Genetic Programming-based Regression for Temporal Data

Cry Kuranga (kurangacry@gmail.com) and Nelishia Pillay (npillay@cs.up.ac.za)
Department of Computer Science University of Pretoria, Lynnwood Road, Hillcrest, Pretoria, South Africa, 0002 (Phone 012 420 5242)

Abstract - Various machine learning techniques exist to perform regression on temporal data with concept drift occurring. However, there are numerous nonstationary environments where these techniques may fail to either track or detect the changes. This study develops a genetic programming-based predictive model for temporal data with a numerical target that tracks changes in a dataset due to concept drift. When an environmental change is evident, the proposed algorithm reacts to the change by clustering the data and then induce nonlinear models that describe generated clusters. Nonlinear models become terminal nodes of genetic programming model trees. Experiments were carried out using seven nonstationary datasets and the obtained results suggest that the proposed model yields high adaptation rates and accuracy to several types of concept drifts. Future work will consider strengthening the adaptation to concept drift and the fast implementation of genetic programming on GPUs to provide fast learning for high-speed temporal data.

Keywords: temporal data, concept drift, model induction, nonlinear model, predictive model, genetic programming.

1. Introduction

Numerous data analytics approaches exist for stationary data, where a static prediction model is constructed once, on the stationary data, and used to predict for instances not used during training, and extract knowledge from the underlying data. However, concepts in real-world are nonstationary, thus, their underlying phenomena change over time [1]. As such, non-adaptive models induced under the false stationarity assumption usually become obsolete as changes occur in the data and may fail terribly at worst or perform sub-optimally at best [1].

Temporal data is a series of events over a period that may be time-stamped at regular or irregular time intervals [2]. Examples of such concepts include weather prediction rules that may change with seasons, patterns of customers' buying preferences that may vary with time (such as holidays, weekend or month-end), the inflation rate or availability of alternatives, among many others [3]. Data streams are a continuous flow of data such as sensor data and network traffic where order and time are unnecessarily the fundamental elements that are central to the meaning of the data [4]. However, a data stream may consist of temporal data or non-temporal data. This work focuses on temporal data.

Temporal data is usually made up of different underlying data generating processes that are produced incrementally and characterized by a high amount of dependency among itself in which appropriate treatment of these dependencies or relationships is essential. Differing underlying data generating processes result in concept drift. As such, what happens in the future cannot be predicted with certainty. Also, the properties of temporal data may be subject to concept drift which may be meant to be discovered by the learning system [5] [6] [7]. Therefore, the learned model should be continuously adapted. Thus, special approaches are sought that treat the most recent historical occurrences as equally significant contributors to the final concept. As such, algorithms that can evolve or adapt models are ideal to track and monitor the underlying changes to adapt the model accordingly.

Adaptation algorithms use either a passive or an active approach to learning in the concept drift occurring environment [8] [9]. A passive approach updates the model whenever the new data is made available whereas an active approach needs to detect a change in the data to adapt the model.

This work develops genetic programming (GP)-based predictive model that is designed for temporal data with a numerical target that dynamically adapts when concept drift occurs and can also be used to extract knowledge from historical data. The proposed approach implements a piecewise approach to predict a target, nonlinear model. The obtained results suggest that the proposed model yields high adaptation rates and accuracy to several types of concept drifts. Also, it was evident that the proposed model can automatically adapt to different types of concept drifts and quickly improve its performance without the use of an explicit drift detector.

The main contributions of this work are to:

- Hybridize a dynamic clustering algorithm, nonlinear regression model and a GP to perform regression on temporal data with concept drift occurring.
- Implement a dynamic GP terminal set that consist of recent and past learned nonlinear regression models. The goal is to induce a predictive model using both the recent occurrences and the past learned knowledge.
- Perform a thorough experimental study on the temporal datasets that exhibit different characteristics of concept drift, namely progressive, recurrent, abrupt and random changes to compare with the state-of-the-art machine learning algorithms to ascertain the performance of the proposed model.

This paper is organized as follows: Section 2 review related work, whereas Section 3 discusses the proposed model. Section 4 discusses the experimental setup and Section 5 presents the obtained results and discussion. Section 6 concludes the paper.

2. Learning in Nonstationary Environments

Temporal data can be categorized into temporal sequences, semantic temporal data, and time series. The temporal sequence is a series of events over time whereas semantic temporal data is described within the perspective of human existence ontology, i.e., male or female [2]. The time series is defined as a progression of successive real-valued data points collected at regular intervals in time.

Nonstationarity in real-world data may be caused by concept drift. Concept drift, in machine learning, is a phenomenon where statistical properties of the concept (target variable), being predicted, change as time passes unexpectedly due to changes in underlying data patterns [11]. Thus, the relations and patterns in such data changes over time (systematic variation in the underlying process) [12]. As a result, the prediction accuracy of a model built to predict such data deteriorates over time.

The two broad categories commonly used to learn in concept drift occurring environments are generally referred to as passive and active approaches [8]. Generally, the active approach copes quite well in environments where drift is abrupt whereas a passive approach is ideal for gradual drifts and recurring concepts [8] [9].

In this section, Section 2.1 discusses the data generating process. Section 2.2 discusses active learning approach whereas passive learning approach is discussed in Section 2.3. Genetic programming for dynamic environments is discussed in Section 2.4

2.1 The Data Generating Process

Considering $\mathcal{P}$ to be a data generating process that provides a sequence of tuples$(\boldsymbol{x}_t, y_t)$, in time t, sampled from a joint probability distribution $p_t(\boldsymbol{x}_t, y_t)$, evidence distributions to be $p_t(\boldsymbol{x})$ and prior probabilities to be $p_t(y|\boldsymbol{x})$ [13]. Also, considering what is changing, drifts can be classified as [1]:

- Real drift, where $p_t(y|\boldsymbol{x})$ changes, independently from variations in the $p_t(\boldsymbol{x})$, over time. Real drift is also referred to as conditional change or concept shift [7];
- Virtual drift, where a change in the $p_t(\boldsymbol{x})$ does not affect $p_t(y|\boldsymbol{x})$. Virtual drift is also referred to as feature changes, sampling shifts or temporary drifts [7]

Some literature defines virtual drift as a change that does not affect $p_t(\boldsymbol{x})$, though the source and the interpretation of such changes vary [1] [12] [14]. If $p_t(\boldsymbol{x})$ and the decision boundaries in the data remain the same, then it becomes insignificant to rebuild the model. However, the detection of virtual drift is still crucial since if it is incorrectly interpreted then they are a high likelihood of making an incorrect decision of retraining the regressor.

The change in probability distributions can further be classified according to the rate at which the drift is happening such as abrupt drifts (concept changes) which results in sudden drifts and gradual drifts, results in slowly evolving distributions over time [15] [16]. These drifts: abrupt and gradual can also be classified as recurrent or cyclical variations. Recurrent or cyclical variations happen to be a desirable and valuable quality sought in adaptive algorithms to promote the retrieval of previously acquired knowledge.

In inductive learning, transfer-learning allows the target task's inductive bias to be affected by source-task knowledge in anticipation of performance improvements in terms of learning speed and generalization capability [10]. Consequently, an inductive bias promotes the prioritization of a single solution (or interpretation) by a learning algorithm. Also, in a regression ensemble, inductive bias plays a pivotal role in diversity maintenance [18][58].

2.2 An Active Approach to Learning in Nonstationary Environments

An active approach to learning in the nonstationary environment with concept drift occurring is based on a change detection concept. An adaptation technique is triggered that aims to react to a detected change by adapting an existing model or inducing a new one. As such, adaptive techniques are commonly referred to as "detect and react" approaches [15]. The goal of the change detector is to assert if there exists a change in the process $\mathcal{P}$ (discussed in Section 2.1) through an inspection of features extracted, to monitor the stationarity of an estimated $p_t(\boldsymbol{x})$, from the data-generating process and /or analyzing the prediction error to determine variations in the estimated $p_t(y|\boldsymbol{x})$ [17]. Thus, obsolete knowledge is discarded and the regressor adapts to a new environment when an environmental change is detected. However, the major challenge of this approach is to distinguish effectively between up-to-date and obsolete data patterns.

2.3 A Passive Approach to Learning in Nonstationary Environments

Unlike an active approach, a passive approach continuously adapts the model parameters whenever new data patterns arrive to cope with uncertainty in the presence of change. Therefore, a passive approach maintains an updated model at all times and avoids the common pitfalls in an active approach which are either falsely detecting non-existent change or failing to detect a change [1].

Passive approaches can be classified as those that adapt/remove/add members of an ensemble-based system and those that adapt a single-regressor. An ensemble-based approach has a higher computational cost than a single-regressor. However, the ensemble-based approach provides a natural fit to learning in nonstationary environments and offers distinct advantages which include: striking a delicate balance along the stability-plasticity spectrum [18] [19]. Thus, it provides a flexible way to incorporate new data, when it is available, into a regression model by simply inserting new members into the ensemble and provide a natural technique to forget irrelevant knowledge by removing irrelevant members from the ensemble. Also, an ensemble approach tends to reduce the variance of error, therefore, become more accurate than single regressor-based approaches. As such, ensemble systems provide a good fit for learning in nonstationary environments, especially, if the drift impacts some parts of the existing knowledge base leaving the other parts relevant.

2.4 Genetic Programming for Dynamic Environments

Genetic programming is an evolutionary computation technique that has been applied by several authors to evolve predictive models with favourable results [20] [21] [22]. Individuals in a GP can be generated using grammars, commonly referred to as grammar-guided GP and has been applied extensively to data mining [23] [24].

GP modifies its population as it converges towards optimality. However, it becomes difficult to re-diversify a converged population once an environment change has occurred. As a result, the population lacks diversity necessary to locate a new optimum. Therefore, a standard GP algorithm may be ineffective in dynamic environments.

Numerous techniques were proposed in the literature to make GP cope with dynamic environments [25] [26]. This section reviews GP variants designed for dynamic environments which are classified as parametric and memory-based approaches. Also, regression analysis is discussed in this section.

2.4.1 Parametric Approach

A dynamic GP that implements an adaptive parametric approach was proposed in [27]. Adaptive control parameters enable the GP algorithm to dynamically adjust as a change in environment is evident. When a change in environment is evident, GP reacts to the change by triggering exploration through adaptive control parameters to locate a new optimum. The following adaptive control parameters were implemented in [27]:

- Elitist proportion
  Whenever the fitness (training accuracy) deteriorated, the percentage of elitist individuals is reduced by 0.1 with a lower bound of 0.1, to promote exploration and increased by 0.1 with an upper bound of 1 when the fitness improves. The decrease in

elitist proportion facilitates the addition of more newly generated material into a population in the hope of locating a new optimum.

- Crossover rate
  Similarly, to elitist proportion, if the fitness deteriorates, the crossover probability is linear decreased, by 0.1 with a lower bound of 0.1 and when the fitness improves, the crossover rate increased by 0.1 with an upper bound of 0.9, to spare the computational effort of generating more offspring to be mutated.
- Mutational rate
  For gradual changes in the environment, a low mutation rate is required to promote exploitation whereas abrupt changes in the environment require high mutation rates to promote exploration. The mutation rates were set to be cyclic using the following probabilities: mutating each node of the tree, applying an operator and mutating an individual. Whenever an environment change is evident, a random number in the range [0; 1] is added to the mutational probability. If the rate exceeds the value of 1, then the value is scaled to the range [0; 1].
- Culling
  Culling facilitates exploration in a GP algorithm. A portion of the worst individuals in a population are replaced by randomly generated individuals. A randomly generated individual consists only of a terminal node as a root node that is mutated before being added into the population.

The adaptive GP (DynGP) was compared to the gradient descend-artificial neural network and standard GP and the adaptive GP outperformed all other training algorithms in all severity of changes in environment modifications [27].

2.4.2 Memory Approach

Numerous variants of memory-based GP exist in the literature which includes dynamic forecasting GP [28][59][60]. The dynamic forecasting GP (DyFor GP) aims at time series forecasting in nonstationary environments that automatically adapts to a changing environment and retains knowledge (implicit memory) from past environments through introns [28]. The DyFor GP implemented a sliding window of analysis to model a natural adaptation for a nonstationary environment. Two sliding windows are defined at the beginning of historical data and slides after a given number of iterations (dynamic generation). Consequently, discovers optimal analysis window dynamically. For each dynamic generation, two runs are executed, one for each analysis window. The prediction accuracy for each analysis window after a dynamic generation is used to adjust the window size. The window that yields the best prediction accuracy is maintained whereas the other window either shrink, if the best window is smaller, or expand. Also, DyFor GP makes use of explicit memory in the form of dormant solution. A dormant solution is an existing solution (inactive) that becomes active only if the applicable conditions arise. Dormant solutions usually speed up the convergence of the algorithm, therefore, when a change in the environment is detected, dormant solutions are injected into the population.

The DyFor GP was compared to the auto-regressive and real-time forecasting system on real-world time series and it outperformed all other training algorithms [28].

2.4.3 Regression Analysis

Regression analysis is a statistical-based model induction technique that models a relationship between variables (quantitative), ideal to predict a selected variable from one or more other variables [29]. Regression analysis use equations to describe the given dataset whereby a regression model consists of the following components [30]:

- A response variable(s), i.e., a real-world output, y
- A vector of predictor variables, i.e., an input vector $\boldsymbol{x} = x_1; x_2; \dots; x_n$
- An unknown parameter, θ, which can be a vector or scalar.

Given that $\boldsymbol{x}$ is an input vector and y is a real-world output, a nonlinear model between $\boldsymbol{x}$ and y is of the form:

$$y = f(\boldsymbol{x}; \theta) + \varepsilon \qquad (1)$$

where θ is the parameter vector and ε a random error. A process of fitting the best approximation to a dataset of $n = |N|$ data points $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ is commonly referred to as least-squares approximation [31] [32]. Nonlinear systems identification can be defined as a least-squares problem using Equation (1) where θ is $n \times 1$, y is $m \times 1$ real-world output, x is $m \times n$ regressor matrix and a white noise residue, ε.

A GP can perform symbolic regression on raw data and variables that show nonlinear correlations [33] [34] [35] [36]. In this regard, GP automatically evolves both the parameters and the structure of the mathematical model. Thus, a GP tree is induced from a function set that contains mathematical functions and the input set that consists of historical data and (or) explanatory variable. Therefore, a GP is an adaptable model for regression analysis of nonlinear data.

3. Proposed Model

The proposed genetic programming-based regression model for temporal data (GP-REG) implements a piecewise approach to predict a target, nonlinear model. The GP-REG consists of three components: a dynamic clustering algorithm (KCDCDynDE) to extract clusters that resemble different data generating processes present in the dataset; a dynamic QPSO-based model induction technique to induce nonlinear models for each generated cluster which approximate mapping between inputs and the target variable; and a GP that evolves model trees that implement piecewise nonlinear predictive models which define the boundaries of nonlinear models expressed as terminal nodes. Algorithm 1 summarized the proposed GP-REG.

3.1 Dynamic Data Clustering

Clustering is performed as the first step to the non-linear model induction. Since nonstationary data is usually made up of different underlying data generating processes, clustering is performed to extract data patterns that resemble different data generating processes present in the dataset. Each cluster can be considered as generated by the same data generating process. A model is then fit on each generated cluster.

A change in the environment may cause cluster centroids to move, data points to migrate between clusters and the number of clusters to increase or decrease. A KCDCDynDE can cluster in a dynamic environment, therefore, it is adopted in this work to cluster nonstationary data [37]. Whenever a change in an environment is evident, KCDCDynDE automatically determines the optimal number of clusters.

Algorithm 1 GP-REG

```
Set the size of the sliding window of analysis to W_S data patterns
Perform nonlinear model induction initialization process using Algorithm 2
Nonlinear models constitute the terminal set of GP algorithm
Initialize Pop(0);

BEGIN
      Slide a data window of analysis
      Perform Environmental change update process using Algorithm 3
      Let t = 0;
      WHILE termination condition(s) not satisfied DO
          Select individuals from Pop(t) to use in reproduction process;
          Perform crossover using selected individuals to create offspring;
          Perform mutation on the offspring;
          Compute fitness for each offspring;
          Select individuals to constitute a new population, Pop(t + 1)
          t = t + 1;
      END WHILE
      REPEAT until no further data to analyze.
END
```

3.2 Nonlinear Regression Models

A *dynamic QPSO-based nonlinear model induction technique (DynQPSO)* is adopted in this work to induce non-linear models which approximate mapping between inputs and the target variable [38]. A DynQPSO induces optimal nonlinear models that describe each cluster. To induce a nonlinear model using a DynQPSO, firstly, the value of the coefficients of the model are determined by a *QR decomposition technique* [32]. Given the value of the coefficients, a charged-PSO determines a structurally optimal nonlinear model which symbolizes the functional mapping between input and output space.

Algorithm 2 builds a dynamic GP terminal set, which consists of nonlinear models, A_r, by performing multiple piecewise approximations of problem space. Each nonlinear model, A_r, is assigned a lifetime which expires if it exceeds the upper-bound to avoid the size of the terminal set to become unnecessarily large which tend to affect the convergence properties of the genetic programming. The lifetime (π_ω) is a user-defined parameter which is initialized to zero and increments by a unit on each generation. However, the parameter is reset whenever a model is deemed useful.

Algorithm 2 Nonlinear Model Induction Initialization Process

```
BEGIN
   Perform data clustering using KCDCDynDE to obtain k clusters
      FOR each cluster
            Perform nonlinear regression using DynQPSO to obtain a model, A_r
            Insert the induced model, A_r, into the terminal set
      END
END
```

3.3 Genetic Programming

Each individual (model tree) in the GP population represents a piecewise nonlinear predictive model. Thus, a GP evolves model trees with its terminal nodes expressed as nonlinear models.

An example of GP-REG individual is graphically illustrated in Figure 1. Considering Figure 1, nodes are recursively added within a maximum tree depth to a root node $(x_2 < v_2)$. A consequent, $y_i, i \in \{1,2,\dots,n\}$, whereby $y = f(x) + \varepsilon$, is a nonlinear model (A_r), x_i is a discrete- or continuous-valued attribute and v_i is a possible value of x_i. The operator, $Op \in \{<, >, =, \neq\}$.

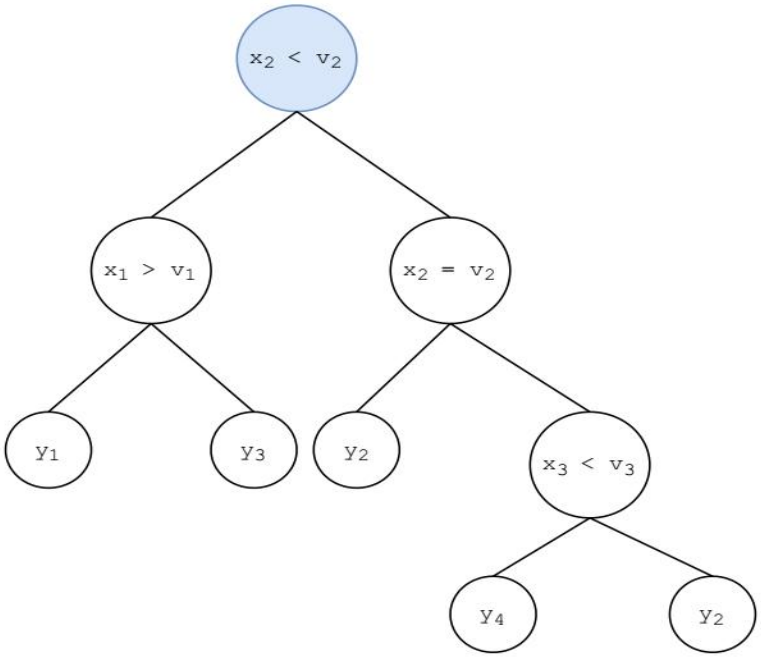

Figure 1: An example of a GP-REG Individual

3.3.1 Reproduction

GP operators are meant to reproduce new individuals with better features and properties over their parents. Commonly used GP operators are reproduction, crossover and mutation [39] [40]. Reproduction passes selected individuals, usually the elite, onto the next population. Crossover is considered as a local search operator that promotes convergence through combining existing genetic material from two selected parents whereas mutation introduces new genetic material to a selected individual [41] [39]. Mutation promotes population diversity, as such, mutation is considered as a global search operator. The parameter, *mutation depth*, controls the size of a subtree created by mutation to conform to the given offspring depth.

The population size of a GP is fixed. As such, application rates parameters commonly referred to as *crossover*, *mutation* and *reproduction rate* determines the total number of individuals to be reproduced by the crossover, mutation and reproduction respectively [41]. A tournament selection technique is adopted in this work.

The crossover operator picks a crossover point (it disallows extreme point) from two selected model trees and then swaps sub-trees to construct a new offspring. The crossover operator also curtails offspring longer than the maximum allowed number of genes. A direct

reproduction operator reproduces an individual by simply copying an elitist parent to the new population.

Mutation operators enable the GP to adapt to concept drift by introducing a new branch on an individual to model the new area covered by terminal nodes [42]. Different mutation operators were adopted to enable the GP to adapt to concept drift. A mutation point is not randomly chosen, instead, the mean square error is calculated for each node (on the target individual), at the selected depth and each terminal node. The calculated mean squared error, E_{MS}, is used to determine the relative error of a subtree or terminal node (A_r). A *nonterminal node*, at the given depth, with a higher E_{MS} (the worst nonterminal node) is altered. This operation enables the optimization of partitions of a model tree described by the nonterminal nodes. Also, a *terminal node* with a higher E_{MS} is altered. This operation enables optimization of nonlinear approximation described by the path to a terminal node by replacing the current model with a randomly selected model. A node altering operator includes changing: relational operators; or terminal nodes.

Environment Change Detection

To simulate dynamic environments, a sliding window of analysis technique is used [43]. The size of the sliding window and the sampling factor are both user-parameterized. The sliding window of analysis technique dismisses the oldest instances as new ones arrive, thereby providing up-to-date data to the predictor.

As the analysis window slides, the easiest way to detect environmental change is to keep track of the best fitness found [27]. Considering an elitist individual in the current population, the best fitness should never deteriorate as long as the environment is static, with no new data that changes the underlying target distribution. A *significant* decrease ($> n_s\%$) in fitness implies an environmental change or new data that change the underlying target distribution is encountered. The parameter, n_s is a user-defined parameter.

If an environmental change is detected, KCDCDynDE dynamically clusters the data. For the dataset that changes, a DynQPSO adapts nonlinear models or induce new models to create an updated GP terminal set. Algorithm 3 summarized the GP-REG coping with concept drift occurring.

Algorithm 3 Environmental Change Update Process

```
BEGIN
   IF an environmental change is detected
      Dynamically cluster data using KCDCDynDE
      IF the data clusters changes,
            Adapt the nonlinear model or induce new models using DynQPSO
            Insert new models into a terminal set to replace the worst models
      END
   END
END
```

As the population moves toward the desired goal, the degree of the model trees' dispersion decreases. It becomes difficult to re-diversify a converged population. As a result, the population lacks diversity necessary to locate a new optimum when a new data generating process is encountered. Therefore, to enhance population diversity whenever new data generating process is detected, the GP-REG implements a culling technique. Thus, a portion

of the worst population (model trees with higher RMSE) are replaced by new model trees that are randomly evolved using an updated GP terminal set.

Fitness Function

Concept drift affects fitness either by increasing or decreasing it. An increase in fitness implies that the new environments follow the previously encountered trends whereas a decrease implies that the new environments describe new trends. Thus, a decrease in fitness requires less exploitation while promoting more of exploration.

The root mean square error (RMSE), which is the square root of E_{MS}, calculated over the dataset on each iteration is used to evaluate the fitness of each individual. RMSE calculates the misfit of the regression estimate to its expected output. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{y}_i))^2} \qquad 1$$

where y_i is the outcome of an i^{th} data point, $f(\boldsymbol{y}_i)$ is the evaluated outcome using i^{th} data point and n is a data-points counter in the dataset.

4 Experimental Setup

All the experiments conducted in this work were implemented in a MATLAB programming environment [44] on an Intel Core i7 processor (3.1 GHz) desktop with 16 GB of memory on a Linux Centos 7 system. The DynGP and DyFor GP described in Section 2 were used to benchmark the GP-REG to evolve predictive nonlinear regression model. The training algorithms traverse the complete dataset for every scenario under consideration. The GP function set, $function\ set \in \{+, -, \div, \times, \sqrt{}, \sin, \cos, \exp, \ln\}$ was implemented.

A collective mean fitness performance measure was implemented in this work. A collective mean fitness is a representative performance measure in which the algorithm performance is reflected across the dynamic range of search space. Thus, a collective mean fitness performance measure provides a clue on the algorithm adaptive properties which depict the entire performance linking of the algorithm and is independent of any extra knowledge about the search space such as global optimum location. A collective mean fitness was computed as [45]

$$\bar{F}(n) = \frac{\sum_{i=1}^{n} f(i)}{n}$$

where n is the maximum number of generations and $f(i)$ is the fitness value (RMSE) at generation i.

For each algorithm, a total of 30 independent runs were executed on each given dataset. Firstly, a Kruskal-Wallis test was performed to establish if there exists a statistically significant difference between the mean fitness values ($\bar{F}$) of the algorithms for a given problem. If there exists a statistical difference between the algorithm's performances, a Mann-Whitney U test was performed at a significance level of 0.05. These tests were performed for every pair of algorithms and all problems. The obtained Mann-Whitney U-values were used to determine

the winning and losing algorithm. A test was performed for the algorithms' mean fitness values, μ_1 and μ_2 , whereby $H_0 : \mu_1 = \mu_2$, and $H_1 : \mu_1 \neq \mu_2$. The overall performances were ranked based on the difference between wins and losses of each algorithm.

4.1 Test Environments

Several dynamic test problems were designed to evaluate the performance of evolutionary algorithms in nonstationary environments such as the DF1 generator [46], the "moving peaks" benchmark [47], the single and multi-objective dynamic test problem generator [48], the dynamic multi-knapsack problem and the traveling salesman problem [49], and the generalized dynamic benchmark generator [50]. However, the mentioned benchmarks were not favorable for the present study. This work aims to assess the ability of GP to track and adapt dynamically, induced structurally optimal nonlinear regression models as the environment changes. As such, a new set of benchmark problems tailored to assess the ability of GP's ability to track and adapts dynamically the induced structurally optimal nonlinear regression models in nonstationary environments were defined.

In this work, four artificially generated, and three real-world nonstationary test environments were used in the experiments. These seven test environments differ in the total number of underlying data generating processes, the chance of having conflicting decision boundaries within a data window, and the dimensionality of the problem. A data pattern consists of inputs and a target output for the given dataset (a tuple).

The commonly used forgetting technique to cater for outdated instances removal; the sliding windows is adopted in this work [51]. A sliding window of analysis technique enables the regressor to reflect and adapt concept drifts in the temporal data. A sliding window of analysis was set to w_s data patterns. For each sliding window, the order of the data points was preserved and the dataset was split into training and generalization subsets using a ratio 4:1 respectively i.e., if the dataset consists of 10 data points, the first 8 data points are used for training and the last 2 data points for generalization.

Changes in the search space are characterized by two major changes: spatial and temporal severities [61]. Temporal severity refers to the frequency at which environmental changes can occur on any timescale. The environmental change can occur periodically, at irregular time intervals or continuously spread over time. Spatial severity refers to the magnitude of change. Change can be in the context of location and/or the objective value of the position where the change occurred. Temporal and spatial severities are indicators of the complexity of the problem caused by concept drift. For temporal severity, the rate at which the concept changes have an impact on the rate at which the decision boundaries change. The severity of concept changes has an impact on the magnitude by which the decision boundaries disappear, appear or shift. For a more severe change, typically, it is harder for the metaheuristic to recover from the change.

Each test environment was characterized by two variables: w_{shift}, to ascertain the *degree of spatial severity* and the frequency of change, f (the number of iterations), to ascertain the *degree of temporal severity*. In each experiment, the frequencies (f) of $25; 50; 100; 250$ iterations were implemented. The spatial severities (w_{shift}) of $50; 100; 250; 500; 1000$ to simulate different dynamic environments from gradual changes to most abrupt were implemented on artificially generated datasets. The effect of parameter values was expected to be stronger for smaller values, therefore, temporal and spatial severity increases nonlinearly. The w_{shift} introduces new decision boundaries in the analysis window as illustrated in Figure 2.

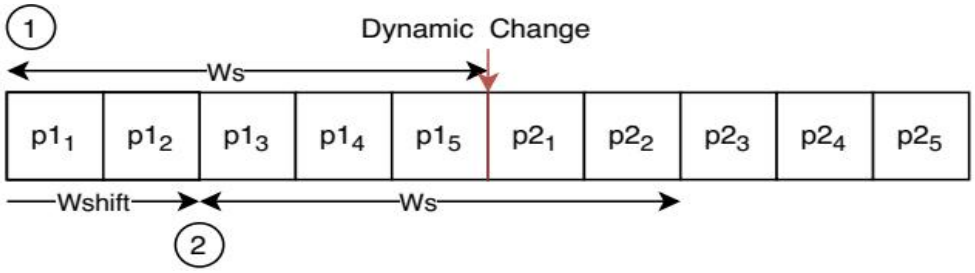

Figure 2: Introducing new decision boundaries by window shifts

a) **Progressive**

A gradual test environment was simulated by using the nonlinear benchmark function, Bennett5, to generate a dataset of 10 000 data patterns with 10 timesteps. The Bennett5 function is computed as [52]:

$$f(x,\theta) = \theta_1 + (\theta_2 + x)^{-\frac{1}{\theta_3}} \qquad (2)$$

To simulate an environmental change, a drift (δ) was added to each parameter $\theta_i = \theta_i + \delta$ which was computed as:

$$f(\delta) = 0.6\delta^2 + 0.02\delta + 0.01$$

The impact of the drift increased as the number of timesteps increased. A sliding window of size of 1000 patterns slides from one timestep, which consists of 1000 patterns, to the next. The change period occurs at each timestep.

b) **Abrupt**

This test environment consists of a dataset of 8000 data patterns generated by four different target functions, each generating 2000 data patterns. Data patterns were recorded in a sequential order, block by block to form a dataset. A sliding window of analysis was set to 2000 data patterns which is the size of the block. The following artificially generated functions were used to simulate an abrupt test environment [53]:

i. $f_3(x_0,x_1,x_2,x_3) = x_0^5 - 5x_1^3 + 4x_2 + 5x_0^5 - 5x_1^3 + 4x_3$ with $x_0,x_1,x_2,x_3 \in [-2,2]$;

ii. $f_4(x_0,x_1,x_2,x_3) = \exp(2x_0 \sin(\pi x_3)) - \sin(x_1 x_2)$ with $x_0,x_1,x_2,x_3 \in [-1,1]$;

iii. $f_5(x_0,x_1,x_2,x_3) = x_0^5 + 5x_1^3 + 4x_2 - 5x_0^5 - 5x_1^3 + 4x_3$ with $x_0,x_1,x_2,x_3 \in [-2,2]$;

iv. $f_6(x_0,x_1,x_2,x_3) = \exp(2x_0 \sin(\pi x_3)) + \sin(x_1 x_2)$ with $x_0,x_1,x_2,x_3 \in [-1,1]$;

c) **Random**

A random test environment dataset was generated using Equation (2) and consists of 10 000 patterns with 10 timesteps. There was a 50% chance of modifying the environment in the current moment whereby each operator had an equal chance of being inverted. Thus, a negative member becomes positive and vice versa. As such, the environment was randomly modified.

d) **Recurrent**

Modification to the environment is realized by alternating two target functions. The target functions differ from each other in only one operator. The dataset consists of 10 000 data patterns. Each target function generated 5 000 data patterns. The data patterns were recorded

in sequential order, alternating as a block of 1 000 patterns, into a single dataset, block by block. The target functions were generated using:

i. $f_1(x_0, x_1) = x_0^5 - 5x_0^3 + 4x_0 + 5x_1^5 - 5x_1^3 + 4x_1$ with $x_0, x_1 \epsilon [-2, 2]$;

ii. $f_2(x_0, x_1) = x_0^5 - 5x_0^3 + 4x_0 - 5x_1^5 - 5x_1^3 + 4x_1$ with $x_0, x_1 \epsilon [-2, 2]$;

The size of the analysis window *(w_s)*, was set to the size of the concept block which was 1000 data patterns.

e) **Electricity Pricing**

Electricity pricing is a real-world dataset built on the electricity market in the Australian state of New South Wales [54]. Schedules for electricity prices and market prices are frequently updated for each power station. Pricing is determined by demand and supply forces. The demand is affected by the density population of the central business, time of the day, weather, and season. Also, the number of active generators affect the supply. The electricity pricing test environment exhibits both short-term irregular changes due to weather fluctuations and long-term regular changes due to seasonal changes [54].

To determine the electricity price, the current electricity demand is matched with the combination of all available power stations with the least expensive electricity. The task is to induce a predictive model to determine the electricity price given electricity demand estimates, time of the day, the day of the week, and transfer. Parameters were recorded every half an hour, for the period 7 May 1996 to 5 December 1998 to create a data set of 27552 data patterns. An analysis window size of 2500 data patterns and w_{shift} of $125, 250; 500; 1250$; and 2500 were implemented in this experiment.

f) **Trend**

Trend variation is a regular long-term change in the level of data which can be linear or non-linear. Trend variation tends to oscillate in a logically predictable pattern. A real-world data set, *historical US Treasury bill contracts,* for the period January 1984 to March 1986 was selected to depict trend variation [55]. The task is to induce a predictive model to determine the next day value given the current day value. An analysis window size of 111 data patterns and w_{shift} of $20; 42$; and 111 were implemented in this experiment.

g) **Stock Market**

The real-world stock market dataset, the Gross Domestic Product (GDP)(US) back-dating to 1951's third quarter was used to generate predictions for the first quarter of 1995 and right through to 2003's first quarter. All 30 economic indicators were used as explanatory variables [56]. The induced model is used to predict the quarterly GDP whenever historical data for at least one month of that quarter is available. The GDP dataset used in this experiment was normalized. An analysis window size of 50 data patterns and w_{shift} of $10; 20$; and 50 were implemented in this experiment.

Parameter Optimization

For each artificially generated dataset, a 10% relative (normal distributed) noise was added to the target output. The optimal values for each parameter of each algorithm implemented in these experiments were obtained using the *F-race algorithm* [57]. The optimized parameter values listed in Table 1 were used where culling, tournament and elitist were expressed as the percentage of the PopSize. The sizes of the two sliding windows of the DyFor GP were

initialized to 20% and 80% of sliding window size (w_s), the minimum window size was set to 5% of w_s and a minimum difference of two sliding windows was set to 2% of w_s. The sliding window size (w_s) was set either to 10% of the dataset size or the block size.

5 Result and Discussion

This section reports the results of applying the GP-REG to seven nonstationary datasets. The training algorithms' training and generalization performance for all test environments were reported in the subsequent tables. The *p-values* corresponding to the comparison of the training algorithms on $E_{MS}T$ (training) and $E_{MS}G$ (generalization) for 30 independent runs, only for scenarios that include a value of $p > 0.05$, were reported in Appendix 1.

Table 1 Parameter Values

GP-REG	GP	P_c	0.8
		P_m	0.1
		P_r	0.1
		$PopSize$	100
		$Elitist$	5%
		$Culling$	50%
		$Tournament$	20%
		π_ω	50
		n_s	10%
	QPSO	$c_1 = c_2$	1.496180
		ω	0.729844
		r_{cloud}	2
		$swarm\_size$	50
	KCDCDynDE	$PopSize$	80
		P_c	0.2
		$[\beta_{min}, \beta_{max}]$	[0.1; 0.9]

Table 2 summarized the obtained average spatial severities for each test environment on $E_{MS}T$, $E_{MS}G$ and computation time ($\bar{t}$) in mins, for each algorithm where w_{shift}=*1* was the least w_{shift}. The following were observed for the results presented in Table 2.

As spatial severity increased: the performance of the GP-REG, on training, for all datasets deteriorated except for the most abrupt scenario, $w_{shift} = 5$ where the performance improved whereas, on generalization, the performance of the GP-REG generally improved as spatial severity increased.

In the Scenario, $w_{shift} = 5$, there were no conflicting boundaries in the sliding window of analysis. The performance of the DynGP was consistent on training and was reduced on generalization as spatial severity increased. As observed for the GP-REG, the DyFor GP performance was consistent on training and generalization, the performance generally improved as spatial severity increased.

The GP-REG obtained the least values of both $E_{MS}T$ and $E_{MS}G$ in all scenarios under consideration whereas the DynGP and the DyFor GP obtained competitive performance on $E_{MS}G$ for the following datasets: Progressive, Abrupt, and Electricity.

Table 2 Average Spatial Severities (w_{shift}) for each Test Environment

		DynGP					GP-REG					DyFor				
Features		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Prog	$E_{MS}T$	2.84	2.71	2.56	2.38	1.65	**0.006**	**0.002**	**2.6e-8**	**8.1e-4**	0.012	0.04	0.27	0.02	0.08	**0.01**
	$E_{MS}G$	4.51	5.36	5.05	2.30	2.32	**0.012**	**0.015**	**8.5e-4**	**3.1e-4**	**0.009**	4.82	4.27	4.25	2.49	2.45
	$\bar{t}$	**0.015**	**0.015**	**0.017**	**0.017**	**0.011**	14.15	7.732	2.947	1.036	0.493	2.95	1.48	0.37	0.20	0.11
Abrp	$E_{MS}T$	0.604	0.604	0.583	0.582	0.543	**0.1064**	**0.097**	**0.089**	**0.074**	**0.120**	0.230	0.232	0.231	0.231	0.232
	$E_{MS}G$	1.052	1.49	0.874	0.732	0.603	**0.1689**	**0.242**	**0.155**	**0.164**	**0.035**	0.759	0.757	0.747	0.717	0.605
	$\bar{t}$	**0.019**	**0.016**	**0.013**	**0.014**	**0.013**	14.83	7.287	1.841	1.092	0.542	2.932	1.517	0.383	0.202	0.103
Recur	$E_{MS}T$	0.031	0.023	0.072	0.095	0.084	**0.008**	**0.004**	**3.1e-6**	**4.7e-3**	0.012	0.024	0.353	0.353	0.353	**3.54e-1**
	$E_{MS}G$	7.7324	7.517	6.866	5.044	5.1555	**0.009**	**0.022**	**4.7e-3**	**2.4e-3**	**0.027**	7.448	4.809	5.257	2.688	3.460
	$\bar{t}$	**0.117**	**0.113**	**0.116**	**0.128**	**0.105**	49.81	25.41	8.499	3.663	1.872	27.67	14.12	4.722	2.035	1.04
Rand	$E_{MS}T$	2.91	2.85	2.90	2.97	2.89	**0.098**	**0.098**	**0.098**	**0.114**	**0.088**	0.327	0.318	0.294	0.261	0.248
	$E_{MS}G$	3.11	3.09	3.11	3.03	3.09	**0.178**	**0.225**	**0.245**	**0.353**	**0.149**	2.47	2.47	2.45	2.41	2.330
	$\bar{t}$	**0.143**	**0.145**	**0.134**	**0.106**	0.12	79.68	43.18	7.835	4.246	2.208	24.77	12.87	2.447	1.325	0.69
Elect	$E_{MS}T$	0.017	0.017	0.017	0.017	0.018	**1.9e-4**	**1.8e-4**	**2.3e-4**	**2.5e-4**	2.1e-4	0.013	0.013	0.013	0.014	0.014
	$E_{MS}G$	0.017	0.019	0.019	0.022	0.027	**1.7e-4**	**1.8e-4**	**2.0e-4**	**1.9e-4**	1.7e-4	0.017	0.019	0.021	0.023	0.029
	$\bar{t}$	**0.190**	**0.11**	**0.106**	**0.096**	0.108	96.32	48.16	16.73	7.27	3.24	60.2	30.1	10.02	4.657	2.122
Tred	$E_{MS}T$	0.071	0.071	0.070	0.072		**2.7e-5**	**2.7e-5**	**2.4e-5**	**3.2e-5**		0.005	0.005	0.006	0.006	
	$E_{MS}G$	0.078	0.075	0.079	0.119		**1.1e-7**	**4.8e-6**	**4.1e-6**	**2.9e-5**		0.006	0.007	0.007	0.007	
	$\bar{t}$	**0.091**	**0.083**	**0.089**	**0.115**		9.57	4.89	2.95	1.42		4.265	2.91	1.457	0.7125	
GDP	$E_{MS}T$	0.024	0.021	0.019			**0.005**	**0.011**	**1.7e-4**			0.004	0.001	0.001		
	$E_{MS}G$	0.041	0.045	0.074			**0.035**	**0.018**	**0.012**			0.100	0.075	0.171		
	$\bar{t}$	**0.099**	**0.131**	**0.090**			15.859	8.361	2.795			3.687	1.945	0.65		

The DyFor GP outperformed the DynGP on Random and Trend test environments whereas the DynGP outperformed the DyFor GP on GDP test environment. The DynGP obtained the least average computational time in all scenarios under consideration whereas the GP-REG obtained the highest average computational time in all scenarios. The computational time of GP-REG was significantly reduced for the following datasets: Recurrent, Trend, and Electricity. The reduction in computational time could have been attributed to the recurrent or cyclical variations happening in these datasets that promoted the retrieval of previously acquired knowledge.

Table 3 summarized the obtained average temporal severities for each test environment on training and generalization for each algorithm where frequency, $f = 1$ was the least frequency. The following were observed for the results presented in Table 3. As temporal severity increased: the performance of all training algorithms improved, on both training and generalization. The GP-REG exhibit the greatest improvement for both training and generalization to outperform all training algorithms. The DyFor GP exhibit superior performance to the DynGP. For GDP test environment, all training algorithm exhibit comparable performance on generalization. The DynGP obtained the least average computational time in all scenarios under consideration whereas the GP-REG obtained the highest average computational time in all scenarios.

Table 4 summarized the overall averages (both spatial and temporal severities) on each test environment for each algorithm. As reported in Table 4, GP-REG obtained the least values on both $E_{MS}T$ and $E_{MS}G$ for all test environments under consideration whereas the DynGP outperformed the DyFor GP on generalization on the following test environments: Progressive, Electricity and GDP. The reported results suggest the capability of the GP-REG to adapt a predictive model in a changing environment with concept drift occurring to yield improved performance.

Table 3 Average Temporal Severities (frequency) for each Test Environment

		DynGP				GP-REG				DyFor			
	Features	1	2	3	4	1	2	3	4	1	2	3	4
Prog	$E_{MS}T$	2.42	2.38	2.35	1.62	**0.005**	**9.3e-4**	**6.8e-3**	**6.1e-3**	0.09	0.08	0.08	0.06
	$E_{MS}G$	2.76	2.41	2.31	1.87	**0.002**	**2.9e-2**	**5.2e-4**	**4.7e-4**	2.49	2.49	2.76	2.35
	$\bar{t}$	**0.007**	**0.006**	**0.012**	**0.033**	0.726	2.845	4.963	12.14	0.149	0.531	0.992	2.428
Abrp	$E_{MS}T$	0.550	0.576	0.583	0.550	**0.191**	**0.097**	**0.089**	**0.13**	0.310	0.227	0.231	0.211
	$E_{MS}G$	0.740	0.735	0.734	0.734	**0.261**	**0.119**	**0.155**	**0.04**	0.718	0.719	0.747	0.72
	$\bar{t}$	**0.002**	**0.007**	**0.014**	**0.037**	0.701	2.361	4.977	12.26	0.147	0.518	0.994	2.452
Recur	$E_{MS}T$	0.087	0.040	0.095	0.117	**0.006**	**8.1e-4**	**7.3e-3**	**8.4e-3**	4.601	0.353	0.353	0.353
	$E_{MS}G$	0.353	4.364	5.044	4.905	**0.004**	**6.2e-2**	**8.5e-4**	**7.3e-4**	2.200	2.306	2.688	3.630
	$\bar{t}$	**0.018**	**0.063**	**0.107**	**0.276**	2.376	7.454	20.13	39.06	1.32	4.306	12.35	21.70
Rand	$E_{MS}T$	2.89	2.90	2.78	2.15	**0.097**	**0.097**	**0.064**	**0.114**	0.330	1.32	1.05	0.248
	$E_{MS}G$	3.12	3.11	2.71	2.59	**0.324**	**0.248**	**0.248**	**0.219**	2.39	2.30	2.16	2.13
	$\bar{t}$	**0.021**	**0.059**	**0.127**	**0.312**	3.467	11.23	20.81	71.29	1.082	3.51	6.818	22.28
Elect	$E_{MS}T$	0.017	0.017	0.017	0.018	**6.6e-4**	**1.7e-4**	**2.5e-4**	**2.1e-4**	0.014	0.014	0.014	0.014
	$E_{MS}G$	0.024	0.022	0.022	0.021	**5.6e-4**	**2.1e-4**	**1.9e-4**	**1.7e-4**	0.023	0.023	0.023	0.021
	$\bar{t}$	**0.018**	**0.043**	**0.122**	**0.306**	5.004	17.08	29.85	85.21	3.128	10.63	18.66	53.26
Tred	$E_{MS}T$	0.072	0.071	0.072	0.072	**3.1e-5**	**2.2e-5**	**2.1e-5**	**1.9e-5**	0.006	0.005	0.005	0.005
	$E_{MS}G$	0.081	0.079	0.078	0.078	**2.8e-5**	**4.5e-6**	**1.0e-7**	**1.1e-34**	0.007	0.007	0.007	0.007
	$\bar{t}$	**0.013**	**0.046**	**0.084**	**0.235**	1.077	2.751	5.531	11.29	0.405	1.365	2.281	5.375
GDP	$E_{MS}T$	0.026	0.024	0.023	0.023	**0.010**	0.001	**1.9e-4**	**1.5e-4**	0.001	**5.2e-4**	4.0e-4	2.7e-4
	$E_{MS}G$	0.052	0.046	0.041	0.014	**0.051**	**0.043**	**0.034**	**0.014**	0.054	0.051	0.049	0.047
	$\bar{t}$	**0.013**	**0.043**	**0.114**	**0.256**	0.567	4.809	8.317	23.74	0.132	1.03	1.98	5.233

The DyFor GP outperformed the DynGP on both training and generalization on the following test environments: Abrupt, Random and Trend. The DynGP obtained comparable performance to the GP-REG on generalization for GDP test environment. The reduced performance of the GP-REG could have been attributed to the reduced size of the analysis window which suggests that the GP-REG requires a sufficiently large size of data patterns to induce optimal models.

Table 4 Averages and Standard Deviation for each Test Environment

	Features	DynGP	GPREG	DyFor
Progressive	$E_{MS}T$	2.3233±0.4791	**0.0041±0.0038**	1.1177±2.7200
	$E_{MS}G$	3.2100±1.3586	**0.0077±0.0097**	3.4855±1.6788
	$\bar{t}$	**0.0154**	5.1869	1.0253
Abrupt	$E_{MS}T$	0.5783±0.0207	**0.1103±0.0346**	0.2397±0.0263
	$E_{MS}G$	0.8704±0.2649	**0.1488±0.0771**	0.7210±0.0468
	$\bar{t}$	**0.0155**	5.4397	1.0278
Recurrent	$E_{MS}T$	0.0718±0.0327	**0.0056±0.0037**	0.7890±1.4339
	$E_{MS}G$	5.2205±2.2044	**0.0147±0.0200**	3.8328±1.7264
	$\bar{t}$	**0.1164**	17.8956	9.9195
Random	$E_{MS}T$	2.8144±0.2513	**0.1114±0.0325**	0.4884±0.4019
	$E_{MS}G$	3.0511±0.1306	**0.1916±0.0911**	2.3788±0.0973
	$\bar{t}$	**0.1299**	26.1932	8.4225
Electricity	$E_{MS}T$	0.0172±0.0004	**0.0002±0.0001**	0.0136±0.0005
	$E_{MS}G$	0.0217±0.0034	**0.0002±0.0001**	0.0230±0.0040
	$\bar{t}$	**0.1223**	34.2736	21.421
Trend	$E_{MS}T$	0.0714±0.0007	**2.48e-5±4.62E-6**	0.0054±0.0005
	$E_{MS}G$	0.0833±0.0135	**8.34e-6±1.16E-5**	0.0068±0.0003
	$\bar{t}$	**0.0948**	7.9424	2.336
GDP	$E_{MS}T$	0.0230±0.0020	**0.0055±0.0072**	0.0009±0.0011
	$E_{MS}G$	0.0467±0.0154	**0.0464±0.0775**	0.0920±0.0397
	$\bar{t}$	**0.1071**	9.7542	2.094

Figures 2 is a graphical illustration as observed under spatial severities for all test environments on generalization. For Progressive test environment, the GP-REG detects a change in the environment and adapts the model as spatial severity increased, evident with reduced $E_{MS}G$. As a result, the GP-REG consistently and significantly outperformed both the DynGP and the DyFor GP. Also, the performance of both the DynGP and the DyFor GP greatly improved as spatial severity increased to yield comparable results for higher values of spatial severity. The abrupt changes happening in the scenario, $w_{shift} = 5$ made this scenario much easier to adapt because the sliding window discarded all patterns from the past data generating processes. Hence, all training algorithms yielded improved performance.

For Abrupt test environment, as observed for Progressive test environment, the GP-REG outperformed both the DynGP and the DyFor GP. However, the GP-REG exhibit performance deterioration for $w_{shift} = 2$ and $w_{shift} = 4$. Generally, the performance of the DyFor GP was consistent as spatial severity increased.

The performance of the DynGP and the DyFor GP was consistent for all cases of spatial severity illustrated in Figure 2 for the Random test environment. The DynGP exhibit the worst performed whereas the GP-REG exhibit superior performance. There was a slight performance deterioration on $w_{shift} = 4$ for the GPREG. The DynGP exhibit significant worst performance for all cases on Trend test environment whereas the DyFor GP and the GP-REG exhibit consistent performance as spatial severity increased. Thus, the DynGP predictive models failed to effectively exploit, as the other training algorithms, fruitful areas of the search space. The DyFor GP yielded an improved performance which was competitive to the GP-REG.

Both GDP and Electricity test environment, the performance of the DynGP and the DyFor GP deteriorated as spatial severity increased. However, the performance of the GP-REG improved as spatial severity increased on GDP test environment and exhibit superior performance which was maintained as spatial severity increased.

Figures 3 is a graphical illustration as observed under temporal severities for all test environments on generalization. For Progressive test environment, as temporal severity increased, the performance of GP-REG was consistent which suggest the capability of GP-REG to detect a change in the environment and adapts the model even for lower frequencies. As a result, GP-REG consistently and significantly outperformed both the DynGP and the DyFor GP. The performance of both the DynGP and DyFor GP improved as temporal severity increased since there were sufficient iterations for each algorithm to converge toward the optimal solution. For both Abrupt and Random test environment, the performance of the DynGP and the DyFor GP improved as temporal severity increased. The GP-REG exhibit superior performance to outperform all training algorithm as temporal severity increased whereas the DyFor GP outperformed the DynGP.

As observed under spatial severity, the DynGP exhibit significant worst performance for all cases on Trend test environment whereas the DyFor GP and the GP-REG exhibit consistent performance as temporal severity increased. The performance of the GP-REG greatly deteriorated on GDP test environment whereas the DynGP improved to yield outstanding performance to outperform all training algorithm as temporary severity increased.

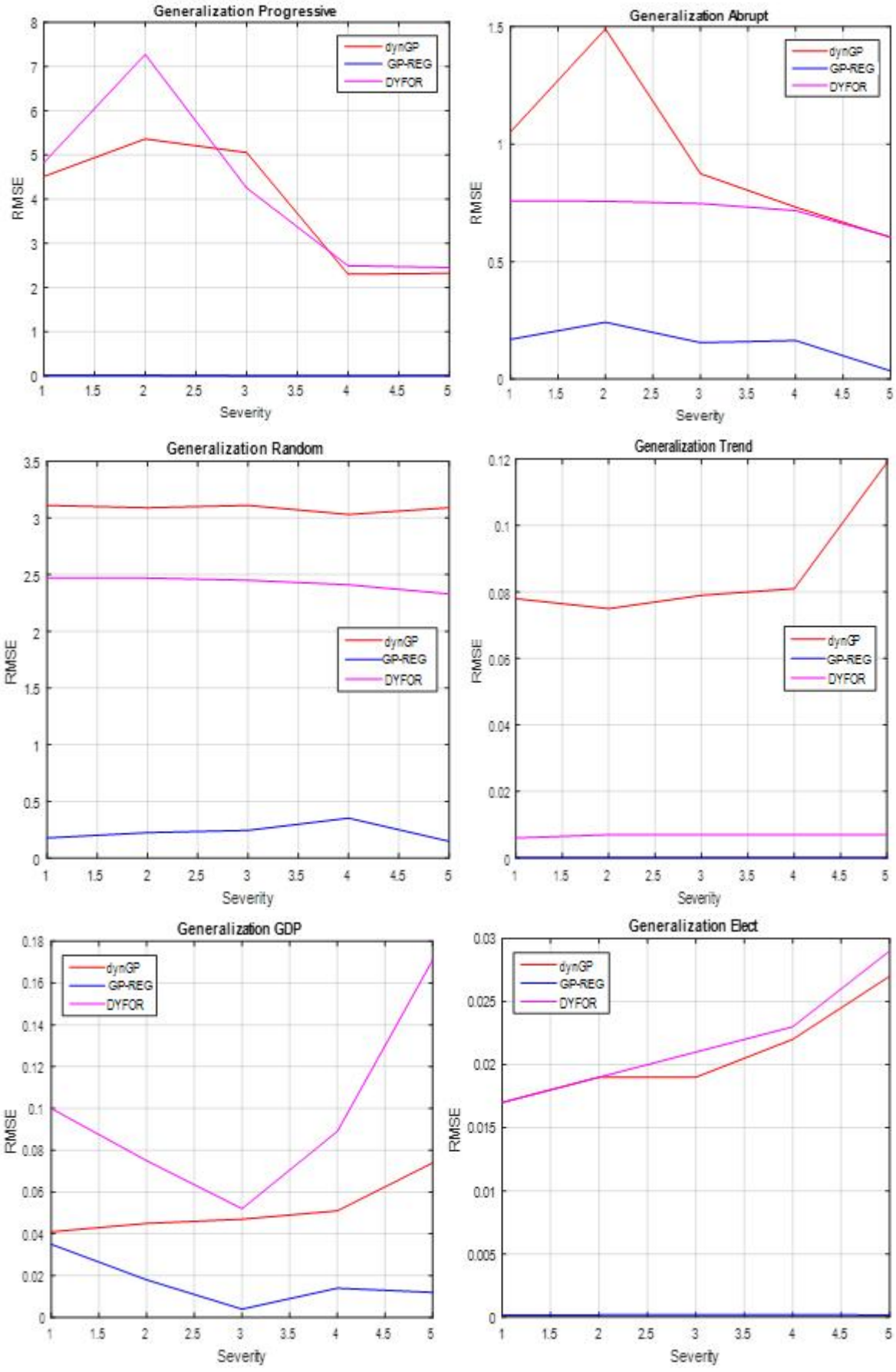

Figure 2. Averages for RMSE on Generalization per Spatial Severity

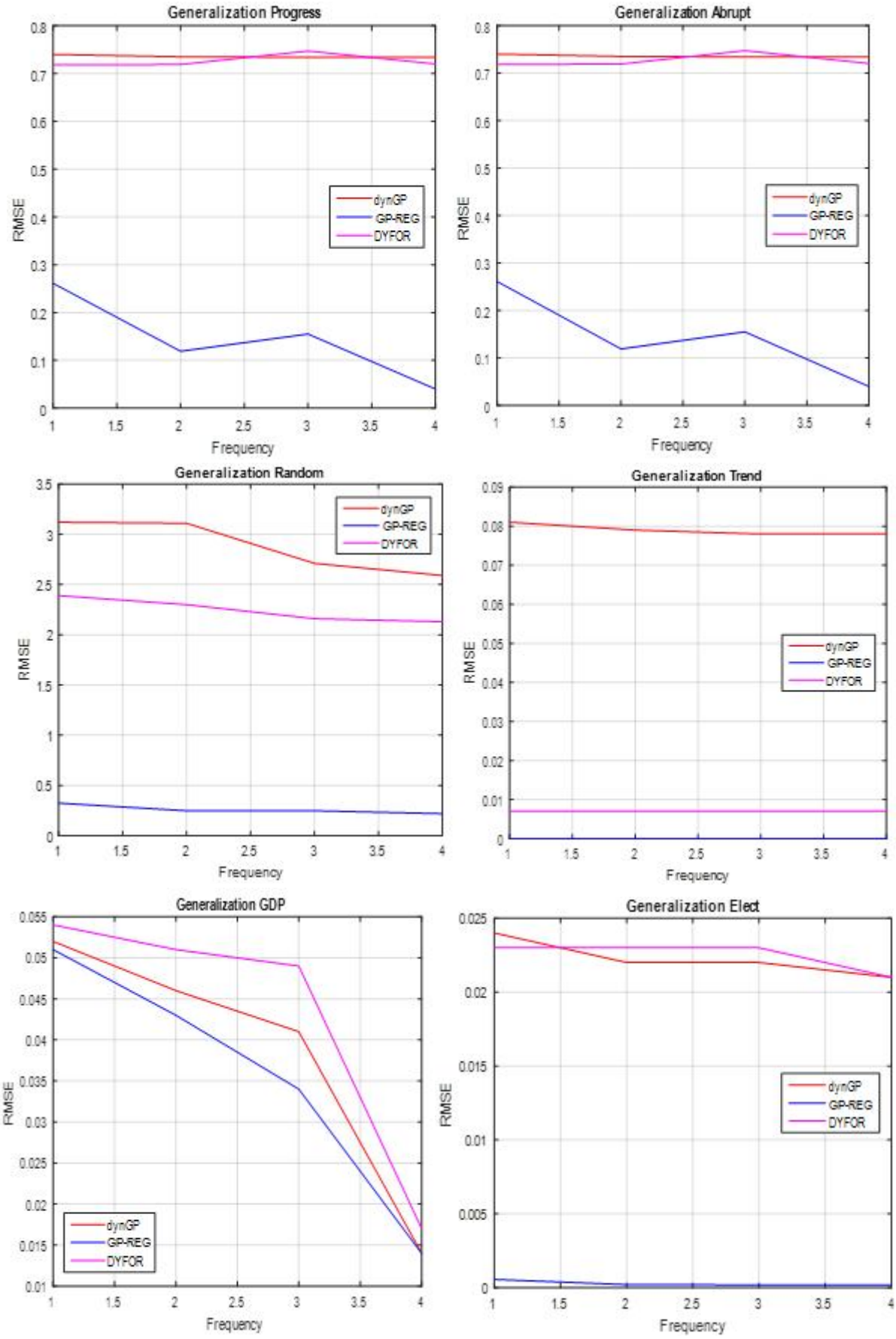

Figure 3. Averages for RMSE on Generalization per Temporal Severity

For Electricity test environment, the performance of the DyFor GP deteriorated as temporal severity increased whereas the DynGP improved to outperform the DYFOR for all cases. The performance of the GP-REG slightly improved as temporal severity increased to exhibit superior performance which was maintained as temporal severity increased.

Table 5 summarized the predictive accuracy of the GP-REG to the state-of-the-art techniques: autoregressive (AR), real-time forecasting system (RTFS), and DyFor GP on the real-world dataset GDP and CPI inflation reported in [28]. The experiments for GP-REG were implemented as described in [28]. As observed in resulted presented in Table 5, GP-REG yielded outperformed other training algorithms.

Table 5: RMSE on a GDP and CPI inflation datasets for the state-of-art techniques

Dataset	AR	RTFS	DyFor GP	GP-REG	CPC
GDP	2.46	1.85	1.57	**0.88**	
CPI			2.40	**2.05**	2.30

The state-of-the-art techniques: random vector functional link network (RVFL) and incremental DWT-EMD-RVFL were implemented on the Australian Energy Market Operator, New South Wales (NSW), 2015 electric load dataset for the following months: January, April, July and October, to perform forecasting for each of the 24 hours of next day using the load value at the same hour of last day [62]. Also, support vector regression (SVR) and the hybrid, support vector regression-ARIMA (SVRARIMA) techniques were implemented to the California electricity market to perform next-week prices forecasting [63]. The experiments for GP-REG were implemented as described in [62] and [63].

Table 6 presents the obtained results of the state-of-the-art techniques and the GP-REG for NSW, 2015 load dataset and the California electricity market dataset. The GP-REG exhibited an outstanding performance to outperform all models under consideration except for DWT-EMD-RVFL in which it performed in the same error range to obtain comparable performance.

Table 6. Forecasting results for Electricity Data

Dataset	Metric	RVFL	DWT-EMD-RVFL	GP-REG	SVR	SVRARIMA
Jan	MAPE (%)	3.87	1.86	**1.72**		
	RMSE	428.908	193.80	**189.07**		
April	MAPE (%)	3.94	2.03	**2.02**		
	RMSE	425.23	212.70	**197.43**		
Jul	MAPE (%)	5.09	**2.96**	2.98		
	RMSE	493.06	**296.74**	299.57		
Oct	MAPE (%)	8.86	5.93	**5.91**		
	RMSE	1004.39	659.41	**596.48**		
1st Week	MAPE (%)			**258.72**	758.69	348.81
	RMSE			**0.57**	1.44	0.75
2nd Week	MAPE (%)			**460.46**	969.37	514.29
	RMSE			**0.87**	2.05	1.07

6 Conclusion

The three training algorithms have a different level of precision and computational load. Generally, the GP-REG exhibit outstanding performance for all test environment to yield reduced $E_{MS}G$ as the temporal and spatial severity increased. The superior performance of the GP-REG was attributed to the algorithm's ability to detect an environmental change, track changing decision boundaries, and adapt a predictive model using the prevailing patterns even in increased temporal and spatial severities. However, GP-REG suffers from computational

load due to the embedded clustering and the nonlinear model induction. Consequently, the DynGP had the least precision with the best computational performance whereas the DyFor GP provides a balance between precision and computation performance.

Future will consider improving the computational load of the GP-REG and also, to strengthen the adaptation to concept drift and the fast implementation of GP on GPUs to provide fast learning for high-speed temporal data. Also, a comparative study of the GP-REG to expand the benchmarks evaluated with many more combinations of types and speeds of concept drift (recurrent, incremental, gradual, mixed, including generators).

Acknowledgements

The authors would like to thank reviewers for their valuable comments which immensely improved the structure of this work.

References

[1] A. Tsymbal, "The Problem of Concept Drift: Definitions and Related Work," *Computer Science Department, Trinity College Dublin*, 106(2), pp.58, 2004.

[2] T. Mitsa, Temporal Data Mining, Chapman&Hall / CRC Data Mining and Knowledge Discovery Series, 2010.

[3] J. Brownlee, "A Gentle Introduction to Concept Drift in Machine Learning," *Machine Learning Mastery*, 2018.

[4] L. Khan, and W. Fan, "In International Conference on Database Systems for Advanced Applications," In *Tutorial: Data stream mining and its applications*, Berlin, Heidelberg, Springer, 2012, pp. 328-329.

[5] E. Lughofer, "On-line active learning: A new paradigm to improve practical useability of datastream modeling methods," *Information Sciences*, vol. 415, pp. 356–376, 2017.

[6] Z. Zhang, and J. Zhou, "Transfer estimation of evolving class priors in data stream classification," *Pattern Recognition*, vol. 43, no. 9, pp. 3151–3161, 2010.

[7] J. Gama, I. Žliobaite, ˙ A. Bifet, M. Pechenizkiy, A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[8] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw*, vol. 22, no. 10, pp. 1517–1531, 2011.

[9] C. Alippi, G. Boracchi, and M. Roveri, "Just in time classifiers: Managing the slow drift case," *Proc. Int. Joint Conf. Neural Networks*, pp. 114–120, 2009.

[10] L. Torrey, and J. Shavlik, "Transfer Learning," In *Handbook of Research on Machine Learning Applications*, J. M. R. M. M. M. a. A. S. E. Soria, Ed., IGI Global, 2009.

[11] J.C. Schlimmer and R.H. Granger, "Incremental learning from noisy data.," *Machine Learning*, vol. 1, no. 3, pp. 317-354, 1986.

[12] G. Widmer and, M. Kubat, "Learning in the presence of concept drift and hidden contexts.," *Machine learning*, vol. 23, no. 1, pp. 69-101, 1996.

[13] G. Ditzler, M. Roveri, and C. Alippi, "Learning in Nonstationary Environments: A survey," *IEEE Computational Intelligence Magazine*, pp. 12-25, November 2015.

[14] S. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, " A case-based technique for tracking concept drift in spam filtering," *Knowl. Based Syst,* vol. 18, no. 4-5, pp. 187-195, 2005.

[15] C. Alippi, Intelligence for Embedded Systems., Berlin, Germany: Springer-Verlag, 2014.

[16] J. Sarnelle, A. Sanchez, R. Capo, J. Haas, and R. Polikar, "Quantifying the limited and gradual concept drift assumption," In *2015 International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, IEEE, 2015.

[17] M. Basseville and I.V. Nikiforov, Detection of Abrupt Changes: Theory and Application, vol. 104, Englewood Cliffs, NJ: Prentice-Hall, 1993.

[18] L.I. Kuncheva, "Classifier ensembles for changing environments," *Proc. 5th Int Workshop of Multiple Classifier Systems,* Vols., pp. 1-15, 2004.

[19] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Inform. Fusion,* vol. 9, no. 1, pp. 56–68, 2008.

[20] J. R. Koza, "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," *Stanford University Computer Science Department Technical Report STAN-CS-90-1314,* 1990.

[21] S. Massimo, and A. Tettamanzi, "Genetic programming for financial time series prediction," *Genetic Programming, Springer,* pp. 361-370, 2001.

[22] M. Kľúčik, J. Juriova, and M. Kľúčik, "Time Series Modeling with Genetic Programming Relative to ARIMA Models," In *Conferences on New Techniques and Technologies for Statistics,* pp. 17-27, 2009.

[23] P.G. Espejo, S. Ventura, and F. Herrera, "A Survey on the Application of Genetic Programming to Classification," *IEEE Trans. on Systems, Man, and Cybernetics, Part C, Applications and Reviews,* vol. 40, no. 2, pp. 121–144, 2010.

[24] K. Nag, and N. Pal, "A Multiobjective Genetic Programming-Based Ensemble for Simultaneous Feature Selection and Classification," *IEEE Trans. on Cybernetics,* vol. 46, no. 2, pp. 499–510, 2016.

[25] L. Vanneschi and G. Cuccu, "A study of genetic programming variable population size for dynamic optimization problems," In *IJCCI,* pp. 119-126, 2009.

[26] Z. Yin, A. Brabazon, C. O'Sullivan, and M. O'Neill, "Genetic programming for dynamic environments," In *2nd International Symposium Advances in Artificial Intelligence and Applications,* vol. 2, pp. 437 - 446.

[27] M. Rieket, K. M. Malan, and A. P. Engelbrecht, "Adaptive Genetic Programming for Dynamic Classification Problems," In *2009 IEEE Congress on Evolutionary Computation,* pp. 674-681, 2009.

[28] N. Wagner, Z. Michalewicz, M. Khouja, and R. McGregor, "Time Series Forecasting for Dynamic Environments: the DyFor Genetic Program Model," *IEEE Transactions on Evolutionary Computation,* 11(4), pp.433-452, 2007.

[29] N.R. Draper, and H. Smith, Applied regression analysis, vol. 326, John Wiley & Sons, 1998.

[30] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed., Springer Series in Statistics, Springer, 2009.

[31] J.B. Fraleigh, and R.A. Beauregard, Linear Algebra, 3. edition, Ed., Upper Saddle River, NJ: Addison-Wesley Publishing Company, 1995.

[32] S. M. Stigler, "Gauss and the Invention of Least Squares," *Ann. Stat.*, vol. 9, no. 3, pp. 465–474, 1981.

[33] A. Kordon, "Future Trends in Soft Computing Industrial Applications," *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pp. 7854-7861, 2006.

[34] E. Alfaro-Cid, A.I. Esparcia-Alcázar, P. Moya, B. Femenia-Ferrer, K. Sharman and J.J. Merelo, "Modeling pheromone dispensers using genetic programming," *In Lecture Notes in Computer Science, SpringerBerlin / Heidelberg*, vol. 5484, pp. 635-644, 2009.

[35] D.P. Searson, D.E. Leahy, and M.J. Willis, "GPTIPS: an open-source genetic programming toolbox for multigene symbolic regression," *In Proceedings of the International multiconference of engineers and computer scientists, Citeseer*, vol. 1, pp. 77-80, 2010.

[36] N.Q. Uy, N.X. Hoai, M. O'Neill, R.I. McKay, and E. Galván-López, "Semantically-based crossover in genetic programming: application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91-119, 2011.

[37] K. Georgieva and A.P. Engelbrecht, "Dynamic Differential Evolution Algorithm for Clustering Temporal Data," *Large Scale Scientific Computing, Lecture Notes in Computer Science*, vol. 8353, pp. 240-247, 2014.

[38] C. Kuranga, Genetic Programming Approach for Nonstationary Data Analytics, PhD Thesis, University of Pretoria, Pretoria, South Africa, 2020.

[39] R. Poli, W. B. Langdon, and N. F. McPhee, A Field Guide to Genetic Programming, Lulu Enterprise, UK Ltd: http://lulu. com, 2008.

[40] L. Vanneschi and R. Poli, "Genetic Programming: Introduction, Application, Theory and Open Issues," in *Handbook of Natural Computing: Theory, Experiments and Applications*, Springer Verlag ed., T. B. a. J. K. Grzegorz Rosenberg, Ed., Springer Verlag, 2010.

[41] W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone, Genetic Programming: An Introduction, vol. volume 1, Morgan Kaufmann San Francisco, 1998.

[42] A. Canoa, and B. Krawczyk, "Evolving Rule-Based Classifiers with Genetic Programming on GPUs for Drifting Data Streams," *Pattern Recognition*, pp.248-268, 2019.

[43] A. Soundarrajan, S. Sumathi, and G. Sivamurugan, "Voltage and frequency control in power generating system using hybrid evolutionary algorithms," *Journal of Vibration and Control*, 18(2), pp.214-227, 2012.

[44] "Mathworks," MATLAB, [Online]. Available: www.mathworks.com.

[45] R.W. Morrison, "Performance Measure in Dynamic Environments," In *GECCO workshop on evolutionary algorithms for dynamic optimization problems* (No. 5-8), 2003.

[46] R.W. Morrison and K.A. De Jong "A test problem generator for non-stationary environments," *Proc. of the 1999 Congr. on Evol. Comput.*, pp. 2047-2053, 1999.

[47] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," *Proc. of the 1999 Congr. on Evol. Comput*, pp. 1875-1882, 1999.

[48] Y. Jin, and B. Sendhoff, "Constructing dynamic optimization test problems using the multiobjective optimization concept," *EvoWorkshop 2004, LNCS 3005*, pp. 526-536, 2004.

[49] C. Li, M. Yang, and L. Kang, "A new approach to solving dynamic TSP," *Proc of the 6th Int. Conf. on Simulated Evolution and Learning*, pp. 236-243, 2006.

[50] C. Li and S. Yang "A Generalized Approach to Construct Benchmark Problems for Dynamic Optimization," *Proc. of the 7th Int. Conf. on Simulated Evolution and Learning*, pp. 391-400, Springer, Berlin, Heidelberg, 2008.

[51] L. Zhang, J. Lin, and R. Karim, "Sliding Window-Based Fault Detection From High-Dimensional Data Streams," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 289–303, 2017.

[52] L. Bennett, L. Swartzendruber, and H. Brown, "Superconductivity Magnetization Modeling," *National Institute of Standards and Technology (NIST), US Department of Commerce, USA*, 1994.

[53] V. Cherkassky, D. Gehring, and F. Mulier, ", Comparison of adaptive methods for function estimation from samples," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 969- 984, 1996.

[54] M. Harries, "Splice-2 comparative evaluation: Electricity pricing. Technical Report UNSW-CSE-TR-9905," *Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney 2052, Australia*, 1999.

[55] R.J. Shiller, "Stock Market Data used in Irrational Exuberance," *Princeton University Press*, 2005.

[56] J. Kitchen and R. Monaco, "Real-time Forecasting in Practice," *Business Economics: The Journal of the National Association of Business Economists*, vol. 38, pp. 10-19, 2003.

[57] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari, "The irace package: Iterated Racing for Automatic Algorithm Configuration," *Operations Research Perspectives*, vol. 3, pp. 43-58, 2016.

[58] G. Brown, J.L. Wyatt and P. Tino, "Managing diversity in regression ensembles," *Journal of Machine Learning Research*, 6(Sep), pp. 1621-1650, 2005.

[59] S. Kelly, J. Newsted, W. Banzhaf, and C. Gondro. "A modular memory framework for time series prediction," In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 949-957, June 2020.

[60] A.J. Turner, and J.F. Miller, "Recurrent cartesian genetic programming of artificial neural networks, " *Genetic Programming and Evolvable Machines*, 18(2), 185-212, 2017

[61] A.S. Rakitianskaia, and A.P. Engelbrecht, "Training Feedforward Neural Network with Dynamic Particle Swarm Optimisation," *Computer Science Department, University of Pretoria,* 2011.

[62] Qiu, X., Suganthan, P. N., & Amaratunga, G. A. (2018). Ensemble incremental learning random vector functional link network for short-term electric load forecasting. *Knowledge-Based Systems*, 145, 182-196.

[63] Che, J., & Wang, J. (2010). Short-term electricity prices forecasting based on support vector regression and auto-regressive integrated moving average modeling. *Energy Conversion and Managemen*t, 51(10), 1911-1917.

Appendix 1: The *p*-values for the Test Environment

		DynGP vs. DyFor GP - Progressive				DynGP vs. DyFor GP - Abrupt				
		1	2	3	4	1	2	3	4	5
A	$E_{MS}T$					0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$					0.0001	0.0001	0.0001	0.0001	**0.0648**
B	$E_{MS}T$	0.0001	**0.2825**	0.0038	0.0024	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0105	0.0163	0.0001	0.0001	0.0001	0.0001	0.0001	**0.0718**
C	$E_{MS}T$					0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$					0.0001	0.0001	0.0001	0.0001	**0.1068**
D	$E_{MS}T$					0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$					0.0001	0.0001	0.0001	0.0001	**0.2413**

		DynGP vs. DyFor GP - Chaotic				DynGP vs. DyFor GP - Elect				
		1	2	3	4	1	2	3	4	
A	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	**0.0511**	**0.2175**	**0.1217**	**0.2458**	0.0351
	$E_{MS}G$	**0.0853**	0.0001	0.0001	0.0001	**0.1094**	**0.3445**	0.0362	0.0023	0.0014
B	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	**0.6284**	**0.9582**	0.0412	0.0481	0.0007
	$E_{MS}G$	**0.3610**	**0.7731**	0.0001	0.0001	0.0396	0.0264	0.0415	**0.0893**	**0.0719**
C	$E_{MS}T$					0.0250	0.0374	0.0116	0.0317	0.0428
	$E_{MS}G$					**0.4475**	**0.2856**	0.0386	0.0432	**0.3822**
D	$E_{MS}T$					0.0001	0.0001	0.0001	0.0379	0.0411
	$E_{MS}G$					0.0001	0.0001	0.0001	**0.5140**	**0.3947**