Forecasting stock prices using Genetic Programming and Chance Discovery

Alma Lilia Garcia-Almanza and Edward P.K. Tsang

Department of Computer Science
University of Essex
Wivenhoe Park, Colchester, CO4 3SQ, UK
algarc@essex.ac.uk, edward@essex.ac.uk

Abstract. In recent years the computers have shown to be a powerful tool in financial forecasting. Many machine learning techniques have been utilized to predict movements in financial markets. Machine learning classifiers involve extending the past experiences into the future. However the rareness of some events makes difficult to create a model that detect them. For example bubbles burst and crashes are rare cases, however their detection is crucial since they have a significant impact on the investment. One of the main problems for any machine learning classifier is to deal with unbalanced classes. Specifically Genetic Programming has limitation to deal with unbalanced environments. In a previous work we described the Repository Method, it is a technique that analyses decision trees produced by Genetic Programming to discover classification rules. The aim of that work was to forecast future opportunities in financial stock markets on situations where positive instances are rare. The objective is to extract and collect different rules that classify the positive cases. It lets model the rare instances in different ways, increasing the possibility of identifying similar cases in the future. The objective of the present work is to find out the factors that work in favour of Repository Method, for that purpose a series of experiments was performed.

1 Introduction

In recent years computers have shown to be a powerful tool in financial applications, for that reason many machine learning techniques has been applied to financial problems. In particular, the use of Genetic Algorithms (GAs), for financial purposes, has increased considerable. As an instance [1] provides excellent readings from various areas of computational finance. GAs are algorithms that emulate evolution and natural selection to solve a problem. A population of *individuals* is evolved applying the *crossover* and *mutation* operators. The selection is based on the *fitness function*, it determines how likely individuals are to reproduce [2]. Genetic Programming (GP) [3] is an evolutionary technique whose population is composed by programs (decision trees).

Financial forecasting is one of the most important areas in computational finance [4]. Tsang et. al [5][6][7] used GP to create a forecasting financial tool whose objective is to detect opportunities for supporting financial decisions. The idea behind this approach is to train a GP with a data set in order to create a decision tree that model this

data. However, when the number of opportunities is extremely small it is very complicated to model the data behavior. For example bubbles, burst and crashes are rare cases, however their detection is crucial since they have a significant impact on the investment.

This work is motivated by the interest to find rare opportunities. Chance Discovery is the discipline that studies the detection of rare, but significant events that may have an important impact [8][9]. When the frequency of an event is very low the training data contains very few positive cases in comparison with the negative cases (unbalanced classes). Machine learning techniques has serious problems to manage unbalanced classes. Specifically GP has limitation to deal with this feature because this favours negative classifications, which has a high chance of being correct due to the nature of the data. During the evolutionary process rules that classify rare cases has low fitness, because they contain few cases, as a consequence they tend to be eliminated by the evolutionary process. To cope with imbalanced classes some techniques has been developed, as an instance [10] presents a compilation of those techniques. In [11] we proposed Repository Method (RM) to deal with unbalanced classes, this method gathers *rules*[1] that contribute with the classification task, taking care of the rule variety. RM was able to improve the *recall* [2] and the *precision*[3] in the prediccions. The objective of the present work is to analyse the factors that work in favour of RM in order to fully identify the impact of each of them.

The remainder of this paper is organized as follows: Section 2 describes the problem that exemplify RM; Section 3 briefly describes the procedure and results of RM, while Section 4 presents the experiments to test RM factors. Finally, Section 5 summaries the conclusions.

2 Problem Description

The objective is to train a GP in order to create a classifier that discover classification rules for predicting future movements in stock prices. This problem has been addressed previously by [5][6][12]. The data to train the classifier describes the behaviour of the stock price by means of a set of attributes or independent variables [13]. These are indicators derived from financial technical analysis. Technical analysis is used in financial markets to analyse the price behaviour of stocks. This is mainly based on historical prices and volume trends. In addition every case in the data set has a signal, it indicates the opportunities for *buying* or *not buying* and *selling* or *not selling*. The signal is calculated looking ahead in a future horizon of n units of time, trying to detect an increase or decrease of at least $r\%$.

3 Repository Method

This section presents a brief overview of RM, more detailed information about this method and its experimental results are available in [11]. The objective of RM is to

[1] A Rule is a minimal set of conditions that satisfy a decision tree

[2] Recall is the proportion of positive cases that were correctly identified

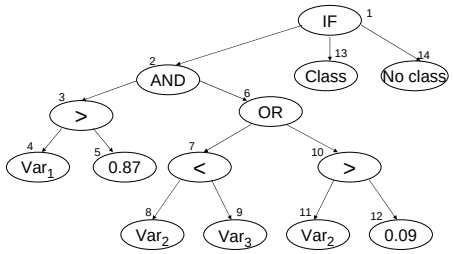
[3] Precision is the proportion of the predicted positive cases that were correct

mine the knowledge acquired by the evolutionary process in order to compile several rules that model the rare cases in diverse ways. Since the number of positive examples is very small, it is important to gather all available information about them. The over-learning produced by this method attempts to compensate the lack of positive cases in the data set. A GP system is able to produce multiple solutions for a single problem. However, the normal procedure is to choose only the best individual of the evolution as the optimal solution of the problem. Regardless that a GP process spends a lot of computational resources evolving complete populations for several generations the rest of the population is discarded. In [11] we showed that the remaining individuals could contain useful information that is not necessarily included in the best individual.

3.1 Repository Method Procedure

A brief description of RM procedure is given in this section, detailed information is available in [11]. In order to understand RM procedure, let *T* be a decision tree generated by a GP. Let define *condition* as the equation that makes a comparison between *i)* two variables (e.g. $Var_1 > Var_2$) or *ii)* one variable and a threshold (e.g. $Var_1 > 0.9$). Let R_k be a rule from tree T, where R_k is defined as a minimal set of conditions, whose intersection satisfy T (see Figure 1).

1. Rule extraction: This involves the analysis of decision trees in order to delimit their rules. Once a rule $R_k \in T$ has been identified this is individually evaluated against the training data. If the precision of R_k achieves a predefined Precision Threshold (PT), then R_k is considered for the next step (rule simplification), otherwise R_k is discarded. In other words we have to identify all minimal set of conditions in tree T, as Figure 1 shows.



R_1 = (Var_1 > 0.87) AND (Var_2 < Var_3) = { C_3, C_7 }
R_2 = (Var_1 > 0.87) AND (Var_2 > 0.09) = { C_3, C_{10} }
Where C_i is the condition composed by nodes *i*, *i* + 1 and *i* + 2

Fig. 1. Example of a decision tree and its rules R_1 and R_2

2. Rule Simplification: The simplification of rules could be a hard task, specially in advanced stages of the evolutionary process, due to decision trees generated by

GP tend to grow. Unfortunately this growth does not necessarily contribute with quality of the proposed solution [14][15][16][17]. The simplification involves *i)* removing noisy and redundant conditions from R_k and *ii)* reducing conditions that hold thresholds, for instance conditions ($Var_1 > .50$) and ($Var_1 > .89$) can be replaced by $Var_1 > .89$.

3. New rule detection: Once the rule R_k has been simplified, it is possible to compare this against the rules in the repository and determine if R_k is not present in that collection. Thus the following actions will be taken:
   - If R_k is a new rule this is be added to the repository
   - If there is a rule R_i in the repository such as R_i and R_k are *similar rules*[4] and Performance(R_i) $<$ Performance(R_k) then R_i is replaced by R_k.
   - Otherwise R_k is discarded

3.2 Repository Method results

The experimental results, reported in [11], showed that by combining rules from the whole population and accumulating these from previous generations RM outperformed the best tree generated by GP, improving the precision and recall.

3.3 Factors that benefit Repository Method

In this paper we examine the impact of the following factors on the RM:

- The collection of rules in a wider part of the population. It means that combining rules from several individuals or even the complete population, it is possible to gather more information about the positive cases in the data set.
- The accumulation of rules from previous generations. We presume that some useful rules can be lost in the evolutionary process, for that reason, it is important to compare the performance of RM when it is composed by rules of decision trees from just a specific generation and the performance of RM when it accumulates rules from previous generations.

A series of experiments was carried out to find out the impact of the mentioned factors in RM performance.

4 Experiments Description

To study the impact of each of the factors that benefit RM a series of experiment was carried out. This section describes the experimental design and results.

[4] R_k and R_i are *similar rules* if they have the same *hard conditions* and *similar flexible conditions*. A hard condition is a comparison of two variables (e.g. $var_1 < var_2$). A flexible condition is the equation between a variable and a threshold (e.g.$var_1 < .8$). When two flexible conditions have the same variable and operator they are defined as *similar conditions* (e.g. $var_1 < 3$ and $var_1 < 2$ are similar conditions)

Table 1. Financial indicators used in the experiment

Indicator name	Short period (Days)	Long period (Days)
Price moving average	12	50
Price Trading breaking rule	5	50
Filter rule	5	63
Price volatility	12	50
Volumen moving average	10	60
Momentum	10	60
Momentum 10 days moving average	10	–
Momentum 60 days moving average	60	–
Generalized Momentum indicator	10	60
FOOTSIE moving average	12	50
LIBOR: 3 months moving average	12	50

4.1 Population creation

In order to analyse the factors that work in favour or RM, this was tested with a series of populations from different stages of the evolutionary process. These populations were created as follows: A population of 1,000 individuals was evolved using a GP system during 100 generations. Every ten generations the entire population was saved, lets call them $P_{10}, P_{20} \ldots, P_{100}$. This process was repeated twenty times. All experimental results given in the subsequent sections are grouped and averaged by generation and PT.

4.2 Data sets description

This section describes the data sets that were used to train and test the GP system. Every data set contain information from the London stock market. Each of them are comprised by 890 records from Barclays stock (from March, 1998 to January, 2005). The attributes of each record are composed by indicators derived from financial technical analysis; these were calculated on the basis of the daily *closing price*[5], volume and some financial indices as the FTSE[6] and LIBOR[7]. The financial indicators used to forecast the data are listed in Table 1. These were calculated using two periods of time, one short and one long. Each period provides a different interpretation. The shorter the time span, the more sensitive the indicator will be to changes. We looked for events when the price decrease in 15% in a horizon of 10 days.

[5] The settled price at which a traded instrument is last traded at on a particular trading day.

[6] An index of 100 large capitalization companies stock on the London Stock Exchange, also known as "Footsie

[7] London interbank offered rate

4.3 Experimental procedure

To find out the factors that favour RM a series of experiments was performed. Lets Accumulated Repository Method (ARM) be the procedure that collects rules through different generations (This experiment was designed to accumulate rules just from previous generations that are multiple of 10) and let Simple Repository Method (SRM) be the process that compiles rules just from a specific generation. *AR* and *SR* denote two rules repositories created by ARM and SRM respectively. Subscripts indicates the number of top individuals that were analyzed to collect rules. Superscript specifies the generation that was analyzed to compile rules. For instance, AR_{100}^{70} is a repository that compiles rules from the top 100 individuals, these rules were accumulated during generations 10,20,...,70. On the other hand SR_{10}^{70} is a repository that comprises rules from the top 10 individuals in just generation 70.

Table 2. Simple Repository Method results. The table displays the recall, precision and accuracy for (a) Best Individual, (b) SR_{10} (c) SR_{100} (d) SR_{1000}. where SR represents an simple repository of rules and subscripts denote the number of top individuals used to created the repository. The Precision Threshold is 60%

Gen	Recall				Precision				Accuracy			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
10	1%	2%	2%	1%	1%	1%	2%	1%	97%	96%	96%	97%
20	0%	1%	3%	4%	0%	3%	3%	2%	96%	97%	96%	95%
30	12%	2%	7%	12%	2%	1%	3%	5%	94%	95%	94%	93%
40	11%	4%	7%	20%	4%	3%	3%	6%	90%	95%	93%	91%
50	11%	3%	14%	28%	6%	2%	5%	8%	93%	94%	92%	90%
60	7%	6%	13%	31%	6%	5%	5%	9%	94%	94%	92%	89%
70	17%	7%	17%	35%	6%	5%	6%	8%	87%	94%	91%	88%
80	10%	10%	21%	41%	4%	7%	6%	9%	93%	93%	90%	87%
90	6%	12%	20%	36%	7%	6%	6%	8%	94%	93%	90%	87%
100	14%	14%	23%	41%	2%	7%	7%	8%	88%	92%	90%	86%

To find out the effects of the number of individuals involved in the rule search, we applied SRM in order to create three simple repositories. The first gathers rules from the top ten individuals of the population, the second collects rules form the top 100 individuals and finally the third compiles rules in the complete population (1000 individuals). Lets call them SR_{10},SR_{100} and SR_{1000}, respectively. This process was applied to population $P_{10}, P_{20}, \ldots, P_{100}$ described in 4.1, using PT = 60%. Table 2 shows the recall (b), precision (d) and accuracy (f) of this experiment.

In order to test the effects of rule accumulation the previous experiment was repeated for ARM, results are displayed in Table 3. Figure 2 shows the graphs that compare the performance of the best individual against the performance of the experiment (a),(c),(e).

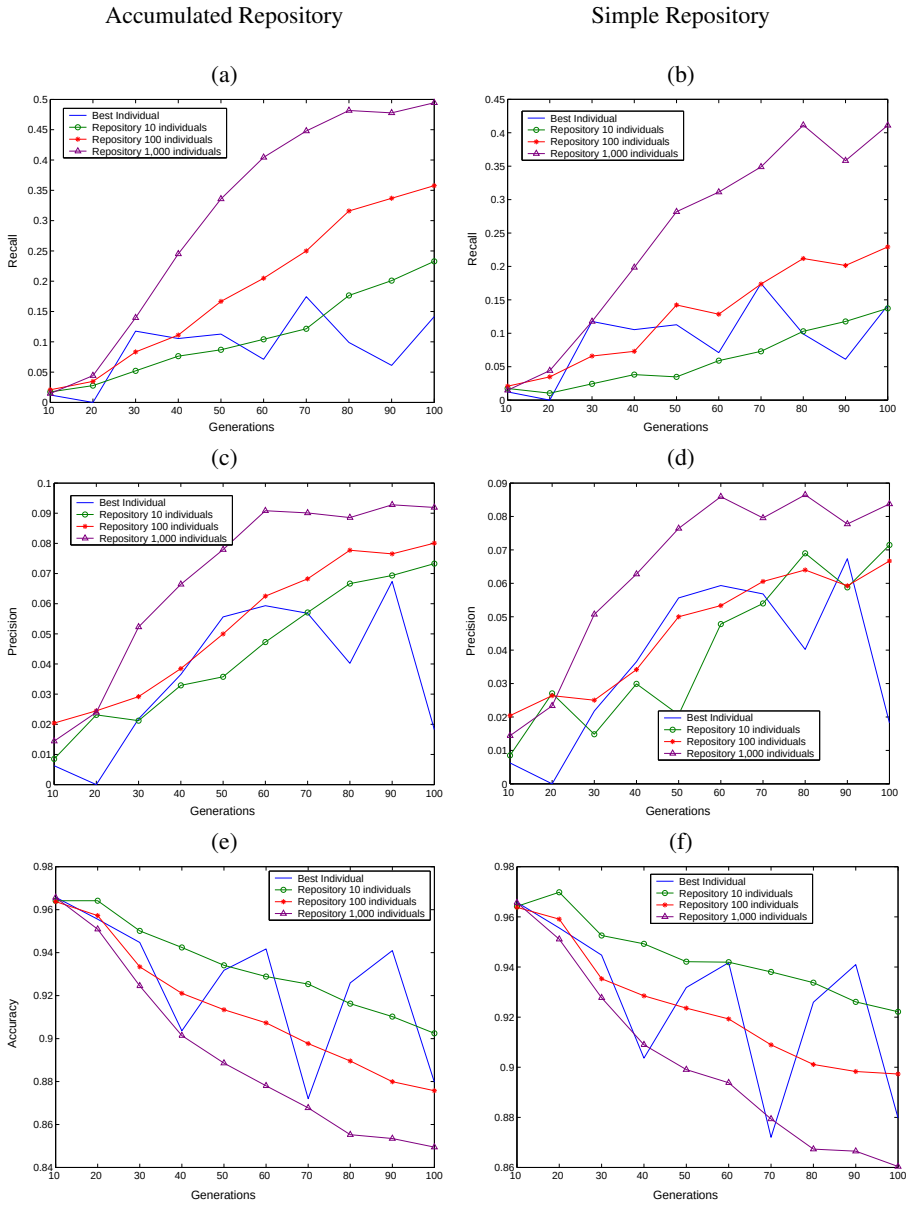


Fig. 2. Graphs show the results using Accumulated Repository Method (ARM) and Simple Repository Method (SRM), the experiment was performed using a PT=60% (a) ARM Recall, (b) SRM Recall, (c) ARM Precision, (d) SRM Precision, (e) ARM accuracy, (f) SRM Accuracy

Table 3. Accumulated Repository (AM) results. The table displays the recall, precision and accuracy for (a) Best Individual, (b) AR_{10} (c) AR_{100} (d) AR_{1000}. where AR represents an accumulated repository of rules and subscripts denote the number of top individuals used to created the repository. Precision Threshold is 60%

Gen	Recall				Precision				Accuracy			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
10	1%	2%	2%	1%	1%	1%	2%	1%	97%	96%	96%	97%
20	0%	3%	3%	4%	0%	2%	2%	2%	96%	96%	96%	95%
30	12%	5%	8%	14%	2%	2%	3%	5%	94%	95%	93%	92%
40	11%	8%	11%	25%	4%	3%	4%	7%	90%	94%	92%	90%
50	11%	9%	17%	34%	6%	4%	5%	8%	93%	93%	91%	89%
60	7%	10%	20%	40%	6%	5%	6%	9%	94%	93%	91%	88%
70	17%	12%	25%	45%	6%	6%	7%	9%	87%	93%	90%	87%
80	10%	18%	32%	48%	4%	7%	8%	9%	93%	92%	89%	86%
90	6%	20%	34%	48%	7%	7%	8%	9%	94%	91%	88%	85%
100	14%	23%	36%	49%	2%	7%	8%	9%	88%	90%	88%	85%

4.4 Experimental results

This section shows the results of the experiments in 4.3. As Figure 2 shows the best individual of the population is not consistent because its recall and precision fluctuate from generation to generation. It may be because during the training procedure the best tree picks a limited number of rules that model the training data, however when the same tree is tested with another data set, it may or may not model the new data. In contrast RM steadily improved its recall and precision. It indicates that the best individual of the population does not necessarily include all the information produced by the evolutionary process, showing that the analysis of the information from a wider part of the population is able to contribute with the classification task. When the repository was created extracting rules from a bigger number of individuals the recall and precision increases. It shows that many useful information could be extracted from a wider part of the population. However, in all cases the accuracy decreases when the recall and precision increases. The reason for this is that the tree is classifying more positive cases. However, despite the fact that precision improved, the classifier produces more mistakes affecting the accuracy.

As can be seen from figure 2, the recall and precision reported by ARM (a),(c) outperformed the SRM results (b),(d). The difference between them is that ARM accumulates rules though generations and SRM does not. It means that ARM compiles more useful rules than SRM. It proves that some useful rules could be lost during the evolutionary process.

4.5 Rule distribution

In order to identify which individuals contribute to the repository rule. We counted the number of rules that every individual provided. This number indicates the number of rules that were included in the rule repository.

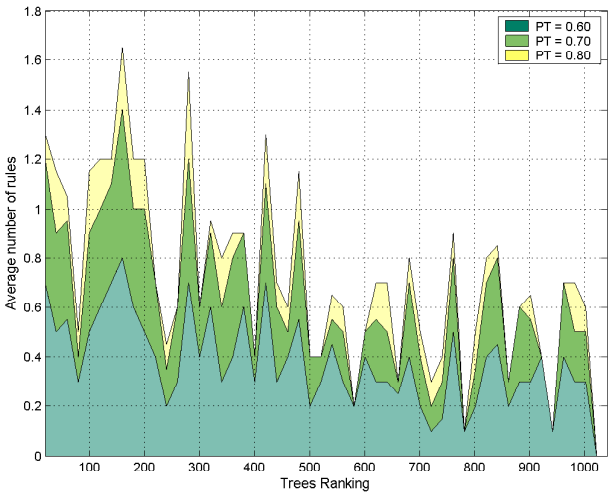


Fig. 3. Distribution of rules

Figure 3 shows the number of rules per individual in an accumulated repository of 1,000 individuals, during 100 generations. The X-axis refers every individual in the population, they are ordered according to their fitness function [8]. The best individual is in the first position, while the worst is in position 1,000, To smooth the graph, the number of rules was averaged on groups of twenty individuals. On the other hand Y-axis represents the average number of rules per individual. This uses different precision thresholds PT =60,70,80. As can be observed ARM picked rules from all sectors of the population. However, the number of rules tends to reduce slightly when the ranking decreases. The fact that the graph oscillates may be due to individuals that have similar fitness hold similar rules. In that case RM does not extract new rules because one condition to be included in the repository is the novelty of the rule. As one might expect the number of rules tends to decrease when PT increases because the method become stricter and fewer rules are selected.

5 Conclusions

Experiments show that RM is able to gather rules that together classify more positive cases. The analysis of a wider part of the population helps to gather more rules that improve the recall and precision. On the other hand the experiment showed that some useful rules can be lost in the evolutionary process. For that reason the accumulation of rules, through generations aids to classify positive cases. From experimental results we can conclude that the factors that benefit RM are the collection of rules from a wider part of the population as the accumulation of rules though generations.

[8] The fitness function is geometric mean of the product of precision and recall. This evaluation is suitable to deal with imbalanced classes [10]

Acknowledgement

The first author thanks to Consejo Nacional de Ciencia y Tecnologia (CONACyT) to support her studies at the University of Essex.

References

1. S.-H. Chen, Ed., *Genetic Algorithms and Genetic Programming in Computational Finance*. Kluwer Academic, 2002.
2. Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*. Springer, 2002.
3. J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts: The MIT Press, 1992.
4. E. P. Tsang and S. Martinez-Jaramillo, "Computational finance," in *IEEE Computational Intelligence Society Newsletter*, 2004, pp. 3–8.
5. E. P. Tsang, J. Li, and J. Butler, "Eddie beats the bookies," in *International Journal of Software, Practice and Experience*, ser. 10, vol. 28. Wiley, August 1998, pp. 1033–1043.
6. E. P. Tsang, P. Yung, and J. Li, "Eddie-automation, a decision support tool for financial forecasting," in *Journal of Decision Support Systems, Special Issue on Data Mining for Financial Decision Making*, ser. 4, vol. 37, 2004.
7. J. Li, *A genetic programming based tool for financial forecasting*. Colchester CO4 3SQ, UK: PhD Thesis, University of Essex, 2001.
8. A. Abe and Y. Ohsawa, "Special issue on chance disocvery," in *New Generation Computing*, ser. 1, vol. 21. Berlin: Springer and Tokyo: Ohmsha, November 2002, pp. 1–2.
9. Y. Ohsawa and P. McBurney, *Chance discovery*. Springer, 2003.
10. M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," in *Machine Learning*, vol. 30. 195-215, 1998.
11. A. L. Garcia-Almanza and E. P. Tsang, "Repository method technical report," http://privatewww.essex.ac.uk/˜algarc/documents/RepositoryMethod-TechicalReport.pdf.
12. E. P. Tsang, S. Markose, and H. Er, "Chance discovery in stock index option and future arbitrage," in *New Mathematics and Natural Computation, World Scientific*, ser. 3, vol. 1, 2005, pp. 435–447.
13. W. F. Sharpe, G. J. Alexander, and J. V. Bailey, *Investments*. Upper Saddle River, New Jersey 07458: Prentice-Hall International, Inc, 1995.
14. P. Angeline, "Genetic Programming and Emergent Intelligence," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. MIT Press, 1994, ch. 4, pp. 75–98.
15. P. Nordin, F. Francone, and W. Banzhaf, "Explicitly Defined Introns and Destructive Crossover in Genetic Programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, J. P. Rosca, Ed., Tahoe City, California, USA, 9 July 1995, pp. 6–22.
16. T. Soule and J. A. Foster, "Code size and depth flows in genetic programming," in *Proceeding of the Second Annual Conference*, J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. R. Riolo, Eds. Morgan Kaufmann, 1997, pp. 313–320.
17. W. B. Langdon, "Quadratic bloat in genetic programming," in *Proceedings of the Genetic and evolutionary Computation Conference (GECCO-2000)*, 2000, pp. 451–458.