Using Attribute Construction to Improve the Predictability of a GP Financial Forecasting Algorithm

Michael Kampouridis and Fernando E. B. Otero
School of Computing
University of Kent, Chatham Maritime, UK
Email: {M.Kampouridis, F.E.B.Otero}@kent.ac.uk

***Abstract*—Financial forecasting is an important area in computational finance. EDDIE 8 is an established Genetic Programming financial forecasting algorithm, which has successfully been applied to a number of international datasets. The purpose of this paper is to further increase the algorithm's predictive performance, by improving its data space representation. In order to achieve this, we use attribute construction to create new (high-level) attributes from the original (low-level) attributes. To examine the effectiveness of the above method, we test the extended EDDIE's predictive performance across 25 datasets and compare it to the performance of two previous EDDIE algorithms. Results show that the introduction of attribute construction benefits the algorithm, allowing EDDIE to explore the use of new attributes to improve its predictive accuracy.**

***Keywords*-genetic programming; financial forecasting; attribute construction**

I. Introduction

Financial forecasting is a vital area in computational finance [1]. There are numerous works that attempt to forecast the future price movements of a stock; several examples can be found in [2], [3]. EDDIE is a well-established genetic programming financial forecasting tool, which has been found to outperform traditional decision rule induction methods, such as C4.5, and return high accuracy results over different international stock markets [4], [5].

Recently, EDDIE 8 (ED8) [7], [8] was introduced, which is one of the latest algorithms from the EDDIE series. While previous EDDIE algorithms were using pre-specified periods for the indicators from technical analysis (e.g., *20 days* Moving Average, *50 days* Momentum), ED8 was the first algorithm to allow these periods to be directly selected by the GP. Thus, instead of the algorithm's user pre-specifying a number of fixed period values for the technical indicators, as it traditionally happens in both academia and industry, ED8 allowed the GP to evolve different periods for each technical indicator. As a result to the above modification, ED8 was able to produce new technical indicators, which improved the algorithm's predictive performance.

The purpose of this paper is to further improve the predictive performance of EDDIE, by using attribute construction. At the moment, ED8's trees are limited in testing conditions in the form of the triple (indicator, relational operator, threshold). For example, a tree from ED8 could be asked to evaluate the boolean *20 days Moving Average* > 0.95, where '20 days Moving Average' is an indicator derived from technical analysis [11], '$>$' is the relational operator, and '0.95' is a threshold value (real number). However, this method has the potential disadvantage that it does not allow for an effective search of the data space. This is because the attributes defining the search space —the set of indicators in this case— might be inadequate and new attributes might be needed to represent the regularities of the space.

To tackle the above limitation, we propose the use of attribute construction by extending EDDIE to allow the creation of new attributes during the construction of its trees. Our goal is to show that this extension is beneficial to the algorithm, and subsequently leads to improvements in its predictive performance.

The rest of this paper is organized as follows: Section II offers a general overview of the EDDIE 8 algorithm. Section III then explains how we incorporated the attribute construction in EDDIE. Sections IV and V then present the experimental setup and discuss the obtained results, respectively. Lastly, Section VI concludes this paper and discusses future work.

II. The EDDIE 8 algorithm

EDDIE 8 (ED8) is a Genetic Programming (GP) [9], [10] financial forecasting algorithm, which learns and extracts knowledge from a set of data. The question ED8 tries to answer is 'will the price increase within the n following days by $r\%$?' The user first feeds the system with a set of past data; EDDIE then uses this data and through a GP process, it creates and evolves Genetic Decision Trees (GDTs), which make recommendations of buy (1) or not-to-buy (0).

The set of data used is composed of three parts: (i) daily closing price of a stock, (ii) a number of attributes, and (iii) signals. Stocks' daily closing prices can be obtained online on websites such as `http://finance.yahoo.com` and from financial statistics databases like *Datastream*.[1]

[1]Available at: `http://thomsonreuters.com/datastream-professional/`

978-1-4799-2528-5/13 $31.00 © 2013 IEEE
DOI 10.1109/.22

<Tree> ::= `If-Then-Else` <Condition> <Tree> <Tree> | Decision
<Condition> ::= <Condition> `AND` <Condition> |
<Condition> `OR` <Condition> |
`NOT` <Condition> |
<VarConstructor> <RelationOperation> Threshold
<VarConstructor> ::= MA period | TBR period | FLR period | Vol period
| Mom period | MomMA period
<RelationOperation> ::= “>” | “<” | “=”
Terminals:
MA, TBR, FLR, Vol, Mom, MomMA are function symbols
Period is an integer within a parameterized range, [MinP, MaxP]
Decision is an integer, Positive or Negative implemented
Threshold is a real number

Figure 1. The Backus Normal Form of ED8

The attributes are indicators commonly used in technical analysis [11]; which indicators to use depends on the user and his belief of their relevance to the prediction. The technical indicators used in this work are: Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).[2]

The signals are calculated by looking ahead of the closing price for a time horizon of n days, trying to detect if there is an increase of the price by $r\%$ [5]. For this set of experiments, n was set to 20 and r to 4%. In other words, the GP is trying to use some of the above indicators to forecast whether the daily closing price is going to increase by 4% within the following 20 days.

After feeding the data to the system, EDDIE creates and evolves a population of GDTs. Figure 1 presents the Backus Normal Form (BNF) [15] (grammar) of ED8. The root of the tree is an `If-Then-Else` statement. The first branch is either a boolean (testing whether a technical indicator is greater than/less than/equal to a value), or a logic operator (`AND`, `OR`, `NOT`), which can hold multiple boolean conditions. The `Then` and `Else` branches can be a new GDT, or a decision, to buy (`1`) or not-to-buy (`0`).

As we can observe from the grammar in Figure 1, there is a function called *“VarConstructor”*, which takes two children. The first one is the indicator, and the second one is the “Period”. “Period” is an integer within the parameterized range [MinP, MaxP] that the user specifies. The advantage of this approach is that ED8 is not constrained to pre-specified periods, as is usually the case in industry.[3] As a consequence, it is up to the GP and the evolutionary process to look for the optimal periods values from the period range provided. For instance, if this range is 2 to 65 days, then ED8 can create Moving Averages with any of these periods, e.g., 20 days MA, 25 days MA, and so on. Furthermore, the periods are leaf nodes and are thus subject to genetic operators, such as crossover and mutation. A sample GDT of ED8 is presented in Figure 2. As we can see, if the *20 days MA* is less than *6.4*, then the user is advised to buy; otherwise, the user is advised to consult another GDT, which is located in the third branch (“else-branch”) of the tree. As explained, the periods 20 and 50 of the figure’s sample tree are leaf nodes; the advantage of this being that the GP can replace them with other, more effective periods, which might have come up during the evolutionary process.

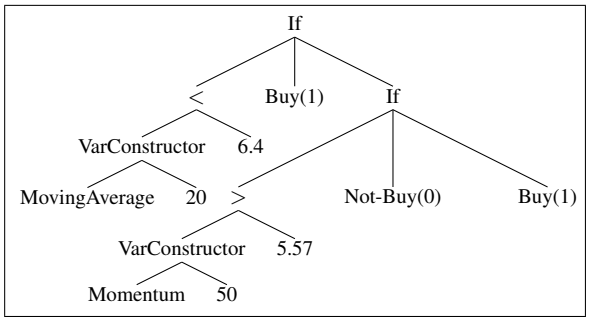

Figure 2. Sample GDT generated by ED8.

Depending on the classification of the predictions, we can have four cases: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). As a result, we can use the metrics presented in Equations 1, 2 and 3:

Rate of Correctness

$$RC = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Rate of Missing Chances

$$RMC = \frac{FN}{FN + TP} \quad (2)$$

Rate of Failure

$$RF = \frac{FP}{FP + TP} \quad (3)$$

The above metrics combined give the following fitness

[2]We use these indicators because they have been proved to be quite useful in developing GDTs in previous works like [12], [13] and [14]. Of course, there is no reason not to use other information like fundamentals or limit order book. However, the aim of this work is not to find the ultimate indicators for financial forecasting.

[3]In the literature, the users of similar algorithms pre-specify certain periods that they consider useful. For instance, *20 days* MA, and *50 days* MA. The indicators (e.g., MA) together with their respective period (e.g., 20) are treated by the GP as a single leaf node. Thus, the numbers 20 and 50 cannot change during the evolutionary process. In our previous work [7], [8], we questioned this approach, because nobody can guarantee that, for instance, a 20 days MA is better than a 25 days MA. To address this issue, we created ED8, which is able to search in the space of technical indicators and their periods.

function, presented in Equation 4:

$$ff = w_1 * RC - w_2 * RMC - w_3 * RF \quad (4)$$

where w_1, w_2 and w_3 are the weights for RC, RMC and RF respectively. These weights are given in order to reflect the preferences of investors. Thus, a conservative investor would avoid failure; thus a higher weight for RF should be used. For the experiments of this paper, the focus is on strategies that mainly target correctness and reduced failure.

This concludes this short presentation of ED8. For more detailed presentations of the EDDIE 8 algorithm, we refer the reader to [7], [8]. The next section discusses attribute construction and explains how it is incorporated in EDDIE.

III. Incorporating Attribute Construction in EDDIE

It is well known that the classification performance is dependent of the quality of the data space representation (i.e., attributes of the data) [19], [20]. If the attributes defining the data space—the set of indicators in the case of EDDIE—are inadequate, it becomes more difficult to create GDTs with a high predictive quality. Attribute construction (also known as feature construction) [19] is usually employed in order to mitigate the potential problem of inadequate attributes. The goal of attribute construction is to create new (high-level) attributes from the original (low-level) attributes, improving the data space representation. The main rationale is that even when the original attributes are individually inadequate, they can be combined to create new attributes with greater predictive power than the original ones, effectively creating a new data space representation where the regularities are more apparent.

Attribute construction methods can be divided into two categories with respect to the use of a classification algorithm, namely the filter and wrapper methods:

Filter methods: are those methods where the attribute construction is independent of the classification algorithm that will be used to create the final classification model. New attributes are created and evaluated by directly analysing the data, without running a classification algorithm, using information-theoretic measures (e.g., based on the entropy measure). One clear advantage is that filter methods tend to be more computational efficient; another advantage is that the results are expected to have a more generic usefulness, since they are not specifically created for a classification algorithm (i.e., their quality is not determined by a specific classification algorithm).

Wrapper methods: are those methods where the attribute construction includes the use of a classification algorithm. When new attributes are created, their quality is evaluated by running a classification algorithm. As a consequence, the created attributes tend to be specific for the classification algorithm, limiting their usefulness for different classification algorithms. One of the main drawbacks of wrapper methods is that they tend to be computationally expensive, since they require the execution of a classification algorithm to evaluate a candidate new attribute—many executions when several different candidate attributes need to be evaluated.

Genetic Programming has been successfully used in attribute construction, following both filter and wrapper strategies. Hu [20] proposed a GP for attribute construction called GPCI following the filter strategy. The terminal set of the GP consists of the booleanized original attributes and the function set consists of the `AND` and `OR` operators; each individual represent a candidate new attribute, created by combining the (now boolean) attributes using the `AND` and `OR` operators. Otero et al. [21] proposed a filter method that uses a GP for attribute construction, without requiring the boolean transformation of the original attributes. Krawiec followed a wrapper strategy in the GP for attribute construction proposed in [22]. More recently, Neshatian et al. [23] proposed a GP-based filter method with the distinctive feature of being able to create multiple new attributes in a single execution. Other GP-based attribute construction methods can be found in [24], [26].

In this paper we propose to incorporate attribute construction in EDDIE in order to improve the predictive accuracy of the created GDTs. Since EDDIE is essentially a GP algorithm, which can be used for attribute construction, we do not follow the filter nor the wrapper strategy. We extended EDDIE to allow the creation of new attributes during the construction of the GDTs. In other words, EDDIE will evolve GDTs that can evolve new attributes at the same time—new attributes are evolved as new boolean conditions that can appear in tests (first branch of a `If-Then-Else` statement) of GDTs. This strategy resembles the embedded feature selection strategy, where the feature selection occurs as part of the classification model creation.[4] Note that our aim is not to propose a new attribute construction method; we are interested in allowing EDDIE decide if new attributes are needed or not to improve the predictive accuracy.

A. The New EDDIE 8-ATTR Algorithm

The current version of EDDIE only creates GDTs involving the combination of tests composed by a triple (attribute, operator, value), where the value is a numeric constant, as most of machine learning algorithm used for knowledge discovery. In order to allow the creation of new attributes, EDDIE's grammar is extended to allow the creation of tests involving the direct comparison of indicator values, presented in Figure 3—the new version is called EDDIE 8-ATTR (ED8-ATTR).

The main modification is the introduction of the production "<*VarConstructor*> <*RelationOperation*>

[4]Refer to [27] for a discussion of different feature selection strategies and methods.

<Tree> ::= If-Then-Else <Condition> <Tree> <Tree> | Decision
<Condition> ::= <Condition> AND <Condition> |
<Condition> OR <Condition> |
NOT <Condition> |
<VarConstructor> <RelationOperation> Threshold |
<VarConstructor> <RelationOperation> <VarConstructor>
<VarConstructor> ::= MA period | TBR period | FLR period | Vol period
| Mom period | MomMA period
<RelationOperation> ::= “>” | “<” | “=”
Terminals:
MA, TBR, FLR, Vol, Mom, MomMA are function symbols
Period is an integer within a parameterized range, [MinP, MaxP]
Decision is an integer, Positive or Negative implemented
Threshold is a real number

Figure 3. The Backus Normal Form of ED8-ATTR

<VarConstructor>” to the symbol “*<Condition>*”, which defines the rules for creating the conditions of If-Then-Else statements of the GDTs. The new grammar allows ED8-ATTR to create GDTs with the same structure as ED8 and also GDTs that can define new attributes, in a similar fashion as GP-based attribute construction methods [20], [21], [22]—i.e., creating new boolean conditions combining indicators (attributes) using AND, OR and NOT operators. A sample GDT of ED8-ATTR is presented in Figure 4. It is important to emphasize that this GDT could not be created by the original ED8, since it involves a condition comparing indicator values directly—i.e., a new boolean attribute represented by the condition “MovingAverage 20 $>$ Momentum 50”.

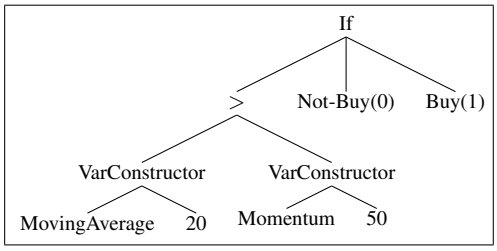

Figure 4. Sample GDT generated by ED8-ATTR using a new boolean attribute represented by the condition “MovingAverage 20 $>$ Momentum 50”.

IV. Experimental setup

Our goal is to investigate whether the introduction of attribute construction is beneficial to the EDDIE algorithm. We are thus going to compare the performance of ED8-ATTR to ED8, and also its predecessor, EDDIE 7 (ED7), which uses pre-specified periods for the technical indicators. The reason for also using the latter algorithm is because although ED8 has been proven superior to the previous EDDIE versions, there can be datasets that it is outperformed by ED7, as it has previously been shown in [7], [8]. We thus consider it important to test the performance of ED8-ATTR to both ED7 and ED8.

Table I
GP parameters values used in the experiments.

GP Parameters	
Max Initial Depth	6
Max Depth	8
Generations	50
Population size	500
Tournament size	2
Reproduction probability	0.1
Crossover probability	0.9
Mutation probability	0.01
Weight $\{w_1, w_2, w_3\}$	{0.6, 0.1, 0.3}
Period (ED8)	[2,65]

In addition to the performance metrics mentioned in Section II (i.e., fitness, RC, RMC, RF), we also use an additional metric for the comparison of the algorithms. This metric is related to the return the algorithm yields, and is called Average Annualized Rate of Return (AARR). The formula for this metric is presented in the Appendix. It should be stated that AARR is not part of the fitness function. However, rate of return is a very important investment metric, and that is why we use it as a reference.

Furthermore, we run tests for 25 datasets. These datasets consist of daily closing prices from 18 stocks from FTSE 100, and 7 international indices. The 18 FTSE 100 stocks are: Aggreko, Amlin, Barclays, British Petroleum (BP), Cadbury, Carnival, Easyjet, First, Hammerson, Imperial Tobacco, Marks & Spencer, Next, Royal Bank of Scotlland (RBS), Schroders, Sky, Tesco, Vodafone and Xstrata. The 7 indices are: Athens Stock Exchange (Greece), DJIA (USA), HSI (Hong Kong), MDAX (Germany), and NASDAQ (USA), NIKEI (Japan), and NYSE (USA). The training period is 1000 days and the testing period 300.

The GP parameters are presented in Table I. For statistical purposes, the GP is run 50 times. The process is as follows. We create a population of 500 GDTs, which are evolved for 50 generations, over a training period of 1000 days. At the last generation, the best performing GDT in terms of fitness is saved and applied to the testing data. As already explained, this procedure is done for 50 individual runs.[5] Next session presents and discusses the results for the above experiments.

V. Results

Table II presents the Mean and Best results for the 5 performance metrics tested in this paper. The higher ranking algorithm per test is indicated by bold fonts. As we can observe, ED8-ATTR gets the top ranking in 8 out of the 10 cases we investigated. More specifically, ED8-ATTR ranks first in all 5 tests in terms of Mean results. In addition, in the case of Best results, ED8-ATTR ranks first for the metrics of Fitness, RC and RMC.

[5]We do not argue that these are the optimal GP parameters. Nevertheless, experience from previous EDDIE experiments has shown that the above GP parameters return effective results.

Table II
FRIEDMAN RANKINGS FOR THE MEAN ANS BEST RESULTS OF ED7, ED8 AND ED8-ATTR.

		Fitness	RC	RMC	RF	AARR
Mean	ED7	2.12	2.16	1.98	2.24	2.12
	ED8	2.04	2.00	2.14	1.92	2.12
	ED8-ATTR	**1.84**	**1.84**	**1.88**	**1.84**	**1.76**
Best	ED7	1.96	1.94	2.14	**1.80**	**1.88**
	ED8	2.20	2.24	2.2	2.12	2.00
	ED8-ATTR	**1.84**	**1.82**	**1.64**	2.08	2.12

Subsequent analysis on the Holm post-hoc test [17], [18] showed that the above results were not significant at 5% level. However, this should not alarm us, because the fact remains that ED8-ATTR was ranked first across the majority of the tests, and specially in all the results concerning the Mean performance. This suggests that ED8-ATTR is a robust algorithm and is expected to produce better results when new data is used. It also indicates that introducing attribute construction does not have a negative impact on the performance of the algorithm, even thought we are effectively increasing the search space of the GP, and instead an overall improvement is observed.

The above results are very important because they demonstrate the superiority of ED8-ATTR across its predecessors. These results also show us that the use of attribute construction has enabled EDDIE to improve its data space representation, and thus return improved forecasting results.

To further investigate the above, we looked into the trees returned by the Hammerson dataset, which was the dataset that ED8-ATTR had the biggest positive effect. After looking into the productions of the tree that yielded the best (testing) fitness results, we found that it was using 10 productions of the form "<*VarConstructor*> <*RelationOperation*> <*VarConstructor*>", and another 10 of the form "<*VarConstructor*> <*RelationOperation*> <*Threshold*>". On the other hand, when we looked into the productions of the worse performing tree of Hammerson, we found that it was only using 6 productions of the form "<*VarConstructor*> <*RelationOperation*> <*VarConstructor*>", and 20 of the form "<*VarConstructor*> <*RelationOperation*> <*Threshold*>". The former tree had a fitness of around 70%, whereas the latter tree had a fitness of about 40%. The difference is of course quite significant. One reason that explains why the first tree performed so well could be because of the fact that it was using more productions (at a rate of 50% of the total productions—10 out of 20) that allow the direct comparison of indicators (i.e., $VarConstructor$). On the other hand, the second tree was only using the new production at a rate of 23% (6 out of 26 productions). Hence, it could be argued that the higher rate of the "<*VarConstructor*> <*RelationOperation*> <*VarConstructor*>" production attributed to the superiority of the first tree.

Table III
AVERAGE COMPUTATIONAL TIME OF A SINGLE RUN ON THE HAMMERSON DATASET.

ED7	ED8	ED8-ATTR
32.26 secs	43.14 secs	44.5 secs

Lastly, Table III presents the average computational time of a single run[6] of each algorithm for the Hammerson dataset. As we can observe, ED8-ATTR is only 1.5 seconds slower than ED8, which is 11 seconds slower than ED7. Overall, these differences are minimal and we can thus conclude that the introduction of attribute construction in EDDIE 8 did not have any significant negative effect on the computational time, while it led to important improvements in the performance metrics.

VI. CONCLUSION

To conclude, this paper presented work on EDDIE 8 (ED8), which is an established GP financial forecasting algorithm. As we explained, a limitation of ED8 is that its trees are constrained in testing conditions in the form of the triple (indicator, relational operator, threshold). However, as this method has the potential disadvantage of ineffective search of the data space, we extended EDDIE to allow the creation of new attributes during the construction of its trees. Results showed that this attribute construction was beneficial to the algorithm, which was able to outperform its two predecessors, EDDIE 7 and EDDIE 8, in 8 of the 10 tests examined. Results also indicated that the introduction of more productions that allow the direct comparison of indicators in a single tree, can have a significantly positive effect to the tree's predictive performance.

Lastly, since we observed performance improvements in the GDTs that use the new grammar production, we believe it would be interesting to implement a mechanism to promote its use. Future work could also focus on using arithmetic operators to combine the indicator values. This could lead to an even better representation of the data space and thus further improvements in the forecasting results.

APPENDIX

Here we present the formulas for the additional performance metric AARR [4]. We would once again like to remind the reader that this metric should be used for reference only, since it is not part of the fitness function.

Hypothetical Trading Behaviour*: We assume that when a positive position is predicted by a GDT, one unit of money*

[6]We ran each algorithm for 50 individual runs, and then divided each resulted computational time by 50.

is invested in a stock reflecting the current closing price. If the closing price does rise by r% or more at day t within the next n trading days, we then sell the portfolio at the closing price of day t. If not, we sell the portfolio on the n_{th} day, regardless of the price.

Given a positive position predicted, for example, the i_{th} positive position, for simplicity, we ignore transaction cost, and annualize its return by the following formula:

$$ARR_i = \frac{250}{t} * \frac{P_t - P_0}{P_0} \tag{5}$$

Where P_0 is the buy price, P_t is the sell price, t is the number of days in markets, 255 is the number of total trading days in one calendar year. Given a GDT that generates N_+ number of positive positions over the period examined, its average ARR is shown in Equation (6):

$$AARR = \frac{1}{N} \sum_{i=1}^{N_+} ARR_i \tag{6}$$

REFERENCES

[1] E. Tsang and S. Martinez-Jaramillo, "Computational finance," *IEEE Computational Intelligence Society Newsletter*, pp. 3–8, 2004.

[2] S.-H. Chen, *Genetic Algorithms and Genetic Programming in Computational Finance*. Springer-Verlag New York, LLC, 2002.

[3] J. Binner, G. Kendall, and S.-H. Chen, Eds., *Applications of Artificial Intelligence in Finance and Economics*, ser. Advances in Econometrics. Elsevier, 2004, vol. 19.

[4] J. Li, "FGP: A genetic programming-ased financial forecasting tool," Ph.D. dissertation, Department of Computer Science, University of Essex, 2001.

[5] E. Tsang, J. Li, S. Markose, H. Er, A. Salhi, and G. Iori, "EDDIE in financial decision making," *Journal of Management and Economics*, vol. 4(4), 2000.

[6] M. Kampouridis and E. Tsang, "EDDIE for investment opportunities forecasting: Extending the search space of the GP," in *Proceedings of the IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 2010, pp. 2019–2026.

[7] ——, "Investment opportunities forecasting: Extending the grammar of a gp-based tool," *International Journal of Computational Intelligence Systems*, vol. 5, no. 3, pp. 530–541, 2012.

[8] J. Koza, *Genetic Programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press, 1992.

[9] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[10] R. Edwards and J. Magee, "Technical analysis of stock trends," *New York Institute of Finance*, 1992.

[11] S. Martinez-Jaramillo, "Artificial financial markets: An agent-based approach to reproduce stylized facts and to study the red queen effect," Ph.D. dissertation, CFFEA, University of Essex, 2007.

[12] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial Economics*, vol. 51, pp. 245–271, 1999.

[13] M. Austin, G. Bates, M. Dempster, V. Leemans, and S. Williams, "Adaptive systems for foreign exchange trading." *Quantitative Finance*, vol. 4(4), pp. 37–45, 2004.

[14] J. Backus, "The syntax and semantics of the proposed international algebraic language of Zurich," in *International Conference on Information Processing*. UNESCO, 1959, pp. 125–132.

[15] E. Tsang, S. Markose, and H. Er, "Chance discovery in stock index option and future arbitrage," *New Mathematics and Natural Computation, World Scientific*, vol. 1(3), pp. 435–447, 2005.

[16] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[17] S. Garcia, F. Herrera, and J.Shawe-Taylor, "An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons", *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[18] Y-J. Hu, "A Genetic Programming Approach to Constructive Induction," *Proc. of the 3rd Anual Genetic Programming Conference*, pp. 146-151, 1998.

[19] YJ. Hu, "Constructive Induction: Covering Attribute Spectrum," *Feature Extraction Construction and Selection*, pp. 257-272, 1998.

[20] F.E.B. Otero, M.M.S. Silva, A.A. Freitas, and J.C. Nievola, "Genetic Programming for Attribute Construction in Data Mining," *Proc. of EuroGP*, LNCS 2610, pp. 384–393, 2003.

[21] K. Krawiec, "Genetic programming-based construction of features for machine learning and knowledge discovery tasks," *Genetic Programming and Evolvable Machines*, vol. 3(4), pp. 329–343, 2002.

[22] K. Neshatian, M. Zhang, and P. Andreae, "A Filter Approach to Multiple Feature Construction for Symbolic Learning Classifiers Using Genetic Programming," *IEEE Transaction on Evolutionary Computation*, vol. 16, no. 5, pp. 645–661, 2012.

[23] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Trans. Syst. Man Cybern. B Cybern.*, vol. 35, no. 1, pp. 89–99, 2005.

[24] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genet. Programming Evolvable Mach.*, vol. 6, no. 3, pp. 265–281, 2005.

[25] I. Witten, E. Frank, and M.A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques", 3rd. ed., Morgan Kaufmann, 2011, 664 pages.