An Analysis of Adaptive Windowing for Time Series Forecasting in Dynamic Environments: Further Tests of The DyFor GP Model

Neal Wagner
Augusta State University
Department of Mathematics and Computer Science
2500 Walton Way
Augusta, GA 30904, USA
nwagner@aug.edu

Zbigniew Michalewicz
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

ABSTRACT

Genetic Programming (GP) has proved its applicability for time series forecasting in a number of studies. The Dynamic Forecasting Genetic Program (DyFor GP) model builds on the GP technique by adding features that are tailored for the forecasting of time series whose underlying data-generating processes are non-static. Such time series often appear for real-world forecasting concerns in which environmental conditions are constantly changing. In a previous study the DyFor GP model was shown to improve upon the performance of GP and other benchmark models for a set of simulated and real time series. The distinctive feature of DyFor GP is its adaptive data window adjustment. This feedback-driven window adjustment is designed to automatically hone in on the currently active process in an environment where the generating process varies over time. This study further investigates this adaptive windowing technique and provides an analysis of its dynamics for constructed time series with non-static data-generating processes. Results show that DyFor GP is able to capture the moving processes more accurately than standard GP and offer insight for further improvements to DyFor GP.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Algorithms

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

Keywords

Genetic programming, time series, dynamic and uncertain environments, forecasting

1. INTRODUCTION

Forecasting is an integral part of everyday life. Businesses, governments, and people alike make, use, and depend on forecasts for a wide variety of concerns. Current methods of time series forecasting require some element of human judgment and are subject to error. When the information to be forecasted is well-understood, the error may be within acceptable levels. However, often the forecasting concern is not well-understood and, thus, methods that require little or no human judgment are desired. Additionally, many forecasting situations are set in environments with continuously shifting conditions. These situations call for methods that can adjust and adapt to the changing conditions.

The Dynamic Forecasting Genetic Program (DyFor GP) model is built on the Genetic Programming (GP) technique and uses an adaptive data windowing technique that is designed to automatically adjust to underlying data-generating processes that vary over time. In a previous study, DyFor GP was introduced and tested on a set of simulated and real time series [18]. Results show that DyFor GP improved upon the performance of GP and other benchmark models.

The aim of this study is to further investigate the adaptive windowing technique of DyFor GP and provide an analysis of its dynamics. Time series whose data-generating processes vary over time are constructed and the DyFor GP model is compared to a standard GP model for a number of forecasting experiments.

The rest of this paper is organized as follows: section 2 is a brief review of existing time series forecasting methods including GP and DyFor GP, section 3 describes the construction of time series with varying underlying processes, section 4 details experiments involving these constructed time series and provides an analysis of DyFor GP's windowing dynamics, and section 5 concludes.

2. REVIEW OF EXISTING TIME SERIES FORECASTING METHODS

Existing time series forecasting methods generally fall into two groups: classical methods which are based on statistical concepts, and modern heuristic methods which are based on algorithms from the field of artificial intelligence. The following gives a review of these forecasting methods and is taken in part from a survey of forecasting methods found in [18].

Classical time series forecasting methods can be subdivided into the following categories: (1) exponential smoothing methods, (2) regression methods, (3) autoregressive integrated moving average (ARIMA) methods, (4) threshold methods, and (5) generalized autoregressive conditionally heteroskedastic (GARCH) methods. The first three categories listed above can be considered as linear methods, that is methods that employ a linear functional form for time series modelling, and the last two as non-linear methods. See Makridakis *et al.* [10] for a discussion of these statistical methods.

Most modern heuristic methods for time series forecasting fall into two major categories: methods based on neural networks and methods based on evolutionary computation. We can refine the latter category by dividing it further into methods based on genetic algorithms, evolutionary programming, and genetic programming (GP).

For methods based on evolutionary computation, the process of biological evolution is mimicked. A population of candidate solutions is created. Each solution is then ranked and new populations of solutions are generated by selecting fitter solutions and applying a crossover or mutation operation. 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 forecasting applications a GP solution represents a forecasting model with operators that are mathematical functions and inputs that are lagged time series values and/or explanatory variables. 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. GP forecasting solutions are ranked based on

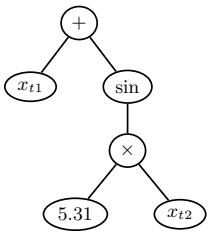

Figure 1: GP representation of forecasting solution $x_{t1} + \sin(5.31x_{t2})$

their prediction error over a set of training data. Crossover in GP is performed by (randomly) selecting a single subtree from each of two parent trees and then swapping them to produce two offspring trees. Mutation is performed by (randomly) selecting a single subtree from a single parent tree and replacing it with a randomly generated tree.

GP was developed by Koza [8] 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.[1] Numerous studies have applied GP to time series forecasting with favorable results. Some examples of these include [3, 7, 6, 5, 4, 13, 18].

The heuristic methods listed above are non-linear and, thus, they are able to capture many aspects displayed by actual data. NN, GP, and EP have the added advantage that the forecasting model need not be prescribed, allowing for automatic discovery of a befitting functional form. However, like the classical methods discussed above, these methods assume a static environment. If the underlying data generating process shifts, the methods must be reevaluated in order to accomodate the new process. Additionally, these methods require that the number of historical time series data used for analysis be designated *a priori*. This presents a problem in non-static environments because different segments of the time series may have different underlying data generating processes. For example, a time series representing the daily stock value of a major U.S. airline is likely to have a different underlying process before September 11, 2001 than it does afterwards. If analyzed time series data span more than one underlying process, forecasts based on that analysis may be skewed.

Consider the subset of time series data shown in figure 2. Suppose this represents the most recent historical data

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

Figure 2: Time series containing segments with differing underlying processes.

and has been chosen for analysis. Suppose further that the subset consists of two segments each with a different underlying 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. Because both segments are analyzed, the forecasting model is distorted unless human judgment is brought to bear. Some degree of human judgment is necessary to assign the number of historical data to be used for analysis. If the time series is not well-understood, then the assignment may contain segments with disparate underlying processes.

The DyFor GP model is built using GP with added features specifically designed for non-static environments. DyFor GP uses an adaptive windowing technique to automatically determine the appropriate analysis window (i.e., the number of recent historical data whose underlying data generating process corresponds to current environment). Also, DyFor GP adapts to changing conditions "on-the-fly" (i.e., without the need for halting and restarting). The following sections describe the features of DyFor GP.

2.1 Natural Adaptation: A Sliding Window of Time

In biological evolution organisms evolve to suit the occurrent conditions of their environment. When conditions shift, successful organisms adapt to the new surroundings. Over

[1] In [8] Koza refers to this as "symbolic regression."

many generations and several environmental shifts, enduring organisms represent highly adaptive solutions that can survive and thrive in a variety of settings. A time series arising from real-world circumstances can be viewed in a similar light. Different segments of the time series may be produced by different underlying data generating processes. Each segment can be thought of as one set of environmental conditions. A successful forecasting model might be seen as an adaptive organism that has evolved through pre-existing environments and gained valuable strengths along the way.

To model this natural adaptation through many environmental settings, a sliding window of time is used. For the DyFor GP model, analysis starts at the beginning of the available historical data. Some initial windowsize (number of data observations to analyze) is set and several generations of DyFor GP are run 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

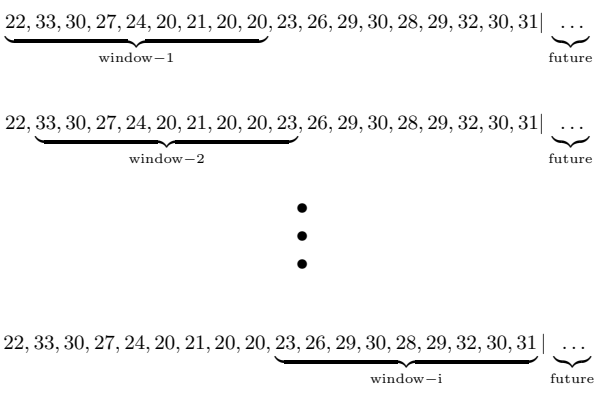

Figure 3: A sliding data analysis window.

of available historical data. The set of several generations run on a single analysis window is referred to as a "dynamic generation." Thus, a single run of the DyFor GP includes several dynamic generations (one for each window slide) on several different consecutive analysis windows.

This sliding window feature allows the DyFor GP to analyze all existing data and take advantage of previously observed patterns. As the window slides through past data, solutions glean useful knowledge making it easier for them to adapt to and predict the current environment.

2.2 Adapting the Analysis Window

As mentioned previously, designating the correct size for the analysis window is critical to the success of any forecasting model. Automatic discovery of this windowsize is indispensable when the forecasting concern is not well-understood. With each slide of the window, the DyFor GP adjusts its windowsize dynamically. 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 dynamic generations at the beginning of the historical data with windowsizes n and $n+i$, use the best solution for each of these two independent runs to predict a number of future data points, and measure their predictive accuracy.
3. 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 $n + 2i$.
4. Slide the analysis window 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.
5. Repeat the previous two steps until the analysis window reaches the end of historical data.

Thus, at each slide of the analysis window, predictive accuracy is used to determine the direction in which to adjust the windowsize.

Consider the following example. Suppose the time series given in figure 4 is to be analyzed and forecast. As depicted in the figure, this time series consists of two segments each with a different underlying data generating process. The sec-

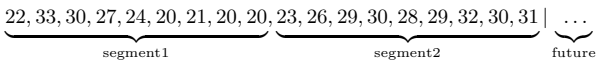

Figure 4: Time series containing segments with differing underlying processes.

ond 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 but may contain patterns that can be learned and exploited when forecasting the current environment. If there is no knowledge available concerning these segments, automatic techniques are required to discover the correct windowsize needed to forecast the current setting. The 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 5 illustrates these steps. In the figure **win1** and **win2** represent data analysis windows of size 3 and 4, respectively, and **pred** represents the future data predicted.

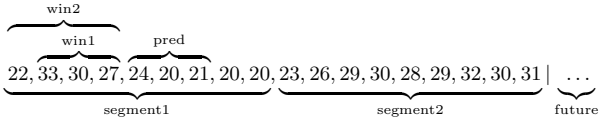

Figure 5: Initial steps of window adaptation.

The data predicted in these initial steps lies inside the first segment's process and, because the dynamic generation

involving analysis 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 analysis window then slides to include the next time series value, two new dynamic generations are run, and the best solutions for each are used to predict future data. Figure 6 depicts these steps. In the figure, data analysis windows **win1** and **win2** now include the next time series value, 24, and **pred** has shifted one value to the right.

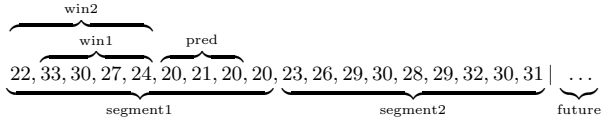

Figure 6: Window adaptation after the first window slide. Note: win1 and win2 have size 4 and 5, respectively.

This process of selecting two new windowsizes, sliding the analysis window, running two new dynamic generations, and predicting future data is repeated until the analysis window reaches the end of historical data. It may be noted that while the prediction data, **pred**, lies entirely inside the first segment, the data analysis windows, **win1** and **win2**, are likely to expand to encompass a greater number of appropriate data. However, after several window slides, when the data analysis window spans data from both the first and second segments, it is likely that the window adjustment reverses direction. Figures 7 and 8 show this phenomenon.

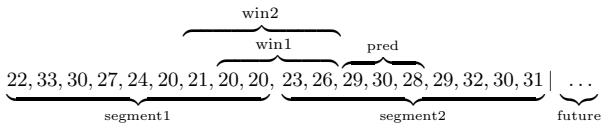

Figure 7: Window adaptation when analysis spans both segments. Note: the smaller analysis window, win1, is likely to have better prediction accuracy because it includes less inappropriate data.

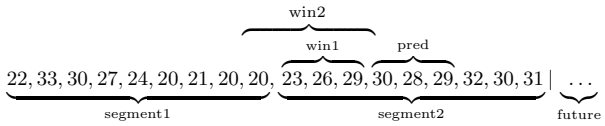

Figure 8: Window adaptation when analysis spans both segments. Note: win1 and win2 have contracted to include less inappropriate data.

In figure 7 **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 analysis window **win1** has better prediction accuracy than that involving **win2** because **win1** includes less data produced by a process that is no longer in effect. If this is so, the two new windowsizes selected for **win1** and **win2** are sizes 3 and 4, respectively. Thus, as the analysis window slides to incorporate the next time series value, it also contracts to include a smaller number of inappropriate data. In figure 8 this contraction is shown.

As illustrated in the above example, the DyFor GP uses predictive accuracy to adapt the size of its analysis window automatically. When the underlying process is stable (i.e., the analysis window is contained inside a single segment), the windowsize is likely to expand. When the underlying process shifts (i.e., the analysis window spans more than one segment), the windowsize is likely to contract. The following section describes the time series data used for experimentation.

3. TIME SERIES DATA

The goal of this study is to further investigate DyFor GP's adaptive windowing technique. DyFor GP was designed for forecasting concerns in which the underlying data-generating process varies over time. Time series are constructed that contain segments with differing underlying processes. The segments are generated by known processes found in the literature. The following presents these processes.

1. Ozaki's simple linear function [15] (referred to below as OZ) has low structural complexity and is given by

$$Y_{t+1} = 1.8708Y_t - Y_{t-1}, \tag{1}$$

where Y_{t+1} is the time series value corresponding to one time period in the future and Y_t and Y_{t-1} are the time series values of the current time period and one time period in the past, respectively.

2. May introduced a difference equation known as the logistic map [11] (referred to below as LG). It is a low-dimensional first-order nonlinear chaotic system. Its structural complexity is also low but higher than that of OZ and is given by

$$Y_{t+1} = 4Y_t(1 - Y_t). \tag{2}$$

3. Henon introduced a low-dimensional second-order nonlinear chaotic difference equation that is known as the Henon map [2] (referred to below as HEN). It may have slightly higher structural complexity than LG and is given by

$$Y_{t+1} = 0.3Y_{t-1} + 1 - 1.4{Y_t}^2. \tag{3}$$

4. The Mackey-Glass equation is known to have the highest structural complexity among frequently-studied chaotic ordinary differential equations [9]. A discretized difference equivalent of the Mackey-Glass differential equation used by Koza [8] and Oakley [14] is employed (referred to below as MG) and is given by

$$Y_{t+1} = Y_t + \frac{0.2Y_{t-30}}{1 + {Y_{t-30}}^{10}} - 0.1Y_t, \tag{4}$$

where Y_{t-30} is the time series value corresponding to 30 time periods in the past.

A single time series is constructed by connecting distinct segments each generated by one of the above listed processes. The time series constructed for this study contain three segments. The first segment (segment1) is generated by one process; the second segment (segment2) is generated by a

different process; and the third segment (segment3) is generated by the same process that was used to generate segment1. The following presents the constructed time series. Each time series listed below consists of 400 values (data points). The first 100 values of each series are generated by segment1's process and are intended for initial training.

1. LG-OZ-LG—this is a time series constructed from the LG and OZ processes. Segment1 is generated by LG (data points 1-200), segment2 is generated by OZ (data points 201-296), and segment3 is generated by LG (data points 297-400). Note that segments 2 and 3 use lagged values from the previous segment to calculate their initial data points. This is the case for the remaining constructed time series as well. Segment1 uses a single lagged value of 0.9000 to calculate its initial data point.

2. MG-HEN-MG—this is a time series constructed from the MG and HEN processes. Segment1 is generated by MG (data points 1-200), segment2 is generated by HEN (data points 201-300), and segment3 is generated by MG (data points 301-400). Segment1's data points were produced by seeding the MG process with a pseudorandom sequence (31 lagged values), generating the MG series, and then discarding the first 1000 data points. This procedure for generating the MG series was used by Oakley [14].

Figures 9 and 10 give a graphical depiction of the constructed time series. The following section details experiments involving these time series.

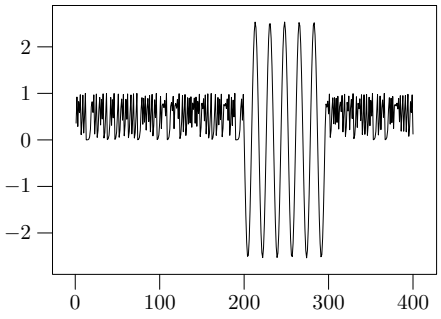

Figure 9: LG-OZ-LG constructed series.

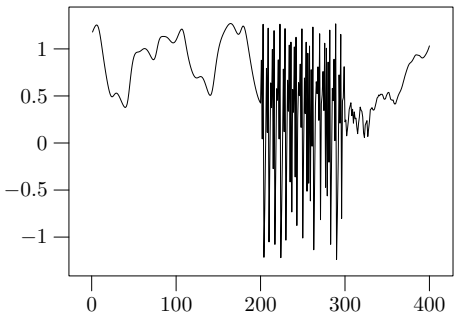

Figure 10: MG-HEN-MG constructed series.

4. EXPERIMENTS

The aim of this study is to compare DyFor GP to standard GP for the forecasting of time series whose data-generating processes vary over time. As discussed in section 2, GP, when used for forecasting, assumes a static environment. A forecaster uses his or her own judgment to select some historical data for GP to train on, GP trains on that data and produces a forecasting model, and then that model is used to forecast future data. In a real-world setting the forecasting environment is unlikely to remain static indefinitely. Thus, practitioners are required to repeat the above procedure from time to time in order to keep their GP forecasting model up to date with current conditions. With this in mind, two versions of standard GP are included in our experiments. In one version GP is run once on some training data and then used to forecast all future data without any updating. In a second version the GP forecasting model is periodically updated (i.e. retrained). The following sections describe the experimental setup and observed results.

4.1 Experimental Setup

Three forecasting models are compared: DyFor GP, standard GP without periodic update, and standard GP with periodic update. Each of the models is run on the constructed time series described in section 3. The first 100 data points are used for initial training of the models and the remaining 300 points are used for testing.

For DyFor GP, forecasts are generated in a "real-time" fashion in the following way. After initial training on the first 100 data points, the first forecast is produced and the analysis window is then slid to incorporate the actual data for that time period. Analysis (training) resumes, and then the second forecast is produced. This procedure is repeated for each forecast until all (300) forecasts have been generated. It should be emphasized that forecasts are generated using an out-of-sample methodology where no data beyond the point of forecast is utilized for analysis or model construction. For the standard GP model without periodic update, GP is run on the initial training data (first 100 data points) and then the resultant model is used to forecast the remaining 300 data points without any subsequent retraining.

For the standard GP model with periodic update, GP runs on the initial training data, forecasts some number of future data, and then, after some specified number of time periods, the GP model is retrained using more recent data. The question is: how often should this GP model be retrained? In a real setting a forecaster would use his or her judgment to determine when or how often to retrain the GP model. For example, a forecaster might decide to retrain the GP model every 50 time periods. This would mean that after initial training, the resultant model is used to produce forecasts for the next 50 time periods and then the data corresponding to those 50 time periods is incorporated as training data and the model is retrained. This procedure is then repeated every 50 time periods until all required forecasts are generated. For these experiments, the retraining is executed after *each* time period representing the maximum possible frequency. This is done in order to give standard GP the best chance of performing efficiently in an environment of changing conditions.

GP (and DyFor GP) employ the elements of a terminal set and a function set as building blocks from which to con-

struct forecasting models.[2] For experiments executed on the LG-OZ-LG series, the terminal set includes two lags of the autoregressive variable as two lags are all that are necessary to specify either the LG or OZ process. For experiments executed on the MG-HEN-MG series, the terminal set includes 30 lags of the autoregressive variable as they are necessary to specify the MG process (the HEN process requires only two lags). The function set for both experiments consists of operators $+$, $-$, $\times$, $\div$, sin, cos, square root, exp, and ln.[3]

DyFor GP requires that a number of parameters be specified before a run. Some of these are general GP parameters commonly found in any GP application. Some of these are special parameters only used by DyFor GP. Tables 1 and 2 give general GP parameter values and specific DyFor GP parameter values, respectively. All parameter values listed

Table 1: General GP parameter settings.

Parameter	Value
crossover rate	0.9
reproduction rate	0.0
mutation rate	0.1
max. no. of generations	41
termination	max. gens. reached
elitism used?	yes
fitness measure	mean squared error
population size	70000 total nodes

in table 1 match those used in [18] with the exception of population size (which is increased here due to greater computational resources).[4] [5]

Table 2: Specific DyFor GP parameter settings.

Parameter	Value
max windowsize	200
min windowsize	20
start windowsize	80
window difference	20

The "max windowsize" and "min windowsize" parameters in table 2 specify the maximum and minimum analysis windowsizes, respectively. As described in section 2.2, DyFor GP's adaptive windowing calls for using two data windows of differing sizes. Parameter "start windowsize" refers to the initial windowsize setting of the smaller of the two windows and parameter "window difference" refers to the size difference between the larger and the smaller window.

The GP algorithm is essentially a fitness-driven random search. When GP is applied to forecasting, the search space is the set of all possible mathematical expressions that can be constructed using specified operands (inputs) and standard mathematical operators with no restrictions concerning the functional form of the expressions created. Thus the space effectively includes any conceivable expression (either linear or non-linear) and is, in general, intractable for conventional deterministic algorithms. The size of the search space coupled with the stochastic nature of the evolutionary process cause the results of a GP-based forecasting experiment to vary from run to run. Thus, a common practice is to execute a set of GP runs (usually 20 to 100—see, for example, [7] and [6]). For experiments of this study, a set of 20 runs is executed for each model. The following section describes observed results.

[2]See Koza [8] for more information on the terminal and function sets.

[3]Operators $\div$, square root, exp, and ln are protected from undefined or unbounded behavior as is done in experiments conducted by Koza [8].

[4]For these experiments population size is specified as a maximum number of solution tree nodes over the entire population instead of a limit on the number of solution trees. For more information on this practice, please see [17] and [18].

[5]These experiments were conducted using 8-12 nodes of an IBM eServer 1350 Linux cluster. Please see [16] for more information.

4.2 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 mean squared error (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. Tables 3 and 4 give the observed results for the constructed series.[6]

Table 3: LG-OZ-LG series forecasting results.

Model	mean MSE	std. dev.
standard GP (without update)	0.9979	6.5540
standard GP (with update)	0.3047	0.0334
DyFor GP	0.2344	0.0567

Table 4: MG-HEN-MG series forecasting results.

Model	mean MSE	std. dev.
standard GP (without update)	0.6039	0.4216
standard GP (with update)	0.1960	0.0830
DyFor GP	0.1880	0.0278

The tables reveal some interesting results. The first observation is that standard GP without periodic update is the worst performing model of the three. It gives the worst performance for both the LG-OZ-LG and MG-HEN-MG series. This is expected since the other models have opportunity to train on data from all segments while this model trains only on data from the first segment of each series and is never updated (retrained).

Comparing standard GP with update and DyFor GP shows that the latter model gives superior performance for both series tested. This is a compelling result for a few reasons.

1. This version of standard GP is updated with maximal frequency and represents the limit of standard GP's ability to deal with changing processes.

2. The essential difference between the two models is DyFor GP's adaptive windowing technique.

[6]In these tables outliers are excluded before the mean MSE is calculated. An outlier is defined to be a MSE value that differs from the overall mean MSE by more than two standard deviations.

3. Giving superior performance for both series provides empirical evidence that DyFor GP's adaptive window is better able to hone in on the currently active process in a non-static environment.

4. Additionally, DyFor GP was able to able to adapt to the changing processes *automatically*, that is without the need for rerunning as was required by standard GP.

In the following section the behavior of DyFor GP's adaptive window is examined.

4.3 An Analysis of Window Dynamics

As described in the previous section, DyFor GP gives the best performance for both series tested. This performance edge can only be due to the adaptive windowing technique as this is the essential difference between DyFor GP and standard GP. Here, the window behavior of DyFor GP is examined. Figures 11 and 12 give the average windowsize (over all 20 runs) of the DyFor GP model at each time period for each series, respectively. In the figures vertical gridlines labeled "SEG2" and "SEG3" mark the beginning of segments 2 and 3 in each series, respectively.

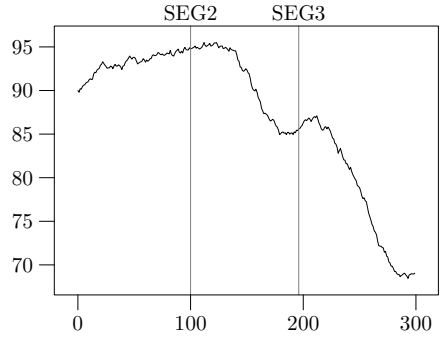

Figure 11: Average DyFor GP windowsize for LG-OZ-LG series.

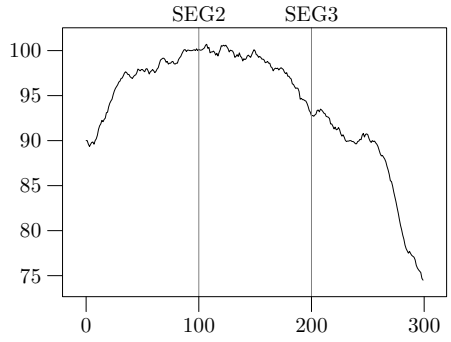

Figure 12: Average DyFor GP windowsize for MG-HEN-MG series.

The figures reveal some interesting window behavior. For both series similar dynamics are seen. The following gives these dynamics.

1. During segment1 expansion occurs.

2. At the beginning of segment2 there is a delay period during which, presumably, DyFor GP notices the process shift and then contraction begins.

3. Towards the end of segment2, the contraction abates.

4. After another delay at the beginning of segment3, contraction picks up again.

These dynamics are consistent with expected DyFor GP window behavior (described in section 2.2). Data window expansion inside segment1 follows as this allows the model to train on a greater number of appropriate data. When the underlying data-generating process shifts to produce segment2, data window contraction allows the model to train on less data from the old (no longer active) process, which in turn gives more weight to newer data produced by the currently active process. Toward the end of segment2 and at the beginning of segment3, contraction abates (but is not reversed) because the data window has now entirely passed out of segment1 but has not had enough time for significant expansion (inside segment2) before segment3 begins. After some delay, DyFor GP notices the process shift that produces segment3 and contraction picks up again.

The window behavior seen for these experiments provide empirical evidence for the efficacy of DyFor GP's adaptive windowing technique. DyFor GP was able to automatically adjust its windowsize to accomodate process movements in a non-static environment. While these experiments show DyFor GP to be a viable model for non-static environments, they also give insight into the shortcomings of the current windowing technique and point to further enhancements that could be made. One shortcoming made apparent by these experiments is that once the model notices that a process shift has occurred, it takes a long time (i.e., many successive contractions over many slides of the data window) to correctly adjust to the new process. If, after a relatively short period, the process shifts again as in these experiments, DyFor GP is now in "catch up" mode where its window adjustment is always lagging behind the actual process movements. This is because the current windowing technique compares only two different windowsizes and can only adjust itself by small increments at each dynamic generation (slide of the data window). The windowing technique could be improved by comparing more windowsizes at each dynamic generation or by allowing for greater adjustment increments when conditions call for them. If computational resources were not an issue, the DyFor GP model could easily be improved by comparing several different windowsizes (rather than just two) at each dynamic generation. This would give a more accurate estimate of the optimal windowsize and would allow the model to "jump" to a good windowsize without having to go through several small adjustments.

While the improvement to DyFor GP's windowing technique suggested above may not be feasible for computational resources available today, the following describes another enhancement along this line of inquiry that might be. Instead of running a single dynamic generation for each compared windowsize as is currently done, the windowsize itself could be included in the GP chromosome and evolved along with the functional form of the forecasting model. This means that each individual in the GP population would contain two parts: a solution tree (representing the functional form

of the forecasting model) and a corresponding windowsize for that forecasting model to train on. In this way a large number of windowsizes (one for each individual) are compared during only one dynamic generation and DyFor GP could, potentially, find the optimal (or near-optimal) windowsize with much less computation. For example, when a process shift is first noticed, the model could potentially find the correct (smaller) windowsize after only one slide of the data window because many windowsizes are compared. Thus, it would spend less time adjusting to the latest process movement and more time training on the currently active process. The suggested improvement to DyFor GP's windowing technique is termed "self-adaptive" rather than "adaptive" because rather than using feedback from the GP search process to make adjustments, the windowsize is evolved as part of an individual's chromosome [1].

4.4 Some Comments on Adaptability and Computational Overhead

For the experiments of this study, DyFor GP exhibits *adaptive* behavior as it automatically adjusts its data window in the presence of changing conditions. This adaptability is achieved through constant retraining and, thus, a computational expense is incurred that is not present for standard (static) GP. This computational overhead may well be a necessary price of any intelligent software system as without retraining a system cannot hope to adapt to environmental changes.

5. CONCLUSION

In this study the DyFor GP model is tested on constructed time series in which the underlying data-generating process varies over time. The idea is to further investigate DyFor GP's adaptive windowing technique and provide an analysis of its window behavior in the presence of continually shifting conditions. DyFor GP is compared to two versions of standard GP, one of which represents the limit of standard GP's ability to deal with changing processes.

Observed results show that DyFor GP's adaptive window is better able to hone in on the currently active process than standard GP and that this is the reason for DyFor GP's superior forecasting performance. The experiments of this study also provide insight into further enhancement of DyFor GP's windowing technique and a new "self-adaptive" windowing technique is described that could significantly increase the speed of window adjustment.

The experiments presented here highlight DyFor GP's potential as an adaptive model for real-world forecasting applications which are often characterized by continually shifting conditions. DyFor GP can be considered an example of Adaptive Business Intelligence [12] as it contains features that optimize to current conditions, predict the future, and adapt to environmental changes.

6. REFERENCES

[1] Eiben A., Hinterding R., and Michalewicz Z. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124–141, 1999.

[2] Henon M. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50:69, 1976.

[3] Iba H. and Nikolaev N. Genetic programming polynomial models of financial data series. In *Proceedings of the 2000 Congress of Evolutionary Computation*, pages 1459–1466. IEEE, 2000.

[4] Kaboudan M. Genetic evolution of regression models for business and economic forecasting. In *Proceedings of the 1999 Congress of Evolutionary Computation*, pages 1260–1268. IEEE, 1999.

[5] Kaboudan M. Genetic programming prediction of stock prices. *Computational Economics*, 6:207–236, 2000.

[6] Kaboudan M. Genetically evolved models and normality of their residuals. *Journal of Economics Dynamics and Control*, 25:1719–1749, 2001.

[7] Kaboudan M. Forecasting with computer-evolved model specifications: a genetic programming application. *Computer and Operations Research*, 30:1661–1681, 2003.

[8] Koza J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[9] Mackey M. and Glass L. Oscillation and chaos in physiological control systems. *Science*, 197:287, 1977.

[10] Makridakis, S., Wheelwright, S., and Hyndman, R. *Forecasting: Methods and Applications*. John Wiley and Sons, Inc., Hoboken, NJ, 1998.

[11] May R. Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467, 1976.

[12] Michalewicz Z., Schmidt M., Michalewicz M., and Chiriac C. *Adaptive Business Intelligence*. Springer, 2007.

[13] Neely C. and Weller P. Predicting exchange rate volatility: genetic programming versus GARCH and RiskMetrics. Technical report, The Federal Reserve Bank of St. Louis, 2002.

[14] Oakley H. Genetic programming for nonlinear equation fitting to chaotic data. In Back T., Fogel D., and Michalewicz Z., editor, *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[15] Ozaki, T. Non-linear time series models and dynamical systems. In Hannan E., Krishnaiah P., Rao M., editor, *Handbook of Statistics*. Elsevier, 1985.

[16] South Australian Partnership for Advanced Computing. High performance and grid computing - hydra cluster. http://www.sapac.edu.au/systems/hydra/index.html, 2007.

[17] Wagner N. and Michalewicz Z. Genetic programming with efficient population control for financial times series prediction. In *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 458–462, 2001.

[18] Wagner N., Michalewicz Z., Khouja M., and McGregor R. Time series forecasting for dynamic environments: the DyFor genetic program model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452, 2007.