Time Series Forecasting for Dynamic Environments: The DyFor Genetic Program Model

Neal Wagner, Zbigniew Michalewicz, Moutaz Khouja, and Rob Roy McGregor

***Abstract*—Several studies have applied genetic programming (GP) to the task of forecasting with favorable results. However, these studies, like those applying other techniques, have assumed a static environment, making them unsuitable for many real-world time series which are generated by varying processes. This study investigates the development of a new "dynamic" GP model that is specifically tailored for forecasting in nonstatic environments. This Dynamic Forecasting Genetic Program (DyFor GP) model incorporates features that allow it to adapt to changing environments automatically as well as retain knowledge learned from previously encountered environments. The DyFor GP model is tested for forecasting efficacy on both simulated and actual time series including the U.S. Gross Domestic Product and Consumer Price Index Inflation. Results show that the performance of the DyFor GP model improves upon that of benchmark models for all experiments. These findings highlight the DyFor GP's potential as an adaptive, nonlinear model for real-world forecasting applications and suggest further investigations.**

***Index Terms*—Dynamic, forecasting, genetic programming, parameter adaptation, time series.**

I. Introduction

FORECASTING is an integral part of everyday life. Businesses, governments, and members of the public 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 forecast 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 aim of this study is to investigate the development of a new adaptive model that is specifically tailored for forecasting time series produced by nonstatic environments. The proposed model is based on genetic programming (GP) with additional features that seek to capture such dynamically changing time series. This Dynamic Forecasting Genetic Program (DyFor GP) model incorporates methods to adapt to changing environments automatically, and retain knowledge learned from previously encountered environments. Such past-learned knowledge may prove useful when current environmental conditions resemble those of a prior setting. Specifically, this knowledge allows for faster convergence to current conditions by giving the model searching process a "head-start" (i.e., by narrowing the model search space).

The rest of this paper is organized as follows. Section II is a brief review of existing time series forecasting methods, Section III describes the DyFor GP model, Section IV details experiments involving the DyFor GP model, and Section V concludes this paper.

Manuscript received January 26, 2006; revised April 25, 2006. This work was supported in part by KBN under Grant 3T11C00728.

N. Wagner is with the Department of Mathematics and Computer Science, Augusta State University, Augusta, GA 30904 USA (e-mail: nwagner@aug.edu).

Z. Michalewicz is with the School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia, and also with the Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, and also with the Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland (e-mail: zbyszek@cs.adelaide.edu.au).

M. Khouja is with the Department of Business Information Systems and Operations Management, University of North Carolina, Charlotte, NC 28223 USA (e-mail: mjkhouja@email.uncc.edu).

R. R. McGregor is with the Department of Economics, University of North Carolina, Charlotte, NC 28223 USA (e-mail: rrmcgreg@email.uncc.edu).

Digital Object Identifier 10.1109/TEVC.2006.882430

II. Review of Existing Time Series Forecasting Methods

Existing time series forecasting methods generally fall into two groups: classical methods which are based on statistical/mathematical concepts, and modern heuristic methods which are based on algorithms from the field of artificial intelligence.

A. Classical Methods

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;
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 modeling, the last two categories can be considered as nonlinear methods.[1]

In exponential smoothing, a forecast is given as a weighted moving average of recent time series observations. The weights assigned decrease exponentially as the observations get older. In regression, a forecast is given as a linear function of one or more explanatory variables. ARIMA methods give a forecast as a linear function of past observations (or the differences of past observations) and error values of the time series itself and past observations of zero or more explanatory variables. See

[1]Regression and ARIMA methods can have a nonlinear functional form, however, this is not common.

Makridakis *et al.* [57] for a discussion of smoothing, regression, and ARIMA methods.

All the linear forecasting methods above assume a functional form which may not be appropriate for many real-world time series. Linear models cannot capture some features that commonly occur in actual data such as asymmetric cycles and occasional outlying observations [57, pp. 433–434]. Regression methods often deal with nonlinear time series by logarithmic or power transformation of the data, however, this technique does not account for asymmetric cycles and outliers.

Threshold methods assume that extant asymmetric cycles are caused by distinct underlying phases of the time series and that there is a transition period (either smooth or abrupt) between these phases. Commonly the individual phases are given a linear functional form and the transition period (if smooth) is modeled as an exponential or logistic function. GARCH methods are used to deal with time series that display nonconstant variance of residuals (error values). In these methods, the variance of error values is modeled as a quadratic function of past variance values and past error values. In [57], [60], and [72], various threshold methods are detailed, while [1], [11], and [22] describe GARCH methods.

The nonlinear methods above, although capable of characterizing features found in actual data such as asymmetric cycles and nonconstant variance of residuals, assume that the underlying data generating process of the time series is constant.[2] The linear methods described above also make this assumption. For actual time series data, this assumption is often invalid as shifting environmental conditions may cause the underlying data generating process to change. For all of the classical forecasting methods listed, human judgment is required to first select an appropriate method, and then set appropriate parameter values for the model's coefficients (or to select an appropriate parameter optimization scheme). In the event that the underlying data generating process changes, the time series data must be reevaluated and a (possibly new) method must be selected with appropriate parameter values. As the task of repeated data monitoring and model selection is complex and time consuming, automatic nonlinear forecasting models that can handle nonstatic environments are desired. The following section contains a discussion of modern heuristic methods used for time series forecasting.

B. Modern Heuristic Methods

Most modern heuristic methods for time series forecasting fall into two major categories:

1) methods based on neural networks (NNs);
2) methods based on evolutionary computation.

We can refine the latter category by dividing it further into methods based on genetic algorithms (GAs), evolutionary programming (EP), and GP.

It is interesting to note that NN, EP, and GP techniques were used to build nonlinear forecasting models, whereas GAs were primarily used to tune the parameters of some (possibly statistical, linear or nonlinear) forecasting model. All of the methods listed above are motivated by the study of biological processes.

NN attempt to solve problems by imitating the human brain. A NN is a graph-like structure that contains an input layer, zero or more hidden layers, and an output layer. Each layer contains several "neurons" which have weighted connections to neurons of the following layer. A neuron from the input layer holds an input variable. For forecasting models, this input is a previous time series observation or an explanatory variable. A neuron from the hidden or output layer consists of an "activation" function [usually the logistic function: $g(u) = 1/(1 + e^{-u})$]. A three-layer feed-forward NN (one hidden layer between an input and output layer) is commonly used for forecasting applications due to its ability to approximate virtually any nonlinear model (if given a sufficient number of neurons at the hidden layer) [88]. Several applications of NN to forecasting are proffered in [28], [75], and [79]. General descriptions of NN can be found in [30] and [88].

For methods based on evolutionary computation, the process of biological evolution is mimicked in order to solve a problem. After an initial population of potential solutions is created, solutions are ranked based on their "fitness." New populations are produced by selecting higher ranking solutions and performing genetic operations of "mating" (crossover) or "mutation" to produce offspring solutions. This process is repeated over many generations until some termination condition is reached.

When GA is applied to forecasting, first an appropriate model (either linear or nonlinear) is selected and an initial population of candidate solutions is created. A candidate solution is produced by randomly choosing a set of parameter values for the selected forecasting model. Each solution is then ranked based on its prediction error over a set of training data. A new population of solutions is generated by selecting fitter solutions and applying a crossover or mutation operation. Crossover is performed by swapping a subset of parameter values from two parent solutions. Mutation causes one (random) parameter from a solution to change. New populations are created until the fittest solution has a sufficiently small prediction error or repeated generations produce no reduction of error.

Consider the following example given by Jeong *et al.* [36]. They choose a linear explanatory model of the form

$$y(k+1) = b_0 + b_1 x_1(k) + b_2 x_2(k) + \cdots + b_n x_n(k) \quad (1)$$

where $y(k+1)$ is the forecast variable at time $k+1$, $x_i(k)$ is the value of explanatory variable x_i at time k, and b_i is the coefficient of explanatory variable x_i. Thus, a candidate solution will be a vector of real numbers representing coefficients b_0 through b_n. Since each coefficient may have a unique range, a scaling technique is used to map the coefficients to the range [0,1]. The process requires that an encoded solution u is decoded back to the original value scheme. The following equation is used for this purpose:

$$b_i(u) = b_i^{\min} + c_i(u)\left|b_i^{\min} - b_i^{\max}\right| \quad (2)$$

[2]Threshold methods do allow for the underlying process to vary between prescribed phases. However, the process is assumed to be constant within each phase and, commonly, only 2 or 4 phases are specified [21], [57].

where $b_i(u)$ is the ith coefficient value for solution $u, i = 0, 1, \ldots, n$, and

$$b_i^{\min} = \begin{cases} (1-\alpha)b_i(s), & \text{if } b_i(s) > 0 \\ (1+\alpha)b_i(s), & \text{if } b_i(s) < 0 \end{cases}$$
$$b_i^{\max} = \begin{cases} (1+\alpha)b_i(s), & \text{if } b_i(s) > 0 \\ (1-\alpha)b_i(s), & \text{if } b_i(s) < 0 \end{cases}.$$

Here, $b_i(s)$ is the ith coefficient value for the current best solution s, $c_i(u)$ is the encoded value of the ith coefficient for solution u, and $\alpha = 0.5$. α serves as a boundary for the coefficients of all solutions in the population, that is they are restricted to be within $\pm 100\alpha\%$ range of the current best solution. The fitness function chosen to evaluate solutions is

$$f = |y(k) - \hat{y}(k)| + \frac{\sum_{j=1}^{k-1} |y(j) - \hat{y}(j)|}{k-1} \tag{3}$$

where $y(k)$ is the observed value for the forecast variable at time k, and $\hat{y}(k)$ is the predicted value for the forecast variable at time k. The first term of (3) gives the deviation of the predicted value from the observed value at current time k, and the second gives the average deviation of predicted values from observed values during the entire training period before time k. Fitter solutions will have lower values for f (with the lowest possible value being 0).

GA has been used successfully for a wide variety of difficult optimization problems including the forecasting of real-world time series. References [5], [61], and [62] give detailed descriptions of GA, while [13], [16], [19], [29], [39], [45], and [82] provide additional examples of GA applied to forecasting.

For EP each candidate solution is represented as a finite-state machine (FSM) rather than a numeric vector. FSM inputs/outputs correspond to appropriate inputs/outputs of the forecasting task. An initial population of FSMs is created and each is ranked according to its prediction error. New populations are generated by selecting fitter FSMs and randomly mutating them to produce offspring FSMs. A parent FSM is mutated by performing one of the following operations:

1) change an output symbol;
2) change a state transition;
3) add a new state;
4) remove a state;
5) change the start state.

EP was devised by Fogel [24] and has applications in many areas. Some examples of successful EP forecasting experiments include [24]–[26], and [73].

In GP, solutions are represented as tree structures instead of numeric vectors or FSMs. Internal nodes of solution trees represent appropriate operators and leaf nodes represent input variables or constants. For forecasting applications, the operators are mathematical functions and the inputs are lagged time series values and/or explanatory variables. Fig. 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 was developed by Koza [49] 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] Numerous studies have applied GP to time series forecasting with favorable results. Some examples of these include [3], [14], [15], [32]–[34], [38], [40]–[44], [52], [63], [66], and [83]. GP has also been used to find successful trading rules from time series data in [27], [59], [67], [68], [80], and [86].

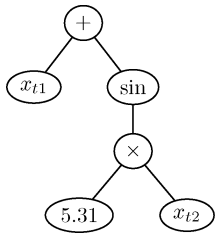


Fig. 1. GP representation of forecasting solution $x_{t1} + \sin(5.31x_{t2})$.

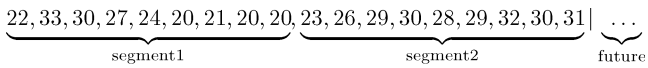


Fig. 2. Time series containing segments with differing underlying processes.

Also, prevalent in the literature are forecasting studies which make use of a technique that is either a close variant to one of the aforementioned methods or a hybrid that employs multiple methods. One common hybrid method is one that combines NN and GA. In these applications, a GA is used to optimize several aspects of a NN architecture [2], [6], [53], [58], [64], [70], [74], [89]. The optimized NN is then used to produce the desired forecasts. Another hybrid method utilizes an EP to evolve both the weights and the topology (i.e., the connectivity) of a NN simultaneously [56], [91]. In [48] and [55], a variant on GA called evolution strategies (ES) is used to generate efficient trading rules for financial time series.

Because the heuristic methods described above are nonlinear, 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 in Section II-A, these methods assume a static environment. If the underlying data generating process shifts, the methods must be reevaluated in order to accommodate 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 nonstatic 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 Fig. 2. Suppose this represents the most recent historical data and has been chosen for analysis. Suppose further that the subset consists of two segments each with a different underlying process. The

[3]In [49], Koza refers to this as "symbolic regression."

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. As 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. This situation highlights the need for forecasting methods that can automatically determine the correct analysis "window" (i.e., the correct number of historical data to be analyzed). This investigation attempts to develop a dynamic forecasting model based on GP that can do just that. Furthermore, this study explores methods that can retain knowledge learned from previously encountered environments. Such past-learned knowledge may prove useful when current environmental conditions resemble those of a prior setting. Specifically, this knowledge allows for faster convergence to current conditions by giving the model searching process a "head-start" (i.e., by narrowing the model search space).

In the following section, the DyFor GP model is presented and its features discussed.

III. The DyFor GP Model

As discussed in the previous section, an adaptive forecasting model that can handle nonstatic environments is sought. The desired model would automatically determine the appropriate analysis window (i.e., the number of recent historical data whose underlying data generating process corresponds to current environment). Also, the model should be able to adapt to changing conditions "on-the-fly" (i.e., without the need for halting and restarting the analysis). An additional boon would be the ability to retain useful knowledge from previously encountered environments so that the current setting can be more accurately captured. In this section, a discussion of the design of such a model is proffered.

A. 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 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 all of the preexisting environments and gained valuable adaptations (strengths) along the way.

To model this natural adaptation through many environmental settings, a sliding window of time is proposed. For the DyFor GP model, analysis starts at the beginning of the available historical data. Some initial window size (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 the available data has been analyzed up to and including the most recent historical data. Fig. 3 illustrates this process. In the figure, | marks the end 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.

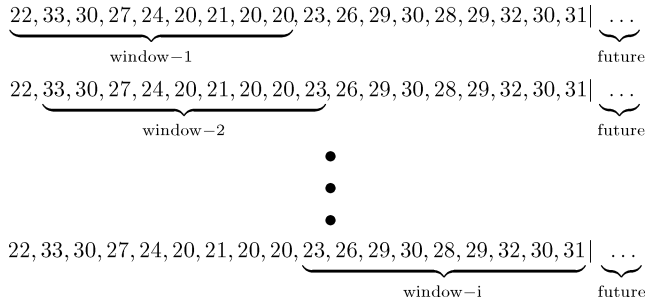


Fig. 3. A sliding data analysis window.

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.

B. Adapting the Analysis Window

As expounded in Section II-B, designating the correct size for the analysis window is critical to the success of any forecasting model. Automatic discovery of this window size is indispensable when the forecasting concern is not well understood. With each slide of the window, the DyFor GP adjusts its window size dynamically. This is accomplished in the following way.

1) Select two initial window sizes, 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 window sizes 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 window sizes based on which window size had better accuracy. For example, if the smaller of the two window sizes (size n) predicted more accurately, then choose two new window sizes, one of size n and one of size $n-i$. If the larger of the two window sizes (size $n+i$) predicted more accurately, then choose window sizes $n+i$ and $n+2i$.
4) Slide the analysis window to include the next time series observation. Use the two selected window sizes 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 window size.

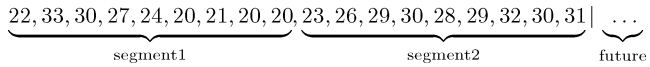


Fig. 4. Time series containing segments with differing underlying processes.

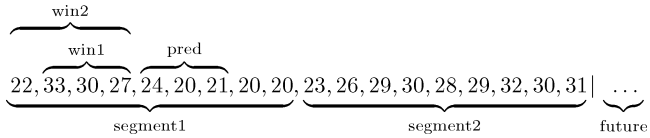


Fig. 5. Initial steps of window adaptation.

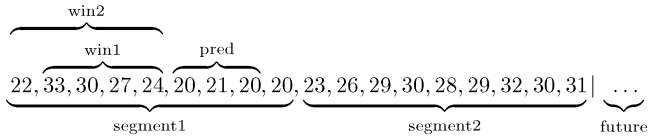


Fig. 6. Window adaptation after the first window slide. Note: **win1** and **win2** have size 4 and 5, respectively.

Consider the following example. Suppose the time series given in Fig. 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 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 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 window size needed to forecast the current setting. The DyFor GP starts by selecting two initial window sizes, one larger than the other. Then, two separate dynamic generations are run at the beginning of the historical data, each with its own window size. After each dynamic generation, the best solution is used to predict some number of future data and the accuracy of this prediction is measured. Fig. 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.

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 window sizes 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. Fig. 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.

This process of selecting two new window sizes, 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

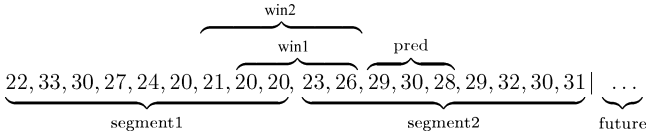


Fig. 7. Window adaptation when analysis spans both segments. Note: the smaller analysis window **win1** is likely to have better prediction accuracy because it includes less inappropriate data.

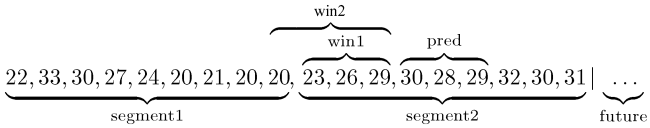


Fig. 8. Window adaptation when analysis spans both segments. Note: **win1** and **win2** have contracted to include less inappropriate data.

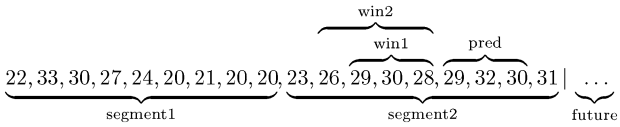


Fig. 9. Window adaptation when analysis lies entirely inside the second segment. Note: the larger analysis window **win2** is likely to have better prediction accuracy because it includes a greater number of appropriate data.

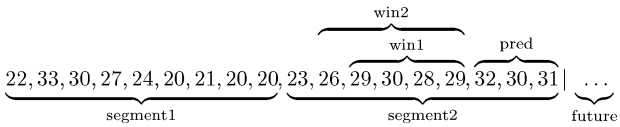


Fig. 10. Window adaptation when analysis lies entirely inside the second segment. Note: **win1** and **win2** have expanded to include a greater number of appropriate data.

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. Figs. 7 and 8 show this phenomenon. In Fig. 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 window sizes 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 Fig. 8, this contraction is shown.

After the data analysis window slides past the end of the first segment, it is likely to expand again to encompass a greater number of appropriate data. Figs. 9 and 10 depict this expansion.

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 window size is likely to expand. When the underlying process shifts (i.e., the analysis window spans more than one segment), the window size is likely to contract. The following section discusses how

the DyFor GP model can retain and exploit knowledge of previously encountered environments.

C. Retaining and Exploiting Knowledge From Past Environments

A primary objective of time series forecasting is to find a model that accurately represents the current environment and use that model to forecast the future. As discussed in Section II-B, existing forecasting methods rely, to some degree, on human judgment to designate an appropriate analysis window, that is the window of historical data whose underlying process corresponds to the current environment and is valid for forecasting future data. If a time series is produced in a nonstatic environment, frequently only the recent historical data that correspond to the current environment are analyzed and historical data that come from previous environments are ignored.

What if the current environmental conditions resemble those of a prior environment? In such a case, knowledge of this prior environment might be used to capture the current environment with greater speed and/or accuracy than a search that ignores this knowledge. Existing forecasting methods, assuming that the analysis window has been correctly set, do not benefit from knowledge of past environments and, thus, must search for a model of the current environment "from scratch." The sliding window feature (described in Section III-A) allows the DyFor GP to analyze all historical data and take advantage of knowledge gleaned from previously encountered environments, giving the model search a "head-start." This knowledge comes in the form of adaptations (i.e., solution subtrees) gained by evolution through these previous environments. Past-evolved subtrees are used by the DyFor GP as promising exploration points from which to search for a model that is appropriate for the current environment. These subtrees are retained and exploited in two ways:

1) implicitly by the evolutionary process when it is coupled with the sliding window feature of the DyFor GP;
2) explicitly through the use of "dormant" solutions.

The following two sections discuss how past-evolved subtrees are maintained and utilized. For the remainder of this paper, we will refer to such subtrees as "adaptations."

D. Implicit Adaptation: The Role of Introns

In biology, unexpressed genotypic regions are commonly called introns. For GP, this term has been adopted to refer to inactive regions in the solution representation, that is subtrees of a solution which do not affect its fitness [4], [12]. Consider the solution tree depicted in Fig. 11. This solution tree represents the expression

$$2.53(x_{t-2})^2 + \left(\frac{2.53x_{t-3}}{x_{t-2}}\right)(x_{t-1} - x_{t-1})$$

which after simplification becomes

$$2.53(x_{t-2})^2.$$

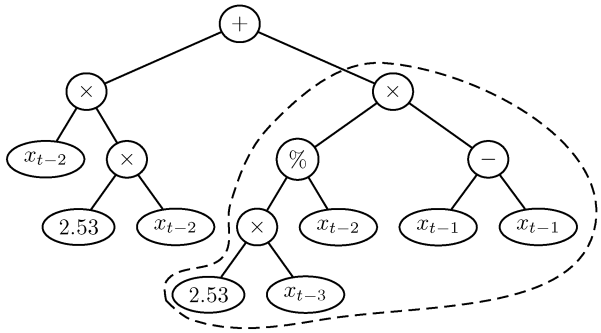


Fig. 11. A GP solution tree containing an intron. Dashed lines enclose the intron subtree.

$$\underbrace{22, 33, 30, 27, 24, 20}_{\text{segment1}}, \underbrace{23, 26, 29, 30, 28, 29}_{\text{segment2}}, \underbrace{31, 32, 30, 30, 31, 28}_{\text{segment3}} \underbrace{\ldots}_{\text{future}}$$

Fig. 12. Time series containing segments with differing underlying processes.

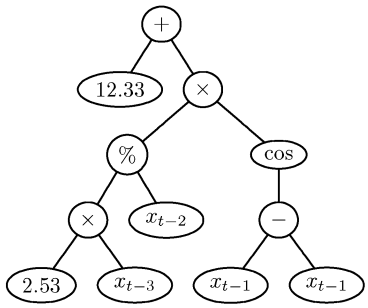


Fig. 13. An evolved solution tree for segment 1.

The % sign in the figure represents a protected division operator that does not allow division by zero. In the figure, the intron subtree (enclosed by dashed lines) does not affect the fitness of the solution as its output simplifies to zero regardless of the values given for variables x_{t-1}, x_{t-2}, and x_{t-3}.

A well known characteristic of the GP process is the tendency for evolved solution trees to have introns make up a significant percentage of the tree structure. This was first recognized by Koza [49, p. 7]. Several studies have suggested that introns are a beneficial component in the evolutionary search for optimal solutions [35], [54], [69]. Introns are seen as particularly valuable when the environment is nonstatic [54]. To understand how introns are utilized to retain and exploit previously learned adaptations in a nonstatic environment, consider the following example.

As discussed in Section II-B, time series arising from real-world circumstances may contain segments with differing underlying data generating processes. For example, a time series representing the monthly value of a U.S. treasury bond might be produced by one underlying process when interest rates are "high" and a different underlying process when interest rates are "medium" or "low." Furthermore, many time series are produced in cyclical environments in which conditions currently in effect are similar to conditions encountered in the past. For example, the current underlying process for a treasury bond time series if interest rates are "high" might be similar to a past underlying process that occurred when interest rates were also "high."

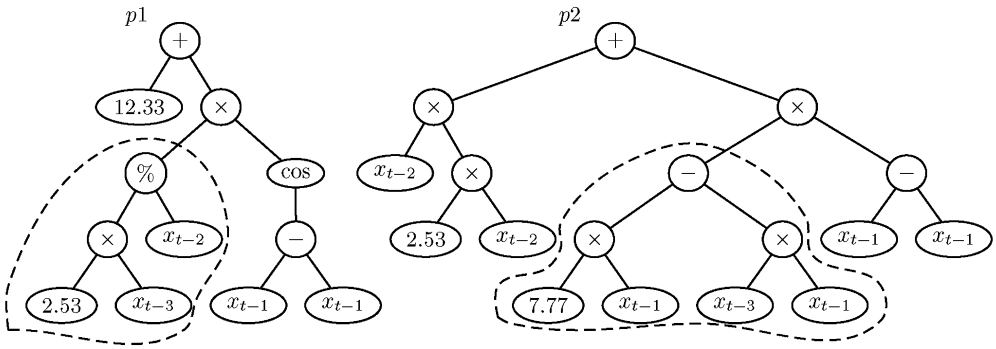


Fig. 14. Retention of a no longer suitable adaptation via crossover. $p1$ and $p2$ are parent solution trees to undergo crossover. Dashed lines enclose subtrees to be exchanged.

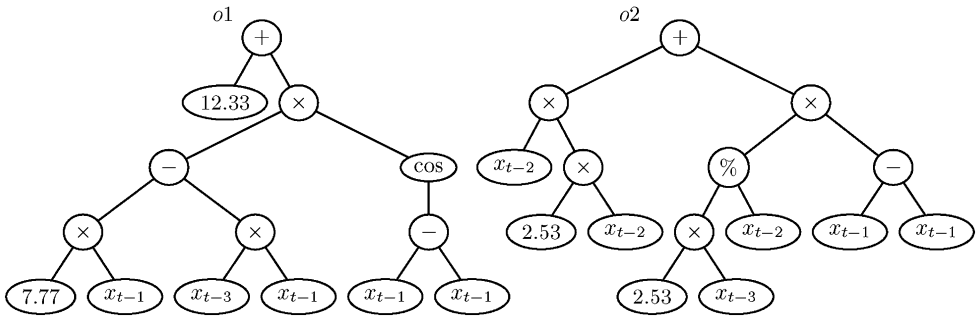


Fig. 15. Retention of a no longer suitable adaptation via crossover. $o1$ and $o2$ are offspring solutions produced after crossover is performed on $p1$ and $p2$ from Fig. 14.

Suppose the time series given in Fig. 12 is such a time series. As depicted in the figure, this time series consists of three segments each with a different underlying data generating process. The third segment's underlying process represents the current environment and is valid for forecasting future data. The first and second segments' processes represent older environments that no longer exist but may contain information that can be used to more accurately capture the current environment. Suppose further that similar environmental conditions produce segments 1 and 3 (e.g., interest rates in the "high" category), while differing conditions produce segment 2 (e.g., interest rates in the "medium" category). The aim is to retain adaptations learned from segment 1 and utilize these adaptations to find an appropriate model for segment 3. The DyFor GP sets its analysis window at the beginning of segment 1's data and starts the evolutionary process in search of an applicable model. Perhaps, after several dynamic generations inside segment 1, the solution tree of Fig. 13 is evolved as a befitting model. This solution tree represents the expression

$$12.33 + \left(\frac{2.53x_{t-3}}{x_{t-2}}\right)(\cos(x_{t-1} - x_{t-1}))$$

which simplifies to

$$12.33 + \left(\frac{2.53x_{t-3}}{x_{t-2}}\right). \tag{4}$$

In the figure, suppose the subtree rooted by the protected division operator (%) is an adaptation that fits the environmental conditions of segment 1. This subtree is equivalent to the second term of (4).

When the DyFor GP's analysis window moves into segment 2, this adaptation is no longer suitable as the environmental conditions have changed. Nevertheless, through crossover, this adaptation can be retained by becoming a part of an intron subtree in a fit solution for segment 2. Figs. 14 and 15 illustrate this phenomenon. In Fig. 14, the adaptation equivalent to the second term of (4) is part of tree $p1$ and is to be exchanged with a subtree of tree $p2$. Fig. 15 gives the offspring solution trees produced. In the figure, the adaptation is now a part of offspring solution tree $o2$. Furthermore, the adaptation is contained in an intron subtree of this offspring (the same intron subtree, as depicted in Fig. 11).

Thus, the adaptation evolved during analysis of segment 1 can be retained as the DyFor GP analyzes segment 2 even though this adaptation is not relevant for segment 2's environment. While this retention of a previously learned adaptation may be possible, one may ask if it is likely. Given that the adaptation in question suits the environment of segment 1, the evolutionary process is likely to produce many solutions containing the adaptation when the DyFor GP analyzes segment 1's data. When the analysis window switches to segment 2's data to start analysis of this new environment, natural selection will tend to favor these fitter solutions from segment 1 and, thus, solutions with this adaptation will be chosen for crossover many times. Therefore, it is likely that a number of those crossovers will result in the adaptation being moved to an intron subtree, as described in the above example, especially given the fact that a large percentage of GP solution trees are made up of

introns. Hence, retention of past-evolved adaptations into intron subtrees is likely.

When the analysis window slides to segment 3's data, it is likely that some solution trees in the population contain the adaptation evolved from segment 1 as a part of an intron subtree. Since segment 3's environment resembles that of segment 1, solutions that contain the adaptation in an active subtree will survive and thrive. Just as crossover can move an adaptation from an active subtree to an intron subtree, it can also move an adaptation from an intron subtree back to an active one. If even one crossover results in such an exchange, natural selection will favor the resulting solution and that solution will multiply.

The above example illustrates how evolved adaptations from previously encountered environments can be retained in nonapplicable environments by becoming part of intron subtrees, and can then be reactivated in applicable environments by moving back to active subtrees. This takes place implicitly via the evolutionary process when it is coupled with the sliding window feature of the DyFor GP. The following section discusses an explicit method of maintaining and exploiting past-evolved adaptations through the use of "dormant" solutions.

E. Explicit Adaptation: Dormant Solutions

The DyFor GP also contains a feature that explicitly saves evolved adaptations from past environments, and then injects them back into the evolutionary process when conditions are suitable. This feature involves the use of "dormant" solutions, that is solutions that remain inactive during environments with inapplicable conditions, becoming active only when applicable conditions arise. Section III-B explains how the DyFor GP adapts the size of its analysis window dynamically. It is noted that when the analysis window lies entirely inside a segment of historical data generated by a single underlying process, the window is likely to expand to encompass a greater number of appropriate data. Conversely, it is shown that when the analysis window spans data generated by more than one underlying process, the window is likely to contract to include a smaller number of inappropriate data. A fortunate side effect of this window size adjustment is that the boundaries of each underlying process can be deduced. Consecutive expansions of the analysis window describe a segment of data with a stable underlying process. Consecutive contractions of the analysis window signal that a shift in environmental conditions has occurred and that a new underlying process is currently coming into effect.

The idea is to save fit solutions evolved during segments when the underlying process is stable to be used later for quicker capture of new environmental conditions when the underlying process shifts. This is accomplished by the following steps.

1) As the analysis window of the DyFor GP slides, note the direction of window adjustment.
   a) N consecutive window expansions are likely to signal the beginning of a stable process. Here, N is a prespecified control parameter of the DyFor GP.
   b) N consecutive window contractions are likely to signal the beginning of a process shift.
2) If a stable process is signaled, save a few fit solutions as potential dormant solutions.
3) For each further window slide in which expansion is observed, replace the potential dormant solutions previously saved with new ones (i.e., fit solutions for the current dynamic generation).
4) When a process shift is signaled, the most recently saved potential dormant solutions become actual dormant solutions and are saved permanently.
5) Now, because a process shift is in effect, inject all dormant solutions saved from previous environments (with the exception of those saved from the most recent previous environment) into the GP population to compete with current solutions. Injected dormant solutions that contain adaptations applicable to the current environment will survive and thrive, while those that do not will die off.
6) Keep injecting these dormant solutions at each window slide until a stable process is again signaled. Once a stable process has been signaled, go back to step 2).

Thus, fit solutions evolved from segments where a stable process exists are saved permanently as dormant solutions representative of the environments from which they evolved. These dormants are the end product of multiple dynamic generations and, therefore, contain adaptations appropriate for their environment. Later in the analysis, when the DyFor GP moves to newer environments, these dormants prove useful if the new environmental conditions resemble those of a previous environment. As described in the above steps, when a new environment is encountered, all dormants are injected into the GP population to compete with current solutions. If the new environment is similar to a past environment, the dormants representing that past environment will contain adaptations suitable for the new environment and, thus, will endure and prosper. In this way, knowledge of past environments can be used to capture the current environment with greater speed and/or accuracy.

The above sections have described several features of the DyFor GP which are designed to allow for the forecasting of time series produced in nonstatic environments. The features discussed include a sliding window of analysis, automatic window size adjustment, and the utilization of knowledge (in the form of evolved adaptations) from previously encountered environments as an aid to forecasting the current environment. These features combine to afford the DyFor GP the following advantages for real-world forecasting concerns which tend to be nonlinear, nonstatic, and not well-understood.

1) As the DyFor GP is based on the GP paradigm, it is not necessary to designate the functional form of the forecasting model in advance and, thus, a befitting (and often nonlinear) model can be automatically discovered.
2) In a nonstatic environment with varying underlying data generating processes, an appropriate data analysis window for the currently existing environment may be found automatically.
3) All available historical data are analyzed, allowing the DyFor GP to learn from past environments and exploit this knowledge when forecasting the current setting.
4) As the features of the DyFor GP are dynamic in nature and can adjust themselves automatically depending on the

environment encountered, the DyFor GP has the potential to forecast time series produced by nonstatic environments with varying data generating processes.

F. The Problem of Bloat

Bloat in GP is the tendency for solution trees to grow large as they approach the optimal [50], [51]. 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 [7]. In the case of the DyFor GP, bloat can be even more severe as generations are run on several consecutive data analysis windows making the total number of generations large. Preliminary experiments employing the DyFor GP confirm this sentiment. In order to allow the DyFor GP to search efficiently for an appropriate forecasting model, bloat must be minimized without sacrificing the quality of solutions. Two methods are proposed to overcome this obstacle:

1) natural nonstatic population control;
2) dynamic increase of diversity.

As described in Section II-B, GP models evolve a population of solutions for a given problem. Thus, a GP model must contain some method to control the number and size of solutions in any population. The standard method of GP population control is due to Koza [49] 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 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 to allow 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. Instead, the population cardinality varies as fitter organisms occupy more available resources and weaker organisms make do with less. Thus, from generation to generation, the population cardinality changes depending on the quality and type of individual organisms present. The proposed natural nonstatic 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;
3) allowing the population cardinality to vary naturally based on the makeup 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 of the DyFor GP, 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 [83] provide a study comparing a GP forecasting model with NNPC to one with the standard population control (SPC) method introduced by Koza [49]. 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.

An important issue in GP is that of how diversity of GP populations can be achieved and maintained. Diversity refers to nonhomogeneity of solutions in a population [54]. A population that is spread out over the search space has a greater chance of finding an optimal solution than one that is concentrated in a small area of the search space. The significance of this concern is recognized in [8], [37], and [87]. As described above, NNPC can be utilized by a GP forecasting model to conserve computer resources. However, although resource usage is controlled, after several generations such a model tends to have populations that are dominated by a small number of bloated solutions. This lack of population diversity affects a model's ability to adapt to changing environments. Even in models that do not employ NNPC (instead opting for SPC), populations tend to have bloated solutions (in this case, a large rather than small number of them). Additionally, bloated solution trees tend to hurt a GP model's generality [46], [69], [71], [78]. Generality refers to a solution's applicability to a wider set of cases than the set presented to the GP model for analysis. For forecasting tasks as well as most other applications, a GP model is presented with some number of input data to analyze in the hopes that solutions evolved using this data will be relevant (i.e., to generalize) to other data not used as input. For these reasons, it is important for a GP model to both reduce bloat and maintain population diversity.

A method that dynamically increases the diversity of a DyFor GP population is proposed to accomplish these objectives. The dynamic increase of diversity (DIOD) method increases diversity by building a new population before each dynamic generation using evolved components from the previous dynamic generation. The following steps outline this procedure.

[4]Koza [49] used population sizes of 500, 1000, and 2000 and maximum solution tree depth of 17 in his early GP experiments.

1) An initial population is constructed (using randomly generated trees in the usual way) for the first dynamic generation.
2) After the dynamic generation is completed, a new initial population is constructed for the next dynamic generation that consists of two types of solution trees:
   a) randomly generated solution trees;
   b) solution trees that are subtrees of fitter solutions from the last population of the previous dynamic generation.
3) The previous step is repeated after each successive dynamic generation.

Thus, each new dynamic generation after the first starts with a new initial population whose solution trees are smaller than those of the last population of the previous dynamic generation but have not lost the adaptations gained from past dynamic generations. In this way, solution tree bloat is reduced without harming the quality of solutions. Additionally, because randomly generated trees make up a portion of the new population, diversity is increased.

Section III-E describes an explicit technique for utilizing adaptations learned from previously encountered environments in order to better forecast the current environment. This technique employs dormant solutions saved from past environments. When this technique is incorporated as a part of the DyFor GP, the steps of DIOD above must be modified slightly to allow for the injection of dormant solutions into the GP population at opportune times. The modified steps are as follows.

1) An initial population is constructed (using randomly generated trees in the usual way) for the first dynamic generation.
2) After the dynamic generation is completed, a new initial population is constructed for the next dynamic generation that consists of two types of solution trees:
   a) randomly generated solution trees;
   b) solution trees that are subtrees of fitter solutions from the last population of the previous dynamic generation.
3) After each successive dynamic generation in which a stable process (as described in Section III-E) is in effect, a new initial population for the next dynamic generation is constructed in the same way as given in the previous step.
4) After each successive dynamic generation in which a process shift (as described in Section III-E) is in effect, a new initial population is constructed for the next dynamic generation that consists of three types of solution trees:
   a) the two types of solution trees listed in step 2;
   b) solution trees that are subtrees of dormant solutions saved from previously encountered stable environments that are further in the past than the most recent stable environment.

As seen from the above steps, when a process shift occurs, dormant solutions are used to contribute adaptations evolved from past environments. If any of these dormants contain adaptations relevant to the current environment, then solutions receiving these adaptations (via crossover) will prosper.

The following section discusses two complementary concerns that have a potentially important impact on the performance of the DyFor GP model.

G. Forecast Combination and Fitness Measures

In many forecasting situations, the "best" forecasting model is not known and, thus, several "good" forecasting models are developed. A forecaster is then faced with the problem of choosing a single forecast from a set of several candidate forecasts produced by each of the forecasting models employed. Many times it is better not to choose just one forecast from the set, but, instead, use some procedure to combine multiple forecasts into one. This issue is called forecast combination and its relationship to the DyFor GP model is discussed below.

Evolution-based techniques such as the DyFor GP use Darwin's principle of "survival of the fittest" and sexual recombination to solve complex, real-world problems. For these kinds of methods, some fitness measure or fitness function is 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. The question of selecting a good fitness measure for the DyFor GP model is addressed in this section as well.

The GP algorithm is essentially a fitness-driven random search. When GP is applied to forecasting complex, nonlinear time series, the search space is the set of all possible mathematical equations that can be constructed using specified operands (explanatory variables) and mathematical operators. This space is quite large and, in general, intractable for most 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 20–100) and designate the forecasts of the best run as the result (see, for example, [40] and [41]). In the real-world, this practice is not useful since one cannot know which run produces the best forecast for a given time period without first knowing the corresponding actual value of that time period.

The DyFor GP model is based on the GP algorithm and, thus, it is necessary to execute a set of DyFor GP runs for any forecasting task. Therefore, at any given time period, there is a set of multiple forecasts to choose from. Here, it becomes necessary to apply some forecast combination method to produce a single useful forecast from the set. To be useful in a real-world setting, the forecast must be generated using an out-of-sample methodology where no data beyond the point of forecast is utilized for analysis, model construction, or forecast combination.

The study of forecast combination has a long history. Mathematicians, economists, and researchers from the data mining community, among others, have developed many combining methods. Combining methods can generally be divided into to two groups, variance-covariance methods and regression methods [21]. In variance-covariance methods, the combination model is

$$F = \alpha_1 f_1 + \alpha_2 f_2 + \cdots + \alpha_n f_n \tag{5}$$

where F is the combined forecast, $f_1, f_2, \ldots, f_n$ are the single forecasts to be combined, and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are corresponding weights subject to the condition that their sum is one. Optimal

weights for this equation are estimated by minimizing the variance of past forecast errors.

For regression methods, the following combination model is used:

$$F = \beta_0 + \beta_1 f_1 + \beta_2 f_2 + \cdots + \beta_n f_n + e \tag{6}$$

where F and $f_1, f_2, \ldots, f_n$ have the same meaning as they do in (5), β_0 is a constant, $\beta_1, \beta_2, \ldots, \beta_n$ are regression coefficients, and e is an error term. Here, the coefficients are estimated by regressing actual time series values on past forecasts for those values.

Variations on these combining methods are numerous and a discussion of these variations can be found in Diebold [21, pp. 347–365]. Some examples of recent studies which focus on forecast combination include [10], [20], [23], [31], and [90].

Since forecast combination is not the focus of this study, we restrict our attention to simple, well-known combining techniques. Thus, the following procedure is selected for combining multiple DyFor GP forecasts produced by a set of multiple DyFor GP runs into a single, out-of-sample forecast.

1) For the first forecast, designate the median forecast of the set as the single forecast to be used.
2) For all remaining forecasts, repeat the following.
   a) Compare the previous forecast of all runs to the actual data for that (already past) time period and rank each run based on its accuracy.
   b) Select the current forecast of the top three runs from this ranking, compute the average of these three forecasts, and designate this average as the single forecast to be used.

This combining procedure is a form of the variance-covariance method described above in which only the most recent past forecast of each run is considered when estimating the combining weights and the weight assigned to forecasts of the three top-ranked runs $= (1/3)$.

As mentioned earlier in this section, choosing the fitness measure to be employed by the DyFor GP model is of great importance. Most GP forecasting applications use a mean squared error (MSE) fitness measure for model evolution. To date, there have been no significant studies investigating alternative fitness measures for GP forecasting applications. One alternative fitness 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 fitness measure to employ would depend upon the characteristics of the time series to be forecast.

Another interesting possibility is to develop a new "combined" fitness (CF) measure that incorporates aspects of both the MSE and MAD measures. The purpose behind the CF measure is to minimize the effect of noise, while still being reactive to shifts in a time series. It tries to accomplish this by making a compromise between MSE (which is preferable for shifts) and MAD (which is preferable for minimizing noise). This CF measure requires a user-specified parameter, Ω, as a threshold between noisy data and non-noisy data. Fig. 16 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 Ω, CF measure values follow those of the squared error. However, when the relative error falls outside of the Ω threshold, CF measure values follow those of the absolute deviation.[5]

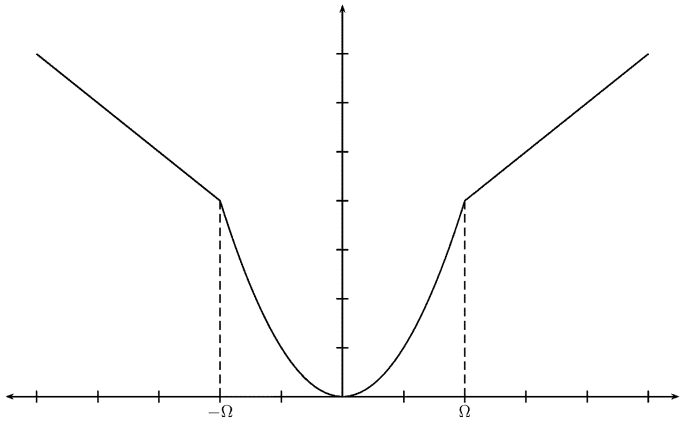


Fig. 16. The CF measure as a function of the relative error.

The DyFor GP model could potentially use any one of the above fitness measures for a given forecasting experiment. Therefore, in order to be applicable over a diverse range of forecasting tasks, the DyFor GP model includes a parameter specifying which of these three fitness measures should be employed during a run.

The following two sections present experiments conducted using the DyFor GP model.

IV. DyFor GP Experiments

In order to test the DyFor GP model, a number of forecasting experiments using both simulated and actual time series data were undertaken. The purpose of these experiments is twofold:

1) to compare the performance of the DyFor GP model (both the "full" version with dormants and a "partial" version without dormants) to that of a conventional GP model;
2) to compare the performance of the DyFor GP model to that of other leading models from benchmark studies.

The experiments are, thus, grouped into two subsets according to these objectives and are discussed in the following two sections. The experiments described below represent an extension of several preliminary experiments made on early versions of the DyFor GP model that were reported in [84] and [85].

A. Comparing DyFor GP to Conventional GP

One subset of experiments is concerned with comparing the performance of the DyFor GP model to that of a conventional GP model. It is also desirable to examine how the inclusion of

[5]The specified value of Ω shown in the figure is for purposes of illustration.

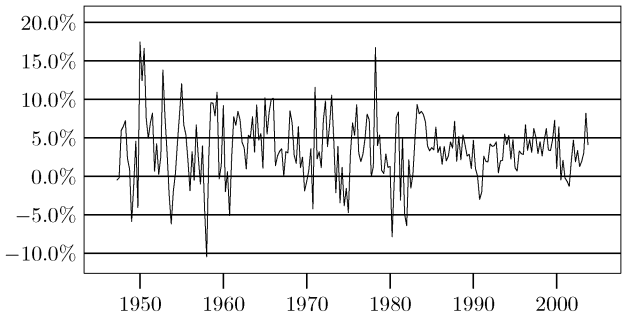


Fig. 17. Gross domestic product (growth): 1947–2003.

dormant solutions affects the DyFor GP model's efficiency. Two time series were chosen for these experiments, one of simulated data and one of real data.

The simulated time series is constructed by concatenating three segments, each segment being a small time series generated by a known process. The first and third segments are generated by similar (but not equivalent) processes, while the second segment is generated by a different process. Equation (7) gives the underlying process used to generate the entire time series. Note that this process is a step function defining each of the three segments

$$f(x) = \begin{cases} \sin(x) + \sqrt{x} & \text{for } 1 \leq x \leq 20 \text{ (segment 1)} \\ e^x + 2 & \text{for } 21 \leq x \leq 40 \text{ (segment 2)} \\ \sin(x) - \sqrt{x} + 22 & \text{for } 41 \leq x \leq 60 \text{ (segment 3)} \end{cases} . \tag{7}$$

The time series is constructed using 60 total values, 20 for each segment. Thus, the first 20 values correspond to segment 1 and are generated by evaluating this function for integer values of $x = 1 \ldots 20$, the next 20 values correspond to segment 2 and are generated by evaluating this function for $x = 21 \ldots 40$, and the final 20 values correspond to segment 3 and are generated by evaluating this function for $x = 41 \ldots 60$. This artificial time series is designed to mimic a series whose underlying process varies over time. Additionally, the similarity of segments 1 and 3 is intended to mirror cyclical behavior that may occur in time series produced by dynamically changing conditions.

The real-data time series chosen for experimentation is the U.S. Gross Domestic Product (GDP). According to the U.S. Department of Commerce [81], 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 nation's economy. The GDP series was selected because it is a widely studied, nonlinear time series with a well-known set of explanatory variables. Fig. 17 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 [47] 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 1995Q1–2003Q1.

The following two sections describe the setup of both experiments and give observed results.

1) Test Setup: For this subset of experiments, three forecasting models are compared: full-version DyFor GP (i.e., with dormants), partial-version DyFor GP (without dormants), and conventional GP. As discussed in Section III-G, the GP process is a stochastic one and, thus, it is necessary to execute a set of GP runs rather than just a single run. $\text{Setsize} = 20$ is used for all GP experiments executed here.

For the simulated time series experiment, values for x in (7) are utilized by all models as inputs and the outputs generated are one-step-ahead forecasts for $f(x)$. The first 14 time series data are used for initial training and then 46 one-step-ahead forecasts are generated that correspond to actual time series values beginning at value #15 and ending at value #60.

For the GDP experiment, 29 of the 30 economic indicators listed in the Kitchen and Monaco study [47] are utilized as inputs[6] by all 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 1951Q3 is used for initial training and one-step-ahead forecasts for 1995Q1–2003Q1 are produced.

The GP process employs the elements of a terminal set and a function set as building blocks from which to construct forecasting models. For both experiments, the terminal set consists of the inputs listed above (x for the simulated data experiment and the 29 indicators for the GDP experiment) and a random constant, while the function set consists of operators $+$, $-$, $\times$, $\div$, sin, cos, square root, exp, and ln.[7]

In both experiments, DyFor GP forecasts are generated in a "real-time" fashion, that is, after the DyFor GP model produces the first forecast, the analysis window is slid to incorporate the actual data for that time period, analysis continues, and then the DyFor GP produces the second forecast. This procedure is continued for each forecast until all required forecasts have been generated. It should be emphasized that for both experiments all forecasts are generated using an out-of-sample methodology where no data beyond the point of forecast is utilized for analysis or model construction.

The DyFor GP model 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 the DyFor GP model. Table I gives the general GP parameter values used by all competing models, while Table II lists parameters values that are used only by the full and partial version DyFor GP models.

All parameter values listed in Table I were selected to match those used by Koza [49] for his experiments in symbolic regression[8] with the following exceptions.

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

[7]Operators $\div$, square root, exp, and ln are protected from undefined or unbounded behavior as is done in experiments conducted by Koza [49].

[8]"Symbolic regression" is the term Koza uses to describe the search for a mathematical expression that closely fits a given finite sample of data. In many cases, this is equivalent to the task of time series forecasting.

TABLE I
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	MSE

TABLE II
SPECIFIC DYFOR GP PARAMETER SETTINGS

Parameter	Value (Sim. Data Experiment)	Value (GDP Experiment)
population size	variable	variable
max. solution tree depth	none	none
soft node limit	20000	35000
hard node limit	25000	38000
no. training dyn. gens.	1	121
windowslide increment	1	1
max window size	14	80
min window size	2	40
start window size	4	54
window difference	6	12
window adj. step size	1	1
N	2	3

1) The "max. no. of generations" parameter has a slightly different meaning when applied to the DyFor GP model. For DyFor GP it means the maximum number of generations used for one dynamic generation, that is a set of generations run on a single analysis window. Since the DyFor GP model executes many dynamic generations over the course of a single run, this parameter is reduced from 51 to 41 generations to decrease computation time.
2) Elitism (reproduction of the best solution of the population) is used.
3) 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 [61] and 2) decreasing the reproduction rate to zero was not thought to harm the effectiveness of the evolutionary process since elitism is used.

Since the DyFor GP employs a nonstatic population control method,[9] there are no specifications for population size or maximum solution tree depth in Table II. Instead, specifications for the soft and hard node limits are used and shown in the table. As discussed in Section III-F, conventional GP usually employs a static population size and a limit for solution tree depth. For these experiments, however, conventional GP is executed using variable population size with the same parameters values, as shown in Table II. This is done for two reasons: 1) to prevent runs from being prematurely aborted due to the presence of numerous bloated solutions and 2) to reduce computation time by placing a limit on the total number of nodes allowed in a population.

In Table II, parameter "no. training dyn. gens." means the number of dynamic generations executed before producing the first forecast and parameter "window slide increment" means the number of newer (more recent) historical data to incorporate at each slide of the analysis window. For the GDP experiment, 121 training dynamic generations means the analysis window slides through 30 years of historical data (4 slides per year, 1 dynamic generation per slide + 1 initial dynamic generation).

[9]This population control method is described in Section III-F.

TABLE III
SIMULATED DATA FORECASTING RESULTS

Forecasting Model	mean MSE	std. dev.
conventional GP	40.04	0.25
DyFor GP (w/o dormants)	2.96	3.49
DyFor GP (w dormants)	1.88	2.31

TABLE IV
GDP FORECASTING RESULTS

Forecasting Model	mean MSE	std. dev.
conventional GP	4.96	1.62
DyFor GP (w/o dormants)	4.28	0.77
DyFor GP (w dormants)	4.01	0.57

The "max window size" and "min window size" parameters in the table specify the maximum and minimum analysis window sizes, respectively. For the GDP experiment, values of 80 and 40 correspond to max/min analysis window sizes of 20 and 10 years, respectively. As described in Section III, the adjustable window size feature of the DyFor GP model calls for using two analysis windows of differing sizes. Parameter "start window size" refers to the initial window size setting of the smaller of the two windows and parameter "window difference" refers to the size difference between the larger and the smaller window. For the simulated data experiment, this means that initial window sizes of 4 and 10 are used, while in the GDP experiment initial window sizes of 54 and 66 are used which correspond to 13.5 and 16.5 years of GDP data, respectively. Parameter "window adj. step size" gives the adjustment amount to use when adjusting the size of the windows. Parameter "N" gives the number of consecutive window expansions or contractions that signal a stable process or a process shift, respectively.

2) Results: As mentioned in the previous section, a set of runs is executed $(\text{setsize} = 20)$ 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. Tables III and IV give the observed results for the simulated data and GDP experiments, respectively.

The tables reveal some interesting results. First, in both experiments the DyFor GP models outperform conventional GP, significantly so in the case of the simulated data experiment. Second, DyFor GP with dormants provides some performance improvement over DyFor GP without dormants for both experiments.

The DyFor GP models' superior performance over conventional GP may be due to its adjustable analysis window which can allow the model to better "hone in" on currently relevant data in a dynamic environment. For the simulated data experiment, the advantage is marked because the shifts in the underlying process occur abruptly, while for the GDP experiment, the advantage is less noticeable, probably because process shifts

TABLE V
SIMULATED DATA FORECASTING RESULTS (SEGMENT 3 ONLY)

Forecasting Model	mean MSE (all segment 3)	mean MSE (first 5 points)
DyFor GP (w/o dormants)	5.89	21.70
DyFor GP (w dormants)	2.07	2.80

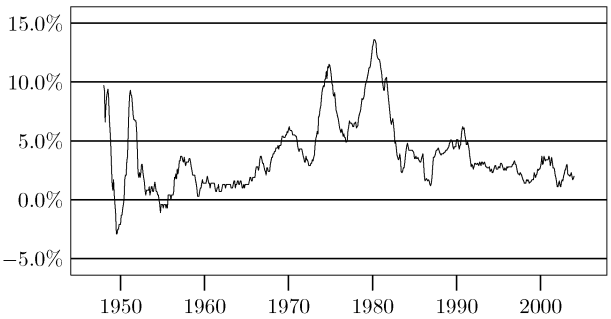


Fig. 18. CPI Inflation: 1948–2003.

occur in a smoother manner. The inclusion of dormants in the DyFor GP model provides some gain in forecasting efficacy, albeit small.

To further compare the two DyFor GP models, we can narrow the focus to examine only forecasts that correspond to segment 3 of the simulated data experiment. Recall that segment 3's underlying process is similar to that of segment 1 but different from segment 2's process. Thus, if dormants are effective, then a DyFor GP model that uses them should have better segment 3 forecasts than a DyFor GP model that does not use them. Table V shows the performance of the two DyFor GP models for segment 3 of the simulated data experiment. The table reports the mean MSE of segment 3 forecasts over all 20 runs. It also reports the mean MSE (over all 20 runs) of forecasts that correspond only to the beginning of segment 3 (the first five values) rather than the entire segment.

The second column of the table shows that the use of dormants provides more accurate forecasts over the entire segment 3. The third column shows that when DyFor GP analysis enters segment 3 (i.e., the underlying process shifts from segment 2's process to that of segment 3), the use of dormants provides for quicker capture of this new process.

The following section describes experiments that compare the DyFor GP model to leading models from benchmark studies.

B. Comparing DyFor GP to Benchmark Models

Two real-data forecasting tasks were selected for this subset of experiments, forecasting the U.S. GDP and the U.S. Consumer Price Index (CPI) Inflation rate. Fig. 18 gives a graphical depiction of the monthly CPI Inflation rate time series (a GDP graph is shown in Fig. 17). In the figure, CPI Inflation is calculated as a year-over-year percent change.

These two forecasting experiments were chosen because both the U.S. GDP and CPI Inflation series are widely studied, nonlinear time series with well-known sets of explanatory variables. Such characteristics are conducive to preparing a DyFor GP experiment and comparing DyFor GP results to those of leading studies.

The following three sections describe the GDP and CPI Inflation benchmark models, detail the setup of both experiments, and give observed results.

1) GDP and CPI Inflation Benchmarks: As discussed in Section IV-A, the model designed by Kitchen and Monaco [47] forecasts the quarterly GDP series using several monthly indicators. The idea is to produce a single, one-step-ahead, quarterly GDP forecast by incorporating the latest monthly indicator values and aggregating their effects. For example, if the national unemployment rate and unemployment insurance claims are selected as (monthly) economic indicators and their latest announced values are for the month of January, then the forecasting model incorporates these latest values and aggregates them to produce a forecast for the current quarter (quarter 1 or Q1). When indicator values for February are announced, the model incorporates them to produce an updated forecast for Q1. Thus, a "real-time" GDP forecast for the current quarter can be constructed and updated as soon as new data become available.

The real-time forecasting system (RTFS) of Kitchen and Monaco [47] makes use of 30 monthly economic indicators as explanatory variables. As mentioned in Section IV-A, these indicators are derived from various economic sectors including employment, financial, survey, and production and sales. A linear regression model is used to relate an indicator to GDP growth

$$y_t = \alpha + \beta(L)x_t + e_t \tag{8}$$

where y_t is the real GDP growth for quarter t at an annualized rate, x_t is an indicator, $\beta(L)$ is a set of coefficients for current and lagged values of the indicator, and e_t is an error term. While (8) theoretically may include numerous indicator lags, Kitchen and Monaco choose zero or four lags and use the Schwarz criterion[10] to determine which results the RTFS should utilize. Each indicator has three separate regression models relating it to GDP growth, one for each (monthly) period of a quarter. When a new month's data for an indicator becomes available, the appropriate regression model is selected and used to produce a forecast for GDP growth that is based only on that indicator. This is repeated for all indicators. Then, all of these single-indicator GDP forecasts are aggregated into one to yield a GDP forecast. RTFS generates one-step-ahead forecasts in a "real-time" fashion, that is each time new data becomes available, the model incorporates this data and produces a new forecast. All RTFS forecasts are made using an out-of-sample methodology where no data beyond the point of forecast is used for model fitting.

The RTFS is used to generate quarterly GDP forecasts when one month, two months, and three months of indicator data are available, respectively. These results are compared with those produced by a linear autoregressive (AR) forecasting model with four lags. Historical data dating back to 1982Q1 is used for analysis and one-step-ahead GDP forecasts are generated for an eight-year range starting with 1995Q1 and ending with 2003Q1. The results of the Kitchen and Monaco study show that the RTFS model outperforms the AR model by a large margin.

The U.S. CPI Inflation rate is a highly scrutinized economic concern with considerable national impact. The inflation time series has monthly frequency and available historical data exists

[10]The Schwarz criterion is defined in [21, p. 26].

dating back to 1947. The Phillips Curve is a bivariate linear forecasting model that is widely considered as a consistent and accurate predictor of U.S. inflation. Stock and Watson [76] provide a recent study that re-investigates the efficacy of this model, both in its conventional form and in several alternate forms that include various macroeconomic variables. The conventional Phillips Curve specification used in their study is meant to forecast inflation over a 12-month period and is given by the following regression model:

$$\pi_{t+h}^{h} - \pi_t = \phi + \beta(L)u_t + \gamma(L)\Delta\pi_t + e_{t+h} \qquad (9)$$

where $\pi_t^h = ((1200/h)) * \ln(P_t/P_{t-h})$ is the h-period inflation rate $(h = 12), \pi_t = (1200) * \ln(P_t/P_{t-1})$ is the monthly inflation rate, u_t is the unemployment rate, and $\beta(L)$ and $\gamma(L)$ are lag operators specifying 0–11 lags. Alternate Phillips Curve specifications are constructed by substituting the unemployment rate u_t of (9) with other macroeconomic variables or indices.

Historical CPI Inflation data dating back to January 1959 are used for analysis and 12-month horizon forecasts are generated for the period of January 1970–September 1996. Forecasting results are presented for two subperiods, 1970–1983 and 1984–1996. As in Kitchen and Monaco's GDP models, Stock and Watson use an out-of-sample methodology.

The results of the Stock and Watson study show that the Phillips Curve in its conventional form outperforms univariate AR models, as well as most alternative Phillips Curve specifications in which the unemployment rate is replaced by a different economic variable. The alternate specifications that do surpass the conventional one are those that replace unemployment with a measure of aggregate economic activity such as real manufacturing and trade sales or capacity utilization. Stock and Watson also develop a new composite index of 168 economic activity measures using principal component analysis and construct another alternative Phillips Curve specification with this index. This composite-index specification proves to be the best CPI Inflation forecasting model overall.

In a recent survey of the literature on output and inflation forecasting, Stock and Watson [77] note that an effective nonlinear model has not yet been found.[11] For this reason, we restricted our focus to linear benchmark models of real GDP growth and CPI inflation.

2) Test Setup: The DyFor GP model was applied to the GDP and CPI Inflation forecasting experiments detailed above. For the GDP experiment, the same setup described in Section IV-A1 was used. For the CPI Inflation forecasting experiment, the goal is to compare the performance of the conventional Phillips Curve specification with that of the DyFor GP model. Therefore, inputs to the DyFor GP model are the same inputs employed by this conventional specification, namely, the unemployment rate and past values of the monthly inflation rate. Historical CPI Inflation data dating back to 1950:01 is used for analysis and forecasts for 1970:01–1983:12 are produced. The terminal set used consists of the conventional Phillips Curve inputs mentioned above and a random constant, while the function set is the same as the one used for the GDP experiment.

[11]In [9], Bidarkota uses a nonlinear regime switching model to forecast inflation but the results offer no significant improvement over the conventional Phillips Curve.

TABLE VI
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	MSE/MAD/CF

TABLE VII
SPECIFIC DYFOR GP PARAMETER SETTINGS

Parameter	Value (GDP Experiment)	Value (Inflation Experiment)
population size	variable	variable
max. solution tree depth	none	none
soft node limit	35000	35000
hard node limit	38000	38000
no. training dyn. gens.	121	121
windowslide increment	1	1
max window size	80	240
min window size	40	12
start window size	54	120
window difference	12	24
window adj. step size	1	1
N	3	3

In both experiments, single run DyFor GP forecasts are generated in a "real-time" fashion in the same way as detailed in Section IV-A1. As discussed in Section III-G, it is necessary to execute a set of DyFor GP runs. Additionally, if a single forecast is to be generated at each time period, it is necessary to use some forecast combination technique to combine the multiple forecasts produced by the multiple runs into one out-of-sample forecast. The forecast combining method used for these experiments is the one described in Section III-G and the number of DyFor GP runs comprising a set is 20. As in the experiments described in Section IV–A1, all forecasts are generated using an out-of-sample methodology.

Tables VI and VII give general GP parameter values and specific DyFor GP parameter values, respectively. All parameter values listed in Table VI are the same as those used for the experiments of Section IV-A (displayed in Table I) except that two additional fitness measures, MAD and the CF measure detailed in Section III-G, have been added.

Each experiment includes three separate DyFor GP runsets, one using each fitness measure listed. It should be noted that the fitness measures given in Table VI are only used for evolution and are not used to measure the quality of forecasts produced by the DyFor GP model. The CF measure requires a user-specified parameter, Ω, to determine which data are outliers and which are not. For these experiments, Ω is set to a value that is 7.5% of the median level of the time series to be forecast. An optimal value for Ω is not known and, thus, intuition was used to specify this parameter.

The DyFor GP parameter values used in the GDP experiment (shown in column 2 of Table VII) are the same as used for the GDP experiment of Section IV-A. Column 3 of Table VII gives the parameter values used in the inflation experiment. For the monthly CPI Inflation series, parameter "no. training dyn. gens." means the analysis window slides through ten years of

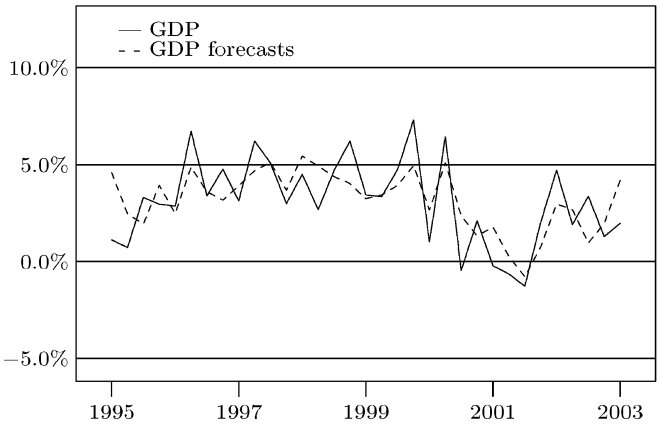


Fig. 19. GDP growth and forecasts produced by the DyFor GP model.

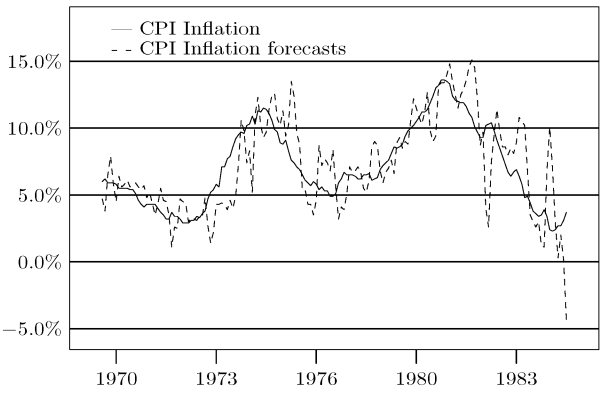


Fig. 20. Annual CPI Inflation and forecasts produced by the DyFor GP model 12 months earlier.

historical data (12 slides per year, 1 dynamic generation per slide + 1 initial dynamic generation) before producing the first forecast. Parameter values for max/min window size in the inflation experiment are 240 and 12 which correspond to window sizes of 20 and 1 years, respectively, and values for the initial smaller and larger window sizes (specified by "start window size" and "window difference") correspond to 10 and 12 years of inflation data, respectively.

3) Results: Tables VIII and IX compare DyFor GP results to those of the benchmark models for the GDP and CPI Inflation experiments, respectively. In the tables, results of three DyFor GP models are shown, one for each of the three fitness measures and the root-mean-squared error (RMSE) of forecasts is reported. Figs. 19 and 20 plot GDP growth and CPI Inflation forecasts produced by the best DyFor GP model with the corresponding actual values of GDP growth and CPI Inflation, respectively.

Tables VIII and IX show that the performance of the DyFor GP model improves upon that of the benchmark models for both experiments. In the CPI Inflation experiment the margin is small, but for the GDP experiment the margin proves larger. Also, the fitness measure employed by the DyFor GP model appears to have an important influence on its performance.

The DyFor GP model's efficient performance in both experiments may be due to its ability to capture nonlinearities present in the GDP and CPI Inflation time series that are not captured

TABLE VIII
GDP FORECASTING RESULTS

Forecasting Model	RMSE
RTFS	1.85
AR	2.46
DyFor GP (MSE fitness measure)	1.87
DyFor GP (CF fitness measure)	1.80
DyFor GP (MAD fitness measure)	1.57

TABLE IX
CPI INFLATION FORECASTING RESULTS

Forecasting Model	RMSE
CPC	2.4
DyFor GP (MSE fitness measure)	2.3
DyFor GP (CF fitness measure)	2.6
DyFor GP (MAD fitness measure)	2.6

by the competing linear models. In the GDP experiment, historical data starting in 1951Q3 is analyzed and forecasts for the 1995Q1–2003Q1 period are produced. In the CPI inflation experiment, historical data starting in 1950 is analyzed and forecasts for the 1970:01–1983:12 period are produced. The behavior of real GDP growth over its forecast horizon is reasonably stable compared with its preceding behavior (mean and standard deviation of 3.26 and 4.06 over the 1951Q3–1994Q4 period versus mean and standard deviation of 2.95 and 2.13 over the 1995Q1–2003Q1 period). Thus, DyFor GP is able to capture and successfully extrapolate real GDP growth. By contrast, the behavior of CPI inflation over its forecast horizon was drastically different from its preceding behavior (mean and standard deviation of 2.16 and 1.95 over the 1950:01–1969:12 period versus mean and standard deviation of 7.13 and 2.97 over the 1970:01–1983:12 period). Thus, DyFor GP is less able to capture and successfully extrapolate CPI inflation. This could be the reason why the DyFor GP's margin of advantage over competitors is smaller for the inflation experiment as opposed to the GDP experiment.

Considering the three fitness measures utilized by the DyFor GP model, the performance ranking order of measures is in reverse order for the two experiments. The GDP experiment gives a ranking order, first to last, of MAD, CF, MSE. The inflation experiment gives a ranking order of MSE, CF, MAD.[12] This may also be explained by the difference in forecast horizon stability between the GDP and inflation series. As discussed in Section III-G, analyses based on MSE are more heavily influenced by the existence of outliers than analyses based on MAD. An outlier datum can represent one of two possibilities: noise or new information representing a process shift. The GDP series is less volatile than the inflation series which may mean that outliers represent noise and should not dramatically affect model construction. Thus, for the GDP case, the MAD measure is the better measure. The inflation series is more volatile and outliers may frequently represent a shift in the underlying process. Therefore, the MSE measure is the most useful because it can more easily track the rapidly occurring shifts of the inflation series. Following this line of reasoning, the CF measure should

[12]The results reported by Stock and Watson carry a precision of one decimal place and, thus, DyFor GP results in Table IX are reported with the same precision. This precision obscures the difference between results produced by DyFor GP models with the CF and MAD fitness measures, respectively.

have median utility for both experiments as it is a combination of MSE and MAD measures.

Other experimental results concerning retention of past adaptations and window behavior proved interesting as well. Forecasting models evolved by the DyFor GP contained adaptations learned from the past. The following describes two examples of this retention.

1) In the GDP experiment the adaptation, e^{prod} ($\text{prod} = \text{industrial production index}$), was evolved as early as 1995 and is retained over the next eight years, showing up in evolved models for 1997, 1998, 1999, 2001, and 2003.
2) In the Inflation experiment the adaptation, $\sin(u_{t-11} - u_t)$ ($u_t = \text{current unemployment rate}, u_{t-11} = \text{unemployment rate 11 months ago}$), was evolved as early as 1976 and is retained over the next seven years, showing up in evolved models for several years up to 1983.

Window adjustment also appeared to have an important effect. In the GDP experiment, the window size was initially set at 16.5 years and the best performing runs generally adjusted their window size to approximately 14 years. In the Inflation experiment, the window size was initially set at 10 years and the best performing runs generally adjusted to approximately 12.5 years.

The DyFor GP model generates forecasts in a real-time fashion, that is, after the first forecast is produced, the analysis window is slid to incorporate the actual data for that time period, analysis continues, and then the second forecast is produced. Thus, the forecasting model changes (evolves) over the course of a forecasting experiment. Usually this means that a new forecasting model is constructed for each forecast. The forecasting models evolved by the DyFor GP often consist of several hundred operators and operands. These evolved models are too large to be displayed in this paper. While complex models, such as those produced by DyFor GP (and GP as well) are hard to understand, our goal here is not to make relationships clear but to forecast well in a dynamic environment. Atheoretical forecasting models whose main purpose is predicting future values are known to be useful [17].

It is worthwhile to discuss computation time and how the DyFor GP model compares with the benchmark models in this regard. Both real-data experiments (GDP and CPI Inflation) require the analysis of several decades of historical data and generate forecasts over approximately a ten-year range. As detailed in Section III, the DyFor GP model runs numerous dynamic generations with each dynamic generation being comprised of numerous (regular) generations run on a single window of training data. Thus, the total number of generations over the entire experiment can be quite large ($\sim$30 000 for the inflation experiment and $\sim$15 000 for the GDP experiment for a single run) which, in turn, makes the computation time required quite large. Moreover, as described in Section III-G, it is necessary to execute not a single DyFor GP run but a set of DyFor GP runs for each experiment. These factors exacerbate the problem of achieving acceptable forecasting accuracy with a reasonable amount of computation time. Some balance must be struck between computation time and the extent of the search. Taking into account the computational resources available[13] and the complexity of the forecasting tasks at hand, the balance arrived at calls for single run execution time of $\sim$72 hours and set ($\text{setsize} = 20$ runs) execution time of $\sim$216 hours. All in all, the total amount of time required to perform experiments was approximately 10–12 weeks.

This is significantly more computation time than is required for the benchmark models. However, even though DyFor GP computation time for these experiments was extensive, businesses might employ the DyFor GP model to conduct similar experiments with considerably less computation time. The reasons for this are the following.

1) The computing environment used to execute these experiments was modest. Larger companies would likely have access to a computing environment with greater power and speed.
2) For true real-world experiments (as opposed to the simulated real-world experiments presented here), the DyFor GP model could be set up so that the analysis of large amounts of historical data takes place only once during some designated preliminary period. This could be realized in the following way.
   a) Set the DyFor GP model's analysis window to the beginning of the historical data and let DyFor GP analysis continue until it has analyzed all data up to the current time period.
   b) Once this preliminary analysis has finished (and a forecast for the next time period is produced), save the current state of the model and wait for new, incoming data to arrive.
   c) When new data arrives, restart the DyFor GP's analysis. Since the most recent state of the model is saved, it is not necessary for the DyFor GP to reanalyze all historical data again, and it can incorporate the new data, continue analysis, and produce a forecast for the next time period with much less computation time than was required to complete the preliminary analysis.[14]

Thus, the DyFor GP model can potentially be used to produce forecasts for real-world concerns that arrive early enough to be useful.

The following section draws conclusions from these experiments and discusses future avenues of exploration.

V. Conclusion and Future Work

In this study, the DyFor GP model is developed and tested for forecasting efficacy on both simulated and real-data time series. Results show that the DyFor GP model improves upon the performance of benchmark models for all experiments. These findings highlight the DyFor GP's potential as an adaptive, nonlinear model for real-world forecasting applications and suggest further investigations. The DyFor GP model presents an attractive forecasting alternative for the following reasons.

1) It is not necessary to specify the functional form of the forecasting model in advance and, thus, a befitting nonlinear model, albeit complex, can be automatically discovered.
2) The DyFor GP is an automatically self-adjusting model. Thus, in a changing environment, it may be able to adapt and predict accurately without human intervention.

[13]Experiments were conducted on a shared IBM p690 cluster with 32 1.3 GHz processors. See [65] for complete details.

[14]For these experiments, the amount of computation time for a single DyFor GP run to incorporate and analyze newly arriving data corresponding to a single time period is approximately 3–5 minutes.

3) It can take advantage of a large amount of historical data. Conventional forecasting models require that the number of historical data to be analyzed be set *a priori*. In many cases this means that a large number of historical data is considered to be too old to represent the current data generating process and is, thus, disregarded. This older data, however, may contain information (e.g., patterns) that can be used during analysis to better capture the current process. The DyFor GP model is designed to analyze all historical data, save knowledge of past processes, and exploit this learned knowledge to capture the current process.
4) Potentially, with greater computing power comes p better forecasting performance. The DyFor GP model is essentially a heuristic, fitness-driven random search. As with any random search, when a larger percentage of the search-space is covered, better results can be expected. Greater computational power allows for greater search-space coverage, and DyFor GP forecasting performance can be improved by simply increasing such power. Many other forecasting models cannot be improved in this manner.

Continued development and testing of the DyFor GP model is planned. One way of possibly improving the forecasting results of the GDP and CPI Inflation experiments would be to increase the computing power employed. The above results were achieved in a modest cluster computing environment (see [65] for details) so there is much room for increasing the computing power. Concerning the CPI Inflation experiment, Stock and Watson [76] report that using an alternative Phillips Curve specification that replaces the unemployment rate explanatory variable with a new composite index of 168 economic variables that they developed yields better forecasting performance than the conventional Phillips Curve specification. Thus, this composite index could be utilized by the DyFor GP model to potentially produce further performance advances.

Results of this study indicate that the choice of fitness measure plays an important part in the forecasting performance of the DyFor GP model. In some cases, a DyFor GP model with MSE measure proved the most effective, while for other cases a DyFor GP model with MAD was best. A novel fitness measure, CF, which combines aspects of MSE and MAD was developed and tested. The CF measure relies on a parameter, Ω, to determine which data are considered outliers and which are not. Since an optimal specification for Ω was not known, intuition was used to set this parameter. Further studies might examine different settings for this parameter and/or develop some algorithm to automatically adjust this parameter toward its optimal setting. They may also investigate the conditions under which the CF measure gives better performance than the MSE and MAD measures and, perhaps, investigate alternative measures such as the Akaike Information Criterion (AIC) or the Schwarz Information Criterion (SIC).[15]

The observations on window behavior and retention of past adaptations discussed in the previous section were made by examining output files produced by the DyFor GP model with the naked eye. Because these files are quite large, this is not an efficient way to analyze such behavior. DyFor GP implementation could be enhanced to produce more detailed information about both of these aspects at every step of the training and forecast period. This would allow for better visualization and analysis. Window behavior could be further studied by applying the DyFor GP model to several artificial time series (similar to the one described in Section IV-A) each with different characteristics such as shorter/longer segment lengths and smooth/abrupt transitions between segments. Experiments of this kind may provide greater understanding of how window size adjustment is affected by a changing process.

Also, affecting window behavior is parameter N (from Tables II and VII) which specifies the number of expansions/contractions that signal a stable process or a process shift, respectively. Like parameter Ω of the CF fitness measure, an optimal value for parameter N was not known and, thus, intuition was used to specify its value. Future studies might investigate optimal settings for parameter N for a variety of time series.

Another direction for DyFor GP development is in the area of forecast combination. As detailed in Section III-G, it is necessary to make multiple DyFor GP runs and use some method to combine the multiple forecasts produced into a single, out-of-sample forecast. The method utilized in this study is a simple one that ranks each DyFor GP run based on the accuracy of its most recent past forecast, selects the top three runs, averages their current forecasts, and designates this average forecast as the single, out-of-sample forecast to be used. It is reasonable to expect that a more sophisticated forecast combining method would result in performance improvements. One interesting method is the following. Suppose the combination model of (5) (redisplayed here) is considered

$$F = \alpha_1 f_1 + \alpha_2 f_2 + \cdots + \alpha_n f_n.$$

In this model, F is the combined forecast, $f_1, f_2, \ldots, f_n$ are the single forecasts to be combined, and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are corresponding weights subject to the condition that their sum is one. Using all past forecasts produced by a set of n DyFor GP runs as training data, a GA could be employed to evolve optimal weights for this model.

Future experiments are also planned in which the DyFor GP is applied to other well-known economic time series, as well as time series important to other fields such as weather-related series, seismic activity, and series arising from biological/medical processes.

All in all, the DyFor GP model is a viable alternative for real-world forecasting applications and may prove to stimulate new advances in the area of time series forecasting.

[15]A definition and description of AIC and SIC can be found in [21].

REFERENCES

[1] V. Akgiray, "Conditional heteroskedasticity in time series and stock returns: Evidence and forecasts," *J. Business*, vol. 62, pp. 55–80, 1989.
[2] A. Andreou, E. Georgopoulos, and S. Likothanassis, "Exchange rates forecasting: A hybrid algorithm based on genetically optimized adaptive neural networks," *Comput. Econ.*, vol. 20, pp. 191–202, 2002.
[3] M. Andrew and R. Prager, "Genetic programming for the acquisition of double auction market strategies," *Advances in Genetic Programming*, vol. 1, pp. 355–368, 1994.
[4] P. Angeline, "Genetic programming and emergent intelligence," *Advances in Genetic Programming*, vol. 1, pp. 75–98, 1994.
[5] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, and Genetic Algorithms*. London, U.K.: Oxford Univ. Press, 1996.
[6] B. Back, T. Laitinen, and K. Sere, "Neural networks and genetic algorithms for bankruptcy predictions," *Expert Syst. Appl.*, vol. 11, pp. 407–413, 1996.
[7] W. Banzhaf and W. Langdon, "Some considerations on the reason for bloat," *Genetic Program. Evol. Mach.*, vol. 3, pp. 81–91, 2002.

[8] W. Banzhaf, *Genetic Programming: An Introduction*. San Mateo, CA: Morgan Kaufmann, 1998.

[9] P. Bidarkota, "Alternative regime switching models for forecasting inflation," *J. Forecasting*, vol. 20, pp. 21–35, 2001.

[10] M. Billio, D. Sartore, and C. Toffano, "Combining forecasts: Some results on exchange and interest rates," *Eur. J. Finance*, vol. 6, pp. 126–145, 2000.

[11] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *J. Econometrics*, vol. 31, pp. 307–327, 1986.

[12] M. Brameier and W. Banzhaf, "A comparison of linear genetic programming and neural networks in medical data mining," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 17–26, 2001.

[13] L. Chambers, Ed., *Practical Handbook of Genetic Algorithms: Applications*. Boca Raton, FL: CRC Press, 1995.

[14] S. Chen and C. Yeh, "Toward a computable approach to the efficient market hypothesis: An application of genetic programming," *Journal of Economics Dynamics and Control*, vol. 21, pp. 1043–1063, 1996.

[15] S. Chen, C. Yeh, and W. Lee, "Option pricing with genetic programming," in *Proc. 3rd Annu. Conf. Genetic Program.*, 1998, vol. 1, pp. 32–37.

[16] S. Chiraphadhanakul, P. Dangprasert, and V. Avatchanakorn, "Genetic algorithms in forecasting commercial banks deposit," in *Proc. IEEE Int. Conf. Intell. Process. Syst.*, 1997, vol. 1, pp. 557–565.

[17] T. Cooley and S. LeRoy, "Atheoretical macroeconometrics: A critique," *J. Monetary Econ.*, vol. 16, pp. 283–308, 1985.

[18] P. Cortez, M. Rocha, and J. Neves, "Evolving time series forecasting ARMA models," *J. Heuristics*, vol. 10, pp. 415–429, 2004.

[19] G. Deboeck, *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic and Financial Markets*. New York: Wiley, 1994.

[20] M. Dekker, K. van Donselaar, and P. Ouwehand, "How to use aggregation and combined forecasting to improve seasonal demand forecasts," *Int. J. Prod. Econ.*, vol. 90, pp. 151–167, 2004.

[21] F. Diebold, *Elements of Forecasting*. London, U.K.: International Thomson Publishing, 1998.

[22] R. Engle, "Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation," *Econometrica*, vol. 50, pp. 987–1008, 1982.

[23] I. Fischer and N. Harvey, "What information do judges need to outperform the simple average?," *Int. J. Forecasting*, vol. 15, pp. 227–246, 1999.

[24] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[25] L. Fogel, P. Angeline, and D. Fogel, "An evolutionary programming approach to self-adaptation on finite state machines," in *Proc. 4th Annu. Conf. Evol. Program.*, 1995, vol. 1, pp. 355–365.

[26] D. Fogel and K. Chellapilla, "Revisiting evolutionary programming," in *Proc. SPIE Appl. Sci. Comput. Intell.*, 1998, vol. 1, pp. 2–11.

[27] C. Fyfe, J. Marney, and H. Tarbert, "Technical analysis versus market efficiency—A genetic programming approach," *Appl. Fin. Econ.*, vol. 9, pp. 183–191, 1999.

[28] E. Gately, *Neural Networks for Financial Forecasting*. New York: Wiley, 1996.

[29] Y. Goto, K. Yukita, K. Mizuno, and K. Ichiyanagi, "Daily peak load forecasting by structured representation on genetic algorithms for function fitting," *Trans. Inst. Elect. Eng. Japan*, vol. 119, pp. 735–736, 1999.

[30] K. Gurney, *An Introduction to Neural Networks*. London, U.K.: UCL Press, 1997.

[31] D. Hendry and M. Clements, "Pooling of forecasts," *Econometrics J.*, vol. 7, pp. 1–31, 2004.

[32] H. Hiden, B. McKay, M. Willis, and M. Tham, "Non-linear partial least squares using gentic programming," in *Proc. 3rd Annu. Conf. Genetic Program.*, 1998, vol. 1, pp. 128–133.

[33] H. Iba and T. Sasaki, "Using genetic programming to predict financial data," in *Proc. Congr. Evol. Comput.*, 1999, vol. 1, pp. 244–251.

[34] H. Iba and N. Nikolaev, "Genetic programming polynomial models of financial data series," in *Proc. Congr. Evol. Comput.*, 2000, vol. 1, pp. 1459–1466.

[35] H. Iba, H. de Garis, and T. Sato, "Genetic programming using a minimum description length principle," *Advances in Genetic Programming*, vol. 1, pp. 265–284, 1994.

[36] B. Jeong, H. Jung, and N. Park, "A computerized causal forecasting system using genetic algorithms in supply chain management," *J. Syst. Softw.*, vol. 60, pp. 223–237, 2002.

[37] E. de Jong, R. Watson, and J. Pollack, "Reducing bloat and promoting diversity using multi-objective methods," in *Proc. Genetic Evol. Comput. Conf.*, 2001, vol. 1, pp. 11–18.

[38] P. Jonsson and J. Barklund, "Characterizing signal behavior using genetic programming," *Evolutionary Computing*, ser. Lecture Notes in Computer Science, vol. 1143, pp. 62–72, 1996.

[39] Y. Ju, C. Kim, and J. Shim, "Genetic based fuzzy models: Interest rate forecasting problem," *Comput. Ind. Eng.*, vol. 33, pp. 561–564, 1997.

[40] M. Kaboudan, "Forecasting with computer-evolved model specifications: A genetic programming application," *Comput. Oper. Res.*, vol. 30, pp. 1661–1681, 2003.

[41] ——, "Genetically evolved models and normality of their residuals," *J. Econ. Dynamics Control*, vol. 25, pp. 1719–1749, 2001.

[42] ——, "Forecasting stock returns using genetic programming in C++," in *Proc. 11th Annu. Florida Artif. Intell. Int. Res. Symp.*, 1998, vol. 1, pp. 502–511.

[43] ——, "Genetic programming prediction of stock prices," *Comput. Econ.*, vol. 6, pp. 207–236, 2000.

[44] ——, "Genetic evolution of regression models for business and economic forecasting," in *Proc. Congr. Evol. Comput.*, 1999, vol. 2, pp. 1260–1268.

[45] D. Kim and C. Kim, "Forecasting time series with genetic fuzzy predictor ensemble," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 523–535, 1997.

[46] K. Kinnear, "Generality and difficulty in genetic programming: Evolving a sort," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, vol. 1, pp. 120–128.

[47] J. Kitchen and R. Monaco, "Real-time forecasting in practice," *Business Economics: The J. National Assoc. Bus. Econ.*, vol. 38, pp. 10–19, 2003.

[48] J. Korczak and P. Lipinski, "Evolutionary building of stock trading experts in a real-time system," in *Proc. Congr. Evol. Comput.*, 2004, vol. 1, pp. 940–947.

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

[50] W. Langdon, "The evolution of size in variable length representations," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, vol. 1, pp. 633–638.

[51] W. Langdon and R. Poli, "Fitness causes bloat," *Soft Comput. Eng. Des. Manuf.*, vol. 1, pp. 13–22, 1997.

[52] D. Lee, B. Lee, and S. Chang, "Genetic programming model for long-term forecasting of electric power demand," *Electric Power Syst. Res.*, vol. 40, pp. 17–22, 1997.

[53] W. Leigh, R. Purvis, and J. Ragusa, "Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: A case study in romantic decision support," *Decision Support Syst.*, vol. 32, pp. 361–377, 2002.

[54] J. Levenick, "Swappers: Introns promote flexibility, diversity, and invention," in *Proc. Genetic Evol. Comput. Conf.*, 1999, vol. 1, pp. 361–368.

[55] P. Lipinski, "Evolutionary data-mining methods in discovering stock market expertise from financial time series," Ph.D. dissertation, Université Louis Pasteur, Strasbourg, 2004.

[56] Y. Liu and X. Yao, "Evolving neural networks for Hang Seng stock index forecasting," in *Proc. Congr. Evol. Comput.*, 2001, vol. 1, pp. 256–260.

[57] S. Makridakis, S. Wheelwright, and R. Hyndman, *Forecasting: Methods and Applications*. New York: Wiley, 1998.

[58] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Netw.*, vol. 5, pp. 39–53, 1994.

[59] S. Markose, E. Tsang, and H. Er, "Evolutionary decision trees in FTSE-100 index options and futures arbitrage," in *Genetic Algorithms and Programming in Computational Finance*, ser. Kluwer Series in Computational Finance. : , 2002, ch. 14, pp. 281–308.

[60] D. G. McMillan, "Nonlinear predictability of stock market returns: Evidence from nonparametric and threshold models," *Int. Rev. Econ. Fin.*, vol. 10, pp. 353–368, 2001.

[61] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1992.

[62] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1996.

[63] B. Mulloy, R. Riolo, and R. Savit, "Dynamics of genetic programming and chaotic time series prediction," in *Proc. 1st Annu. Conf. Genetic Program.*, 1996, vol. 1, pp. 166–174.

[64] A. Nag and A. Mitra, "Forecasting daily foreign exchange rates using genetically optimized neural networks," *J. Forecasting*, vol. 21, pp. 501–511, 2002.

[65] "High Performance and Grid Computing." North Carolina State University, Raleigh, NC, 2004 [Online]. Available: http://www.ncsu.edu/itd/hpc/main.php

[66] C. Neely and P. Weller, "Predicting exchange rate volatility: Genetic programming versus GARCH and RiskMetrics™," The Federal Reserve Bank of St. Louis, St. Louis, 2002.

[67] C. Neely, P. Weller, and R. Dittmar, "Is technical analysis in the foreign exchange market profitable? A genetic programming approach," *J. Fin. Quant. Anal.*, vol. 32, pp. 405–426, 1997.

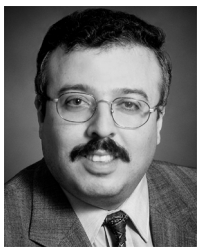
[68] C. Neely, "Risk-adjusted, ex ante, optimal technical trading rules in equity markets," *Int. Rev. Econ. Fin.*, vol. 12, pp. 69–87, 2003.
[69] P. Nordin and W. Banzhaf, "Complexity compression and evolution," in *Proc. 6th Int. Conf. Genetic Algorithms*, 1995, vol. 1, pp. 310–317.
[70] P. Phua, D. Ming, and W. Lin, "Neural network with genetically evolved algorithms for stocks prediction," *Asia-Pacific J. Oper. Res.*, vol. 18, pp. 103–107, 2001.
[71] J. Rosca, "Generality versus size in genetic programming," in *Proc. 1st Annu. Conf. Genetic Program.*, 1996, vol. 1, pp. 381–387.
[72] N. Sarantis, "Nonlinearities, cyclical behavior and predictability in stock markets: International evidence," *Int. J. Forecasting*, vol. 17, pp. 459–482, 2001.
[73] R. Sathyanarayan, S. Birru, and K. Chellapilla, "Evolving nonlinear time series models using evolutionary programming," in *Proc. Congr. Evol. Comput.*, 1999, vol. 1, pp. 243–253.
[74] R. Sexton, "Identifying irrelevant input variables in chaotic time series problems: Using genetic algorithms for training neural networks," *J. Comput. Intell. Finance*, vol. 6, pp. 34–41, 1998.
[75] K. Smith and J. Gupta, *Neural Networks in Business: Techniques and Applications*. Hershey, PA: Idea Group Pub., 2002.
[76] J. Stock and M. Watson, "Forecasting inflation," *J. Monetary Econ.*, vol. 44, pp. 293–335, 1999.
[77] ——, "Forecasting output and inflation: The role of asset prices," *J. Econ. Literature*, vol. 41, pp. 788–829, 2003.
[78] W. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, vol. 1, pp. 135–142.
[79] R. Trippi and E. Turban, Eds., *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real-World Performance*. Chicago, IL: Irwin Professional Pub., 1996.
[80] E. Tsang and J. Li, "EDDIE for financial forecasting," in *Genetic Algorithms and Programming in Computational Finance*, ser. Kluwer Series in Computational Finance. : , 2002, ch. 7, pp. 161–174.
[81] U.S. Department of Commerce Bureau of Economic Analysis. [Online]. Available: http://www.bea.doc.gov/bea/glossary/glossary_g.htm, 2004
[82] R. Venkatesan and V. Kumar, "A genetic algorithms approach to growth phase forecasting of wireless subscribers," *Int. J. Forecasting*, vol. 18, pp. 625–646, 2002.
[83] N. Wagner and Z. Michalewicz, "Genetic programming with efficient population control for financial times series prediction," in *Proc. Genetic Evol. Computa. Conf., Late Breaking Papers*, 2001, vol. 1, pp. 458–462.
[84] ——, "Forecasting with a dynamic window of time: The DyFor genetic program model," *Lecture Notes in Computer Science*, vol. 3490, pp. 205–215, 2005.
[85] ——, "Time series forecasting for dynamic environments: The DyFor genetic program model," in *Proc. Int. Seminar Soft Comput. Intell. Syst.*, 2005, vol. 1, pp. 152–164.
[86] J. Wang, "Trading and hedging in S&P 500 spot and futures markets using genetic programming," *J. Futures Markets*, vol. 20, pp. 911–942, 2000.
[87] D. Whitley, "The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, vol. 1, pp. 116–121.
[88] H. White, *Artificial Neural Networks: Approximation and Learning Theory*. Oxford, U.K.: Blackwell, 1992.
[89] J. White, "A genetic adaptive neural network approach to pricing option: A simulation analysis," *J. Comput. Intell. Finance*, vol. 6, pp. 13–23, 1998.
[90] Y. Yang, "Combining forecasting procedures: Some theoretical results," *Econometric Theory*, vol. 20, pp. 176–222, 2004.
[91] X. Yao and Y. Liu, "EPNet for chaotic time series prediction," in *Proc. 1st Asia-Pacific Complex Syst. Conf.*, 1997, vol. 1, pp. 146–156.

Neal Wagner received the B.A. degree in mathematics from the University of North Carolina, Asheville, and the M.S. degree in computer science and the Ph.D. degree in information technology both from the University of North Carolina, Charlotte.

Currently, he is an Assistant Professor of Computer Science at Augusta State University, Augusta, GA. His research interests are in the areas of evolutionary computation, computational forecasting, bioinformatics, and adaptive business applications.

Zbigniew Michalewicz is a Professor at the School of Computer Science, University of Adelaide, Adelaide, Australia. He also holds professor positions at the Institute of Computer Science, Polish Academy of Sciences, and the Polish-Japanese Institute of Information Technology. He is also Chairman of the Board of SolveIT Software, a technology company he co-founded in Australia. He serves as editor for 11 international journals. He has published over 200 articles and 15 books; these include *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, 1996), *How to Solve It: Modern Heuristics* (Springer, 2004), with D. B. Fogel, as well as two very recent books *Adaptive Business Intelligence* (Springer, 2006), with M. Schmidt, M. Michalewicz, and C. Chiriac, and *Winning Credibility: A Guide for Building a Business From Rags to Riches* (Hybrid Publishers, 2006), with M. Michalewicz. His current research interests are in modern heuristic methods.

Dr. Michalewicz was the General Chair of the First IEEE International Conference on Evolutionary Computation (Orlando, FL, 1994).



Moutaz Khouja received the B.S. degree in mechanical engineering and the M.B.A. degree from the University of Toledo, Toledo, OH, and the Ph.D. degree in operations management from Kent State University, Kent, OH.

Currently, he is a Professor of Operations Management at the Belk College of Business Administration, University of North Carolina, Charlotte. His publications have appeared in many leading journals including *Computers and Operations Research*, *Decision Sciences*, *IIE Transactions*, *European Journal of Operational Research*, *International Journal of Production Research*, *International Journal of Production Economics*, *Journal of the Operational Research Society*, and *Omega*. His research interests are in the areas of inventory management, production planning and control, pricing, and forecasting.

Rob Roy McGregor received the B.A. degree in economics and the M.A. degree in economics from Clemson University, Clemson, SC, in 1982 and 1984, respectively. After spending 1984–1987 as a lecturer in economics at the University of North Carolina, Charlotte, he returned to graduate school and received the Ph.D. degree in economics from the University of South Carolina, Columbia, in 1991 .

He is a Professor of Economics and Coordinator of the M.S. in economics program. He has been on the faculty at the University of North Carolina since 1991. He teaches courses in macroeconomics, monetary economics, econometrics, and forecasting. He has also presented several continuing education seminars on the Federal Reserve and its role in U.S. financial markets and the U.S. economy. His principal research interests center on the monetary policy decision process of the Federal Reserve's Federal Open Market Committee (FOMC). His research on the FOMC has been published in the *Quarterly Journal of Economics*, the *Journal of Economics and Business, Economics and Politics*, the *Review of Economics and Statistics*, the *Southern Economic Journal*, the *Journal of Money, Credit, and Banking*, and *Challenge: The Magazine of Economic Affairs*. His book *Committee Decisions on Monetary Policy: Evidence from Historical Records of the Federal Open Market Committee* (MIT, 2005), was coauthored with H. W. Chappell, Jr., of the University of South Carolina and T. Vermilyea, of the Federal Reserve Bank of Philadelphia.