# Parameter Adaptation for GP Forecasting Applications

Neal Wagner[1] and Zbigniew Michalewicz[2]

[1] Department of Mathematics and Computer Science, Augusta State University, Augusta, GA 30904, USA nwagner@aug.edu
[2] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia, Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, and Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland zbyszek@cs.adelaide.edu.au

**Summary.** Genetic Programming (GP) has been applied to time series forecasting often with favorable results. However, for forecasting tasks several open issues concerning parameter settings exist. Many real-world forecasting tasks are dynamic in nature and, thus, static parameter settings may lead to inferior performance. This paper presents the results of recent studies investigating non-static parameter settings that are controlled by feedback from the GP search process. Specifically, non-static settings for population size and training data size are explored.

Applications based on Evolutionary Algorithms (EA) rely on numerous parameters (e.g., population size, mutation and recombination rates, etc.) to guide and control algorithm execution. The values of these parameters have a significant effect on the performance of an EA. A common practice is to tune the parameters, that is to make several pre-experiment runs with various parameter settings in order to find good values. These values are then used for the actual experiment and typically remain fixed for the duration of the experiment. Manually tuning parameters is problematic because parameters are not necessarily independent of each other and, thus, trying all possible combinations requires an enormous amount of experiments.

For many applications, especially those used in dynamic environments, optimal parameter settings may vary over the course of a run. For such applications the use of static parameters can lead to inferior performance. *Adaptive* parameter control refers to the use of feedback from the evolutionary search process to adjust (adapt) parameter values during a run [34].

EA and other biologically inspired methods (e.g., Neural Networks) have often been applied to forecasting tasks [4, 7, 8, 11, 13, 17, 20, 27, 37, 38, 40]. Genetic Programming (GP) is a sub-category of EA that is also commonly used for forecasting [1, 5, 6, 14, 15, 16, 19, 21, 22, 23, 24, 25, 32, 35, 36, 41].

While GP forecasting applications often yield good results, a number of open issues concerning parameter settings exist. Kaboudan [26] recognizes some of these issues including: 1) how to determine the training data size (i.e., the number of data to be utilized for training) and 2) how to determine the population size. A few GP forecasting studies have examined good (static) values for training data and population sizes empirically (for example, [10, 12]), but there is still no agreement in the literature on what may be optimal. This paper presents the results of two recent studies which investigate adaptive parameter control for training data size and population size, respectively. Additionally, we will discuss the another aspect of the GP search process for forecasting, that of finding an optimal evaluation (fitness) function.

The rest of this paper is organized as follows: section 1 gives a brief description of the application of GP to forecasting tasks, sections 2 and 3 detail algorithms for adaptively controlling the training data and population sizes, respectively, section 4 discusses the search for an optimal fitness function, and section 5 concludes.

## 1 The Use of GP for Forecasting

In order to have a clear picture of how GP is used for forecasting, it is necessary to understand the different kinds of forecasting problems that exist. In general, there are three types:

1. classification problems,
2. regression problems, and
3. time series problems.

In classification problems the goal is to predict the *class* or *category* of a newly occurring event by examing old events and their corresponding classes. For example, credit card companies need to classify newly occurring credit card transactions into one of two classes, fraudulent or non-fraudulent. For this type of problem, the issue of time is of secondary importance and either does not play a part in the classification process or is incorporated as one of many features of the process.

Regression problems try to learn about the relationship between several independent (predictor) variables and a dependent (criterion) variable. For regression, the desired output is a number. For example, business analysts may wish to predict a manager's salary given knowledge of his or her position, number of supervised employees, number of years at the position, education level, etc. As for classification problems, the issue of time is either non-existant or is included as a feature of the problem.

For time series problems the dominant feature is time and the data to be analyzed presents itself in the form of numerous observations/measurements given in sequential order of time. The goal of this type of problem is to predict a future (numeric) value of some variable by examing past values of that

variable and/or past values of other related variables. For example, investment firms try to predict the future value of a particular stock based on past values of that stock and past values of other related variables such as trading volume, interest rates, etc.

GP has been used for all three of the above-listed forecasting problems, however we will focus on time series problems as GP applications for this type of problem are prevalent.

In GP solutions are represented as tree structures. Internal nodes of solution trees represent appropriate operators and leaf nodes represent input variables or constants. For time series forecasting applications, the operators are mathematical functions and the inputs are lagged time series values and/or explanatory variables. Each solution tree represents a candidate forecasting model. Figure 1 gives an example solution tree for time series forecasting. Variables $x_{t1}$ and $x_{t2}$ represent time series values one and two periods in the past, respectively. Crossover is performed by exchanging subtrees from two

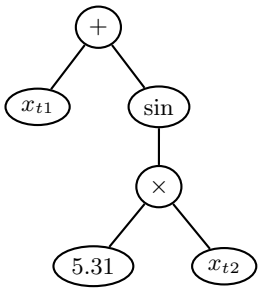


**Fig. 1.** GP representation of forecasting solution $x_{t1} + \sin(5.31x_{t2})$

parent solution trees. Mutation is performed by selecting a subtree of a single solution tree and replacing it with a randomly-constructed subtree.

GP starts by creating an initial population of candidate forecasting models. A candidate model is produced by (randomly) constructing a tree made up of operators and inputs. Each model is ranked based on its prediction error over a set of training data and new populations are generated by selecting fitter models and applying the crossover and mutation operations. New populations are created until the fittest model has a sufficiently small prediction error, repeated generations produce no reduction of error, or some limit for maximum generations has been reached. GP was developed by Koza [29] as a problem-solving tool with applications in many areas. He was the first to use GP to search for model specifications that can replicate patterns of observed time series.[3]

[3] In [29] Koza refers to this as "symbolic regression."

## 2 Adapting the Training Data Size

The primary objective of time series forecasting is to find a model that accurately represents the underlying data generating process and use this model to forecast the future. In order to accomplish this, GP requires a set of training data that sufficiently represents the data generating process in question. If the training data size is too small, forecasting models evolved by GP are unlikely to accurately represent the data generating process and, thus, will have poor forecasting performance. If the training data size is too large, GP models may suffer from "over-fitting" [9]. Over-fitting refers to forecasting models that fit both the underlying data generating process and extent noise in the time series. The problem of setting the training data size is compounded when a time series is produced by a non-constant process (i.e., one that varies over time). In such a case different segments of a time series may have different underlying processes. If data from two different segments is specified for GP training, forecasting models evolved may be skewed. Consider the subset of time series data shown in figure 2. Suppose this represents recent histori-

$$\underbrace{22, 33, 30, 27, 24, 20, 21, 20, 20,}_{\text{segment1}}\ \underbrace{23, 26, 29, 30, 28, 29, 32, 30, 31}_{\text{segment2}}\ |\ \underbrace{\ldots}_{\text{future}}$$

**Fig. 2.** Time series containing segments with differing underlying processes.

cal data and has been chosen as training data for GP. Suppose further that the subset consists of two segments each with a different underlying process. The second segment's underlying process represents the current process and is valid for forecasting future data. The first segment's process represents an older environment that no longer exists. Because both segments are analyzed, evolved forecasting models will be distorted.

A recent study by Wagner and Michalewicz [42] investigates a novel algorithm for adapting the training data size by using "windows" of training data that slide with time and expand or contract based on feedback from the GP search. In this GP forecasting application (called "Dynamic Forecasting GP" or "DyFor GP"), training starts at the beginning of the available historical time series data. Some initial windowsize (training data size) is set and several generations are run (in the manner described in the previous section) to evolve a population of solutions. Then the data window slides to include the next time series observation. Several generations are run with the new data window and then the window slides again. This process is repeated until all available data have been analyzed up to and including the most recent historical data. Figure 3 illustrates this process. In the figure, | marks the end of available historical data. The set of several generations run on a single data window is referred to as a "dynamic generation." Thus, a single run of

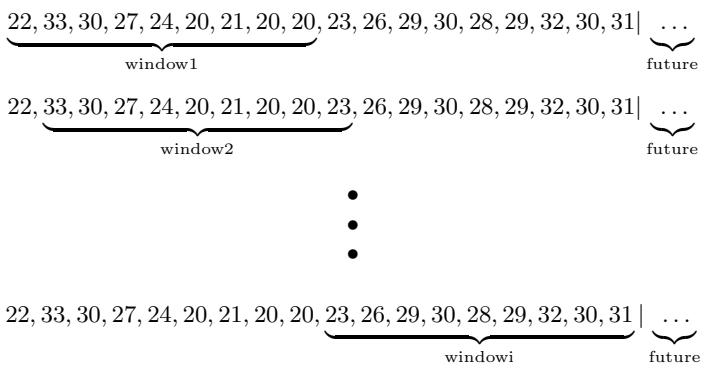


**Fig. 3.** A sliding data window.

DyFor GP includes several dynamic generations (one for each window slide) on several different consecutive data windows.

DyFor GP adjusts the data windowsize using feedback from the GP search process. This is accomplished in the following way.

1. Select two initial windowsizes, one of size $n$ and one of size $n + i$ where $n$ and $i$ are positive integers.
2. Run a dynamic generation at the beginning of the historical data with windowsize $n$.
3. At the end of the dynamic generation, use the best evolved solution (forecasting model) to predict a number of future data and then measure the prediction's accuracy.
4. Run another dynamic generation also at the beginning of the historical data with windowsize $n + i$.
5. At the end of the dynamic generation, predict future data and measure the prediction's accuracy. Note which windowsize generated the better prediction.
6. Select another two windowsizes based on which windowsize had better accuracy. For example if the smaller of the 2 windowsizes (size $n$) predicted more accurately, then choose 2 new windowsizes, one of size $n$ and one of size $n - i$. If the larger of the 2 windowsizes (size $n + i$) predicted more accurately, then choose windowsizes $n + i$ and $n + 2i$.
7. Slide the data windows to include the next time series observation. Use the two selected windowsizes to run another two dynamic generations, predict future data, and measure their prediction accuracy.
8. Repeat the previous two steps until the the data windows reach the end of historical data.

Thus, predictive accuracy is used to determine the direction in which to adjust the windowsize toward the optimal. Successive window slides bring the windowsize closer and closer to this destination.

Consider the following example. Suppose the time series given in figure 2 is to be analyzed and forecasted. As depicted in the figure, this time series consists of two segments each with a different underlying data generating process. The second segment's underlying process represents the current environment and is valid for forecasting future data. The first segment's process represents an older environment that no longer exists. If there is no knowledge available concerning these segments, automatic techniques are required to discover the correct windowsize needed to forecast the current setting. DyFor GP starts by selecting two initial windowsizes, one larger than the other. Then, two separate dynamic generations are run at the beginning of the historical data, each with its own windowsize. After each dynamic generation, the best solution is used to predict some number of future data and the accuracy of this prediction is measured. Figure 4 illustrates these steps. In the figure **win1** and **win2** represent data windows of size 3 and 4, respectively, and **pred** represents the future data predicted.

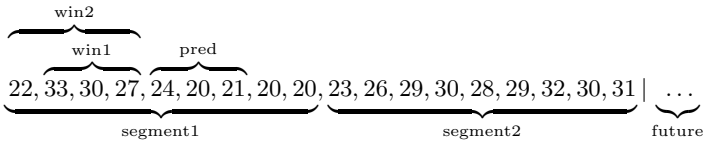


**Fig. 4.** Initial steps of window adaptation.

The data predicted in these initial steps lies inside the first segment's process and, because the dynamic generation involving data window **win2** makes use of a greater number of appropriate data than that of **win1**, it is likely that **win2**'s prediction accuracy is better. If this is true, two new windowsizes for **win1** and **win2** are selected with sizes of 4 and 5, respectively. The data windows then slide to include the next time series value, two new dynamic generations are run, and the best solutions for each used to predict future data. Figure 5 depicts these steps. In the figure data windows **win1** and **win2** now include the next time series value, 24, and **pred** has shifted one value to the right.

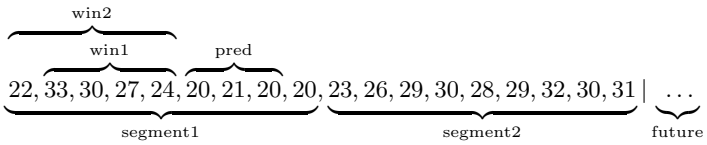


**Fig. 5.** Window adaptation after the first window slide. Note: **win1** and **win2** have size 4 and 5, respectively.

This process of selecting two new windowsizes, sliding the data windows, running two new dynamic generations, and predicting future data is repeated until the data windows reach the end of historical data. It may be noted that while the prediction data, **pred**, lies entirely inside the first segment, the data windows, **win1** and **win2**, expand to encompass a greater number of appropriate data. However, after several window slides, when the data windows span data from both the first and second segments, it is likely that the window adjustment reverses direction. Figures 6 and 7 show this phenomenon. In figure

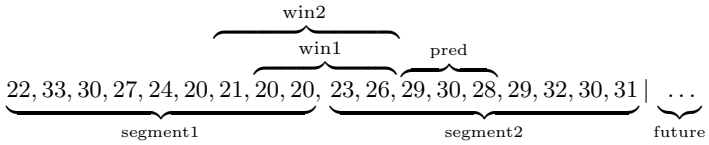


**Fig. 6.** Window adaptation when analysis spans both segments. Note: the smaller window, **win1**, is likely to have better prediction accuracy because it includes less erroneous data.

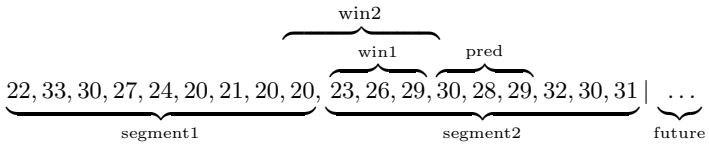


**Fig. 7.** Window adaptation when analysis spans both segments. Note: **win1** and **win2** have contracted to include less erroneous data.

6 **win1** and **win2** have sizes of 4 and 5, respectively. As the prediction data, **pred**, lies inside the second segment, it is likely that the dynamic generation involving data window **win1** has better prediction accuracy than that involving **win2** because **win1** includes less erroneous data. If this is so, the two new windowsizes selected for **win1** and **win2** are sizes 3 and 4, respectively. Thus, as the data windows slide to incorporate the next time series value, they also contract to include a smaller number of spurious data. In figure 7 this contraction is shown.

After the data windows slide past the end of the first segment, they expand again to encompass a greater number of appropriate data. Figures 8 and 9 depict this expansion.

As illustrated in the above example, DyFor GP uses predictive accuracy to adapt the size of its training data automatically towards the optimal. Automatic determination of an appropriate training window is beneficial for forecasting concerns in which the number of recent historical data whose underlying data generating process corresponds to the current environment is not

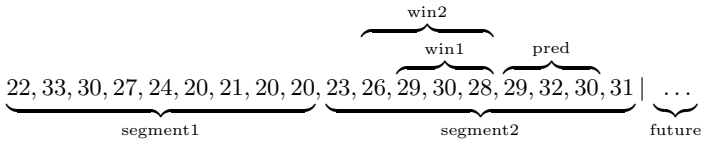


**Fig. 8.** Window adaptation when analysis lies entirely inside the second segment. Note: the larger data window, **win2**, is likely to have better prediction accuracy because it includes a greater number of appropriate data.

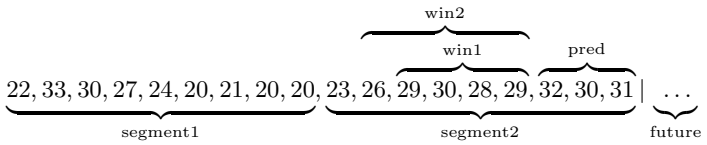


**Fig. 9.** Window adaptation when analysis lies entirely inside the second segment. Note: **win1** and **win2** have expanded to include a greater number of appropriate data.

known. Furthermore, this adaptation takes place *dynamically*. This means that as DyFor GP moves through (analyzes) historical time series data, the training window expands or contracts depending on the environment encountered. In the study conducted by Wagner and Michalewicz [42], DyFor GP with adaptive training data size is compared to conventional GP (with static training data size) for forecasting performance. Experimental results show that DyFor GP yields the more efficient forecasts. The following section describes how population size can be controlled for GP forecasting applications.

## 3 Adapting the Population Size

Bloat in GP is the tendency for solution trees to grow large as they approach the optimal [30, 31]. Solutions may become so large that they exhaust computer resources. Additionally, bloat hinders a GP model's adaptability as solutions become too specialized to adjust to changing conditions. Bloat is a problem for any GP model regardless of the application [3]. GP evolves a population of solutions for a given problem and, thus, must contain some method to control the number and size of solutions in the population. The standard method of GP population control is due to Koza [29] and uses a static population cardinality and a maximum tree depth for solutions.[4] However, this method does not protect a GP model from bloat. If numerous solutions in

[4] Koza [29] used population sizes of 500, 1000, and 2000 and maximum solution tree depth of 17 in his early GP experiments.

a population have full or nearly full trees of depth close to the maximum, available resources may be exhausted. Additionally, the artificial limit for tree depth prohibits the search process from exploring solutions of greater complexity, which, especially for many real-world problems, may be solutions of higher quality.

An alternative method for GP population control is presented Wagner and Michalewicz [41] that allows natural growth of complex solutions in a setting that more closely emulates the one found in nature. In nature the number of organisms in a population is not static. From generation to generation, the population cardinality changes depending on the quality and type of individual organisms present. The proposed natural non-static population control (NNPC) is based on a variable population cardinality with a limit on the total number of tree 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,
2. keeping resource consumption within some specified limit, and
3. allowing the population cardinality to vary naturally based on the make-up of individual solutions present.

By not limiting the tree depth of individual solutions, natural evolution of complex solutions is permitted. By restricting the total number of tree nodes in a population, available resources are conserved. Thus, for a GP model that employs NNPC, the number of solutions in a population grows or declines naturally as the individual solutions in the population vary. This method is described in more detail below.

NNPC works in the following way. Two node limits for a population are specified as parameters: the soft node limit and the hard node limit. The soft node limit is defined as the limit for adding new solutions to a population. This means that if adding a new solution to a population causes the total nodes present to exceed the soft node limit, then that solution is the last one added. The hard node limit is defined as the absolute limit for total nodes in a population. This means that if adding a new solution to a population causes the total nodes present to exceed the hard node limit, then that solution may be added only after it is repaired (the tree has been trimmed) so that the total nodes present no longer exceeds this limit. During the selection process, a count of the total nodes present in a population is maintained. Before adding a new solution to a population, a check is made to determine if adding the solution will increase the total nodes present beyond either of the specified limits.

Wagner and Michalewicz [41] compare a GP forecasting model with NNPC to one with the standard population control (SPC) method introduced by Koza [29]. Observed results indicate that the model with NNPC was significantly more efficient in its consumption of computer resources than the model with SPC while the quality of forecasts produced by both models remained

equivalent. The following section discusses the search for an optimal fitness function for GP forecasting applications.

## 4 The Search for an Optimal Fitness Function

Evolution-based techniques such as GP require that some fitness function be used to measure the quality of candidate solutions. However, it may not be clear how to select such a measure for a particular problem. It may be that a single measure performs well under certain conditions but badly in others. Most GP forecasting applications use a mean squared error (MSE) fitness function for model evolution. To date, there have been no significant studies investigating alternative fitness functions for GP forecasting applications. One alternative measure that might be considered is the mean absolute deviation (MAD). Comparing the MSE and MAD measures, it can be seen that the error value of MSE grows quicker than that of MAD when outlier data are present. Thus, outliers tend to influence analyses based on MSE more than they do analyses based on MAD. An outlier datum can represent one of two possibilities: noise (which should be ignored or have reduced impact on model construction) or new information representing a shift in the underlying process. For series in which outlier data represent noise, MAD might be the more effective measure. For series in which outlier data represent a process shift, MSE might be preferable. The question of which measure to employ would depend upon the characteristics of the time series to be forecast.

A novel fitness function is presented by Wagner and Michalewicz [42] that incorporates aspects of both the MSE and MAD measures. The idea is to use MSE when data encountered is not considered an outlier and MAD when data encountered is considered an outlier. This new measure (called "CF" for "combined fitness") requires a user-specified parameter, $\Omega$, to determine which data are outliers and which are not. Figure 10 gives a graphical depiction of the CF measure as a function of the relative error. From the figure, when the relative error is within the threshold given by $\Omega$, CF measure values follow those of the squared error. However, when the relative error falls outside of the $\Omega$ threshold, CF measure values follow those of the absolute deviation.

A number of new GP forecasting experiments were conducted on real data to compare the MSE, MAD, and CF fitness functions. Here, we forecast the U.S. Gross Domestic Product (GDP) using several relevant economic variables. The following three sections describe the GDP time series, experimental setup, and observed results, respectively.

### 4.1 Forecasting the U.S. Gross Domestic Product

According to the U.S. Department of Commerce [39], the GDP is defined as "the market value of goods and services produced by labor and property in the United States." The GDP is a metric frequently employed as a measure of the

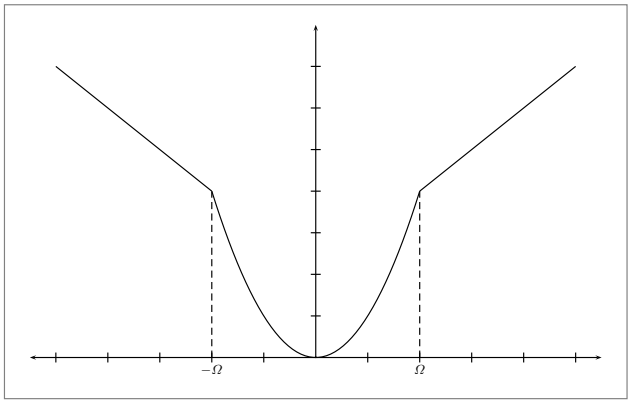


**Fig. 10.** The CF measure as a function of the relative error.

nation's economy. The GDP series was selected because it is a widely-studied, non-linear time series with a well-known set of explanatory variables. Figure 11 gives a graphical depiction of the quarterly GDP (growth) time series. In the figure real GDP growth is calculated as a quarter-over-quarter annualized percent change. A contemporary study conducted by Kitchen and Monaco

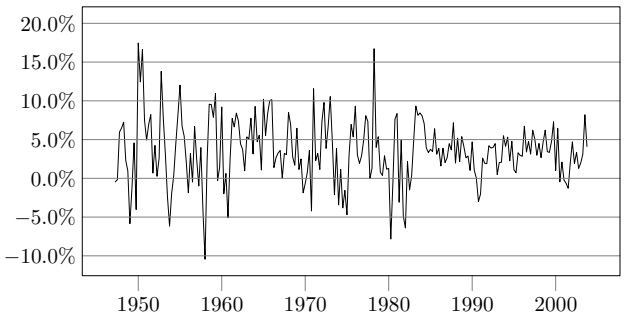


**Fig. 11.** Gross Domestic Product (growth): 1947-2003.

[28] forecasts the GDP, a time series with quarterly frequency, using multiple economic indicators that are measured monthly. Thirty indicators are utilized in all and can be subdivided into the following categories: employment (6), financial (4), survey (6), production and sales (12), and other (2). The results of their study show that forecasting models constructed using these indicators provided efficient forecasting performance for the period of 1995Q1 through 2003Q1.

### 4.2 Experimental Setup

For this set of GP forecasting experiments, three fitness functions are compared: MSE, MAD, and the CF measure described in section 4. Recall that the CF measure requires that a parameter, $\Omega$, be specified representing the threshold between outlier and non-outlier data. For these experiments three different settings of $\Omega$ are tested: one that is 5.0% of the median level of the time series, one that is 7.5% of the median level, and one that is 10.0%.

29 of the 30 economic indicators listed in the Kitchen and Monaco's GDP forecasting study [28] are utilized as inputs[5] by all competing models and the outputs are one-step-ahead, quarterly forecasts for the current quarter when only one month of historical data for that quarter is available. Historical GDP data dating back to 1965Q1 is used for training the GP and one-step-ahead forecasts for 1995Q1 through 2003Q1 are produced. Table 1 gives the GP parameter values used by all competing models.

**Table 1.** GP parameter settings.

| Parameter | Value |
|---|---|
| crossover rate | 0.9 |
| reproduction rate | 0.0 |
| mutation rate | 0.1 |
| population size | 1000 |
| max. no. of generations | 51 |
| elitism used? | yes |
| fitness function | MSE/MAD/CF |

All parameter values listed in table 1 were selected to match those used by Koza [29] with the following exceptions.

1. Elitism (reproduction of the best solution of the population) is used.
2. Parameter values for "reproduction rate" and "mutation rate" were exchanged. This was done for two reasons: 1) increasing the mutation rate allows for greater search-space exploration [33] and 2) decreasing the reproduction rate to zero was not thought to harm the effectiveness of the evolutionary process since elitism is used.

Because the GP search process is a stochastic one, a set of GP runs is executed rather than just a single run. Setsize = 20 is used for all GP forecasting experiments executed here.

[5] One of the indicators, "Business Week Production Index," could not be obtained at the time of the experiments.

### 4.3 Results

As mentioned in the previous section, a set of runs is executed (setsize = 20) for each competing model. For a single run, forecasting performance is measured by calculating the MSE of all forecasts. For a set of runs, forecasting performance is measured by calculating the mean and standard deviation of MSE values over all (20) runs. Table 2 gives the observed results for the GDP experiments. In the table, the first column shows the fitness function used to drive the GP process. Note that three separate experiments employing the CF measure are shown, one for each of three $\Omega$ settings (5.0%, 7.5%, and 10.0%, respectively).

**Table 2.** GDP forecasting results.

| GP fitness function | mean MSE | std. dev. |
|---|---|---|
| MSE | 4.96 | 1.62 |
| MAD | 4.46 | 1.42 |
| CF($\Omega = 5.0\%$) | 3.99 | 1.19 |
| CF($\Omega = 7.5\%$) | 3.69 | 0.70 |
| CF($\Omega = 10.0\%$) | 3.96 | 1.19 |

Table 2 reveals some interesting results. All three CF fitness functions outperform both the MSE and MAD fitness functions with the one utilizing a 7.5% setting for $\Omega$ giving the best performance overall. Between the MSE and MAD fitness functions, MAD yields the better forecasts.

The MAD fitness function's better performance when compared to MSE may mean that for the GDP series outlier data more frequently represents noise rather than a shift in the underlying data generating process. The fact that the CF fitness function outperforms both MSE and MAD may mean that it is better able to distinguish noise from true process shifts and, thus, can lessen the effect of noisy data and still respond to changes in the process.

The following section draws conclusions and discusses potential areas for future work.

## 5 Conclusions and Future Work

In this paper, parameter adaptation for GP forecasting applications is discussed. Novel algorithms from recent studies are presented that seek to control two significant GP parameters, training data size and population size. Additionally, GP fitness functions MSE and MAD are detailed and compared. A new fitness function from a recent study that combines aspects of both the MSE and MAD measures is described and new experiments are conducted that compare the performance of this new measure, called the CF measure,

with that of the MSE and MAD measures for a real time series of U.S. GDP data. The CF measure is shown to give better forecasting performance than MSE and MAD measures for the GDP series.

This CF measure contains a parameter, $\Omega$, that represents a threshold between outlier and non-outlier data in a time series. The experiments presented in this study test several different settings for $\Omega$. Further studies might examine the CF measure for other time series and, perhaps, develop some algorithm for adapting the $\Omega$ parameter toward its optimal setting.

Although the algorithm detailed here for adapting the training data size is only applicable to GP forecasting applications, the described algorithm for adapting the population size may be appropriate for other GP applications as well. Future investigations could compare this adaptive population size to the commonly-used static population size for several prevalent GP applications such as circuit design, database query optimization, etc.

## References

1. Andrew M. and Prager R. 'Genetic programming for the acquistion of double auction market strategies.' *Advances in Genetic Programming*, vol. 1 (1994), pg. 355-368.
2. Angeline P. 'Genetic programming and emergent intelligence.' *Advances in Genetic Programming*, vol. 1 (1994), pg. 75-98.
3. Banzhaf W. and Langdon W. 'Some considerations on the reason for bloat.' *Genetic Programming and Evolvable Machines*, vol. 3 (2002), pg. 81-91.
4. Chambers L., editor. *Practical Handbook of Genetic Algorithms: Applications*. CRC Press, 1995.
5. Chen S. and Yeh C. 'Toward a computable approach to the efficient market hypothesis: an application of genetic programming.' *Journal of Economics Dynamics and Control*, vol. 21 (1996), pg. 1043-1063.
6. Chen S., Yeh C., and Lee W. 'Option pricing with genetic programming.' *Genetic Programming 1998: Proceedings of the Third Annual Conference*, vol. 1 (1998), pg. 32-37.
7. Chiraphadhanakul S., Dangprasert P., and Avatchanakorn V. 'Genetic algorithms in forecasting commercial banks deposit.' *Proceedings of the IEEE International Conference on Intelligent Processing Systems*, vol. 1 (1997), pg. 557-565.
8. Deboeck G., editor. *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic and Financial Markets*. John Wiley and Sons, Inc., 1994.
9. Diebold F. *Elements of Forecasting*. International Thomson Publishing, 1998.
10. Fernandez T. and Evett M. 'Training period size and evolved trading systems.' *Genetic Programming 1997: Proceedings of the Second Annual Conference*, vol. 1 (1997), pg. 95.
11. Gately E. *Neural Networks for Financial Forecasting*. John Wiley and Sons, Inc., 1996.
12. Gathercole C. and Ross P. 'Small populations over many generations cand beat large populations over few generations in genetic programming.' *Genetic*

*Programming 1997: Proceedings of the Second Annual Conference*, vol. 1 (1997), pg. 111-118.
13. Goto Y., Yukita K., Mizuno K., and Ichiyanagi K. 'Daily peak load forecasting by structured representation on genetic algorithms for function fitting.' *Transactions of the Institute of Electrical Engineers of Japan*, vol. 119 (1999), pg. 735-736.
14. Hiden H., M$^{c}$Kay B., Willis M., and Tham M. 'Non-linear partial least squares using gentic programming.' *Genetic Programming 1998: Proceedings of the Third Annual Conference*, vol. 1 (1998), pg. 128-133.
15. Iba H. and Sasaki T. 'Using genetic programming to predict financial data.' *Proceedings of the Congress of Evolutionary Computation*, vol. 1 (1999), pg. 244-251.
16. Iba H. and Nikolaev N. 'Genetic programming polynomial models of financial data series.' *Proceedings of the 2000 Congress of Evolutionary Computation*, vol. 1 (2000), pg. 1459-1466.
17. Jeong B., Jung H., Park N. 'A computerized causal forecasting system using genetic algorithms in supply chain management.' *The Journal of Systems and Software*, vol. 60 (2002), pg. 223-237.
18. de Jong E., Watson R., and Pollack J. 'Reducing bloat and promoting diversity using multi-objective methods.' *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, vol. 1 (2001), pg. 11-18.
19. Jonsson P. and Barklund J. 'Characterizing signal behavior using genetic programming.' *Evolutionary Computing: Lecture Notes in Computer Science*, vol. 1143 (1996), pg. 62-72.
20. Ju Y., Kim C., and Shim J. 'Genetic based fuzzy models: interest rate forecasting problem.' *Computers and Industrial Engineering*, vol. 33 (1997), pg. 561-564.
21. Kaboudan M. 'Forecasting with computer-evolved model specifications: a genetic programming application.' *Computer and Operations Research*, vol. 30 (2003), pg. 1661-1681.
22. Kaboudan M. 'Genetically evolved models and normality of their residuals.' *Journal of Economics Dynamics and Control*, vol. 25 (2001), pg. 1719-1749.
23. Kaboudan M. 'Forecasting stock returns using genetic programming in C++.' *Proceedings of 11$^{th}$ Annual Florida Artificial Intelligence International Research Symposium*, vol. 1 (1998), pg. 502-511.
24. Kaboudan M. 'Genetic programming prediction of stock prices.' *Computational Economics*, vol. 6 (2000), pg. 207-236.
25. Kaboudan M. 'Genetic evolution of regression models for business and economic forecasting.' *Proceedings of the Congress of Evolutionary Computation*, vol. 2 (1999), pg. 1260-1268.
26. Kaboudan M. 'A measure of time series' predictability using genetic programming applied to stock returns.' *Journal of Forecasting*, vol. 18 (1999), pg. 345-357.
27. Kim D. and Kim C. 'Forecasting time series with genetic fuzzy predictor ensemble.' *IEEE Transactions on Fuzzy Systems*, vol. 5 (1997), pg. 523-535.
28. Kitchen J. and Monaco R. 'Real-time forecasting in practice.' *Business Economics: the Journal of the National Association of Business Economists*, vol. 38 (2003), pg. 10-19.
29. Koza J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

30. Langdon W. 'The evolution of size in variable length representations.' *1998 IEEE International Conference of Evolutionary Computation*, vol. 1 (1998), pg. 633-638.
31. Langdon W. and Poli R. 'Fitness causes bloat.' *Soft Computing in Engineering Design and Manufacturing*, vol. 1 (1997), pg. 13-22.
32. Lee D., Lee B., and Chang S. 'Genetic programming model for long-term forecasting of electric power demand.' *Electric Power Systems Research*, vol. 40 (1997), pg. 17-22.
33. Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.
34. Eiben A., Hinterding R., and Michalewicz Z. 'Parameter control in evolutionary algorithms.' *IEEE Transactions on Evolutionary Computation*, vol. 3 (1999), pg. 124-141.
35. Mulloy B., Riolo R., and Savit R. 'Dynamics of genetic programming and chaotic time series prediction.' *Genetic Programming 1996: Proceedings of the First Annual Conference*, vol. 1 (1996), pg. 166-174.
36. Neely C. and Weller P. 'Predicting exchange rate volatility: genetic programming versus GARCH and RiskMetrics$^{TM}$.' *The Federal Reserve Bank of St. Louis*, (2002).
37. Smith K., Gupta J. *Neural Networks in Business: Techniques and Applications*. Idea Group Pub., 2002.
38. Trippi R., Turban E., editors. *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real-World Performance*. Irwin Professional Pub., 1996.
39. U.S. Department of Commerce Bureau of Economic Analysis. http://www.bea.doc.gov/bea/glossary/glossary_g.htm, 2004.
40. Venkatesan R. and Kumar V. 'A genetic algorithms approach to growth phase forecasting of wireless subscribers.' *International Journal of Forecasting*, vol. 18 (2002), pg. 625-646.
41. Wagner N. and Michalewicz Z. 'Genetic programming with efficient population control for financial times series prediction.' *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, vol. 1 (2001), pg. 458-462.
42. Wagner N., Michalewicz Z., Khouja M., and McGregor R. 'Time series forecasting for dynamic environments: the DyFor genetic program model.' To appear in *IEEE Transactions on Evolutionary Computation*, 2006.