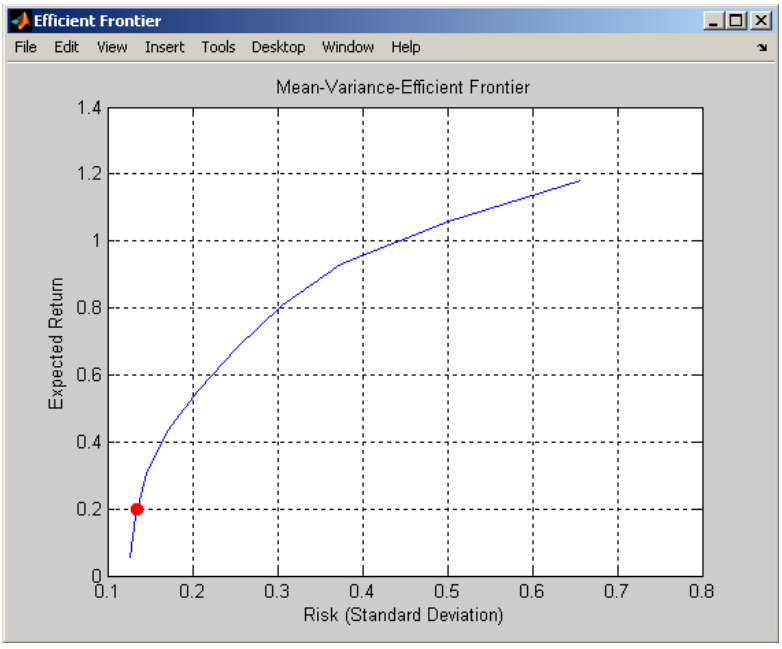


**Fig. 18 - Visualisation of the recommended portfolio and the efficient frontier of the source pool of assets**

In case that the user is not satisfied with the results of the recommended portfolio, the program can be re-run with different user variables, e.g. the target return or the number of equities in the final portfolio can be adjusted.

### 4.2.2 The Solution in Detail

In this section, specific m-files, functions and their parts forming the whole program are described in more detail.

On the following picture, the dependencies of all m-files in the application are presented. They are ordered according to when they are called within the program.

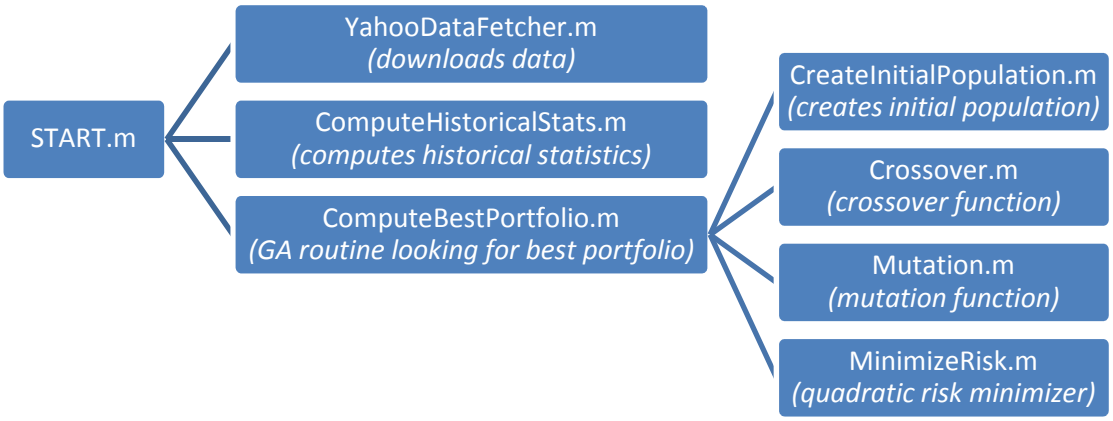


**Fig. 19 - Scheme of Function dependencies**

The following START.m file is a wrapper for the whole portfolio optimisation program. It obtains user-defined variables, loads data via *YahooDataFetcher* function, computes historical statistics via *ComputeHistoricalStats*, runs the GA optimisation routine *ComputeBestPortfolio* to identify the best portfolio, and finally outputs the recommended portfolio weights, along with computed overall statistics for both the portfolio as a whole and the selected benchmark. At the very end, the portfolio is rendered on a chart to visualise its position related to the efficient frontier calculated for the source pool of equities.

```
display('--------------------');
display('.');
fromDate = datenum(input('Enter the start of the computation period
(MM-DD-YYYY, in commas): '));
toDate = datenum(input('Enter the end of the computation period (MM-
DD-YYYY, in commas): '));
numOfStocksInPortfolio = input('Enter the number of securities that
should be included in the final portfolio: ');
targetReturnOfPortfolio = input('Enter the required return of the
portfolio (in %): ');

load stocksTickers;
benchmarkTicker = '^GSPC';

assetSymbols = [benchmarkTicker stocksTickers'];
[data dates assetSymbols] = YahooDataFetcher(assetSymbols, fromDate,
toDate);

[annualReturn,annualCovariance,annualStandardDeviation] =
ComputeHistoricalStats(data);

benchmarkReturn = annualReturn(1);
benchmarkStandardDeviation = annualStandardDeviation(1);
expectedReturn = annualReturn(2:end);
expectedStandardDeviation = annualStandardDeviation(2:end);
expectedCovariance = annualCovariance(2:end,2:end);

[finalPortfolioStd, finalPortfolioReturn, finalPortfolioWeights,
indicesInFinalPortfolio] =
ComputeBestPortfolio(expectedReturn,expectedCovariance,numOfStocksInPo
rtfolio,targetReturnOfPortfolio/100);

display('.');
display('RECOMMENDED PORTFOLIO STRUCTURE:');

for i=1:numOfStocksInPortfolio
    j = indicesInFinalPortfolio(1,i);
    percent = sprintf('%0.2f',finalPortfolioWeights(j)*100);
    display([assetSymbols{j+1} ' = ' percent ' %']);
end

display('.');
display('RECOMMENDED PORTFOLIO CHARACTERISTICS:');
finalPortfolioReturn_pr = sprintf('%0.2f',finalPortfolioReturn*100);
display(['Return: ' finalPortfolioReturn_pr ' %']);
finalPortfolioStd_pr = sprintf('%0.2f',finalPortfolioStd*100);
display(['Standard Deviation: ' finalPortfolioStd_pr ' %']);

display('.');
display(['BENCHMARK (' benchmarkTicker ') CHARACTERISTICS:']);
benchmarkReturn_pr = sprintf('%0.2f',benchmarkReturn*100);
display(['Return: ' benchmarkReturn_pr ' %']);
benchmarkStandardDeviation_pr =
sprintf('%0.2f',benchmarkStandardDeviation*100);
display(['Standard Deviation: ' benchmarkStandardDeviation_pr ' %']);

portopt(expectedReturn,expectedCovariance); hold on
plot(finalPortfolioStd , finalPortfolioReturn , '.r');
```

The following function *ComputeHistoricalStats* computes historical statistics from the price data of all tickers. Namely, it calculates total return, standard deviation for each ticker, and covariance matrix for all tickers. Note that because the historical data are for the whole year, all these statistics are annual.

```
function [annualReturn,annualCovariance,annualStandardDeviation] =
ComputeHistoricalStats(prices)

returns = tick2ret(prices,[],'continuous');
numReturns = size(returns,1);

[annualReturn,annualCovariance] =
geom2arith(mean(returns),cov(returns),numReturns);

annualReturn = annualReturn';
annualStandardDeviation = sqrt(diag(annualCovariance));
```

The following function *ComputeBestPortfolio* is a wrapper function for the whole optimisation task. It calls *CreateInitialPopulation* function to create initial population, sets population size to $n$-$k$+1, sets custom crossover and mutation functions, defines fitness function *PortfolioFitness*, and finally runs the GA. The GA finally outputs a vector of assets that should be included in the final portfolio, together with related weights. The fitness function itself calls *MinimizeRisk* routine to get the minimum risk value for the actual generation.

```
function [finalPortfolioStd, finalPortfolioReturn,
finalPortfolioWeights, indicesInFinalPortfolio] =
ComputeBestPortfolio(expectedReturn,expectedCovariance,numOfStocksInPo
rtfolio,targetReturn)

poolSize = length(expectedReturn);
initialPopulation =
CreateInitialPopulation(poolSize,numOfStocksInPortfolio);
options = gaoptimset;
options = gaoptimset(options,'PopulationSize' ,poolSize-
numOfStocksInPortfolio+1 );
options = gaoptimset(options,'InitialPopulation' , initialPopulation);
options = gaoptimset(options,'CrossoverFcn' ,@Crossover);
options = gaoptimset(options,'MutationFcn' ,@Mutation);
options = gaoptimset(options,'Display' ,'off');
options = gaoptimset(options,'PlotFcns' ,{  @gaplotbestf
@gaplotbestindiv });

[portBits,fval,gaExit] =
ga(@PortfolioFitness,poolSize,[],[],[],[],[],[],[],options);

portBits = logical(portBits);
subsetCovariance = expectedCovariance(portBits,portBits);
subsetReturn = expectedReturn(portBits);
```

```
 [subsetWeights,subVar,mvExit] =
MinimizeRisk(subsetCovariance,subsetReturn,targetReturn);
finalPortfolioReturn = subsetWeights'*subsetReturn;
finalPortfolioStd = sqrt(subVar);

indices = 1:poolSize;
indicesInFinalPortfolio = indices(portBits);
finalPortfolioWeights = zeros(poolSize,1);
finalPortfolioWeights(indicesInFinalPortfolio) = subsetWeights;

    function risk = PortfolioFitness(x)
        x = logical(x);
        subsetCovariance = expectedCovariance(x,x);
        subsetReturn = expectedReturn(x);

        [weights,fval,mvExit] =
MinimizeRisk(subsetCovariance,subsetReturn,targetReturn);

        if( mvExit == 1 )
            risk = sqrt(fval);
        else
            risk = 1;
        end
    end
end
```

The following function *CreateInitialPopulation* creates the initial population of *n*-*k*+1 chromosomes. Each chromosome contains *n* bits (genes). *k* bits in each chromosome are equal to one, the other bits are equal to zero. No two chromosomes have the same bit configuration. Each chromosome represents one possible configuration of the portfolio. In the chromosome, every asset from the initial pool of *n* securities is represented by one bit. If the value of the bit is equal to one, the particular asset is included in the portfolio, if the bit is equal to zero, the asset is not present in the portfolio.

```
function initialPopulation = CreateInitialPopulation(n,k)

initialPopulation = zeros(n-k+1,n);

for i=1:n-k+1
    initialPopulation(i,i:i+k-1) = 1;
end
```

Te following crossover function *xoverKids* created by MathWorks (32) is written to work on a population of vectors of zeros and ones with the same amount of ones in each vector. The children that are produced from two parents will have the same genes for every element they agree on, and random choices of zeros and ones for the elements they do not agree on. The number of each is set so that all children have the same number of ones as their parents. All children automatically satisfy this constraint so

there is no need to impose these constraints. For examples, please see the full commented source code in Appendices.

```
function xoverKids  =
crossoverNcK(parents,options,GenomeLength,FitnessFcn,unused,thisPopula
tion)

nKids = length(parents)/2;

xoverKids = zeros(nKids,GenomeLength);

index = 1;

num1s = sum(thisPopulation(1,:));
indexVec = 1:GenomeLength;

for i=1:nKids

    r1 = parents(index);
    index = index + 1;
    r2 = parents(index);
    index = index + 1;

    p1 = thisPopulation(r1,:);
    p2 = thisPopulation(r2,:);
    matching1s = ~xor(p1,p2) & (p1 == 1);
    matching0s = ~xor(p1,p2) & (p1 == 0);
    matching1sIndx = indexVec(matching1s);
    matching0sIndx = indexVec(matching0s);
    nonmatchingIndx =
setdiff(indexVec,[matching0sIndx,matching1sIndx]);
    numMatching1s = numel(matching1sIndx);
    num1sToFill = num1s - numMatching1s;
    Indx1sToFill = randsample(nonmatchingIndx,num1sToFill);
    xoverKids(i,matching1sIndx) = 1;
    xoverKids(i,Indx1sToFill) = 1;
end
```

The following mutation function *mutationChildren* created by MathWorks (32) is written to work on a population of vectors of zeros and ones with the same amount of ones in each vector. The mutated child is formed by randomly permuting the elements of the parent.

```
function mutationChildren =
mutationNcK(parents,options,GenomeLength,FitnessFcn,state,thisScore,th
isPopulation)

mutationChildren = zeros(length(parents),GenomeLength);
numVars = length(thisPopulation(1,:));
for i=1:length(parents)
    child = thisPopulation(parents(i),:);
    mutationChildren(i,:) = child( randperm(numVars) );
end
```

The following function calculates the weights of each asset in particular portfolio. The goal is to calculate them in a way that would minimize the risk. But, at the same time, the target return has to be met. The function uses quadratic programming solver *quadprog* for this task, since it is recommended for cases where we already know which items will be included in the portfolio (32).

```
function [weights,fval,qpExit] =
MinimizeRisk(covarianceMatrix,indvRet,targetReturn)

portfolioSize = length(indvRet);
f = zeros(portfolioSize,1);
Aeq = [ ones(1,portfolioSize); indvRet' ];
beq = [ 1; targetReturn ];
lbnds = zeros(portfolioSize,1);
ubnds = ones(portfolioSize,1);
qoptions = optimset('Display','off','LargeScale','off');
[weights,fval,qpExit] =
quadprog(covarianceMatrix,f,[],[],Aeq,beq,lbnds,ubnds,[],qoptions);
fval = 2*fval;
```

### 4.2.3 Results

The presented portfolio optimisation model allows solving difficult task of portfolio selection. In minutes, it is able to pick the best solution from a tremendous number of possibilities. E.g. if we want to select 10 equities out of 100, there are $1.7*10^{13}$ possible combinations how this selection might be done. If we wanted to solve this task using 'brute force', we would need to calculate all necessary characteristics for each of these combinations, compare all of them, and finally pick the best one. Even when using the most powerful computers, this would constitute a serious problem. For a small investor such as BELTREX, it is now possible to solve this task quickly and without excessive investments into highly specialised computing machines.

When we assess the example on which the model was presented, it is clear that the performance of the recommended portfolio is significantly better when compared to the S&P 500 benchmark. For the given required return, which is by one third higher than the one of the benchmark, the standard deviation of the index is twice as high as of the recommended optimal portfolio. This result is in agreement with the MPT, which says that diversification is a useful tool to minimise risk.

The quality of the model can be further demonstrated on the fact that the recommended portfolio is located on the efficient frontier, i.e. it has the best possible return-to-risk ratio for given level of required return.

### 4.2.4 General Notes and Limitations

An advantage of this solution is the fact that it is customisable. The user can find a portfolio that would fully suit their needs – it is possible to restrict the size of the portfolio, to define the pool from which the portfolio will be generated, the model can work with custom periods of data, etc.

However, there are also some constraints. Firstly, the maximal size of the initial pool is 100, which is enough for a smaller investor, but might be a limitation in some cases, e.g. if it is needed to asses hundreds of stocks, bonds, funds, derivatives and other assets simultaneously. Secondly, the user input is not validated and the program itself is not treated against most errors caused by inappropriate inputs. Therefore, before use it is advisable to consult the author of the model, or the source code. On the other hand, such validation can be easily implemented into the application. Finally, we have to emphasize that the solution does not include any transaction costs, taxes, and other associated charges.

It is important to note that the presented solution is by no means a complete trading strategy. The model is rather a tool, which can be used to develop and perform such strategy. For instance, its output, i.e. the recommended portfolio structure, might be utilised during the decision-making processes performed by a portfolio investor.

The application is constructed to use historical data but, after slight customisation, it might work well also with predictions, e.g. provided by neural networks. The crucial problem in such case would be to obtain high quality prediction. When using historical data, one has to bear in mind, that past returns are not necessarily indicative of future returns. However, the historical data is still one of the best sources of information available and is often used as a projection of future.

# 5 CONCLUSIONS

This thesis was focused on possible application of neural networks and genetic algorithms in the field of stock market prediction and portfolio optimisation. In agreement with the goals set at the beginning of the research, two models utilising these methods were developed in the Matlab programming environment.

The application based on neural network is able to predict following day's price trend for S&P 500 ETF. During the testing period, it generated 25.6 % return in simulated trading environment and the trend direction was indicated correctly in 74 % of cases. Furthermore, the development of the equity curve was stable, without excessive draw-downs. These results generally proved that neural networks have good generalisation capability and are appropriate for similar prediction tasks.

To solve the portfolio optimisation problem a genetic-algorithm-based model was developed. It proved to have good ability to solve such complicated optimisation task quickly and efficiently. In the case study presented in this research, the application was able to select optimal portfolio out of $1.7*10^{13}$ possible combinations in just few seconds. The output of this model can be used as one of decision-making factors during the portfolio selection.

Both models can be used as a basis for custom trading strategy development. The prediction model should help to answer the question whether to speculate on rise or fall of the price. It is constructed in a way that is particularly suitable for short-term speculators. On the other hand, the portfolio optimisation program can facilitate the portfolio selection task. It helps to find minimum risk portfolio with given requested return and therefore exploit full potential of investment diversification. Due to the nature of portfolio investment, it is suitable especially for mid- to long-term investors.

# 6 REFERENCES

***Printed sources:***


1) ARNONE, Salvatore; LORASCHI, Andrea; TETTAMANZI, Andrea. *Genetic Approach to Portfolio Selection*. Neural Network World : International Journal on Neural and Mass-Parallel Computing and Information Systems, 1993. p. 597 - 604.
2) BRADOVÁ, Klára. *Volba optimálního portfolia cenných papírů jakožto investiční hlavolam*. Brno, 2010. 87 p. Diploma Thesis. VUT v Brně.
3) CHAN, Ernest P. *Quantitative Trading : How to Build Your Own Algorithmic Trading Business*. New Jersey : John Wiley & Sons, Inc., 2009. 181 p. ISBN 978-0-470-28488-9.
4) CORDÓN, Oscar, et al. *Genetic Fuzzy Systems : Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore : World Scientific Publishing Co. Pte. Ltd., 2001. 462 p. ISBN 981-02-4017-1.
5) DOSTÁL, Petr. *Advanced Economic Analyses*. 1st edition. Brno : CERM, s.r.o., 2008. 80 p. ISBN 978-80-214-3564-3.
6) DOSTÁL, Petr; SOJKA, Zdeněk. *Financial Risk Management*. 1st edition. Zlín : Univerzita Tomáše Bati ve Zlíně, 2005. 60 p. ISBN 80-7318-343-9.
7) DOSTÁL, Petr. *Pokročilé metody analýz a modelování v podnikatelství a veřejné správě*. 1st edition. Brno : CERM, s.r.o., 2008. 340 p. ISBN 978-80-7204-605-8.
8) ELTON , Edwin J.; GRUBER, Martin J. Modern portfolio theory, 1950 to date. In *Journal of Banking & Finance*. [s.l.] : [s.n.], 1997. p. 1743 - 1759 .
9) FROMLET, Hubert Behavioral Finance : Theory and Practical Application. In *Business Economics*. [s.l.] : [s.n.], 2001. p. 63 - 69.
10) GALUSHKIN, A. *Neural networks theory*. New York : Springer, 2007. 396 p. ISBN 978-3-540-48124-9.
11) GATELY, Edward J. *Neural Networks for Financial Forecasting*. New York : John Wiley & Sons, New York,, 1996. 169 p. ISBN 978-0471112129.
12) HASSAN, Rafiul. *StockMarket Forecasting Using Hidden Markov Model : A New Approach*. Carlton, 2005. 5 p. Research Paper. The University of Melbourne.

13) KOHARA, Kazuhiro, et al Stock Price Prediction Using Prior Knowledge and Neural Networks. In *Intelligent Systems in Accounting, Finance & Management*. [s.l.] : John Wiley & Sons, Ltd., 1997. p. 11 - 22.

14) KOHOUT, P.; HLUŠEK, M. *Peníze, výnosy a rizika. Příručka investiční strategie*. Praha: Ekopress, 2. vyd., 2002. 216 p. ISBN 80-86119-48-3.

15) LAWRENCE, Ramon. *Using Neural Networks to Forecast Stock Market Prices*. Manitoba, 1997. 21 p. Research Paper. University of Manitoba.

16) LIN, Chi-Ming; GEN, Mitsuo An Effective Decision-Based Genetic Algorithm Approach to Multiobjective Portfolio Optimization Problem. In *Applied Mathematical Sciences*. Vol. 1., 2007, no. 5., p. 201 - 210.

17) MAŘÍK, Vladimír, et al. *Umělá inteligence : (1)*. Praha : Academia, 2004. 264 p. ISBN 80-200-0496-3.

18) MUNAKATA, T. *Fundamentals of the new artificial intelligence : neural, evolutionary, fuzzy and more*. 2nd edition. London : Springer, 2008. 255 p. ISBN 978-1-84628-838-8.

19) MURPHY, John J. *Technical Analysis of the Financial Markets : A Comprehensive Guide to Trading Methods and Applications*. New York : New York Institute of Finance, 1999. 542 p. ISBN 0-7352-0066-1.

20) REJNUŠ, Oldřich. *Finanční trhy*. 1st edition. Ostrava : KEY Publishing s.r.o., 2008. 559 p. ISBN 978-80-87071-87-8.

21) RUSSEL, Stuart; NORVIG, Peter. *Artificial Intelligence : Modern Approach*. London : Prentice Hall, Inc., 1995. 845 p. ISBN 0-13-360124-2.

22) RYAN, Bob. *Finance and Accounting for Business*. 2nd Revised edition. London : Thomson Learning, 2008. 544 p. ISBN 978-1844808977.

23) VAN EYDEN, Robert J. *The Application of Neural Networks in the Forecasting of Share Prices*. [s.l.] : Finance and Technology Publishing, 1996.

24) VEDARAJAN, Ganesh; CHAN, Louis Chi; GOLDBERG, David E. *Late Breaking Papers at the Genetic Programming 1997 Conference*. Stanford University, California : Stanford Bookstore, 1997. Investment Portfolio Optimization using Genetic Algorithms, p. 255 – 263.

25) WHITE, Halbert Economic prediction using neural networks : The case of IBM daily stock returns. In *Neural Networks in Finance and Investing*. [s.l.] : Probus Publishing Company, 1993. p. 315 - 328.

26) YOON, Youngohc. *Predicting Stock Price Performance : A Neural Network Approach*. Springfield, 1995. 7 p. Research Paper. Southwest Missouri State University.

### *Electronic sources:*

27) GOETZMANN, William N. *An Introduction to Investment Theory* [online]. New Haven : YALE School of Management, 1997 [cit. 2010-08-22]. Available at WWW: <http://viking.som.yale.edu/will/finman540/classnotes/notes.html>.

28) *Hedge Fund Consistency Index* [online]. 2008 [cit. 2010-08-22]. Standard Deviation – Definition and Other Information. Available from WWW: <http://www.hedgefund-index.com/d_standarddev.asp>.

29) HING, Mark. *Automatic Investor* [online]. 2008 [cit. 2010-08-22]. Using Modern Portfolio Theory with Automatic Investor . Available at WWW: <http://www.automaticinvestor.com/articles/mpt.html>.

30) *MedLibrary.org* [online]. 2009 [cit. 2010-08-22]. Modern portfolio theory. Available at WWW: <http://medlibrary.org/medwiki/Modern_portfolio_theory>.

31) *Neuro AI* [online]. 2010 [cit. 2010-08-22]. Stock market prediction. Available at WWW: <http://www.learnartificialneuralnetworks.com/stockmarketprediction.html>.

32) ROSEN, Oren. *MathWorks* [online]. 2005 [cit. 2010-08-22]. Recorded Webinar: Using MATLAB to Develop Portfolio Optimization Models . Available at WWW: <http://www.mathworks.com/company/events/webinars/wbnr30412.html?id=30412&p1=31085&p2=31086>.

33) The MathWorks, Inc.. Documentation for MathWorks Products [online]. [2009] [cit. 2009-07-05]. Available at WWW: <http://www.mathworks.com/access/helpdesk/help/helpdesk.html>.

34) TSENG, Sean. *Stock Price Time Series Forecasting by Neural Networks* [online]. Evanston, 2009. Research Paper. Northwestern University. Available at WWW: <http://sites.google.com/site/nueecs349f09stocks/>.

35) WANG, Jiang. *Portfolio Theory : Chapter 10* [online]. Cambridge, 2003. 48 p. Lecture Notes. Massachusetts Institute of Technology . Available at WWW: <http://web.mit.edu/15.407/file/Ch10.pdf>.

36) *Yahoo! Finance* [online]. 2010 [cit. 2010-08-22]. Available at WWW: <http://finance.yahoo.com>.

# 7 LIST OF ABBREVIATIONS AND SYMBOLS

AI – Artificial Intelligence

ANN – Artificial Neural Network

EMH – Efficient Market Hypothesis

ETF – Exchange Traded Fund

GA – Genetic Algorithm

MPT – Modern Portfolio Theory

NN – Neural Network

NYSE – New York Stock Exchange

OHLC – open, high, low and close prices

P&L, P/L – Profit and Loss

RWT – Random Walk Theory

# 8 LIST OF FIGURES

# 9 LIST OF APPENDICES

# 10 APPENDICES

## *Appendix A – Source Code of the NN Prediction Model*

### predictNextDayClose.m File:

```
clear all;

display('.');
display('-------------------------------------------------------------');
display('.');

%% Define basic settings

predictForDate = datenum(input('For which day do you want to predict
the price (MM-DD-YYYY, in commas): '));

% Numbers of neurons in hidden layers
numHiddenNeurons = [100 100 100 100 100 100 100 100 100 100];

% On how many days are we going to train the network?
learningWindow = 60;

%% Get the Data from Yahoo

assetSymbol = 'SPY';

% Calculate the period for data to be downloaded
toDate = predictForDate - 1; % We will download data for days prior to
the user-defined one.
fromDate = datenum(toDate) - learningWindow - 50;

% Download the data
[data] = yahooDataFetcher(assetSymbol, fromDate, toDate);

% Transform the data
rowData = data';
Close = rowData(5,1:end);
ts_data = fints(data(:,1), data(:,2:5), {'OPEN' 'HIGH' 'LOW'
'CLOSE'});

%% Calculate TA indicators

SMA5 = tsmovavg(Close, 's', 5); % Simple Moving Average 5 days
SMA15 = tsmovavg(Close, 's', 15); % Simple Moving Average 15 days
EMA5 = tsmovavg(Close, 'e', 5); % Exponential Moving Average 5 days
EMA15 = tsmovavg(Close, 'e', 15); % Exponential Moving Average 15 days
[MACD NINEPERMA] = macd(Close); % MACD + Exponential Moving Average 9
days, which is usually used as a signal line
RSI14 = rsindex(data(1:end,5),14)'; % RSI 14 days - column format +
covert to row format
```

```
STOSCkd = stochosc(data(1:end,3), data(1:end,4), data(1:end,5), 10, 3,
'e'); % Stochastic oscillator - column format
STOSCk = STOSCkd(1:end,1)'; % Stochastic oscillator %K (10 days) +
convert to row format
STOSCd = STOSCkd(1:end,2)'; % Stochastic oscillator %D (3 days => slow
=> generates little false signals) + covert to row format

%% Prepare the input matrix

netInputsComplete = rowData([1:6],:);
netInputsComplete(end+1,:) = SMA10;
netInputsComplete(end+1,:) = EMA10;
netInputsComplete(end+1,:) = MACD;
netInputsComplete(end+1,:) = NINEPERMA;
netInputsComplete(end+1,:) = RSI14;
netInputsComplete(end+1,:) = STOSCk;
netInputsComplete(end+1,:) = STOSCd;
netInputsComplete(end+1,:) = CloseBolling(1,:);
netInputsComplete(end+1,:) = CloseBolling(2,:);
netInputsComplete(end+1,:) = CloseBolling(3,:);

% We keep the data only for the needed time window
netInputsComplete = netInputsComplete(:,end-learningWindow:end); %
includes dates
netInputs = netInputsComplete(2:end, :); % strip dates

%% Prepare input matrices for training and simulation

netInputsCycle = netInputs(:, 1:end-1);
simInputsCycle = netInputs(:, 2:end);
netInputsCycleSeq = con2seq(netInputsCycle);


%% Prepare the target vector for training

% The target vector is formed by tomorrow's closing prices
netTargetsCycle = netInputs(4, 2:end);
netTargetsCycleSeq = con2seq(netTargetsCycle);

%% Set the network

% Feed-forward backpropagation network with Fletcher-Powell conjugate
gradient backpropagation
net = newff(netInputsCycleSeq, netTargetsCycleSeq, numHiddenNeurons,
{'logsig' 'logsig' 'logsig' 'logsig' 'logsig' 'logsig' 'logsig'
'logsig' 'logsig' 'logsig' 'purelin'},'traincgf');

% Options
trset = 60; % trainRatio
net.divideParam.trainRatio = trset;
net.divideParam.valRatio = (100 - trset) / 2;
net.divideParam.testRatio = (100 - trset) / 2;
net.trainParam.epochs = 600;
net.trainParam.goal = mean(netTargetsCycle) * 0.001;

%% Train and simulate Network on cell data

% Train Network
[net,tr] = train(net, netInputsCycleSeq, netTargetsCycleSeq);
```

```
% Simulate Network
simInputsCycleSeq = con2seq(simInputsCycle);
Y = sim(net, simInputsCycleSeq);
z = seq2con(Y);

% Output prices of the simulation; the last item is the predicted
price for
% the user-defined day
netOutputs = z{1,1};

%% Plot real historical data and data provided by the simulation

simDatesForHistoricalPrices = netInputsComplete(1,2:end);
simDatesForPredictedPrices = simDatesForHistoricalPrices(:,2:end);
predictedDayDate = simDatesForHistoricalPrices(1,end) + 1;
if weekday(predictedDayDate) == 7 % saturday, we cannot predict prices
for weekend, the markets are closed
    predictedDayDate = predictedDayDate + 2;
end
simDatesForPredictedPrices(:,end+1) = predictedDayDate;


tsReal = timeseries(netTargetsCycle,
datestr(simDatesForHistoricalPrices) );
tsReal.Name = 'Real Price';
tsReal.TimeInfo.Units = 'Days';

tsPrediction = timeseries(netOutputs,
datestr(simDatesForPredictedPrices) );
tsPrediction.Name = 'Predicted Price';
tsPrediction.TimeInfo.Units = 'Days';

figure
plot(tsReal,'--bs','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','k',...
                'MarkerSize',5), grid on;
hold on;
plot(tsPrediction,'--rs','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','k',...
                'MarkerSize',5);
title('Time Series: Real and predicted prices');
xlabel('Date');
ylabel('Price (USD)');
legend('Real Price','Predicted Price','Location','northwest');

netOutputs(1,end);

%% Print the predicted price for the following day

% Round the predicted price
predictedPrice = sprintf('%0.4f',netOutputs(1,end));
display('.');
display('-------------------------------------------------------');
display('.');
```

```
display(['Predicted price for '
datestr(simDatesForPredictedPrices(:,end),1) ' is: ' predictedPrice
]);
display('.');
if netTargetsCycle(1,end) > predictedPrice
    display('The price will decrease.');
else
    display('The price will increase.');
end
display('.');
display('------------------------------------------------------------');
```

## yahooDataFetcher.m File:

```
function [data] = yahooDataFetcher(assetSymbol,fromDate,toDate)

Connect = yahoo;

yahooData = fetch(Connect, assetSymbol, {'Open', 'High', 'Low',
'Close', 'Volume'}, fromDate, toDate);

data = flipud(yahooData);
```

## *Appendix B – Source Code of the GA Portfolio Optimisation Model*

### START.m File:

```
clear all;

%% User Defined variables

display('--------------------');
display('.');
fromDate = datenum(input('Enter the start of the computation period
(MM-DD-YYYY, in commas): '));
toDate = datenum(input('Enter the end of the computation period (MM-
DD-YYYY, in commas): '));
numOfStocksInPortfolio = input('Enter the number of securities that
should be included in the final portfolio: ');
targetReturnOfPortfolio = input('Enter the required return of the
portfolio (in %): ');

%% Get and prepare the data

% The variable containing indices of initial pool of assets (max. 100
indices).
% The indices are organised in one column.
% By default 100 manually selected stocks were extracted from the SnP
500 index.
load stocksTickers;

% Ticker of the benchmark against which the portfolio will be
compared.
% Usually an index, e.g. ^GSPC (S&P 500).
benchmarkTicker = '^GSPC';

assetSymbols = [benchmarkTicker stocksTickers']; % prepend all the
loaded stock tickers with the index ticker

% Load the data from Yahoo
[data dates assetSymbols] = YahooDataFetcher(assetSymbols, fromDate,
toDate);

% If there is more stocks than days of data, the stocks at the end
need to
% be cut.
if ((size(data,1) - size(data,2)) < 2)
    data = data(:,1:size(data,1)-2);
end


%% Compute historical statistics for all tickers incl. the benchmark

% All the stats together. We get:
%   * annualReturn = total return for each ticker
%   * annualCovariance = covariances matrix for the tickers
%   * annualStandardDeviation = standard deviation for each ticker
```

```
[annualReturn,annualCovariance,annualStandardDeviation] =
ComputeHistoricalStats(data);

benchmarkReturn = annualReturn(1); % Total Return of the  benchmark
benchmarkStandardDeviation = annualStandardDeviation(1); % Standard
Deviation of the benchmark
expectedReturn = annualReturn(2:end); % Total Returns of all the
assets excl. the benchmark
expectedStandardDeviation = annualStandardDeviation(2:end); % Standard
Deviations of all the assets excl. the benchmark
expectedCovariance = annualCovariance(2:end,2:end); % Covaricance
matrix for all the assets excl. the benchmark

%% Find the optimal portfolio

[finalPortfolioStd, finalPortfolioReturn, finalPortfolioWeights,
indicesInFinalPortfolio] =
ComputeBestPortfolio(expectedReturn,expectedCovariance,numOfStocksInPo
rtfolio,targetReturnOfPortfolio/100);

%% Print Results

display('.');
display('RECOMMENDED PORTFOLIO STRUCTURE:');

for i=1:numOfStocksInPortfolio
    j = indicesInFinalPortfolio(1,i);
    percent = sprintf('%0.2f',finalPortfolioWeights(j)*100);
    display([assetSymbols{j+1} ' = ' percent ' %']);
end

display('.');
display('RECOMMENDED PORTFOLIO CHARACTERISTICS:');
finalPortfolioReturn_pr = sprintf('%0.2f',finalPortfolioReturn*100);
display(['Return: ' finalPortfolioReturn_pr ' %']);
finalPortfolioStd_pr = sprintf('%0.2f',finalPortfolioStd*100);
display(['Standard Deviation: ' finalPortfolioStd_pr ' %']);

display('.');
display(['BENCHMARK (' benchmarkTicker ') CHARACTERISTICS:']);
benchmarkReturn_pr = sprintf('%0.2f',benchmarkReturn*100);
display(['Return: ' benchmarkReturn_pr ' %']);
benchmarkStandardDeviation_pr =
sprintf('%0.2f',benchmarkStandardDeviation*100);
display(['Standard Deviation: ' benchmarkStandardDeviation_pr ' %']);


%% Plot the Efficient Frontier for the Input Tickers

portopt(expectedReturn,expectedCovariance);

% Show the computed portfolio on the Efficient Frontier chart
hold on
plot(finalPortfolioStd , finalPortfolioReturn , '.r');
```

## ComputeHistoricalStats.m File:

```
function [annualReturn,annualCovariance,annualStandardDeviation] =
ComputeHistoricalStats(prices)

% This function computes the following historical statistics
% from the price data of all tickers:
%   * annualReturn = total return for each ticker
%   * annualCovariance = covariances matrix for the tickers
%   * annualStandardDeviation = standard deviation for each ticker

returns = tick2ret(prices,[],'continuous');
numReturns = size(returns,1);

[annualReturn,annualCovariance] =
geom2arith(mean(returns),cov(returns),numReturns);

annualReturn = annualReturn';
annualStandardDeviation = sqrt(diag(annualCovariance));
```

## ComputeBestPortfolio.m File:

```
function [finalPortfolioStd, finalPortfolioReturn,
finalPortfolioWeights, indicesInFinalPortfolio] =
ComputeBestPortfolio(expectedReturn,expectedCovariance,numOfStocksInPo
rtfolio,targetReturn)

% Generate initial population for GA
poolSize = length(expectedReturn);
initialPopulation =
CreateInitialPopulation(poolSize,numOfStocksInPortfolio);

% Start with default options
options = gaoptimset;

% Set population options
options = gaoptimset(options,'PopulationSize' ,poolSize-
numOfStocksInPortfolio+1 );
options = gaoptimset(options,'InitialPopulation' , initialPopulation);

% Set evolution options
%options = gaoptimset(options,'CrossoverFraction' ,0.9 );
options = gaoptimset(options,'CrossoverFcn' ,@Crossover);
options = gaoptimset(options,'MutationFcn' ,@Mutation);

% Set diaplay options
options = gaoptimset(options,'Display' ,'off');
options = gaoptimset(options,'PlotFcns' ,{  @gaplotbestf
@gaplotbestindiv });

% Run GA
[portBits,fval,gaExit] =
ga(@PortfolioFitness,poolSize,[],[],[],[],[],[],[],options);

% Output of GA is logical vector of the best equities to include in
the portfolio
portBits = logical(portBits);
```

```
subsetCovariance = expectedCovariance(portBits,portBits);
subsetReturn = expectedReturn(portBits);
[subsetWeights,subVar,mvExit] =
MinimizeRisk(subsetCovariance,subsetReturn,targetReturn);
finalPortfolioReturn = subsetWeights'*subsetReturn;
finalPortfolioStd = sqrt(subVar);

% Extract tickers for best portfolio
indices = 1:poolSize;
indicesInFinalPortfolio = indices(portBits);
finalPortfolioWeights = zeros(poolSize,1);
finalPortfolioWeights(indicesInFinalPortfolio) = subsetWeights;

%% Fitness function for GA
function risk = PortfolioFitness(x)
    x = logical(x);
    subsetCovariance = expectedCovariance(x,x);
    subsetReturn = expectedReturn(x);

    [weights,fval,mvExit] =
MinimizeRisk(subsetCovariance,subsetReturn,targetReturn);

    % If the optimization routine in MinimizeRisk failed,
    % set risk to high level.
    if( mvExit == 1 )
        risk = sqrt(fval);
    else
        risk = 1;
    end
end

end
```

## CreateInitialPopulation.m File:

```
function initialPopulation = CreateInitialPopulation(n,k)

% This function creates the initial population of n-k+1 chromosomes.
% Each chromosome contains n bits (genes).
% k bits are equal to one, the rest is equal to zero.

initialPopulation = zeros(n-k+1,n);

for i=1:n-k+1
    initialPopulation(i,i:i+k-1) = 1;
end
```

## Crossover.m File:

```
function xoverKids  = Crossover
 (parents,options,GenomeLength,FitnessFcn,unused,thisPopulation)
% Oren Rosen
%
% This custom crossover function is written to work on a population of
```

```
% vectors of zeros and ones with the same amount of ones in each
vector.
% The children that are produced from 2 parents will have the same
genes
% for every element they agree on, and random choices of zerps and
ones for
% the elements they don't agree on. The number of each is set so that
all
% children have the same number of ones as their parents. All children
% automatically satisfy this constraint so there is no need to impose
these
% constraints.

% How many children to produce?
nKids = length(parents)/2;

% Allocate space for the kids
xoverKids = zeros(nKids,GenomeLength);

% To move through the parents twice as fast as thekids are
% being produced, a separate index for the parents is needed
index = 1;

% *** Initialize ***
% Assumes all members of thisPopulation have the same number of ones.
num1s = sum(thisPopulation(1,:));
indexVec = 1:GenomeLength;

% for each kid...
for i=1:nKids

    % *** Get parents ***
    r1 = parents(index);
    index = index + 1;
    r2 = parents(index);
    index = index + 1;

    p1 = thisPopulation(r1,:);
    p2 = thisPopulation(r2,:);

    % *** Find Matching 1's and 0's ***
    % Ex: If           p1 == [ 1 0 1 0 0 1 1 0 0 0 ]
    %                  p2 == [ 1 0 0 1 0 0 1 0 1 0 ]
    %     Then matching1s == [ 1 0 0 0 0 0 1 0 0 0 ]
    %     Then matching0s == [ 0 1 0 0 1 0 0 1 0 1 ]
    matching1s = ~xor(p1,p2) & (p1 == 1);
    matching0s = ~xor(p1,p2) & (p1 == 0);

    % *** Find Matching Indices ***
    %     If       matching1s == [ 1 0 0 0 0 0 1 0 0 0 ]
    %              matching0s == [ 0 1 0 0 1 0 0 1 0 1 ]
    %     Then matching1sIndx == [ 1 7 ]
    %          matching0sIndx == [ 2 5 8 10 ]
    %         nonmatchingIndx == [ 3 4 6 9 ]
    matching1sIndx = indexVec(matching1s);
    matching0sIndx = indexVec(matching0s);
    nonmatchingIndx =
setdiff(indexVec,[matching0sIndx,matching1sIndx]);
```

```
    % *** Create Child ***
    % Ex: If   num1s == 4
    %          matching1sIndx == [ 1 7 ]
    %          nonmatchingIndx == [ 3 4 6 9 ]
    %     Then numMatching1s == 2
    %     num1sToFill == 2
    %     Indx1sToFill == 2 random choices from [ 3 4 6 9 ]
    numMatching1s = numel(matching1sIndx);
    num1sToFill = num1s - numMatching1s;
    Indx1sToFill = randsample(nonmatchingIndx,num1sToFill);

    % *** Fill in 1s ***
    % Ex: If               p1 == [ 1 0 1 0 0 1 1 0 0 0 ]
    %                      p2 == [ 1 0 0 1 0 0 1 0 1 0 ]
    %     Then xoverKids(i,:) == [ 1 0 ? ? 0 ? 1 0 ? 0 ]
    %     With exactly 2 of the '?' equal to 1, the rest 0.
    xoverKids(i,matching1sIndx) = 1;
    xoverKids(i,Indx1sToFill) = 1;
end
```

## Mutation.m File:

```
function mutationChildren =
Mutation(parents,options,GenomeLength,FitnessFcn,state,thisScore,thisP
opulation)
% Oren Rosen
%
% This custom mutation function is written to work on a population of
% vectors of zeros and ones with the same amount of ones in each
vector.
% The mutated child is formed by randomly permuting the elements of
the
% parent.

mutationChildren = zeros(length(parents),GenomeLength);
numVars = length(thisPopulation(1,:));

for i=1:length(parents)
    child = thisPopulation(parents(i),:);
    mutationChildren(i,:) = child( randperm(numVars) );
end
```

## MinimizeRisk.m File:

```
function [weights,fval,qpExit] =
MinimizeRisk(covarianceMatrix,indvRet,targetReturn)

    % This function calculates the weights of each assset in
particular
    % portfolio. The goal is to calculate them in a way that would
    % minimize the risk. But, at the same time, the target return has
to be
    % met.
    % The function uses quadratic programming solver for this.
```

```
    portfolioSize = length(indvRet);
    f = zeros(portfolioSize,1);
    Aeq = [ ones(1,portfolioSize); indvRet' ];
    beq = [ 1; targetReturn ];
    lbnds = zeros(portfolioSize,1);
    ubnds = ones(portfolioSize,1);
    qoptions = optimset('Display','off','LargeScale','off');
    [weights,fval,qpExit] =
quadprog(covarianceMatrix,f,[],[],Aeq,beq,lbnds,ubnds,[],qoptions);
    % Double result (quadprog minimizes (1/2)w'Hw)
    fval = 2*fval;
end
```

## YahooDataFetcher.m File:

```
function [data dates assetSymbols] =
YahooDataFetcher(assetSymbols,fromDate,toDate)

% This function retrieves closing price data for all tickers in
assetSymbols between
% fromDate and toDate.

Connect = yahoo;

invalidSymbols = [];

numberOfTickers = size(assetSymbols,2);

for i=1:numberOfTickers

    assetSymbol = assetSymbols{1,i};
    yahooData = fetch(Connect, assetSymbol, 'Close', fromDate,
toDate);

    if i == 1
       numberOfDays = size(yahooData,1);
       data = zeros(numberOfDays, numberOfTickers);
       dates = yahooData(1:end,1);
    end

    %% Identify indices with history shorter than the requested time
window
    numberOfDaysThisIteration = size(yahooData,1);
    if numberOfDaysThisIteration == numberOfDays
        data(:,i) = yahooData(:,2);
    else
        invalidSymbols = [invalidSymbols i];
    end

end

%% Exclude indices with history shorter than the requested time window
i = size(invalidSymbols,2);
while i > 0
    assetSymbols(invalidSymbols(1,i)) = [];
    data(:,invalidSymbols(1,i)) = [];
    i = i-1;
```

```
end

dates = flipud(dates);
data = flipud(data);
```

## stocksTickers.mat File:

This file contains the tickers of 100 US stocks used to demonstrate the GA optimisation model. The data in the original file are organised in single column, here however, 4-column representation is more suitable.

| | | | |
|---|---|---|---|
| XOM | COP | CAT | LMT |
| AAPL | ABT | EMC | DVN |
| MSFT | MCD | LLY | SO |
| WMT | GS | TGT | MA |
| PG | SLB | UNP | ESRX |
| JNJ | OXY | F | WAG |
| IBM | DIS | TWX | EBAY |
| GE | UTX | NKE | GLW |
| JPM | V | EMR | PRU |
| BAC | UPS | MS | MHS |
| GOOG | MMM | UNH | KMB |
| T | QCOM | DD | MON |
| CVX | AMGN | NWSA | EOG |
| WFC | CMCSA | MET | EXC |
| CSCO | AMZN | DTV | TRV |
| PFE | KFT | GILD | DHR |
| KO | AXP | HON | BAX |
| C | HD | BK | DE |
| INTC | BA | NEM | COST |
| MRK | USB | PNC | CCL |
| ORCL | BMY | LOW | DELL |
| HPQ | MO | TXN | SPG |
| PEP | CVS | APA | PX |
| PM | MDT | FCX | GIS |
| VZ | CL | DOW | CELG |

**Fig. 20 - Contents of the stocksTickers variable**