Genetic Programming: The Applications of Syntax Trees

Yongheng Li

Binghamton University

Abstract

This paper argues that the utilization of syntax trees in Genetic Programming (GP), as opposed to the use of fixed vectors in Genetic Algorithms (GAs), will enhance the flexibility and efficiency during the searching and optimization process. The studies of Petr Kroha and Matthias Friedrich support the scientific claim that the superiority of GP, in terms of its data representation, makes it more suitable than GAs to develop trading algorithms. In comparison to classical search algorithms, GP has the advantage of a global perspective and higher accuracy for approximating discrete variables. GP is a search algorithm that mimics the natural selection and evolution processes through three genetic operations: selections, crossovers, and mutations. Selections identify the fittest genotypes and determine which individuals produce offspring, crossovers create new generations using the genotypes of the fittest parents, and mutations generate random changes in offspring to avoid population convergence. By utilizing GP-based trading strategies, financial institutions will be able to significantly reduce their operating cost and increase their revenue, thereby benefiting the vast majority that entrusts their savings to these corporations.

Genetic Programming: The Applications of Syntax Trees

Introduction

Inspired by the power of natural evolution, Genetic Algorithms empower programs to solve search and optimization problems by simulating the evolutionary and natural selection process (Liu, 2014, p. 2). GAs make it possible for programs to analyze complicated data and find hidden patterns among sets of random data in an efficient way. Similar to natural evolution, GAs involve three genetic operations: crossovers, mutations, and selections. Survival of the fittest applies to GAs as well because only the stronger individuals will be selected to reproduce the next generation.

Genetic Programming is a generalization and extension of Genetic Algorithms. Both GAs and GP belong to the category of Evolutionary Algorithms, but they characterize data differently. GAs represent genotypes as fixed vectors or strings whereas GP characterizes genotypes as variable tree structures (Reid, 2013, p. 5). Due to the use of tree structures, GP has the benefits of a high-level symbolic representation for data. In addition, GP is able to automatically vary the size of its tree structure to adapt to different genotypes because of the utilization of tree structures. Its flexibility allows GP to be used to conduct security analysis and trading decision discovery in finance (Hu, et. al., 2015, p. 534). Genetic Programming is superior to Genetic Algorithms in terms of its flexibility and efficiency in searching and optimization; thus, Genetic Programming shows greater potential to generate robust trading strategies. It is important to understand the ways in which GAs evolve its population to approach solutions.

Background

Genetic Algorithms excel at parameter optimization and problem solving by mimicking the natural evolution process. GAs operate on a population containing different types of

individuals represented by genotypes that are fixed strings (Reid, 2013, p. 5). Complying with the Darwinian natural selection principle, the fittest individuals survive and pass their genes to the next generation. The fitness function assigns each individual a fitness score based on how close it is to the desirable results. GAs simulate natural evolution through the three genetic operations: crossovers, mutations, and selections. Crossovers refer to modeling sexual and asexual reproduction so that offspring are more likely to have stronger characteristics because they take desirable aspects from their parents. Mutations bring random changes to offspring to avoid population convergence, and mutations provide genetic diversity to population, which greatly improves the accuracy of results (Manahov, 2015, p. 96). Selection is the process where fitter individuals are chosen for later breeding. The evolutionary process makes it possible for GAs to solve search and optimization problems which can in turn, be implemented in security analysis in finance.

Security analysis refers to the evaluation of financial instruments such as securities based on technical or financial factors. There are two main analytical approaches, technical analysis and fundamental analysis. Inspired by the Newton's First Law, technical analysis forecasts the price of securities by studying the past price movement. In comparison, fundamental analysis evaluates the intrinsic value of securities based on financial factors, such as revenue and income tax (Hu, et. al., 2015, p. 534). Security analysis could be a classification problem as every stock is categorized by trading rules to make investment decisions. By implementing Genetic Programming, a decisions tree can be developed to evaluate the value of securities and make trading decisions (Reid, 2013, p. 6).

In financial markets, institutional investors are the major users of GP-based trading strategies since financial institutions are the only ones that have the ability and resources to

support the development of trading algorithms and because they are equipped with rich trading experience and technical knowledge. Luis Aguilar, a commissioner from U.S Securities and Exchange Commission, suggests that institutional investors are the dominant market players since they manage over 67% of U.S public equities (2013). Millions of people entrust their money to financial institutions in order to ensure a secure retirement plan and help them save for things like a car, a house, college education and so on. Due to the great power of institutional investors, their success is representative of the success of the majority of people in the society. Individual investors are in a less favorable position because they show less predictive ability for future markets and they are more likely to lose money due to a lack of sophistication (Chang, Luo, & Ren, 2013). In addition, individual investors do not have the access to GAs like financial institutions do. Before the adoption of Genetic Algorithms, optimization and search problems were mainly solved by traditional algorithms, which caused many difficulties for developers.

Precedents and Related Work

One problem of traditional methods is specialization because traditional algorithms are designed to solve particular problems (Deb, 2014, p. 1). For instance, geometric programming is designed to solve optimization problems for constraints with specific forms. Another example is that the conjugate gradient method works well for solving quadratic functions, but it cannot find solutions to problems having multiple optimal solutions (Deb, 2014, p. 2). Thus, traditional algorithms are capable of efficiently solving their desired problems, but they are not expected to solve other types of problems.

In addition, as a dominant traditional method, gradient-based algorithms cannot efficiently handle problems involving discrete variables. In the past, traditional algorithms approximated the discrete values based on the assumption that solutions can be represented by

real values. To make the values feasible, the nearest results are selected within a certain tolerance. However, many problems occur during the approximation process. For instance, if too many infeasible values are involved, traditional algorithms make functions evaluate before converging, which results in ineffective searching (Deb, 2014, p. 3). Additionally, the approximation process is time-consuming because programs need to examine two values, the closest lower and upper values, for each infeasible discrete value. Furthermore, approximation by only these two values cannot guarantee the best optimization results. All of these problems can be solved if search and optimization algorithms are able to use discrete variables.

Premises imply that traditional algorithms might not meet the requirements for an efficient search and optimization algorithm, however, Genetic Algorithms are becoming more attractive because they overcome the deficiencies of traditional search and optimization algorithms. Genetic Algorithms have global perspectives since they are applicable to various scenarios as long as the fitness function is given. Moreover, Genetic Algorithms have a better approximation for discrete variables since they use evolution to simulate the calculation process, instead of using gradient values. The mechanism behind Genetic Algorithms makes it a better method for search and optimization problems.

Support

Technical Details

How Genetic Programming works. GP operates on a population containing various individuals and simulates the natural evolution process to solve complicated problems. Figure 1 exhibits the procedure of a general GP algorithm (Khadka, 2013, p. 26). After initializing a population with various genotypes, the fitness function calculates the distance from the expected value for each individual. Then the algorithm performs three genetic operations: selections,

crossovers, mutations. This process continues until a generation satisfies the termination criteria, which consistently conducts fitness evaluation and genetic operations (Kroha & Friedrich, 2014, p. 384). Using an appropriate way to represent data is extremely important for GP since it is a population based search algorithm.

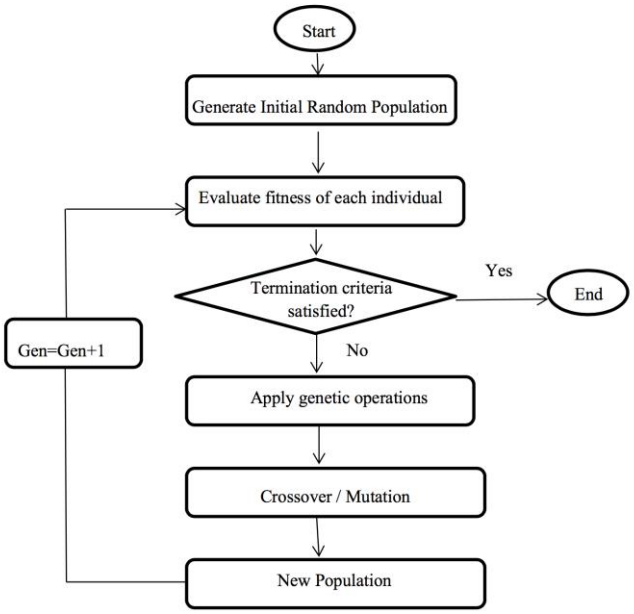

Figure 1. Flowchart of genetic programming.

Syntax Tree. A genotype is a classification of DNA, which contains the essential genetic information for building an organism (Armani, 2014, p. 44). Syntax trees in GP help represent genotypes in a population in order to perform genetic operations. In a syntax tree, nodes are connected with each other by links, which represent the dependencies between operations and arguments (Armani, 2014, p. 44). It turns out that the syntax tree is a perfect data structure to represent mathematical expressions. For instance, a mathematical function, $f(x) = x^3 + \cos(y)$, can be converted to a syntax tree as shown in Figure 2 (Armani, 2014, p. 54). Syntax trees allow GP to perform the genetic operations by changing the shape, size, and content of nodes. Before conducting genetic operations, each individual needs to be assessed by fitness functions.

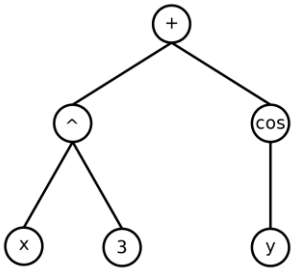

Figure 2. Syntax tree of $f(x) = x^3 + \cos(y)$.

Fitness function. Fitness functions evaluate genotypes. They assign every individual a fitness value based on how close individuals are to the desired results. A smaller fitness value implies an individual is closer to the desired results. For example, for a gun firing problem, the fitness function determines how far the bullet missed the targets (Khadka, 2013, p. 26).

The first fitness function was introduced by Koza in 1992, which can be represented by an equation $F(k,t) = \sum_{j=1}^{m} \left| f(x_j) - \tilde{f}_{k,t}(x_j) \right|$ (Armani, 2014, p. 54). Koza calls this fitness function "raw fitness" and other complicated fitness functions can be developed based on it. In the equation, fitness values are represented by the sums of the absolute errors. $F(k,t)$ represents the fitness value, where *k* is the individual and *t* is the individual's generation; *m* is the number of fitness cases in the initial population; $f(x_j)$ is the actual response of the system at the fitness case x_j; " $\tilde{f}_{k,t}(x_j)$ is the value returned by individual *k* at the generation *t* for the fitness case x_j" (Armani, 2014, p. 54). While fitness functions serve as a driving force in the evolution process, they also suffer from some inadequacies. For instance, a longer genotype has a statistical advantage on a shorter genotype because it has more parameters to evaluate. These parameters might result in the longer genotype having a higher fitness value. The fitness function is the critical for the next stage, selection.

Selection. The selection process discovers the fittest individuals in the current generation. Similar to the idea of "survival of the fittest", only the individuals with higher fitness values will be selected and have the right to pass their genotypes to the next generation (Khadka, 2013, p. 26). In this process, each individual will be given a probability figure, based on its fitness value, which is easily calculated by the normalizing fitness value from the range 0 to 1. After the probability figure is calculated, another number will be randomly generated in the same range from 0 to 1. The individual will be selected if their normalized fitness score is greater than that random number (Armani, 2014, p. 54). Selection has a profound influence on the evolution process because if the selective pressure is too low, individuals with lower fitness values will be selected and thus programs will take more evolution cycles to approach the expected results. On the other hand, if there is too much emphasis on the individuals with higher fitness scores, the evolution search will diminish genetic diversity and be limited to a sub-region of the desired space. After selection, the fittest parents are ready to reproduce.

Crossover. Crossover refers to the process of using the genotypes of parents to produce new generations. The most common crossover operator is the "standard crossover", also known as "subtree cross over", which generates two offspring from the two parents by randomly swapping one subtree of one parent, while keeping the other subtree to compose the new offspring (Khadka, 2013, p. 26). Figure 3 illustrates how the crossover happens between two parents, *Parent 1*: $\frac{\log x^2 + 3}{\cos(y)}$ and *Parent 2:* $2x + \sin(z^3)$ (Armani, 2014, p. 66). Node 3 in *Parent 1* and node 5 in *Parent 2* were randomly swapped to produce two new offspring,

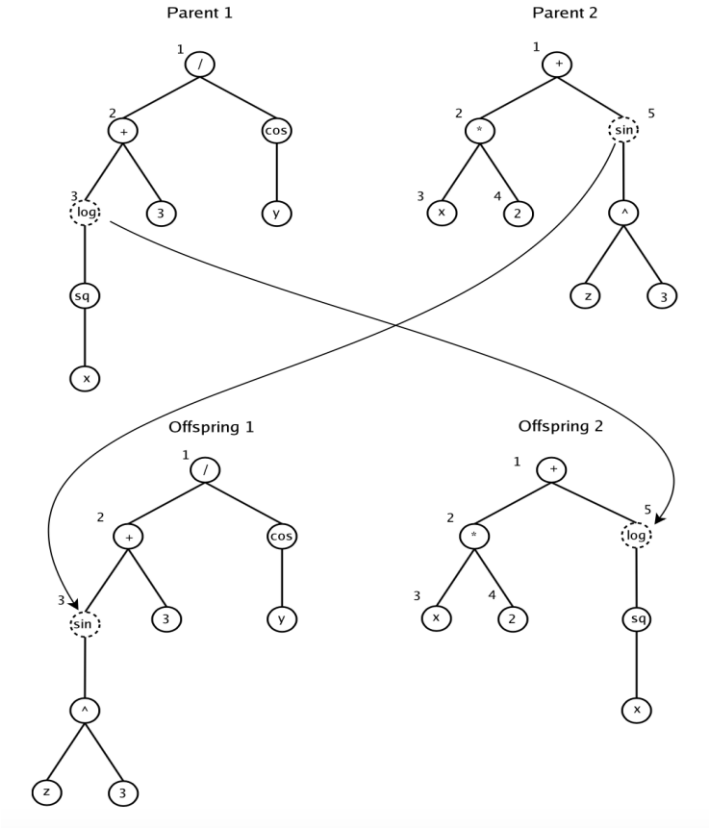

Figure 3. Standard crossover or subtree crossover.

Offspring 1: $\frac{\sin(z^3)+3}{\cos(y)}$ and *Offspring 2:* $2x + \log x^2$. The new offspring are similar to their parents but also include new features. While many GP algorithms utilize the method of standard crossover, it also comes with limitations. For instance, a standard crossover fails to introduce drastic modifications to existing genotypes and thus it weakens the global exploration of the search space. This shortcoming of crossovers makes mutation necessary to the evolution process.

Mutation. Mutation creates random changes in the genotypes of individuals. As the most common mutation operator, point mutation performs the smallest changes to a syntax tree. Since the smallest variation happens at a node level, there are two possible consequences: a function will be randomly changed but the arguments remain the same, or a function remains the same but one of its arguments will be changed. An example of point mutation is shown in Figure 4

(Armani, 2014, p. 69). The mutation point is located in node 2, which is a function node. The original function, addition (+) was randomly replaced by division (/).

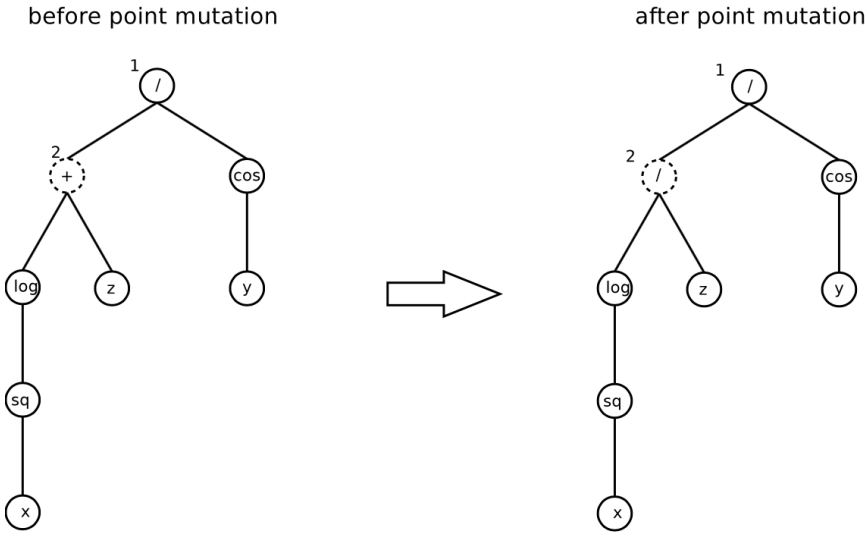

Figure 4. Point mutation.

Thus, the *Parent* ($\frac{\log x^2 + z}{\cos(y)}$) generated the *Offspring* ($\frac{\log x^2}{z}\frac{1}{\cos(y)}$) by mutation. As shown, a small change in a node of the parent's genotype can have a significant impact on its offspring. Generally, mutation has a greater influence on evolutionary processes than crossover (Armani, 2014, p. 69). This is because mutation introduces new genetic material to the gene pool, whereas crossover only swaps the existing genetic material (Khadka, 2013, p. 50). The creator of GP, Koza, believes that a successful Genetic Programming algorithm is expected to explore a global search space and evolve its existing good individuals as well (Armani, 2014, p. 72). While GP allows parents and children to conduct point mutation using tree structures, GAs have more limitations conducting mutation because of the way fixed vectors restrict genetic diversity.

Genetic Algorithms versus Genetic Programming. Both GAs and GP are Evolutionary Algorithms and thus have the same evolution process, including fitness evaluation and genetic

operations. The only difference between GAs and GP is the way they represent data. GAs represent genotypes as fixed vectors whereas GP characterizes genotypes as variable tree structures (Reid, 2013, p. 5). An example of crossover in GAs is shown in

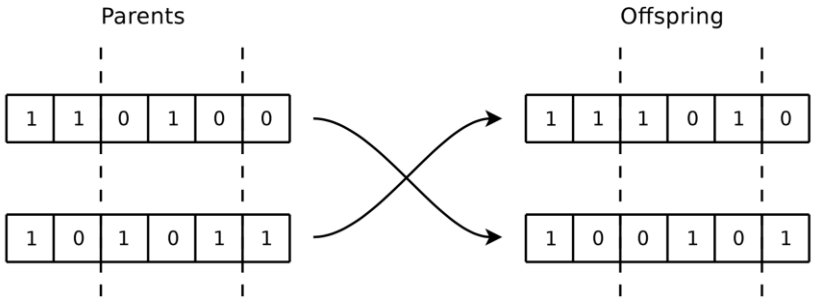

Figure 5. Crossovers in GAs.

Figure 5. The fixed vector representation suffers from many limitations. For instance, the length of a vector is fixed and needs to be predefined by developers. As shown in Figure 5, the genotypes of two parents and two offspring are the same length, which largely constrains the genetic diversity (Armani, 2014, p. 50). In addition, due to the way they represent chromosomes, GAs are able to handle integer variables, but underperform for other types of data. In comparison, due to the use of tree structures, GP has the benefits of a high-level symbolic representation for data, which allows it to increase the size, shape, and complicity of its genotypes (Armani, 2014, p. 50). The superiority of GP makes it applicable to many fields, especially in terms of finance.

Applications

Genetic Programming is widely implemented in finance because of the superiority of its syntax tree. The syntax tree not only generates possible solutions that are similar to the decisions made by real human traders, but also enables GP to have the flexibility and efficiency to

approach complicated tasks, such as building trading strategies (Manahov, 2014, p. 66). Kroha and Friedrich (2014) have conducted an experiment on two trading strategies based on two trading strategies based on two genetic systems employing GAs and GP (p. 384). The first genetic system implements GAs to derive trading strategies while the other utilizes GP to do so.

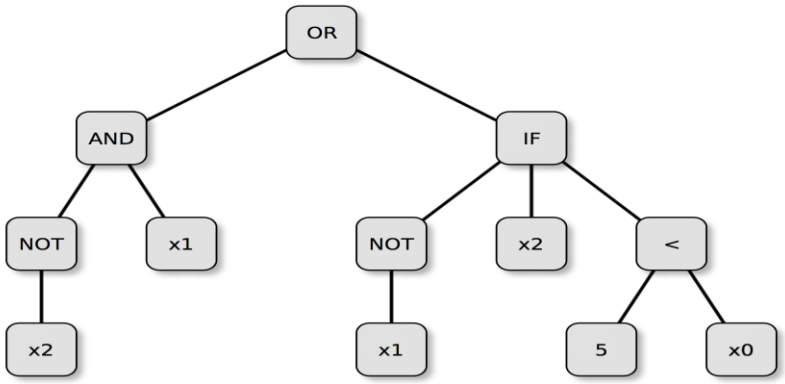

Figure 6. An instance of a tree in genetic programming.

In the GP system, each possible trading strategy is represented by a tree structure as shown in Figure 6 (Kroha & Friedrich, 2014, p. 387). For instance, functions (e.g. or, and, not, if-then-else) are converted to tree nodes; parameters (e.g. close price and technical indicators) are represented by tree leaves. After conducting experiments, Kroha and Friedrich (2014) suggest that both genetic systems outperform the simple buy-and-hold strategy. Between the two systems, the GP-based system is superior in terms of generating robust and profitable trading strategies (p. 393). GP can also be used to predict financial market activities in a wide range of scenarios. A professor at American University of Beirut, Karatahansopoulos (2014) and his colleagues also proved the benefits of using GP to predict Greek stock markets (p. 608). Their studies show that GP-based trading algorithms are able to generate accurate forecasts of stock prices one day ahead in the Greek stock market. In addition, Yu-Tzu Chiu (2008), a Taiwanese computer scientist, has developed a genetic algorithm that is able to predict the distress of public

companies two years in advance (p. 1). The advantages that GP brings to financial institutions are not necessarily advantages for the individual investors.

Ethical analysis. While institutional investors benefit from GP-based trading strategies, it might cause harmful effects on individual investors. According to Rawls, an action is not ethical if it leaves the least-advantaged people in a worse position than they were before. Compared to institutional investors, individual stockholders are already in an unfavorable position in the financial market because they do not have inside information and have the privilege of managing the same amounts of capital that institutional investors do. Whenever financial institutions move large blocks of shares, the market will be greatly influenced and the individual investors will suffer. The introduction of GP-based trading strategies enlarges the gap, since only financial institutions have access to these algorithms. Considering the volatility of public company stock, institutional investors are able to detect the best time to buy shares. Equipped with GP-based trading strategies, institutions make these trades automatically. As a result, individual investors have less opportunities to make profitable trading decisions. GP-based trading algorithms have an ethical problem according to Rawls's theory because individual investors are left worse off than before.

While individual investors do not benefit from the implementation of Genetic Programming in developing trading strategies, it could be beneficial to society as a whole. As a part of consequentialist theory, utilitarianism emphasizes the consequences instead of intentions. Mill believes that an ethical action is supposed to produce the greatest good for the greatest amount of people. Financial institutions not only make more profit by using GP-based trading algorithms but also shorten their research and development time of investigating other profitable trading strategies. With the implementation of GP, financial companies are able to reduce their

operating costs and increase their revenue; therefore, their utility increases during the process. On the other hand, individual investors have limited access to GP and thus their utility might decrease. However, financial institutions are the major participants in markets and they also represent the majority of people who entrust their money to these institutions. As a result, institutional investors carry more of the ethical burden than individual stockholders. According to Mill, GP-based trading strategies are ethical as they maximize the utility for the vast majority of society. Since society generally benefits from GP-based trading algorithms, Genetic Programming will be continually developed and enhanced.

Conclusion

Due to the special modeling process, Genetic Programming is widely implemented to handle optimization and other complicated problems. GP mimics natural evolution through genetic operations, which enables the program to efficiently obtain solutions. In comparison with classical search algorithms, GP has the advantages of a global perspective and higher accuracy for approximating discrete variables. Genetic Algorithms represent genotypes as fixed vectors whereas GP characterizes genotypes as variable tree structures. Equipped with tree structures, GP surpasses GAs in terms of its high-level symbolic representation for data, which enhances the flexibility and efficiency during the search and optimization processes. The superiority of GP makes it applicable for generating profitable trading strategies. By implementing GP, financial institutions will be better off from robust trading strategies, which ultimately benefit the vast majority of people. While a more widespread implementation of GP produces trading algorithms, computer professionals must be aware of the limitations of GP-based trading strategies because, due to the special modeling process, trading strategies created by GP models might confuse potential users who do not have solid knowledge about GP. While traditional trading strategies

are supported by financial theories and experiment results, the GP model cannot fully demonstrate its validity in the financial sector. Thus, when GP-based trading strategies perform poorly, especially during market downtrends, corporations may become apprehensive if computer professionals cannot offer convincing justification for them. In the future, computer researchers should search for stronger evidence in support of GP-based trading algorithms in order to appeal to a wider audience.

References

Aguilar, L. (2013). *Institutional investors: Powers and Responsibility*. Retrieved from: http://www.sec.gov

Armani, U. (2014). *Development of a hybrid genetic programming technique for computationally expensive optimization problems.* (Doctoral dissertation). Available from White Rose eTheses Dissertations and Theses database. (Identification No. uk.bl.ethos.631392)

Chang, E. C., Luo, Y., & Ren, J. (2014). Short-selling, margin-trading, and price efficiency: Evidence from the Chinese market. *Journal of Banking & Finance, 47*, 411-424. doi://doi.org/10.1016/j.jbankfin.2013.10.002

Chiu, Y. (2008, May). Update: Taiwanese software spots stock-market stinkers. *IEEE Spectrum, 45*(5), 1. doi: 10.1109/MSPEC.2008.4505296

Deb, D. (2014). *Genetic algorithm in search and optimization: The technique and applications.* Retrieved from: http://www.semanticscholar.org

Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E. W. T., & Liu, M. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, *36*(C), 534-551. doi://doi.org/10.1016/j.asoc.2015.07.008

Karatahansopoulos, A., Sermpinis, G., Laws, J., & Dunis, C. (2014). Modelling and trading the greek stock market with gene expression and genetic programing algorithms. *Journal of Forecasting, 33*(8), 596-610. doi:10.1002/for.2290

Khadka, M.S (2013). *A new forecasting method based on concordance and Genetic Programming.* (Doctoral dissertation). Available from ProQuest Dissertations and Theses database. (UMI No. 3598922)

Kroha, P., & Friedrich, M. (2014). Comparison of genetic algorithms for trading strategies. In V. Geffert, B. Preneel, B. Rovan, J. Štuller & A. M. Tjoa (Eds.), *SOFSEM 2014: Theory and practice of computer science: 40th international conference on current trends in theory and practice of computer science, nový smokovec, slovakia, january 26-29, 2014, proceedings* (pp. 383-394). Cham: Springer International Publishing. doi:10.1007/978-3-319-04298-5_34

Liu, C. (2014). *Finding technical trading rules in High-Frequency data by using genetic programming* (Masters dissertation). Available from ProQuest Dissertations and Theses database. (UMI No. 1568873)

Manahov, V (2014). *An investing of the behavior of financial markets using agent-based computational models*. (Doctoral dissertation, Newcastle University). Retrieved from: https://theses.ncl.ac.uk/dspace/handle/10443/2472

Manahov, V., Hudson, R., & Hoque, H. (2015). Return predictability and the "wisdom of Crowds": Genetic Programming trading algorithms, the Marginal Trader Hypothesis and the Hayek Hypothesis. *Journal of International Financial Markets, Institutions & Money*, *37*, 85-98. doi://doi.org/10.1016/j.intfin.2015.02.009

Stuart, R. (2013). *Genetic Programming for Security Analysis*. Retrieved from University of Pretoria School of Engineering website: http://www.cs.up.ac.za