Genetic Programming with Efficient Population Control for Financial Time Series Prediction

Neal Wagner
Computer Science Department
University of North Carolina at Charlotte
Charlotte, NC 28223 USA
nwagner@uncc.edu

Zbigniew Michalewicz
Computer Science Department
University of North Carolina at Charlotte
Charlotte, NC 28223 USA
zbyszek@uncc.edu

Abstract

Genetic Programming (GP) uses variable size representations as solutions. The size of individual solutions in a population can influence the efficiency of the search process. The research presented in this paper compares an alternative method for GP population control to the method commonly found in the literature for the task of financial time series prediction. It is shown that the alternative method gives comparable results with significantly more efficient use of computer resources.

1 INTRODUCTION

In Genetic Programming (GP), solutions are represented as trees of variable size and depth. The GP population control method commonly found in the literature is due to [Koza, 1992]. This method specifies a static population number and a maximum tree depth[1]. This method appears to be unsuitable for many real-world problems because it does not allow solutions to grow naturally past a certain point (tree depth), and, thus, solutions of greater complexity are never considered even though these solutions may be of higher quality. Additionally, the method, although restricting the depth of a solution, places no restriction on the size (number of nodes) of a solution. This allows the possibility for many large solutions in a population, which in turn causes a significant consumption of computer resources.

GP has been applied to the task of time series prediction by [Koza, 1992], [Oakley, 1994], [Iba *et al*, 1993], [Mulloy, Riolo, and Savit, 1996] and others. However, no studies have investigated alternative methods of GP population control for time series prediction. All investigations have used the method proposed by [Koza, 1992].

This paper presents a new method for GP population control and compares it to Koza's method on several financial time series. The rest of this paper is structured as follows. Section 2 explains the problem of time series prediction and reviews in more detail Koza's method for GP population control. Section 3 describes in detail the alternative method for GP population control and all other methods and parameters used in this study. Section 4 presents results, and section 5 summarizes the study and discusses possible future studies.

2 BACKGROUND

2.1 TIME SERIES PREDICTION

The problem of time series prediction is this: given values of the past, x, one must find a function, f, which predicts values of the future. Past values, $x(t - i)$, can be considered as a vector,

$$\boldsymbol{x}(t) = (x(t), x(t-1), \ldots, x(t-i)).$$

Future values, $x(t + j)$, are estimated by a function of previous values, $f(\boldsymbol{x}(t))$. In this paper we consider the problem of searching for $f(\boldsymbol{x}(t))$ given the vector

$$\boldsymbol{x}(t) = (x(t), x(t-1), \ldots, x(t-i)) \quad (for\ i=9),$$

which will predict the actual value $x(t + 1)$.

To evaluate how close a particular function is to predicting the actual value of a time series, one must impose a metric. A simple method, and the one used for this paper, is to calculate the squared error between the actual time series value and the predicted value,

$$SE = (x(t+1) - f(\boldsymbol{x}(t)))^2.$$

In GP a solution (for time series prediction this is a function) is evaluated for some number of test cases. The fitness of a solution can be determined by taking the mean of the squared errors for all test cases,

$$MSE = 1/N * \sum_t^T (x(t+1) - f(\boldsymbol{x}(t)))^2 ,$$

[1]The method used by [Koza, 1992] is described in more detail in section 2.2 of this paper.

where N is the number of test cases and t and T correspond to the time series data of the first and last test cases, respectively. Fitter individuals have lower *MSE* values (with the lowest possible *MSE* value being zero).

To measure the predictive performance of a solution, the following method is used. First, a vector of past time series values for the last (most recent) test case is applied to the solution (function) and the predicted future value, $x(t + 1)$, is calculated. Then this predicted value is appended to the past values to produce a new vector which is again applied to the solution to generate the next predicted future value, $x(t + 2)$. This process is continued until the desired number of future values to predict is generated. Finally, the mean of the squared errors between the actual time series values and the predicted values is calculated. For this paper the number of future values to predict is 5, that is this process is used to generate future values $x(t + 1), \ldots, x(t + 5)$.

2.2 KOZA'S METHOD FOR GP POPULATION CONTROL

Koza's [Koza, 1992] method for GP population control (KPC) uses a static population number, usually 500, 1000, or 2000 depending on the complexity of the problem, and a maximum tree depth of 17 for individual solutions in the population. This limit for tree depth prohibits the search process from exploring solutions of greater complexity, which, for many real-world problems, may be solutions of higher quality. This limit does not, however, necessarily limit the size (number of nodes) of solutions. Therefore, if numerous solutions in a population have full or nearly full trees of depth close to the maximum, available resources may be strained or exhausted.

3 METHODS

3.1 AN ALTERNATIVE METHOD FOR GP POPULATION CONTROL

An alternative method for GP population control (APC) is proposed that is based on a non-static population number with a limit for the total number of nodes present in a population and no limit for solution tree depth. This method addresses the following issues: 1) allowing natural growth of complex solutions of greater quality and 2) keeping resource consumption within some specified limit. By not limiting the tree depth of individual solutions, natural evolution of complex solutions is permitted. By restricting the total number of nodes in a population, available resources are conserved. The method is described in more detail below.

The APC based on limiting the total nodes present in a population works in the following way. Two node limits for a population are specified (as parameters): 1) the soft node limit and 2) the hard node limit. The soft node limit is defined as the limit for adding new individuals to a population. This means that if adding a new individual to a population causes the total nodes present to exceed the soft node limit, then that individual may be added but no more individuals may be added afterwards. The hard node limit is defined as the absolute limit for total nodes in a population. This means that if adding a new individual to a population causes the total nodes present to exceed the hard node limit, then that individual may be added only after it is repaired (nodes have been trimmed) so that the total nodes present does not exceed the limit. During the selection process of the genetic program, a count of the total nodes present in a population is maintained. Before adding a new individual to a population, a check is made to determine if adding the individual will increase the total nodes present beyond either of the specified limits.

3.2 GP PARAMETERS

As explained previously, the task of time series prediction is accomplished by using previous values of a time series to predict future values. Therefore, the GP terminal set will consist of previous values of the time series. The terminal set[2] used for this paper employs the ten previous values and a random constant,

$$\mathbf{T} = \{ x(t), x(t-1), \ldots, x(t-9), \Re \}.$$

The random constant ranges from [-1000, 1000]. The requirement of the function set is the ability to produce nonlinear expressions. Thus, the function set[3] used is

$$\mathbf{F} = \{ +, -, *, \% \},$$

where % represents protected division[4]. Fitness is determined by comparing the values returned by a GP solution tree to each value of 80 test cases (i.e., 80 time series data points). The objective function to be minimized is the mean of the squared errors between the actual time series value and the GP predicted value for all test cases,

$$MSE = 1/N * \sum_t^T (x(t+1) - f(\mathbf{x}(t)))^2 ,$$

where the variables of the above equation are as described in section 2.1.

The GP parameters used that are not related to either of the population control methods already described are shown in Table 1. These parameters are the same parameters that [Koza, 1992] uses except where noted. The parameters used that are related to the population control methods already described are shown in Table 2. As in Koza's experiments, the values chosen for these parameters are not intended to be optimal, only sufficient for comparison of the two population control methods

---

[2]This terminal set may not be optimal. Choosing the optimal set is an area that deserves further research, but it is not the purpose of this paper.

[3]This function set may not be optimal. See footnote [2] for further explanation.

[4]Protected division returns 1.0 if the denominator is 0 and returns the result of division otherwise [Koza, 1992].

previously described. The following is a detailed discussion of the parameters related to the population control methods.

Table 1: GP parameters that are not related to either of the population control methods being compared.

Maximum number of generations to be run = 51.
Probability of crossover = 90%.
Probability of reproduction = 10%.
Probability of choosing internal points for crossover = 90%.
Maximum depth for initial random solution trees = 6.
Probability of mutation = 0%.
Generative method for initial random population is ramped half-and-half.
Basic selection method is fitness proportionate.
Spousal selection method is fitness proportionate.
Elitist strategy[5] is used.

Table 2: GP parameters that are related to the population control methods being compared.

KPC (static population with maximum solution tree depth).	APC (limiting total nodes present in a population).
Population size[6] = 1000.	Soft node limit = 25,000.
Maximum depth for solution trees created during the run = 17.	Hard node limit = 35,000.
	Population size is unspecified (non-static).
	No maximum depth for solution trees created during the run.

The parameter values chosen for KPC are the same values that [Koza, 1992] uses. The parameter values of APC were chosen in the following way. 20 pre-experiment runs were made for each time series to be used in the actual experiment. These pre-experiment runs were made with parameter values as shown in column 1 of Table 2 (KPC). During each run, counts of the total number of nodes present in the initial and final populations were kept. It was seen that the average total number of nodes present in the initial and final population of all runs was ~12,500 and ~140,000 respectively. Therefore, the soft node limit parameter was chosen as approximately twice the average for the initial population and the hard node limit parameter was chosen as only 10,000 nodes greater than the soft node limit (significantly less than the average for the final population). The purpose of the pre-experiment runs was to discover fair values for the soft and hard node limits. This task can be hard to accomplish in any precise manner because KPC does not control the total number of nodes in a population at any given generation. Thus, the relatively small values chosen for these limits (less than half the average for total nodes present in the final population of all pre-experiment runs) are an attempt to err on the side favoring KPC.

[5][Koza, 1992] does not use elitism. [Mulloy, Riolo, and Savit, 1996] report that GP generally lost its best trees the generation after they were found. Thus, elitism is used to remedy this problem.

[6][Koza, 1992] uses population size of 1000 or 2000 for problems of greater complexity. The problem of predicting real-world time series seems to be complex enough to warrant such a population size.

4 RESULTS

The results shown are based on 20 runs for each experimental condition. Each of the 20 runs has a different initial population. Table 3 shows the 10 financial time series chosen for the prediction task using GP. A total of 95 data points are used for each time series with starting and ending dates of July 31, 2000 and December 12, 2000 respectively. The first 90 data points are used as training (80 test cases with 10 past values for the initial test case) and the final 5 data points for testing.

Table 3: The 10 financial time series chosen for the prediction task.

Delta Airlines, Inc. (DAL)
US Airways Group, Inc. (U)
Ingles Markets, Inc. (IMKTA)
Win-Dixie Stores, Inc. (WIN)
Chevron Corporation (CHV)
Microsoft Corporation (MSFT)
Sun Microsystems, Inc. (SUNW)
Dow Jones Industrial Average (DJI^)
S&P 500 Index (SPC^)
PG&E Corporation (PCG)

Table 4 shows the following results for each time series and population control method:

1) The average mean squared test error (AMS),
2) The mean squared test error of the best run (BMS),
3) The total number of aberrant runs[7] (#AR).

[7]The meaning of an aberrant run is discussed later in this section.

Table 5 shows the following results for each time series and population control method:

1) The average total number of nodes present in the final population (A#N),
2) The total number of trivial runs[8] (#TR).

Table 4: Results from KPC and APC runs.

Time Series / Method	KPC			APC		
	AMS	BMS	#AR	AMS	BMS	#AR
DAL	1.32	1.16	2	1.62	1.16	2
U	11.36	8.58	0	12.64	10.93	0
IMKTA	0.09	0.07	0	0.14	0.07	0
WIN	1.54	1.20	0	1.38	0.89	0
CHV	0.65	0.63	1	0.76	0.61	1
MSFT	17.22	12.82	0	16.29	8.23	0
SUNW	64.83	46.44	0	60.60	31.92	0
DJI^	43007	13545	2	45742	36118	0
SPC^	391.5	233.8	0	329.8	240.3	2
PCG	9.81	7.57	0	10.32	7.50	0

The following is an explanation of the meaning of trivial and aberrant runs. Some runs produced trivial results. A trivial run means that the best solution tree of the final generation is a one-node tree consisting of the terminal $x(t)$ (the most recent past value). A few runs produced aberrant results. An aberrant run means that the mean squared test error for that run differs from that of other runs with the same experimental conditions by an order of magnitude. When calculating the values for AMS and BMS of Table 4 and for A#N of Table 5, trivial and aberrant runs were disregarded.

As shown in Table 4, the average and best mean squared test errors for KPC and APC are comparable for each time series tested. Also shown in Table 4, the number of aberrant runs produced by KPC and APC are the same (each produced 5 aberrant runs out of a total of 200 runs). However, as shown in Table 5, KPC and APC give significantly different values for the average total number of nodes present in the final population and the number of trivial runs. The average total number of nodes present in the final population for APC is near 25,000 (the specified soft node limit) for each time series while that of KPC ranges from 40,567 to 224,629. In fact, some KPC runs exhausted computer memory and aborted before completing the maximum number of generations. KPC also produced 91 trivial runs while APC produced only 13

Table 5: Results from KPC and APC runs.

Time Series / Method	KPC		APC	
	A#N	#TR	A#N	#TR
DAL	153,783	10	25,397	0
U	212,444	14	25,349	2
IMKTA	59,225	13	25,275	7
WIN	124,662	9	25,320	0
CHV	40,567	16	25,190	1
MSFT	164,391	8	25,208	0
SUNW	161,312	10	25,238	2
DJI^	224,629	3	25,264	0
SPC^	186,654	2	25,225	0
PCG	61,287	6	25,214	1

(out of 200). The reason for this difference in the number of trivial runs may be because KPC does not allow solution trees to grow naturally past depth 17, and, thus, does not explore more complex solutions that may be better than the trivial solution. A run that ends with a trivial solution may be considered as a waste of computer resources because those resources may have been put to use to find a more complex solution of higher quality.

5 CONCLUSIONS AND FUTURE WORK

The following is a summary of the investigation. Two population control methods, KPC, which uses a static population number and a maximum solution tree depth, and APC, which limits the total number of nodes present in a population, were compared for the task of time series prediction of several financial time series using GP. It is shown that both methods give comparable results, but APC is significantly more efficient in its use of computer resources. This more efficient use of computer resources manifests itself in two ways: 1) using less computer memory and 2) producing fewer trivial runs.

[Mulloy, Riolo, and Savit, 1996] report that a new crossover operator that forbids one-node tree crossovers (FONTX) successfully avoids premature convergence to trivial solution trees. One possible direction for future work is to combine this crossover operator with the alternative population control method investigated in this paper. Another direction may be to use a more sophisticated fitness measure that guides the search process to solution trees that are less complex but sufficiently accurate such as the one proposed by [Zhang and Mühlenbein, 1995].

[8]The meaning of a trivial run is discussed later in this section.

References

[Bäck, 1998] Bäck, Th. 1998. On the behavior of evolutionary algorithms in dynamic environments. *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 446-451, IEEE Press.

[Bäck, Fogel, and Michalewicz, 1997] Bäck, Th., Fogel, D., and Michalewicz Z. 1997. *Handbook of Evolutionary Computation*. Institute of Physics Publishing Ltd, Bristol and Oxford University Press.

[Casdagli, 1989] Casdagli, Martin. 1989. Nonlinear Prediction of Chaotic Time Series. *Physica D*, **35**:335-356.

[Eiben, Hinterding, and Michalewicz, 1999] Eiben, A.E., Hinterding, R., and Michalewicz, Z. 1999. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, **3**(2):124-141.

[Goldberg, Deb, and Clark, 1992] Goldberg, D.E., Deb, K., and Clark, J.H. 1992. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, **6**:333-362.

[Hinterding, Michalewicz, and Eiben, 1997] Hinterding, R., Michalewicz, Z., and Eiben, A. E. 1997. Adaptation in evolutionary computation: A survey. *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 65-69, IEEE Press.

[Iba et al, 1993] Iba, Hitoshi, Takio Kuita, Hugo de Garis and Taisuke Sato. 1993. System Identification using Structured Genetic Algorithms. In Forest, Stephanie. (editor). *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 279-286, Morgan Kaufmann.

[Kaboudan, 1999] Kaboudan, M. A. 1999. A Measure of Time Series' Predictability Using Genetic Programming Applied to Stock Returns. *Journal of Forecasting*, **18**:345-357.

[Koza, 1992] Koza, John. 1992. *Genetic Programming*. The MIT Press.

[Michalewicz, 1996] Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd edition, Springer-Verlag.

[Mulloy, Riolo, and Savit, 1996] Mulloy, Brian S., Riolo, Rick L., Savit, Robert S. 1996. Dynamics of Genetic Programming and Chaotic Time Series Prediction. *Genetic Programming: Proceedings of the First Annual Conference*, pages 166-174, The MIT Press.

[Oakley, 1994] Oakley, Howard. 1994. Two Scientific Applications of Genetic Programming: Stack Filters and Non-Linear Equation Fitting to Chaotic Data. In Kinnear, Kim (editor). *Advances in Genetic Programming*, pages 369-389, The MIT Press.

[Zhang and Mühlenbein, 1995] Zhang, B. and Mühlenbein, H. 1995. Balancing Accuracy and Parsimony in Genetic Programming. *Evolutionary Computation*, **3**(1):17-38.