Genetic Programming Approach for Non-Stationary Data Analytics

By

Cry Kuranga

Submitted in fulfilment of the requirements for the degree of
Philosophiae Doctor
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria

December 2020

ABSTRACT

Nonstationary data with concept drift occurring is usually made up of different underlying data generating processes. Therefore, if the knowledge of the existence of different segments in the dataset is not taken into consideration, then the induced predictive model is distorted by the past existing patterns. Thus, the challenge posed to a regressor is to select an appropriate segment that depicts the current underlying data generating process to be used in a model induction.

The proposed genetic programming approach for nonstationary data analytics (GPANDA) provides a piecewise nonlinear regression model for nonstationary data. The GPANDA consists of three components: dynamic differential evolution-based clustering algorithm to split the parameter space into subspaces that resemble different data generating processes present in the dataset; the dynamic particle swarm optimization-based model induction technique to induce nonlinear models that describe each generated cluster; and dynamic genetic programming that evolves model trees that define the boundaries of nonlinear models which are expressed as terminal nodes.

If an environmental change is detected in a nonstationary dataset, a dynamic differential evolution-based clustering algorithm clusters the data. For the clusters that change, the dynamic particle swarm optimization-based model induction approach adapts nonlinear models or induces new models to create an updated genetic programming terminal set and then, purple the genetic programming evolves a piecewise predictive model to fit the dataset.

To evaluate the effectiveness of GPANDA, experimental evaluations were conducted on both artificial and real-world datasets. Two stock market datasets, GDP and CPI were selected to benchmark the performance of the proposed model to the leading studies. GPANDA outperformed the genetic programming algorithms designed for dynamic environments and was competitive to the state-of-art-techniques.

Keywords: nonstationary data, concept drift, nonlinear model, model trees, dynamic data clustering, predictive model, dynamic particle swarm optimization, dynamic genetic programming, dynamic differential evolution.

Supervisor : Prof. N. Pillay

Department : Department of Computer Science

Degree : Philosophiae Doctor

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Declaration: Plagiarism

I, Cry Kuranga, declare that:

(i) the research reported in this thesis, except where otherwise indicated or acknowledged, is my original work;

(ii) this thesis has not been submitted in full or in part for any degree or examination to any other university;

(iii) this thesis does not contain other persons data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons;

(iv) this thesis does not contain other persons writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

(a) their words have been re-written but the general information attributed to them has been referenced;

(b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced;

(v) this thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signature: . [illegible] ...... 19-12-2020.........

Cry Kuranga Date

Declaration: Supervisor

I confirm that this work was done under my supervision and it is the candidate's original work. As the candidate's supervisor, I have approved this thesis for submission.

Signature:	....
Prof. N. Pillay	Date

Declaration: Publications

The following publications are associated with the research presented in this thesis:

(1) Chapter 5: Kuranga C., Pillay N. (2020) "Regression in Dynamic Environments Using Particle Swarm Optimization". In: *MartÃn-Vide C., Vega-RodrÃguez M.A., Yang MS. (eds) Theory and Practice of Natural Computing. TPNC 2020. Lecture Notes in Computer Science*, vol 12494., pp. 133-144, Springer, Cham. `https://doi.org/10.1007/978-3-030-63000-3_11`

(2) Chapter 6: C. Kuranga and N. Pillay (2020), "Genetic Programming Approach for Nonstationary Data Analytics" , (Under review-*Genetic Programming & Evolvable Machine*)

(3) Chapter 5: C. Kuranga and N. Pillay (2020), "A Comparative Study of Nonlinear Regression and Autoregressive Techniques in Hybrid with Particle Swarm Optimization for Time-series Forecasting" , (Under review-*Expert Systems with Applications*)

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. N. Pillay for her patience, assistance and continuous guidance during my studies.

My dear wife, Florence Kuranga, who continuously supported me throughout the journey. Her remarks and encouragement reinvigorated me, especially during the trying times which posed frustrating moments of despair and despondency.

My son, Cryson and daughter, Crystal may not remain unmentioned.

Contents

List of Figures

List of Tables

List of Acronyms

GP	Genetic Programming
GPANDA	Genetic Programming Approach for Nonstationary Data Analytics
PSO	Particle Swarm Optimisation
DynPSO	Dynamic PSO Particle Swarm Optimisation
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error
SSE	Sum of Squared Errors
ANN	Artificial Neural Networks
DE	Differential Evolution
KCDCDynDE	K-independent Cooperative Data Clustering DE Evolution
DynGP	Dynamic Genetic Programming
DyFor GP	Dynamic Forecasting Genetic Programming
SVM	Support Vector Machine
AR	Autoregressive
QR	QR Decomposition
DynPSO	dynamic PSO-based nonlinear regression
NARX	Nonlinear AR with exogenous inputs model
NSW	New South Wales
AEMO	Australian Energy Market Operator
GDP	Gross Domestic Product
CPI	Consumer Price Index inflation rate
QPSO	quantum-inspired Particle Swarm Optimization
mPSO	multi-swarm Particle Swarm Optimization
rePSO	re-initialized Particle Swarm Optimization
QR-QPSO	QR-based quantum-inspired Particle Swarm Optimization
QR-mPSO	QR-based multi-swarm Particle Swarm Optimization
QR-rePSO	QR-based re-initialized Particle Swarm Optimization
E_{MS}	Mean Square Error
SVR	Support Vector Regression
RVFL	Random Vector Functional Link network

SA	South Australia
QLD	Queensland
TAS	Tasmania
lBest	local Best position
pBest	personal Best position
gBest	global Best position
$E_{MS}T$	training mean square error
$E_{MS}G$	generalization mean square error
SVR	Support Vector Regression
RVFL	Random Vector Functional Link network
ARIMA	Autoregressive Integrated and Moving Average
GLMLF-B	General Linear Model based Load Forecaster-Benchmark
EMD	Empirical mode decomposition
DWT	Discrete Wavelet Transform

Chapter 1

Introduction

Numerous data analytic approaches exist for stationary data where a static prediction model is constructed once and used to [55]:

(a) predict for instances not used during training; and

(b) extract knowledge from the underlying data.

For example, predicting the price of an apartment given the features such as apartment size, location and status. In this instance, the dataset is stationary. However, concepts in real-world are nonstationary (or drifting or evolving), thus, they change with time. Also, the underlying data distribution may change [1]. Such changes make a model built on older historical data inconsistent with the recent historical data. As such, it becomes necessary to regularly update the model. Also, if the target concept is stationary and only data distribution changes, updating the current model to be suitable with the new data distribution is still essential to reduce the variance of error [2].

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) or availability of alternatives; electricity load that may vary due features that affect demand and supply such as climatic seasons or availability of alternatives such solar power; the inflation rate, among many others [3]. Temporal data is usually made up of generating processes which change over the time where order and time are usually the fundamental elements that are central to the meaning of the data. Also, temporal data is characterized by a high amount of dependency among itself in which appropriate treatment of these dependencies or relationships is essential.

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.

Learning in nonstationary environments can be considered as a framework in which numerous problem domains and machine learning concepts can be listed [55]. Initially, a learning modality such as supervised, unsupervised or semi-supervised is selected [201] [202]. How data arrives, either online [205] or incremental is then considered [203] [204]. However, traditionally, these modalities are based on stationarity assumption.

Concept drift is a phenomenon where decision boundaries change due to changes in the underlying patterns which lead to changes in the target concepts [4]. Concept drift makes the task of learning a model very complicated due to the hidden cause of change which is not known a priori. Therefore, special approaches are sought that treat the most recent historical occurrences as equally significant contributors to the final concept. Figure 1.1 graphically illustrates the mindmap of concept drift [55]. Also, Figure 1.1 illustrates connection between different areas within machine learning and concepts drift including the applications that involve concept drifts.

Machine learning is a branch of artificial intelligence that applies algorithms to automatically learn from historical data or past experience without explicitly programmed. Thus, machine learning infers knowledge from data through empirical learning e.g. regression, learning association, classification, prediction, image processing, medical diagnosis etc. [5]. An empirical learning strategy is supervised if training examples have known targets or labels. Supervised learning is tailored to solve classification and regression problems. In classification problems, the supplied example labels are classes (categorical). In regression problems, the supplied example labels are numeric. The focus of this thesis is on regression problems.

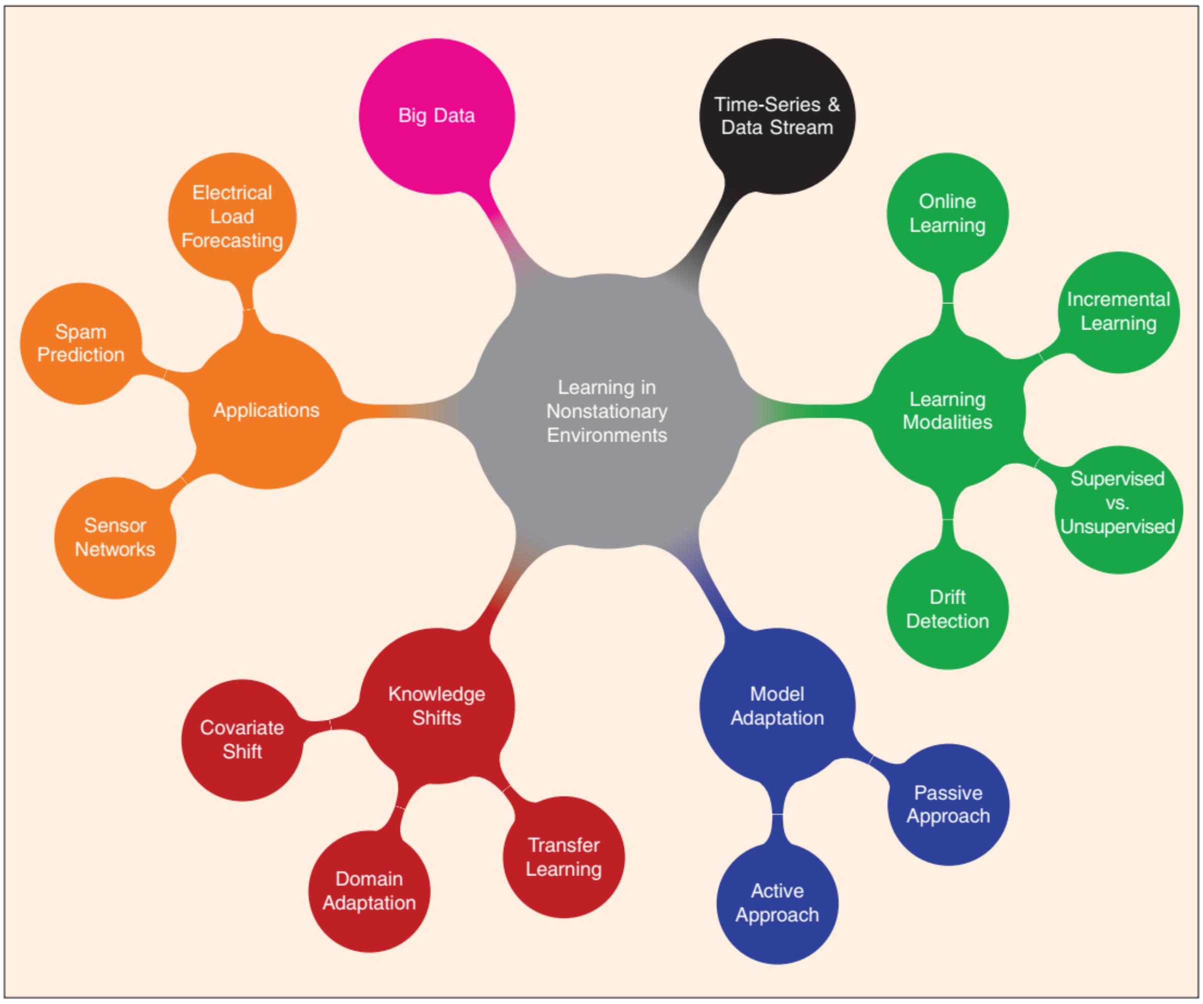


Figure 1.1: Mindmap of concept drift [55]

Static search problems are classic representatives of a static environment which happen to have numerous parallels in the computer search problem domain. In a static environment, the optima are fixed on their positions in the search space. The algorithm typically provides feedback as a reflection on the relative merits of several solutions found during the search.

Various studies have applied metaheuristics to induce a predictive model on the assumption that the environment is static [35] [36]. However, real-world datasets such as electric load and weather data exhibit dynamic characteristics, such as concept drift, that are usually ignored by many metaheuristics.

The dynamic characteristics whose values usually change during the observation period are described better by temporal properties. However, temporal properties increase the problem complexity, since every time temporal property changes, an existing solution will have to be modified [37] [38]. Thus, the objective function tends to change, which results in changes in the search space structure and the position of optima [39]. New optima may appear whereas existing optima may disappear.

Also, the properties of temporal data may be subject to concept drift which may be meant to be discovered by the learning system [47] [48] [49]. Thus, the learned model should be continuously adapted. 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 [50] [51]. 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.

Numerous ways exist to address concept drift in regression problems. One of the ways is to use a static model that is periodically updated with the data collected from the prior period [6] [2]. A sliding window may be used to extract the most recent historical data that captures a new relationship between inputs and outputs [6]. For some regression problem domains, a weighted data approach is used. A higher weight is assigned to the most recent data and a smaller weight to the older data [1] [7]. For domains that expect abrupt changes, techniques that detect changes and select a specific prediction model are more appropriate [6].

A sharp increase in the use of mobile devices, network sensors, internet-of-things technology and emerging of the 4^{th} industrial revolution has led to an ever-increasing and enormous amount of numerical data in various fields such as banking, stock pricing, and electricity pricing and demand. The focus of this thesis is on the application of machine learning to develop a solution to data analytics on nonstationary data where data points are numerical. Data points depict a tuple in a dataset. The proposed algorithm can adapt the induced model when concept drift occurs.

1.1 Motivation

Numerous predictive models are built based on statistics on the assumption of drawing data (training and testing) from the same distribution, however, nonstationary data is usually made up of generating processes which change over time. Therefore, if the knowledge of the existence of different data segments within the dataset is not taken into consideration, then, the induced predictive model is distorted by irrelevant data segments to the current data generating process.

Nonlinear data can be modeled using linear models by performing a piecewise approximation of the problem space. A model tree provides a piecewise linear regression model by building a decision tree hierarchy that fits several smaller data segments of the dataset to yield an improved model that best fits the entire dataset. In most regression applications, linear models generally provide satisfactory approximations.

Concepts in real-world are nonstationary. 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]. 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.

Metaheuristic techniques, such as particle swarm optimization, differential evolution and genetic programming, are adaptable by nature, require slight changes to their standard algorithm structures, are capable to perform in noisy environments and different problem spaces. Also, genetic programming (GP) can induce decision trees from the given datasets.

In this work, metaheuristic algorithms are hybridized to induce a predictive model on nonstationary environments with concept drift occurring. The proposed GP approach for nonstationary data analytics (GPANDA) implements a piecewise approach to predict a target, nonlinear model and consists of three components: dynamic differential evolution-based clustering algorithm to extract clusters that resemble different data generating processes present in the dataset; the dynamic particle swarm optimization-based model induction technique to induce optimal nonlinear models that describe each cluster which approximate mapping between inputs and the target variable; and a standard GP that evolves model trees that define the boundaries of nonlinear models which are expressed as terminal nodes.

The proposed approach dynamically adapts whenever an environmental change is detected. As such, the knowledge of the past environment may prove useful in quickly capturing the current environment or can improve the accuracy of a search. Existing nonlinear models resemble knowledge acquired in the past and are exploited when a change in an environment occurs to model the current environment.

1.2 Objectives

The main goal of this work is to develop a (hybrid) predictive approach for nonstationary data with a numerical target that dynamically adapts when concept drift occurs which can also be used to extract knowledge from historical data. The purpose of the hybridization is to bring together the strengths of the different approaches to solve the problem at hand. The justification for the selection of the given approaches is provided in the following sections: dynamic clustering algorithm in Section 3.5.1, dynamic PSO in Section 3.5.2 and GP in Section 3.5.3. To meet this goal, the following objectives have been identified:

(a) To investigate the effectiveness of hybridizing dynamic PSO with a regression technique, either least-squares approximation or autoregressive, (DynPSO) to induce optimal nonlinear regression models in nonstationary environments;

(b) To investigate the effectiveness of hybridizing a GP with a dynamic clustering algorithm and nonlinear model induction technique (GPANDA) to perform regression on nonstationary data with concept drift occurring;

(c) To compare the performance of DynPSO to optimize the induced model in a nonstationary environment, to dynamic PSO algorithms, namely multi-swarm, reinitialized, and charged PSOs;

(d) To compare the performance of GPANDA in terms of predictive accuracy and computational time to the best performing dynamic GP algorithms and the state-of-the-art techniques on nonstationary datasets that exhibit different characteristics of concept drift such as progressive, recurrent, abrupt and random changes on varying temporal and spatial severities.

In this work, predictive accuracy is measured by the adjusted coefficient of determination, R_a^2, root mean square error (RMSE) or mean absolute percentage error (MAPE) which indicates computational performance.

1.3 Contributions

The main contribution of this work is coming up with a GPANDA technique that evolves model trees, with terminal nodes expressed as predictive models whereby the decision structure and terminal nodes of the tree are dynamically modified to cope with changes occurring in the nonstationary data due to concept drift. As such, GPANDA provides an effective data analytics technique for nonstationary data. Other contributions include:

- Hybridization of dynamic PSO with a regression technique (DynPSO), either least-squares approximation or autoregressive, has shown to be effective to induce optimal nonlinear regression models in nonstationary environments and also, performed competitively with the state-of-the-art techniques.
- Hybridization of dynamic clustering algorithm, DynPSO and GP (GPANDA) has performed satisfactorily on regression problems for nonstationary data with concept drift occurring. The obtained results suggest the capability of GPANDA to

adapt the induced predictive model as the environment changes due to concept shifts occurring to outperform the state-of-the-art techniques.

1.4 Thesis Outline

This thesis is structured as follows (also refer to Figure 1.2): Chapter 2 provides a synopsis of machine. Two machine learning techniques: decision trees and data clustering are discussed. A discussion of concept drift and optimization in dynamic environments is provided.

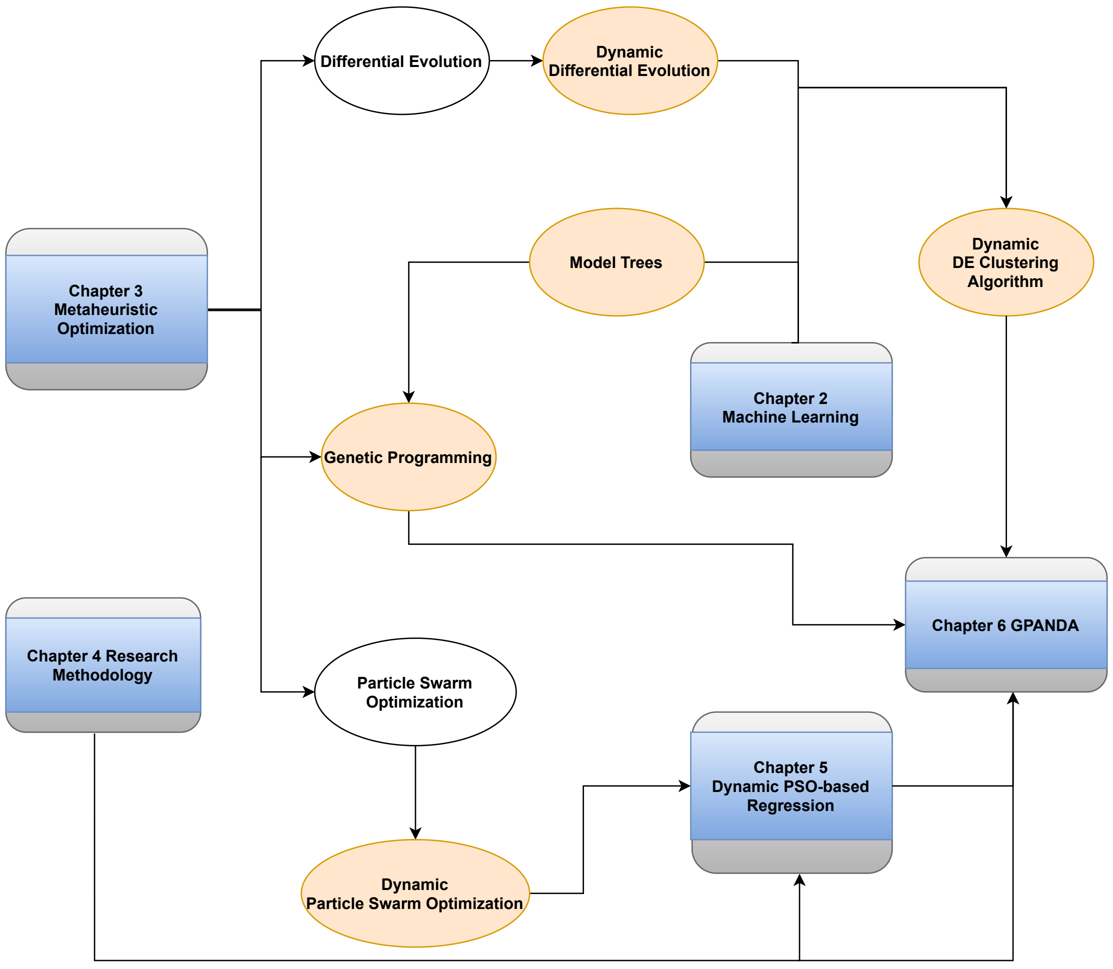

Differential Evolution
Dynamic Differential Evolution
Model Trees
Dynamic DE Clustering Algorithm
Chapter 3 Metaheuristic Optimization
Chapter 2 Machine Learning
Genetic Programming
Chapter 6 GPANDA
Chapter 4 Research Methodology
Particle Swarm Optimization
Chapter 5 Dynamic PSO-based Regression
Dynamic Particle Swarm Optimization

Figure 1.2: Thesis Outline

A decision tree is used in this thesis to induce a predictive model from nonstationary data. A discussion of decision tree induction techniques is provided and also, decision tree variants: classification; regression; and model trees. A discussion of performance measures applicable to data clustering is provided. The following temporal data clustering algorithms are discussed: dynamic clustering PSO; and dynamic clustering differential evolution.

Chapter 3 discusses metaheuristics. A discussion of the following metaheuristics is provided: particle swarm optimization; differential evolution; and GP. The following metaheuristics designed for dynamic environments were discussed: dynamic GP; dynamic differential evolution and dynamic PSOs. A detailed critical analysis is provided.

Chapter 4 discusses the research methodology for the study presented in this thesis. Also, the datasets to be used in this thesis are presented.

Chapter 5 discusses the proposed dynamic particle swarm optimization-based nonlinear regression (DynPSO) to induce optimal nonlinear regression models. A detailed description of each component of DynPSO is provided.

Chapter 6 discusses the proposed GPANDA to evolve predictive models for nonstationary data. A detailed description of each component of GPANDA is provided.

Chapter 7 presents the obtained results and provides a discussion on the DynPSO and GPANDA experiments.

Chapter 8 presents the conclusion of the study and recommends potential areas of future research.

Chapter 2

Machine Learning

2.1 Introduction

This chapter discusses machine learning concepts and approaches applicable to nonstationary data analytics.

In this section, concept drift is discussed in Section 2.2 whereas learning in a nonstationary environment is discussed in Section 2.3. Decision trees are discussed Section 2.4 whereas Section 2.5 discusses data clustering.

2.2 Concept Drift

Concept drift, in machine learning, is a phenomenon where statistical properties of the concept (target variable), being predicted, changes unexpectedly as time passes due to changes in underlying patterns [4]. As a result, the model's prediction accuracy deteriorates over time.

Changes in the search space are characterized by two major changes: spatial and temporal severities [40]. Temporal severity refers to the frequency at which environmental changes can occur on any time-scale. The environmental change can occur periodically, at irregular time intervals or continuously spread over time [39].

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 [41]. Spatial and temporal severities are indicators of the difficulty of a dynamic optimization problem.

Concept drift implies changes in the decision boundaries that separate patterns in a dataset. New boundaries may appear as the environment changes and old boundaries may become obsolete [4]. 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 [4]. For a more severe change, typically, it is harder for the metaheuristic to recover from the change. Figure 2.1 illustrates the different types of concepts drift.

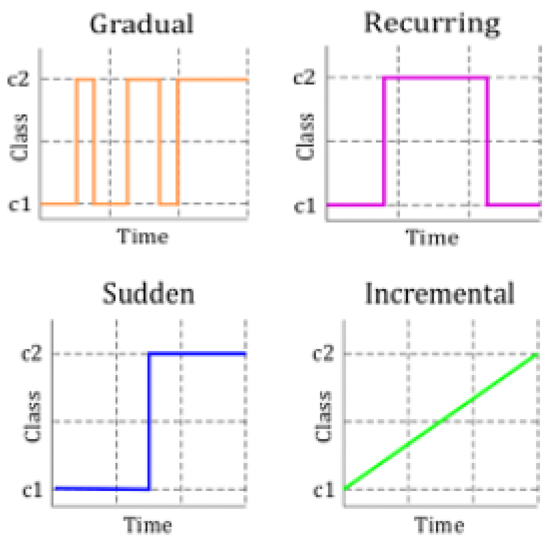


Figure 2.1: Types of Concept Drifts [42]

As illustrated in Figure 2.1, sudden drift happens when there is a sudden switch between the existing environment (class c1) and the new environment (class c2). As a result, all cases from the instant of a switch emanate from the new environment [43]. As such, prediction models induced from the past environment becomes irrelevant to the current prevailing environment. Therefore, metaheuristic tends to lose the diversity that is necessary to locate a new optimal causing the algorithm to be trapped in the local minimum.

In a gradual drift, the existing environment (c1) is replaced with a new environment (c2). The replacement is not abrupt as in sudden drift, rather, the current environment (c1) is discontinued gradually. Therefore, it becomes easier for the metaheuristic to adapt to the changing environment.

Incremental drift is characterized by smaller incremental changes as the existing environment, c1, is substituted by a new environment, c2. In recurring drift, environments (classes) are substituted back and forth as illustrated in Figure 2.1. This recurrence of environments may be periodic or non-periodic. Such a phenomenon is quite natural whereby environments have a seasonal influence. As such, past information becomes relevant when a change happens which implies that obtained optimal solutions from the past environments become applicable to the current prevailing environment. Thus, an algorithm that has a memory can capitalize on the existing past learned solutions.

Recurrent or cyclical variations happen to be a desirable and valuable quality sought in adaptive algorithms to promote the retrieval of previously acquired knowledge, therefore, enables transfer-learning. As illustrated in Figure 2.1, firstly, c1 is replaced with c2 and then after a given time interval c2 is replaced with c1, the previous existing environment (recurrent).

The data reduction technique, instance selection, is commonly used to handle concept drift [44]. Instance selection reduces the training dataset to a manageable volume, which results in the reduction of the computational resources required to perform the prediction process [45]. The simplest instance selection technique, sliding window of analysis, is defined by a fixed number of instances. The sliding window of analysis is a forgetting technique that caters for outdated instances removal [46] by dismisses the oldest instances as new ones arrive, thereby providing up-to-date data to the predictor [3]. A sliding window of analysis technique enables the regressor to reflect and adapt concept drifts in the temporal data.

2.3 Learning in Nonstationary Environments

Commonly used techniques to learn in concept drift occurring environments are generally referred to as passive and active approaches [50]. Generally, the active approach copes quite well in environments where the drift is abrupt whereas a passive approach is ideal for gradual drifts and recurring concepts [50][51].

The following techniques are discussed in this section: transfer, active and passive learning.

(a) Transfer Learning

Transfer-learning is a technique that facilitates the transfer of learned knowledge from the past to the most recent historical occurrences, assuming that there are certain parts of the past learned knowledge which can still be relevant to the new task [52]. Therefore, if transfer-learning is implemented in metaheuristics then it is most likely to greatly improve its performance in terms of learning speed and generalization capability [52][53].

(b) An Active Approach to Learning in Nonstationary Environments

An active approach to learning in a 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 [54].

Considering $\mathcal{P}$ to be a data generating process that provides a sequence of tuples $(\mathbf{x}_t, y_t)$ sampled from a joint probability distribution $p_t = (\mathbf{x}_t, y_t)$ and p_t to be evidence distributions and $p_t\ (y \mid \mathbf{x})$ be prior probabilities, in each point in time, t [55].

The goal of the change detector is to assert if there exists a change in the process $\mathcal{P}$ through an inspection of features extracted [56]. Also, to monitor the stationarity of estimated p_t, from the data-generating process and /or analyzing the prediction error to determine variations in the estimated $p_t\left(y \mid \mathbf{x}\right)$ [56]. 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 patterns.

(c) A Passive Approach to Learning in Nonstationary Environments

Unlike an active approach, a passive approach continuously adapts the model parameters whenever new 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 pitfall 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 [57] [58]. 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 impact some parts of the existing knowledge base leaving the other parts relevant.

2.4 Decision trees

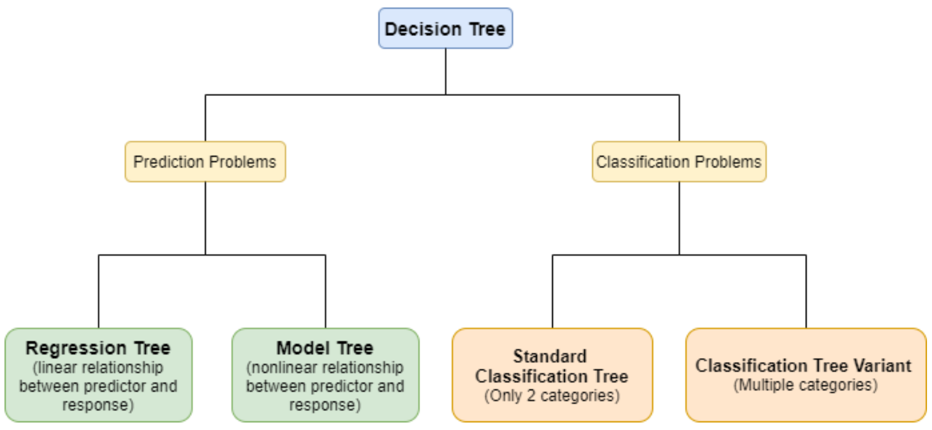
A decision tree is expressed as a directed tree that recursively partitions the dataset based on a selected feature [4]. The recursive partitioning results in a tree structure which once constructed can be used as a classifier or predictor.

Decision trees provide a fast and simple way to learning a function (f) that maps data (x) to output (y). Data (x) can be a numerical variable, categorical, or both numerical and categorical. Output (y) can be numerical for regression or categorical for classification [11] [12] [13].

Decision trees can be classified based on the type of target/dependent variable as [14]:

- Classification trees - the target is a discrete-valued variable (categorical);
- Regression trees - the target is a numeric (continuous) value; and
- Model trees - the target is a linear or non-linear model of the independent variables/features.

Figure 2.2 depicts the classification of decision trees based on the target variable.



Decision Tree
Prediction Problems
Classification Problems
Regression Tree (linear relationship between predictor and response)
Model Tree (nonlinear relationship between predictor and response)
Standard Classification Tree (Only 2 categories)
Classification Tree Variant (Multiple categories)

Figure 2.2: A Classification of Decision Trees[14]

Decision trees are automatically constructed from a given dataset by algorithms called inducers. Top-down decision tree inducers build a decision tree in a divide and conquer approach based on the splitting metric at each decision node. Examples of greedy top-down decision tree inducers include ID3 [8], C5.0 [15], CART [16]. CART uses

Gini impurity as its splitting metric, ID3 and C5.0 use information gain [16] [9]. The information gain is to be maximized whereas Gini impurity is to be minimized. Genetic programming (discussed in Section 3.3.1) also evolves decision trees. Algorithm 1 summarizes a decision tree induction process.

Algorithm 1 Decision Tree Induction Process

```
1: BEGIN
2: Let $D$ be the root node that comprises of all data points in a dataset $S$
3: Use attribute selection measure to find the best attribute in $D$
4: Divide $D$ to come up with a set $D_s$ that consists of subsets that contains all possible
   values for the best attributes
5: Create a decision tree node that comprises of the best attribute
6: Using $D_s$ created in line 4, recursively create new decision tree until no further
   classification of the nodes
7: END
```

This section is outlined as follows: Section 2.4.1 presents classification trees. Regression trees are discussed in Section 2.4.2 whereas model trees are discussed in Section 2.4.3.

2.4.1 Classification Trees

A classification tree splits a dataset into classes, based on the response variable [14] [17]. A response variable is a variable being measured. A response variable can be binary or multi-class. A binary response variable usually has two classes, false or true (categorical data) which may be numerically categorized as 0 or 1 respectively. Commonly used classification trees include ID3 [8], CART [16] or C5.0 [15]. Figure 2.3 is an example of a classification tree.

A standard classification tree splits the dataset based on data homogeneity. For example, considering Fisher's Iris data illustrated in Figure 2.3 [18], if the dataset has shown that 97 % of the flowers with a petal width of less than 0.8 are Setosa, a split is performed and Setosa becomes a top node in the tree. Thus, the split has made the data '97 % ' pure.

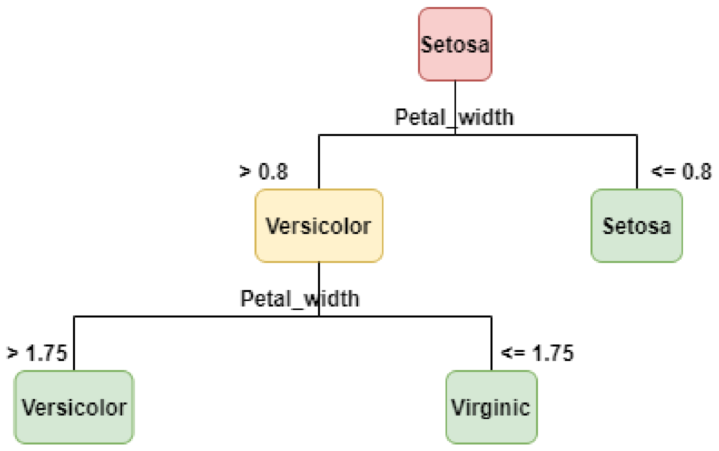


Figure 2.3: An Example of a Classification Tree [18]

Classification trees quantify data homogeneity through the use of impurity which is either a Gini index or entropy. The impurity is rigorously measured by the computing proportion of the data that belong to a class.

2.4.2 Regression Trees

A regression tree is a decision tree having a independent variable(s) which are continuous and are used to predict the value of that dependent variable [19]. As such, regression trees apply to prediction problems.

A regression tree partitions a dataset and then fits a piecewise constant function. The data is split at several split points for each independent variable whereby the sum of squared errors (SSE) between the actual value and the predicted values at each split point is calculated. Then, a comparison is performed and the split point with the lowest SSE is selected to be the root node/split point.

The path of each continuous target value, starting from the root node to the terminal node, corresponds to a regression rule. Two popular regression tree inducers are CART [16] and ANN [20].

Consider, for example, an experimental dataset $(x, y) \in X \times Y$, where X denotes the instance space covered by n predictor variables, x (scalar variable) (both categorical and numerical, the objective is to predict the continuous response (dependent), variable

Y. As such, a regression tree makes use of a piecewise constant function to estimate a function $y = g(x)$ [21]. Thus, the splitting nodes partition the instance space, whereas the regression nodes perform linear regression.

An example of a regression tree which predicts the mileage in kilometers per liter based on cylinders and horsepower, a car will average is illustrated in Figure 2.4.

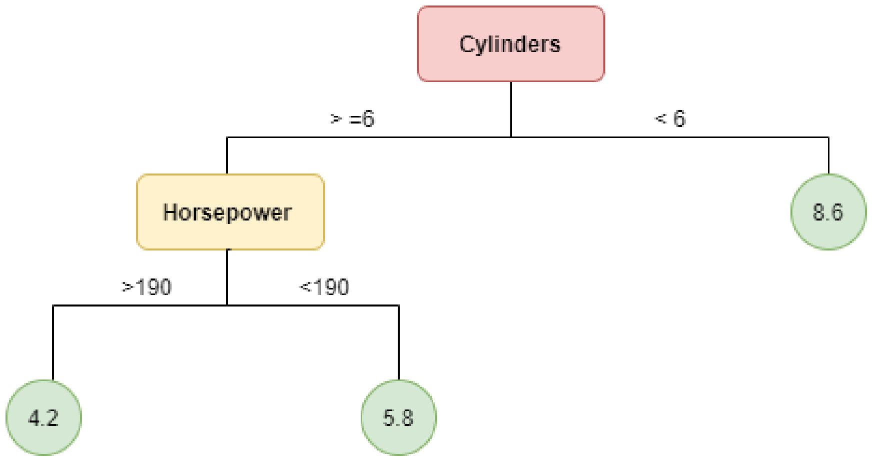


Figure 2.4: Regression Tree for Predicting Mileage based on Cylinders and Horsepower

2.4.3 Model Trees

The use of a piecewise constant function makes regression trees lose their ability to handle highly non-linear parameters [22]. In such cases, model trees are preferable.

A model tree is a regression tree with its leaves expressed as multiple regression models [23] [24]. A model tree makes use of a piecewise linear function to approximate a function $y = g(x)$. Thus, a model tree splits the parameter space into subspaces and then fits a linear function for each subspace.

Figure 2.5 is a graphical illustration of a model tree indicative of a piecewise approach splitting of a nonlinear dataset into subsets and then fitting a model for each subset.

Examples of model trees inducers include M5 [25], CART [16]. Also, regression and model trees can be induced directly by GP (discussed in Section 3.3.1). In this respect, the terminal set contains values in regression trees or models in model trees.

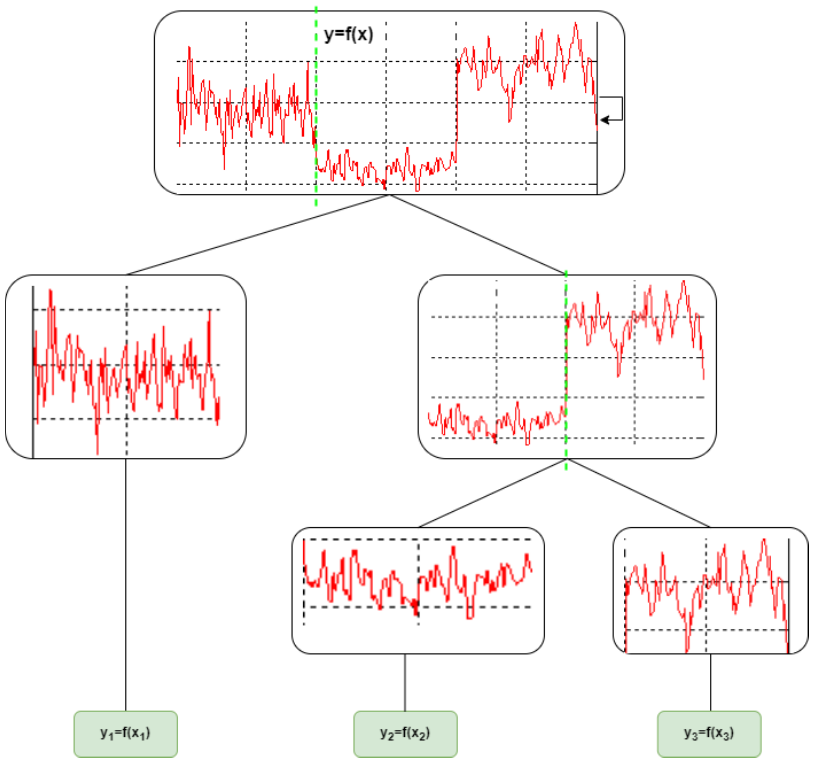


Figure 2.5: An Example of a Model Tree

2.5 Data Clustering

A data clustering is an unsupervised learning process in which datasets are grouped according to regions with high data point density. Nonstationary data (temporal data) is usually dynamic, possessing data patterns that change with time, therefore, clustering is required each time a change is detected. Since clustering process groups together similar objects, commonly referred to as data patterns [106][107][108], therefore, a similarity measure has to be established to determine the proximity of two data patterns to each other [109]. The structure of the cluster is derived by optimizing the data partitions using objective criteria so that data patterns among clusters are dissimilar and data patterns within a cluster are more similar [110].

Clustering algorithms can be broadly categorized into hierarchical and partitional. Hierarchical data clustering implements a tree-like structure that initially considers each data pattern as a leaf node [111]. As the tree grows, most similar nodes are paired to create the internal node. This process iterates until all the data patterns are merged into one cluster, the root node. As such, a hierarchical tree poses different levels of data clusters at each depth. Also, a hierarchical tree can start with the root node that contains all data patterns, and then cascades down to the leaf nodes which consists of a single data pattern.

Clustering algorithms recursively partition data to decrease similarities between data patterns in different clusters and increase the similarities of data patterns within the same cluster [112]. Therefore, the obtained set of clusters contains similar data patterns.

2.5.1 Clustering Performance Measures

In clustering, the quality of the obtained solution is ascertained by validity indices. Numerous validity indices exist in the literature: Ray-Turi [113]; Davies-Bouldin [114]; Dunn validity index [115]; silhouette coefficient [116]; the CS measure [117]; and cluster similarity [118].

In this section, Section 2.5.1 discusses intra-cluster distance and Section 2.5.2 discusses inter-cluster distance. Silhouette coefficient is discussed in Section 2.5.3 and the fitness measure in Section 2.5.4.

(a) Intra-cluster Distance

The intra-cluster distance ascertains cluster compactness by measuring the mean distance between cluster centroid and the corresponding data patterns [119]. Thus, the intra-cluster is the within-cluster distance. Therefore, the intra-cluster distance is supposed to be minimal.

The intra-cluster distance measure is computed as:

$$D_{intra} = \sum_{i=1}^{K} \sum_{\forall z \in C_i} d(z, c_i) \tag{2.1}$$

where C_i is the i^{th} cluster and d is the Euclidean distance between the centroid, c_i and the data pattern, z, K is the total number of clusters.

(b) Inter-cluster Distance

The separability among the cluster centroids is measured by the inter-cluster distance [119]. Therefore, the inter-cluster distance is supposed to be as large as possible. After computation of inter-cluster distance, the minimum value is considered. The inter-cluster distance measure is computed as:

$$D_{inter} = \frac{2}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} d(c_i, c_j) \tag{2.2}$$

where d is the Euclidean distance between cluster centroids c_i and c_j.

(c) Silhouette Coefficient

A silhouette coefficient quantifies the quality of the clustering solution by ascertaining data patterns that fit well and those that are not fitting within a cluster [116]. The silhouette coefficient is expressed in the range $[-1, 1]$. A higher value of the silhouette coefficient signifies a better clustering solution. A silhouette coefficient is calculated as [120]:

$$s(z_p) = \frac{b(z_p) - a(z_p)}{max\{a(z_p), b(z_p)\}} \tag{2.3}$$

where $a(z_p)$ is a dissimilarity measure of data pattern, (z_p), to its cluster and $b(z_p)$ is the lowest mean dissimilarity of (z_p) to other clusters. Dissimilarity measures the degree to which two data patterns are different. A small value of $a(z_p)$ implies that (z_p) is well fit whereas a larger value of $b(z_p)$ means (z_p) is poorly fit.

(d) Fitness Measure

Numerous fitness measures have been used in data clustering algorithms which include squared error [121] and quantization error [122]. The squared error, for all clusters, is a measure of total squared distance between the cluster centroid and the data patterns in that cluster [121][123] and is computed as:

$$SE = \sum_{i=1}^{K} \sum_{j=1}^{|C_i|} d(c_i, z_j)^2 \tag{2.4}$$

The quantization error, for all clusters, is a measure of the average distances between the cluster centroid and the data patterns in that cluster [122] and is computed as:

$$Q_e = \frac{\sum_{i=1}^{K} \frac{\sum_{\forall z \in C_i} d(z, c_i)}{|C_i|}}{K} \tag{2.5}$$

The squared error is adopted in the dynamic clustering algorithm discussed in Section 2.5.2, as the fitness measure to guide the clustering algorithm towards an optimal solution. Also, intra-cluster measure, inter-cluster measure, and the silhouette coefficient are used as conditions in the dynamic clustering algorithm discussed in Section 2.5.2.

2.5.2 Clustering of Temporal Data

Temporal data is a series of events over a period that may be time-stamped at regular or irregular time intervals such as weather or stock market data [35]. Usually, temporal data consists of data patterns that change with time and characterized by a high amount of dependency among itself in which appropriate treatment of these dependencies or relationships is essential. Temporal data is encountered in many applications, including budgetary [124], stock modeling analysis [125], classification [126] and clustering [127].

Clustering is usually performed once for a static dataset. However, temporal data usually possesses data patterns that change with time and therefore, clustering is required each time a change is detected [109]. The changes that occur in data patterns may be of position, composition, split or merge [128]. Consequently, these changes may cause new clusters to appear or existing clusters to disappear. Also, data patterns may migrate within clusters leading to an increase or decrease in cluster size.

Clustering temporal data using metaheuristics such as DE and PSO pose two problems, discussed in Section 3.4.1, which are diversity loss and outdated memory of individuals in the population. Therefore, metaheuristics designed for dynamic environments discussed in Section 3.3 and Section 3.4 can be adapted to perform clustering in temporal data by tracking clusters through time. Also, maintaining a diverse population of individuals at all times to facilitate a search for a new solution.

This section discusses dynamic clustering PSO, dynamic clustering DE and k-independent dynamic clustering DE.

Dynamic Clustering Particle Swarm Optimization

A dynamic PSO (discussed in Section 3.4.1) can be modified to cluster temporal data. Numerous applications of dynamic clustering PSO algorithms exist in the literature [122][129][130][131]. A multiple swarm technique was implemented in [122] whereby a particle represents a group of centroid positions, however, the entire dataset was used to evaluate the fitness of each particle. A hierarchical approach was implemented in [130] in which a temporary sub-swarm was created that was further refined by re-running the algorithm. This technique was effective though computationally expensive.

Dynamic Differential Evolution Data Clustering

A dynamic DE (discussed in Section 3.3) can be modified to cluster temporal data by implementing a multi-population approach [132][133]. An individual in a dynamic DE clustering algorithm can be encoded as a vector that represents a group of centroids. The resulting algorithm enables each sub-population to optimize all the data centroids and yields a global best *(gBest)* as the final solution over all sub-populations.

To ensure that the population is always diverse, the following techniques are implemented in dynamic DE. The *gBest* of each sub-population is assigned an exclusion radius, r_{excl}, to create a sphere of influence. The sub-population with the weakest *gBest* is re-initialized if two gBest are within r_{excl} of each other. Also, for each sub-population, a percentage of weakest individuals are promoted to become Brownian individuals (discussed in Section 3.3).

The DE-based clustering outperformed both the PSO and the GA on partitional clustering problems [134]. Also, a dynamic clustering DE outperformed a dynamic clustering PSO on dynamic problems in [135].

K-independent Cooperative Data Clustering Dynamic Differential Evolution

An optimal number of clusters, K, in the real-world dataset is unknown a prior. Therefore, it is ideal to automatically determine K. Two common techniques to determine K automatically are the split and merge approach, and an iterative approach.

In a split and merge approach, as the algorithm progresses, the optimal value of $K$is dynamically determined by splitting and merging clusters [136][137][130]. In an iterative approach, the algorithm is run several times for different values of K and the value of K that gives the best validity index is then selected [138][139]. An iterative approach requires several runs to obtain an optimal value of K and therefore, not favorable.

The K-independent Cooperative Data Clustering Dynamic Differential Evolution (KCDCDynDE) is a dynamic clustering DE that dynamically determines the optimal number of clusters, K using the split and merges approach [135]. As such, the algorithm groups together similar data patterns in temporal data whenever a change in the dataset is detected. Algorithm 2 depicts a KCDCDynDE algorithm [135].

The KCDCDynDE is a multi-population dynamic clustering DE algorithm in which each dimension of the problem is optimized by a particular sub-population [140]. Thus, for a population of K sub-population, each sub-population, s_k optimizes one centroid at a time. Then, the solution of the problem referred to as the context individual, is the combined best positions from each sub-population [135].

An individual's fitness from a sub-population, s_k, is calculated by substituting the dimension, K, in the context individual with the solution of s_k. Then, the resulting context individual is evaluated to give the individual fitness [135]. Figure 2.6 is a graphical illustration of fitness calculation in which the context individual is made up of x_1, x_2, x_3, x_4 and x_5 centroids. The fitness of individual x_2 is calculated by substituting the dimension x_2 in the context individual with its solution x_6.

Algorithm 2 KCDCDynDE [135]

```
1: BEGIN
2: Initialize a Pop, $S$ with $s$, subPop where $s \geq 2$
3: Assign: $k \rightarrow popIndex$ and $S_k.\hat{y}_i \rightarrow bestPosition$ for $S$
4: DO
5: Create a context individual, $b(k, S_k.\hat{y}_i)$
6: for each subPop, $s_k$ do
7:     for each individual, $S_k.x_i$ do
8:         $b(k, S_k.x_i \rightarrow b(k, S_k.\hat{y}_i$
9:         for each data pattern, $z_p$ do
10:            Compute $d(z_p, x_i)$
11:            Compute cluster centroid $s_k.x_{ik_j}$ of $b(k, S_k.x_i)$
12:            Assign pattern $z_p$ to centroid $S_k.x_{ik_j}$ such that
13:            $d(z_p, S_k.x_{ik_j}) = \min_{\forall k_j=1 \ldots K}\{d(z_p, S_k.x_{ik_j})\}$
14:        end for
15:        Calculate fitness $f(x_i(t))$
16:    end for
17: end for
18: Update $gBest$ of each $s$
19: if $d(gBest(s_i), gBest(s_j)) \leq r_{excl}$ then
20:    Randomly remove half of each SubPop: $s_i$ and $s_j$
21:    Merge the remaining halves of SubPop, $s_i$ and $s_j$
22: end if
23: if pop-converged then
24:    Compute silhouette of the $gBest$ position
25:    Recompute the intra-cluster and inter-cluster distances
26:    if silhouette $< 0.71$ AND conditions-for-adding-cluster $\geq 1$ then
27:        Add a randomly initialized cluster
28:    end if
29:    Update intra-cluster and inter-cluster
30: end if
31: for each $s$ do
32:    for each individual, $x_i \in S$ do
33:        Calculate fitness $f(x_i)$
34:        Generate a trial vector and offspring using Eqn (3.2) and (3.3)
35:        if $f(x_i^{'})$ is better than $f(x_i)$ then
36:            Add $x_i^{'}(t)$ to $S(t+1)$
37:        else
38:            Add $x_i(t)$ to $S(t+1)$
39:        end if
40:    end for
41: end for
42: Promote $M\%$ of the weakest individuals to be Brownian individuals
43: END
44: *Return* the best individual as the solution
45: END
```

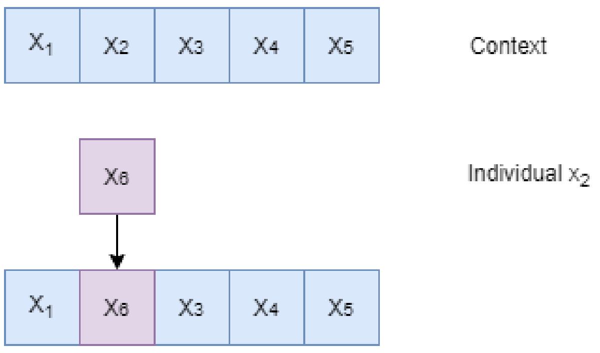


Figure 2.6: Cooperative Fitness Calculations

In KCDCDynDE, a predefined radius, r_{conv}, is used to establish the convergence of the population where the average distance between the best individual and the entire population should be smaller than r_{conv}. Distance measures: intra-cluster and inter-cluster are computed after the population convergence. Obtained values are used for comparison with previously calculated values.

The silhouette coefficient is used to measure the quality of the current solution to avoid the addition of a new cluster if a good solution has been found. Therefore, a new cluster is added if the silhouette value is lower than a given silhouette threshold. The silhouette threshold value is user-defined.

Adding a new cluster favors exploration thereby improving the clustering solution. A new cluster is added, with its individuals randomly initialized within the search space, if:

(a) There is an increase in the intra-cluster and inter-cluster distance which entails more separation among clusters.

(b) Distance measures have improved, thus an increase in the inter-cluster distance and the decrease in the intra-cluster distance result in more compact and more separate clusters. Adding a new population may improve the clustering solution further by decreasing intra-cluster distance and increasing inter-cluster distance.

(c) Both distance measures have worsened. This could have been caused by a prior merging of the population.

The populations within r_{excl} (discussed in Section 3.3) of each other are merged. Consequently, the specified numbers of population individuals from the merged populations are randomly selected to maintain the population size and then re-initialized to random positions in the search space [135].

2.6 Summary

This chapter discussed machine learning techniques: decision trees and data clustering. A discussion of decision trees and their characteristics that are relevant to this thesis was provided. Decision tree variants: classification trees, regression trees and model trees were discussed. Illustrative examples of classification, regression and model trees were presented. A general overview of the data clustering was presented. Clustering performance measures relevant to the current work were presented and a discussion of dynamic clustering differential evolution algorithms designed for dynamic environments was provided.

Optimization is the challenging problem that triggers many machine learning algorithms. As such, metaheuristics are becoming pivotal to optimization. The next chapter discusses metaheuristics designed for both static and dynamic environments.

Chapter 3

Metaheuristics

3.1 Introduction

This chapter explores metaheuristics designed for both static and dynamic environments. A metaheuristic is an iterative process that guides a given heuristic by integrating different concepts for the exploration and exploitation of the search space, to come up with near-optimal solutions [26]. Metaheuristics are problem-independent techniques that can be applied to a broad range of problems [26].

In this chapter, Section 3.2 discusses metaheuristics in nonstationary environments. Section 3.3 discusses evolutionary algorithms while particle swarm optimization is discussed in Section 3.4. Section 3.5 present a critical analysis of the literature under review.

3.2 Metaheuristics in Nonstationary Environments

Metaheuristic algorithms usually imitate natural phenomena [27] such as human memory in tabu search [28], social behavior within a bird flock in particle swarm optimization [29], physical annealing in simulated annealing [30] and evolution in evolutionary algorithms [31] [32]. Numerous metaheuristics implement some form of stochastic optimization; therefore, the obtained solution depends on the generated set of random variables [33]. Thus, a globally optimal solution is not guaranteed in some classes of problems [34]. Metaheuristic algorithms were successfully applied to dynamic environments in [15] [16] [17].

In this thesis, optimization refers to an iterative process that refines provided candidate solutions toward an optimal solution [39]. In this regard, an optimization problem refers to the problem of coming up with a suitable solution from the feasible space of the search space [38]. A model tree induction can be regarded as a special case of a function optimization problem, whereby the goal is to minimize the prediction error of the induced model trees.

The main goal of implementing an algorithm to a dynamic optimization problem is to come up with optimal solutions during the optimization process at all time steps [37]. Therefore, a dynamic optimization algorithm should be able to detect temporal properties changes and modify the existing solution (respond to the change), if necessary [40].

In a nonstationary environment, the objective function changes, which results in changes in the search space structure and the position of optima [39]. New optima may appear whereas existing optima may disappear. Changes in the search space are characterized by two major changes: spatial and temporal severity discussed in Section 2.2.

Spatial and temporal severity are indicators of the difficulty of a dynamic optimization problem. Considering both spatial and temporal severity, dynamic environments can be categorized mainly into [41], though some other possibilities may exist :

- Quasi-static, where environmental modifications are irrelevant in comparison to the the scale of the problem.
- Progressive, where the environmental changes are smooth and progressive that is characterized by small frequent alterations.
- Abrupt, where the changes are severe though rare that results in the relocation of the optima to possibly distant locations in the search space and/or significantly changing their fitness.
- Chaotic, where the temporal and spatial severity are high. The optimal is randomly repositioned at every iteration in extreme cases.

Figure 3.1 illustrates the behavioral classes of metaheuristics, considering spatial and the temporal severities of the changes that an environment can experiences.

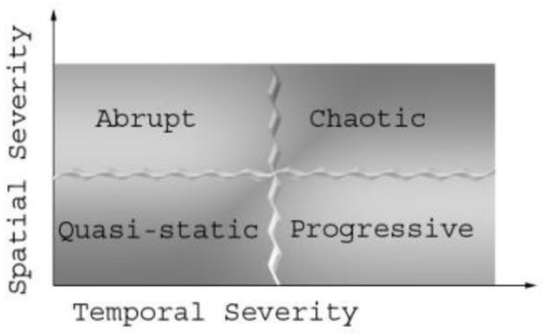


Figure 3.1: Behavioral Classes for Metaheuristics [41]

As illustrated in Figure 3.1, chaotic environment is the most difficult since its characterized by both extreme spatial and temporal severities whereas quasi-static is the least difficult since it closely resembles a static environment.

3.2.1 Performance Measure

In a nonstationary environment, the algorithm does much more than just finding an optimum solution, rather it detects changes in the optimum solution, keeps track of the optimum and comes up with better optima as they surface. Therefore, a dynamic environment calls for a representative performance measure in which the algorithm performance is reflected across the dynamic range of search space. A collective mean fitness is considered in this work and computed as:

$$\bar{f}(n) = \frac{\sum_{i=1}^{n} f(i)}{n} \tag{3.1}$$

where n is maximum iteration and $f(i)$ is the fitness of the best individual at iteration i.

The 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.

The following performance metric will be implemented in the experiments: MAPE and RMSE, defined as:

$$MAPE = \sum_{i=1}^{n} |\frac{(\bar{y_i} - y_i)}{y_i}| \times 100 \tag{3.2}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\bar{y_i} - y_i)^2}{n}} \tag{3.3}$$

where n is the number of data patterns in the testing dataset, y_i is the i_{th} actual value and $\bar{y_i}$, is the i_{th} predicted value.

3.3 Evolutionary Algorithms

Evolutionary algorithms (EA) implement computational models of evolution processes which are based on natural selection, reproduction and survival of the fittest, as the building blocks of the algorithms [31][59]. Examples of EA include genetic algorithms [31], genetic programming (GP) [60] and differential evolution (DE) [41].

In this section, Section 3.3.1 discusses GP and Section 3.3.2 discusses DE.

3.3.1 Genetic Programming

Genetic programming is an algorithm that evolves a population of programs or algorithms which start from randomly generated population (individuals) [60] [61]. As such, the GP solves a problem at hand by evolving computer programs that explore a program space. A GP generates each computer program using the building blocks that are required to solve the given problem. Each generated program is then evaluated to obtain its fitness. Algorithm 3 summarizes a GP.

Individuals are selected, based on their fitness', to act as parents which are evolved by applying genetic operators to produce new individuals (offspring) until the desired goal is hopefully attained. Two commonly used selection techniques are fitness proportionate

Algorithm 3 Genetic Programming [61]

```
1:  BEGIN
2:  Let $t = 0$
3:  Randomly create individuals that constitute the $Pop(0)$ using terminal and function
    sets
4:  Compute fitness for each individual
5:      WHILE termination condition(s) not satisfied DO
6:          Select individuals from $Pop(t)$ to use in reproduction process
7:          Perform crossover using selected individuals to create offspring
8:          Perform mutation on the offspring
9:          Compute fitness for each offspring
10:         Select individuals to constitute a new population, $Pop(t + 1)$
11:         $t = t + 1$
12:     UNTIL termination condition satisfied
13:   END
```

selection and tournament selection [62]. The fitness function, which is usually problem-dependent, is used to measure the effectiveness of the generated individuals [61] [63].

Genetic operators are meant to reproduce new individuals with better features and properties over their parents. Commonly used GP operators are reproduction, crossover and mutation [62] [64]. Reproduction passes the 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 [61] [62]. 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 usually fixed. As such, application rate parameters commonly referred to as crossover rate, mutation rate and reproduction rate determines the total number of individuals to be reproduced by crossover, mutation and reproduction respectively [61].

GP terminates if the given condition is satisfied. Either a problem-specific solution is achieved or a user-defined maximum number of generations is reached [64]. Optimal parameter values, such as a maximum number of generations, are usually obtained through parameter tuning.

Traditionally, GP programs were expressed as parse trees, however, other forms of representation were introduced which includes graph-based GP [65] and linear GP [66]. A parse tree is induced from variables and constants in question which constitutes the terminal and function sets. The functional set usually consists of functional, mathematical, logical or deterministic operators which are used to link variables [67] [61]. The arity of the selected node from the function set determines the branching factor of that node [67].

A GP-induced parse tree is constructed within the restriction of the maximum tree depth by randomly selecting a root node from the function set. The parameter, tree depth, is usually user-defined. Nodes thereafter are picked from either a terminal set or a function set [39]. If a node is picked from the terminal set, it then, logically, becomes a leaf node else an internal node. Figure 3.2 illustrates a parse tree indicative of the root node as the top most node, internal nodes and terminal nodes as the leaves of the tree.

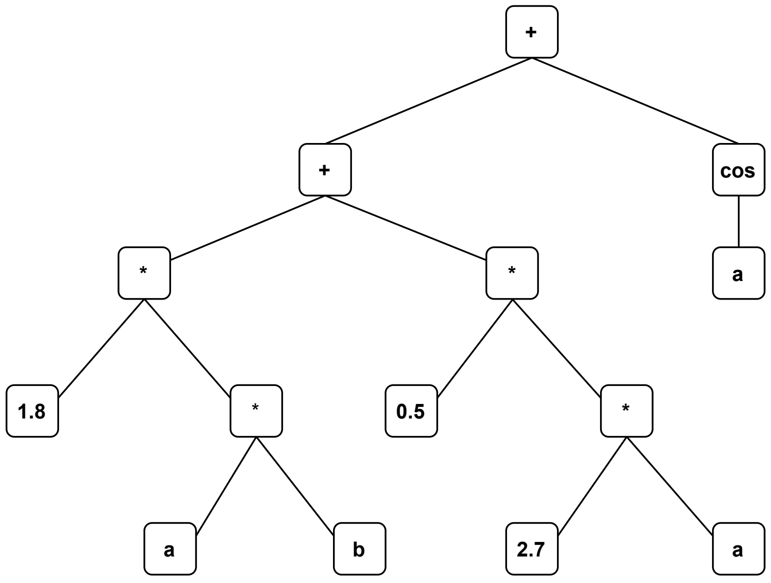


Figure 3.2: An Example of a GP Tree

Despite the promising problem-solving capabilities of a GP, this algorithm suffers from scaling problems, especially to larger problems like data mining [68] [69]. Such problems include the bloating of the solution with introns (extraneous code), and the complexity of the GP solution. Controlling the complexity of GP solutions (introns) is a challenging task. Methods to deal with introns, such as individual parsing, alternative crossover and selection are suggested in the literature [70].

Genetic Programming for Nonstationary Environments

GP modifies its population as it converge 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 is ineffective in nonstationary environments.

Numerous techniques were proposed in the literature to make GP cope with dynamic environments. Instead of maintaining a fixed population size like in standard GP, dynamic population size can be implemented [71]. Thus, when fitness is improving, the population size is reduced to promote exploitation. If the fitness deteriorates, the population size is increased by adding new randomly generated individuals to facilitate exploration.

This section reviews GP algorithms designed for nonstationary environments which are classified as parametric and memory-based approaches.

(a) Parametric Approach

A dynamic GP (dynGP) that implements an adaptive parametric approach was proposed in [72]. Adaptive control parameters enable the GP to dynamically adjust as a change in environment is evident. When a change in an 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 [72]:

- Elitist proportion

  Whenever fitness deteriorated, the percentage of elitist individuals is reduced by 0.1 with a lower bound of 0.1, to promote exploration. 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

  If the fitness deteriorates, the crossover rate is linear increased, by 0.1 with an upper bound of 1, to generate more offspring. However, when the fitness improves, the crossover rate is reduced, 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 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. 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 dynGP was compared to gradient descend-artificial neural network and standard GP and outperformed all other training algorithms in all severity of changes of the environment modifications [72].

(b) Memory Approach

The dynamic forecasting GP (DyFor GP), aims at time series forecasting, automatically adapts to a changing environment and retains knowledge (implicit memory) from the past environments through introns [73]. The DyFor GP implemented a sliding window of analysis to model a natural adaptation for a nonstationary environment. Two sliding windows of different sizes are defined at the beginning of the historical data and slides after a given number of iterations (dynamic generation). The process repeats until all the available data have been analyzed. During a dynamic generation, DyFor GP evolves a population of solutions in which the best solution is selected as the predictive model for that analysis window.

For each dynamic generation, the prediction accuracy for each analysis window 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, else expand. Consequently, DyFor GP discovers optimal window as the algorithm iterates.

The sliding window of analysis enables DyFor GP to analyze all historical data, gleaning knowledge as the window slides. The gleaned knowledge is adapted as GP subtrees either implicitly through the evolution process as the window slides or explicitly through the use of 'dormant' solutions. Dormant solutions usually speed up the convergence of the algorithm, therefore, when a change in the environment is detected, dormants are injected into the population.

3.3.2 Differential Evolution

Differential evolution (DE) is a stochastic, population-based algorithm [74] [75]. The population of a DE consists of vectors (individuals). Individuals are adapted through mutation, crossover and selection. Unlike other evolutionary algorithms, the mutation in DE is applied first to generate a trial vector which is then used in the crossover to produce a single offspring. The search process is guided by direction and distance information about individuals in the current population [76][74].

Individuals in DE are randomly initialized. The selection operation is used to select a target individual that is mutated to produce a trial vector [77]. A trial vector, $u_i(t)$ is generated as follows:

- Select a target individual, $x_{i_1}(t)$ and two vectors $x_{i_2}(t)$ and $x_{i_3}(t)$ such that $i \neq$

$i_1 \neq i_2 \neq i_3$ and $i_2, i_3 \sim U(1, n_s)$ where n_s is the population size.

- Perturb the target individual $x_{i_1}(t))$, to calculate a trial vector using $x_{i_2}(t)$ and $x_{i_3}(t)$:

$$u_i(t) = x_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \tag{3.4}$$

  where the scale factor, $\beta \in (0, \infty)$ controls the differential variation amplification, $x_{i_2}(t) - x_{i_3}(t)$.

Then, a crossover process (binomial) is applied to the trial vector $u_i(t)$, to generate a single offspring, $x^{'}(t)$ through the implementation of discrete recombination of the parent vector, $x_i(t)$ and the trial vector, $u_{(i,j)}(t)$ as follows:

$$x^{'}_{i,j}(t) = \begin{cases} u_{(i,j)}(t) & \text{if} \quad j \in J \\ x_i(t) & \text{otherwise} \end{cases} \tag{3.5}$$

where $x_{i,j}(t)$ refers to the j^{th} elements of the vector $\mathbf{x}_i(t)$ and J is the set of element indices that will undergo perturbation [39]. A crossover probability, p_r, ensures that one randomly selected dimension is forced to be within the set J. Alternative crossover strategies exist in the literature [39] [80]. Other strategies for the mutation process have been provided in the literature [78][39][79]. Algorithm 4 depicts a standard DE algorithm.

The DE implements a replacement strategy when generating the population of the next generation to ensure that there is no deterioration of the average fitness of the population. Exploration is directly influenced by the population size. Thus, a population of many individuals has several differential vectors, therefore, explores in many directions. However, an increase in population size tends to increase computational complexity.

In addition to population size, n_s, crossover probability, p_r, and the scale factor, β, also influence the performance of a DE. Smaller values of scaling factor, β, bring smaller mutation step sizes which promote slower convergence. Larger values for scale factor, β, facilitate exploration, though may result in overshooting good optima [39].

Algorithm 4 Differential Evolution

```
1:  BEGIN
2:  Set $t \rightarrow 0$ be the generation counter
3:  Initialize $\beta$ and $p_r$
4:  Generate and initialize a population, $Pop(0)$ with $n_s$ individuals
5:
6:  WHILE termination condition(s) not satisfied DO
7:
8:  for $x_i(t) \in Pop(t)$ do
9:      Compute fitness, $f(x_i(t))$
10:     Select a target individual, $x_i$ and two different individuals
11:     Generate $u_{(i,t)}$ using Equation (3.4)          ▷ generate a trial vector
12:     Generate $x_i^{'}(t)$ using Equation (3.5)          ▷ generate an offspring
13:
14:     if $f(x_i^{'}(t)) > f(x_i(t))$ then                 ▷ if $f(x_i^{'}(t))$ is superior
15:         $x_i^{'}(t) \cup Pop(t+1)$
16:     else
17:         $x_i(t) \cup Pop(t+1)$
18:     end if
19: end for
20: END
```

Larger values of crossover probability, p_r, bring more deviation in the new population, thus increasing exploration capabilities and increasing diversity. Therefore, decreasing p_r increases search robustness whereas an increasing, p_r, often results in faster convergence [81] [82].

The DE was successfully implemented in a static environment to solve different optimization problems [83][84][85][86]. Like any other evolutionary algorithm, DE modifies its population as it converges towards optimality. Therefore, DE in a changing environment is prune to diversity loss (discussed in Section 3.4.1) [87].

Differential Evolution for Dynamic Environments

The standard DE can be modified to adapt to the dynamic environment by introducing: the exclusion process and the Brownian individual. The modification makes the standard DE a multi-population algorithm where the resulting sub-populations are then used to search for a solution.

In an exclusion process, each individual's fitness *(lBest)* is assigned an exclusion radius that creates an area of influence. A repulsion force exists for any *lBest* position within the defined radius of each other. This repulsion force repels the worst of the two *lBest*. Consequently, the sub-population of the repelled *lBest* is re-initialized randomly within the search space.

The addition of the Brownian individual to the algorithm is based on the conception of the individual's position to the sub-population *lBest* position instead of the standard crossover and mutation operators. A pre-determined percentage of the weakest individuals are chosen to be Brownian individuals in each generation. The Brownian individuals are modified by adding gaussian noise to the individual's *(lBest)* position using:

$$\omega(x,j) = y_j + N(0,\sigma) \tag{3.6}$$

where the standard deviation, σ, defines the Gaussian distribution width, y_j is the Brownian individual and $\omega(x,j)$ is its modification.

3.4 Particle Swarm Optimization

The PSO algorithm is a population-based metaheuristic. The particle swarm concept, originally, was developed to simulate, graphically, the predictable and graceful choreography of birds [29][39] to observe the pattern that governs the capability of birds to fly all together and to abruptly change direction and then re-organize optimally [88]. It is from this initial intention that the particle swarm concept was developed into an algorithm that is simple and efficient.

Each particle is assigned a position, x , velocity, v and retains its personal best position *(pBest)*. Particle position representation symbolizes a potential solution. Velocity directs how a particle position is adjusted and portrays the exchanged particle's neighborhood information and the particle's experiential knowledge. Thus, an optimization process is driven by the velocity vector.

Particles search through the multidimensional search space by adjusting their position basing on their own experience and that of its neighbors, imitating the simple behavior observed in bird flocking [39]. As such, a particle emulates its successes and the success of its neighboring particles.

The two properties that determine the search process of each particle, i, are the particle position, $x_i(t)$ at a discrete time, t and the velocity, $v_i(t)$, that is used to renew the particle's position. Considering $x_i(t)$ as the current position of the particle, i, in the search space and adding velocity, $v_i(t)$, to $x_i(t)$, gives the particle a new position as [29]:

$$x_i(t) = x_i(t) + v_i(t+1) \tag{3.7}$$

where $x_i(t) \backsim U(x_{min}, x_{max})$.

Velocity update per dimension is calculated using Equation (3.8) which is indicative of the particle's social and cognitive components.

$$v_{ij}(t) = \omega v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y_{ij}}(t) - x_{ij}(t)] \tag{3.8}$$

where ω is the inertia weight, $j = 1, \ldots, n_x$ is the search space, $v_{ij}(t))$ represents a velocity of particle, i, at a discrete time, t, $x_i(t))$ refers to the particle's current solution, $v_i(t)$ is *pBest*, $\hat{y_{ij}}(t)$ is the swarm global best position *(gBest)* and the random coefficient, r, in the range $[0, 1]$.

The inertia weight, ω, determines the weight of the current velocity that contribute to the next velocity. Variables, c_1 and c_2 determine the weight of the cognitive and social components respectively [89]. Also, c_1 and c_2 controls the maximum value of random coefficients. Furthermore, r, enables a particle to explore new regions in hyperdimensional space, as such, it diversifies the search. Algorithm 5 depicts a *gBestPSO*.

The connection that exists between particles determine the speed at which information navigates through the population [90]. A well-connected swarm tends to converge faster towards a good solution, nevertheless, this usually does not give the particles a chance to

Algorithm 5 Particle Swarm Optimization

```
1:  BEGIN
2:  Generate and initialize $n_x$ −dimensional swarm
3:
4:  DO
5:
6:  for each particle $x_i, i = 1, \ldots, n_s$ do
7:      Evaluate, $f(x_i)$
8:
9:      if $f(x_i) > f(y_i)$ then                ▷ if $f(x_i))$ is better than pBest
10:         $y_i(t) = x_i(t)$
11:     end if
12:
13:     if $f(y_i) > f(\hat{y})$ then            ▷ if $f(y_i))$ is better than gBest
14:         $\hat{y} = y_i(t)$
15:     end if
16:     for each particle $x_i, i = 1, \ldots, n_s$ do
17:         Use Eqn (3.7) to update the position
18:         Use Eqn (3.8) to update the velocity
19:     end for
20:
21: end for
22:
23:  UNTIL termination condition satisfied
24: END
```

explore all areas of the search space, therefore, exposing them to local minima. On the contrary, the swarm with fewer connections tends to converge slowly since the particles spend more time exploring the search space.

The performance of a PSO is influenced by parameters such as number of iterations and swarm size [91][92]. Number of iterations is problem-dependent whereby too few iterations may lead to premature termination and a very large number of iterations may unnecessarily increase computational complexity given that the termination condition is based on the number of iterations.

A large swarm size, n_s, promotes swarm diversity per iteration that leads to fewer iterations. A large number of particles increases the computational complexity, that results in the degradation of the search to a parallel random search. Nevertheless, the optimum swarm size, n_s, is problem-dependent, though the literature gives a general heuristic of $n_s \in [20, 30]$ [93][94].

The PSO has been extremely successful in static environments where it mainly optimized functions with continuous-valued parameters [149][154]. Moreover, the algorithm was widely applied in fields such as machine learning [95], function optimization [96], model classification [97] and biological application [98].

3.4.1 Particle Swarm Optimization for Dynamic Environment

The PSO in a changing environment is prune to outdated memory problem and diversity loss [87]. In essence, outdated memory occurs when the value of the *pBest* is no longer relevant to the current prevailing environment due to the shift of the optimum solution.

As the swarm moves toward the desired goal, the degree of the particle's dispersion decreases. This is commonly referred to as diversity loss. Diversity loss occurs when an algorithm converge towards the same solution as it approaches optimality. As such, it becomes difficult to explore the new search space when a change in the environment occurs which usually leads to a failure of the algorithm to adapt to new environments

[87].

However, in a static environment, the diversity loss is not a problem because particles cluster around the optimum, thereby facilitating exploitation towards the end of the search. Consequently, it is difficult to re-diversify a converged swarm once an environment change is detected. As a result, the swarm lacks diversity necessary to locate a new optimum. Therefore, standard PSO is ineffective in dynamic environments.

Outdated memory in PSO can be handled by re-evaluating all *pBest* positions when a change is detected [87]. To detect a change, PSO tracks the fitness as the algorithm progresses. If the fitness is improving or the same, then the environment is considered static, otherwise, a change has taken place, therefore, the algorithm responds by re-evaluating the *pBest* position. As such, detection and response strategy enables the algorithm to toggle between exploitation and exploration. Thus, the algorithm focuses on exploitation when the environment is static and exploration when the search space changes [99].

The detection strategy, however, poses some challenges if some changes go undetected. As such, a threshold value can be used to ensure the detection of rigorous changes or to wait for a considerable change of the search space before a response is triggered [41]. Also, a threshold is set to ignore small changes, especially for environments with high temporal severities since response in such an environment is triggered too often.

Diversity loss can be addressed by either implementing a repulsion technique among particles or using multiple swarms [87]. In the repulsion technique, a particle is assigned a radius to create a sphere of influence so that any particle within this space is repelled to a new position [100].

A multiple swarm technique implements sub-swarms that are separated to each other by a repulsive force to ensure that each sub-swarm converges around a different solution. Also, a population can consist of both neutral and charged particles to maintain diversity or can have quantum particles (discussed in Section 3.4.2).

Another method is to make use of sentry points to store a copy of fitness value at a random location [101]. As such, the environment is monitored by comparing sentry points with new fitness at each iteration. Sentries can be evenly distributed across the entire search space, or periodically relocated randomly to ensure good monitoring. Also, depending on the computational cost, re-evaluation of sentries can be at the iteration level (which results in a high cost) or periodically.

The following dynamic PSOs classified, based on the description provided in [102], as memory scheme (re-initialized PSO); multi-population scheme (multiPSO); and diversity maintenance scheme (chargedPSO) are discussed in the subsequent sections.

3.4.2 Diversity Maintenance Scheme - Charged PSO

A charged PSO, commonly referred to as quantum-inspired (QPSO), proposed by Blackwell and Branke, is a metaheuristic inspired by the principles and concepts of quantum computing [103] [104] [100]. In QPSO, quantum swarm referred to as the quantum cloud, consists of both neutral and charged particles. Neutral particles behave like standard PSO particles that use velocity and position equations to improve the current solution. Therefore, the quantum particles search for new solutions [103].

At each iteration, the quantum particles are randomly placed within the quantum cloud which is a multidimensional spherical area, B_n, of radius, r_{cloud}, centered on the *gBest* as:

$$x_i(t+1) = \begin{cases} x_i(t) + v_i(t+1) & \text{if} \quad C_i = 0 \\ B_n(r_{cloud}) & \text{if} \quad C_i \neq 0 \end{cases} \tag{3.9}$$

where C_i is the charge of the particle.

Randomizing at each iteration preserves the swarm diversity to discourage the quantum cloud to completely converge on a small area. Additionally, a random positioning of particles in a non-fully connected neighborhood topology facilitates easy information exchange between inter-connected particles. Algorithm 6 depicts a quantum-inspired

PSO.

Algorithm 6 Quantum-inspired Particle Swarm Optimization

```
1: BEGIN
2: Generate and initialize $n_x$ −dimensional swarm
3: Set $n\%$ of the swarm to be quantum particles and $(100-n)\%$ to be neutral particles
4: DO
5:
6: for each particle $x_i, i = 1, \ldots, n_s$ do
7:     Evaluate, $f(x_i)$
8:
9:     if $f(x_i) > f(y_i)$ then                ▷ if $f(x_i))$ is better than pBest
10:        $y_i(t) = x_i(t)$
11:    end if
12:
13:    if $f(y_i) > f(\hat{y})$ then             ▷ if $f(y_i))$ is better than gBest
14:        $\hat{y} = y_i(t)$
15:    end if
16:    for each particle $x_i, i = 1, \ldots, n_s$ do
17:        Use Eqn (3.9) to update the position
18:        Use Eqn (3.8) to update the velocity
19:    end for
20:
21: end for
22:
23:  UNTIL termination condition satisfied
24: END
```

3.4.3 Multi-population Scheme - Multi-swarm PSO

Multi-swarm PSO implements multiple populations, each optimizing a single solution from the set of solutions [103]. Algorithm 7 depicts a multi-swarm PSO. The algorithm keeps swarm diverse by using anti-convergence methods and repulsion. One sub-swarm, usually the one with the worst fitness, is reinitialized if *gBest* values of the two sub-swarms are within each other exclusion radius, r_{excl}. Sub-swarm's best particle, p, is commonly referred to as swarm attractor.

Algorithm 7 Multi-swarm Particle Swarm Optimization

```
1: BEGIN
2: Generate and initialize $C(n_x)$ - dimensional swarm, $S$
3: DO
4:
5: if all swarms converged then
6:     Reinitialize the worstSwarm
7: end if
8: for each swarm $S_k$ do
9:     Run a single iteration of $S_k$ using Algorithm 5
10:    for each swarm $S_k \neq S_l$ do
11:        if distance between $p_k$ and $p_l$ is $\leq r_{excl}$ then
12:            if fitness $(p_k)$ is $\leq$ fitness $(p_l)$ then
13:                reinitialize $S_k$
14:            else
15:                reinitialize $S_l$
16:            end if
17:        end if
18:    end for
19: end for
20: UNTIL termination condition satisfied
21: END
```

3.4.4 Memory Scheme - Reinitializing PSO

In a reinitialized PSO, the swarm or part of it is randomly re-initialized within the search space when a change in the environment occurs to enhance diversity [105]. Usually, the best particles are maintained within their neighborhood to monitor the best-known positions. The particle velocity is reset and the particle's current position becomes the *pBest*. As such, particles are discouraged to be attracted to their former position. Algorithm 8 depicts a reinitializing PSO.

3.5 Critical Analysis

Data analytics is the art of analyzing raw data to find trends and answer questions. Numerous data analytics processes and techniques have been automated into algorithms and mechanical processes that work over raw data. Examples of such techniques include predictive analytics which is used to answer questions about what will happen in the

Algorithm 8 Reinitializing PSO

```
1: BEGIN
2: Generate and initialize $n_x$ - dimensional swarm
3:
4: DO
5: Run a single iteration of Algorithm 5
6: if a change is detected then
7:     for each particle excluding $gBest$ do
8:         Randomly select a new position
9:         $v_{it}(t) = 0$
10:        $pBest = x_i(t)$
11:    end for
12: end if
13:
14: UNTIL termination condition satisfied
15: END
```

future. Predictive analytics, which includes machine learning and statistical techniques i.e. regression [141] and decision trees [14], make use of historical data to identify trends and determine their likelihood to recur.

Metaheuristic algorithms discussed above are ideal for nonstationary environments since the algorithms are adaptable by nature, require slight changes to their standard algorithm structures, are capable to perform in noisy environments and different problem spaces [142][143]. Metaheuristics were successfully applied to nonstationary environments in [144][103][145].

Nonstationary data is usually is made up of generating processes which change over time. Therefore, if the knowledge of the existence of a different segment is not taken into consideration, then, the induced predictive model is distorted by the past existing patterns. Thus, the challenge posed to a regressor is to select an appropriate segment that depicts the current underlying data generating process to be used in a model's construction.

In this work, metaheuristic algorithms will be hybridized to induce a predictive model on nonstationary environments with concept drift occurring. Thus, it is hypothesized that a metaheuristic hybridization is suitable for the induction of the predictive model on nonstationary data with a numerical target that tracks changes in a dataset due to concept drift. The hypothesized metaheuristic predictive approach will implement a piecewise approach to predict a target, nonlinear model and will consist of three components: dynamic DE-based clustering algorithm to extract clusters that resemble different data generating processes present in the dataset; the dynamic PSO-based model induction approach to induce nonlinear models that describe each generated cluster which approximate mapping between inputs and the target variable; and a dynamic GP that evolves model trees that define the boundaries of nonlinear models which are expressed as terminal nodes.

Figure 3.3 is an illustration of the hypothesized metaheuristic predictive approach indicative of its three components. This hypothesis is based on a critical analysis of the literature presented in the subsequent sections.

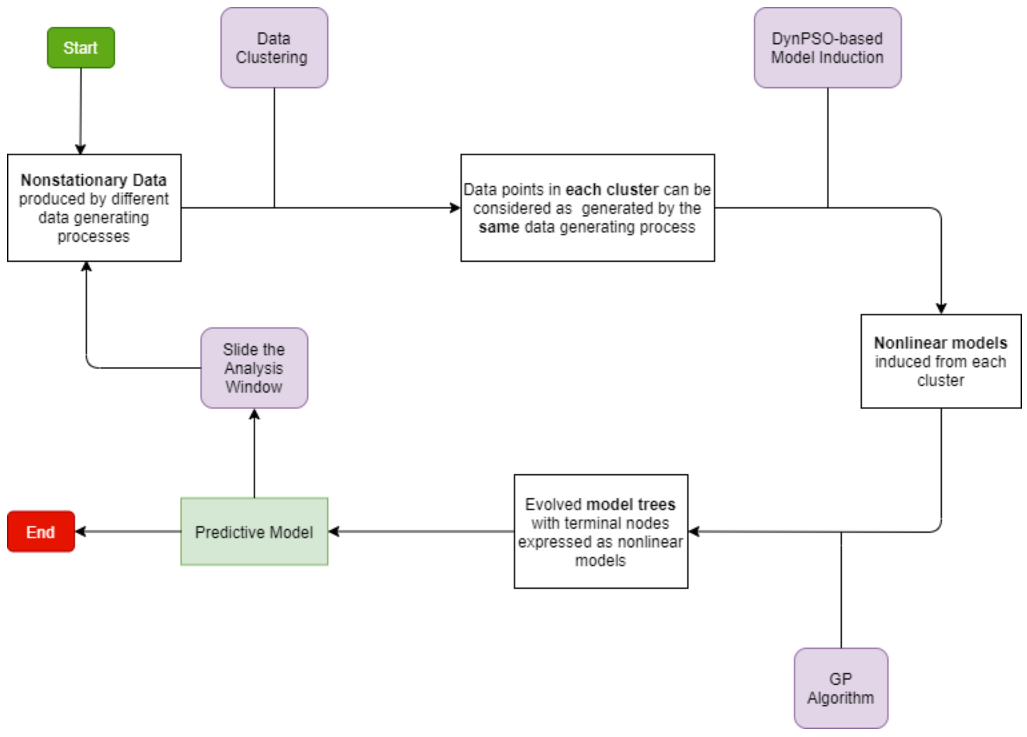

Start
Data Clustering
DynPSO-based Model Induction
Nonstationary Data produced by different data generating processes
Data points in each cluster can be considered as generated by the same data generating process
Nonlinear models induced from each cluster
Slide the Analysis Window
End
Predictive Model
Evolved model trees with terminal nodes expressed as nonlinear models
GP Algorithm

Figure 3.3: An Overview of Metaheuristic Predictive Approaches

3.5.1 Nonstationary Data Analytics

Nonstationary (temporal) data is usually made up of generating processes which change over time. By way of illustration, the temporal data represented in Figure 3.4, on inspection, shows three different patterns.

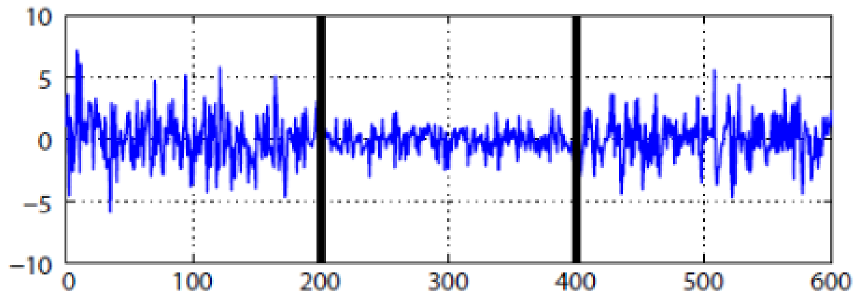


Figure 3.4: Data Series Generated by Processes which Change Over Time

That being so, it is logical to use the current segment (401-600) to generate a predictive model, since the first two segments (1-200 and 201-400) represent the patterns from past environments. Consequently, the inclusion of a considerable amount of irrelevant data and the trimming of a significant amount of data tends to reduce the quality of the model that is being generated [73]. Predictive models generated by a dataset that spans more than one underlying generating process are usually skewed. Therefore, it is essential to enumerate an optimal sliding window of analysis for any successful predictive model.

A model tree discussed in Section 2.4.3, can build a decision tree hierarchy in trying to fit several smaller data segments (i.e. 1-200, 201-400 and 401-600 in Figure 3.4) of the dataset to yield an improved model that best fits the entire training dataset. Therefore, a model tree provides a piecewise linear regression model. Thus, a model tree splits the parameter space into subspaces and then fit a linear regression model for each subspace.

Numerous methods exist such as artificial neural networks (ANN) [146] and support vector machines (SVMs) [147][148] that do the same thing as decision trees. However, decision trees possess favorable traits [14]. Decision trees are self-explanatory and may easily be understood by non-experts. Also, a decision tree is a non-parametric, which means there are no assumptions about the tree structure.

Clustering, discussed in Section 2.5, can be harnessed to split the parameter space into subspaces. Thus, clustering can group similar data patterns within a dataset so that data patterns within a cluster are more similar and data patterns among clusters are dissimilar [106][107][110]. However, an optimal number of clusters, k, in the real-world dataset is unknown prior. Therefore, it is ideal to automatically determine k. As such, the k-independent CDCDynDE (KCDCDynDE), discussed in Section 2.5.2, becomes ideal to automatically determine the optimal number of clusters whenever a change in an environment is evident [135].

3.5.2 Nonlinear Regression in Nonstationary Environments

Numerous techniques exist to solve least-squares problems such as normal equation, singular value decomposition and QR decomposition technique. The normal equation technique is very fast though the least precise, whereas singular value decomposition is the most precise though the slowest [149]. The QR decomposition technique strikes the delicate balance between precision and computational load [150].

Time-series data are usually associated with various exogenous factors such as economic fluctuation, weather conditions and special conditions which entail high nonlinearity and complicated patterns, making forecasting a difficult task [200]. An autoregressive (AR) model is a classical forecasting technique commonly used in time-series modeling which is most valuable in the presence of the vast amount of data that is highly aggregated and when there is no need for explicit separation of seasonal indices. The advantages of the AR model include strong expansion ability and is made up of simple features though fails to forecast complicated series, particularly when concept shifts occur a lot.

Nonlinear regression-based prediction can be considered as an optimization problem where the optimal parameters can be estimated by either classical or heuristic optimization techniques. However, classical techniques are usually trapped in local minima [151]. As such, metaheuristics have been considered as an alternative [152][153].

Particle swarm optimization has been successfully applied to nonlinear regression problems, in static environments, with respect to accuracy [154][149]. Usually, real-world nonlinear regression problems are dynamic. As such, the objective function in a dynamic environment tends to change, which results in changes in the structure of the search space and the position of optima. Therefore, the performance of the prediction model constructed using the past environment is bound to deteriorate. Thus, making a continuous adaptation of the prediction model becomes a necessity. However, standard PSO in a changing environment is prune to outdated memory problem and diversity loss [87]. As such, numerous PSO variants that can adapt in dynamic environments exists [102] such as reinitialized PSO; multi-population PSO, and charged PSO.

Therefore, hybridizing a classical technique (either QR decomposition or AR), to determine the coefficients of the model, and a dynamic PSO, to induce an optimal model structure that can adapt whenever a change in the environment occurs, is expected to decrease the performance deterioration in nonlinear regression designed for nonstationary environments that usually results from the environmental changes and consequently, improves the algorithm's performance.

3.5.3 Genetic Programming for Data Analytics

Genetic programming can induce nonlinear models directly from data and regression analysis on variables that show nonlinear correlations [155] [156] [157]. Also, the GP can directly induce model trees. As such, individuals in a GP can be generated using grammars, commonly referred to as grammar-guided GP, which has been applied extensively to data mining [158] [159]. A GP has been applied to induce predictive models by several authors with favorable results [160] [161] [162] [163] [164] [165].

Numerous techniques were proposed in the literature to make GP cope with nonstationary environments such as implementing: a dynamic population size [71]; adaptive parameters [72]; and the adaptive analysis window [73].

Based on the above discussion in which the knowledge of the existence of a different segment in a dataset is paramount to the induction of a predictive model of improved precision, the proposed GPANDA provides a piecewise nonlinear regression approach for nonstationary data that will hybridize a dynamic DE-based clustering algorithm, dynamic PSO-based model induction approach and dynamic GP.

3.6 Summary

In this chapter, a general overview of optimization was provided. Evolution-based metaheuristics: GP and DE were discussed. A swarm-based metaheuristic, PSO was also discussed. Also, evolution-based metaheuristics and swarm-based metaheuristic designed for dynamic environments were discussed. The discussion of metaheuristics provided in this chapter was confined to the concepts used in this thesis. A critical analysis of the relevant literature was provided.

In the next chapter, methodology presents the methodology to be applied in this work after considering the critical analysis of the relevant literature.

Chapter 4

Methodology

4.1 Introduction

A detailed research methodology used in this work to realize the objectives outlined in Section 1.2 is discussed in this chapter.

This chapter is outlined as follows: Section 4.2 discusses research methods applicable to computer science. Section 4.3 discusses the proof by demonstration methodology, whereas dataset to be used in this work is presented in Section 4.4. A comparative analysis is outlined in Section 4.5, whereas experiments to be conducted in this work are presented in Section 4.6. Technical specifications are outlined in Section 4.7.

4.2 Research Methodologies

The foundations of computer science are based on other fields such as engineering, science, philosophy and mathematics [166]. As such, myriad research methodologies in the field of computer science exist. Generally, these research methodologies can be categorized as simulation-based, experimental and theoretical. The simulation-based research methodology is more suitable when examining phenomena with great complexity which can be realized by building computer models [167]. An experimental approach solves the complex problem using software solutions (either existing or to develop from scratch). The theoretical research methodology builds theories and derives rules to prove the theories using Logic and Mathematics.

Examples of computer science research methodologies include design and creation, action research, hermeneutics, mathematical proof, empiricism, and proof by demonstration [168][169][170][171]. The design and creation methodology is most appropriate for the development of new software artefacts whereas the action research is most appropriate for the real-world computing problems [172][168][173]. Hermeneutics methodology is most appropriate for the development of software that requires continual involvement of users, therefore, the system is deployed and observed in a real-world environment whereas the mathematical proof evaluates a given hypothesis using concepts of formal mathematics [169][171].

The empiricism methodology is most appropriate for the evaluation of a certain hypothesis which can be rejected or accepted using statistical tests. The proof by demonstration methodology is analogous to methods implemented in engineering, which iteratively perform testing and refinement of a computer system until a satisfied condition is achieved or no further improvements. The feedback is used at each iteration as a corrective measure to make appropriate modifications of the system towards the expected goal [170][171].

The main goal of this work is to develop a predictive model for nonstationary data. As such, the research methodology - proof by demonstration becomes the most appropriate for this work.

4.3 Proof by Demonstration

In this section, a detailed description of the proof by demonstration on how to realize the first two objectives, *Objective (a) and (b)* outlined in Section 1.2, is proffered.

The following algorithms will be developed to achieve *Objective (a) and (b)*, the dynamic PSO-based nonlinear regression (DynPSO) and a GP approach for nonstationary data analytics (GPANDA).

The proof by demonstration methodology requires the continual testing of the implemented algorithm and use of feedback to refine the algorithm towards the desired outcome [171]. Therefore, the proof by demonstration methodology is a spiral process that alternates between testing and refinement until no further improvements or the termination conditions are satisfied.

In this section, Section 4.3.1 discusses the DynPSO algorithm whereas Section 4.3.2 discusses GPANDA algorithm.

4.3.1 The DynPSO

The DynPSO aims to develop a PSO-based nonlinear regression technique for nonstationary environments whereby the induced model dynamically adjusts when an environmental change is detected. As such, this work hybridizes dynamic PSO discussed in Section 3.4 with a classical technique (either QR decomposition or NARX), to induce optimal nonlinear regression models in dynamic environments. A detailed discussion of DynPSO is provided in Chapter 5.

The DynPSO will initially be configured using parameters provided in the literature discussed in Section 5.5. The DynPSO will be tested and the feedback will be used to refine the system until the desired outcome is realized. The steps to be taken using the proof by demonstration methodology to developing DynPSO algorithm are as follows:

1. *Create an initial algorithm*

   Based on the above critical analysis, the initial DynPSO that induces optimal nonlinear regression models in nonstationary environments will be created to realize *Objective (a)* listed in Section 1.2.

2. *Define the evaluation criterion and evaluation*

   The DynPSO will be evaluated by examining the effectiveness of the classical technique (either QR decomposition or nonlinear autoregressive with exogenous inputs model (NARX)) to determine the coefficients of the regression model and a dynamic PSO to induce an optimal model that can adapt whenever a change in the environment occurs using the datasets discussed in Section 4.4 and performance measure discussed in Section 3.2.1.

- The DynPSO is stochastic, therefore, several independent runs (= 30) will be executed using different problem instances (discussed in Section 4.4) to test the algorithm.

3. *Development*

   The DynPSO will be designed, analyzed and implemented.

4. *Refinement*

   If DynPSO fails to come up with the desired solution for at least one instance for each problem tested then at least one of the following will be adjusted:

   - The DynPSO parameters and features such as particle representation, swarm initialization technique, fitness function and the number of iterations.
   - The QR decomposition parameters such as the maximum number of terms and the order of attribute.

5. *Iteratively*, execute Step 3 - 4 until the desired outcome is realized.

4.3.2 GPANDA

This work will develop a genetic programming-based predictive approach (GPANDA) designed for nonstationary data with a numerical target that dynamically adapts when concept drift occurs and can also be used to extract knowledge from historical data. GPANDA will hybridize a dynamic clustering algorithm (KCDCDynDE), nonlinear regression approach (DynPSO) and GP to perform regression on nonstationary data with concept drift occurring. A detailed discussion of GPANDA is provided in Chapter 6.

GPANDA will initially be configured using parameters provided in the literature discussed in Section 6.6. GPANDA will be tested and the feedback will be used to refine the algorithm until the desired outcome is realized. The steps to be taken using the proof by demonstration approach to developing GPANDA are as follows:

1. *Create an Initial Algorithm* Based on the above critical analysis, the initial GPANDA that can perform regression on temporal data with concept drift occurring will be created to realize *Objective (b)* listed in Section 1.2.

2. *Define the evaluation criterion and evaluation* GPANDA will be evaluated by examining the effectiveness of the dynamic clustering algorithm, the nonlinear regression approach and GP, to perform regression on nonstationary data with concept drift occurring using the datasets discussed in Section 4.4 using the performance measure discussed in Section 3.2.1.

    - GPANDA is stochastic, therefore, several independent runs (= 30) will be executed using different problem instances (discussed in Section 4.4) to test the algorithm.

3. *Development* GPANDA will be designed, analyzed and implemented.

4. *Refinement* If GPANDA fails to come up with the desired solution for at least one instance for each problem tested then at least one of the following will be adjusted:

    - Parameters and features such as representation, initialization technique, tree-depth, offspring-depth, fitness function, the total number of generations, reproduction-operator-application-rates, and selection technique.

5. *Iteratively*, execute Step 3 - 4 until the desired outcome is realized.

4.4 Datasets

This section presents the problem instances to be used in the experiments. The real-world time series dataset to be used in this work will be differenced. For each artificially generated dataset, 10% of randomly selected target output is modified by adding Gaussian noise to the target output, y_j, to increase the complexity of the problem using:

$$y_j = y_j + N(0, \sigma) \tag{4.1}$$

where the standard deviation, σ, defines the Gaussian distribution width.

Real-world nonlinear regression datasets in dynamic environments enable performance evaluation of the induced model in real-world conditions. However, the existence of a real drift in the data is unknown, or if the drift exists, it may be unknown when exactly it occurs. As such, it becomes difficult to have an in-depth analysis of the behavior of the predictive models. Therefore, an artificially generated dataset with induced drifts becomes favorable.

Several dynamic test problems were designed to evaluate the performance of evolutionary algorithms in nonstationary environments such as the DF1 generator [186], the 'moving peaks' benchmark [187], the single and multi-objective dynamic test problem generator [188], the dynamic multi-knapsack problem and the traveling salesman problem [189], and the generalized dynamic benchmark generator [190]. However, the mentioned benchmarks are 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 GP's ability to track and adapts dynamically the induced structurally optimal nonlinear regression models in nonstationary environments will be defined.

The parameter, w_{shift}, introduces new decision boundaries in the sliding window as illustrated in Figure 4.1, thereby increasing complexity to problem under study. As illustrated in Figure 4.1, as the sliding window shift by w_{shift}, the new analysis window will consists of data points, $(p1_3, p1_4, p1_5, p2_1, p2_5)$, from two different data generating processes. As such, the sliding window of analysis will be made up of generating processes that change over time. Therefore, an algorithm/approach that is able to enumerate changes in the underlying data generating processes will yield a prediction model of higher precision.

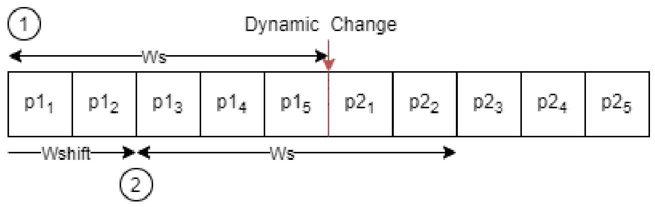


Figure 4.1: Introducing new decision boundaries by window shifts

In this work, four artificially generated, and three real-world nonstationary test environments will be used in the experiments. These 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. The following test environments will be implemented in this section.

Progressive-Bennett5A Variation

Auto-generated datasets of 10 000 patterns with 100 time-steps will be used in the experiments to be conducted in this work on different types of change period. Time-step is an incremental change in time for which the concept drift is introduced. Simply put, time-steps determine the total number of changes injected in the dataset. The datasets will be generated using the benchmark nonlinear Bennett5 function computed as [184]:

$$y = \theta_1 + (\theta_2 + x)^{-\frac{1}{\theta_3}} \tag{4.2}$$

The starting values for Bennett5 function are provided in the literature [184]. An environmental change will be simulated by adding drift to each parameter, $\theta_i = \theta_i + \delta\sigma$ where δ is the drift and σ is the probability of altering the direction of change. The following equation will be used to simulate the drift:

$$\delta = 0.6\delta^2 + 0.02\delta + 0.01 \tag{4.3}$$

The impact of the drifts will be smaller at the beginning and then improves along with an increase in the parameter *(h)* defined in Section 4.6.1. The sliding window size (w_s)

will be set to 100 data points.

New South Wales Electricity Pricing

Electricity pricing is a real-world dataset built on the electricity market in the Australian state of New South Wales (NSW) [185]. The electricity pricing dataset exhibits both short-term irregular changes due to weather fluctuations and long-term regular changes due to seasonal changes [185]. Parameters were recorded every half an hour, for the period 7 May 1996 to 5 December 1998 to create a dataset of 27 552 data points.

For DynPSO experiments, the sliding window size (w_s) will be set to 100 data points. Each algorithm requires 275 slides to traverse the complete dataset. Given that 100 iterations will be executed before the sliding window slides, therefore, for the training algorithm to traverse the complete dataset, 27 600 iterations will be executed.

For GPANDA experiments, an analysis window size of 2 500 data points and w_{shift} of 125; 250; 500; 1 250 and 2 500 will be implemented in this dataset.

Australian Energy Market Operator Electric Load

The Australian Energy Market Operator (AEMO) electric load datasets consist of 15 electric load datasets of five states of Australia of the year 2013-2015, namely Victoria (VIC), South Australia (SA), Queensland (QLD), Tasmania (TAS) and New South Wales (NSW). Each day have 48 data points sampled every half an hour; therefore, each dataset consists of 17 520 data points for each year [197]. Table 4.1 summarizes the statistics of AEMO electric load datasets.

The following temperature data for each state will also be considered: Melbourne for VIC, Adelaide for SA, Brisbane for QLD, Launceston airport and Hobart for TAS, and Canberra and Sydney for NSW [198]. The min and max daily temperature data will be considered as exogenous variables.

The entire training set is scaled and then split into training (first 9 months) and test

Table 4.1: Summary of AEMO electric load dataset

Dataset	Year	Data points	Max	Min	Mean	std
VIC	2015	17520	8579.9	3369.1	5194.6	864.7
	2014	17520	10240	3272.9	5324.4	921.4
	2013	17520	9587.5	3551.6	5511.8	895.9
SA	2015	17520	2870.4	696.3	1398.5	306.0
	2014	17520	3245.9	682.5	1403.3	312.8
	2013	17520	2991.3	728.6	1426.6	301.7
TAS	2015	17520	1667.2	479.4	1138.2	145.3
	2014	17520	1630.1	569.1	1109.7	139.0
	2013	17520	1650.3	659.5	1129.3	142.3
NSW	2015	17520	12602	5337.4	7979.8	1232.7
	2014	17520	11846	5138.1	7917.8	1170.1
	2013	17520	13788	5113.0	7981.6	1190.9
QLD	2015	17520	8808.7	4281.4	6035.4	777.2
	2014	17520	8445.3	4073.0	5745.7	794.0
	2013	17520	8278.4	4148.7	5703.7	747.0

sets (remaining 3 months). Scaling is done using:

$$\bar{x_i} = \frac{x_u - x_i}{x_u - x_l} \tag{4.4}$$

where $\bar{x_i}$ is the scaled value, x_i is the data point, x_l is the minimum value and x_u is the maximum value in the dataset.

The AEMO dataset will serve the purpose of providing a comparison between the DynPSO results and those of the leading studies [198].

Progressive-Bennett5B Variation

A gradual problem set will be generated using the nonlinear benchmark function, Bennett5, to generate a dataset of 10 000 data points with 10 time-steps. The Bennett5 function is defined in Equation (4.2). The change period occurs at each time-step. A sliding window of analysis of $w_s = 1000$ data points slides from one time-step, which consists of 1000 patterns, to the next using a w_{shift}.

Table 4.2 presents 20 different dynamic scenarios, denoted as A, B, C, and D, which will be applied to a progressive problem set. Different combinations of spatial and temporal severity presented in Table 4.2 simulates different dynamic environments from most abrupt changes to gradual changes as Scenario A - Abrupt; Scenario B - Quasi-Abrupt; Scenario C - Quasi-Progressive and Scenario D - Progressive. The set of values to be considered for temporal *(te)* and spatial severity (w_{shift}) are summarized in Table 4.2.

Recurrent Variation

Table 4.2: Severity Change Values for Progressive Dataset

	w_{shift}				
te	50	100	250	500	1000
25	A_1	A_2	A_3	A_4	A_5
50	B_1	B_2	B_3	B_4	B_5
100	C_1	C_2	C_3	C_4	C_5
250	D_1	D_2	D_3	D_4	D_5

In the recurrent problem set, modification to the environment will be realized by alternating two target functions. The target functions differ from each other in only one operator. The dataset consists of 10 000 data points. Each target function generates 5 000 data points. The data points will be recorded in sequential order, alternating as a block of 1000 patterns, into a single dataset, block by block. The target functions will be generated using:

$$f_1(x_0, x_1) = x_0^5 - 5x_0^3 + 4x_0 + 5x_1^5 - 5x_1^3 + 4x_1 \text{ with } x_0, x_1 \in [-2, 2]$$

$$f_2(x_0, x_1) = x_0^5 - 5x_0^3 + 4x_0 - 5x_1^5 - 5x_1^3 + 4x_1 \text{ with } x_0, x_1 \in [-2, 2]$$

The size of the sliding window of analysis, w_s, will be set equal to the size of the concept block which is 1000 data points.

Abrupt Variation

The abrupt problem set consists of a dataset of 8 000 data points generated by four different target functions, each generating 2 000 data points. Data points will be recorded in a sequential order, block by block to form a dataset. A sliding window of analysis, w_s, will be set to 2 000 data points. The following artificially generated functions will be used to simulate an abrupt problem set [191]:

$$f_3(x_0, x_1, x_2, x_3) = x_0^5 - 5x_1^3 + 4x_2 + 5x_0^5 - 5x_1^3 + 4x_3 \text{ with } x_0, x_1, x_2, x_3 \in [-2, 2]$$

$$f_4(x_0, x_1, x_2, x_3) = exp(2x_0 \sin(\pi x_3)) - \sin(x_1 x_2) \text{ with } x_0, x_1, x_2, x_3 \in [-1, 1]$$

$$f_5(x_0, x_1, x_2, x_3) = x_0^5 + 5x_1^3 + 4x_2 - 5x_0^5 - 5x_1^3 + 4x_3 \text{ with } x_0, x_1, x_2, x_3 \in [-2, 2]$$

$$f_6(x_0, x_1, x_2, x_3) = exp(2x_0 \sin(\pi x_3)) + \sin(x_1 x_2) \text{ with } x_0, x_1, x_2, x_3 \in [-1, 1]$$

Random Variation

A random problem set will be generated using Equation (4.2) and consists of 10 000 patterns with 10 time-steps. There will be a 50% chance of modifying the data in the current moment whereby each operator has an equal chance of being inverted. Thus, a negative member becomes positive and vice-versa. As such, the dataset will be randomly modified. A sliding window of analysis of $w_s = 1000$ data points slides from one time-step, which consists of 1000 patterns, to the next using a w_{shift} of 250; 500 and 1 000.

Trend Variation

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 dataset, *historical US Treasury bill contracts*, for the period January 1984 to March 1986 is selected to depict trend variation [192]. A sliding window of analysis of $w_s = 111$ data points and w_{shift} of 20; 42 and 111 will implemented in this dataset.

Stock Market

Two real-world stock market datasets, the Gross Domestic Product (GDP)(US), and the Consumer Price Index inflation rate (CPI)(US) will be implemented in the current study. The GDP dataset is from January 1982 to March 2003 whereas CPI dataset is from January 1950 to December 1983. The selected datasets will serve the purpose of providing a comparison between GPANDA results and those of the leading studies [181][182][73].

A Table 4.3 summarizes the characteristics of the seven problem datasets discussed in above indicating the dataset used to test a given algorithm(s)

Table 4.3: Summary of Datasets and Corresponding Algorithms Used

Test Environment	**Dataset**	**Algorithm(s)**
Progressively Changing	Bennett5A	DynPSO
	Bennett5B	GPANDA
Chaotically Changing	Random	GPANDA
Cyclic	Recurrent	GPANDA
Abruptly Changing	Abrupt	GPANDA
Real-world	NSW Electricity Pricing	DynPSO, GPANDA
	AEMO Electric Load	DynPSO, GPANDA
	Trend	GPANDA
	CPI Inflation	GPANDA
	GDP (US)	GPANDA

4.5 Comparative Analysis

To achieve *Objective (c) and (d)* outlined in Section 1.2, a comparative analysis will be performed on the results obtained from the execution of the experiments discussed in Section 4.6 using the datasets discussed in Section 4.4.

- DynPSO will be run and compared to dynamic PSOs, namely multiPSO, charged-PSO and reinitialize-PSO discussed in Section 3.4;
- DynPSO will be run and compared to conventional Support Vector Regression (SVR) [175][176] and standard Random Vector Functional Link network (RVFL) with direct input-output connections that retain biases [177][178]; and
- GPANDA will be run and compared to DynGP and DyFor GP discussed in Section 3.3.

Each state-of-the-art algorithm will be run until it converges before a sliding window of analysis is applied to ensure fairness.

4.5.1 Statistical Tests

To ascertain the significance of the results obtained by the proposed algorithms discussed above and the dynamic GPs and dynamic PSOs, statistical analysis will be implemented.

A direct comparison between the two best mean fitness values for algorithms that are stochastic in most cases is misguiding. Several experiments are sensible to yield a meaningful sample of values for best fitness. For each algorithm, a total of 30 independent runs will be executed on each given dataset. A Kruskal-Wallis test will be performed first to establish if there exists a statistically significant difference between the mean fitness values of the algorithms for a given problem. If there exists a statistical difference between the algorithm's performances, a Mann-Whitney U test will be performed at a significance level of 0.05. The obtained Mann-Whitney U-values determine the winning and losing algorithm. The overall performances will be ranked based on the difference between wins and losses of each algorithm.

A test will be performed for the algorithms' mean fitness values, μ_1 and μ_2 , whereby $H_0 : \mu_1 = \mu_2$ and $H_1 : \mu_1 \neq \mu_2$. These tests will be performed for every combination of algorithms and all problems.

4.6 Experiments

This section describes the experiments that will be conducted for DynPSO and GPANDA. A brief description of each experiment is given in the subsequent sections.

The optimal values for each parameter of each algorithm implemented in the experiments will be obtained using the F-race algorithm [174].

4.6.1 Dynamic PSO-based Nonlinear Regression Approach

To achieve *Objective (c)*, a comparative analysis to the dynamic PSOs, namely multi-PSO, charged-PSO and reinitialize-PSO discussed in Section 3.4 will be used to benchmark the performance of DynPSO.

For Progressive-Bennett5A Variation and New South Wales Electricity Pricing datasets, described in Section 4.4, a sliding window of $w_s = 100$ data points slides from one time-step, which consists of 100 data points, to the next. The environmental change occurs at severities and frequencies (discussed in Section 2.2) of 1 to 5.

The severity of change determines the probability of altering the direction of change, σ, where a value of 5 implies that the reverse direction of change will be certain at each environmental change. To simulate a dynamic environment, drifts will be introduced in the dataset at a given time-step. The time-step at which the change occurs (changePoint) will be determined as:

$$changePoint = \frac{h}{10} \times T \tag{4.5}$$

where T is the total number of iterations and $h \ \epsilon \ \{1 - 5\}$ determines the time-step at which change will be introduced in the dataset. A high value of h implies that fewer changes will be occurring to the dataset, i.e., if $h = 2$, then change occurs after every fifth of the total iteration whereas if $h = 1$, then change occurs after every tenth of the total iteration.

For electric load dataset, the input features: month, day, hour and the load value $x_{(t-48)}$ will be used to induce a multiple linear regression model. The induced regression model will be used to predict x_t for 48 steps horizon (one day).

4.6.2 Genetic Programming Approach for Nonstationary Data Analytics

To achieve *Objective (d)*, a comparative analysis to the dynamic GPs, namely DynGP and DyFor GP (discussed in Section 3.3), will be performed on the obtained results of applying GPANDA to the datasets described in Table 4.3.

Four artificially generated, and three real-world nonstationary problem sets will be used in the experiments. These seven problem sets differ in the total number of underlying data generating processes and the dimensionality of the problem.

Each problem set is characterized by two variables: w_{shift}, to ascertain the degree of spatial severity and the frequency of change, g (number of iterations), to ascertain the degree of temporal severity. In each experiment, the frequencies *(g)* of $25; 50; 100; 250$ iterations are implemented. The spatial severities (w_{shift}) of $50; 100; 250; 500; 1000$ to simulate different dynamic environments from gradual changes to most abrupt will be implemented on the artificially generated datasets. The effect of parameter values will be expected to be stronger for smaller values, therefore, temporal and spatial severity increases nonlinearly.

4.7 Technical Specification

All the experiments conducted in this work will be implemented in a MATLAB programming environment [193] on an Intel Core i7 processor (3.1 GHz) with 16 GB of memory on a Linux Centos 7 system. The statistical significance of the differences in the algorithm's performances will be calculated using the R statistical package.

4.8 Summary

This chapter discussed the research methodology to be implemented to examine GPANDA and DynPSO. The datasets and performance measure to be used in the experiments were presented. Also, experiment to be executed were presented and statistical test to used were discussed.

The next chapter discusses DynPSO presented in this chapter.

Chapter 5

Dynamic PSO-based Nonlinear Regression Approach

5.1 Introduction

This chapter presents the proposed dynamic PSO-based nonlinear regression approach tailored for nonstationary environments. The proposed technique hybridizes a dynamic PSO with a regression model (either linear regression - QR decomposition or autoregressive) to induce optimal nonlinear regression models.

This chapter is structured as follows: Section 5.2 discusses the proposed PSO-based nonlinear regression approach. Section 5.3 discusses linear regression whereas Section 5.4 discusses autoregressive. Section 5.5 discusses parameter optimization.

5.2 Particle Swarm Optimization in Regression Analysis

A hybrid technique, dynamic PSO-based nonlinear regression approach (DynPSO), that consists of dynamic PSO and a regression model (either QR decomposition or NARX) is proposed. DynPSO fits a nonlinear relationship between the value of the independent variable and corresponding dependent variable(s). The regression model: NARX or QR decomposition, is used to estimate the parameters of the model whereas QPSO selects the model structure based on the data characteristics. Thus, QPSO optimizes the functional form of the multidimensional polynomial fit to experimental data. As such, QPSO reduces the number of terms required in comparison to a least-square fit using all possible terms. Also, DynPSO adapts whenever a change in the environment occurs. Algorithm 9 summarizes DynPSO.

Considering Algorithm 9, swarm particles are initialized in line 2. Line 4 calls a dynamic PSO algorithm which is executed for the given number of iterations. Line 5 simulates a dynamic environment. If an environmental change is detected, line 6 adapts the current model to the prevailing data points. The algorithm terminates in line 8.

Algorithm 9 Dynamic PSO-based Regression Algorithm

```
1: BEGIN
2: Determine the coefficient of the regression model using either QR or NARX
3: Initialize swarm particles using Equation (5.4) or Equation (5.7) that consist of
   unique, term-coefficient mappings from a set, $\xi$ defined in (5.5)
4: DO
5: Slide a data window of analysis
6: Run $n$ iterations of the dynamic PSO
7: if  a change in the environment is detected then
8:     Reduce $b \approx A\tau$
9:     Update the coefficients of term-coefficient mappings in each particle
10: UNTIL all data points are analyzed.
11: END
```

Three versions of DynPSO are created by implementing three dynamic PSO algorithms (discussed in Section 3.4), namely multi-swarm, reinitialized, and charged PSOs. As such, the training algorithms are categorized into QR-PSOs and nonQR-PSOs. The QR-PSOs consist of dynamic PSOs namely, charged PSO (QPSO), multi-swarm PSO (mPSO) and re-initialized PSO (rePSO) that are implemented in the proposed Algorithm 9 in line 4 and therefore, referred to as QR-QPSO, QR-mPSO and QR-rePSO respectively. Table 5.1 summarizes the mentioned six DynPSOs.

Table 5.1: Summary of DynPSOs Algorithms

nonQR-PSOs	**QR-PSOs**
charged PSO (QPSO)	QR-based charged PSO (QR-QPSO)
multi-swarm PSO (mPSO)	QR-based multi-swarm PSO (QR-mPSO)
re-initialized PSO (rePSO)	QR-based re-initialized (QR-rePSO)

To benchmark the performance of the proposed techniques, nonQR-PSOs (QPSO, mPSO and rePSO) are implemented as discussed in Section 3.4. A total of six experiments (described in Section 4.6) are executed.

To model nonstationary time series, another version of DynPSO is created by replacing the QR component in DynPSO discussed above by NARX model (discussed in Section 5.4).

The best performing dynamic PSO algorithm from the three dynamic PSO algorithms mentioned above will be hybridized with NARX model. This hybridization aims to evaluate the effectiveness of the regression models, namely QR decomposition and autoregressive, in DynPSO technique.

5.2.1 Fitness Function

The adjusted coefficient of determination, R_a^2, is implemented in all algorithms mentioned above to measure the fitness of each particle and is defined as:

$$R_a^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_{j,i})^2}{(y_i - \bar{y})^2} \times \frac{n-1}{n-k} \tag{5.1}$$

where the predicted output for j^{th} particle for the i^{th} data point is $y_{j,i}$, y_i is the target output, n is number of samples, $\bar{y}$ is the mean value of target output and k is the number of coefficients. The R_a^2 penalizes a model that has a larger number of coefficients, k. Thus, the objective of R_a^2 is to minimize the model's architecture, whereas maximizing the correlation between the dataset and the induced model.

5.2.2 Detecting Environmental Changes

In this work, a simple and efficient method to detect environment change used in [195] is adopted that uses the *personal Best position* of each particle which is re-evaluated before being updated. As such, fitness deterioration implies that an environmental change had occurred. This environmental change detection technique is implemented in all algorithms mentioned above.

5.3 QR Decomposition

Considering the linear system:

$$\mathbf{A}x = b$$

where $\mathbf{A}$ is an $m \times n$ matrix, $m \geq n$ with linearly independent columns, thus $\mathbf{A} \in \mathbb{C}_{\rho}^{m\times n}$, and the data $b = \mathbf{C}^n$.

Equation $\mathbf{A}x = b$ is simply the least squares problem that minimizes $\|\mathbf{A}x = b\|_2^2$ in which a solution exists that is defined by:

$$x_{LS} = \{\ x \in \mathbb{C}^n : \|\mathbf{A}x = b\|_2^2\ \ is\ minimized\ \}$$

and the minimizers are the affine set computed by:

$$x_{LS} = \mathbf{A}^{+}b + (\mathbf{I}_n - \mathbf{A}^{+}\mathbf{A})y,\ \ y \in \mathbb{C}^n$$

The goal is to come up with a solution with the smallest norm, thereby solving the optimization problem:

$$minimize\|x\|$$
$$x \in \Re^n$$
$$subject\ to\ \mathbf{A}x = b$$

Given that $\mathbf{x}$ is an input vector and y is the output, a regression model between $\mathbf{x}$ and y is of the form:

$$y = f(\mathbf{x};\beta) + \epsilon$$

where f can be linear or nonlinear function, β is the parameter vector and ϵ a random error. The parameter vector is calculated using the least-squares method. Least-squares approximation minimizes the least-squares error and is computed as:

$$E_{ss} = \sum_{i=1}^{p} [y_i - \hat{y}_i]^2$$

where p is the number of data points, y_i is the i^{th} output and $\hat{y}_i$ is the i^{th} predicted output. The predicted output can be expressed as:

$$\begin{aligned} \hat{y}_i &= \tau_0 + \tau_1 x_1 + \ldots + \tau_{n-1} x_i^{n-1} + \tau_n x_i^n \\ &= \sum_{j=0}^{n} \tau_j x_i^j \end{aligned} \tag{5.2}$$

where τ_j is the j^{th} term coefficient, n is the number of terms. The determination of coefficients $\tau_0, \ldots, \tau_n x_n$ is realized through solving the normal equation:

$$\mathbf{y} = X\tau \tag{5.3}$$

where $\tau^T = [\tau_0, \tau_1, \ldots, \tau_{n-1}, \tau_n]$ and $\mathbf{y}^T = [y_0, y_1, \ldots, y_{n-1}, y_n]$.

A QR decomposition technique is used to solve least-squares problems [194]. The QR decomposition, implemented in this work is computed using Gram-Schmidt technique.

Considering a QR decomposition technique, the predicted output, $\hat{y}_i$, in Equation (5.2), when the dimensionality of the input space, d, is taken into consideration, is expressed as [5]:

$$\hat{y}_i = \sum_{\sum_{j=1}^{d} \beta_j = 0}^{m} \left(\tau_{(\beta_1, \beta_2, \ldots, \beta_d)} \prod_{q=1}^{d} x_{i,q}^{\beta_d}\right) \tag{5.4}$$

where m is the maximum polynomial order, $\tau_{(\beta_1, \beta_2, \ldots, \beta_d)}$ is a real-valued coefficient and β_d is the natural-valued order. QR decomposition technique determines the value of the coefficients, $\tau_{(\beta_1, \beta_2, \ldots, \beta_d)}$. Therefore, the dynamic PSO is tasked to determine only the structure of the model.

The structure of each particle in dynamic PSO is the structure of the nonlinear polynomial in one variable. Thus, each particle is a representation of Equation (5.4) and consists of unique, term-coefficient *(t)* mappings from a set, S: [5]

$$S = \{(t_0 \rightarrow \tau_0), ..., (t_k \rightarrow \tau_k)\} \tag{5.5}$$

where $\tau_j, j \in \{0, ..., k\}$ is a real-valued coefficient and k is the maximum number of terms. Each term, t_j, consists of a set, T, of unique, variable-order mappings, e.g.

$$T = \{(x_{i,1} \rightarrow \beta_1), ..., (x_{i,d} \rightarrow \beta_d)\}$$

where $x_{i,j}, j \in 1 \dots q$ is an input variable-integer representation, q is the number of input variables. The coefficients of term-coefficient mappings are determined by reducing $\mathbf{y} = X\tau$.

Each particle in a swarm is initialized by randomly selecting variable-order pairs, in order to build a term, up to a maximum polynomial order $e \in \{0, ..., m\}$. This process is repeated until the number of terms equals the maximum number of terms n. The dynamic PSO is then tasked to optimize the form of a polynomial, i.e., reducing the number of terms required in comparison to a least-squares fit using all possible terms [200].

5.4 Autoregressive Model

Autoregressive models estimate future value using a weighted sum of past values. An AR model can be expressed as:

$$y_t = f(y_{(t-1)}, y_{(t-2)}, ..., y_{(t-l)}) + \varepsilon_t \tag{5.6}$$

where f is a nonlinear function, y_t is the observed data at time period t, l is the number of lags and ε_t is the forecasting error term at time period t.

A variant of autoregressive model tailored for nonlinear regression, nonlinear autoregressive with exogenous inputs model (NARX), is adopted in this study. NARX model estimates future value using both the weighted sum of past and current values and external past inputs values that affect the target output.

The NARX model can be expressed as:

$$y_t = f(y_{(t-1)}, y_{(t-2)}, \ldots, y_{(t-l)}; \chi_{(t-1)}, \chi_{(t-2)}, \ldots, \chi_{(t-q)}) + \varepsilon_t$$

where $l \in \{0, ..., N\}$ and q are number of lags and χ is the exogenous variable.

Also, the NARX model can be expressed as:

$$\begin{bmatrix} y_{l+1} \cdots y_N \end{bmatrix} = \begin{bmatrix} A_1 \cdots A_l & B_1 \cdots B_l \end{bmatrix} \begin{bmatrix} y_{l+1} \cdots y_{N-1} \\ \vdots \ddots \vdots \\ y_1 \cdots y_{N-l} \\ \chi_l \cdots \chi_{N-1} \\ \vdots \ddots \vdots \\ \chi_{l-q} \cdots \chi_{N-q} \end{bmatrix} + \begin{bmatrix} \varepsilon_{l+t} \cdots \varepsilon_N \end{bmatrix} \tag{5.7}$$

where matrix A describes the relationship between the inputs and matrix B describes the relationship between each input and exogenous variables. The coefficients of the matrix are predicted by applying the unconstraint least-squares approximation based on the Granger causality which aims to minimize the error between data and model's outputs.

As such, NARX is used to enumerate the value of the coefficients and dynamic PSO is then tasked to determine only the structure of the model. For a NARX-based DynPSO approach, each particle in Algorithm 9 is a representation of Equation (5.7).

5.5 Parameter Tuning

The optimal values for each parameter of each algorithm implemented are obtained using the F-race algorithm, only a single run is performed [174]. The parameter optimization is done across domains. The cross-domain technique aims to come up with a parameter setting that produces favorable performance across several domains [196]. The swarm of each PSO is initialized within the bounds of the dataset and the optimized parameter values listed in Table 5.2 are implemented in the experiments. The *quantumSize* and *reInit* (swarm to be reinitialized) are expressed as a percentage of *swarmSize*. For RVFL, the number of hidden neurons is set to 100 and the regularization parameter: λ, is set to be 2^C where C is tuned over [-5, 14] [178].

Table 5.2: DynPSO Parameter Values

Algorithm	**Parameter**	**Bounds**	**Value**
PSO	$c_1 = c_2$	[0,3]	1.496
	ω	[0,1]	0.730
	swarmSize	[0,50]	30
	iterations	[0,∞]	100
QPSO	r_{cloud}	[0,5]	2
	quantumSize	[0,100]	50%
mPSO	r_{excl}	[0,5]	1.3
rePSO	re_{init}	[0,100]	50%
QR	numberOfTerms	[1,20]	15
	maxPower	[1,25]	20
SVR	ϵ	[0,1]	0.2

5.6 Summary

In this chapter, the proposed DynPSO to induce nonlinear regression model in a nonstationary environment was discussed. A discussion of DynPSO that consists of two components: regression model (either - QR decomposition or NARX) to estimate parameter coefficients of the model and QPSO to extract and optimize the structure of the model was provided. Three variants of DynPSO were presented namely: QR-QPSO, QR-rePSO and QR-mPSO. To simulate a nonstationary environment, a sliding window of analysis was adopted. An environmental change detection technique used in DynPSO to adapt to nonstationary environments was discussed.

In the next chapter, a GP-based approach for nonstationary data analytics discussed in the methodology chapter is discussed.

Chapter 6

Genetic Programming Approach for Nonstationary Data Analytics

6.1 Introduction

This chapter presents the proposed GP approach for nonstationary data analytics (GPANDA). Section 6.2 provides an overview of GPANDA and Section 6.3 discusses GPANDA's individuals and initial population generation. Section 6.4 discusses the nonlinear models in GPANDA which are expressed as GP terminal nodes. Section 6.5 discusses the environment change detection used in GPANDA whereas Section 6.6 discusses GPANDA parameter tuning.

6.2 An Overview of Genetic Programming Approach for Nonstationary Data Analytics

GPANDA implements a piecewise approach to predict a target, nonlinear model and consists of three components: a clustering algorithm, nonlinear regression technique, and GP algorithm. GP algorithm evolves model trees with terminal nodes expressed as nonlinear regression models. Algorithm 10 summarizes GPANDA.

Considering Algorithm 10, the dataset is clustered to enumerate clusters that represent data points from different data generating processes whereby a model is fit for each cluster in line 4. Each induced nonlinear regression model forms part of GP terminal set. An initial GP population is created and the fitness of each individual is calculated in line 5. In line 7, an updated sliding window of analysis that depict the current data is presented to GPANDA. Line 8 handles environmental changes in the dataset. The GP optimization cycle is entered in line 9, which terminates if the termination condition is met in line 16.

Algorithm 10 GPANDA

```
1:  BEGIN
2:  Set the size of the sliding window of analysis to $w_s$ data points
3:  Let $t = 0$
4:  Create GP terminal set using Algorithm 11
5:  Create individuals that constitute the $Pop(0)$ using Algorithm 12
6:  DO
7:      Slide a sliding window of analysis
8:      Perform Environmental change update process using Algorithm 13
9:        WHILE termination condition(s) not satisfied DO
10:           Select individuals from $Pop(t)$ to use in reproduction process
11:           Perform crossover using selected individuals to create offspring
12:           Perform mutation on the offspring
13:           Compute fitness for each offspring
14:           Select individuals to constitute a new population, $Pop(t + 1)$
15:           $t = t + 1$
16:       UNTIL no further data to analyze
17:   END
18: END
```

Selection in GP is performed using the tournament selection technique. The crossover operator picks a crossover point (it disallows extreme points) 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 subtrees.

The detailed discussion of the main aspects of GPANDA is provided in the subsequent sections.

6.3 Individuals and Initial Population

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

An example of a GPANDA model tree is graphically illustrated in Figure 6.1. Considering Figure 6.1, nodes are recursively added within a maximum tree depth to a root node ($x_2 < v_2$). As illustrated in Figure 6.1, y_i is the i^{th} nonlinear model from the terminal set. Nonlinear models, y_i, are picked randomly from the terminal set. The discrete or continuous-valued attribute is represented as x_i whereas v_i is a possible value of x_i and the set of operators, $Op \, \epsilon \, \{<, >, =, \neq\}$.

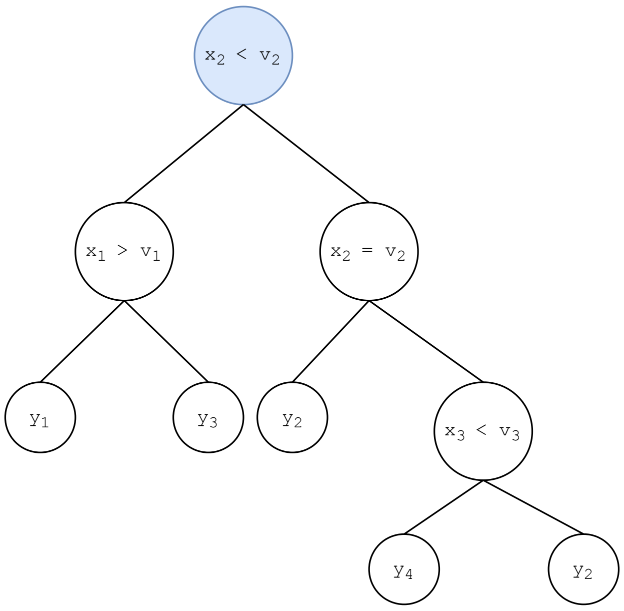


Figure 6.1: An example of GP Model tree

It is ideal to initialize a small tree which is expected to grow during the evolutionary process if an increased complexity is necessary. GP trees are evolved using the grow method.

Considering Algorithm 11, data clustering is performed in line 2 to split the parameter space into subspaces. Line 3-6 fits a nonlinear regression model, A_r, on each generated cluster which then constitutes GP terminal set.

Each nonlinear regression model is assigned a lifetime which expires if it exceeds the upper-bound to avoid the size of the terminal set becoming unnecessarily large which will tend to make the search space larger. 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 11 GP Terminal Set Initialization Process

```
1: BEGIN
2: Perform data clustering using KCDCDynDE to obtain $k$ clusters
3: for each cluster do
4:     Perform nonlinear regression using DynPSO to obtain a model, $A_r$
5:     Insert the induced model, $A_r$, into the terminal set
6: end for
7: END
```

Considering Algorithm 12, a model tree is created by recursively adding nodes within the maximum tree depth set in line 3. In line 4, $U(0,1)$ is a uniformly distributed random number and Op is a operator set. In line 5, there is an equal chance of picking either a terminal node or an internal node. If an internal node is selected, line 6 - 10 picks the function from the internal set and an attribute value for the attribute, A_ϵ, to be used in tree building in line 11 - 14, else a terminal node, A_r, is picked from the terminal set in line 18 - 21. Once the termination condition is satisfied, a *Caller* in line 23 returns the induced model tree.

The root mean square error, which is the square root of E_{MS}, discussed in Section 3.2.1, calculated over the dataset on each generation is used to evaluate the fitness of individuals. RMSE calculates the misfit of the regression estimate to its expected output.

6.3.1 Mutation Operator

Different mutation operators used to introduce diversity into the population of programs are adopted in this work [5]. 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 mean squared error (defined in Section 3.2.1), E_{MS}, is used to determine the relative error of a subtree or terminal node, A_r. The following mutation operators are implemented:

- Perturb-worst-non-terminal-node operator

Algorithm 12 Evolving GP individual

```
1:  BEGIN
2:  Set CALLER (a calling node) to $Nil$
3:  Set the depth to $maxTreeDepth$
4:  if fitness $U(0,1) < 0.5)$ and depth $< maxTreeDepth$ then
5:      Select $A_\epsilon$ from the attribute set
6:      if $A_\epsilon$ is a continuous-valued then
7:          $Op(\epsilon) \in \{<, >, =, \neq\}$
8:      else
9:          $Op(\epsilon) \in \{=\}$
10:     end if
11:     For attribute, $A_\epsilon$, select an attribute value, $a_{(\epsilon,i)}$ from a training pattern, $\imath$, such
          that $v_\epsilon = a_{(\epsilon,i)}, i \in \{1, \ldots, |P|\}$ where $v_\epsilon$ is a possible value of $A_\epsilon$
12:     Create a node, $N_\gamma$ with the antecedent, $ant_\gamma = (A_\epsilon\ op(\epsilon)\ v_\epsilon)$, and consequent
          $con_\gamma = Nil$
13:     Cover CALLER with the $ant_\gamma$ (left node) and increment depth by 1
14:     Cover CALLER with $\neg ant_\gamma$ of $N_\gamma$ (right node) and increment depth by 1
15: else
16:     Select $A_r$
17:     Build a node $N_\gamma$ with the antecedent, $ant_\gamma = Nil$ and consequent, $con_\gamma = A_r$:
18:     $ant_\gamma \leftarrow Nil$
19:     $\neg ant_\gamma \leftarrow Nil$
20: end if
21: Assign CALLER := $N_\gamma$ and return
22: END
```

A nonterminal node in an individual, at the given depth, with a higher relative error is perturbed. This operation enables an adjustment on the internal node described by the worst nonterminal nodes. Figure 6.2 is a graphical illustration of a perturb-worst-nonterminal-nodes operator where N_j is the worst nonterminal node to be perturbed.

- Perturb-worst-terminal-node operator

  A terminal node in a target model tree with a higher E_{MS} is perturbed. This operation enables the replacement of a terminal node with a randomly selected nonlinear model, A_r, from the terminal set. Figure 6.3 is a graphical illustration of the perturb-worst-terminal-node operator where N_x is the worst terminal node to be perturbed.

6.4 Nonlinear Models

The nonlinear regression technique constructs optimal nonlinear models, which constitutes the GP terminal set, which approximate mapping between inputs and the target variable.

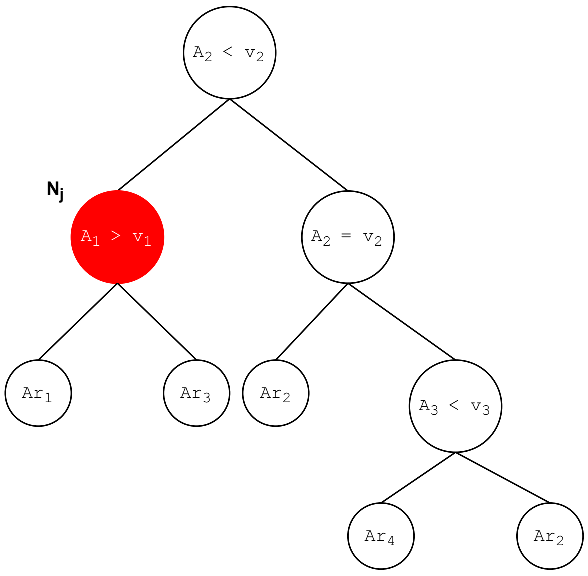


Figure 6.2: An illustration of an model tree perturb-worst-nonterminal-node operator

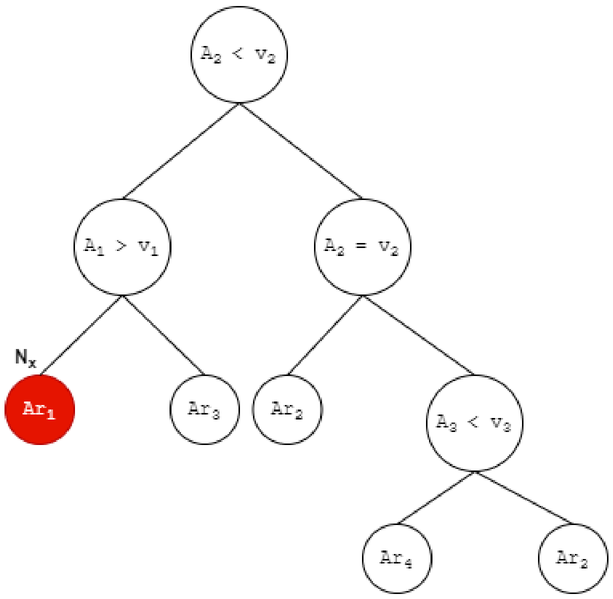


Figure 6.3: An illustration of a model tree perturb-worst-terminal-node operator

Thus, the nonlinear regression technique induces nonlinear models which approximate mapping between inputs and the target variable. The nonlinear regression technique consists of two steps: nonstationary data clustering using KCDCDynDE discussed in Section 2.5.2, and nonlinear model induction using the dynamic PSO-based nonlinear regression approach (DynPSO), discussed in Chapter 5.

A KCDCDynDE clusters data in a nonstationary environment, therefore, it is adopted in this work to perform clustering on nonstationary data [135]. 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. Therefore, whenever a change in an environment is evident, KCDCDynDE automatically determines the optimal number of clusters. For each generated cluster, a model is fitted using DynPSO.

6.5 Environment Change Detection

The easiest way to detect an environmental change in a nonstationary environment is to keep track of the best fitness found [72]. Considering an elitist individual in the current GP 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 changes the underlying target distribution is encountered. The parameter, n_s, is a user-defined parameter. Algorithm 13 summarizes the environmental change update process.

Considering Algorithm 13, if an environmental change is detected, in line 2, KCDC-DynDE dynamically clusters the data in line 3. For the dataset that changes in line 4, the DynPSO adapts nonlinear models or induces new models to create an updated GP terminal set in line 5 - 6.

Algorithm 13 Environmental Change Update Process

```
1: BEGIN
2: if an environmental change is detected then
3:     Perform data clustering using KCDCDynDE to obtain $k$ clusters
4:     if the clusters change then
5:         Adapt the nonlinear model or induce new models using DynPSO
6:         Insert new models into a terminal set
7:     end if
8: end if
9: END
```

The GP population converge towards a promising solution. As a result, the population lacks diversity necessary to locate a new optimum when a new data generating process is encountered. Therefore, GPANDA implements a culling technique to enhance population diversity. Thus, a portion of the worst population (individuals with higher RMSE) are replaced by new individuals that are evolved using an updated GP terminal set.

6.6 Parameter Tuning

The optimal values for each parameter of each algorithm implemented are obtained using the F-race algorithm, only a single run is performed [174]. The parameter optimization is done across domains, discussed in Section 5.5. The optimized parameter values listed in Table 6.1 are used where culling, tournament, Brownian individual implemented in KCDCDynDE and elitist are expressed as the percentage of the *PopSize*.

For the DyFor GP experiments, the sizes of the two sliding windows are initialized to 20% and 80% of the sliding window size (w_s), the minimum window size is set to 5% of (w_s) and a minimum difference of two sliding windows is set to 2% of (w_s) (discussed in Section 3.3.1). The quantum size is expressed as a percentage of *swarmSize*.

The function set, $\{+, -, \div, \times, \sqrt{}, \sin, \cos, \exp, \ln\}$ is used in both DynGP and DyFor GP.

6.7 Summary

In this chapter, GPANDA designed to evolve a predictive model in nonstationary environments was presented. Three components that make up GPANDA: dynamic clustering; nonlinear model induction approach and model tree induction were discussed. An environmental change detection technique implemented in GPANDA to adapt to nonstationary data was also discussed.

In the next chapter, results and discussion for the experiments done on the proposed approaches are presented.

Table 6.1: GPANDA Parameter Values

Algorithm	Parameter	Bounds	Value
GP	p_c	[0,1]	0.8
	p_m	[0,1]	0.1
	p_r	[0,1]	0.1
	PopSize	[10,500]	100
	Elitist	[0,100]	5%
	Culling	[0,100]	50%
	Tournament	[0,100]	20%
	π_ω	[0,500]	50
	n_s	[0,100]	10%
QPSO	$c_1 = c_2$	[0,3]	1.496
	ω	[0,1]	0.730
	swarmSize	[0,50]	30
	r_{cloud}	[0,5]	2
	iterations	[0,∞]	100
KCDCDynDE	PopSize	[10,500]	80
	p_c	[0,1]	0.5
	Scale parameter	[0,1]	0.5
	r_{excl}	[0,5]	3.5
	R_{conv}	[0,5]	2.0
	Brownian	[0,100]	10%
	$[\beta_{min}, \beta_{max}]$	[0,1]	[0.1, 0.9]
	iterations	[0,∞]	80

Chapter 7

Results and Discussion

7.1 Introduction

Experimental evaluations of the DynPSO discussed in Chapter 5 and GPANDA discussed in Chapter 6 were carried out. The results obtained are presented and discussed in this chapter. Also, a comparative analysis was carried out on the obtained results with the selected state-of-the-art techniques. In the subsequent tables, bold labelling indicates the best performing approach.

The objectives outlined in Section 1.2 are listed below and the obtained results to meet the objectives are presented in this chapter.

(a) To investigate the effectiveness of hybridizing dynamic PSO with a regression technique, either least-squares approximation or autoregressive, (DynPSO) to induce optimal nonlinear regression models in nonstationary environments;

(b) To investigate the effectiveness of hybridizing a GP with a dynamic clustering algorithm and nonlinear model induction technique (GPANDA) to perform regression on nonstationary data with concept drift occurring;

(c) To compare the performance of DynPSO to optimize the induced model in a nonstationary environment, to dynamic PSO algorithms, namely multi-swarm, reinitialized, and charged PSOs;

(d) To compare the performance of GPANDA in terms of predictive accuracy and computational time to the best performing dynamic GP algorithms and the state-of-the-art techniques on nonstationary datasets that exhibit different characteristics of concept drift such as progressive, recurrent, abrupt and random changes on varying temporal and spatial severities.

In this chapter, Section 7.2 presents the results of applying the DynPSO defined in *Objective (a)* to experiments outlined in Section 4.6.1 using the dataset presented in Section 4.4 and a comparative analysis defined in *Objective (c)*. Section 7.3 presents the results of applying GPANDA proposed in *Objective (b)* to experiments defined in Section 4.6.2 using the dataset presented in Section 4.4 and a comparative analysis defined in *Objective (d)*.

7.2 Dynamic PSO-based Nonlinear Regression Approach

In this section, the obtained mean squared errors (E_{MS}) and adjusted coefficient of determination (R_a^2) on training and generalization for different severities and frequencies are reported. To ascertain the significance of the results obtained by DynPSO and the best performing dynamic PSOs, statistical analysis was used. The *p-values* corresponding to the comparison of the algorithms on $E_{MS}T$ (training) and $E_{MS}G$ (generalization) for 30 independent runs are reported in Appendix 1 where $p \leq 0.0001$ was recorded as 0.0001 for convenience. Table 7.1 presents four versions of DynPSO (QR-PSOs/NARX-PSO) and three state-of-the-art dynamic PSO algorithms (nonQR-PSOs)were implemented in this experiment.

Table 7.1: Summary of Dynamic PSO Algorithms

nonQR-PSOs	**QR-PSOs/NARX-PSO**
charged PSO (QPSO)	QR-based charged PSO (QR-QPSO)
multi-swarm PSO (mPSO)	QR-based multi-swarm PSO (QR-mPSO)
re-initialized PSO (rePSO)	QR-based re-initialized (QR-rePSO)
	NARX-based charged PSO (NARX-QPSO)

This section is structured as follows: Section 7.2.1 presents the results and the comparative analysis for the Progressive-Bennett5A dataset. Section 7.2.2 presents the results and the comparative analysis for the NSW Electricity pricing dataset. Section 7.2.3 presents an evaluation of the hybrid regression approaches proposed in this work. Section 7.2.4 presents the results and the comparative analysis of DynPSO to the state-of-the-art techniques whereas Section 7.2.5 provides the discussion of the presented results.

7.2.1 Results for Progressive-Bennett5A Dataset

Table 7.2 presents the average and standard deviation for R_a^2 and (E_{MS}) on training and generalization (testing) for each algorithm on the Progressive-Bennett5A dataset.

Table 7.2: Averages and Standard deviation for R_a^2 and E_{MS} for Progressive-Bennett5A Dataset

Algorithm	**Training**		**Testing**	
	R_a^2	E_{MS}	R_a^2	E_{MS}
QR-QPSO	$\mathbf{0.7075 \pm 0.0731}$	0.3196 ± 0.0142	$\mathbf{0.8236 \pm 0.0598}$	$\mathbf{0.2183 \pm 0.0621}$
QPSO	0.4227 ± 0.1782	$\mathbf{0.3103 \pm 0.0097}$	0.7598 ± 0.0699	0.4504 ± 0.0954
QR-rePSO	0.6169 ± 0.0901	0.3113 ± 0.0099	0.8059 ± 0.0762	0.4043 ± 0.1150
rePSO	0.0769 ± 0.0110	0.3507 ± 0.0223	0.4237 ± 0.0452	0.4992 ± 0.0774
QR-mPSO	0.6407 ± 0.0583	0.3141 ± 0.0132	0.8121 ± 0.0649	0.4907 ± 0.0849
mPSO	0.0912 ± 0.0973	0.3328 ± 0.0192	0.4068 ± 0.0527	0.4271 ± 0.0326

The results presented in Table 7.2 shows that QR-QPSO obtains the best R_a^2 on both training and generalization whereas mPSO obtains the worst performance for generalization and rePSO on training. The QPSO obtains the best E_{MS} on training whereas QR-QPSO for generalization.

As reported in Table 7.2, QR-PSOs outperforms nonQR-PSOs for both R_a^2 and E_{MS}, except for QPSO for R_a^2. This performance improvement can be attributed to the capability of the proposed technique to track and adapt the nonlinear regression model as the environment changes. Also, the value of R_a^2 above 0.7 suggests structurally optimal models generated by QR-PSOs. Considering QR-PSOs only, QR-QPSO exhibits superior performance whereas QR-rePSO exhibits the worst performance.

The averages for R_a^2 and E_{MS} for generalization per frequency and severity for Progressive-Bennett5A dataset are graphically illustrated in Figure 7.1. The frequencies and severities used in this experiment are defined in Section 4.4. As illustrated in Figure 7.1, the performance of all algorithms improved as the frequency increased for both R_a^2 and E_{MS}. However, QR-PSOs exhibit superior performance evident with higher R_a^2 and lower E_{MS}. Also, QR-QPSO outperforms all other algorithms for both R_a^2 and E_{MS} in all frequencies and severities.

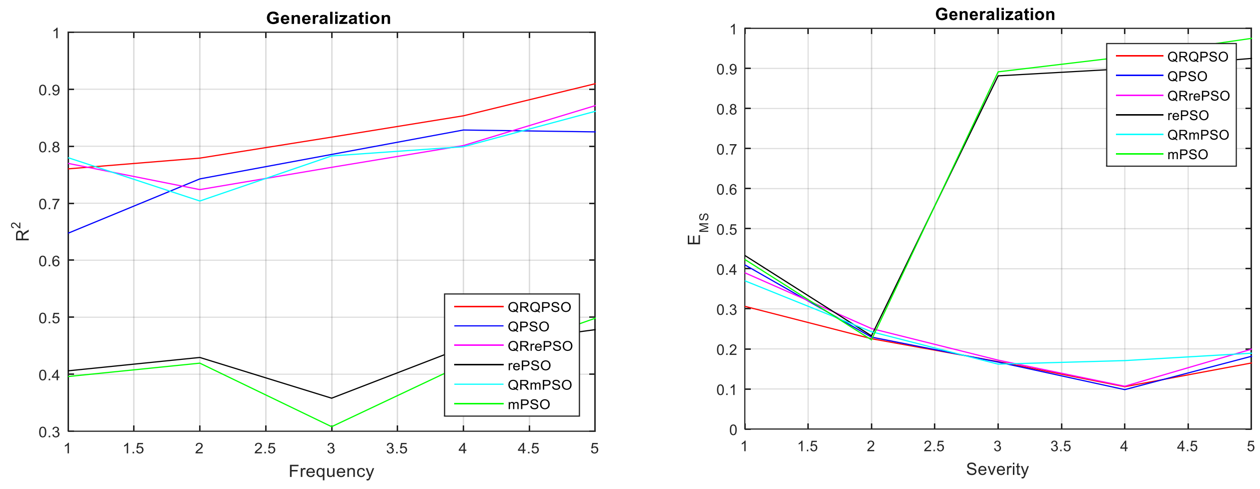


Figure 7.1: R_a^2 and E_{MS} for Generalization per Frequency and Severity on Progressive-Bennett5A

The average ranks (discussed in Section 4.5.1) obtains on algorithms for R_a^2 and E_{MS} for both training and generalization for the given frequencies and severities are illustrated in Figure 7.2.

As illustrated in Figure 7.2, rePSO and mPSO exhibits the worst performance for both R_a^2 and E_{MS}. For both training and generalization (testing), QR-QPSO outperforms all other algorithms for both R_a^2 and E_{MS}. Generally, as the frequency increases, the performance of nonQR-PSOs (discussed in Section 5.2) deteriorate for both R_a^2 and E_{MS} especially for QPSO.

To ascertain the significance of the obtained results for Progressive-Bennett5A dataset, statistical analysis was used and the obtained p-value ≤ 0.05 indicates that the performance was statistically significant.

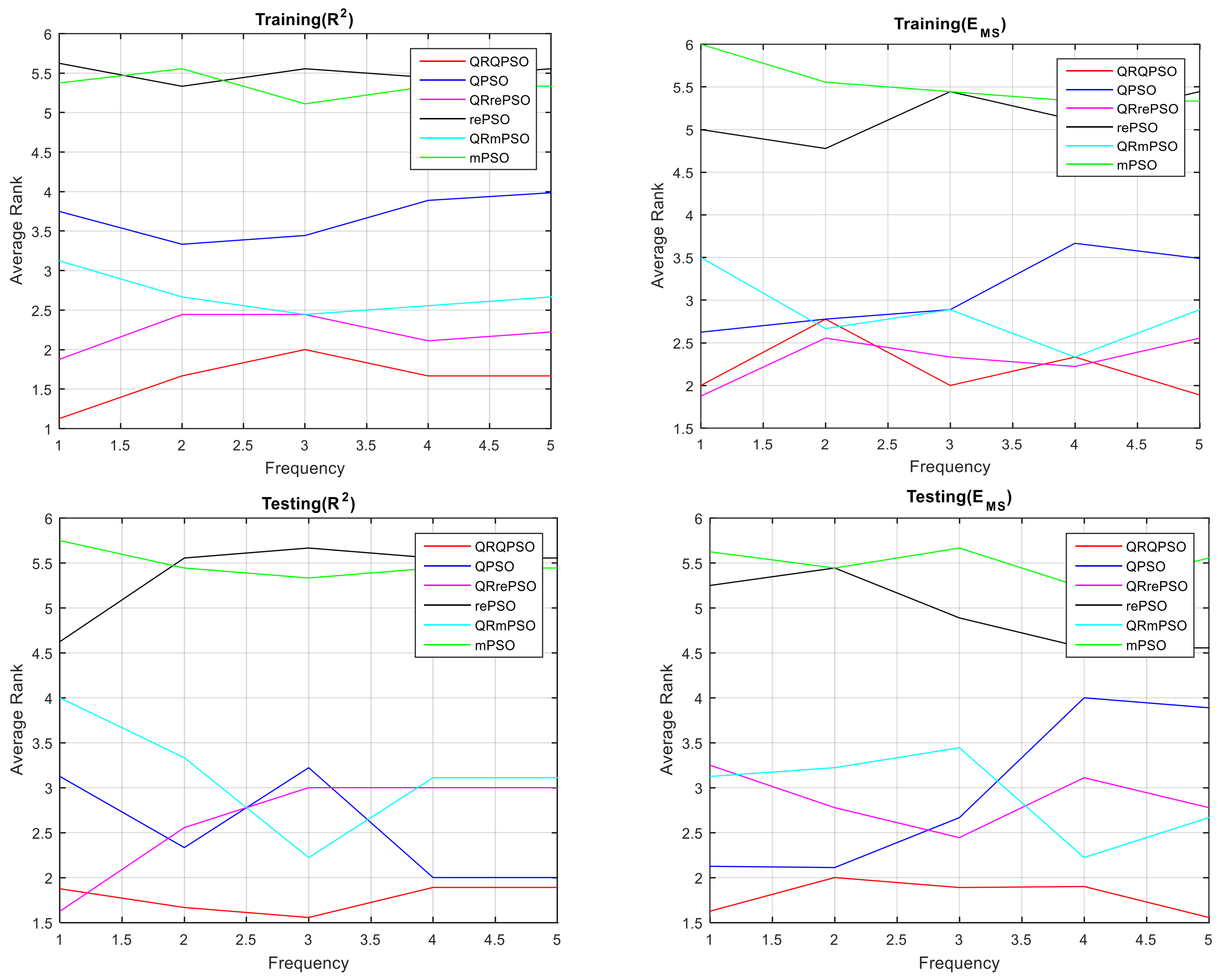


Figure 7.2: Ranks of R_a^2 and E_{MS} per frequency on the training and Generalization for Bennett5A Dataset

7.2.2 Results for NSW Electricity Pricing Dataset

Table 7.3 presents the average and standard deviation (at the 95% confidence interval) for R_a^2 and E_{MS} on training and generalization for each algorithm for NSW Electricity pricing dataset.

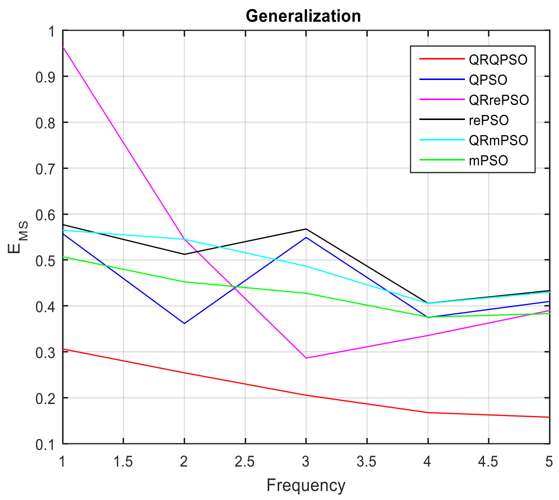
As observed in Table 7.2, similar traits are also observed in Table 7.3: the QR-QPSO obtains the best R_a^2 for both training and generalization whereas rePSO obtains the worst performance for both training and generalization. Also, the value of R_a^2 above 0.75 suggests structurally optimal models generated by QR-PSOs. Considering QR-PSOs, the QR-QPSO exhibits superior performance and the QR-rePSO exhibits the worst performance.

Table 7.3: Averages and Standard deviation for R_a^2 and E_{MS} for NSW Electricity Pricing

Algorithm	**Training**		**Testing**	
	R_a^2	E_{MS}	R_a^2	E_{MS}
QR-QPSO	$\mathbf{0.8874 \pm 0.1794}$	0.0001 ± 0.0004	$\mathbf{0.8097 \pm 0.2979}$	0.0011 ± 0.0005
QPSO	0.8646 ± 0.1630	0.0001 ± 0.0004	0.7316 ± 0.3697	$\mathbf{0.0010 \pm 0.0954}$
QR-rePSO	0.8671 ± 0.1496	0.0001 ± 0.0004	0.7744 ± 0.3384	0.0032 ± 0.0011
rePSO	0.6670 ± 0.1931	0.0003 ± 0.0012	0.7259 ± 0.3234	0.0043 ± 0.0007
QR-mPSO	0.8694 ± 0.1712	$\mathbf{0.0001 \pm 0.0003}$	0.7837 ± 0.2649	0.0028 ± 0.0008
mPSO	0.7108 ± 0.1253	0.0002 ± 0.0009	0.7305 ± 0.3234	0.0042 ± 0.0009

The averages for R_a^2 and E_{MS} for generalization per frequency and severity on NSW Electricity dataset are graphically illustrated in Figure 7.3. As illustrated in Figure 7.3, the performance of all algorithms improve as the frequency (discussed in Section 4.6.1) increases. The QR-QPSO outperforms all other algorithms for both R_a^2 and E_{MS} in all frequencies and severities. Also, QR-PSOs exhibit superior performance outperforming nonQR-PSOs.

The average ranks obtained on algorithms for R_a^2 and E_{MS} for both training and generalization for the given frequencies and severities are illustrated in Figure 7.4. As illustrated in Figure 7.4, QR-QPSO also outperforms all other algorithms whereas rePSO and mPSO exhibit the worst performance.



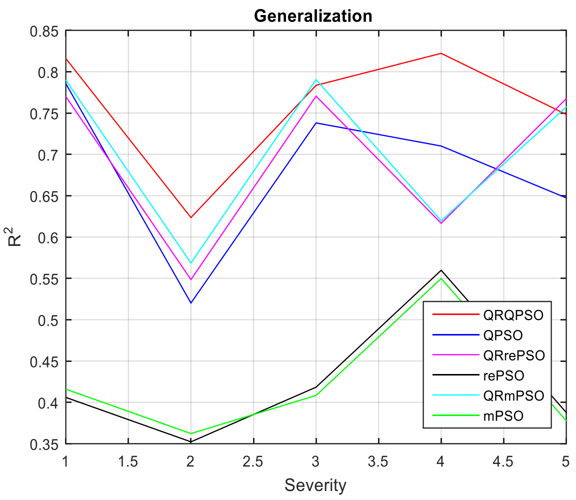


Figure 7.3: R_a^2 and E_{MS} for Generalization per Frequency and Severity for NSW Electricity

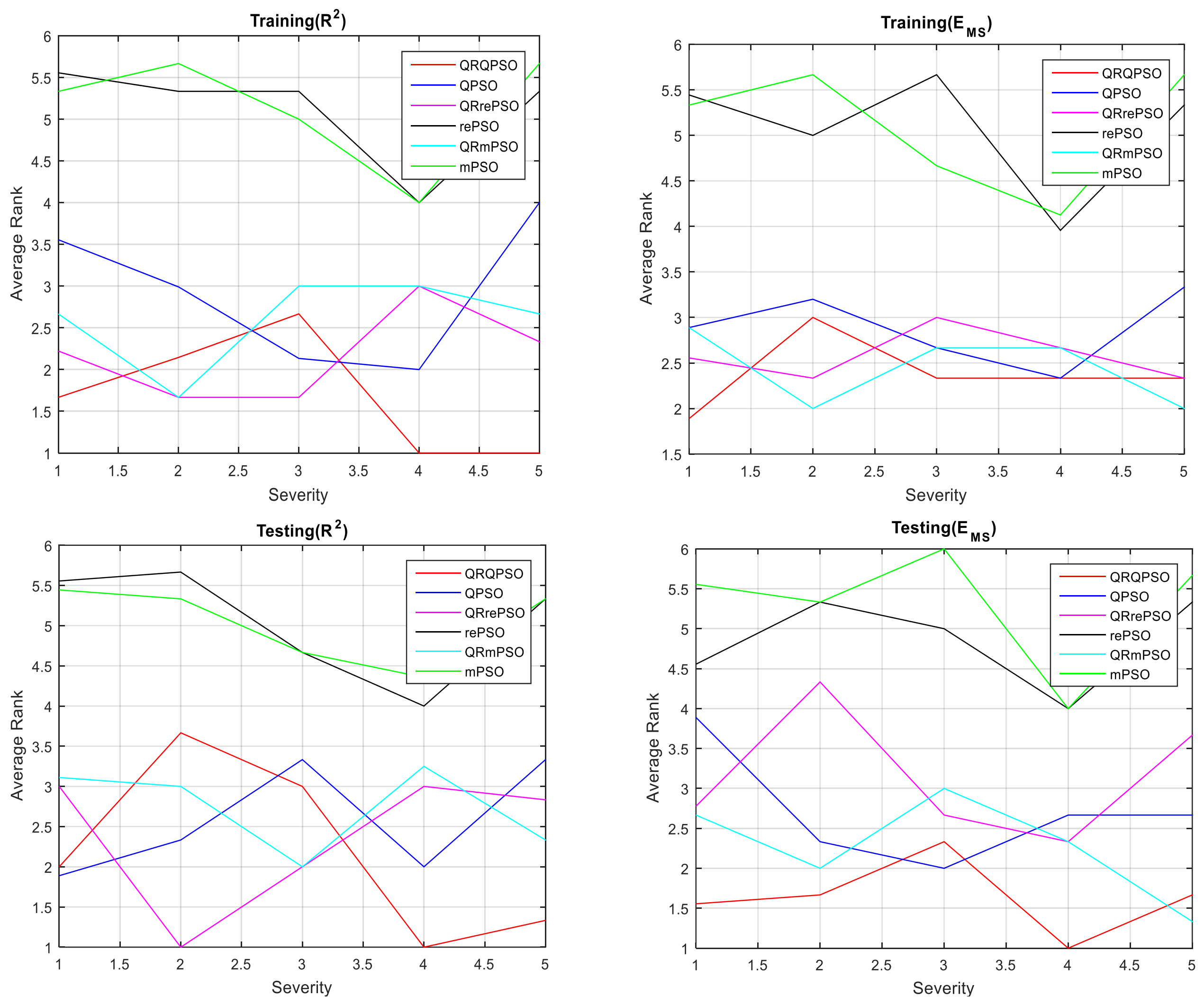


Figure 7.4: Ranks of R_a^2 and E_{MS} per severity on training and testing for NSW Electricity

The performance of QR-mPSO improves as the severity increases outperforming QR-rePSO for generalization for both R_a^2 and E_{MS}. This performance improvement for QR-mPSO suggests the improved adaptive traits of QR-mPSO under severe changing environment. Also, QR-PSOs outperform nonQR-PSOs for both R_a^2 and E_{MS}. As already explained, the outstanding performance can be attributed to the capability of the proposed technique to track and adapt the nonlinear regression model as the environment changes.

To ascertain the significance of the results obtained by DynPSO and dynamic PSOs for NSW Electricity pricing dataset, statistical analysis was used and the obtained p-value ≤ 0.05 indicates that the performance was statistically significant.

7.2.3 An Evaluation of the Hybrid Regression Approaches

The section aims to evaluate the effectiveness of the regression models, namely QR decomposition and autoregressive, NARX, to improve the predictive performance of DynPSO to forecast nonstationary time series.

In this experiment, five algorithms are executed namely, QR-QPSO, NARX-QPSO, QR decomposition, NARX, and QPSO. For the hybrid models and QPSO, the data points in the testing dataset are presented incrementally (one at a time) to enable the techniques to cope with concept shifts happening in the dataset. However, the models for NARX and QR decomposition (QR) were fixed during the generalization.

Table 7.4 presents the obtained results for each algorithm for AEMO dataset for electric load forecasting described in Section 4.4. There exist statistically significant differences between the performance of each algorithm as suggested by the obtained p-values. The results presented in Table 7.4 shows that NARX-QPSO obtains the best MAPE and RMSE whereas QR decomposition obtains the worst performance. As presented in Table 7.4, hybrid models: NARX-QPSO and QR-QPSO significantly outperforms classical techniques. The outstanding performance can be hypothesized to be the ability to adapt the forecasting model as the environment changes due to concept shifts.

To ascertain the significance of the results obtained by DynPSO and individual algorithms for AEMO dataset, statistical analysis was used and the obtained p-value ≤ 0.05 indicates that they exist statistically significant differences between the performance of each algorithm as suggested by the obtained p-values.

Figure 7.5 is a graphical illustration of the average computational time of the models/approaches under study for electric load forecasting for the year 2015 for the five states. The NARX obtains the least average computational time whereas the QPSO obtains the highest average computational time. Consequently, NARX-QPSO obtains a reasonable computational time due to the best average computational time of NARX.

Table 7.4: Prediction Results for AEMO dataset

Dataset	Year	Metric	QR	QPSO	NARX	QR-QPSO	NARX-QPSO
VIC	2015	MAPE(%)	12.0497	8.9638	10.2595	8.1672	**7.3618**
		RMSE	724.5641	549.8639	637.5787	517.4586	**481.9153**
	2014	MAPE(%)	10.9645	7.4861	9.8606	7.1062	**6.4636**
		RMSE	610.8749	481.3562	602.7269	436.7005	**401.2668**
	2013	MAPE(%)	10.5291	7.1846	10.8892	6.7139	**6.5673**
		RMSE	613.1647	517.8379	593.7182	465.1925	**439.7815**
SA	2015	MAPE(%)	14.9372	12.5845	13.0364	12.2528	**11.0629**
		RMSE	373.8404	221.7641	239.3121	219.1468	**210.6389**
	2014	MAPE(%)	12.9952	10.5833	11.9697	9.9153	**9.1486**
		RMSE	171.8473	163.3109	176.6305	159.8215	**146.8107**
	2013	MAPE(%)	11.0995	9.8574	9.9451	9.6637	**9.5538**
		RMSE	189.4768	167.1382	171.4686	157.6888	**157.3617**
TAS	2015	MAPE(%)	4.7624	4.2093	4.2693	**4.1425**	4.1805
		RMSE	79.7361	68.6835	72.9453	60.9459	**59.7324**
	2014	MAPE(%)	6.0527	5.7896	6.7442	4.9763	**4.6135**
		RMSE	82.3723	76.8709	81.9906	61.2159	**60.9814**
	2013	MAPE(%)	6.2641	5.21538	5.9443	4.8315	**4.3618**
		RMSE	88.5911	70.5837	85.8395	66.8192	**63.0092**
NSW	2015	MAPE(%)	8.6217	6.1831	6.7942	5.8473	**5.1674**
		RMSE	859.5385	609.3729	734.2208	558.7924	**557.6981**
	2014	MAPE(%)	6.3401	5.2967	6.3758	4.5683	**4.8183**
		RMSE	721.3416	544.8648	625.6111	512.0017	**501.0148**
	2013	MAPE(%)	6.5826	5.5714	6.2171	5.2661	**5.1826**
		RMSE	701.9968	569.2851	639.8593	523.6393	**536.7814**
QLD	2015	MAPE(%)	4.7964	3.7286	5.2568	3.2782	**3.2485**
		RMSE	396.8471	334.2874	373.8961	317.4186	**298.1975**
	2014	MAPE(%)	5.3157	4.2739	5.7194	4.0694	**3.3047**
		RMSE	472.9735	348.3916	383.6083	**319.8156**	330.4635
	2013	MAPE(%)	4.8893	3.5920	4.3709	3.3121	**3.2967**
		RMSE	410.5834	307.5394	354.6884	296.8568	**266.454**

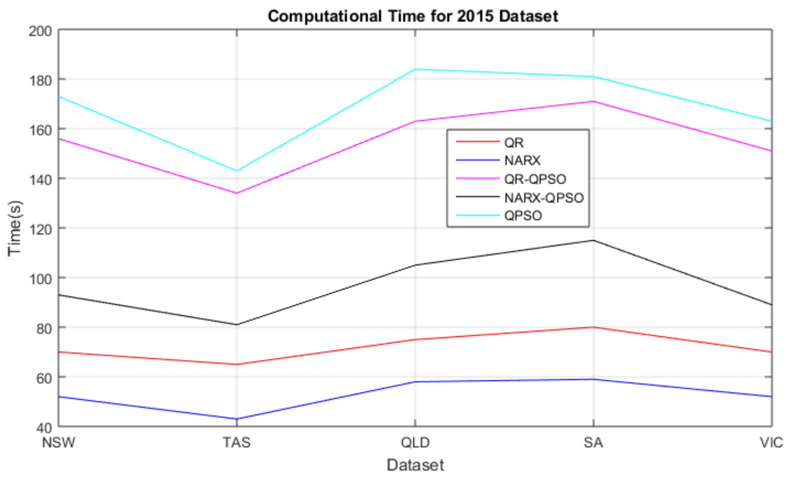


Figure 7.5: Average Computational Time for Electric Load Forecasting

7.2.4 Comparison to the State-of-the-art Techniques

The best performing prediction techniques for nonstationary time series: conventional Support Vector Regression (SVR) [175][176] and standard Random Vector Functional Link network (RVFL) with direct input-output connections that retain biases [177][178], are compared to DynPSO for the following datasets: Progressive-Bennett5A and NSW Electricity pricing discussed in Section 4.4.

Table 7.5 presents the E_{MS} for the DynPSO to the state-of-the-art techniques: SVR and RVFL for generalization for the datasets: Progressive-Bennett5A and NSW Electricity pricing.

Table 7.5: Average E_{MS} for Progressive-Bennett5A and NSW Electricity Pricing Datasets

Dataset	QR-QPSO	QR-rePSO	QR-mPSO	SVR	RVFL
NSW Electricity	**0.0011**	0.0032	0.0028	0.0016	0.0056
Bennett5A	**0.2183**	0.4043	0.4907	0.6197	0.9005

As observed in the results presented in Table 7.6, the QR-PSOs outperforms both SVR and RVFL for the Progressive-Bennett5A datasets. However, SVR outperforms both QR-rePSO and QR-mPSO on Electricity dataset. The QR-QPSO outperforms all algorithms for both datasets.

To ascertain the significance of the results obtained by DynPSO, RVFL and SVR for NSW Electricity datasets, statistical analysis was used and the obtained p-value ≤ 0.05 indicates that they exist statistically significant differences between the performance of each algorithm as suggested by the obtains p-values.

In [179], the state-of-the-art techniques were applied to AEMO electric load datasets for the New South Wales for the months: January, April, July and October of the year 2015 to assess the effectiveness of the models/approaches when seasonality was taken into consideration. In this work, the same simulations are applied to the DynPSO to evaluate the effect of different seasons. Table 7.6 presents the obtained forecasting results of the state-of-the-art techniques and the DynPSO for the AEMO, NSW 2015 electric load dataset.

The NARX-QPSO significantly outperforms all other techniques except DWT-EMD-RVFL whereas QR-QPSO outperformed Persistence, RVFL and GLMLF-B. As already explained, the performance improvement could have been attributed to the capability of the proposed technique to optimize the induce forecasting model and adapt the forecasting model as the environment changes due to concept shifts. Considering the factors of different seasons, the performance of the DynPSO is relatively stable, as suggested by the results reported in Table 7.6.

Table 7.6: Prediction results for NSW, 2015

Dataset	Metric	GLMLF-B	Persistence	RVFL	DWT-EMD-RVFL	QR-QPSO	NARX-QPSO
Jan	MAPE (%)	5.60	7.39	3.87	**1.86**	1.99	1.93
	RMSE	612.30	842.73	428.908	193.80	197.52	**192.21**
April	MAPE (%)	5.26	6.80	3.94	2.03	2.01	**1.99**
	RMSE	525.15	769.61	425.23	212.70	207.72	**198.62**
Jul	MAPE (%)	6.14	9.83	5.09	**2.96**	3.21	3.16
	RMSE	614.71	989.37	493.06	**296.74**	312.85	309.63
Oct	MAPE (%)	9.40	14.89	8.86	**5.93**	6.07	5.99
	RMSE	1091.05	1620.51	1004.39	659.41	613.28	**599.37**

The SVR, SVRARIMA and ARIMASVR algorithms were applied to the California electricity market to perform next-week prices (short-term electricity prices) forecasting in the California electricity market [180]. In this work, the same simulations are applied to the DynPSO to perform next-week prices forecasting [180].

Table 7.7 presents the forecasting results of the state-of-the-art techniques and the DynPSO for the California electricity market dataset. The NARX-QPSO exhibits outstanding performance outperforming all algorithms under consideration.

Table 7.7: Prediction Results for California Electricity Market, 2000

Metric		SVR	SVRARIMA	ARIMASVR	QR-QPSO	NARX-QPSO
MAPE (%)	1st Week	758.69	348.81	2.04e03	331.86	**296.73**
RMSE	1st Week	1.44	0.75	3.98	0.85	**0.63**
MAPE (%)	2nd Week	969.37	514.29	597.78	498.73	**482.81**
RMSE	2nd Week	2.05	1.07	1.26	1.05	**0.98**

7.2.5 Discussion

The performance of nonQR-PSOs deteriorate as the severity increases whereas QR-PSOs exhibit outstanding performance as the severity increases.

The obtained results suggest the capability of the proposed DynPSO to track and adapt the induced model as the environment changes. As such, the DynPSO outperforms both the dynamic PSOs and the state-of-the-art techniques, SVR and RVFL, on the given datasets. The hybridization indeed decreased the performance deterioration of the induced model that resulted from the environmental changes. The obtained values of R_a^2 suggests that the DynPSO induced structurally optimal nonlinear regression models.

The QR-QPSO induces optimal model structure as suggested by the obtained R_a^2 values and also, performed competitively to NARX-QPSO in nonstationary time series forecasting, therefore, is adapted in GPANDA as the dynamic QPSO (dynQPSO).

7.3 GPANDA Results and Discussion

In this section, a thorough scalability investigation was carried out on GPANDA discussed in Chapter 7 for the change in severities defined in each problem set. Experiments were carried out to find out if and how the change in severity affects the performance of GPANDA. The obtained results for each problem set are presented.

Experiments were also carried out on the problem sets using DynGP and DyFor GP (discussed in Section 3.4). The results obtained from GPANDA are compared to the results from DynGP and DyFor GP to determine the effectiveness of the proposed approach to the techniques already in existence. The *p-values* corresponding to the comparison of the algorithms on $E_{MS}T$ (training) and $E_{MS}G$ (generalization) for 30 independent runs, for scenarios that only included value of $p > 0.05$, are reported in Appendix 1. Each dataset (sliding window of analysis) was split into training and generalization subsets using a ratio 4:1 respectively.

7.3.1 Progressive Variation: Progressive-Bennett5B Dataset

The algorithms under study traverse the complete dataset for every scenario under consideration.

Scenario A - Abrupt Variation

The frequency of change is the same for all cases in Scenario A which is 25 iterations. Figure 7.6 - 7.7 are graphical illustrations as observed under Scenario $A_1 - A_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time. As illustrated in Figure 7.6 - 7.7, adaptation to change in the environment becomes less difficult as spatial severity increase from A_1 to A_5 and error peaks rise high after every change in the environment.

In Scenario A_5, as the window slides, new data points replaced all contents of the sliding window from the past data generating processes (discussed in Section 4.4), thereby presenting only one concept in a sliding window and consequently, data points from only a single data generating process. Therefore, the abrupt changes happening in Scenario A_5 make this scenario much easier to adapt.

As illustrated in Figure 7.6 - 7.7, DyFor GP outperforms DynGP in all cases in Scenario A except A_1 - training and A_5 - generalization. The superior performance of DyFor

GP suggests that the algorithm adapts the predictive model faster as the environment changes rapidly.

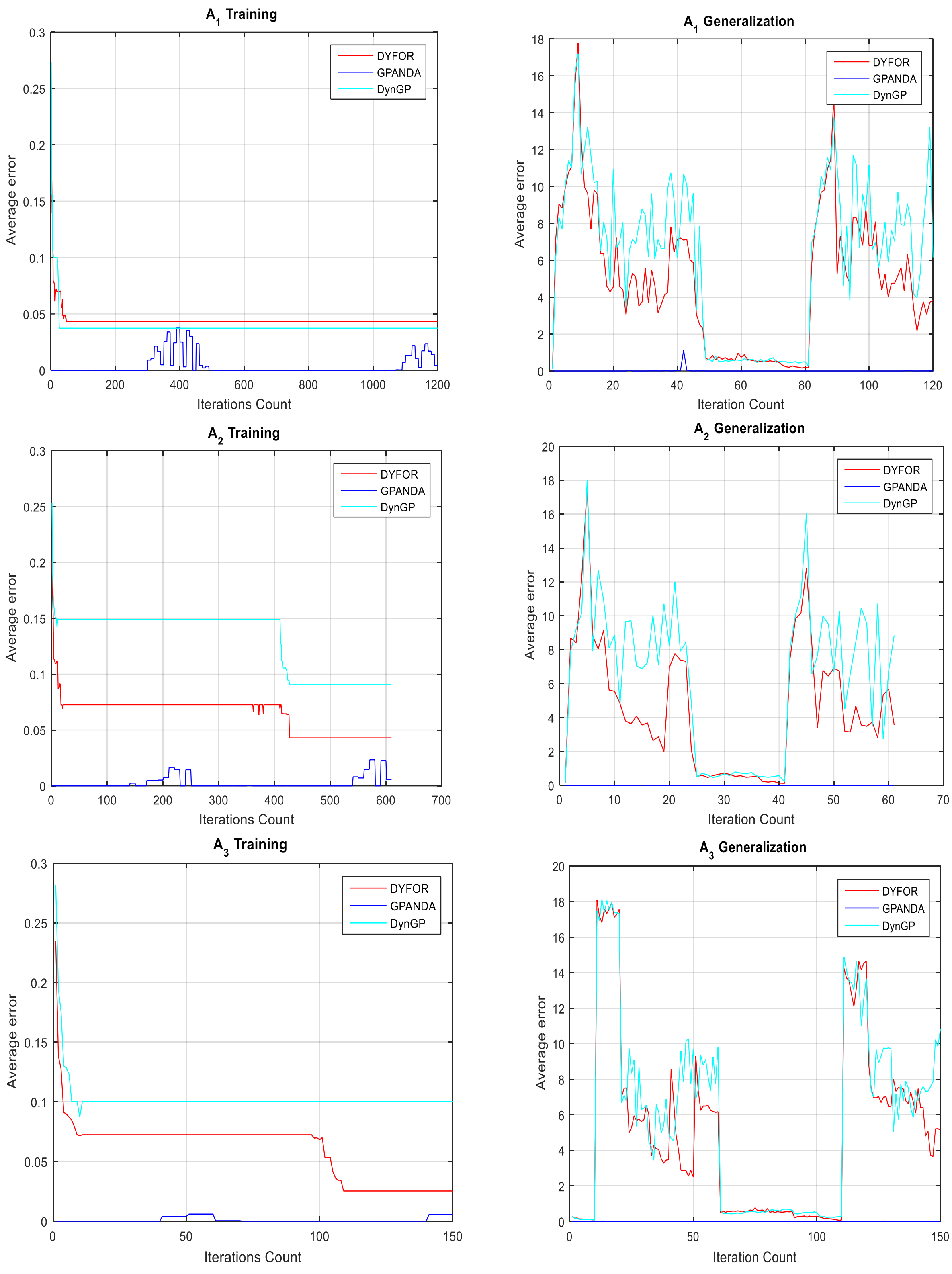


Figure 7.6: Training and Generalization E_{MS} for Bennett5B, Scenario $A_1 - A_3$

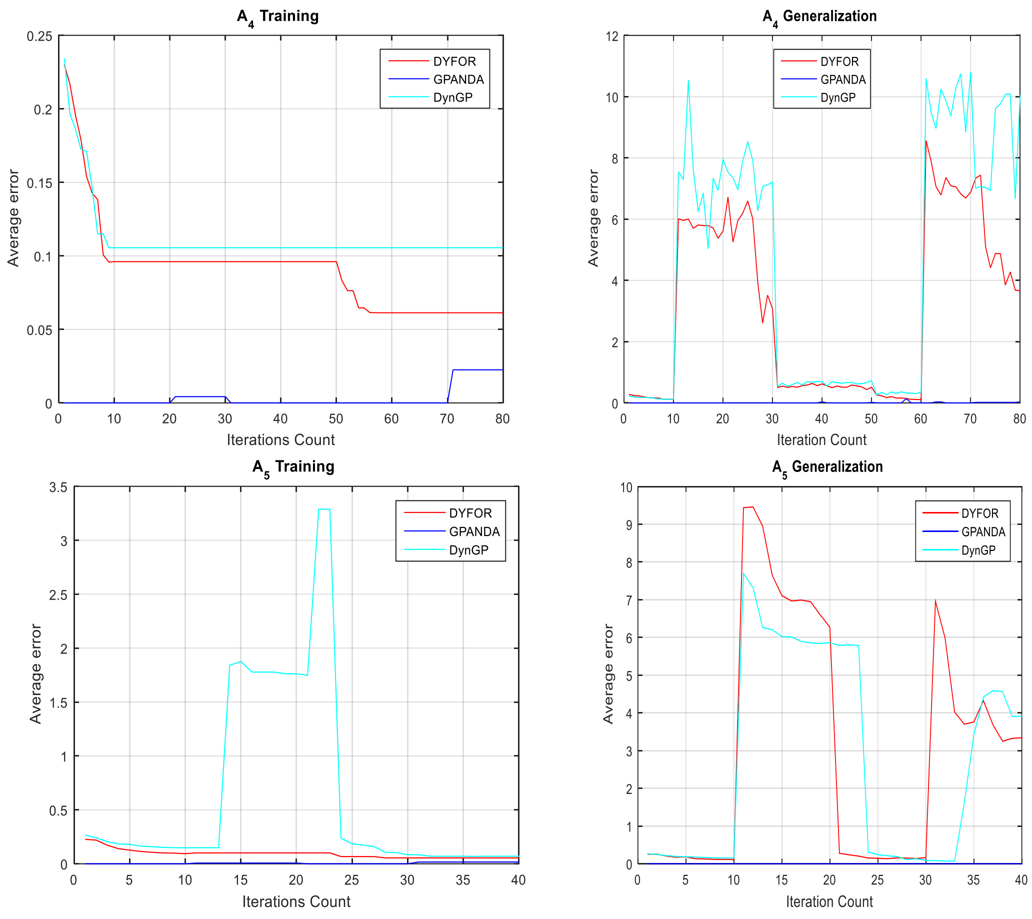


Figure 7.7: Training and Generalization E_{MS} for Progressive, Scenario $A_4 - A_5$

GPANDA detect changes happening in the environment and then adapts the model. As a result, GPANDA outperforms all other algorithms in all cases in Scenario A for both training and generalization. For generalization, GPANDA consistently maintains the obtained minimum compared to DynGP and DyFor GP after a change in the environment is detected.

Table 7.8 presents the obtained results for algorithms under study on the Progressive dataset, Scenario A. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.8:

- GPANDA outperforms all other algorithms in all cases in Scenario A for both training and generalization.

- DyFor GP outperforms DynGP for both training and generalization except A_1 (training) and A_5 (generalization).
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario A_5.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.8: Results of Predictive Performance for Progressive Environment Scenario A

Model	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0390	6.2034	0.1306	6.3398	0.1031	5.7803	0.1117	4.3563	0.6240	2.7544	0.2016	5.0868
GPANDA	**0.0061**	**0.0115**	**0.0025**	**2.65e-4**	**0.0011**	**8.50e-4**	**0.0034**	**0.0060**	**0.0059**	**2.23e-22**	**0.0038**	**0.0037**
DyFor GP	0.0445	4.8204	0.0646	4.4663	0.0601	4.9481	0.0911	3.0471	0.0922	3.0535	0.0705	4.0670

The algorithms under study are classified according to training and generalization performance. Table 7.9 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario A. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. Conversely, DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.9: Algorithm Ranking for Progressive Dataset Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	2	3	3	3	3	3	3	3	3	2	2.8	2.8
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	3	2	2	2	2	2	2	2	2	3	2.2	2.2

Discussion

The performance exhibited by DynGP can be attributed to the failure to adapt to a rapidly changing environment with concept drift occurring which suggests that DynGP requires a sufficient number of iterations to converge. The performance of DyFor GP can be attributed to the ability of the algorithm to optimize the sliding window of analysis by eliminating irrelevant data points. As such, DyFor GP outperforms DynGP.

GPANDA outperforms all other algorithms in all cases in Scenario A for generalization. The outstanding performance of GPANDA suggests the capability of the algorithm to adapt to a changing environment with concept drift occurring. The superior performance of GPANDA can be attributed to a high precision predictive model induced by the underlying dynQPSO.

The progressive changes happening in Scenario $A_1 - A_4$ (discussed in Section 4.4) increase the complexity of the predictive task since the sliding window at any given instance is made up of two different data generating processes. However, GPANDA detects an environment change in such cases to yield superior performance.

The performance of all models improved in Scenario A_5 due to abrupt change happening to data points in the sliding window of analysis since the sliding window discards all data points from past generating processes (discussed in Section 4.4), thus, generating a predictive model using the current prevailing data generating process only.

Scenario B - Quasi-Abrupt

Scenario B simulates less frequent changes having a sliding window shifting of 50 iterations instead of 25 iterations in Scenario A. Figure 7.8 - 7.9 are graphical illustrations as observed in Scenario B for the progression of $E_{MS}T$ and $E_{MS}G$ over time. As illustrated in Figure 7.8 - 7.9, less frequent changes has a positive effect on DynGP whereby the algorithm has sufficient iterations to converge to yield outstanding performance.

The performance of DynGP and DyFor GP is the same on training in Scenario B_1 and generalization in Scenario B_4. In Scenario $B_2 - B_4$, the performance of DynGP on training is reduced. Consequently, DynGP outperforms DyFor GP for both training and generalization in Scenario B_5.

However, DyFor GP outperforms DynGP for generalization in Scenario $B_1 - B_3$ where the training dataset has more data points generated by the previous data generating process compared to the current prevailing data generating process as illustrated in Figure 7.8 - 7.9. As such, it becomes difficult for DynGP to quickly adapt the model to fit the generalization dataset.

The significant error peaks illustrated in generalization plots in Figure 7.8 - 7.9 signify changes in the environment which result in reduced predictive performance in other algorithms.

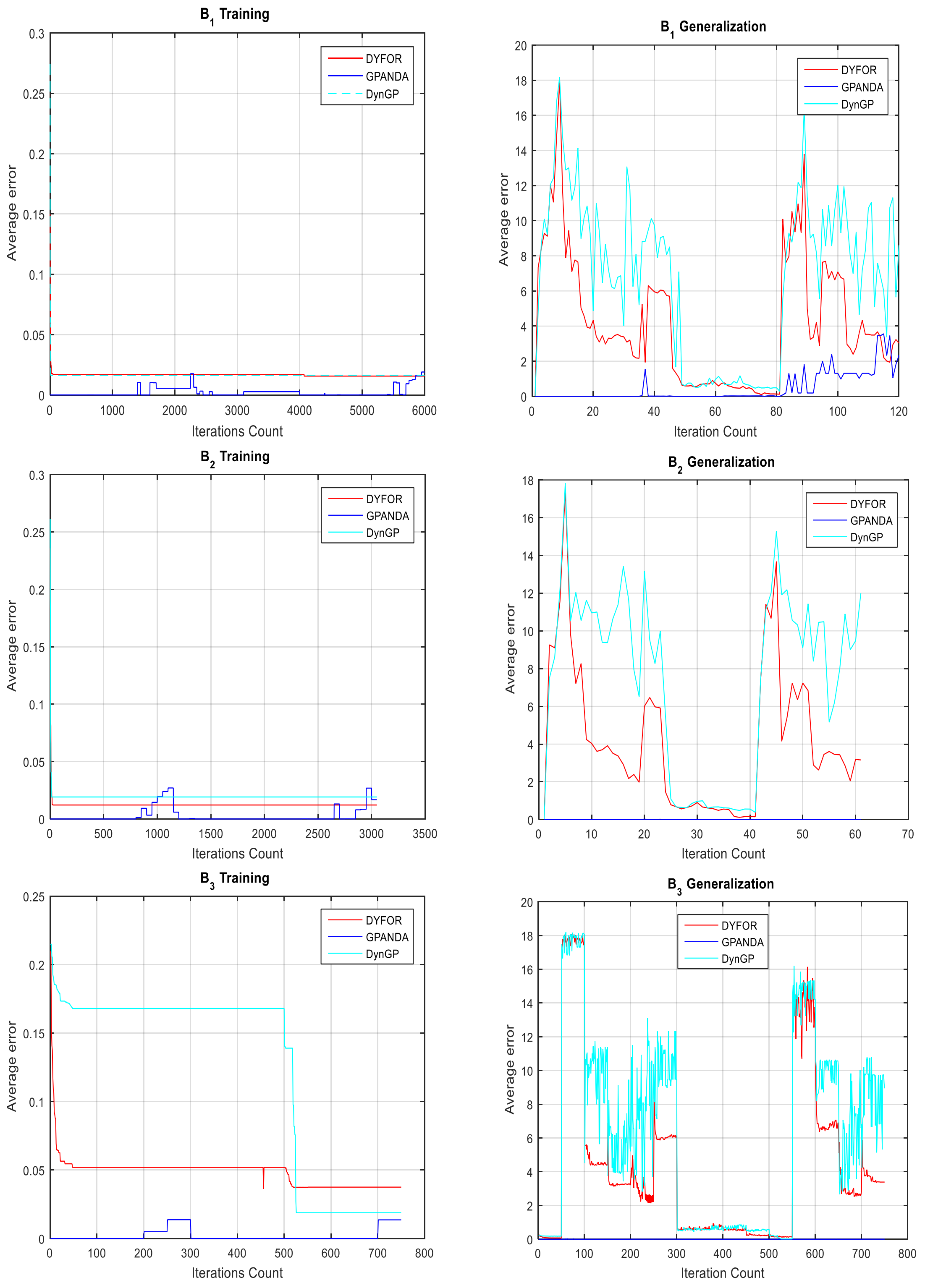


Figure 7.8: Training and Generalization E_{MS} for Progressive, Scenario $B_1 - B_3$

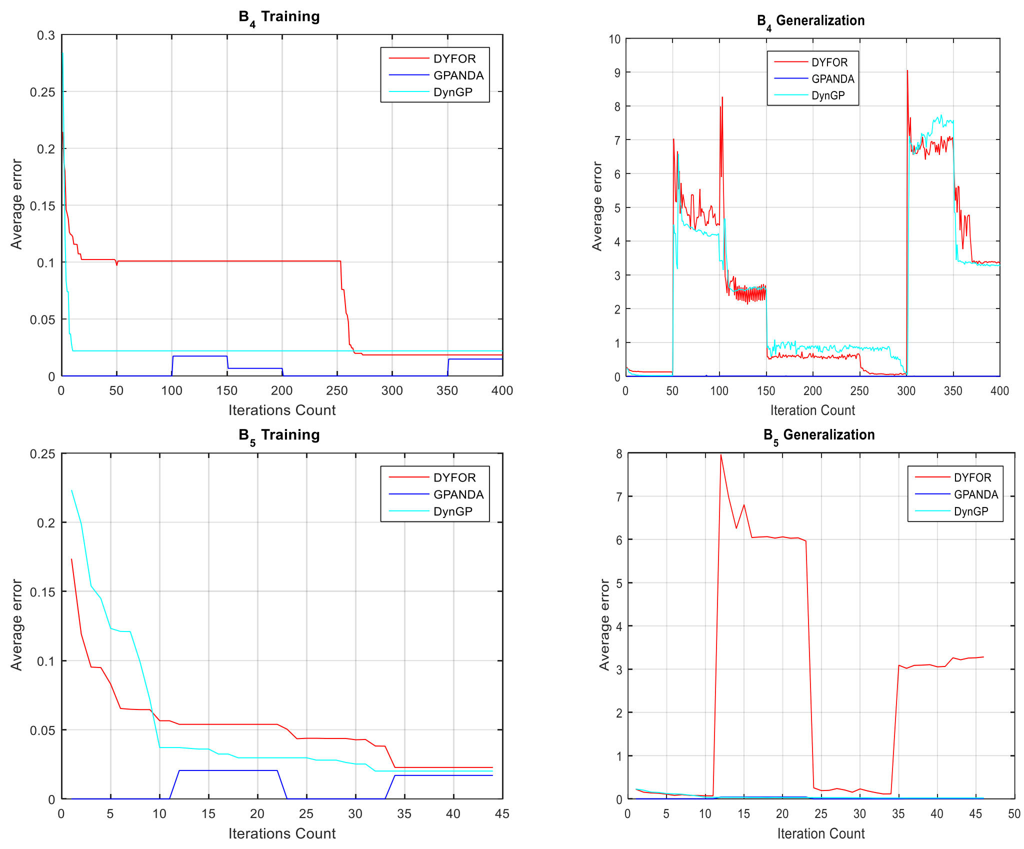


Figure 7.9: Training and Generalization E_{MS} for Progressive, Scenario $B_4 - B_5$

However, GPANDA tracks those gradual changes to yield outstanding performance.

Table 7.10 presents the obtained results for the Progressive dataset, Scenario B. To ascertain the significance of the results obtained for Scenario B, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1.

The following can be observed for the results presented in Table 7.11:

- GPANDA outperforms all other algorithms in all cases in Scenario B.
- DynGP outperforms DyFor GP in Scenario B_4 and for generalization in Scenario B_5.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario B_5.
- The result that GPANDA performs better than DyFor GP was found to be sta-

tistically significant at the 5% level of significance.

Table 7.10: Results of Predictive Performance for Progressive Environment Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0164	6.7128	0.0195	8.4172	0.1225	4.8776	0.0239	2.4075	0.0487	0.0480	0.0462	4.2926
GPANDA	**0.0103**	**0.4721**	**0.0029**	**0.0012**	**0.0022**	**9.51e-4**	**0.0049**	**0.0021**	**0.0094**	**0.0122**	**0.0059**	**0.0977**
DyFor GP	0.0166	4.1172	0.0125	4.1620	0.0488	4.2950	0.0731	2.5101	0.0448	2.5231	0.0391	3.5214

The algorithms under study are classified according to training and generalization performance. Table 7.11 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario B. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. Conversely, DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.11: Algorithm Ranking for Progressive Dataset Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	1.5	3	3	3	3	3	2	2	3	2	2.5	2.6
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	1.5	2	2	2	2	2	3	3	2	3	2.1	2.4

Discussion

The poor generalization performance of the model produced by DynGP can be attributed to conflicting decision boundaries present within the sliding window of analysis (discussed in Section 4.4), whereby the training data points consists of more from the past data generating processes which are no longer relevant to the current data generating process. The performance of DynGP greatly improved outperforming DyFor GP in cases where the analysis window has sufficient data points from the current data generating process.

As observed in Scenario A, GPANDA outperforms all other algorithms in all cases in Scenario B for both training and generalization. The superior performance of GPANDA suggests the capability of the algorithm to adapt to a changing environment with concept drift occurring.

Scenario C - Quasi-Progressive Variation

Scenario C simulates less frequent changes of 100 iterations before sliding an analysis window. Figure 7.10 - 7.11 are graphical illustrations as observed under Scenario $C_1 - C_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

The less frequent changes of 100 iterations availed sufficient iterations to all algorithms which promote convergence before an analysis window slides.

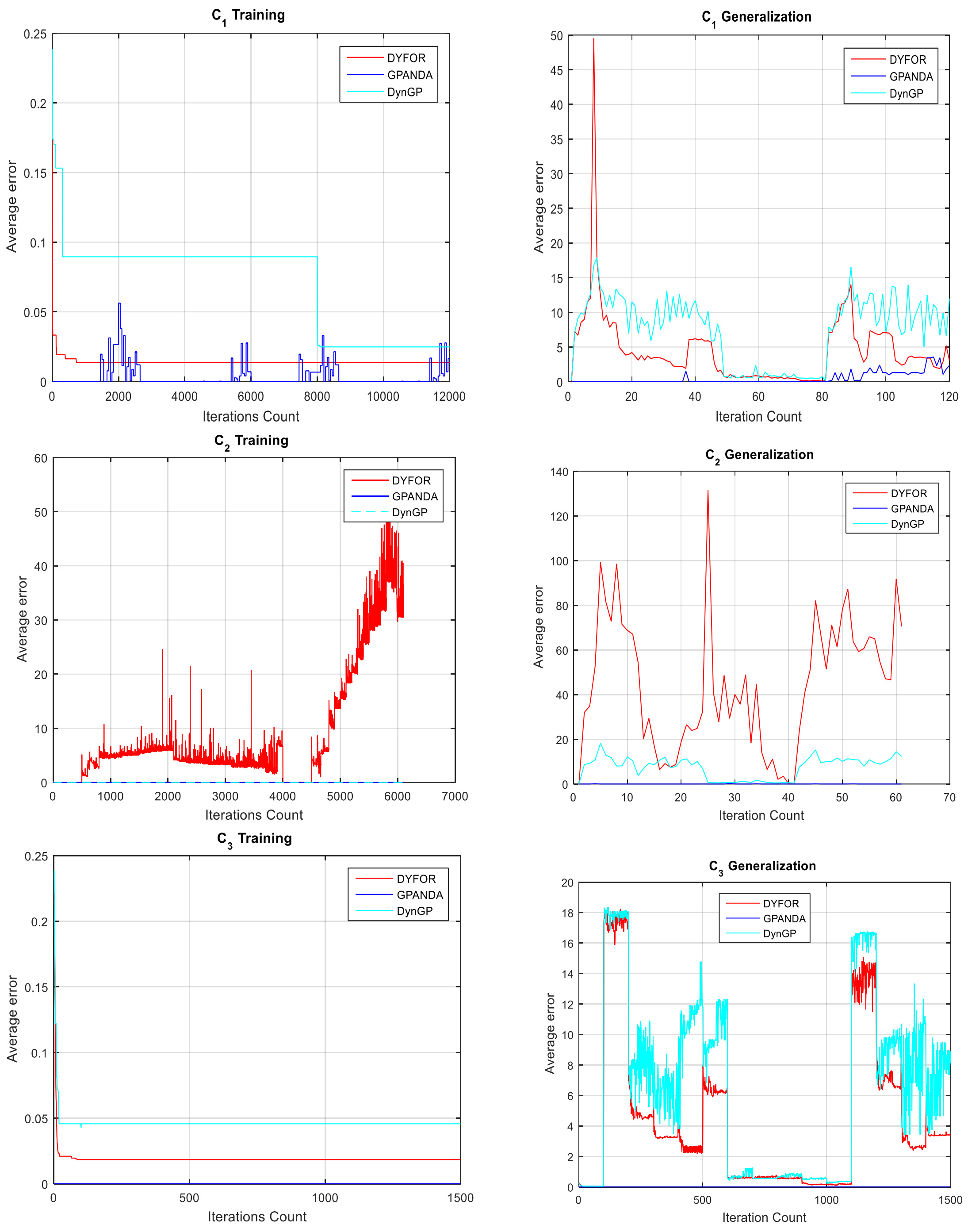


Figure 7.10: Training and Generalization E_{MS} for Progressive, Scenario $C_1 - C_3$

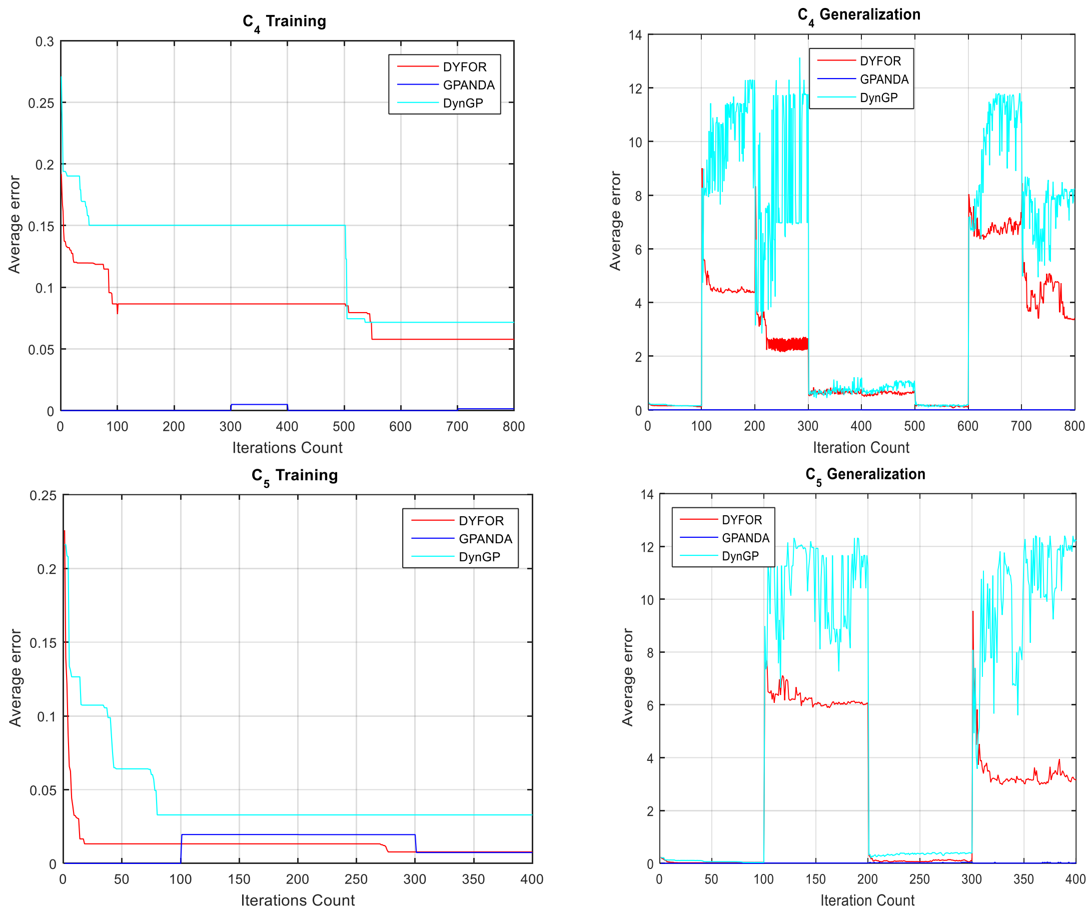


Figure 7.11: Training and Generalization E_{MS} for Progressive, Scenario $C_4 - C_5$

As illustrated in Figure 7.10 - 7.11, the performance of DyFor GP greatly deteriorates in Scenario C_2 for both training and generalization. However, DyFor GP outperforms DynGP in Scenario $C_3 - C_5$ for both training and generalization. GPANDA detect changes in the environment and adapts the model to yield outstanding performance. As such, GPANDA outperforms all algorithms in all cases in Scenario C for both training and generalization.

Table 7.12 presents the obtained results for the Progressive dataset, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.12:

- As observed in Scenario B, GPANDA outperforms all algorithms in all cases in

Scenario C.

- DyFor GP outperforms DynGP in all cases in Scenario C.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.12: Results of Predictive Performance for Progressive Environment Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0698	8.5283	0.0195	8.4040	0.0465	6.4442	0.1234	4.7274	0.0448	5.2541	0.0608	6.2716
GPANDA	**0.0059**	**0.2317**	**0.0017**	**0.0154**	**2.67e-8**	**1.0366**	**0.0049**	**8.05e-4**	**0.0116**	**0.0095**	**0.0048**	**0.2588**
DyFor GP	0.0141	4.4687	0.0105	3.8834	0.0193	4.2579	0.0811	2.4962	0.0135	2.4552	0.0277	3.5122

The algorithms under study are classified according to training and generalization performance. Table 7.13 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario C. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ whereas DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.13: Algorithm Ranking for Progressive Dataset Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	3	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2	2	2

Discussion

The performance of DyFor GP can be attributed to the ability of the algorithm to optimize the sliding window of analysis. As such, the GP in DyFor GP is presented with the data points from only a single data generating process. Thus, DyFor GP aptly tracks the changing environment and exhibits outstanding performance.

GPANDA outperforms all other algorithms in all cases in Scenario C. As already mentioned, the superior performance of GPANDA suggests the ability of the algorithm to adapt to environments with concept drift occurring. Also, the superior performance of GPANDA can be attributed to the algorithm's adaptive nature and its ability to fine-tune the predictive model using prevailing patterns.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms are given enough iterations to induce an optimal predictive model even in abruptly changing environments before an environment change. Consequently, failure to adapt or induce an optimal solution may not be attributed to the temporal severity property of the problem set under consideration but the algorithm.

Figure 7.12 - 7.13 are graphical illustrations as observed in Scenario D for the progression of $E_{MS}T$ and $E_{MS}G$ over time. The algorithms under study show similar traits as observed in Scenario C: DyFor GP outperforms DynGP in Scenario D for both training and generalization. GPANDA exhibits superior performance, consequently, outperforming all other algorithms in Scenario D for both training and generalization. The generalization performance of DyFor GP is reduced in the least abrupt changes, Scenario $D_4 - D_5$.

Table 7.14 presents the obtained results for the Progressive dataset, Scenario D. To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario C, the following are also observed in Table 7.14:

- GPANDA outperforms all other algorithms in all cases in Scenario D.
- DyFor GP outperforms DynGP in all cases in Scenario D.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

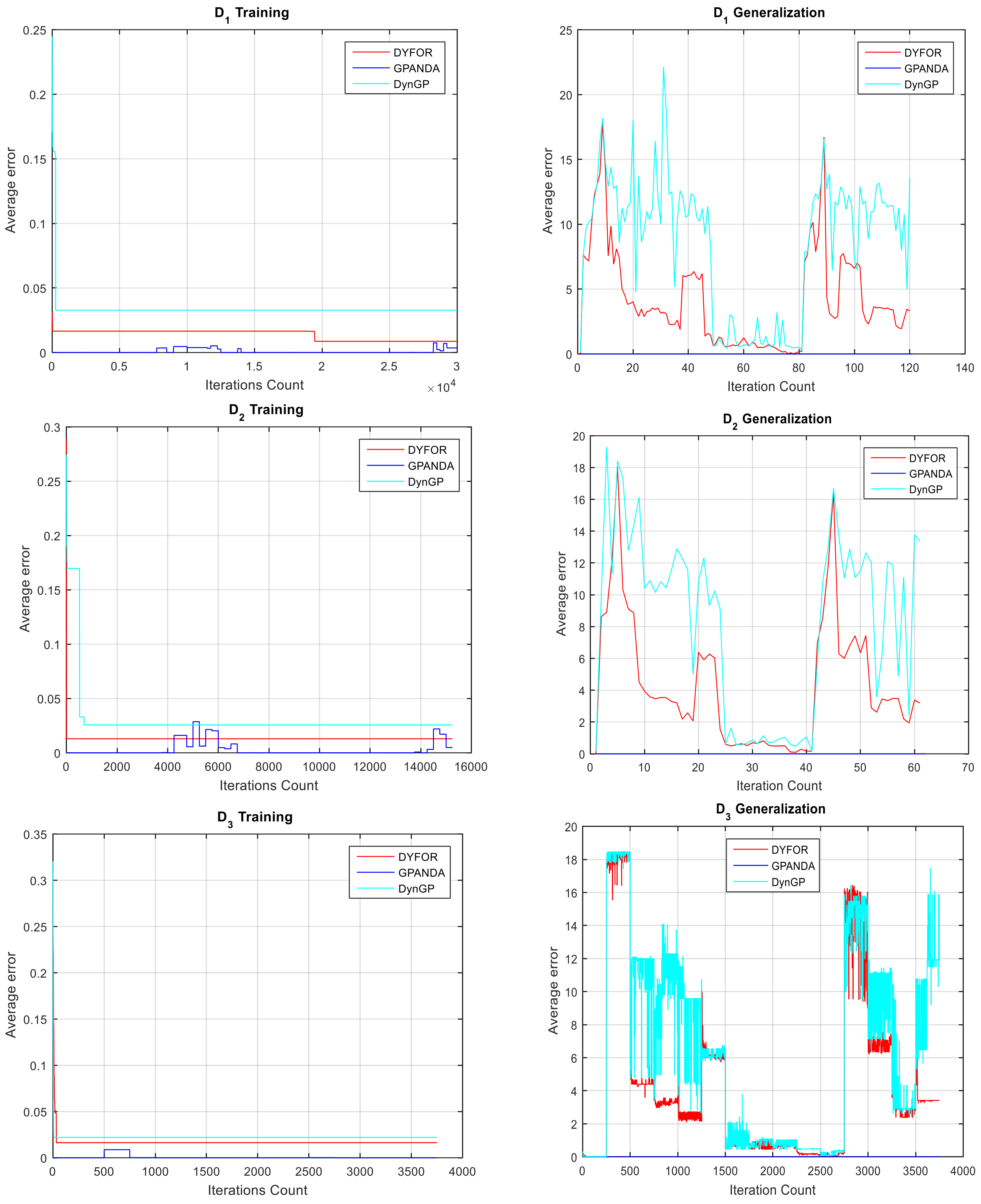


Figure 7.12: Training and Generalization E_{MS} for Progressive, Scenario $D_1 - D_3$

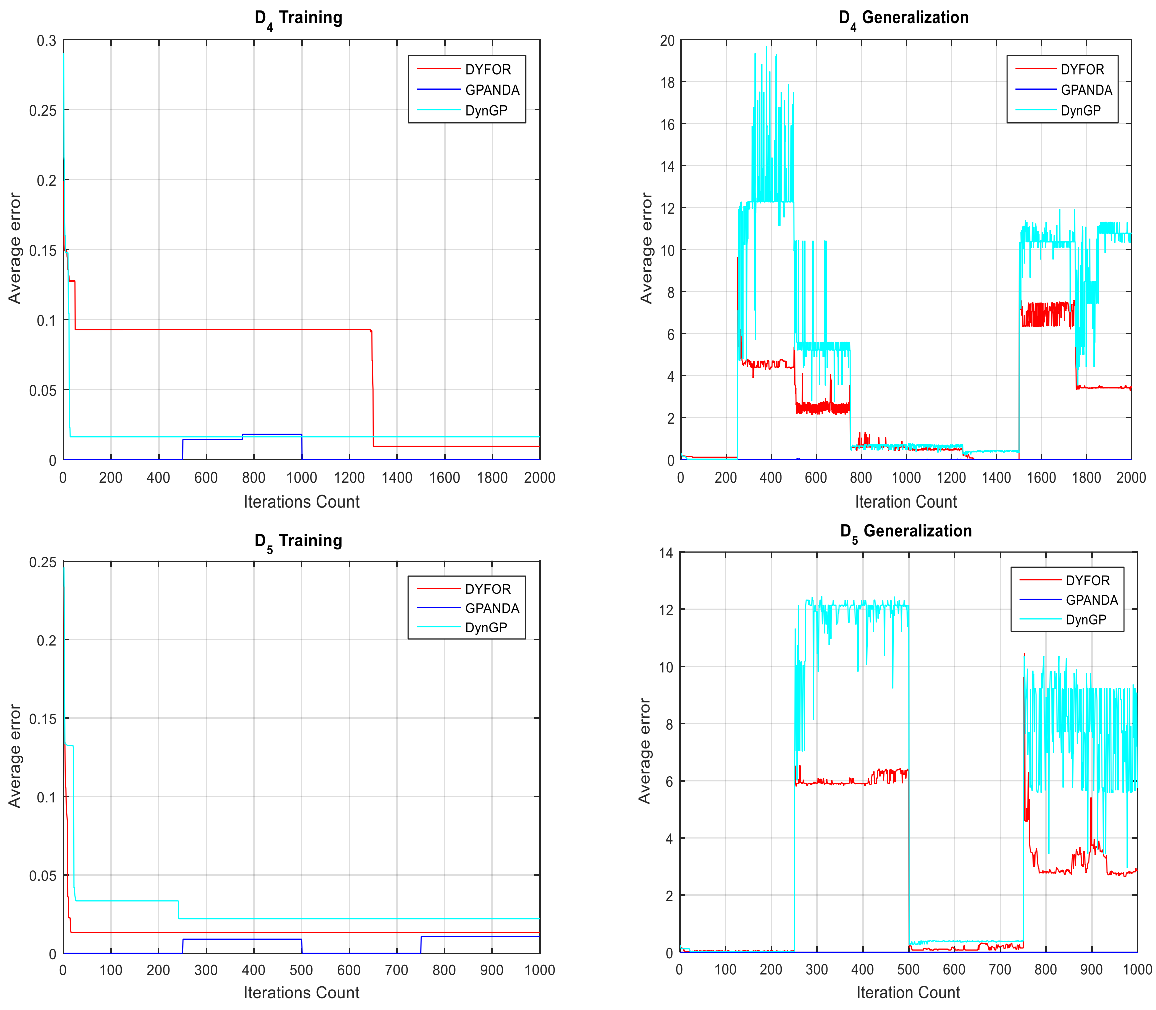


Figure 7.13: Training and Generalization E_{MS} for Progressive, Scenario $D_4 - D_5$

The algorithms under study are classified according to training and generalization per-

Table 7.14: Results of Predictive Performance for Progressive Environment Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0339	8.5067	0.0307	8.2164	0.0222	6.4235	0.0180	4.9665	0.0271	4.9884	0.0232	6.2758
GPANDA	**0.0024**	**3.68e-4**	**0.0041**	**4.14e-4**	**0.0011**	**3.45e-4**	**0.0040**	**0.0014**	**0.0049**	**6.93e-24**	**0.0033**	**0.0005**
DyFor GP	0.0138	4.1280	0.0129	4.2621	0.0171	4.3239	0.0646	2.3599	0.0142	2.3537	0.0239	3.4855

formance. Table 7.15 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario D. As observed in Scenario C, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ while DynGP is categorized as having the least effective performance for both training and generalization.

Discussion

The Progressive dataset is generated from data generating processes which gradually

Table 7.15: Algorithm Ranking for Progressive Dataset Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	3	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2	2	2

drifted, therefore, previously learned information is useful after an environment change.

GPANDA outperforms all other algorithms in all cases in Scenario D for both training and generalization. The superior performance of GPANDA suggests the capability of the algorithm to adapt to a changing environment with concept drift occurring. Also, the superior performance of GPANDA can be attributed to the repetitive nature of the dataset.

DyFor GP outperforms DynGP for both training and generalization. As already mentioned, the outstanding performance of DyFor GP can be attributed to the optimization of the analysis window, thereby evolving the prediction model using the relevant data points. DynGP has difficulties to adapt especially when the sliding window consists of more data points from the previous data generating process.

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.16 presents the average $E_{MS}G$ and average computational time ($\bar{t}$) in minutes for Scenario A-D. DynGP obtains the least average computational time in all scenarios under consideration in the Progressive dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance.

Table 7.16: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for Progressive Dataset - Scenario A-D

	A		B		C		D		Average	
Algorithm	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	5.0868	**0.007**	4.2926	**0.006**	6.2716	**0.012**	6.2758	**0.033**	5.4817	**0.0145**
GPANDA	**0.0037**	0.726	**0.0977**	2.845	**0.2588**	4.963	**0.0005**	12.14	**0.0901**	5.1685
DyFor GP	4.0670	0.149	3.5214	0.531	3.5122	0.992	3.4855	2.428	3.6465	1.025

Table 7.17 presents the average ranks obtained on $E_{MS}G$ for Scenario A-D. GPANDA is

ranked as the overall winner whereas DynGP is categorized as having the least effective performance.

Table 7.17: Overall Algorithm Ranking for Progressive Dataset

Algorithm	A	B	C	D	Average
DynGP	2.8	2.6	3	3	2.85
GPANDA	1	1	1	1	1
DyFor GP	2.2	2.4	2	2	2.15

DyFor GP optimizes the analysis window by discarding irrelevant data points to the current data generating process whereas DynGP uses dynamic parameter tuning to promote exploration to adapt to the changed optimal solution. However, GPANDA implements a piecewise approach to enumerate the optimal solution for the prevailing data points which proves to be more effective.

7.3.2 Recurrent Variation Dataset

The algorithms under study traverse the complete dataset for every scenario under consideration.

Scenario A - Abrupt Variation

The frequency of change is the same for all cases in Scenario A which is 25 iterations. Figure 7.14 - 7.15 are graphical illustrations as observed under Scenario $A_1 - A_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

DynGP outperforms DyFor GP in all cases on training. Conversely, the generalization performance of DynGP deteriorates. DyFor GP outperforms DynGP in Scenario A_2 - A_3 for generalization. However, in Scenario A_1, the generalization performance of DynGP and DyFor GP is the same. Along with the iteration's progression, the performance of DynGP and GPANDA is the same in Scenario $A_1 - A_2$ on training.

GPANDA adapts the predictive model whenever an environmental change is detected. As a result, GPANDA outperforms all other algorithms in all cases in Scenario A for both training and generalization.

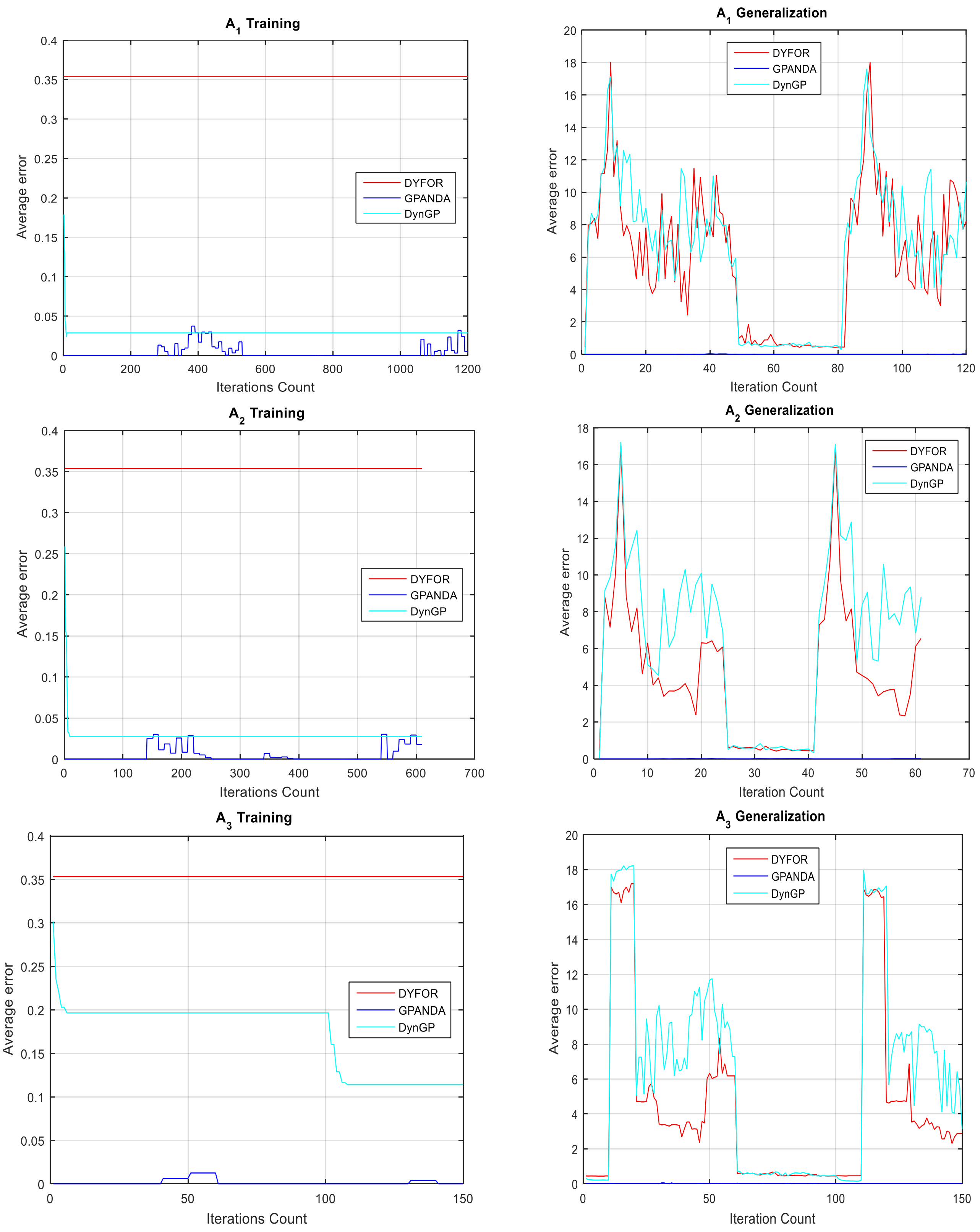


Figure 7.14: Training and Generalization E_{MS} for Recurrent, Scenario $A_1 - A_3$

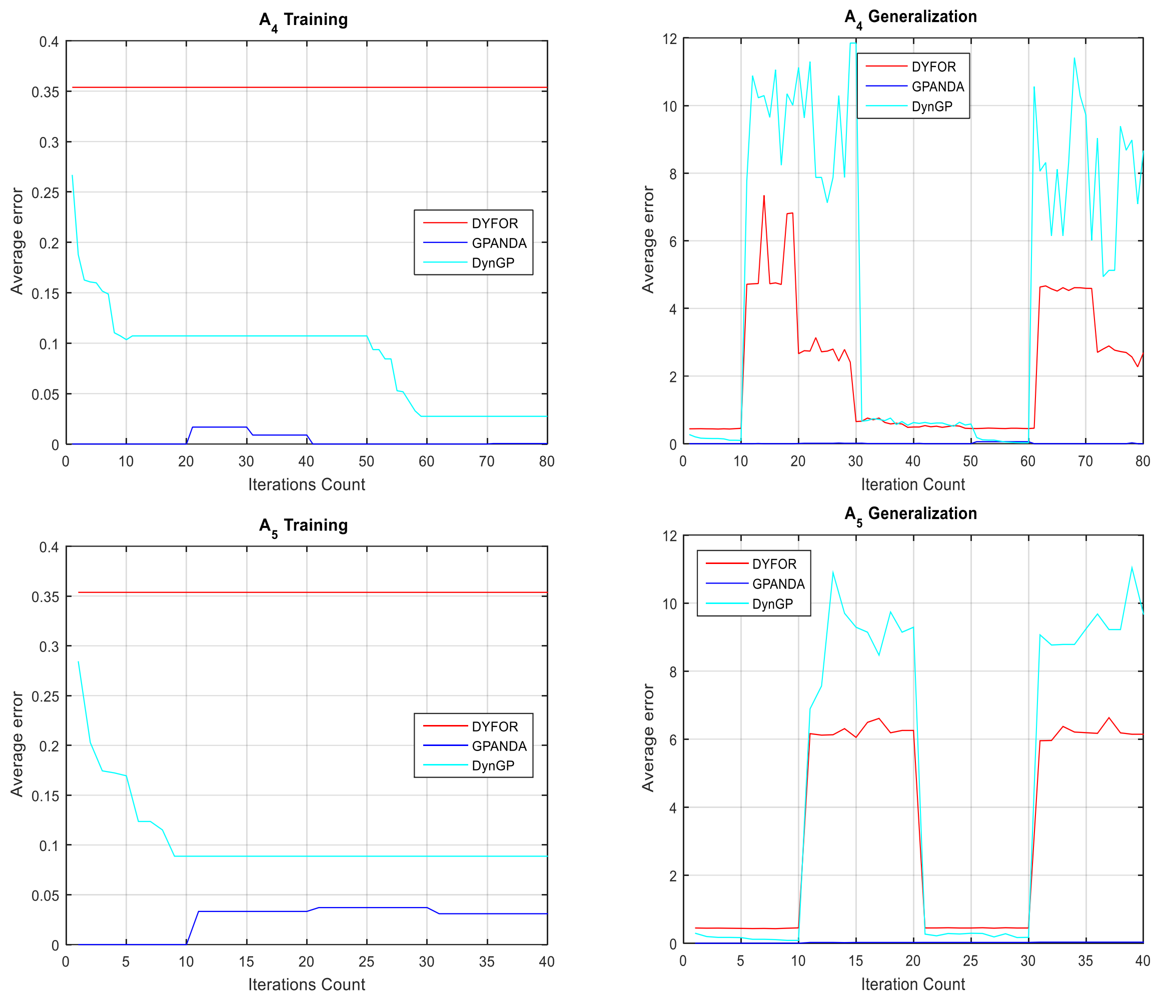


Figure 7.15: Training and Generalization E_{MS} for Recurrent, Scenario $A_4 - A_5$

GPANDA, for generalization, consistently maintains the obtained minimum after every change in the environment compared to DynGP and DyFor GP.

Table 7.18 presents the obtained results for algorithms under study on the Recurrent dataset, Scenario A. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.18:

- GPANDA outperforms all other algorithms in Scenario A for both training and generalization.
- DyFor GP outperforms DynGP for generalization in all cases in Scenario A.
- The result that DyFor GP performs better than DynGP was found to be statisti-

cally significant at the 5% level of significance.

- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.19 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario A. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for generalization. Conversely, DyFor GP is categorized as having the least effective performance on training.

Table 7.18: Results of Predictive Performance for Recurrent Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0293	6.4601	0.0288	6.5379	0.1717	6.1618	0.0874	4.6017	0.1051	4.6880	0.0844	5.6899
GPANDA	**0.0107**	**0.0040**	**0.0051**	**0.0047**	**0.0015**	**0.0032**	**0.0033**	**0.0103**	**0.0253**	**0.0198**	**0.0091**	**0.0084**
DyFor GP	0.3539	5.9055	0.3537	4.4641	0.3534	4.3038	0.3538	2.0986	0.3537	3.1848	0.3537	3.9913

Table 7.19: Algorithm Ranking for Recurrent Dataset Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	2	3	2	3	2	3	2	3	2	3	2	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	3	2	3	2	3	2	3	2	3	2	3	2

Discussion

DyFor GP evolves predictive models of outstanding generalization performance in all cases in Scenario A. The outstanding generalization performance of DyFor GP can be attributed to the ability of the algorithm to induce the model using the data points from the current data generating process.

GPANDA outperforms all other algorithms in all cases under consideration. The superior performance of GPANDA suggests the capability of the algorithm to adapt to a changing environment with concept drift occurring.

Scenario B - Quasi-Abrupt

Scenario B simulates less frequent changes having a sliding window shifting of 50 iterations. Figure 7.16 - 7.17 are graphical illustrations as observed under Scenario $B_1 - B_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

DyFor GP outperforms DynGP on training in Scenario B_1 and generalization in Scenario $B_3 - B_4$. Conversely, DynGP outperforms DyFor GP on training in Scenario $B_2 - B_5$. The generalization performance of DyFor GP and DynGP is the same in Scenario B_2. GPANDA outperforms all other algorithms in Scenario B for both training and generalization which suggests that the algorithm detects the environmental changes due to concept drifts occurring and then adapts the predictive model to yield outstanding performance.

Table 7.20 presents the obtained results for the Recurrent dataset, Scenario B. To ascertain the significance of the results obtained for Scenario B, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. Similar traits, as observed in Scenario A, are also observed in Table 7.20 except for Scenario B_1:

- GPANDA outperforms all algorithms in Scenario B.
- DyFor GP outperforms DynGP in Scenario B except for Scenario B_1.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.20: Results of Predictive Performance for Recurrent Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0618	5.8157	0.0248	8.2919	0.0885	6.9583	0.0405	4.3645	0.0599	5.1190	0.0539	6.1076
GPANDA	**0.0055**	**0.0068**	**0.0042**	**4.60e-8**	**0.0019**	**0.0011**	**0.0015**	**0.0245**	**0.0182**	**0.4424**	**0.0032**	**0.0081**
DyFor GP	0.0268	8.3844	0.3537	5.2725	0.3532	4.6026	0.3532	2.3048	0.3534	3.4516	0.2717	4.8910

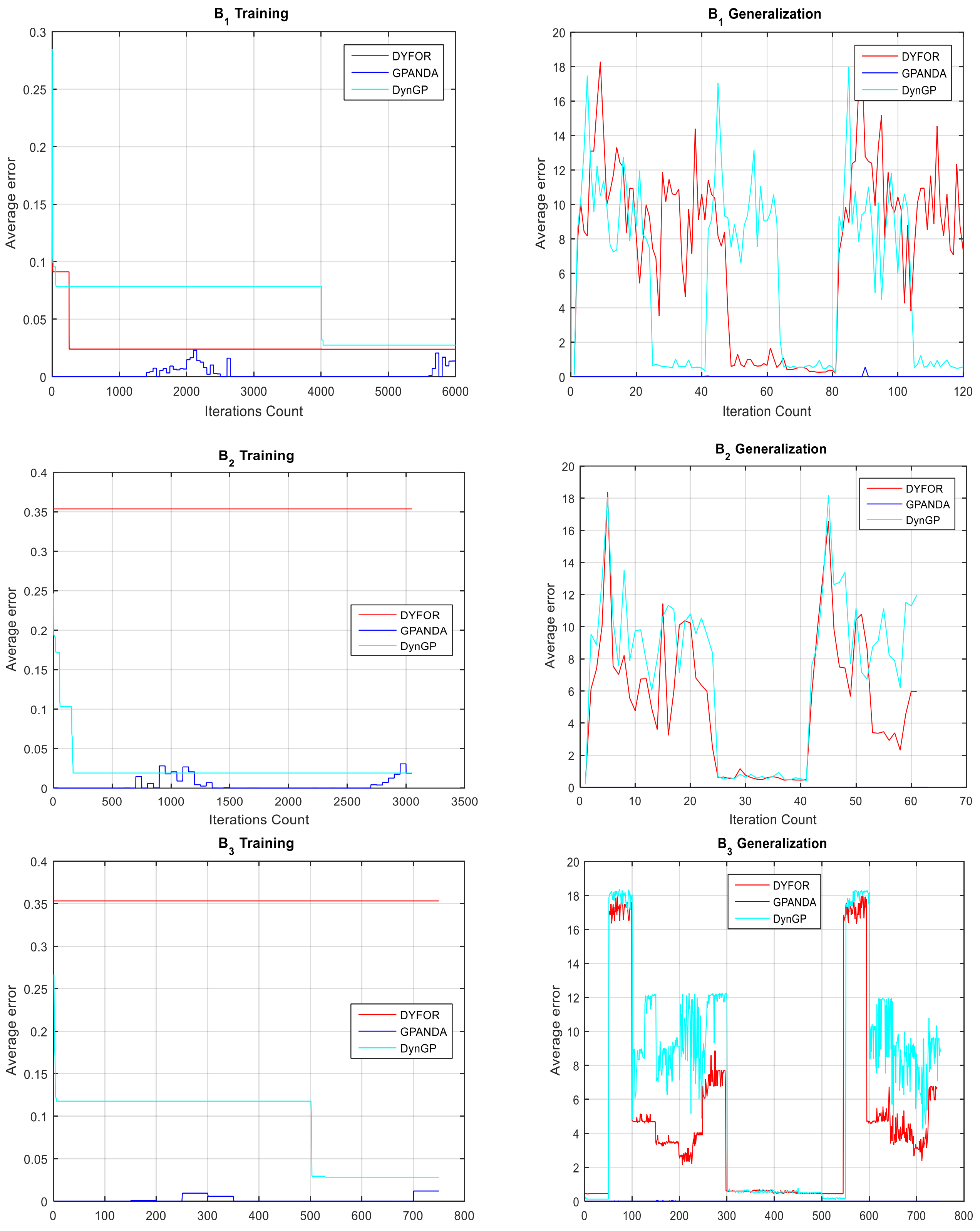


Figure 7.16: Training and Generalization E_{MS} for Recurrent, Scenario B_1 - B_3

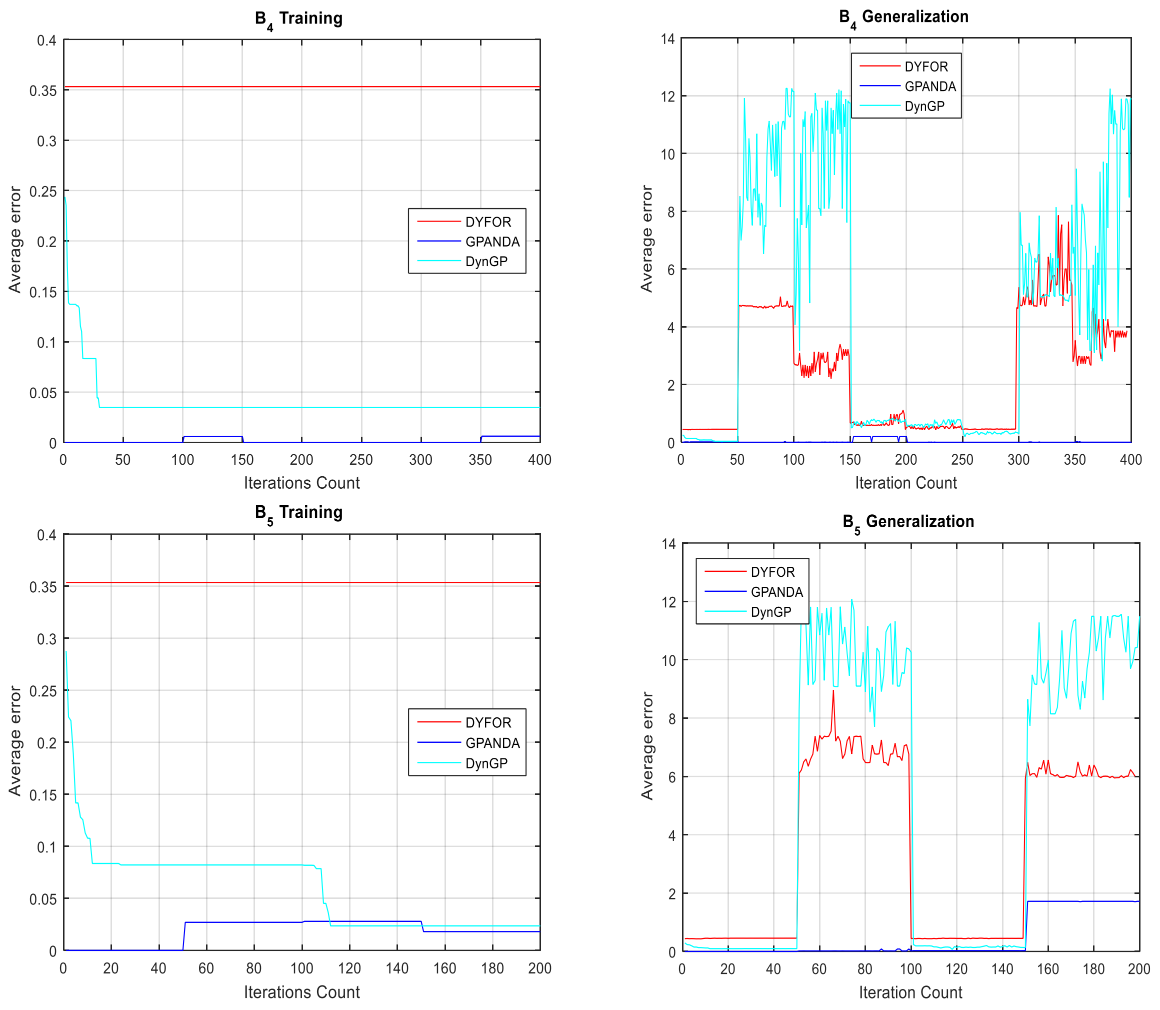


Figure 7.17: Training and Generalization E_{MS} for Recurrent, Scenario B_4 - B_5

The algorithms under study are classified according to $E_{MS}T$ and $E_{MS}G$. Table 7.21 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ in Scenario B. As observed in Scenario A, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ in Scenario B. DynGP is categorized as having the least effective performance for generalization. Conversely, DyFor GP is categorized as having the least effective performance on training.

Table 7.21: Algorithm Ranking for Recurrent Dataset Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2	2	3	2	3	2	3	2	3	2.2	2.8
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	3	3	2	3	2	3	2	3	2	2.8	2.2

Discussion

The poor generalization performance of the model produced by DynGP can be attributed to conflicting decision boundaries present in the sliding window of analysis.

As observed in Scenario A, DyFor GP evolves predictive models of outstanding generalization performance in all cases in Scenario B which can be attributed to the ability of the algorithm to induce the model using only the data points from the current data generating process.

GPANDA outperforms all algorithms in Scenario B for both training and generalization which suggests the effectiveness of the piecewise regression approach of the algorithm to adapt to a changing environment with concept drift occurring.

Scenario C - Quasi-Progressive Variation

Scenario C simulates less frequent changes of 100 iterations before a sliding window shifting which availed sufficient iterations to all algorithms. Figure 7.18 - 7.19 are graphical illustrations as observed under Scenario $C_1 - C_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario B, similar traits are also observed in Scenario C: DyFor GP outperforms DynGP on training in Scenario C_1 and generalization in Scenario $C_2 - C_4$ as graphically illustrated in Figure 7.18 - 7.19. Conversely, DynGP outperforms DyFor GP on training in Scenario $C_2 - C_5$.

GPANDA outperforms all other algorithms in Scenario C for both training and generalization.

Table 7.22 presents the obtained results for the Recurrent dataset, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario B, similar traits are also observed in Table 7.22:

- GPANDA outperforms all algorithms in Scenario C for both training and generalization.
- DyFor GP outperforms DynGP for generalization in Scenario C.

- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.

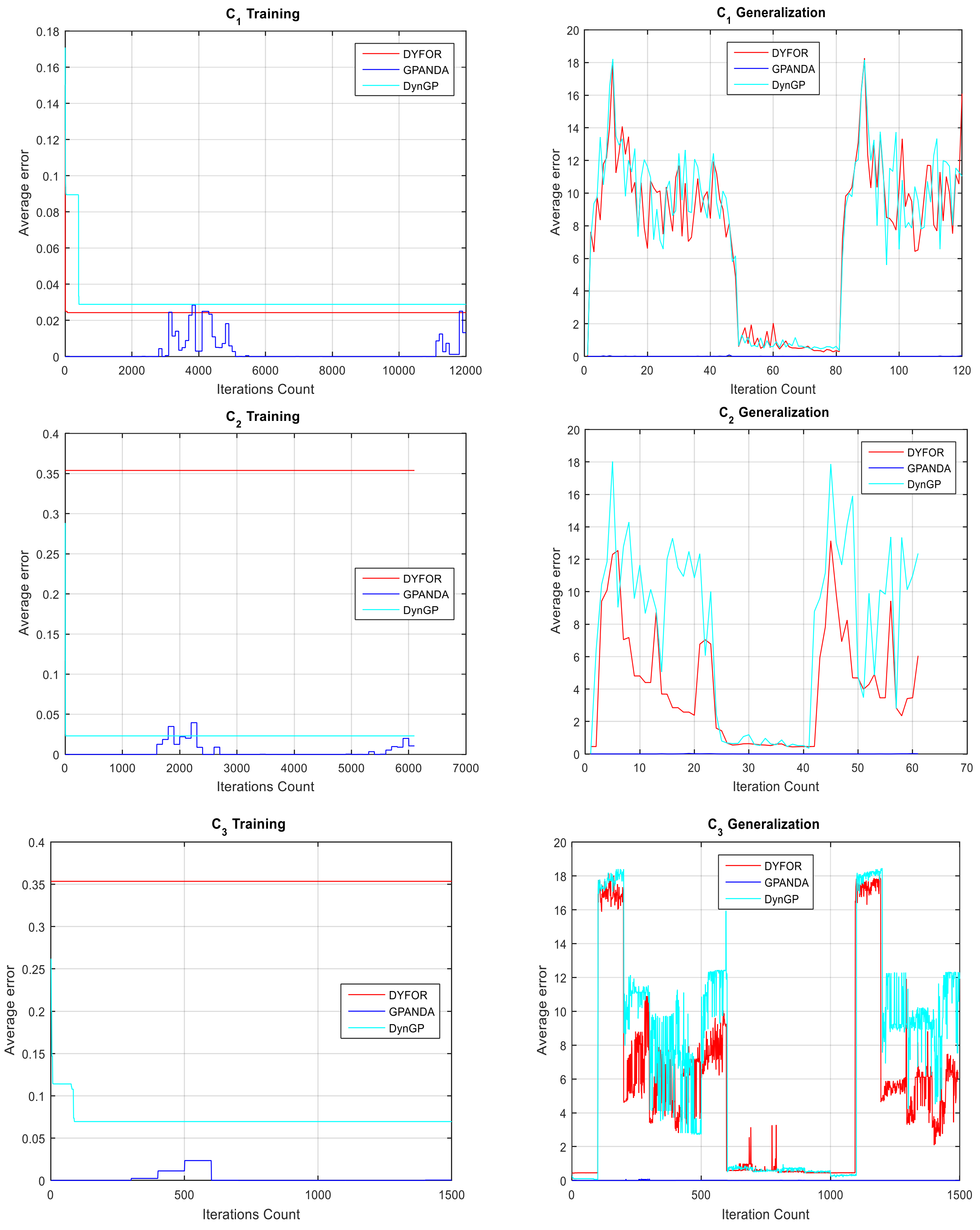


Figure 7.18: Training and Generalization E_{MS} for Recurrent, Scenario $C_1 - C_3$

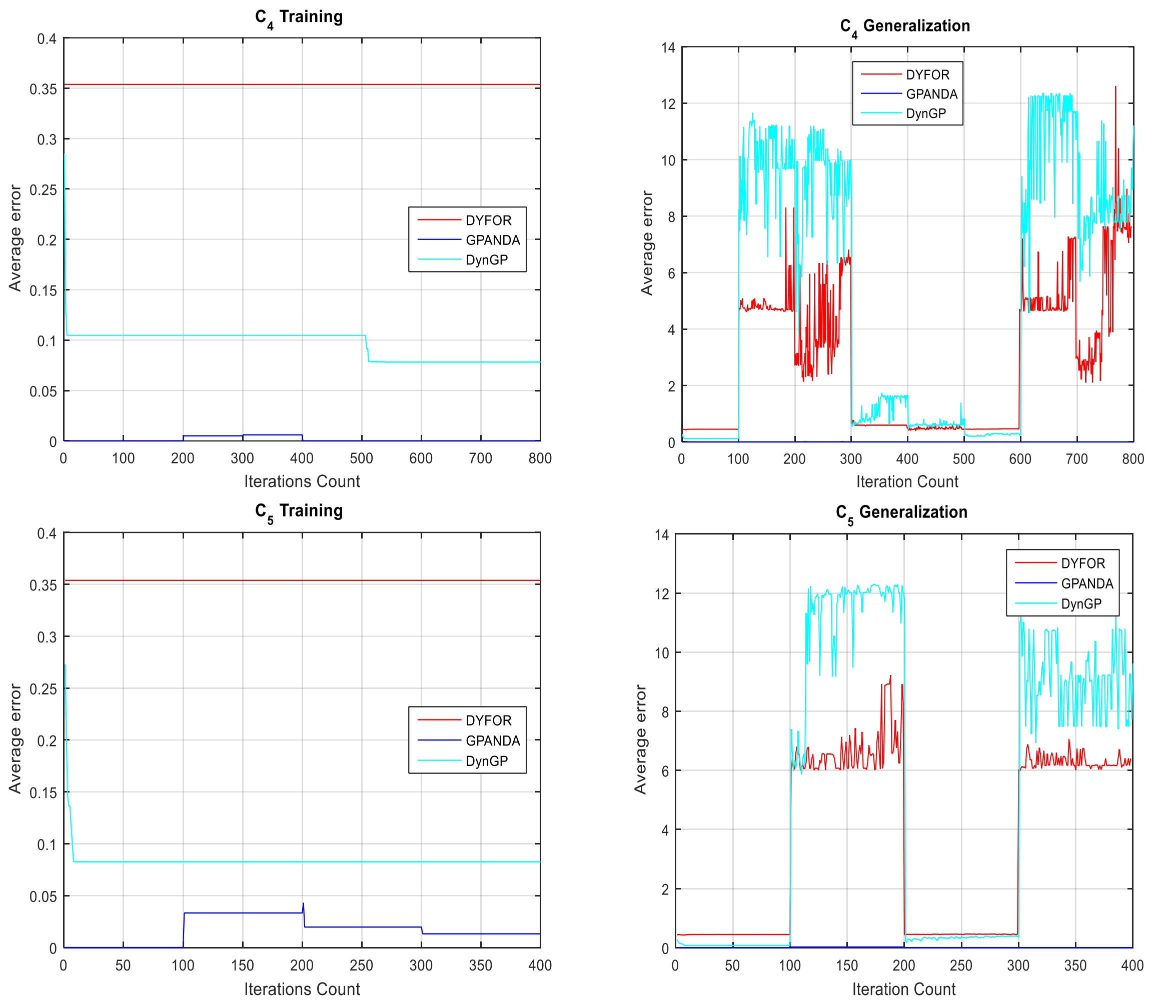


Figure 7.19: Training and Generalization E_{MS} for Recurrent, Scenario $C_4 - C_5$

Table 7.22: Results of Predictive Performance for Recurrent Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0310	8.7324	0.0232	8.5170	0.0723	6.8666	0.0840	5.1555	0.0840	5.1555	0.0589	6.4854
GPANDA	**0.0035**	**0.0039**	**0.0039**	**0.0027**	**0.0025**	**0.0022**	**0.0166**	**0.0064**	**0.0166**	**0.0064**	**0.0086**	**0.0043**
DyFor GP	0.0243	8.4486	0.3539	4.0830	0.3534	5.2485	0.3538	3.4459	0.3538	3.4459	0.2878	4.7343

- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.23 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ in Scenario C. As observed in Scenario B, GPANDA is ranked as the overall winner for both $E_{MS}T$ and

$E_{MS}G$ in Scenario C. DynGP is categorized as having the least effective performance for generalization. Conversely, DyFor GP is categorized as having the least effective performance on training.

Table 7.23: Algorithm Ranking for Recurrent Dataset Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	2	3	2	3	2	3	2	3	2.2	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	3	2	3	2	3	2	3	2	2.8	2

Discussion

The technique to optimize the sliding window of analysis in DyFor GP makes it easier for the algorithm to distinguish between data points from the past and current data generating processes, therefore, the algorithm aptly tracks the changing environment to yield outstanding performance for generalization.

GPANDA outperforms all algorithms in all cases in Scenario C for both training and generalization. As already mentioned, the superior performance of GPANDA can be attributed to the effectiveness of the piecewise regression approach of the algorithm to adapt to a changing environment with concept drift occurring.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms are given enough iterations to induce an optimal predictive model even in abruptly changing environments before an environment change. Consequently, failure to adapt or induce an optimal solution may not necessarily be attributed to the temporal severity property of the problem set under consideration but may be of the algorithm.

Figure 7.20 - 7.21 are graphical illustrations as observed in Scenario D for the progression of $E_{MS}T$ and $E_{MS}G$ over time. The algorithms under study show similar traits as observed in Scenario A: DynGP outperforms DyFor GP in all cases on training. Conversely, the generalization performance of DynGP deteriorates and DyFor GP outperforms DynGP in Scenario D for generalization. For some instances in Scenario

$D_1 - D_2$, the performance of DynGP and GPANDA is the same.

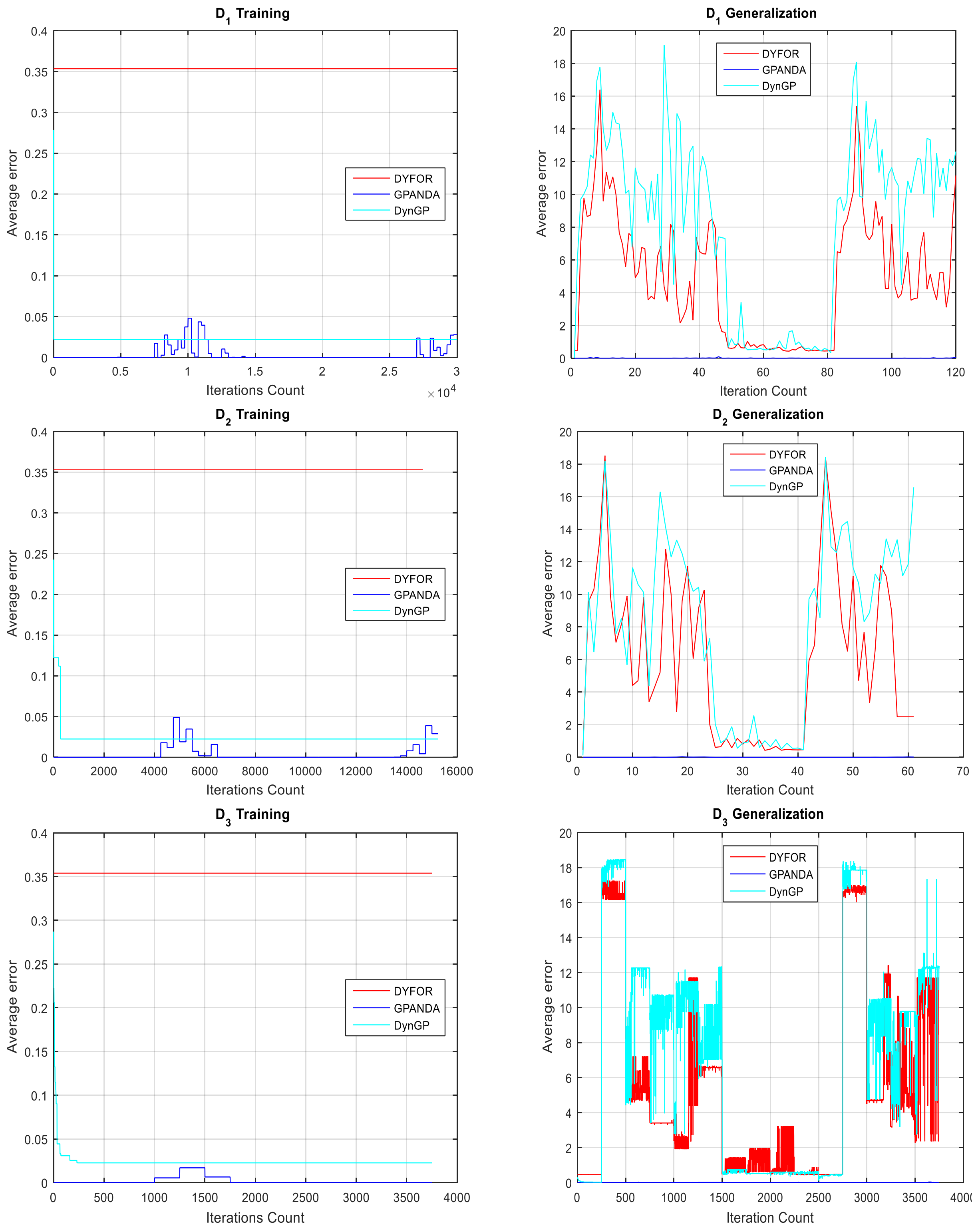


Figure 7.20: Training and Generalization E_{MS} for Recurrent, Scenario $D_1 - D_3$

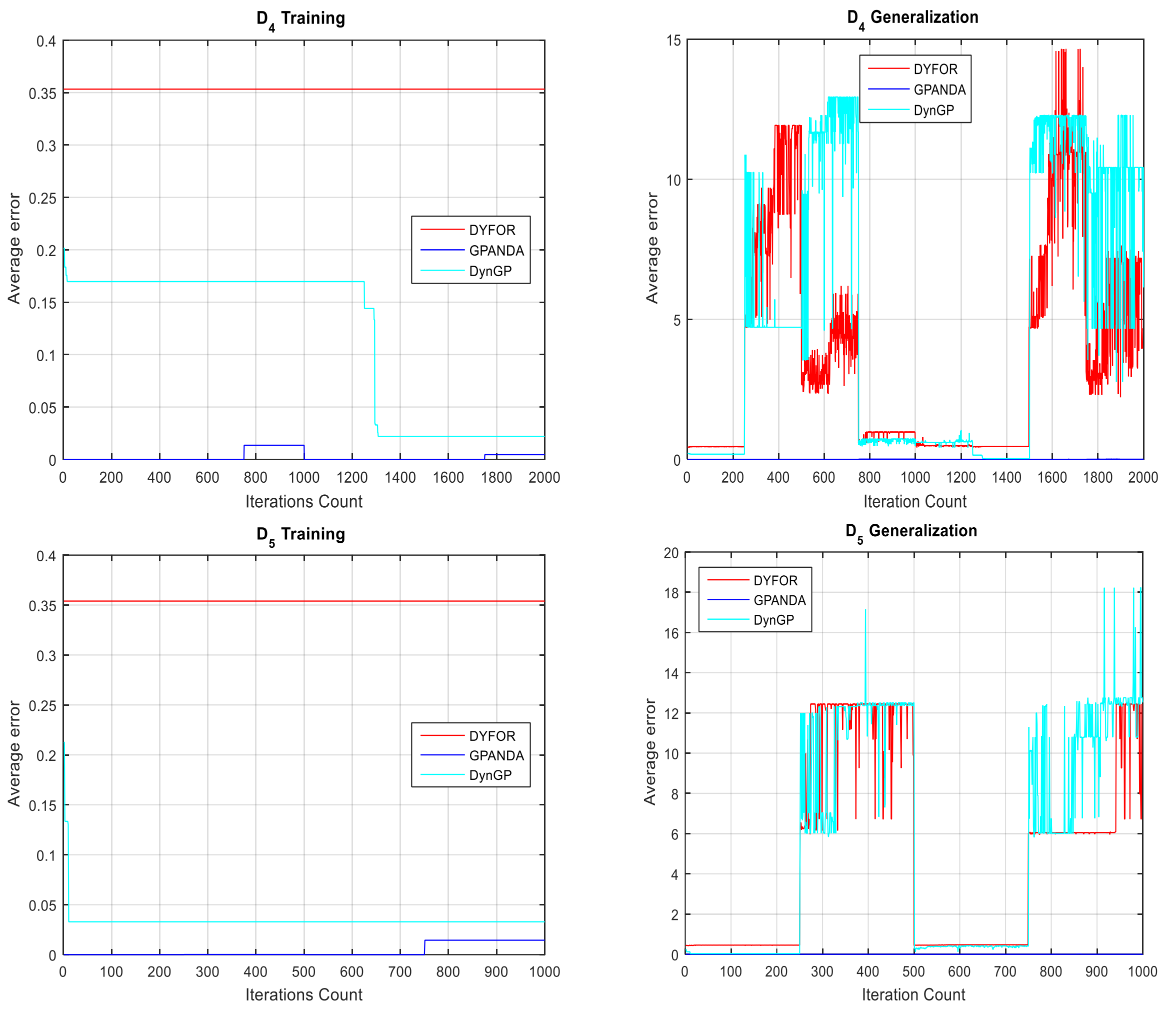


Figure 7.21: Training and Generalization E_{MS} for Recurrent, Scenario $D_4 - D_5$

GPANDA outperforms all algorithms under study in all cases in Scenario D for both training and generalization. GPANDA, for generalization, consistently maintains the obtained minimum compared to DynGP and DyFor GP after every change in the environment as graphically illustrated in Figure 7.20 - 7.21.

Table 7.24 presents the obtained results for all algorithms on the Recurrent dataset, Scenario D. To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario A, similar traits can be observed in Table 7.24:

- GPANDA outperforms all algorithms in Scenario D for both training and generalization.

- DyFor GP outperforms DynGP for generalization in all cases in Scenario D.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.24: Results of Predictive Performance for Recurrent Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0221	8.2310	0.0242	8.2264	0.0241	8.0734	0.1171	4.9055	0.0340	5.3824	0.0443	6.7637
GPANDA	**0.0040**	**0.0049**	**0.0042**	**0.0047**	**0.0019**	**0.0505**	**0.0278**	**0.0013**	**0.0036**	**1.33e-4**	**0.0083**	**0.0123**
DyFor GP	0.3534	5.5553	0.3537	6.0061	0.3539	5.3096	0.3533	3.6301	0.3540	4.9052	0.3536	5.0812

The training algorithms under study are classified according to training and generalization performance. Table 7.25 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario D. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for generalization. Conversely, DyFor GP is categorized as having the least effective performance on training.

Table 7.25: Algorithm Ranking for Recurrent Dataset Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	2	3	2	3	2	3	2	3	2	3	2	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	3	2	3	2	3	2	3	2	3	2	3	2

Discussion

GPANDA outperforms all algorithms in all cases under consideration for both training and generalization. The superior performance of GPANDA suggests the ability of the algorithm to adapt to a changing environment with concept drift occurring.

DyFor GP evolves predictive models of improved generalization performance in all cases in Scenario D. The superior generalization performance of DyFor GP can be attributed to the ability of the algorithm to induce the model using the data points from the current data generating process.

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.26 presents the average $E_{MS}G$ and average computational time $(\bar{t})$ in minutes for Scenario A-D. As observed in the Progressive dataset, DynGP obtains the least average computational time in all scenarios under consideration in the Recurrent dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance.

Table 7.26: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for Recurrent Dataset - Scenario A-D

	A		B		C		D		Average	
Algorithm	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	5.6899	**0.018**	6.1076	**0.063**	6.4854	**0.107**	6.7637	**0.276**	6.2616	**0.116**
GPANDA	**0.0084**	2.376	**0.0081**	8.454	**0.0043**	20.13	**0.0123**	39.06	**0.0082**	18.255
DyFor GP	3.9913	1.32	4.8910	4.306	4.7343	12.35	5.0812	21.70	4.6744	9.919

Table 7.27 presents the obtained results for Scenario A-D. GPANDA is ranked as the overall winner whereas DynGP is categorized as having the least effective performance.

Table 7.27: Overall Algorithm Ranking for Recurrent Dataset

Algorithm	A	B	C	D	Average
DynGP	3	2.8	3	3	2.95
GPANDA	1	1	1	1	1
DyFor GP	2	2.2	2	2	2.05

7.3.3 Abrupt Variation Dataset

The algorithms under study traverse the complete data set for every scenario under consideration.

Scenario A - Abrupt Variation

The frequency of change is the same for all cases in Scenario A which is 25 iterations. Figure 7.22 - 7.23 are graphical illustrations as observed under Scenario A_1 - A_5 for the progression of $E_{MS}T$ and $E_{MS}G$ over algorithm's iterations.

The adaptation to change in the environment becomes less difficult as spatial severity increase from A_1 to A_5 as illustrated in Figure 7.22 - 7.23.

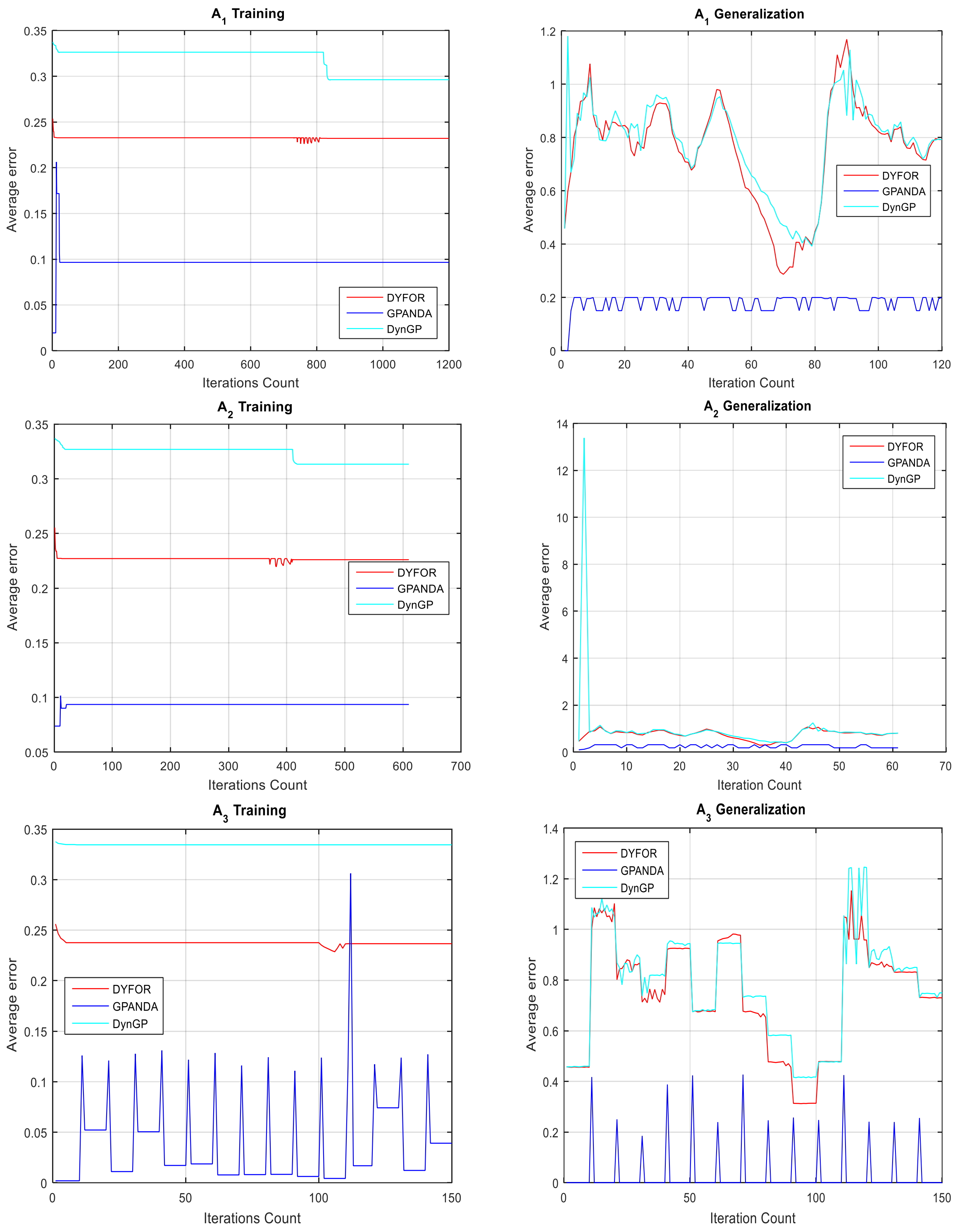


Figure 7.22: Training and Generalization E_{MS} for Abrupt, Scenario A_1 - A_3

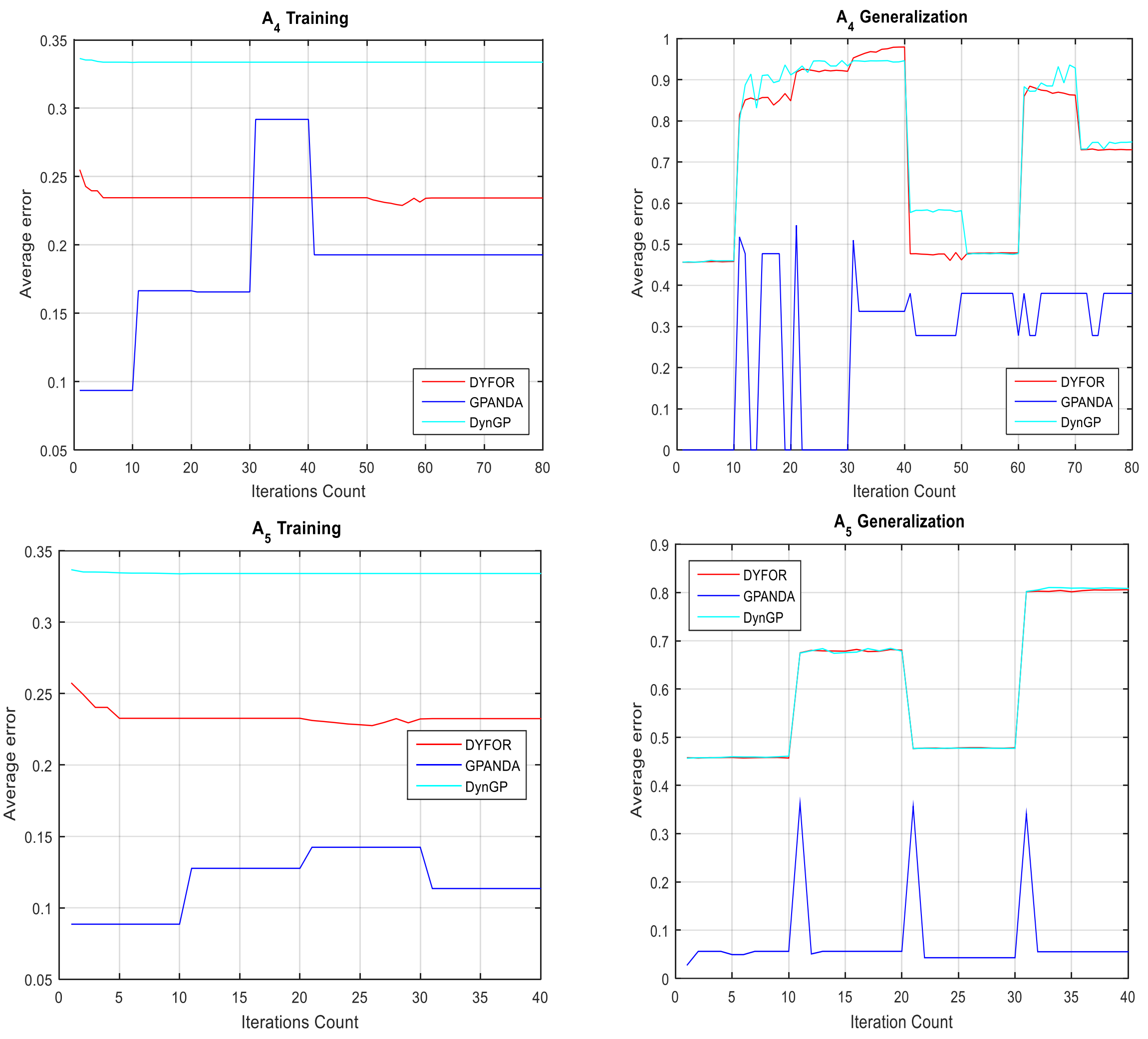


Figure 7.23: Training and Generalization E_{MS} for Abrupt, Scenario A_4 - A_5

The error profiles for generalization for all algorithms are generally higher in Scenario A_1 - A_3 compared to other scenarios. Thus, the predictive model is evolved using more of the previous data generating process data points, consequently, reduces the generalization performance. Higher error peaks are attributed to abrupt changes that tend to suddenly discard old historical data and replace it with the recent historical data from the current data generating process at every change of the environment.

As graphically illustrated in Figure 7.22 - 7.23, the performance of DynGP and DyFor GP is the same for generalization in Scenario A. However, DyFor GP outperforms DynGP on training in all cases in Scenario A. GPANDA exhibits superior performance, consequently, outperforms all algorithms by a greater margin in Scenario A for both training and generalization.

The significant error peaks are evident in GPANDA error profile for Scenario A_3 - A_5 illustrated in Figure 7.22 - 7.23, which signified frequent environmental changes. The performance of GPANDA deteriorates in Scenario A_4. The reduced performance of GPANDA can be attributed to the presence of different patterns within a sliding window, especially with more of those from the past data generating process.

As already mentioned, the abrupt change happening in Scenario A_5 made this scenario much easier to adapt since the sliding window discards all patterns from the previous data generating process.

Table 7.28 presents the obtained $E_{MS}T$ and $E_{MS}G$ results for the Abrupt dataset Scenario A. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.28:

- GPANDA outperforms all algorithms in all cases in Scenario A for both training and generalization.
- DyFor GP outperforms DynGP.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario A_5.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance except for Scenario A_5 between DyFor GP and DynGP.

Table 7.28: Results of Predictive Performance for Abrupt Environment Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.3170	0.7811	0.3227	0.9906	0.3345	0.7718	0.3338	0.7410	0.3343	0.6057	0.3284	0.7780
GPANDA	**0.0965**	**0.1790**	**0.0933**	**0.2432**	**0.0331**	**0.0282**	**0.1918**	**0.2611**	**0.1494**	**0.0741**	**0.1128**	**0.1571**
DyFor GP	0.2325	0.7539	0.2267	0.7511	0.2372	0.7360	0.2347	0.7188	0.2334	0.6046	0.2329	0.7128

The algorithms under study are classified according to training and generalization performance. Table 7.29 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario A. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance on training.

Table 7.29: Algorithm Ranking for Abrupt Dataset Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	2.5	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2.5	2	2

Discussion

GPANDA outperforms all algorithms in all cases under consideration for both training and generalization. The superior performance of GPANDA suggests the capability of the algorithm to adapt to an abrupt changing environment with concept drift occurring. As already mentioned, the superior performance of GPANDA is attributed to the algorithm's adaptive nature and its ability to fine-tune the predictive model using prevailing patterns.

DyFor GP outperforms DynGP on training, however, the generalization performance of DyFor GP is reduced to yield performance the same to DynGP.

The progressive changes happening in Scenario A_1 - A_4 brought new data points from a different data generating process after an environmental change, thereby presenting two concepts on each sliding window. However, GPANDA detects an environmental change and adapts the underlying predictive model to yield superior performance in all cases in Scenario A.

The abrupt change happening in Scenario A_5 yields an increased performance for generalization by all algorithms since the sliding window discards all patterns from the past data generating process, thus, generating a predictive model using the recent historical data points generated from current data generating process.

Scenario B - Quasi-Abrupt

Scenario B simulates less frequent changes having a sliding window shift of 50 iterations. Figure 7.24 - 7.25 are graphical illustrations as observed under Scenario B_1 - B_5 for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario A, the same traits are also observed: DyFor GP outperforms DynGP on training in Scenario B whereas for generalization the performance of DyFor GP and DynGP is the same.

GPANDA detects an environmental change and thereby adapted the predictive model, evident with obtained improved predictive performance as graphically illustrated in Figure 7.24 - 7.25.

GPANDA outperforms all other algorithms in Scenario B for both training and generalization. However, the generalization performance of GPANDA deteriorates in Scenario $B_3 - B_4$. As also observed in Scenario A, the significant error peaks are evident in GPANDA error profile for Scenario $B_2 - B_4$.

Table 7.30 presents the obtained results for the Abrupt dataset, Scenario B. To ascertain the significance of the results obtained for Scenario B, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario A, similar traits are also observed in Scenario B for the results presented in Table 7.30:

- DyFor GP outperforms DynGP for generalization in all cases in Scenario B except for Scenario B_5.
- GPANDA outperforms all algorithms in all cases in Scenario B for both training and generalization.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario B_5.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

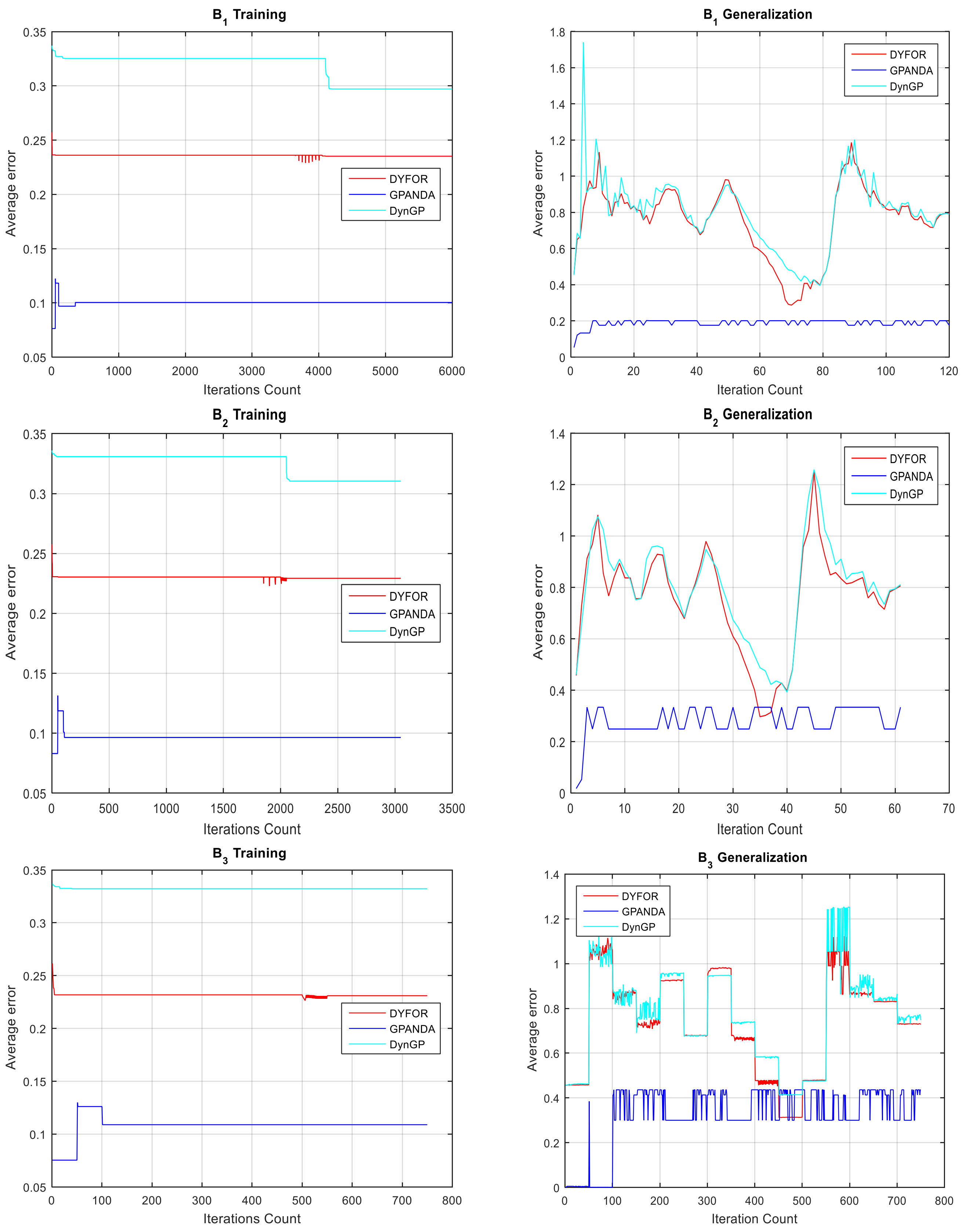


Figure 7.24: Training and Generalization E_{MS} for Abrupt, Scenario B_1 - B_3

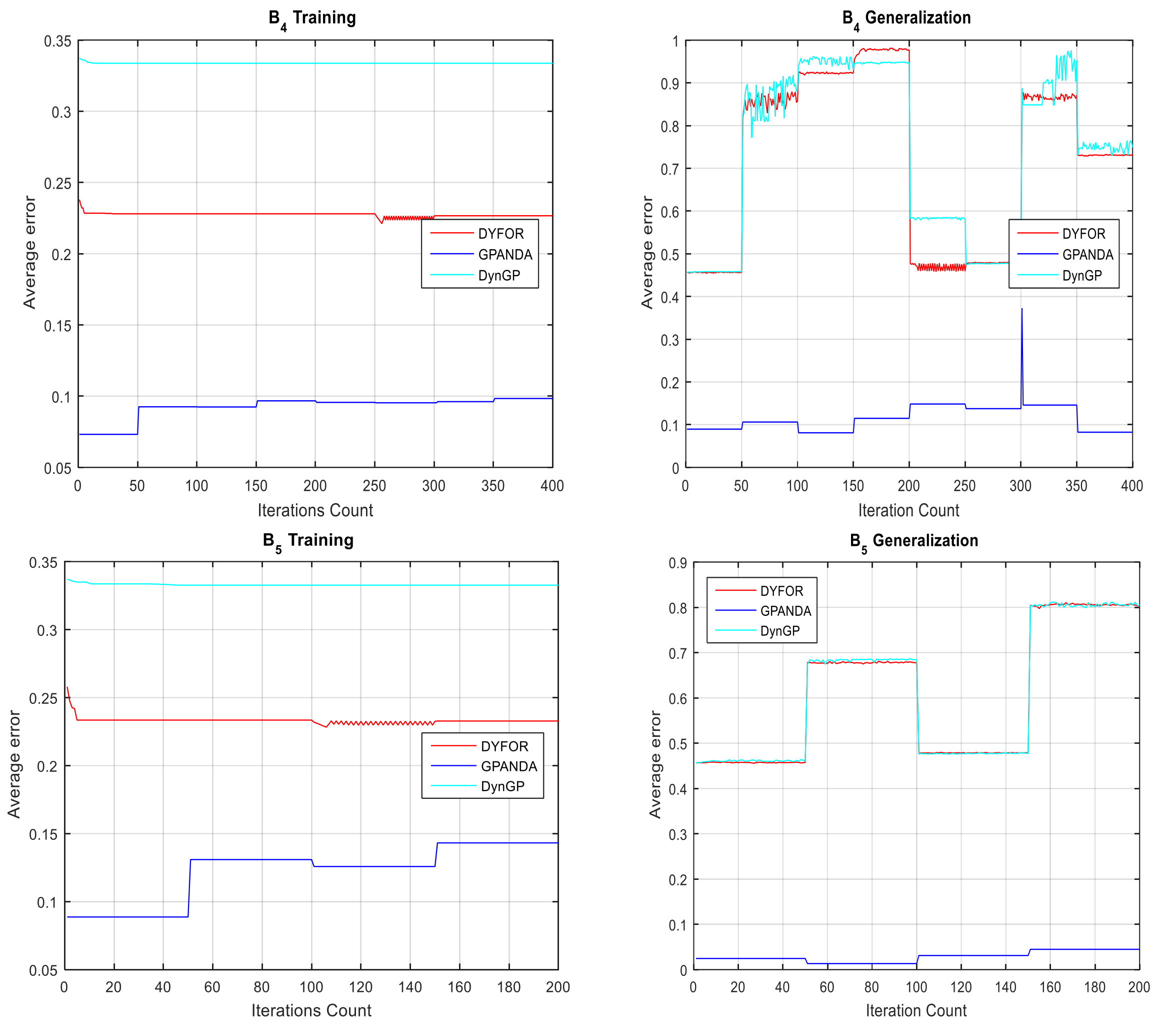


Figure 7.25: Training and Generalization E_{MS} for Abrupt, Scenario B_4 - B_5

The algorithms under study are classified according to training and generalization performance. Table 7.31 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario B. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for both $E_{MS}T$ and $E_{MS}G$. **Discussion** GPANDA outperforms all algorithms in all cases in Scenario B for both training and generalization. As already mentioned, the superior performance of GPANDA suggests the capability of the algorithm to adapt to an abrupt changing environment with con-

Table 7.30: Results of Predictive Performance for Abrupt Environment Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.3166	0.7988	0.3242	0.7985	0.3323	0.7712	0.3338	0.7409	0.3329	0.6068	0.3279	0.7432
GPANDA	**0.1002**	**0.1889**	**0.0966**	**0.2811**	**0.1078**	**0.3218**	**0.0978**	**0.1192**	**0.1285**	**0.0330**	**0.1061**	**0.1888**
DyFor GP	0.2358	0.7577	0.2300	0.7567	0.2315	0.7371	0.2274	0.7198	0.2330	0.6048	0.2315	0.7152

Table 7.31: Algorithm Ranking for Abrupt dataset Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	2.5	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2.5	2	2

cept drift occurring. The superior performance of GPANDA can be attributed to the algorithm's ability to detect an environmental change and adapts a predictive model using the prevailing patterns even in increased temporal and spatial severities.

As observed in Scenario A, DyFor GP significantly outperforms DynGP on training, however, the generalization performance of DyFor GP is reduced to yield performance the same to DynGP.

As already explained, the abrupt change happening in Scenario B_5 yields an increased performance since the sliding window discards all patterns from the past data generating process and presents data points from the current data generating process.

Scenario C - Quasi-Progressive Variation

Scenario C simulates less frequent changes having a sliding window shift of 100 iterations. Figure 7.26 - 7.27 are graphical illustrations as observed under Scenario $C_1 - C_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario B, DyFor GP outperforms DynGP on training in Scenario C whereas the performance of DyFor GP and DynGP for generalization is the same. As illustrated in Figure 7.26 - 7.27, GPANDA exhibits superior performance, consequently, outperforms all other algorithms in Scenario C for both training and generalization.

As also observed in Scenario B, the significant error peaks are evident in GPANDA error profile for Scenario $C_3 - C_4$. The significant error peaks indicate changes in the environment which entail the deteriorated predictive performance of other algorithms under study.

Table 7.32 presents the obtained $E_{MS}T$ and $E_{MS}G$ results for the Abrupt dataset, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1.

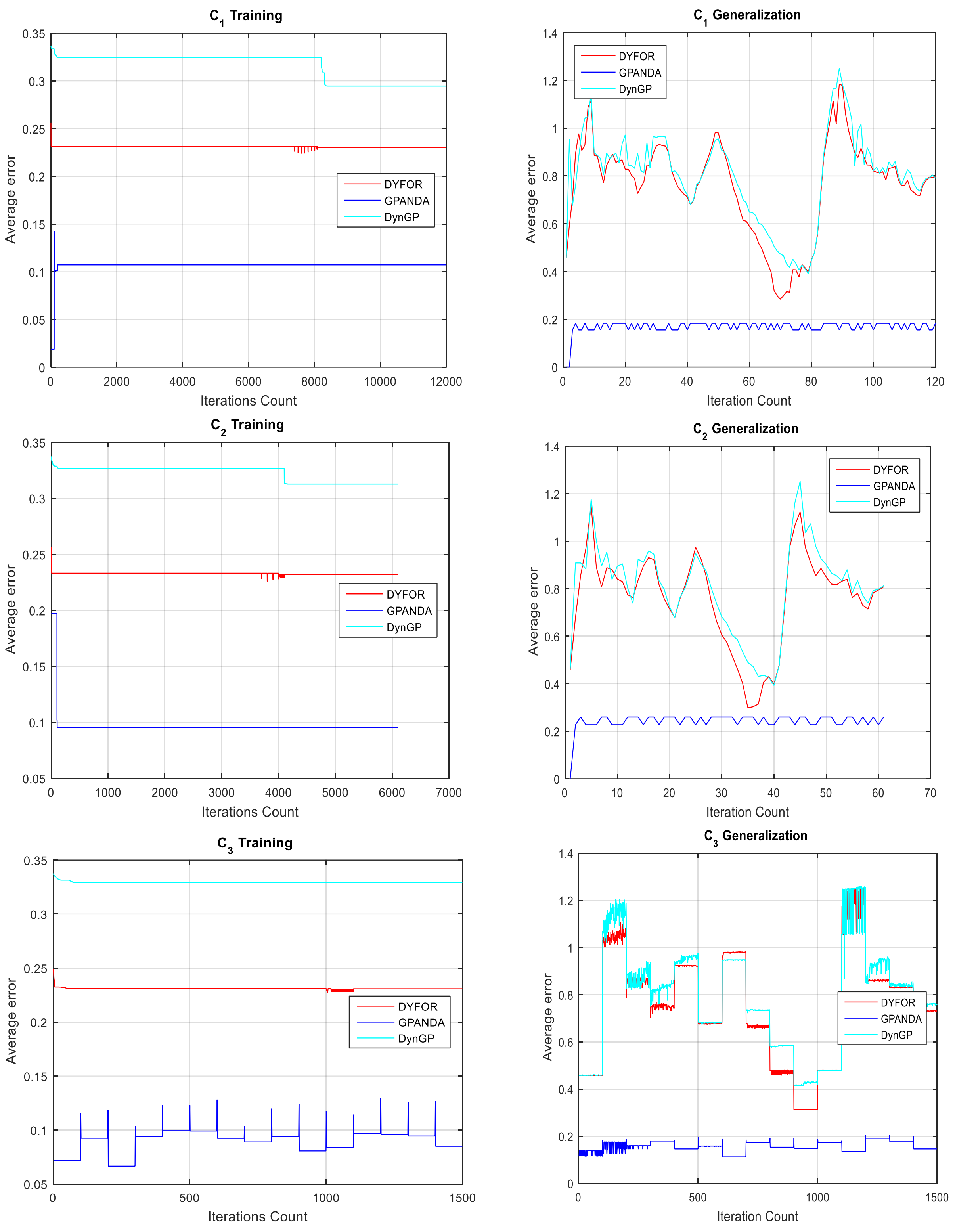


Figure 7.26: Training and Generalization E_{MS} for Abrupt, Scenario C_1 - C_3

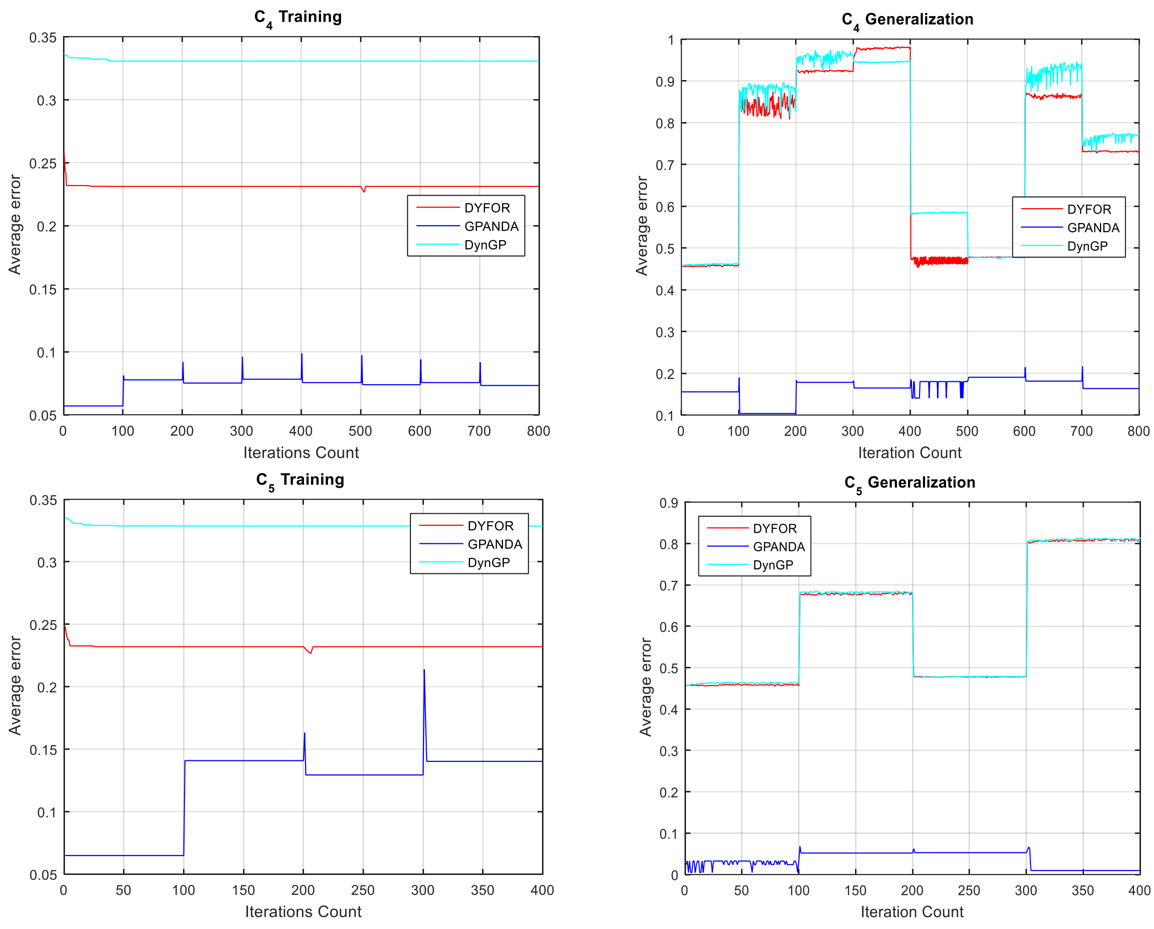


Figure 7.27: Training and Generalization E_{MS} for Abrupt, Scenario C_4 - C_5

The following can be observed for the results presented in Table 7.32:

- GPANDA outperforms all algorithms in all cases in Scenario C for both training and generalization.
- DyFor GP outperforms DynGP for generalization in all cases in Scenario C except for Scenario C_5.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario C_5.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.33 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario C.

Table 7.32: Results of Predictive Performance for Abrupt Environment Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.3154	0.8009	0.3223	0.9348	0.3295	0.7872	0.3308	0.7482	0.3288	0.6079	0.3253	0.7758
GPANDA	**0.1064**	**0.1689**	**0.0970**	**0.2425**	**0.0893**	**0.1558**	**0.0734**	**0.1649**	**0.1204**	**0.0359**	**0.0973**	**0.1536**
DyFor GP	0.2307	0.7599	0.2327	0.7573	0.2310	0.7474	0.2313	0.7174	0.2321	0.6052	0.2315	0.7174

As observed in Scenario B, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for both training and generalization. However, there is no statistically significant differences in performance between DyFor GP and DynGP with regards to performance for generalization in Scenario C_5 .

Table 7.33: Algorithm Ranking for Abrupt Dataset Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	2.5	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2.5	2	2

Discussion

Since the Abrupt dataset is generated from different data generating processes, previously learned information may not be very useful after an environment change. Thus, the usefulness of previously learned information is reduced in this problem set.

GPANDA outperforms all algorithms in all cases in Scenario C for both training and generalization. This superior performance of GPANDA suggests the capability of the algorithm to adapt to environments with abrupt concept drift occurring. As already mentioned, the superior performance of GPANDA can be attributed to the algorithm's ability to detect a changing environment and adapts a predictive model accordingly. DyFor GP outperforms DynGP on training, however, its generalization performance deteriorates as the spatial severity increases.

As already mentioned, low abrupt changes in Scenario $C_1 - C_4$, presents two concepts on each sliding window. However, GPANDA exhibits adaptive traits to changes in the environment with concept drift occurring to yield outstanding performance throughout the entire algorithm's progression, evident with results presented in Table 7.34.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms are given enough iterations to induce an optimal predictive model, even in abruptly changing environments, before a change in the environment occurs. Consequently, failure to adapt or induce an optimal predictive model may not be attributed to the temporal severity property of the problem set under consideration but of the algorithm.

Figure 7.28 - 7.29 are graphical illustrations as observed under Scenario D for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As illustrated in Figure 7.28 - 7.29, all algorithms show similar traits as observed under Scenario C: DyFor GP outperforms DynGP on training whereas the performance of DyFor GP and DynGP for generalization is the same. For some instances along with the iteration's progression, DyFor GP outperforms GPANDA for generalization in Scenario D_3 .

GPANDA exhibits superior performance and outperforms all other algorithms for both training and generalization. As illustrated in Figure 7.28 - 7.29, the significant error peaks are evident in the GPANDA error profile for Scenario $D_4 - D_5$. As already mentioned, the significant error peaks indicate changes in the environment which result in deterioration of the predictive performance of other algorithms under study.

Table 7.34 presents the obtained results for the Abrupt dataset, Scenario D. To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario C, similar traits are also observed in the results presented in Table 7.34:

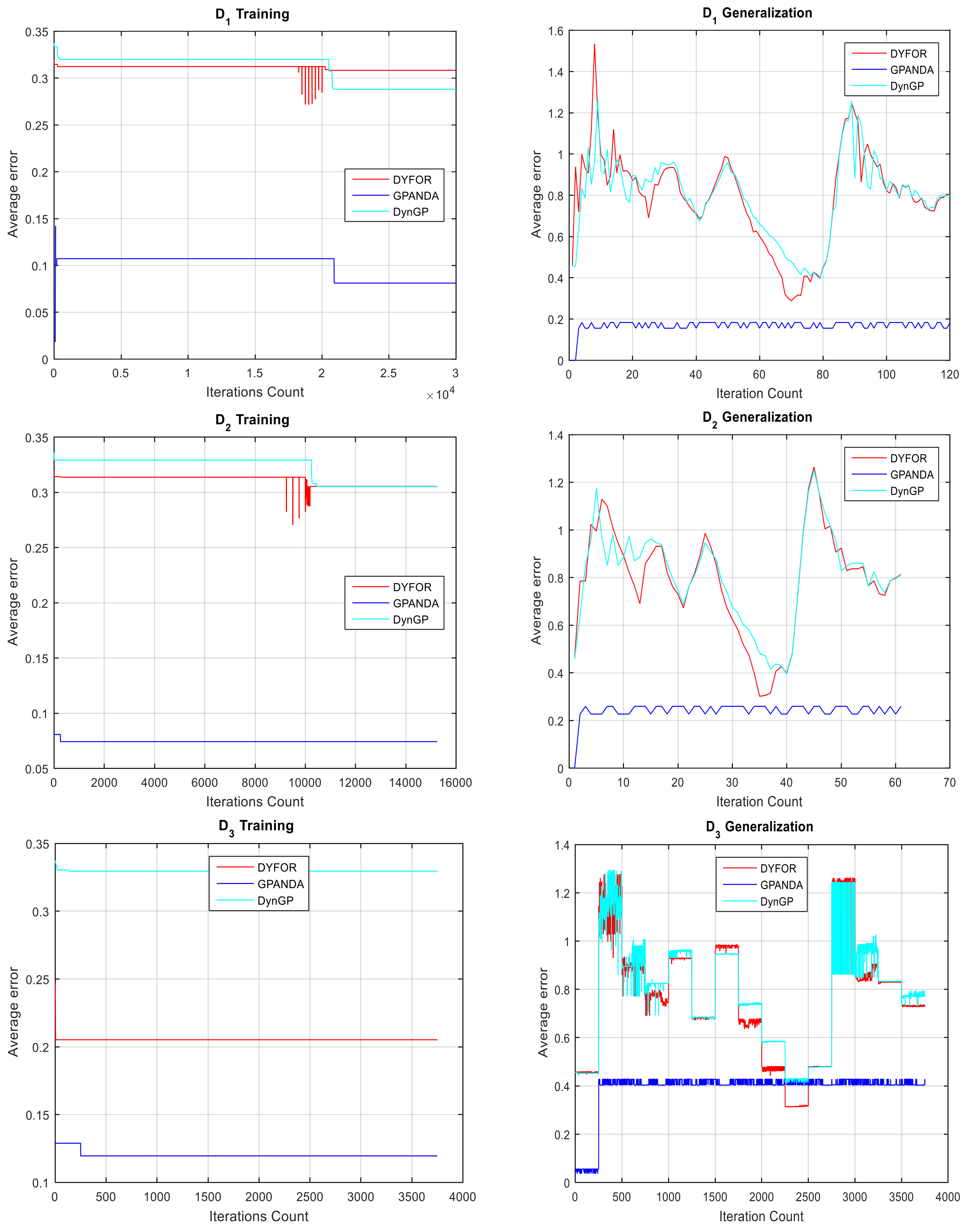


Figure 7.28: Training and Generalization E_{MS} for Abrupt, Scenario $D_1 - D_3$

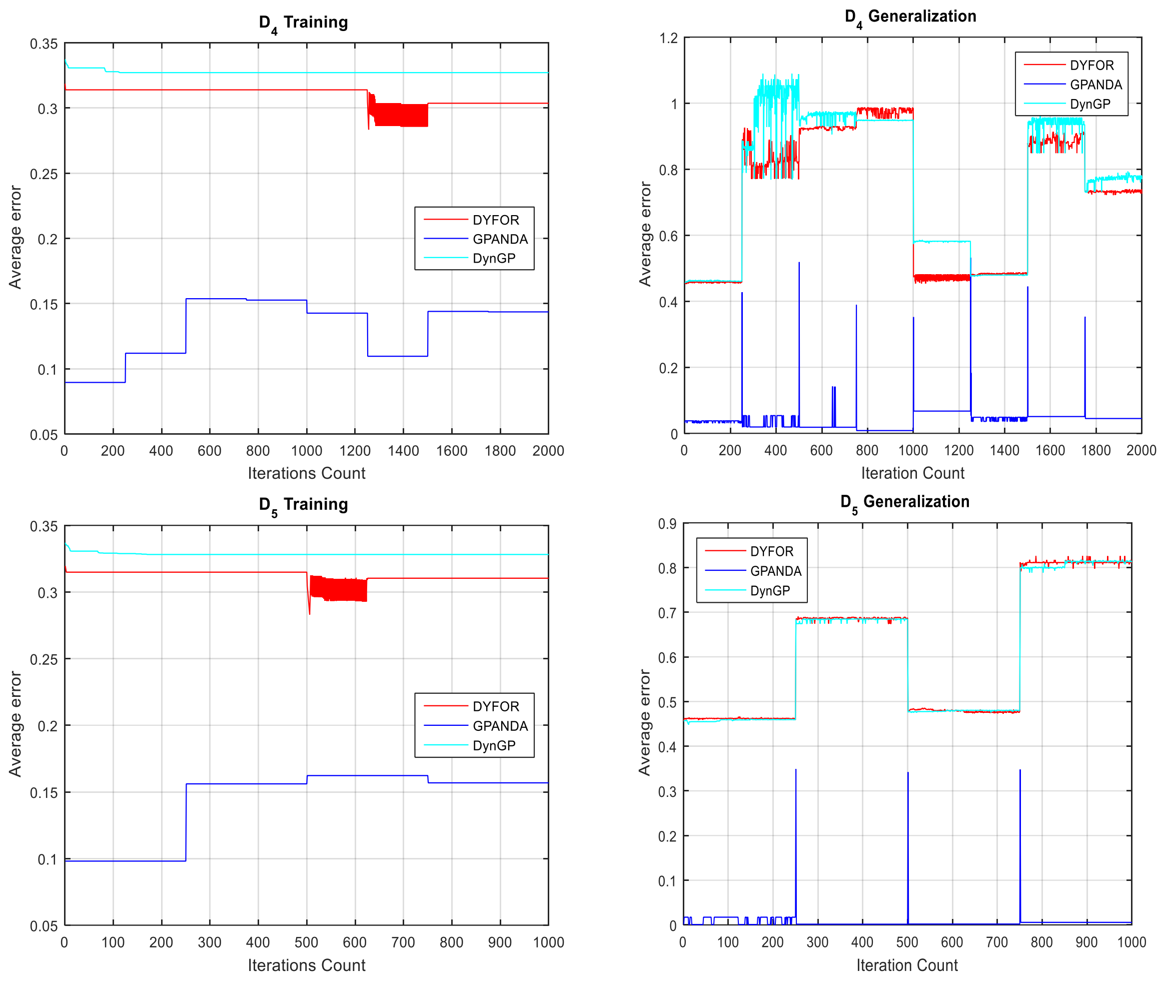


Figure 7.29: Training and Generalization E_{MS} for Abrupt, Scenario $D_4 - D_5$

- GPANDA outperforms all algorithms in all cases in Scenario D for both training and generalization.
- DyFor GP outperforms DynGP for generalization in all cases in Scenario D except for Scenario D_5.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario D_5.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.35 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario D.

As observed in Scenario C, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$.

Table 7.34: Results of Predictive Performance for Abrupt Environment Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.3275	0.8764	0.3230	0.7925	0.3296	0.7801	0.3275	0.7658	0.3284	0.6071	0.3272	0.7643
GPANDA	**0.0946**	**0.1173**	**0.0744**	**0.2745**	**0.1201**	**0.3900**	**0.1310**	**0.0395**	**0.1446**	**0.0060**	**0.1129**	**0.1654**
DyFor GP	0.3111	0.7889	0.3108	0.7838	0.2052	0.7580	0.3091	0.7195	0.3116	0.6095	0.2895	0.7319

DynGP is categorized as having the least effective performance for both training and generalization. However, there is no statistically significant differences in performance between DyFor GP and DynGP with regards to performance on $E_{MS}G$ in Scenario D_5 .

Table 7.35: Algorithm Ranking for Abrupt Dataset Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	2.5	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2.5	2	2

Discussion

GPANDA outperforms all algorithms in all cases in Scenario D for both training and generalization. As already mentioned, the superior performance of GPANDA can be attributed to the algorithm's ability to detect and adapt a predictive model. The generalization performance of DyFor GP deteriorates as the spatial severity increases.

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.36 presents the average $E_{MS}G$ and average computational time ($\bar{t}$) in minutes for Scenario A-D. As observed in Recurrent dataset, DynGP obtains the least average computational time in all scenarios under consideration in the Abrupt dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance.

Table 7.36: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for Abrupt Dataset - Scenario A-D

	A		B		C		D		Average	
Algorithm	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	0.7780	**0.002**	0.7432	**0.007**	0.7758	**0.014**	0.7643	**0.037**	0.7653	**0.015**
GPANDA	**0.1571**	0.701	**0.1888**	2.361	**0.1536**	4.977	**0.1654**	12.26	**0.1662**	5.074
DyFor GP	0.7128	0.147	0.7152	0.518	0.7174	0.994	0.7319	2.452	0.7193	1.027

Table 7.37 presents the obtained results for Scenario A - D. GPANDA is ranked as the overall winner whereas DynGP is categorized as having the least effective performance.

Table 7.37: Overall Algorithm Ranking for Abrupt Dataset

Algorithm	A	B	C	D	Average
DynGP	3	3	3	3	3
GPANDA	1	1	1	1	1
DyFor GP	2	2	2	2	2

7.3.4 Random Variation Dataset

The algorithms under study traverse the complete dataset for every scenario under consideration.

Scenario A - Abrupt Variation

Figure 7.30 is a graphical illustration as observed under Scenario $A_1 - A_3$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time. As illustrated in Figure 7.30, adaptation to a change in the environment becomes less difficult as spatial severity increase from A_1 to A_3 .

As illustrated in Figure 7.30, the performance of DynGP and DyFor GP is the same for generalization in Scenario A_1 and A_3. DyFor GP outperforms GPANDA in Scenario A_1 on training. Along with the iteration's progression, the performance of GPANDA on training is reduced in Scenario A_3 to yield the same performance to DyFor GP.

It is evident from Figure 7.30 that generalization error profiles for both DynGP and DyFor GP are generally higher compared to GPANDA. Higher error peaks can attributed to the changes that brought two or more decision boundaries into the sliding analysis window at each analysis window slide.

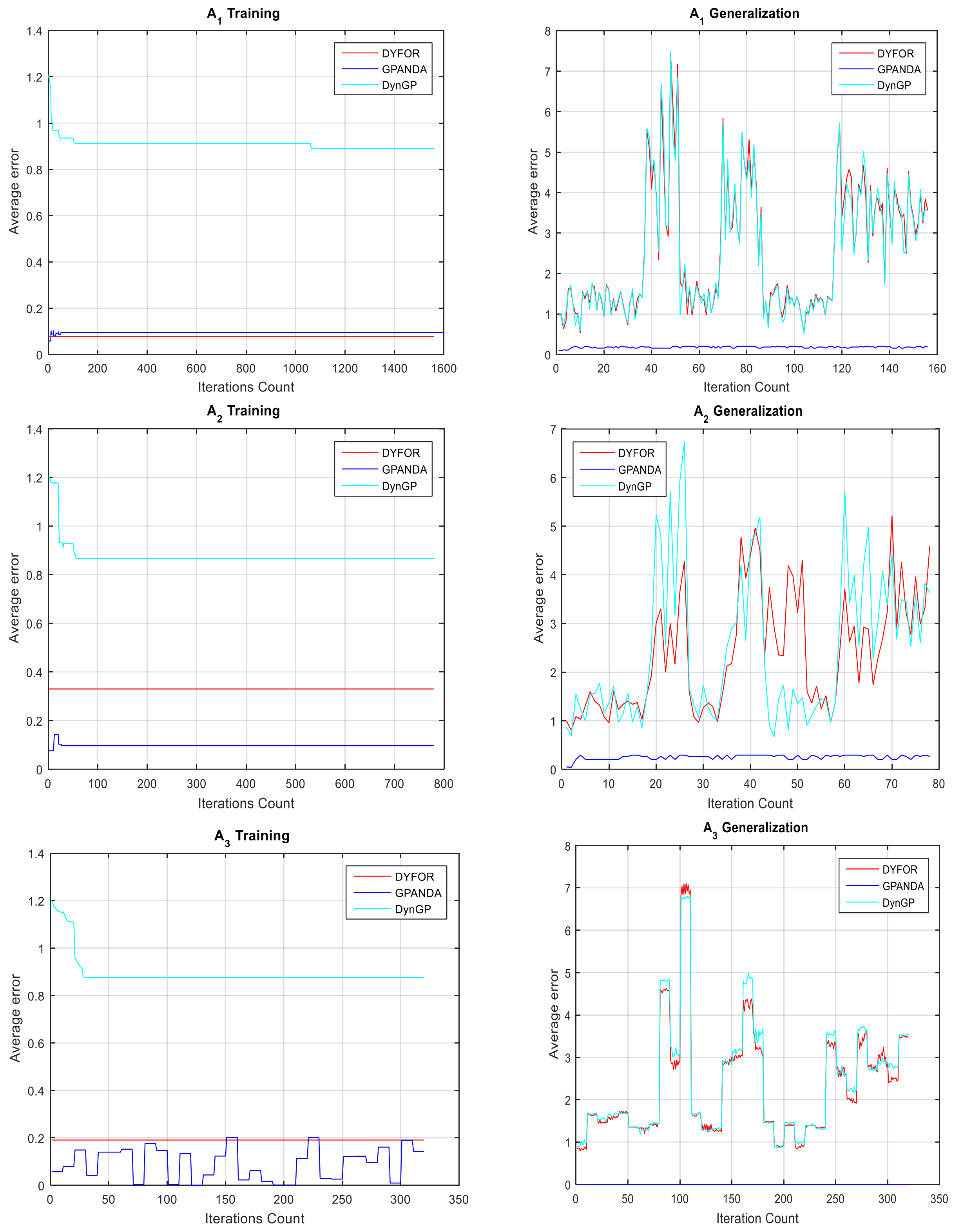


Figure 7.30: Training and Generalization E_{MS} for Random, Scenario $A_1 - A_3$

GPANDA detects a change in the environment and responds by adapting the predictive model, suggested by the lowering of the error profile after a change is detected as illustrated in Figure 7.30. As such, GPANDA exhibits superior performance, consequently, outperforms all algorithms by a greater margin in Scenario A for generalization.

Table 7.38 presents the obtained results for the Random dataset. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.38:

- GPANDA outperforms all algorithms in all cases in Scenario A for both training and generalization
- DyFor GP outperforms DynGP for generalization in Scenario A except for Scenario A_1.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario A_1.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.38: Results of Predictive Performance for Random Environment Scenario A

Algorithm	A_1		A_2		A_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.9090	2.5164	0.8777	2.4615	0.8939	2.3885	0.8935	2.4554
GPANDA	**0.0946**	**0.1811**	**0.0972**	**0.2488**	**0.1098**	**0.3531**	**0.1005**	**0.2610**
DyFor GP	0.0782	2.5473	0.3300	2.3983	0.1908	2.3085	0.1996	2.4180

The algorithms under study are classified according to training and generalization performance. Table 7.39 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario A. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ whereas DynGP exhibits the worst performance for both training and generalization.

Table 7.39: Algorithm Ranking for Random Dataset Scenario A

Algorithm	A_1		A_2		A_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2.5	3	3	3	3	3	2.8
GPANDA	1	1	1	1	1	1	1	1
DyFor GP	2	2.5	2	2	2	2	2	2.2

Discussion

The worst performance exhibited by DynGP can be attributed to the failure of the algorithm to adapt to a changing environment with concept drift occurring.

GPANDA outperforms all algorithms in all cases in Scenario A. The superior performance exhibited by GPANDA suggests the capability of the algorithm to adapt to a randomly changing environment.

DyFor GP outperforms DynGP which suggests better adaptive traits of DyFor GP to a randomly changing environment. However, the inferior performance of DyFor GP to GPANDA suggests the reduced performance of the predictive model which entails that the predictive model was evolved from a dataset with some irrelevant data points.

The progressive changes happening in Scenario $A_1 - A_2$ increase the complexity of the predictive task since the sliding window at any given instance was made up of data points from two different data generating processes.

As observed in Scenario A, similar traits are also observed in Scenario B, the performance of DynGP and DyFor GP is the same for generalization. As illustrated in Figure 7.30, the generalization error profiles for both DynGP and DyFor GP are generally higher compared to GPANDA.

Scenario B - Quasi-Abrupt

Scenario B simulates less frequent changes, having a frequency of severity of 50 iterations. Figure 7.31 is a graphical illustration as observed under Scenario $B_1 - B_3$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

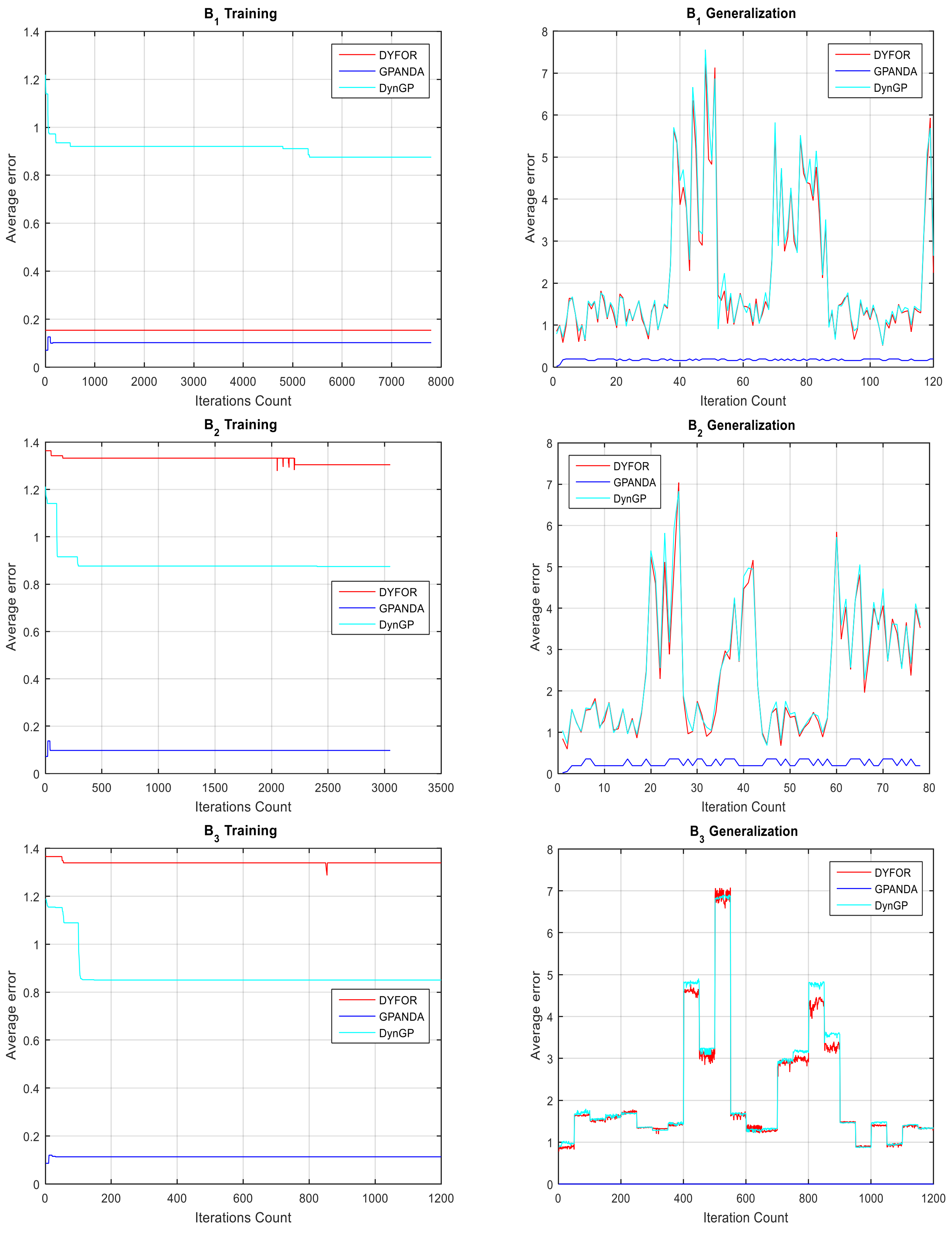


Figure 7.31: Training and Generalization E_{MS} for Random, Scenario B_1 - B_3

GPANDA detects a change in the environment and responds by adapting the induced model, which is evident by the lowering of the error profile after a change is detected. Therefore, GPANDA exhibits superior performance, consequently, outperforms all algorithms by a greater margin in Scenario B for both training and generalization cases.

Table 7.40 presents the obtained results for the Random dataset, Scenario B. To ascertain the significance of the results obtained for Scenario B, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.40:

- GPANDA outperforms all algorithms in all cases in Scenario B for both training and generalization.
- DyFor GP outperforms DynGP on $E_{MS}G$ in Scenario B except for Scenario B_1 and B_3.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario B_1 and B_3.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.40: Results of Predictive Performance for Random Environment Scenario B

Algorithm	B_1		B_2		B_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.9087	2.5344	0.8846	2.4981	0.8680	2.4067	0.8871	2.4797
GPANDA	**0.1021**	**0.1744**	**0.0978**	**0.2489**	**0.1130**	**0.3756**	**0.1043**	**0.2663**
DyFor GP	0.1538	2.4506	1.3254	2.4030	1.3405	2.3245	0.9399	2.3927

The algorithms under study are classified according to training and generalization performance. Table 7.41 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario B. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ whereas DynGP is ranked as having the least effective performance for generalization whereas DyFor on training.

Discussion

Table 7.41: Algorithm Ranking for Random Dataset Scenario B

Algorithm	B_1		B_2		B_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2.5	2	3	2	2.5	2.3	2.7
GPANDA	1	1	1	1	1	1	1	1
DyFor GP	2	2.5	3	2	3	2.5	2.7	2.3

As observed in Scenario A, similar traits are also observed in Scenario B. The superior performance of GPANDA suggests the capability of the algorithm to adapt to changes in the environment with concept drift occurring.

As already explained, the inferior performance of DyFor GP to GPANDA for generalization implies that the predictive model induced by DyFor GP contains irrelevant data points (data points from the past data generating processes).

Scenario C - Quasi-Progressive Variation

Scenario C simulates less frequent changes, having a frequency of severity of 100 iterations. Figure 7.32 is a graphical illustration as observed under Scenario $C_1 - C_3$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario B, Scenario C exhibits similar traits: GPANDA outperforms all other algorithms in Scenario C for both training and generalization. As illustrated in Figure 7.32, DyFor GP outperforms DynGP in all cases in Scenario C on training. However, its generalization is reduced to yield performance the same to DynGP in all cases in Scenario C.

Table 7.42 presents the obtained results for the Random dataset, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario B, the following are also observed for the results presented in Table 7.42:

- GPANDA outperforms all algorithms in all cases in Scenario C for both training and generalization.

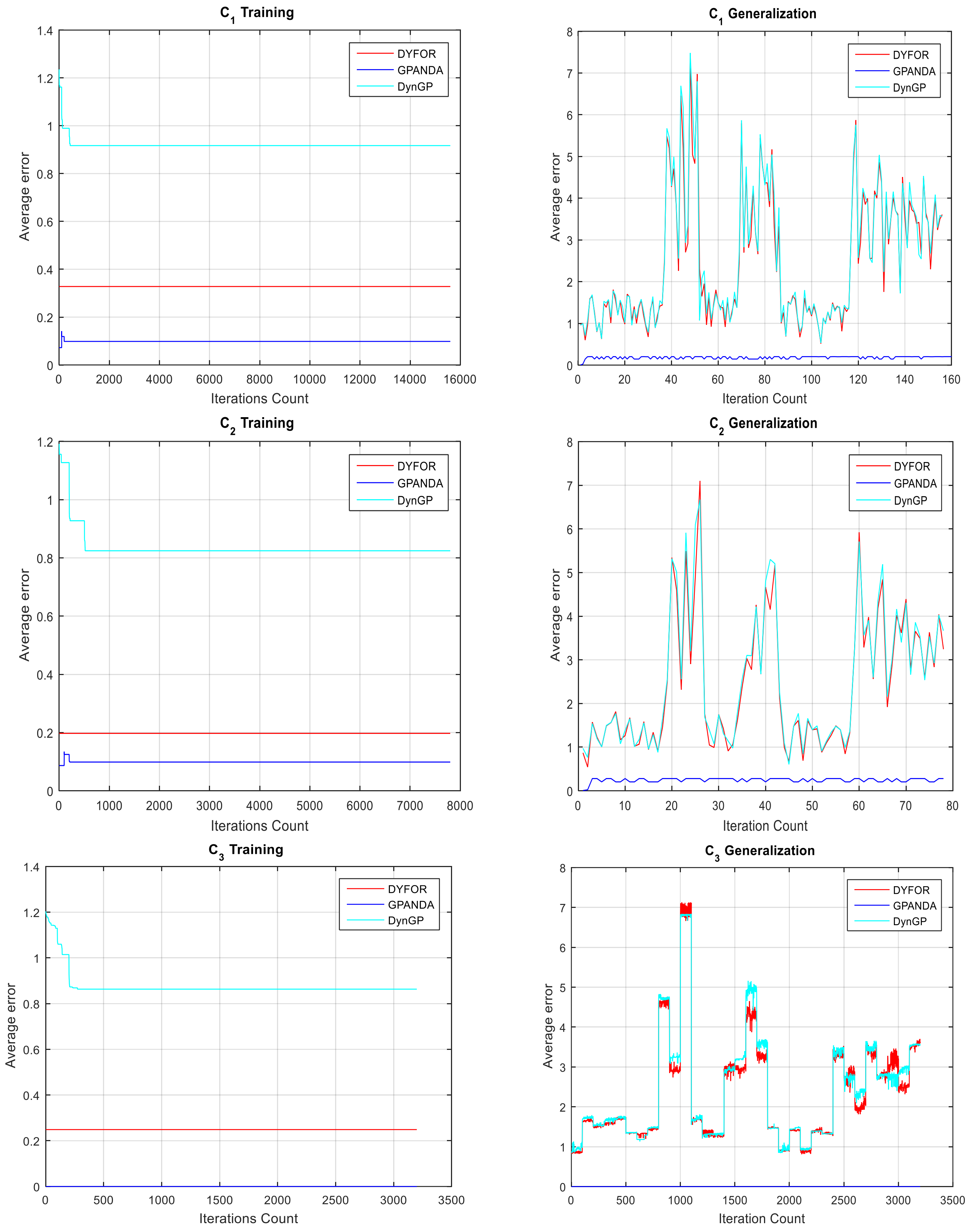


Figure 7.32: Training and Generalization E_{MS} for Random, Scenario $C_1 - C_3$

- DyFor GP outperforms DynGP in Scenario C.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario C_1 and C_3.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.43 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario C. As observed under Scenario B, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ whereas DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.42: Results of Predictive Performance for Random Environment Scenario C

Algorithm	C_1		C_2		C_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.9201	2.5519	0.8365	2.5105	0.8779	2.4034	0.8781	2.4886
GPANDA	**0.0983**	**0.1788**	**0.0988**	**0.2451**	**0.1140**	**0.3538**	**0.1037**	**0.2592**
DyFor GP	0.3279	2.4726	0.1971	2.4196	0.2489	2.3300	0.2579	2.4074

Table 7.43: Algorithm Ranking for Random Dataset Scenario C

Algorithm	C_1		C_2		C_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2.5	3	3	3	2.5	3	2.7
GPANDA	1	1	1	1	1	1	1	1
DyFor GP	2	2.5	2	2	2	2.5	2	2.3

Discussion

As observed under Scenario B, similar traits are also observed in Scenario C: GPANDA outperforms all algorithms in all cases in Scenario C. The superior performance of GPANDA suggests the capability of the algorithm to adapt to environments with concept drift occurring.

The inferior performance of DyFor GP and DynGP for generalization to that of GPANDA suggests that the models are evolved with some irrelevant data points from the past data generating processes.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms under study are given enough iterations before an environment change to evolve an optimal predictive model. Consequently, failure to adapt or to evolve an optimal solution may not be attributed to the temporal severity property of the problem set under consideration, but the algorithm.

Figure 7.33 is a graphical illustration as observed in Scenarios $D_1 - D_3$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time. All algorithms showed similar traits as observed in Scenario C: the performance of DynGP and DyFor GP is the same for generalization.

As illustrated in Figure 7.33, GPANDA exhibits superior performance for generalization, consequently, outperforms all algorithms in Scenario D for generalization.

However, the training performance of GPANDA is reduced in Scenario $D_1 - D_2$. The performance of DyFor GP is consistent in all training cases and outperforms GPANDA in Scenario $D_1 - D_2$.

Table 7.44 presents the obtained results for the Random dataset, Scenario D. To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.44:

- GPANDA outperforms all algorithms in all cases in Scenario D for generalization.
- DyFor GP outperforms DynGP in Scenario D.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.

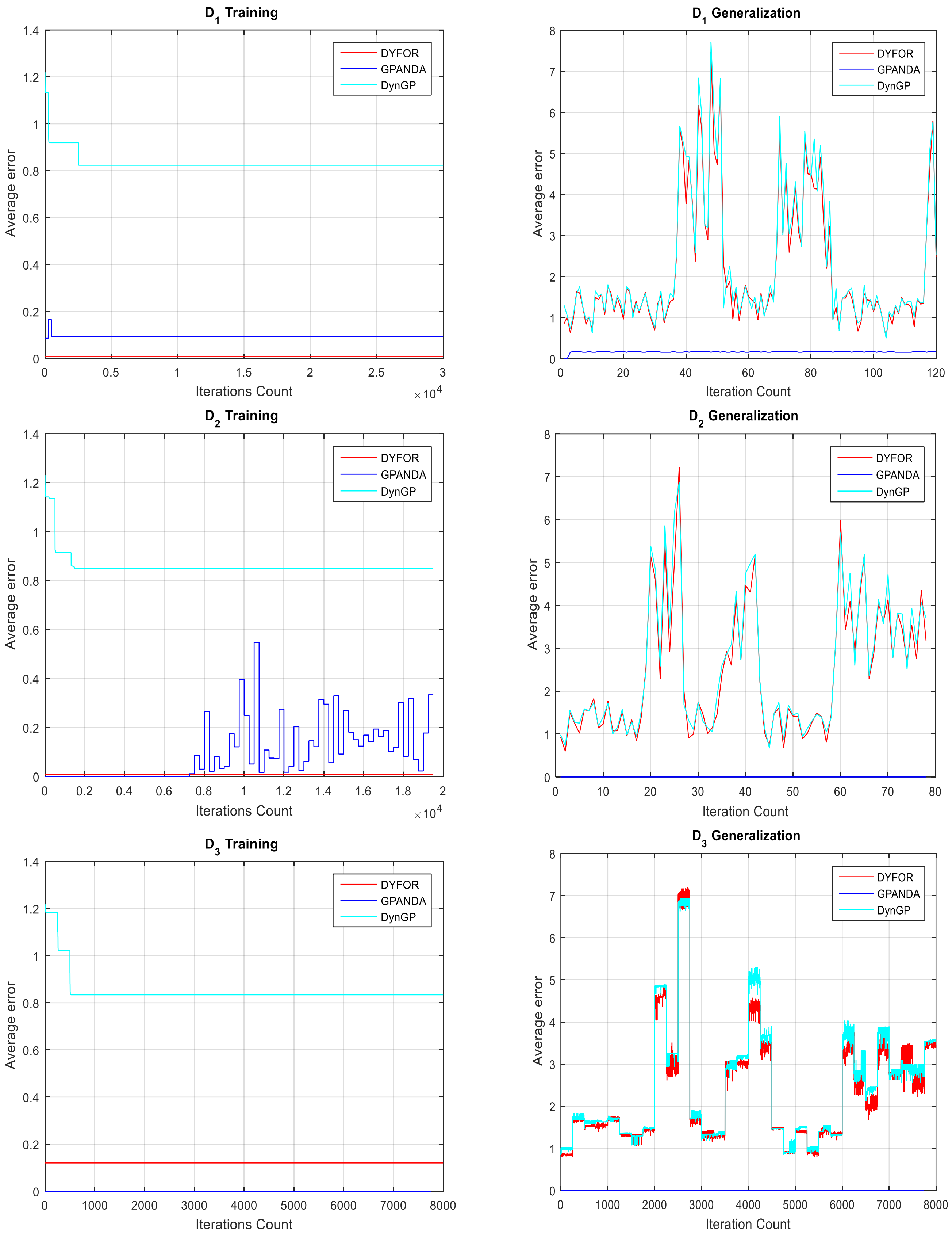


Figure 7.33: Training and Generalization E_{MS} for Random, Scenario $D_1 - D_3$

- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.45 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ in Scenario D. DyFor GP is ranked as the overall winner on $E_{MS}T$ whereas GPANDA is ranked as the overall winner on $E_{MS}G$. DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.44: Results of Predictive Performance for Random Environment Scenario D

Algorithm	D_1		D_2		D_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.8334	2.2817	0.8602	2.5563	0.8509	2.4572	0.8481	2.4310
GPANDA	**0.0934**	**0.1658**	**0.0957**	**1.34e-26**	**5.91e-25**	**1.59e-30**	**0.0630**	**0.0552**
DyFor GP	0.0087	2.4843	0.0072	2.4395	0.1202	2.3434	0.0453	2.4224

Table 7.45: Algorithm Ranking for Random Dataset Scenario D

Algorithm	D_1		D_2		D_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2	3	3	3	3	3	2.7
GPANDA	2	1	2	1	1	1	1.7	1
DyFor GP	1	3	1	2	2	2	1.3	2.3

Discussion

GPANDA outperforms all other algorithms in all cases in Scenario D for generalization. The superior performance of GPANDA suggests the effectiveness of a piecewise predictive approach to a changing environment with concept drift occurring.

The inferior performance of DyFor GP to GPANDA implies that the model is evolved including some irrelevant data points that reduced the predictive performance of the model. The reduced performance of DynGP algorithm can be attributed to conflicting decision boundaries within the sliding window of analysis.

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.46 presents the average $E_{MS}G$ and average computational time ($\bar{t}$) in minutes for Scenario A-D. As observed in the Abrupt dataset, DynGP obtains the least average computational time in all scenarios under consideration in the Random dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance. Table 7.47 presents the obtained results for Scenario A

Table 7.46: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for Random Dataset - Scenario A-D

	A		B		C		D		Average	
Algorithm	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	2.4554	**0.021**	2.4797	**0.059**	2.4886	**0.127**	2.4317	**0.312**	2.4638	**0.1297**
GPANDA	**0.2610**	3.467	**0.2663**	11.23	**0.2592**	20.81	**0.0552**	71.29	**0.2104**	26.6992
DyFor GP	2.4180	1.082	2.3927	3.51	2.4074	6.818	2.4224	22.28	2.4101	8.4225

- D. GPANDA is ranked as the overall winner while DynGP is categorized as having the least effective performance.

Table 7.47: Overall Algorithm Ranking for Random Dataset

Algorithm	A	B	C	D	Average
DynGP	2.8	2.7	2.7	2.7	2.7
GPANDA	1	1	1	1	1
DyFor GP	2.2	2.3	2.3	2.3	2.3

7.3.5 Trend Variation Dataset

The algorithms under study traverse the complete dataset for every scenario under consideration.

Scenario A - Abrupt Variation

The frequency of change in Scenario A is very high which is 25 iterations. Figure 7.34 are graphical illustrations as observed under Scenario $A_1 - A_4$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

GPANDA outperforms all other algorithms in all cases in Scenario A for both training and generalization whereas DyFor GP outperforms DynGP.

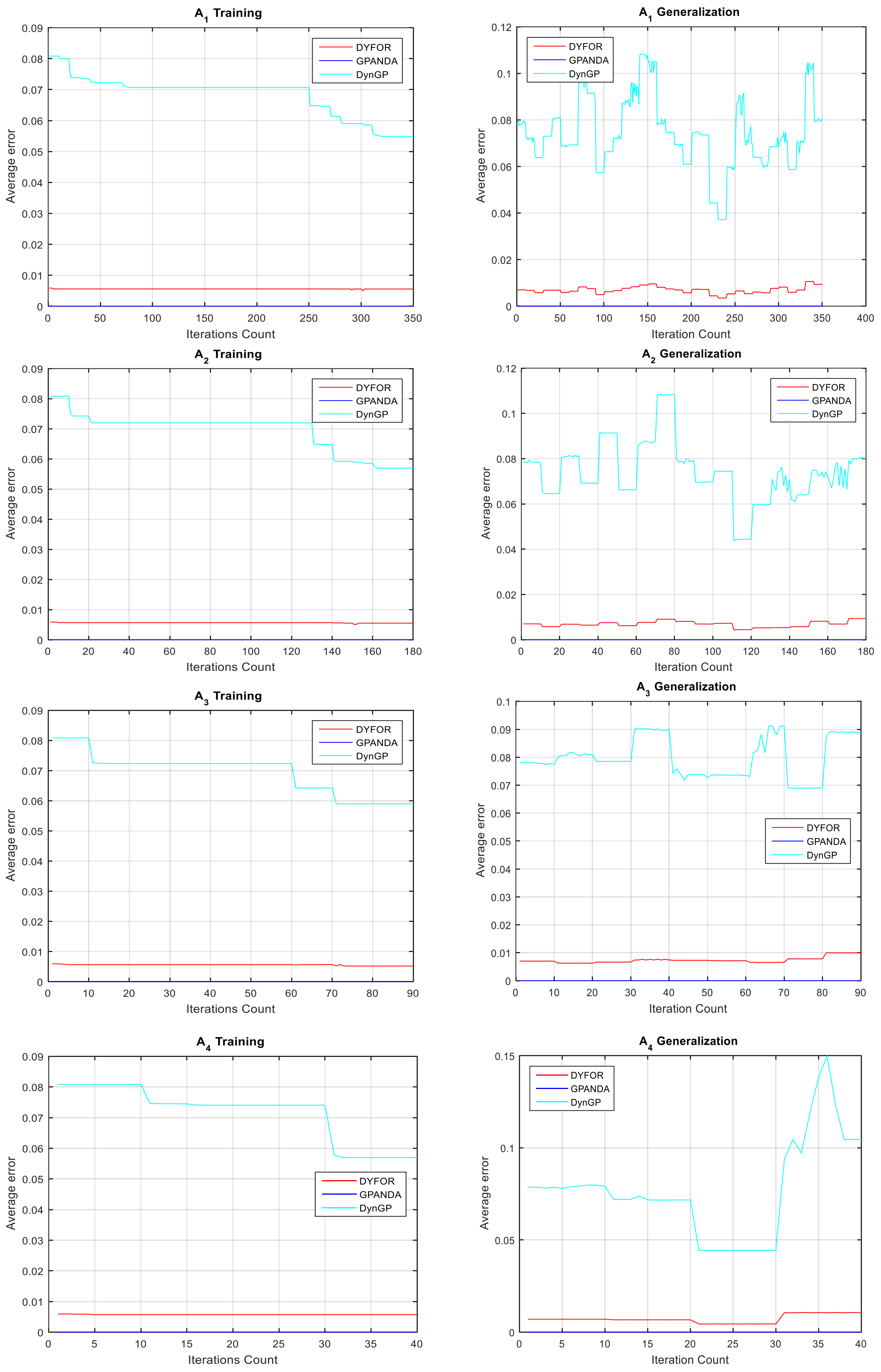


Figure 7.34: Training and Generalization E_{MS} for Trend, Scenario $A_1 - A_4$

As illustrated in Figure 7.34, the staircase/up-down patterns on DynGP error profile for both training and generalization can be attributed to changes that tend to bring two or more decision boundaries into the sliding window at each slide. Thus, the predictive model is induced using more of the data points from the past data generating processes.

However, both GPANDA and DyFor GP maintain a steady performance throughout the iteration's progression. Thus, GPANDA and DyFor GP detect changes in the environment and respond by adapting the predictive model, suggested by lowering of the error profiles.

Table 7.48 presents the obtained results for the Trend dataset, Scenario A. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.48:

- GPANDA outperforms all algorithms in all cases in Scenario A for both training and generalization.
- DyFor GP outperforms DynGP.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
  item The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.48: Results of Predictive Performance for Trend Environment Scenario A

Algorithm	A_1		A_2		A_3		A_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0681	0.0740	0.0691	0.0740	0.0694	0.0798	0.0715	0.0773	0.0695	0.0762
GPANDA	**2.65e-5**	**1.33e-33**	**2.21e-5**	**1.01e-29**	**2.24e-5**	**1.01e-34**	**2.50e-28**	**1.73e-28**	**1.77e-05**	**4.57e-29**
DyFor GP	0.0056	0.0069	0.0057	0.0069	0.0056	0.0074	0.0058	0.0072	0.0056	0.0071

Table 7.49 presents the obtained results for Scenario A. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$ whereas DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.49: Algorithm Ranking for Trend dataset Scenario A

Algorithm	A_1		A_2		A_3		A_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2

Discussion

As observed in the previous experiments, GPANDA outperforms all other algorithms in all cases in Scenario A. As already mentioned, the outstanding performance of GPANDA suggests the capability of the algorithm to adapt to changes happening to real-world, historical US Treasury bill contracts.

As already mentioned, the performance gain exhibited by the DyFor GP can be attributed to the ability of the algorithm to optimize the dataset presented to GP by eliminating irrelevant data points present in the analysis window.

The worst performance exhibited by DyFor GP suggests the reduced effectiveness of the dynamic parameter tuning technique to adapt to time series dataset with very few independent variables.

The progressive changes happening in Scenario A_1 - A_4 increases the complexity of the predictive task, however, GPANDA and DyFor GP exhibit slight to no performance deterioration in all cases in Scenario A. This implies that the data generating process follows a trend variation with gradual changes that makes easier for the algorithms under study to evolve and adapt predictive models.

Scenario B - Quasi-Abrupt

Figure 7.35 are graphical illustrations as observed under Scenario $B_1 - B_4$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario A, similar traits are also observed in Scenario B, DyFor GP outperforms DynGP in all cases in Scenario B.

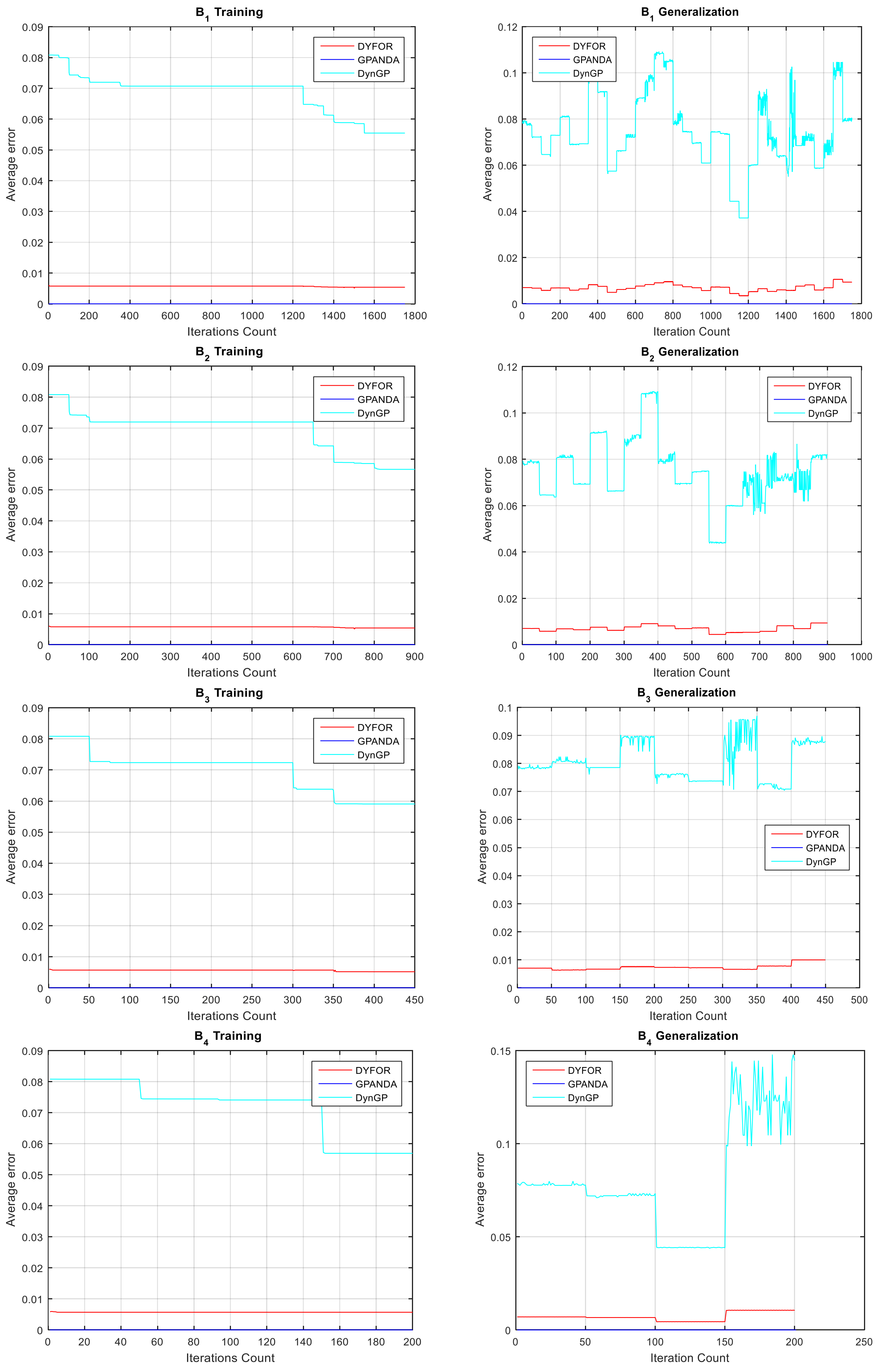


Figure 7.35: Training and Generalization E_{MS} for Trend, Scenario $B_1 - B_4$

As illustrated in Figure 7.35, GPANDA detects the change in the environment with concept drifts occurring and adapted the predictive model accordingly which is suggested by better predictive performance. As such, GPANDA outperforms all other algorithms in Scenario B for both training and generalization.

Table 7.50 presents the obtained results for the Trend dataset, Scenario B. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. Scenario B followed the similar traits of Scenario A:

- GPANDA outperforms all algorithms in all cases in Scenario A for both training and generalization.
- DyFor GP outperforms DynGP.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.51 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario B. As observed in Scenario A, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.50: Results of Predictive Performance for Trend Environment Scenario B

Algorithm	B_1		B_2		B_3		B_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0682	0.0745	0.0690	0.0744	0.0694	0.0805	0.0716	0.0793	0.0695	0.0771
GPANDA	**2.14e-5**	**9.03e-7**	**2.44e-5**	**2.86e-8**	**3.14e-5**	**2.85e-5**	**2.10e-5**	**5.37e-28**	**2.45e-5**	**8.35e-6**
DyFor GP	0.0057	0.0069	0.0057	0.0069	0.0056	0.0074	0.0057	0.0072	0.0056	0.0071

Table 7.51: Algorithm Ranking for Trend dataset Scenario B

Algorithm	B_1		B_2		B_3		B_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2

Discussion

As observed in Scenario A, similar traits are also observed in Scenario B: GPANDA outperforms all algorithms in all cases. As already explained, the superior performance of GPANDA can be attributed to adaptive traits of GPANDA to the changing environment with concept drift occurring.

DyFor GP significantly outperforms DynGP. The outstanding performance exhibited by DyFor GP suggests the capability of the algorithm to adapt a predictive model to a dataset that exhibits trend variation.

As already mentioned, the inferior performance exhibited by DynGP can be attributed to the dataset having few independent variables which made it difficult for DynGP through dynamic parameter tuning to adapt a predictive model to yield a model of high precision.

However, GPANDA using its underlying dynQPSO yields predictive models of a higher precision that outperforms all other algorithms under study. Also, GPANDA and DyFor GP are insensitive to change of spatial severity from Scenario $B_1 - B_4$. This implies that the data generating process follows a trend variation with gradual changes that makes it easier for these algorithms to adapt a predictive model with outstanding performance.

Scenario C - Quasi-Progressive Variation

Figure 7.36 are graphical illustrations as observed in Scenario C for the progression of $E_{MS}T$ and $E_{MS}G$ over time. As observed in Scenario B, GPANDA detect changes in the environment and adapts the predictive model yielding increased predictive performance and consequently, outperforms all other algorithms in Scenario C.

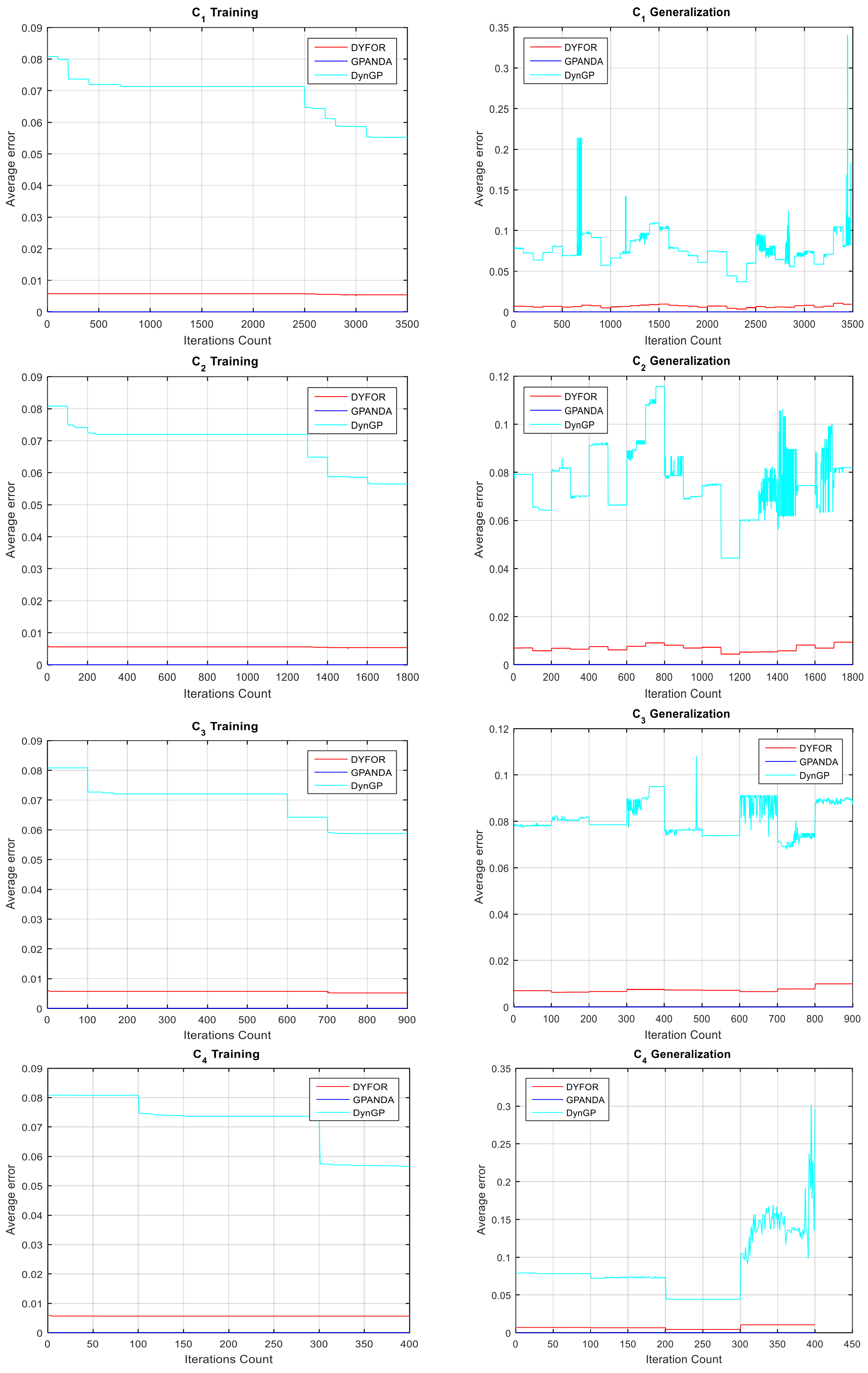


Figure 7.36: Training and Generalization E_{MS} for Trend, Scenario $C_1 - C_4$

As illustrated in Figure 7.36, DynGP exhibits poor adaptive traits compared to other algorithms and consequently, yields the worst performance for both training and generalization in all cases in Scenario C.

Table 7.52 presents the obtained results for the Trend dataset, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. Scenario C followed the similar traits of Scenario B:

- GPANDA outperforms all algorithms in all cases in Scenario A for both training and generalization.
- DyFor GP outperforms DynGP.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.53 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario C. As observed in Scenario B, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.52: Results of Predictive Performance for Trend Environment Scenario C

Algorithm	C_1		C_2		C_3		C_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0684	0.0963	0.0690	0.0761	0.0692	0.0808	0.0713	0.0851	0.0694	0.0845
GPANDA	**2.73e-5**	**1.96e-8**	**2.77e-5**	**4.53e-6**	**2.18e-5**	**4.53e-6**	**3.29e-5**	**2.97e-5**	**2.74e-5**	**9.69e-6**
DyFor GP	0.0057	0.0069	0.0055	0.0069	0.0056	0.0074	0.0060	0.0072	0.0057	0.0071

Discussion

As observed in Scenario B, similar traits are also observed in Scenario C: GPANDA

Table 7.53: Algorithm Ranking for Trend Dataset Scenario C

Algorithm	C_1		C_2		C_3		C_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2

outperforms all other algorithms in all cases. As already explained, the superior performance of GPANDA can be attributed to adaptive traits of the algorithm to the changing environment with concept drift occurring.

The outstanding performance exhibited by DyFor GP suggests the capability of the algorithm to adapt a predictive model to a dataset that exhibits trend variation.

As already mentioned, the inferior performance exhibited by DynGP can be attributed to the dataset having few independent variables which made it difficult to evolve a predictive model of high precision. An increase in temporal severity did not yield any performance gain on DynGP as expected, instead, the performance deteriorates especially in Scenario C_4.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms are given enough iterations before an environment change to induce an optimal predictive model even in abruptly changing environments. Consequently, failure to adapt or induce optimal solutions may not be attributed to the temporal severity property of the problem set under consideration but the algorithm.

Figure 7.37 are graphical illustrations as observed in Scenario D for the progression of $E_{MS}T$ and $E_{MS}G$ over time. As illustrated in Figure 7.37, algorithms under study show similar traits as observed in Scenario C: GPANDA exhibits superior performance, consequently, outperforms all algorithms in Scenario D for both training and generalization. DyFor GP outperforms DynGP in all cases in Scenario D for both training and generalization.

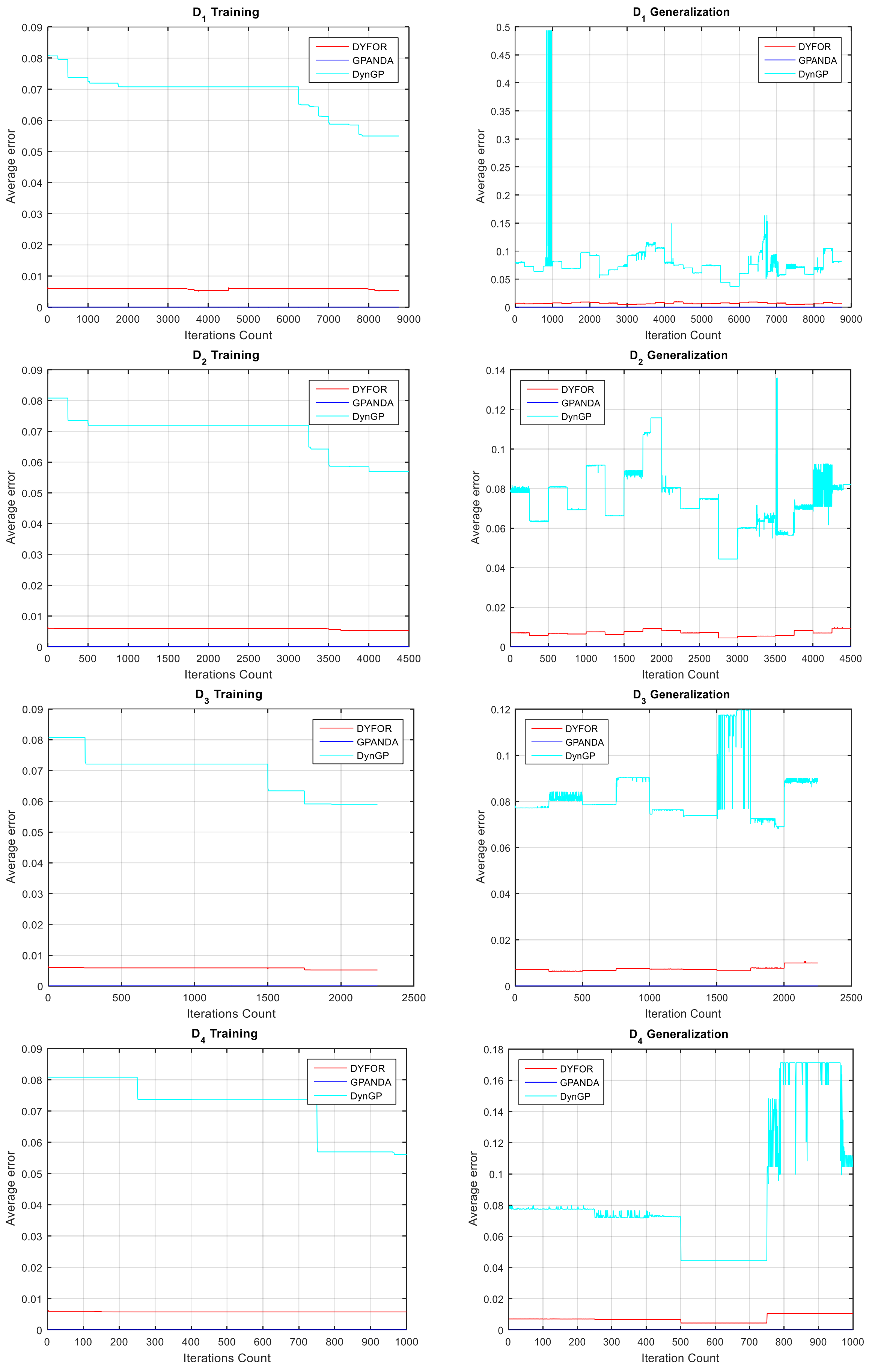


Figure 7.37: Training and Generalization E_{MS} for Trend, Scenario $D_1 - D_4$

To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario C, similar traits are also observed for the results presented in Table 7.54:

- GPANDA outperforms all algorithms in all cases in Scenario A for both training and generalization.
- DyFor GP outperforms DynGP.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.54: Results of Predictive Performance for Trend Environment Scenario D

Algorithm	D_1		D_2		D_3		D_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0681	0.0881	0.0690	0.0742	0.0692	0.0836	0.0690	0.0742	0.0688	0.0800
GPANDA	**2.10e-5**	**4.87e-8**	**2.10e-5**	**4.87e-8**	**1.97e-5**	**1.14e-7**	**2.52e-5**	**8.52e-8**	**2.17e-5**	**8.41e-8**
DyFor GP	0.0058	0.0069	0.0058	0.0069	0.0057	0.0074	0.0712	0.0871	0.0221	0.0270

The algorithms under study are classified according to training and generalization performance. Table 7.55 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario D. As observed in Scenario C, GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for both training and generalization.

Table 7.55: Algorithm Ranking for Trend Dataset Scenario D

Algorithm	D_1		D_2		D_3		D_4		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	3	3	3	3	3
GPANDA	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	2	2	2	2	2

Discussion

As observed in Scenario C, similar traits are also observed in Scenario D: GPANDA outperforms all algorithms in all cases. The outstanding performance of GPANDA suggests the capability of the algorithm to adapt to environments with trend variation.

As already mentioned, DyFor GP significantly outperforms DynGP. The increased performance in DyFor GP can be attributed to the relevant data points presented to the GP through an analysis window optimization technique.

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.56 presents the average $E_{MS}G$ and average computational time ($\bar{t}$) in minutes for Scenario A-D. As observed in the Random dataset, DynGP obtains the least average computational time in all scenarios under consideration in the Trend dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance.

Table 7.56: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for Trend Dataset - Scenario A-D

	A		*B*		*C*		*D*		Average	
Algorithm	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	0.0762	**0.013**	0.0771	**0.046**	0.0845	**0.084**	0.0800	**0.235**	0.0794	**0.0945**
GPANDA	**4.57e-29**	1.077	**8.35e-6**	2.751	**9.69e-6**	5.531	**8.41e-8**	11.29	**4.27e-6**	5.1622
DyFor GP	0.0071	0.405	0.0071	1.365	0.0071	2.281	0.0270	5.375	0.0120	2.3565

Table 7.57 presents the average obtained ranks on $E_{MS}G$ for Scenario A - D. GPANDA is ranked as the overall winner whereas DynGP is categorized as having the least effective performance.

Table 7.57: Overall Algorithm Ranking for Trend Dataset

Algorithm	*A*	*B*	*C*	*D*	Average
DynGP	3	3	3	3	3
GPANDA	1	1	1	1	1
DyFor GP	2	2	2	2	2

7.3.6 New South Wales Electricity Pricing Dataset

The algorithms under study traverse the complete dataset for every scenario under consideration.

Scenario A - Abrupt Variation

The frequency of change is the same for all cases in Scenario A which is 25 iterations. Figure 7.38 are graphical illustrations as observed under Scenario $A_1 - A_5$ for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As illustrated in Figure 7.38 - 7.39, the performance of DynGP and DyFor GP is the same in all cases in Scenario A. GPANDA detect changes in the environment with concept drifts occurring and adapts the predictive model which result in a steady error profile.

DynGP and DyFor GP exhibit an improving performance as the iterations progress. This performance improvement suggests the dataset follows either trend progression or the underlying changes are happening in a progressive manner which makes it easier for the algorithms to adapt the predictive model in the environment with concept drift occurring. GPANDA consistently maintains the obtained minimum as compared to other algorithms.

Table 7.58 presents the obtained results for the Electricity demand dataset, Scenario A. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.58:

- GPANDA outperforms all algorithms in all cases in Scenario A.
- DyFor GP outperforms DynGP.
- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario $A_1 - A_4$.

- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

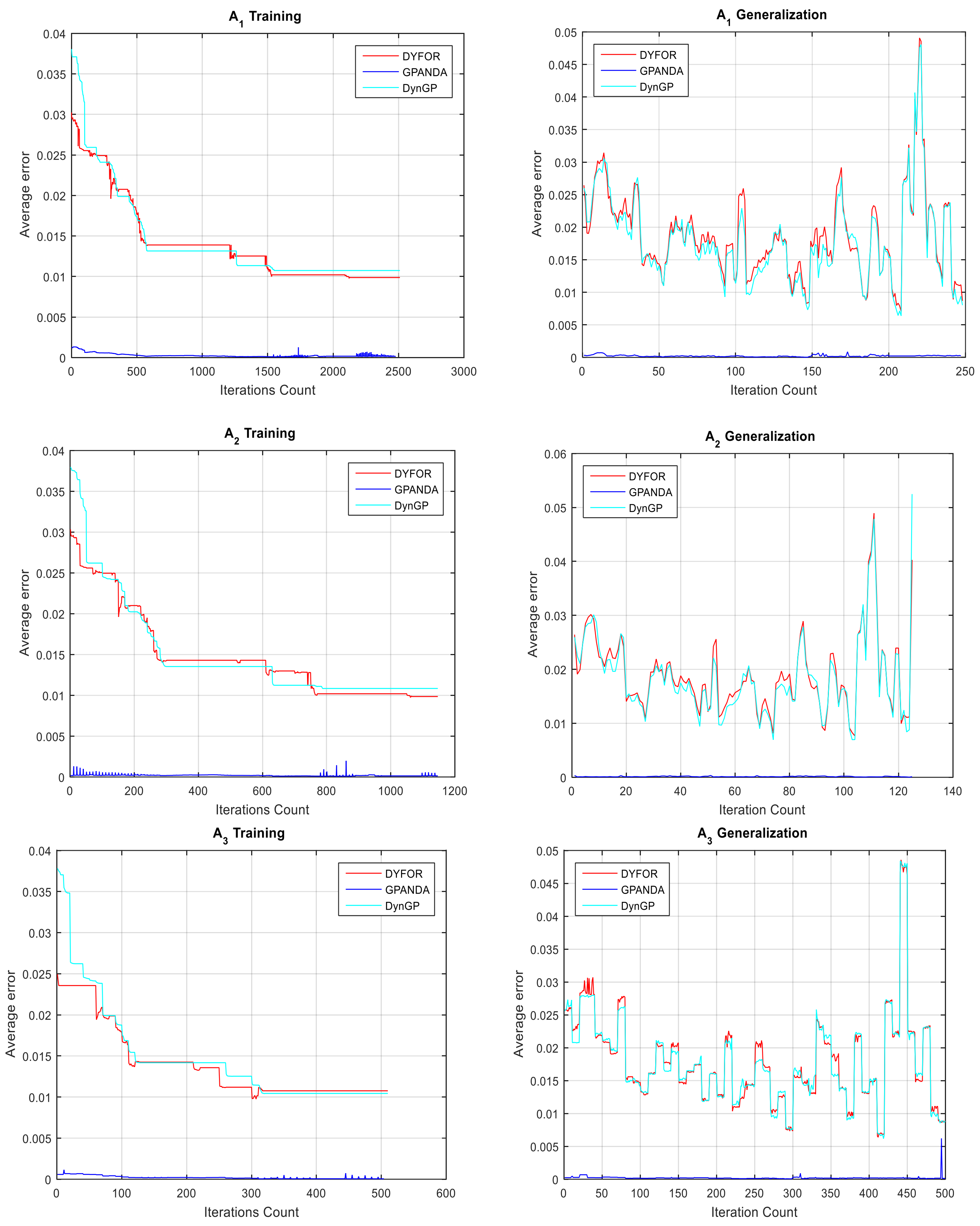


Figure 7.38: Training and Generalization E_{MS} for Electricity, Scenario $A_1 - A_3$

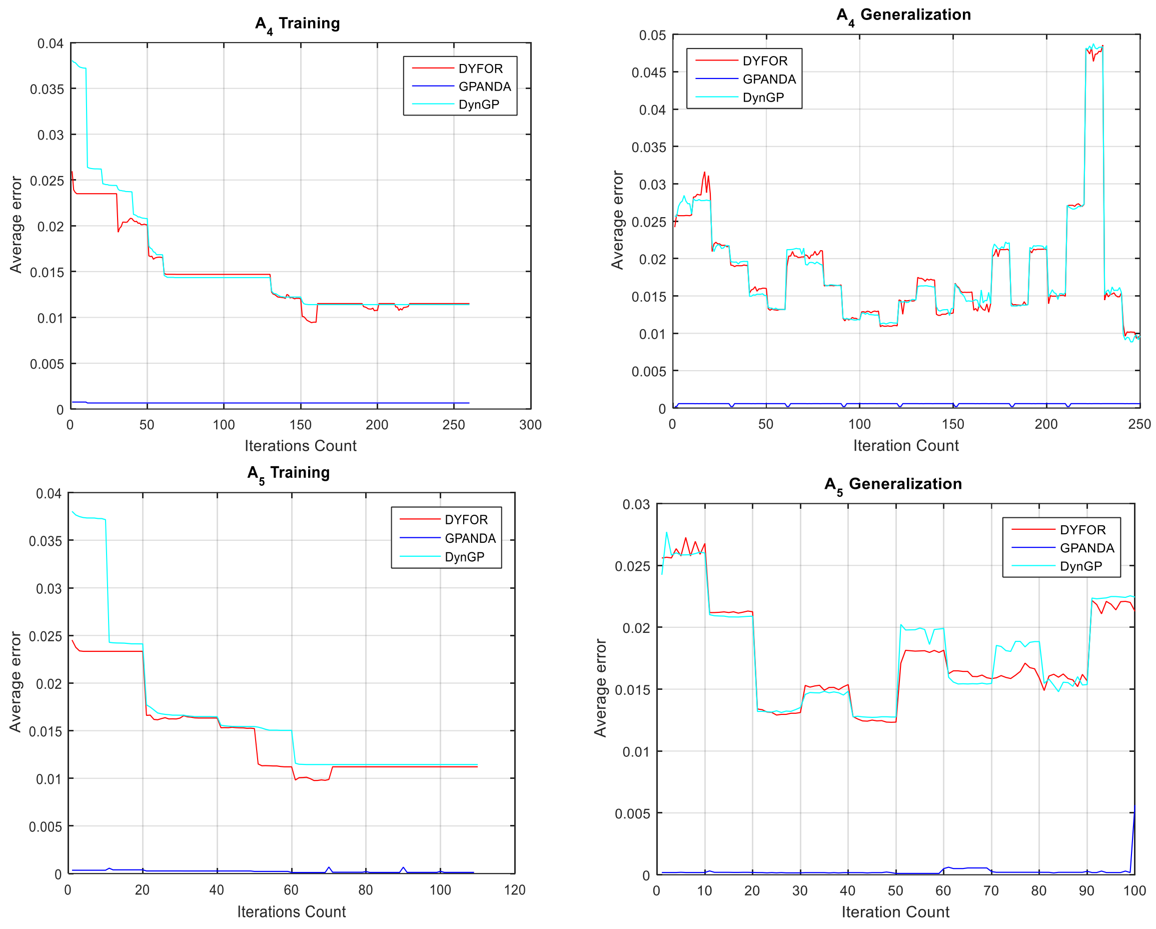


Figure 7.39: Training and Generalization E_{MS} for Electricity, Scenario $A_4 - A_5$

The algorithms under study are classified according to training and generalization performance. Table 7.59 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario A. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DyFor GP slightly outperforms DynGP for both training and generalization.

Table 7.58: Results of Predictive Performance for Electricity Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0145	0.0190	0.0147	0.0194	0.0149	0.0208	0.0154	0.0235	0.0166	0.0297	0.0152	0.0224
GPANDA	**3.92e-4**	**2.17e-4**	**1.92e-4**	**1.94e-04**	**2.10e-4**	**2.02e-4**	**6.61e-4**	**5.62e-4**	**2.56e-4**	**2.14e-4**	**0.0003**	**0.0002**
DyFor GP	0.0142	0.0196	0.0143	0.0195	0.0140	0.0202	0.0146	0.0237	0.0146	0.0282	0.0143	0.0498

Discussion

GPANDA outperforms all algorithms in all cases in Scenario A. As already explained, the superior performance of GPANDA suggests the capability of the algorithm to adapt

Table 7.59: Algorithm Ranking for Electricity Scenario A

Algorithm	A_1		A_2		A_3		A_4		A_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	3	3	2.6	2.6
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2	2	2.4	2.4

to a changing environment. There is a slight performance difference between DyFor GP and DynGP in Scenario A.

Scenario B - Quasi-Abrupt

Scenario B simulates less frequent changes having a frequency of severity of 50 iterations. Figure 7.40 - 7.41 are graphical illustrations in Scenario B for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario A, similar traits are also observed in Scenario B: the performance of DyFor GP and DynGP is the same in all cases in Scenario B for both training and generalization.

GPANDA detects an environment change and thereby adapts the predictive model, evident with improved predictive performance. GPANDA outperforms all other algorithms in Scenario B for both training and generalization. All cases in Scenario B shows that GPANDA consistently maintains the obtained minimum throughout the iteration's progression.

Table 7.60 presents the obtained results for the Electricity dataset, Scenario B. To ascertain the significance of the results obtained for Scenario B, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario A, the following can also be observed for the results in Table 7.60:

- GPANDA outperforms all algorithms.
- DyFor GP outperforms DynGP.

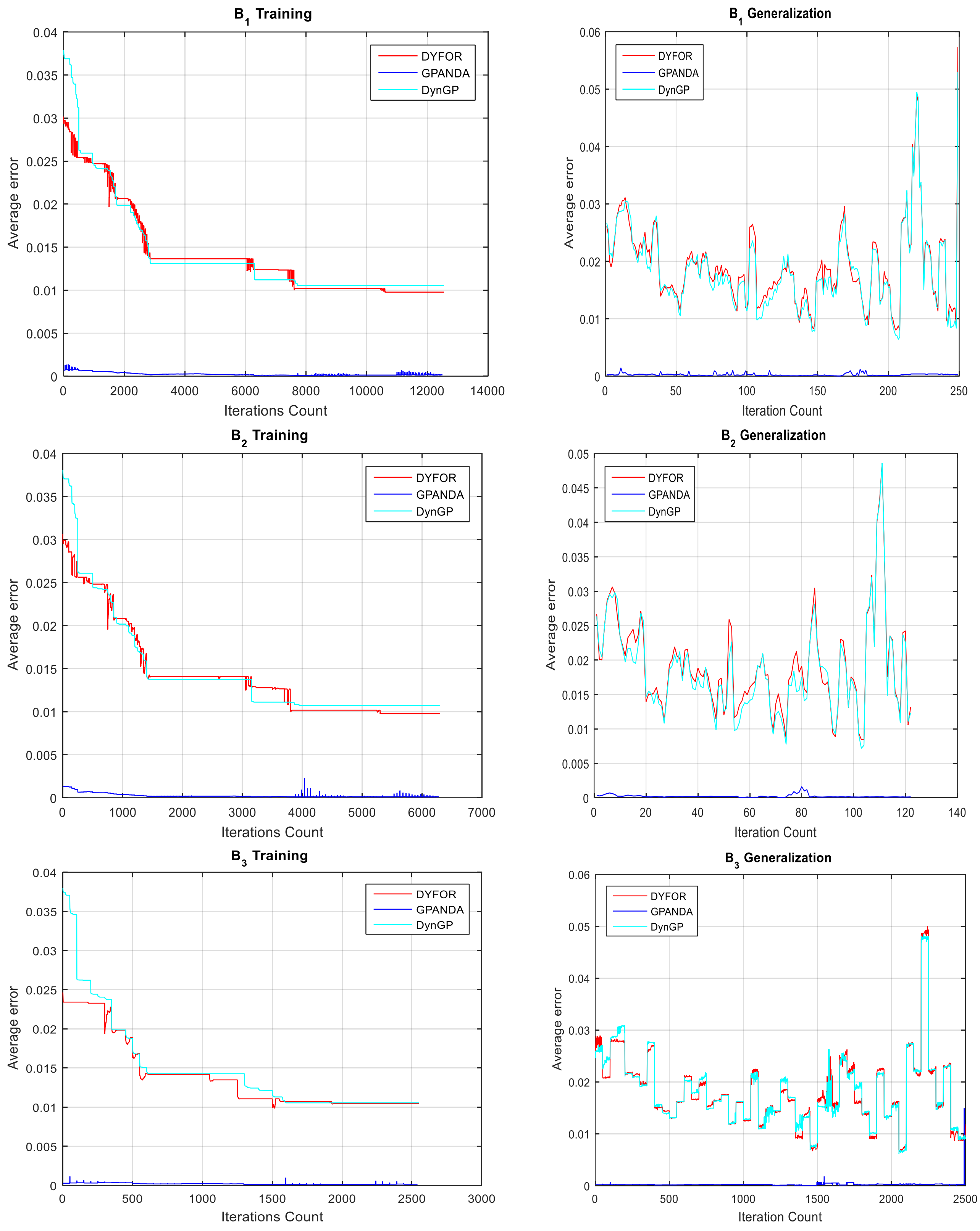


Figure 7.40: Training and Generalization E_{MS} for Electricity, Scenario $B_1 - B_3$

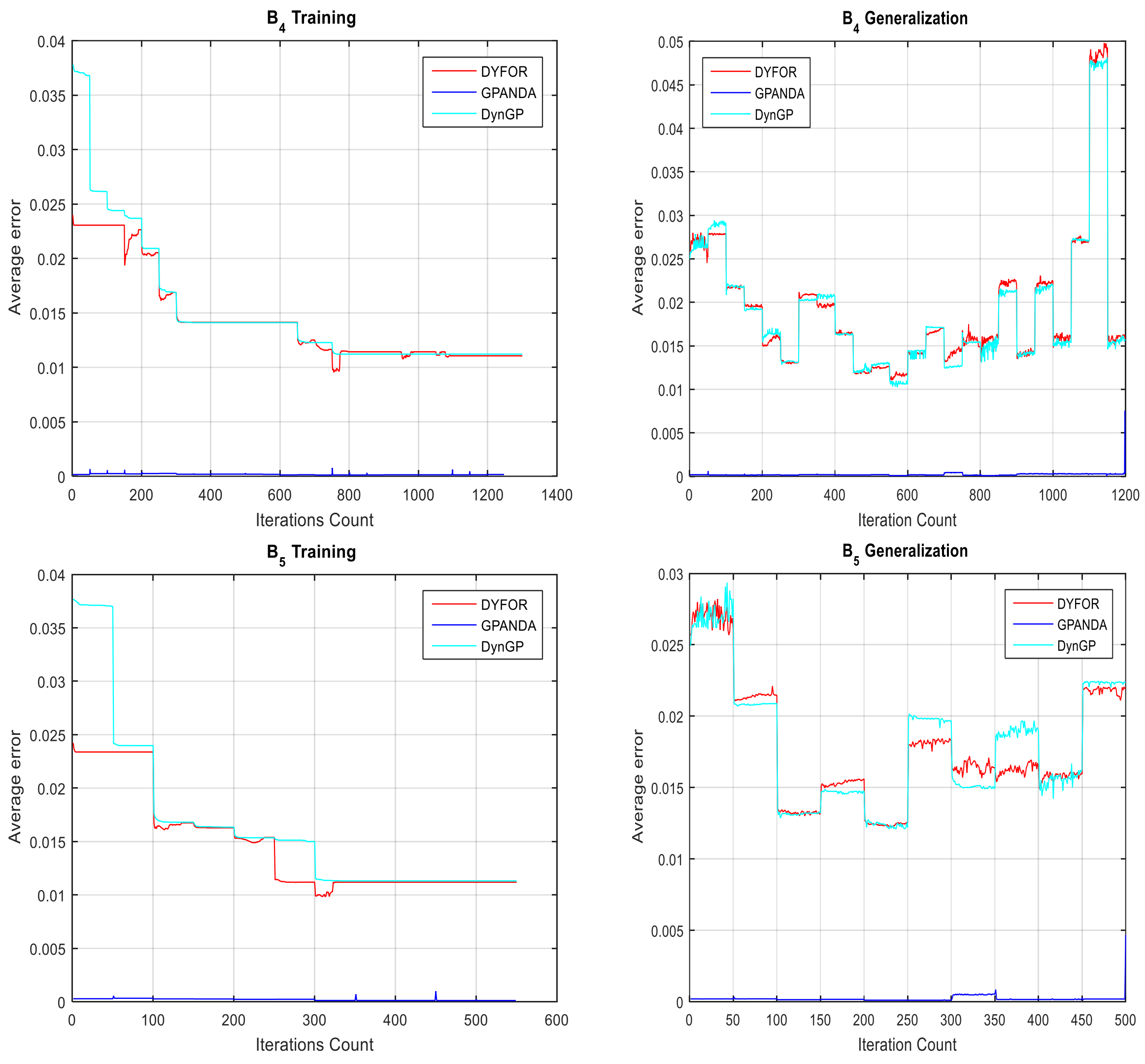


Figure 7.41: Training and Generalization E_{MS} for Electricity, Scenario $B_4 - B_5$

- The result that DyFor GP performs better than DynGP was found to be statistically significant at the 5% level of significance except for Scenario $B_1 - B_4$.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.60: Results of Predictive Performance for Electricity Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0144	0.0193	0.0146	0.0196	0.0149	0.0210	0.0152	0.0237	0.0165	0.0302	0.0151	0.0227
GPANDA	**2.38e-4**	**2.61e-4**	**2.39e-4**	**2.19e-4**	**1.97e-4**	**2.34e-4**	**1.72e-4**	**2.09e-4**	**2.04e-4**	**1.96e-4**	**0.0002**	**0.0002**
DyFor GP	0.0141	0.0201	0.0143	0.0203	0.0139	0.0205	0.0144	0.0236	0.0147	0.0300	0.0142	0.0229

The algorithms under study are classified according to training and generalization performance. Table 7.61 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario

B. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance on training whereas DynGP and DyFor GP are ranked on the same position for generalization.

Table 7.61: Algorithm Ranking for Electricity Scenario B

Algorithm	B_1		B_2		B_3		B_4		B_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	3	2.5	2.6	2.5
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2	2.5	2.4	2.5

Discussion

As observed in Scenario A, GPANDA outperforms all algorithms in all cases in Scenario B. As already explained, the superior performance of GPANDA suggests the capability of the algorithm to adapt the predictive model as changes in the environment occurred.

The generalization performance of DyFor GP is the same to DynGP in all cases in Scenario B.

Scenario C - Quasi-Progressive Variation

Scenario C simulates less frequent changes of 100 iterations before a sliding window shift. Figure 7.42 - 7.43 are graphical illustrations as observed in Scenario C for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As observed in Scenario B, similar traits are also observed in Scenario C: the performance of DyFor GP and DynGP is the same in all cases in Scenario C for both training and generalization except in Scenario C_1 on training.

GPANDA detects an environment change and thereby adapts the predictive model, evident with improved predictive performance as illustrated in Figure 7.42 - 7.43. Consequently, GPANDA outperforms all other algorithms in Scenario C for both training and generalization.

Table 7.62 presents the obtained results for the Electricity dataset, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. As observed in Scenario B, the following are also observed for the results presented in Table 7.62:

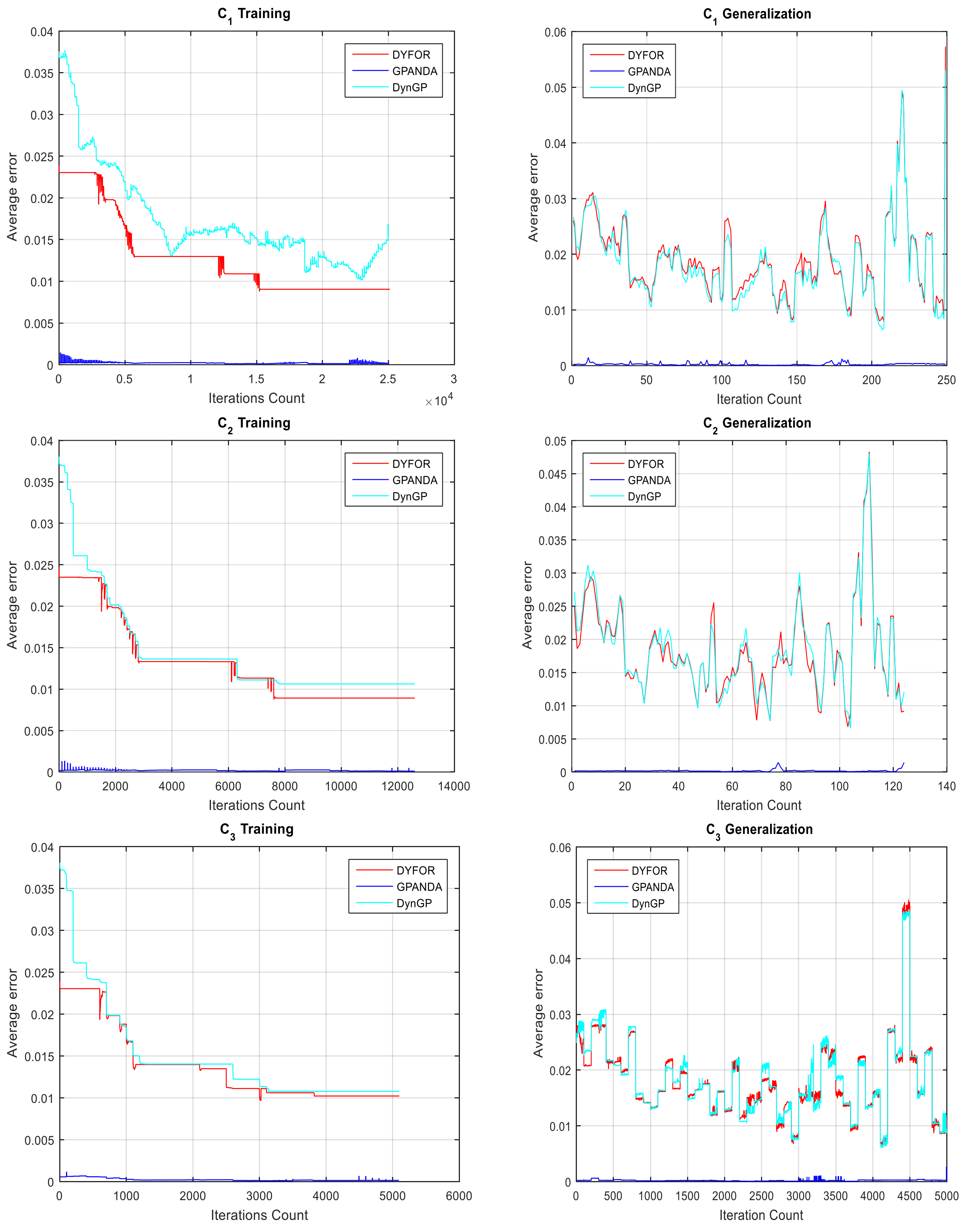


Figure 7.42: Training and Generalization E_{MS} for Electricity, Scenario $C_1 - C_3$

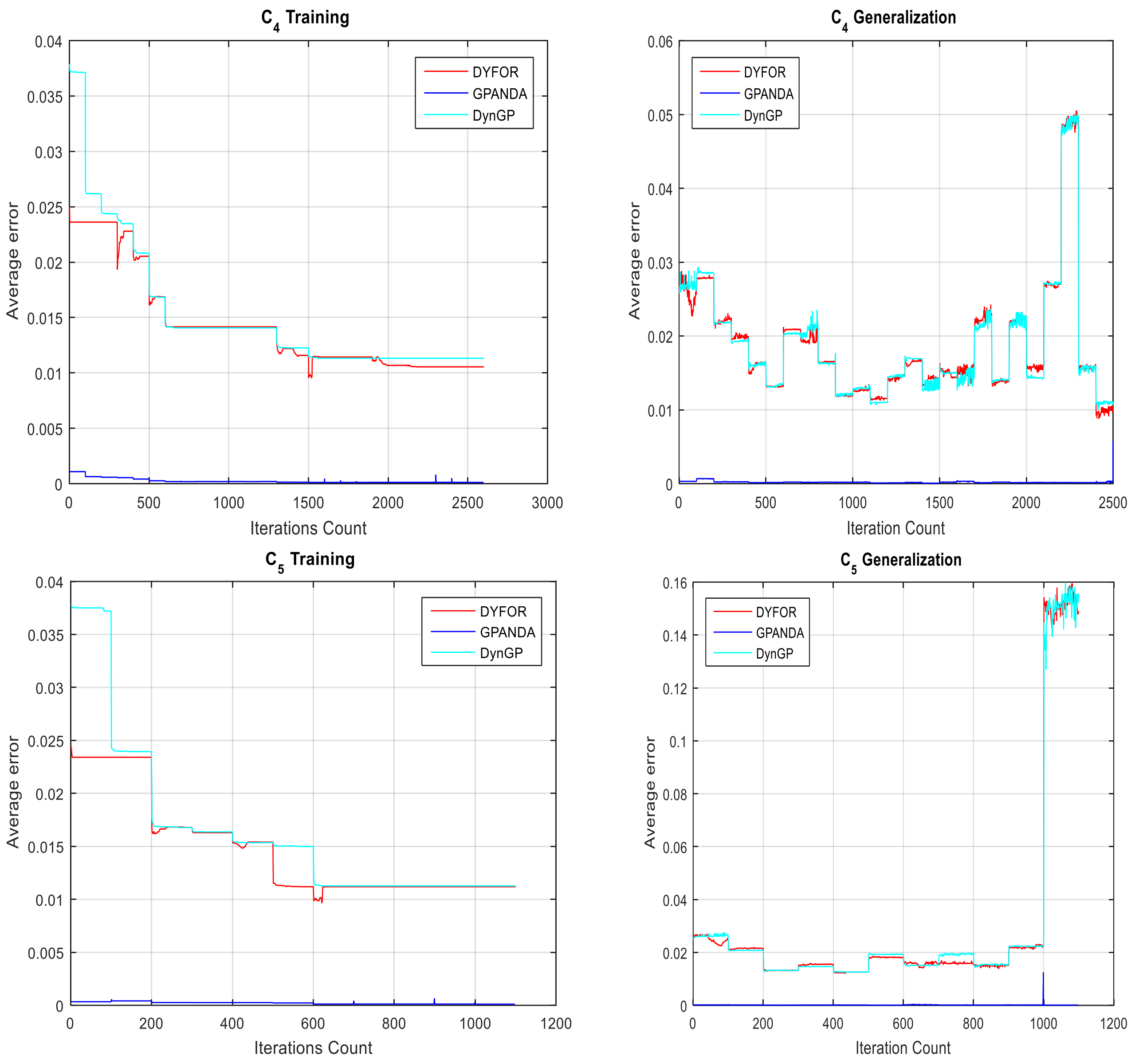


Figure 7.43: Training and Generalization E_{MS} for Electricity, Scenario $C_4 - C_5$

- GPANDA outperforms all algorithms.
- DyFor GP outperforms DynGP.
- No statistically significant differences exist in performance between DyFor GP and DynGP with regards to performance for generalization in Scenario C.
- The result that GPANDA performs better than both DyFor GP and DynGP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.63 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario

C. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance on training whereas DynGP and DyFor GP are ranked on the same position for generalization.

Table 7.62: Results of Predictive Performance for Electricity Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0173	0.0177	0.0146	0.0198	0.0149	0.0210	0.0153	0.0243	0.0165	0.0300	0.0157	0.0225
GPANDA	**1.91e-4**	**1.71e-4**	**1.88e-4**	**1.89e-4**	**2.37e-4**	**2.07e-4**	**2.51e-4**	**1.97e-4**	**2.15e-4**	**1.71e-4**	**0.0002**	**0.0001**
DyFor GP	0.0130	0.0179	0.0131	0.0196	0.0138	0.0205	0.0144	0.0237	0.0147	0.0297	0.0138	0.0222

Table 7.63: Algorithm Ranking for Electricity Scenario C

Algorithm	C_1		C_2		C_3		C_4		C_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3 2.5	3	2.5	3	2.5	3	2.5	3	2.5	3	2.5	
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2.5	2	2.5	2	2.5	2	2.5	2	2.5	2	2.5

Discussion

As observed in Scenario B, GPANDA outperforms all algorithms in all cases in Scenario C. As already explained, the superior performance of GPANDA suggests the capability of the algorithm to adapt the predictive model as changes in the environment occurred.

The generalization performance of DyFor GP is the same to that of DynGP in all cases in Scenario C.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms are given enough iterations to induce an optimal predictive model even in abruptly changing environments before an environment change. Consequently, failure to adapt or induce an optimal solution may not be attributed to the temporal severity property of the problem set under consideration but of the algorithm.

Figure 7.44 - 7.45 are graphical illustrations as observed in Scenario D for the progression of $E_{MS}T$ and $E_{MS}G$ over time. As illustrated in Figure 7.44-7.45, DyFor GP slightly outperforms DynGP on training in Scenario $D_1 - D_2$ whereas in Scenario $D_3 - D_5$, the performance of DyFor GP and DynGP is the same.

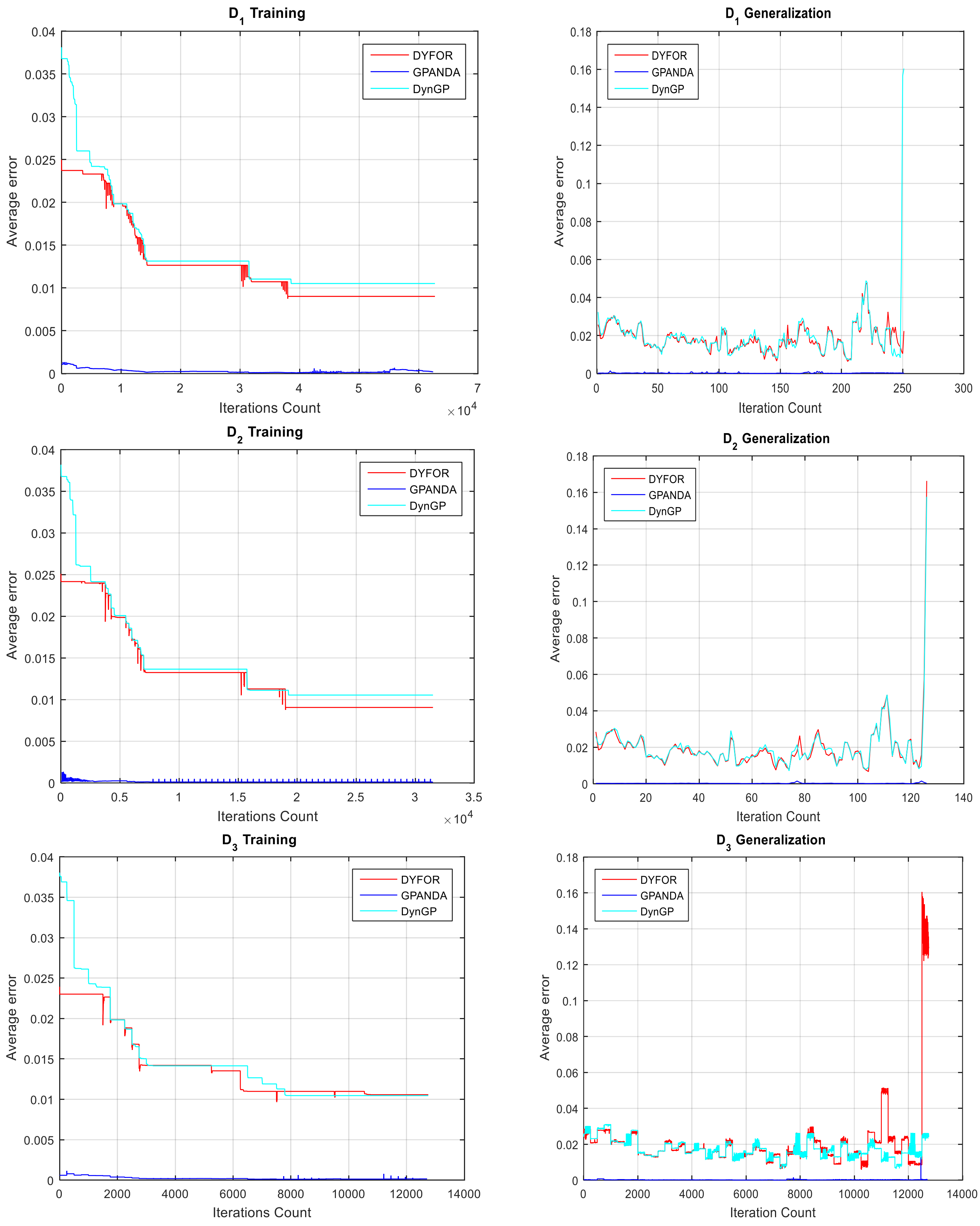


Figure 7.44: Training and Generalization E_{MS} for Electricity, Scenario $D_1 - D_3$

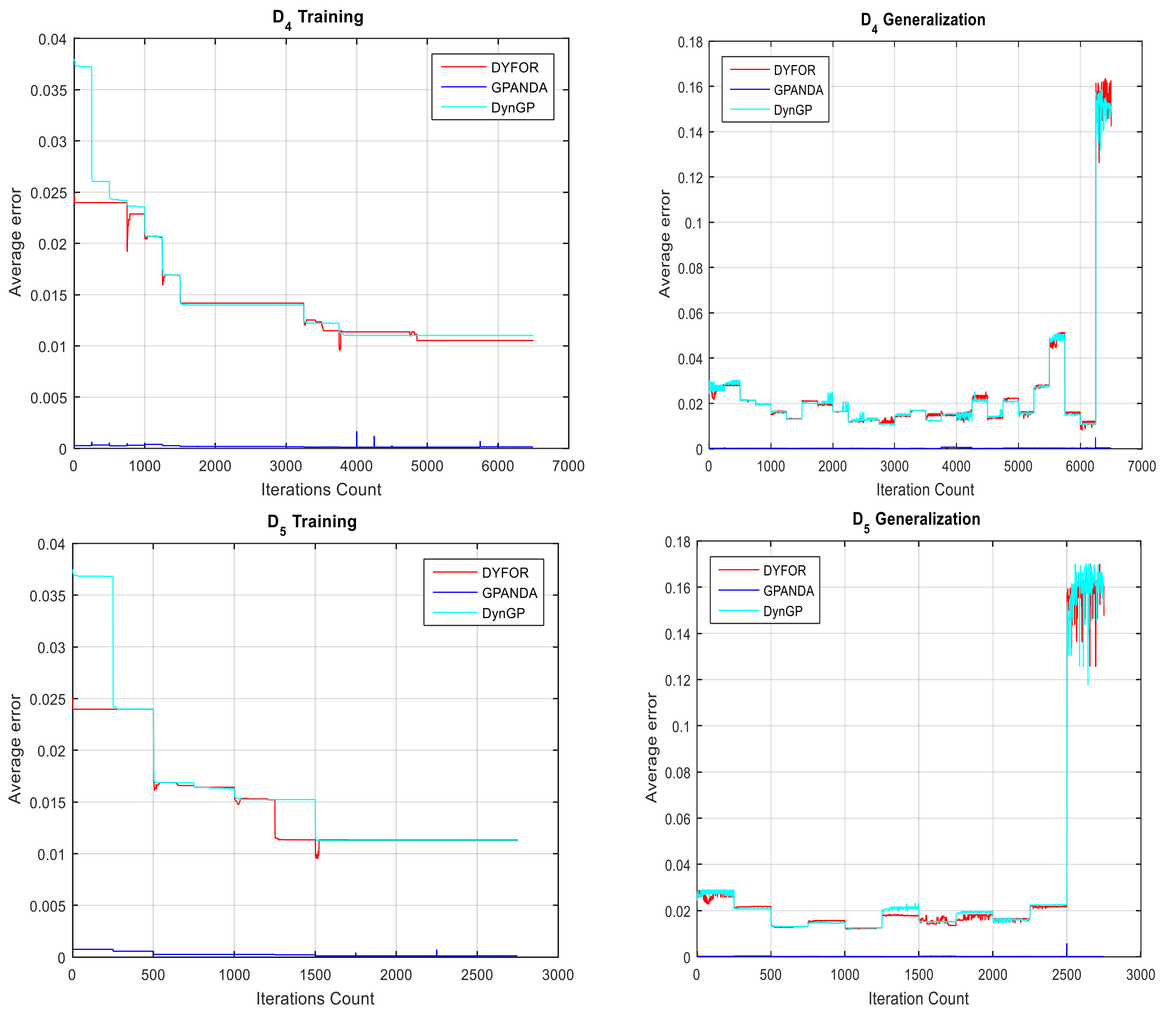


Figure 7.45: Training and Generalization E_{MS} for Electricity, Scenario $D_4 - D_5$

The performance of DyFor GP and DynGP for generalization is the same and for some instances along with the iteration's progression, DynGP outperforms DyFor GP for generalization. GPANDA exhibits superior performance and outperforms all other algorithms for both training and generalization.

Table 7.64 presents the obtained results for the Electricity demand dataset, Scenario D. To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.64:

- GPANDA consistently outperforms all algorithms in all cases in Scenario D for both training and generalization.

- No statistically significant differences exist in performance between DyFor GP and DynGP with regards to performance for generalization in Scenario D_4 and Scenario D_5.
- The result that DynGP performs better than DyFor GP was found to be statistically significant at the 5% level of significance for Scenario D_3, conversely, DyFor GP outperforms DynGP for Scenario $D_1 - D_2$.
- The result that GPANDA performs better than DyFor GP was found to be statistically significant at the 5% level of significance.

Table 7.64: Results of Predictive Performance for Electricity Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0272	0.0246	0.0219	0.0209	0.0169	0.0181	0.0151	0.0239	0.0164	0.0311	0.0195	0.0237
GPANDA	**2.84e-4**	**2.81e-4**	**1.28e-4**	**1.85e-4**	**2.35e-04**	**2.33e-4**	**1.84e-4**	**2.28e-4**	**2.67e-4**	**1.75e-4**	**0.0002**	**0.0002**
DyFor GP	0.0129	0.0184	0.0133	0.0197	0.0140	0.0204	0.0144	0.0241	0.0149	0.0305	0.0139	0.0226

The algorithms under study are classified according to training and generalization performance. Table 7.65 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario D. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance for both training and generalization.

Discussion

As observed in Scenario C, similar traits are also observed in Scenario D: GPANDA outperforms all algorithms in all cases in Scenario D.

Table 7.65: Algorithm Ranking for Electricity Scenario D

Algorithm	D_1		D_2		D_3		D_4		D_5		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	3	3	3	3	2	3	2.5	3	2.5	3	2.6
GPANDA	1	1	1	1	1	1	1	1	1	1	1	1
DyFor GP	2	2	2	2	2	3	2	2.5	2	2.5	2	2.4

Slight performance deterioration of DynGP can be attributed to the disruptive effects of dynamic parameter tuning when there is an adequate number of iterations. Thus, dynamic parameter tuning tends to overshot the best-obtained solution if the number of iterations increases.

Table 7.66: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for Electricity Dataset - Scenario A-D

	A		*B*		*C*		*D*		Average	
Algorithm	$E_{MS}G$	$avg\bar{t}$	$E_{MS}G$	$avg\bar{t}$	$E_{MS}G$	$avg\bar{t}$	$E_{MS}G$	$avg\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	0.0224	**0.018**	0.0227	**0.043**	0.0225	**0.122**	0.0237	**0.306**	0.0228	**0.1222**
GPANDA	**0.0002**	5.004	**0.0002**	18.08	**0.0001**	29.85	**0.0002**	85.21	**0.0001**	34.28
DyFor GP	0.0498	3.128	0.0229	10.63	0.0222	18.66	0.0226	53.26	0.0293	21.41

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.66 presents the average $E_{MS}G$ and average computational time ($\bar{t}$) in minutes for Scenario A-D. As observed in the Trend dataset, DynGP obtains the least average computational time in all scenarios under consideration in the Electricity dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance.

Table 7.67 presents the obtained results for Scenario A - D. GPANDA is ranked as the overall winner while DynGP is categorized as having the least effective performance.

Table 7.67: Overall Algorithm Ranking for Electricity Dataset

Algorithm	*A*	*B*	*C*	*D*	Average
DynGP	2.6	2.5	2.5	2.6	2.6
GPANDA	1	1	1	1	1
DyFor GP	2.4	2.5	2.5	2.4	2.5

7.3.7 Gross Domestic Product Dataset

The algorithms under study traverse the complete dataset for every scenario under consideration.

Scenario A - Abrupt Variation

The frequency of change is the same in Scenario A which is 25 iterations. Figure 7.46 is a graphical illustration as observed in all cases in Scenario A for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

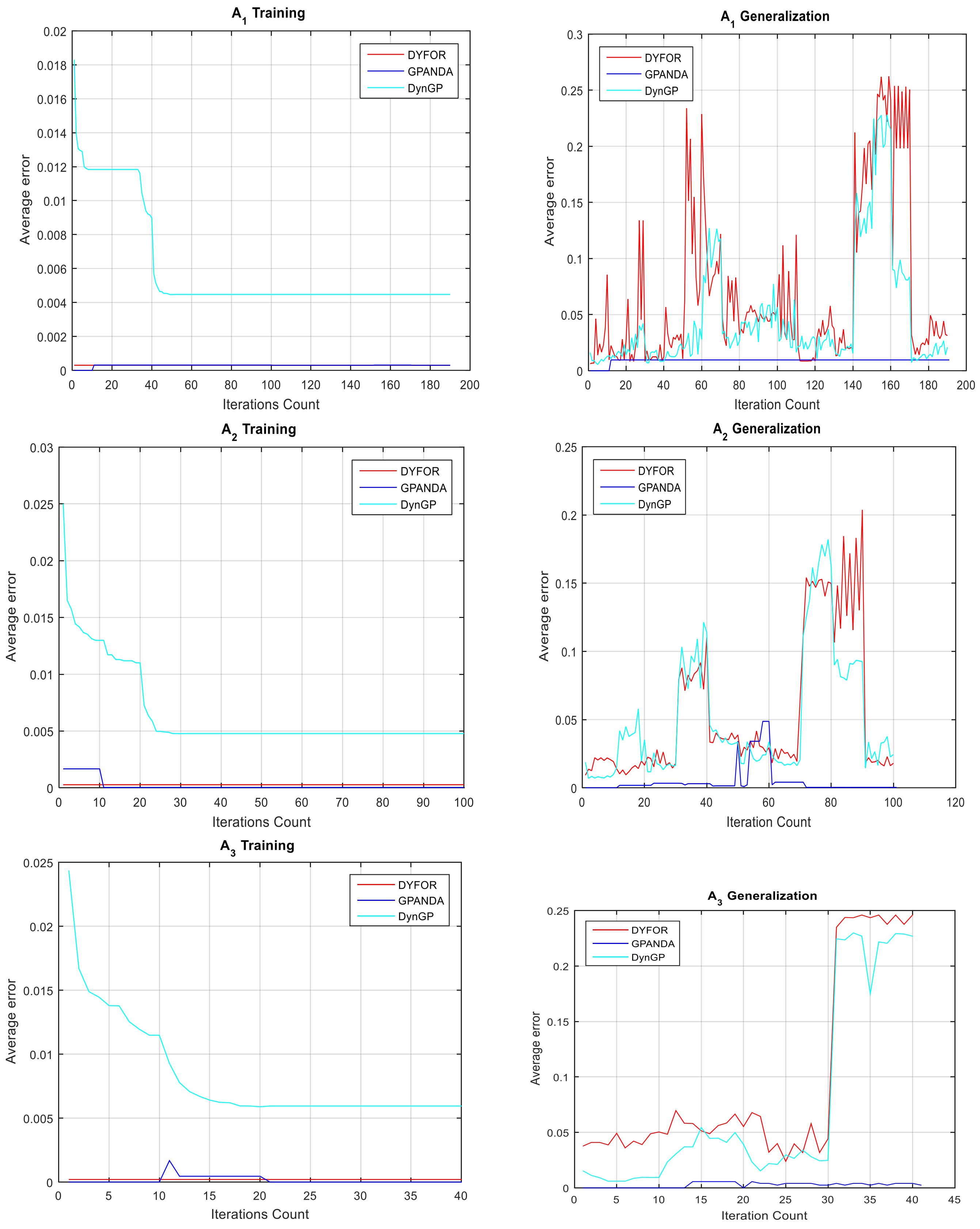


Figure 7.46: Training and Generalization E_{MS} for GDP, Scenario $A_1 - A_3$

Figure 7.46 illustrates that the performance of GPANDA deteriorates. DyFor GP outperforms GPANDA in Scenario A_1 on training and then yields the same performance to GPANDA in Scenario $A_2 - A_3$ on training.

As illustrated in Figure 7.46, for some instances along with the iteration's progression, DynGP outperforms GPANDA in Scenario A_1 on training. DyFor GP outperforms DynGP in all cases in Scenario A on training. However, the generalization performance of DyFor GP deteriorates in all cases in Scenario A. DynGP outperforms DyFor GP for generalization in all cases in Scenario A.

GPANDA outperforms both DyFor GP and DynGP for generalization though yield results which are the same to both DyFor GP and DynGP for some instances along with the iterations progression.

Table 7.68 presents the obtained results for the GDP dataset, Scenario A. To ascertain the significance of the results obtained for Scenario A, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.68:

- GPANDA outperforms all algorithms.
- DyFor GP outperforms DynGP.
- The result that DynGP performs better than DyFor GP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DynGP was found to be statistically significant at the 5% level of significance.

The algorithms under study are classified according to training and generalization performance. Table 7.69 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario A. Since there is no statistically significant differences between GPANDA and DyFor GP on training, both algorithms are ranked the overall winner on training whereas GPANDA

Table 7.68: Results of Predictive Performance for GDP Scenario A

Algorithm	A_1		A_2		A_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0060	0.0474	0.0066	0.0505	0.0083	0.0737	0.0069	0.0572
GPANDA	**9.18e-4**	**0.0036**	**1.97e-4**	**0.0364**	**1.53e-4**	**0.0027**	**0.0004**	**0.0142**
DyFor GP	9.75e-4	0.0702	2.74e-4	0.0544	1.27e-4	0.0964	0.0004	0.0736

is ranked as the overall winner on $E_{MS}G$. DyFor GP is categorized as having the least effective performance for generalization whereas DynGP is on training.

Table 7.69: Algorithm Ranking for GDP Environment Scenario A

Algorithm	A_1		A_2		A_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2	3	2	3	2	3	2
GPANDA	3	2	3	2	3	2	3	2
DyFor GP	1.5	3	1.5	3	1.5	3	1.5	3

Discussion

The predictive performance of GPANDA outperforms DynGP for generalization in all cases in Scenario A. The superior performance of GPANDA suggests the induction of predictive model with a higher generalization performance.

The progressive changes happening in Scenario $A_1 - A_2$ increase the complexity of the predictive task since the sliding window at any given instance is made up of two or more different data generating processes. Thus, as the window slides, new data points from a new data generating process constitutes the generalization dataset whereby old data points from the past generating process constitutes the training dataset. However, all algorithms detect an environmental change in such scenarios, consequently, adapts the generated predictive model.

Scenario B - Quasi-Abrupt

Scenario B simulates less frequent changes having a frequency of severity of 50 iterations. Figure 7.47 is a graphical illustration as observed in Scenario B for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

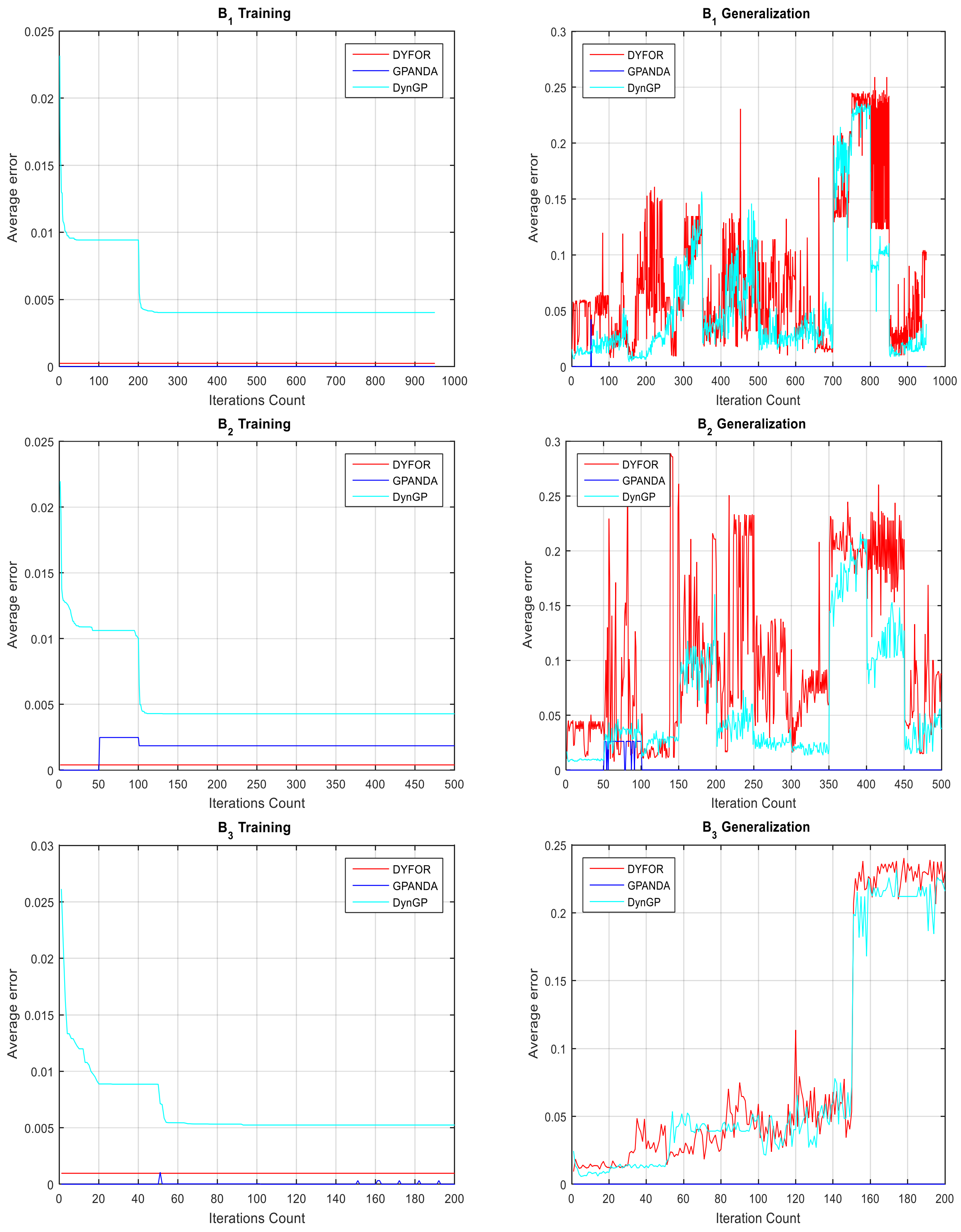


Figure 7.47: Training and Generalization E_{MS} for GDP, Scenario $B_1 - B_3$

As observed in Scenario A, DynGP outperforms DyFor GP for generalization in all cases in Scenario B. Also, GPANDA outperforms all other algorithms for generalization in all cases in Scenario B, though, for some instances along with the iteration's progression in Scenario B_2 , all algorithms yield the same performance as illustrated in Figure 7.47.

GPANDA and DyFor GP yield the same performance in all cases in Scenario B on training which can be attributed to many independent variable attributes that enable the algorithms to induce predictive models of improved performance. Consequently, DynGP exhibits the worst performance on training in all cases in Scenario B.

Table 7.70 presents the obtained results for the GDP dataset, Scenario B. To ascertain the significance of the results obtained for Scenario B, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.70:

- GPANDA outperforms all algorithms.
- DyFor GP outperforms DynGP.
- The result that DynGP performs better than DyFor GP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DynGP was found to be statistically significant at the 5% level of significance.

Table 7.70: Results of Predictive Performance for GDP Scenario B

Algorithm	B_1		B_2		B_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0052	0.0574	0.0057	0.0578	0.0066	0.0775	0.0058	0.0642
GPANDA	**1.39e-6**	**4.45e-5**	**0.0022**	**4.07e-5**	**2.23e-5**	**9.12e-31**	**0.0007**	**2.84e-5**
DyFor GP	3.82e-4	0.0779	4.01e-4	0.0618	9.69e-4	0.0843	0.0005	0.0746

Table 7.71 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario B. GPANDA is ranked the overall winner for both training and generalization whereas DynGP is categorized as having the least effective performance on training and DyFor GP for generalization.

Table 7.71: Algorithm Ranking for GDP Environment Scenario B

Algorithm	B_1		B_2		B_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2	3	2	3	2	3	2
GPANDA	1	1	2	1	1	1	1.3	1
DyFor GP	2	3	1	3	2	3	1.7	3

Discussion

The predictive performance of GPANDA outperforms both DynGP and DyFor GP for generalization in all cases in Scenario B. Consequently, the predictive performance of DynGP outperforms DyFor GP. As already mentioned, the superior performance of GPANDA suggests the induction of predictive model with a higher generalization performance.

Scenario C - Quasi-Progressive Variation

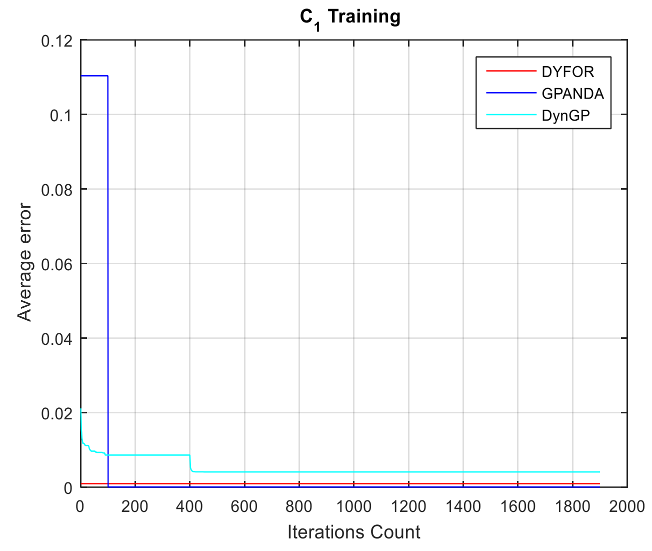
Scenario C simulates less frequent changes having a frequency of severity of 100 iterations. Figure 7.48 is a graphical illustration as observed in Scenario C for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

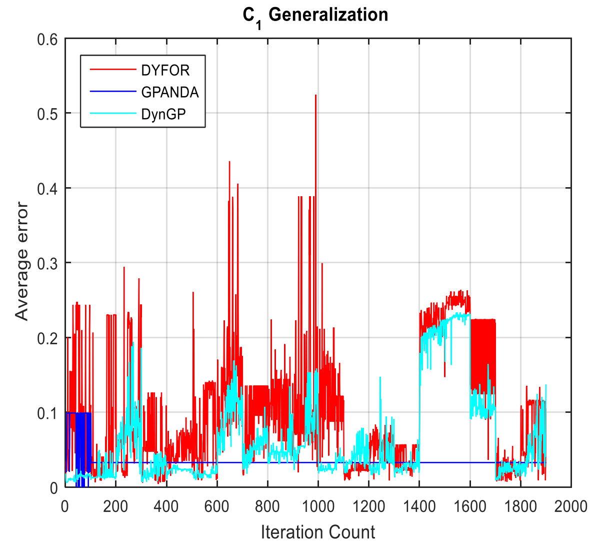
As illustrated in Figure 7.48, GPANDA exhibits the worst performance for the first 100 iterations, then the performance improved along with the iteration's progression yielding the same performance to DyFor GP on training in Scenario $C_1 - C_2$.

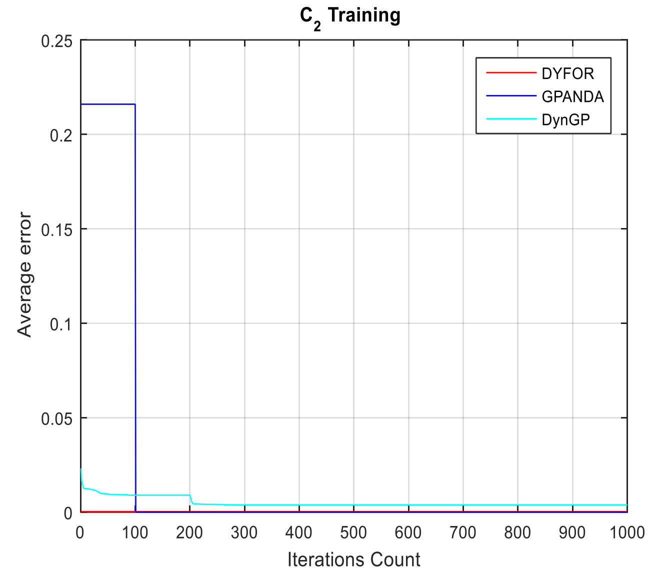
DyFor outperforms DynGP on training in all cases in Scenario C. As observed in Scenario B, similar traits are also observed in Scenario C for generalization: DynGP outperforms DyFor GP for generalization in Scenario C_1 and Scenario C_3. Also, GPANDA outperforms all other algorithms in all cases in Scenario C for generalization, though, for some instances along with the iteration's progression, all algorithms yields the same performance as illustrated in Figure 7.48.

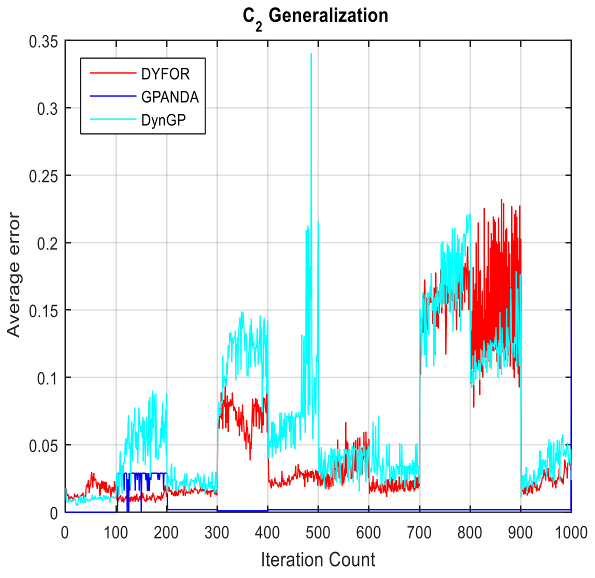
Table 7.72 presents the obtained results for the GDP, Scenario C. To ascertain the significance of the results obtained for Scenario C, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in

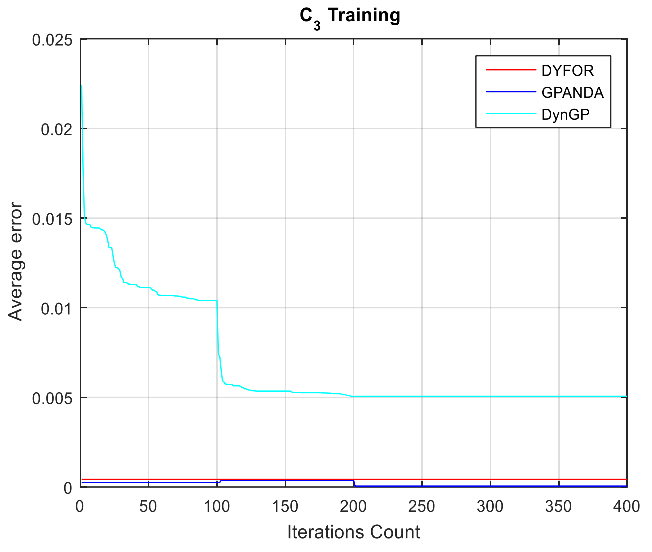
Appendix 1.











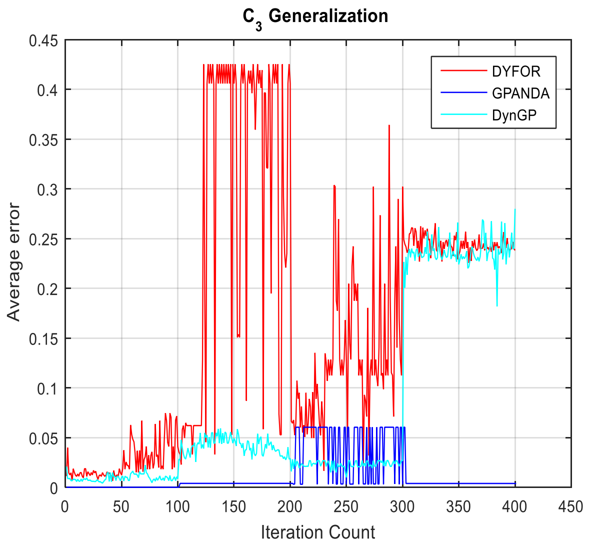


Figure 7.48: Training and Generalization E_{MS} for GDP, Scenario $C_1 - C_3$

The following can be observed for the results presented in Table 7.72:

- GPANDA outperforms all algorithms.
- DynGP outperforms DyFor GP.
- The result that DynGP performs better than DyFor GP was found to be statistically significant at the 5% level of significance.
- The result that GPANDA performs better than DynGP was found to be statistically significant at the 5% level of significance.

Table 7.72: Results of Predictive Performance for GDP Scenario C

Algorithm	C_1		C_2		C_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0051	0.0632	0.0051	0.0680	0.0068	0.0776	0.0056	0.0696
GPANDA	**0.0059**	**0.0351**	**0.0216**	**0.0042**	**1.79e-4**	**0.0122**	**0.0092**	**0.0171**
DyFor GP	0.0042	0.1001	2.66e-4	0.0527	0.0010	0.1710	0.0018	0.1079

The algorithms under study are classified according to training and generalization performance. Table 7.73 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario C. GPANDA is ranked the overall winner for generalization whereas DyFor GP is ranked the overall winner on training. DynGP and GPANDA are categorized as having the least effective performance on training whereas DyFor GP for generalization.

Table 7.73: Algorithm Ranking for GDP Environment Scenario C

Algorithm	C_1		C_2		C_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	2	2	2	3	3	2	2.3	2.3
GPANDA	3	1	3	1	1	1	2.3	1
DyFor GP	1	3	1	2	2	3	1.3	2.7

Discussion

The predictive performance of GPANDA outperforms both DynGP and DyFor GP for generalization in all cases in Scenario C. The outstanding generalization performance of GPANDA suggests the effectiveness of the piecewise regression approach that induced predictive models with higher generalization performance.

Scenario D - Progressive Variation

Scenario D simulates environments of low temporal severity with 250 iterations. Thus, all algorithms are given enough iterations to induce an optimal predictive model even in abruptly changing environments before an environment change. Consequently, failure to adapt or induce an optimal solution may not be attributed to the temporal severity property of the problem set under consideration but of the algorithm.

Figure 7.49 is a graphical illustration as observed in Scenario D for the progression of $E_{MS}T$ and $E_{MS}G$ over time.

As illustrated in Figure 7.49, algorithms under study show similar traits to Scenario C: DynGP outperforms DyFor GP for generalization in all cases in Scenario D. Also, GPANDA outperforms all algorithms in all cases in Scenario D for both training and generalization, though, for some instances along with the iteration's progression, all algorithms yield the same performance in Scenario D_1 and D_2.

Table 7.74 presents the obtained results for the GDP dataset for Scenario D. To ascertain the significance of the results obtained for Scenario D, statistical analysis was performed and p-values, only for scenarios that included a value of $p > 0.05$, are presented in Appendix 1. The following can be observed for the results presented in Table 7.74:

- GPANDA outperforms all algorithms.
- DyFor GP outperforms DynGP.
- The result that DynGP performs better than DyFor GP was found to be statistically significant at the 5% level of significance.
  item The result that GPANDA performs better than DynGP was found to be statistically significant at the 5% level of significance.

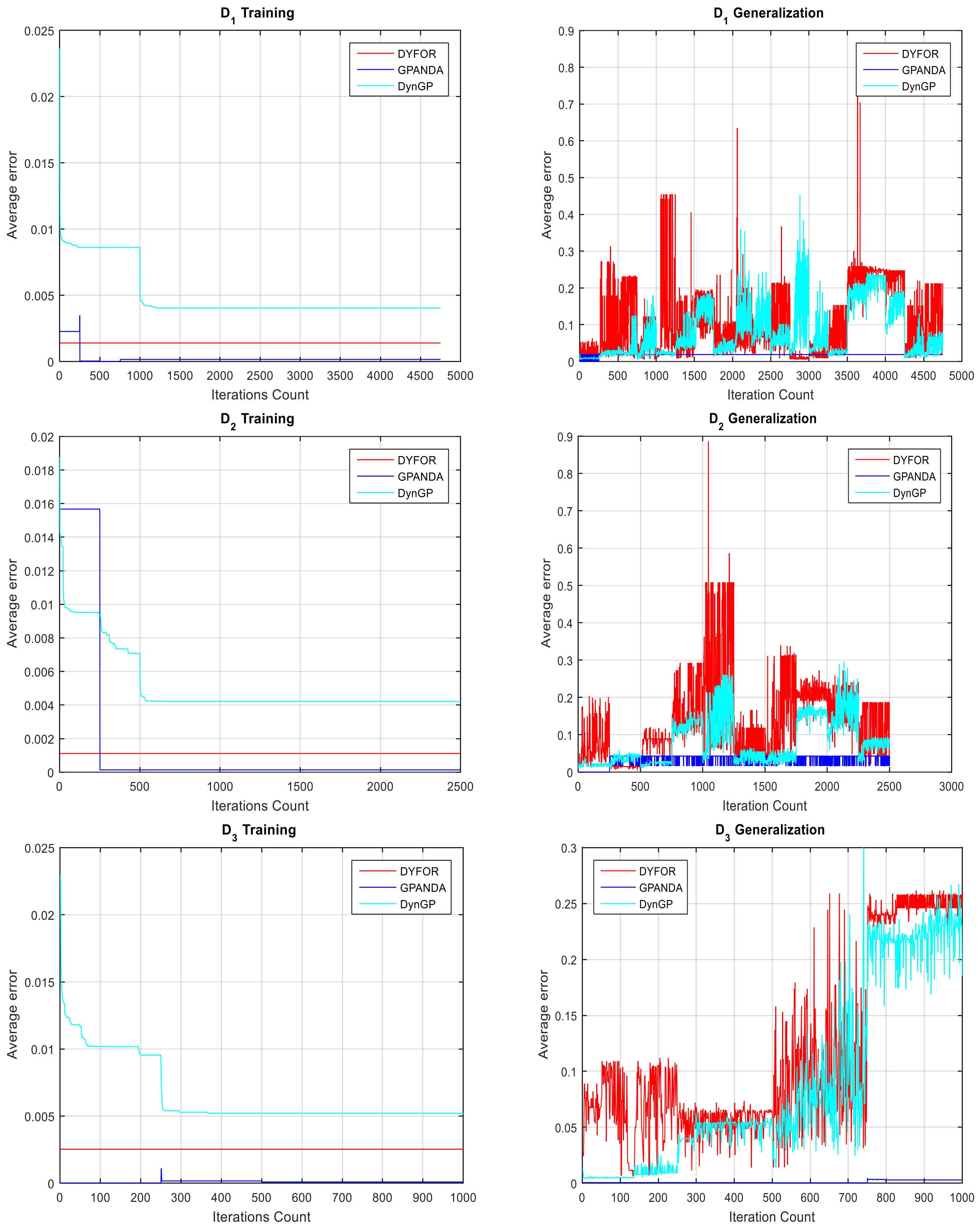


Figure 7.49: Training and Generalization E_{MS} for GDP, Scenario D_1 - D_3

The algorithms under study are classified according to training and generalization per-

Table 7.74: Results of Predictive Performance for GDP Scenario D

Algorithm	D_1		D_2		D_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	0.0050	0.0780	0.0051	0.0823	0.0066	0.0892	0.0055	0.0831
GPANDA	**2.57e-4**	**0.0187**	**0.0007**	**0.0348**	**9.09e-5**	**0.0011**	**0.0003**	**0.0182**
DyFor GP	0.0014	0.1067	0.0011	0.1378	0.0025	0.1171	0.0016	0.1205

formance. Table 7.75 presents the obtained ranks on $E_{MS}T$ and $E_{MS}G$ for Scenario D. GPANDA is ranked as the overall winner for both $E_{MS}T$ and $E_{MS}G$. DynGP is categorized as having the least effective performance on training whereas DyFor GP for generalization.

Table 7.75: Algorithm Ranking for GDP Environment Scenario D

Algorithm	D_1		D_2		D_3		Average	
	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$	$E_{MS}T$	$E_{MS}G$
DynGP	3	2	3	2	3	2	3	2
GPANDA	1	1	1	1	1	1	1	1
DyFor GP	2	3	2	3	2	3	2	3

Discussion

The predictive performance of GPANDA outperforms both DynGP and DyFor GP for generalization in all cases in Scenario D. As already mentioned, the outstanding performance of GPANDA suggests the capability of the algorithm to induce predictive models with improved generalization performance.

The algorithms under study are classified according to generalization performance for all cases in Scenario A-D. Table 7.76 presents the average $E_{MS}G$ and average computational time ($\bar{t}$) in minutes for Scenario A-D. As observed in the Electricity dataset, DynGP obtains the least average computational time in all scenarios under consideration in the GDP dataset, whereas GPANDA obtains the highest average computational time in all scenarios. Conversely, GPANDA yields the best performance whereas DynGP has the least effective performance.

Table 7.76: Averages for $\bar{t}$ (in mins) and $E_{MS}G$ for GDP Dataset - Scenario A-D

	A		B		C		D		Average	
Algorithm	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$	$E_{MS}G$	$\bar{t}$
DynGP	0.0572	**0.013**	0.0572	**0.043**	0.0696	**0.114**	0.0831	**0.256**	0.0667	**0.1065**
GPANDA	**0.0142**	0.567	**0.0142**	4.809	**0.0171**	8.317	**0.0182**	23.74	**0.0159**	9.3582
DyFor GP	0.0736	0.132	0.0736	1.03	0.1079	1.98	0.1205	5.233	0.0939	2.0937

Table 7.77 presents the average obtained ranks on $E_{MS}G$ for Scenario A - D. GPANDA is ranked as the overall winner whereas DyFor GP is categorized as having the least effective performance.

Table 7.77: Overall Algorithm Ranking for GDP Dataset

Algorithm	A	B	C	D	Average
DynGP	2	2	2.3	2	2.1
GPANDA	1	1	1	1	1
DyFor GP	3	3	2.7	3	2.9

7.3.8 Overall Discussion

Table 7.78 presents the average results for spatial severities for each problem set on $E_{MS}G$ and computation time ($avg\bar{t}$) in mins, for each algorithm. The following can be observed for the results presented in Table 7.78. As spatial severity increases: the performance of GPANDA, for generalization, generally improved.

The performance of DynGP is reduced for generalization as spatial severity increases. As observed for GPANDA, DyFor GP performance is consistent for generalization, the performance generally improved as spatial severity increases. GPANDA obtains the least values of $E_{MS}G$ in all scenarios under consideration whereas DynGP and DyFor GP obtains competitive performance on $E_{MS}G$ for the following datasets: Progressive, Abrupt, and Electricity.

DyFor GP outperforms DynGP on Random and Trend datasets whereas DynGP outperforms DyFor GP on GDP dataset. DynGP obtains the least average computational

Table 7.78: Average Spatial Severities for each Dataset

		DynGP					GPANDA					DyFor				
	Features	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Prog	$E_{MS}G$	8.528	8.401	6.444	4.724	5.254	**0.231**	**0.0154**	**1.036**	**8.1e-4**	**0.0095**	4.468	3.883	4.257	2.496	2.455
	$avg\bar{t}$	**0.015**	**0.015**	**0.017**	**0.017**	**0.011**	14.15	8.732	2.947	1.036	0.493	2.95	1.48	0.37	0.20	0.11
Abrp	$E_{MS}G$	0.801	0.934	0.787	0.748	0.607	**0.1689**	**0.242**	**0.155**	**0.164**	**0.035**	0.759	0.757	0.747	0.717	0.605
	$avg\bar{t}$	**0.019**	**0.016**	**0.013**	**0.014**	**0.013**	14.83	8.287	1.841	1.092	0.542	2.932	1.517	0.383	0.202	0.103
Recur	$E_{MS}G$	8.732	8.517	6.866	5.044	5.155	**0.003**	**0.002**	**0.002**	**0.006**	**0.006**	8.448	4.809	5.257	3.460	3.460
	$avg\bar{t}$	**0.117**	**0.113**	**0.116**	**0.128**	**0.105**	49.81	25.41	8.499	3.663	1.872	28.67	14.12	4.722	2.035	1.04
Elect	$E_{MS}G$	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
	$avg\bar{t}$	**0.190**	**0.11**	**0.106**	**0.096**	**0.108**	96.32	48.16	16.73	8.27	3.24	60.2	30.1	10.02	4.657	2.122
Tred	$E_{MS}G$	0.088	0.075	0.083	0.0742		**4.87e-8**	**4.87e-6**	**1.14e-7**	**8.52e-8**		0.006	0.007	0.007	0.008	
	$avg\bar{t}$	**0.091**	**0.083**	**0.089**	**0.115**		9.57	4.89	2.95	1.42		4.265	2.91	1.457	0.7125	
Rand	$E_{MS}G$	2.551	2.510	2.403			**0.178**	**0.225**	**0.353**			2.472	2.419	2.419		
	$avg\bar{t}$	**0.143**	**0.145**	**0.134**			79.68	43.18	8.835			24.77	12.87	2.447		
GDP	$E_{MS}G$	0.063	0.068	0.077			**0.035**	**0.004**	**0.012**			0.100	0.052	0.171		
	$avg\bar{t}$	**0.099**	**0.131**	**0.090**			15.859	8.361	2.795			3.687	1.945	0.65		

time in all scenarios under consideration whereas GPANDA obtains the highest average computational time in all scenarios.

The computational time of GPANDA is reduced for the following datasets: Recurrent, Trend, and Electricity. The reduction in computational time can be attributed to the recurrent or cyclical variations happening in these datasets which proves to be a desirable and valuable quality that promoted the retrieval of previously acquired knowledge.

Table 7.79 presents the average results for temporal severities for each dataset for generalization for each algorithm. The following can be observed for the results presented in Table 7.79. As temporal severity increases: the performance of all algorithms improved. GPANDA exhibits the greatest improvement outperforming all algorithms. DyFor GP exhibits superior performance to DynGP. For GDP dataset, all algorithms exhibit a competitive performance for generalization. DynGP obtains the least average computational time in all scenarios under consideration whereas GPANDA obtains the highest average computational time in all scenarios.

Table 7.80 presents the overall averages for each dataset for generalization for each algorithm. As reported in Table 7.80, GPANDA obtains the least values for $E_{MS}G$ for all datasets under consideration whereas DynGP outperforms DyFor GP for generalization on the following datasets: Progressive, Electricity and GDP.

Table 7.79: Average Temporal Severities for each Dataset

		DynGP				GPANDA				DyFor			
	Features	1	2	3	4	1	2	3	4	1	2	3	4
Prog	$E_{MS}G$	5.0868	4.2926	6.2716	6.2758	**0.0037**	**0.0977**	**0.2588**	**0.0005**	4.0670	3.5214	3.5122	3.4855
	$avg\bar{t}$	**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	$E_{MS}G$	0.7780	0.7432	0.7758	0.7643	**0.1571**	**0.1888**	**0.1536**	**0.1654**	0.7128	0.7152	0.7174	0.7319
	$avg\bar{t}$	**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	$E_{MS}G$	5.6899	6.1076	6.4854	6.7637	**0.0084**	**0.0081**	**0.0043**	**0.0123**	3.9913	4.8910	4.7343	5.0812
	$avg\bar{t}$	**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	$E_{MS}G$	0.0224	0.0227	0.0225	0.0237	**0.0002**	**0.0002**	**0.0001**	**0.0002**	0.0498	0.0229	0.0222	0.0226
	$avg\bar{t}$	**0.018**	**0.043**	**0.122**	**0.306**	5.004	18.08	29.85	85.21	3.128	10.63	18.66	53.26
Tred	$E_{MS}G$	0.0762	0.0771	0.0845	0.0800	**4.57e-29**	**8.35e-6**	**9.69e-6**	**8.41e-8**	0.0071	0.0071	0.0071	0.0270
	$avg\bar{t}$	**0.013**	**0.046**	**0.084**	**0.235**	1.077	2.751	5.531	11.29	0.405	1.365	2.281	5.375
Rand	$E_{MS}G$	2.4554	2.4797	2.4886	2.4317	**0.2610**	**0.2663**	**0.2592**	**0.0552**	2.4180	2.3927	2.4074	2.4224
	$avg\bar{t}$	**0.021**	**0.059**	**0.127**	**0.312**	3.467	11.23	20.81	71.29	1.082	3.51	6.818	22.28
GDP	$E_{MS}G$	0.0572	0.0572	0.0696	0.0831	**0.0142**	**0.0142**	**0.0171**	**0.0182**	0.0736	0.0736	0.1079	0.1205
	$avg\bar{t}$	**0.013**	**0.043**	**0.114**	**0.256**	0.567	4.809	8.317	23.74	0.132	1.03	1.98	5.233

Table 7.80: Averages and Standard Deviation for each Dataset

	Features	DynGP	GPANDA	DyFor
Prog	$E_{MS}G$	5.4817±0.8403	**0.0901±0.1048**	3.6465±0.2431
	$avg\bar{t}$	**0.0145**	5.1869	1.025
Abrp	$E_{MS}G$	0.7653±0.0137	**0.1662±0.0137**	0.7193±0.0074
	$avg\bar{t}$	**0.0155**	5.4397	1.0278
Recur	$E_{MS}G$	6.2616±0.4039	**0.0082±0.0028**	4.6744±0.4131
	$avg\bar{t}$	**0.1164**	18.8956	9.9195
Elect	$E_{MS}G$	0.0228±0.0005	**0.0001±4.33e-05**	0.0293±0.0117
	$avg\bar{t}$	**0.1223**	34.2736	21.421
Tred	$E_{MS}G$	0.0794±0.0032	**4.27e-6±4.51e-06**	0.0120±0.0086
	$avg\bar{t}$	**0.0948**	5.1622	2.336
Rand	$E_{MS}G$	2.4638±0.0221	**0.2104±0.0896**	2.4101±0.0114
	$avg\bar{t}$	**0.1299**	26.1932	8.4225
GDP	$E_{MS}G$	0.0667±0.0106	**0.0159±0.0017**	0.0939±0.0207
	$avg\bar{t}$	**0.1065**	9.3582	2.094

The reported results suggest the capability of GPANDA to adapt a predictive model in a changing environment with concept drift occurring to yield outstanding performance though the process is computationally expensive. DyFor GP outperforms DynGP for generalization on the following datasets: Abrupt, Random and Trend. The performance of GPANDA is reduced for GDP dataset. The reduced performance of GPANDA can be attributed to the reduced size of the analysis window which suggests that GPANDA requires a sufficiently large size of data points to induce optimal models.

Figures 8.50 is a graphical illustration as observed under spatial severities for all datasets for generalization. For Progressive dataset, GPANDA detects a change in the environment and adapts the model as spatial severity increases, evident with reduced $E_{MS}G$. As a result, GPANDA outperforms both DynGP and DyFor GP. Also, the performance of both DynGP and DyFor GP greatly improved as spatial severity increases yielding the same 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 discards all data points from the past data generating processes. Hence, all algorithms yield outstanding performance.

For Abrupt dataset, GPANDA outperforms both DynGP and DyFor GP. However, GPANDA exhibits performance deterioration for $w_{shift} = 2$ and $w_{shift} = 4$. Generally, the performance of DyFor GP is consistent as spatial severity increases. The performance of DynGP and DyFor GP is consistent for all cases of spatial severity illustrated in Figure 7.50 for the Random dataset. DynGP exhibits the worst performance whereas GPANDA exhibits a superior performance. There is a slight performance deterioration on $w_{shift} = 4$ for GPANDA.

DynGP exhibits significant worst performance for all cases on Trend dataset whereas DyFor GP and GPANDA exhibit consistent performance as spatial severity increases.

For both GDP and Electricity demand datasets, the performance of DynGP and DyFor GP deteriorate as spatial severity increases. However, the performance of GPANDA improved as spatial severity increases on GDP dataset and exhibits superior performance which is maintained as spatial severity increases.

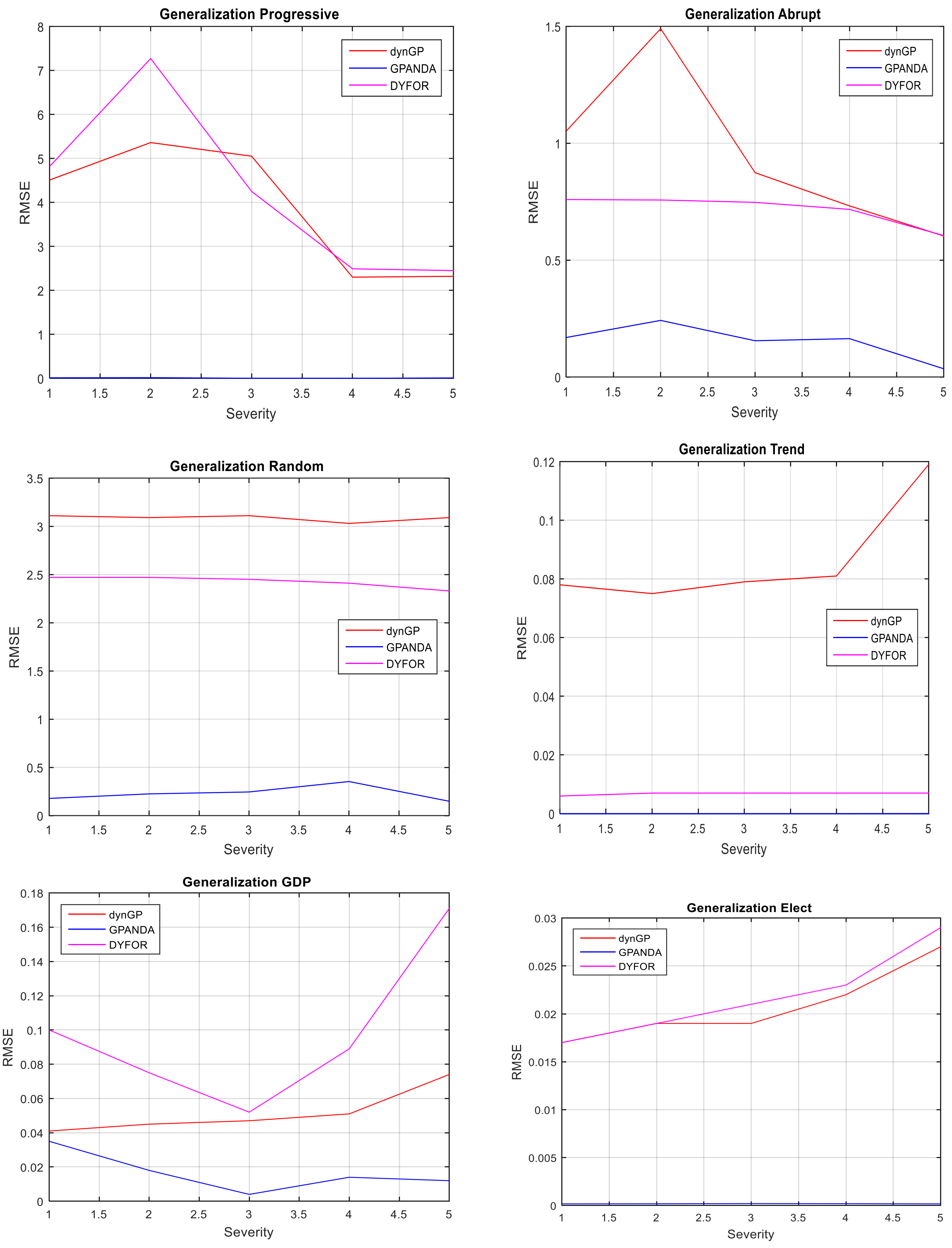


Figure 7.50: Averages for RMSE for Generalization per Spatial Severity

Figures 8.51 is a graphical illustration as observed under temporal severities for all datasets for generalization. For Progressive dataset, as temporal severity increases, the performance of GPANDA is consistent which suggest the capability of GPANDA to detect a change in the environment and adapts the model even for lower frequencies. As a result, GPANDA outperforms both DynGP and DyFor GP.

The performance of both DynGP and DyFor GP increase as temporal severity increases since there are sufficient iterations for each algorithm to converge toward the optimal solution.

For both Abrupt and Random dataset, the performance of DynGP and DyFor GP improved as temporal severity increases. GPANDA exhibits superior performance outperforming all other algorithms as temporal severity increases whereas DyFor GP outperforms DynGP.

As observed under spatial severity, DynGP exhibits significant worst performance for all cases on Trend dataset whereas DyFor GP and GPANDA exhibit consistent performance as temporal severity increases. The performance of GPANDA greatly deteriorates for GDP dataset whereas DynGP increases.

For Electricity dataset, the performance of DyFor GP deteriorates as temporal severity increases whereas DynGP increases outperforming DyFor GP for all cases. The performance of GPANDA slightly improved as temporal severity increases to exhibit superior performance which is maintained as temporal severity increases.

7.3.9 Comparing GPANDA to State-of-the-Art Techniques

To evaluate the effectiveness of the proposed approach to the existing techniques in the literature, comparative experiments are conducted with the state-of-the-art techniques on the given datasets. Each comparative experiment is discussed in the subsequent sections.

Two stock market datasets, Gross Domestic Product (US), GDP and Consumer Price Index inflation rate (US), CPI are selected to benchmark the performance of GPANDA to the leading studies [181][182][73]. The experiments for GPANDA are implemented as described in [73].

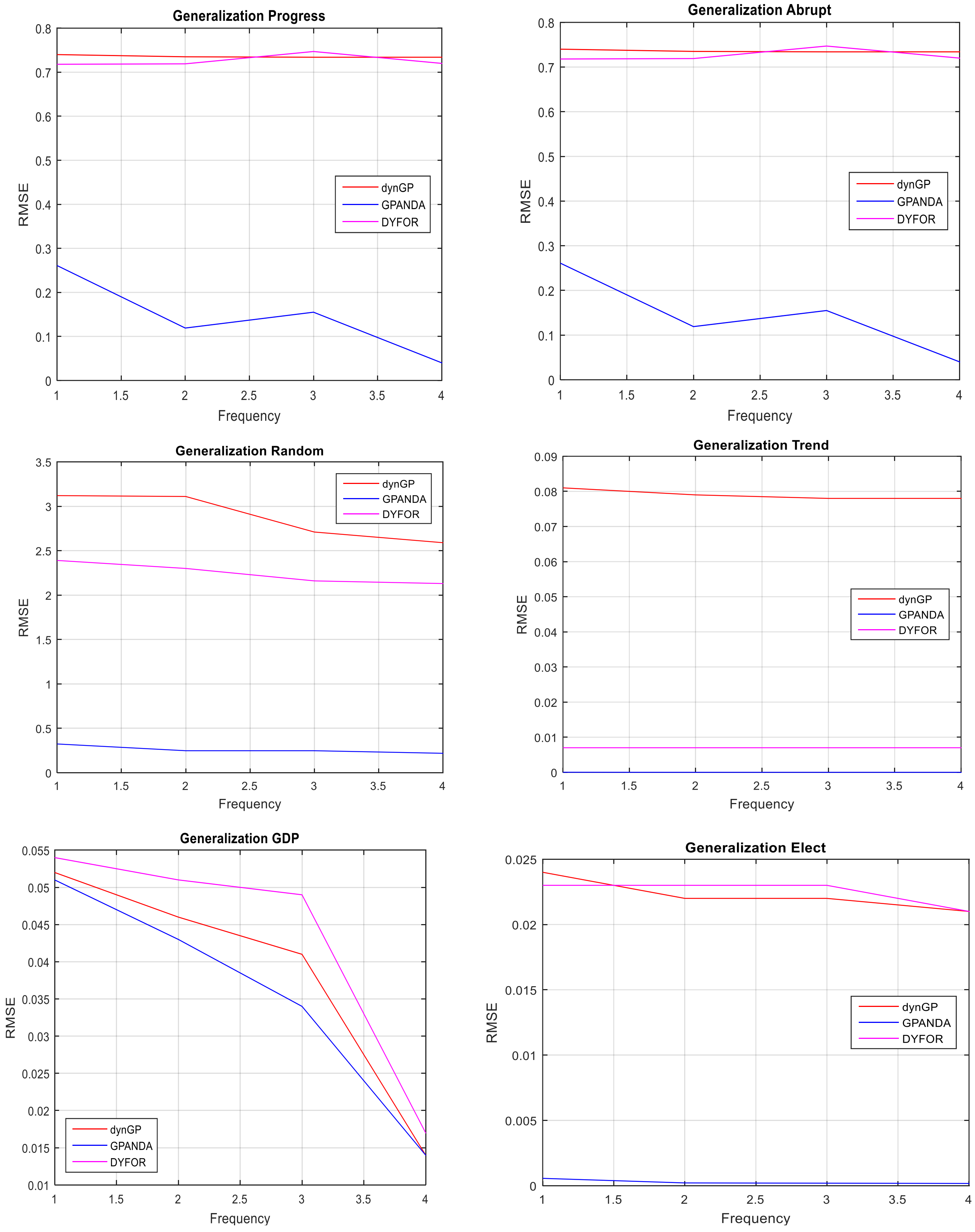


Figure 7.51: Averages for RMSE for Generalization per Temporal Severity

Table 7.81 presents the results for GPANDA and the state-of-the-art techniques: autoregressive (AR), real-time forecasting system (RTFS), and DyFor GP on the real-world dataset GDP reported in [197]. In this experiment, the GDP dataset is applied to GPANDA to perform one-step-ahead, quarterly GDP forecast using the same simulations used in [197].

As observed in the result presented in Table 7.81, GPANDA outperforms all other techniques.

Table 7.81: RMSE on GDP Dataset for the State-of-art Techniques

Forecasting Model	RMSE
AR	2.46
RTFS	1.85
DyFor GP	1.57
GPANDA	**0.88**

Table 7.82 presents the results for GPANDA on CPI inflation to the state-of-the-art techniques: conventional Phillips Curve (CPC) and DyFor GP on the real-world dataset CPI inflation reported in [197]. In this experiment, the CPI inflation dataset is applied to GPANDA to perform 12-month horizon forecasts using the same simulations used in [197].

Table 7.82: RMSE on CPI Inflation Dataset for the State-of-art Techniques

Forecasting Model	RMSE
CPC	2.30
GPANDA	**2.05**
DyFor GP	2.40

As observed in the results presented in Table 7.82, GPANDA outperforms all other techniques, though with a small margin compared to results presented in Table 7.81.

The predictive performance of GPANDA improves upon that of the state-of-the-art techniques reported in Table 7.81 and Table 7.82. The outstanding performance of

GPANDA suggests the capability of the approach to capture non-linearities present in the time series datasets: CPI inflation and GDP that are not captured by the competing state-of-the-art techniques.

In [179], the Australian Energy Market Operator electric load datasets were applied to the following approaches: General Linear Model-based Load Forecaster-Benchmark (GLMLF-B), random vector functional link network (RVFL), Persistence technique and ensemble incremental learning RVFL (DWT-EMD-RVFL) for the New South Wales (NSW) for the months: January, April, July and October of the year 2015 to assess the effectiveness of the approaches when seasonality was taken into consideration. In this work, the same simulations are applied to GPANDA to evaluate the effect of different seasons.

The SVR, SVRARIMA and ARIMASVR algorithms were applied to the California electricity market to perform next-week prices (short-term electricity prices) forecasting in the California electricity market [180]. The forecasting model was induced using the information available for each of the considered weeks including the 7 days' hourly historical price previous to the day of the week whose prices are to be predicted. In this work, the same datasets are applied to the GPANDA to perform next-week prices forecasting using the same simulations used in [180].

Table 7.83 and Table 7.84 presents the obtained results for the state-of-the-art techniques and GPANDA for NSW load dataset for the year 2015 and the California electricity market dataset. GPANDA exhibits outstanding performance outperforming all techniques under consideration except for DWT-EMD-RVFL that yields the same performance.

7.4 Summary

This chapter presented the results and provided a discussion for the experiments conducted. Two experiments were carried out: DynPSO, and GPANDA. The first experiment revealed that the proposed DynPSO induced nonlinear predictive model of

Table 7.83: Forecasting results of Electricity Data for the State-of-art Techniques

Dataset	Metric	RVFL	DWT-EMD-RVFL	GPANDA
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**

Table 7.84: Forecasting results of Electricity Data for the State-of-art Techniques

Dataset	Metric	SVR	SVRARIMA	GPANDA
1st Week	MAPE (%)	758.69	348.81	**258.72**
	RMSE	1.44	0.75	**0.57**
2nd Week	MAPE (%)	969.37	514.29	**460.46**
	RMSE	2.05	1.07	**0.87**

improved prediction performance on nonstationary regression problems.

In the second experiment, a thorough scalability investigation was carried out on the proposed GPANDA to the change in severities defined in different datasets. It was evident from the presented results that GPANDA successfully evolved predictive models with superior performance on nonstationary regression problems with concept drift occurring.

The next chapter concludes this work.

Chapter 8

Conclusion and Future Work

This chapter provides a conclusion to the work done in this thesis and discusses the directions for related future research. Section 8.1 presents an overview of this thesis and Section 8.2 discusses how the objectives have been met. Section 8.3 summarizes the findings obtained from the experimental study. Section 8.4 discusses some directions for related future research.

8.1 Summary of the Thesis

Nonstationary data is usually made up of generating processes which change over time. Therefore, if the knowledge of the existence of different segments is not taken into consideration, then, the induced predictive model is distorted by the past existing patterns. Thus, the challenge posed to a regressor is to select an appropriate segment that depicts the current underlying data generating process to be used in a model's construction.

A decision tree is a predictive modeling approach used in machine learning which has been applied to regression problems [8][9][10]. A model tree can build a decision tree hierarchy in trying to fit several smaller data segments of the dataset to yield an improved model that best fits the entire training dataset. Therefore, a model tree provides a piecewise linear regression model. Thus, a model tree splits the parameter space into subspaces and then fit a linear regression model for each subspace. As such, a model tree became the basis of this study.

The proposed GPANDA provides a piecewise nonlinear regression model for nonstationary data. GPANDA consists of three components: dynamic DE-based clustering algorithm to split the parameter space into subspaces that resemble different data generating processes present in the dataset; the dynamic PSO-based model induction approach to induce nonlinear models that describe each generated cluster; and a dynamic GP which evolves model trees that define the boundaries of nonlinear models expressed as terminal nodes.

The dynamic DE-based clustering algorithm, KCDCDynDE, automatically determines the number of clusters, K, in a dataset, therefore, becomes suitable to automatically determines the optimal number of clusters in nonstationary data whenever a change in an environment is evident [135]. Consequently, hybridizing a QR decomposition technique, to determine the coefficients of the model, and a dynamic PSO, to induce an optimal model that can adapt whenever a change in the environment occurs, to come up with DynPSO, decreases the performance deterioration in nonlinear regression designed for nonstationary environments that usually results from the environmental changes. Furthermore, a GP directly induce model trees and preserves genetic information expressed as subtree that models existing knowledge.

As such, if an environmental change is detected in a nonstationary dataset, KCDC-DynDE dynamically clusters the data. For the clusters that change, the DynPSO adapts nonlinear models or induce new models to create an updated GP terminal set and the GP evolves a piecewise predictive model that model nonstationary data with concept drift occurring.

To evaluate the effectiveness of GPANDA, two experimental evaluations were conducted. Also, to ascertain the significance of the results obtained by GPANDA and the best performing dynamic GPs, statistical analysis was used.

In Section 7.2, the performance of DynPSO was experimentally evaluated and the obtained results revealed that DynPSO outperforms the dynamic PSOs under study and the state-of-the-art techniques.

In Section 7.3, a thorough scalability investigation was carried out on the proposed GPANDA with respect to the change in severities defined in different datasets. Experiments were carried out to find out if and how the change in severity affects the performance of GPANDA. Two stock market datasets, GDP and CPI were selected to benchmark the performance of the proposed approach to the leading studies [198][199][197]. It is evident from the presented results that GPANDA outperforms the selected dynamic GPs and was competitive to the state-of-art techniques.

8.2 Objectives

The main goal of this work was to develop a predictive approach for nonstationary data with a numerical target that dynamically adapts when concept drift occurs which can also be used to extract knowledge from historical data. The objectives outlined in Section 1.2 are listed below and the discussion on how they are met is provided.

(a) *To improve the performance of dynamic PSO (DynPSO) by hybridizing it with a regression technique (either least-squares approximation or autoregressive) to induce optimal nonlinear regression models in nonstationary environments;*

To achieve this objective, a dynamic PSO-based nonlinear regression (DynPSO) algorithm that induced optimal nonlinear regression models in dynamic environments was developed. The DynPSO hybridized a dynamic PSO with a regression model (either QR decomposition or NARX). The regression model determines the coefficients of the model and a dynamic PSO optimizes the induced model which can adapt whenever a change in the environment occurs. Experiments were carried out using given datasets and the obtained results suggest that this hybridization decreases the performance deterioration that usually results from the environmental changes and consequently, improves the algorithm's performance.

(b) To improve the performance of GP by hybridizing it with a dynamic clustering algorithm and nonlinear model induction approach (GPANDA) to perform regression on nonstationary data with concept drift occurring;

To achieve this objective, a GPANDA algorithm to perform regression on nonstationary data with concept drift occurring was developed. GPANDA implemented a piecewise approach to predict a target, nonlinear model and consists of three components: dynamic DE-based clustering algorithm to extract clusters that resemble different data generating processes present in the dataset; DynPSO to fit a nonlinear regression model on each generated cluster; and GP to evolve model trees that define the boundaries of nonlinear models which were expressed as terminal nodes.

When an environmental change occurs, GPANDA reacts to the change by clustering the data and fitting a nonlinear regression model on each cluster. Nonlinear models become terminal nodes of GP-based model trees. Then, GP evolves predictive model trees that model nonstationary data with concept drift occurring.

Experiments were carried out using nonstationary datasets and the obtained results show that GPANDA yields high adaptation rates and accuracy to several types of concept drift.

(c) *To compare the performance of DynPSO to optimize the induced model in a nonstationary environment, to dynamic PSO algorithms, namely multi-swarm, reinitialized, and charged PSOs;*

To achieve this objective, the DynPSO was evaluated experimentally and compared with the dynamic PSOs, namely multi-swarm, reinitialized, and charged PSOs, to optimize the induced model and the regression parameters in the dynamic environment. To ascertain the significance of the results obtained by DynPSO and the dynamic PSOs, statistical analysis was used. The predictive accuracy of the regression model induced by DynPSO was compared to the predictive accuracy of the regression model induced by the multi-swarm, reinitialized and charged PSOs. The obtained results show that the DynPSO was adaptive to the changing environment, consequently, outperforms the dynamic PSOs for the given datasets. The NARX-based DynPSO outperforms QR decomposition-based DynPSO on nonstationary time series forecasting.

A comparative analysis of DynPSO with the state-of-the-art techniques was performed on the obtained results of DynPSO for Electricity dataset. The DynPSO yields competitive performance to the state-of-the-art techniques.

(d) *To compare the performance of GPANDA in terms of predictive accuracy and computational time to the best performing dynamic GP algorithms and the state-of-the-art techniques on nonstationary datasets that exhibit different characteristics of concept drift such as progressive, recurrent, abrupt and random changes on varying temporal and spatial severities.*

To achieve this objective, the proposed GPANDA was evaluated experimentally and compared with the dynamic GPs, namely DyFor GP and adaptive GP (DynGP), to evolve nonlinear predictive models in the dynamic environment with concept drift occurring. The predictive accuracy and computational time of GPANDA were compared to the predictive accuracy and computational time of DyFor GP and DynGP. To ascertain the significance of the results obtained by GPANDA and the best performing dynamic GPs, statistical analysis was used.

GPANDA obtained the best predictive accuracy for all datasets under consideration. The reported results suggested the capability of GPANDA to adapt a predictive model in a changing environment with concept drift occurring to yield improved performance. The performance of DynGP was the same to that of GPANDA for GDP dataset. The reduced performance of GPANDA for GDP dataset can be attributed to the reduced size of the analysis window which suggests that GPANDA requires a sufficiently large size of data points to induce optimal models.

DynGP obtained the least average computational time in all scenarios under consideration whereas GPANDA obtained the highest average computational time in all scenarios. The average computational time of GPANDA was reduced for the following datasets: Recurrent, Trend, and Electricity. The reduction in computational time can be attributed to the recurrent or cyclical variations happening in these datasets which proves to be a desirable and valuable quality that promoted the retrieval of previously acquired knowledge.

A comparative analysis of GPANDA with the state-of-the-art techniques: autoregressive, real-time forecasting system, conventional Phillips Curve and DyFor GP, was performed on the obtained results of GPANDA for GDP and CPI inflation rate datasets. GPANDA yields competitive performance to the state-of-the-art techniques.

8.3 Conclusion

This research has established the feasibility of GPANDA to evolve predictive models of improved performance on a nonstationary environment with concept drift occurring. The obtained results suggest the capability of DynPSO to track and adapt the induced model as the environment changes. The hybridization of the dynamic PSOs with a regression model indeed decreased the performance deterioration of the induced model that resulted from the environmental changes.

GPANDA exhibited outstanding performance in terms of predictive accuracy for all dataset on different temporal and spatial severities. The superior performance of GPANDA 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, GPANDA suffers from computational load due to the embedded clustering and the nonlinear model induction approach. Consequently, DynGP has the least precision with the best computational cost whereas DyFor GP provides a balance between precision and computational cost.

The finding of this work highlights the potential of GPANDA as an adaptive technique for real-world prediction application and suggests further investigation. The proposed GPANDA is considered as an attractive prediction alternative because:

- In GPANDA, there is no need for extensive data pre-processing, only data differencing is done for time series data, therefore, the technique can easily be applied to other problems besides those evaluated in this thesis.

- GPANDA induces a prediction model automatically, regardless of complexity, fitting a nonlinear model without necessarily specifying the model structure.
- GPANDA is a self-adaptive technique that tracks and automatically adapts the prevailing prediction model whenever an environmental change is detected to achieve an outstanding prediction accuracy.
- GPANDA treats the most recent historical occurrences as equally significant contributors to the final concept. As such, GPANDA adapts/evolves predictive models as underlying changes to data happens due to generating processes which change over time.
- Knowledge of the past may prove useful in quickly capturing the current environment or can improve the accuracy of a search. GPANDA takes advantage of existing nonlinear models that resembles knowledge acquired in the past which are exploited when a change in an environment occurs to model the current environment.
- The unfavorable computational cost of GPANDA can be greatly improved by increasing computational power since nowadays they is access to a computing environment with greater power and speed.

In general, GPANDA is a viable option for real-world prediction application and prove to stimulate new advances in the area of prediction.

8.4 Future Research

Future work could consider a scalability study of GPANDA. GPANDA splits the parameter space into subspaces that resemble different data generating process in which a model is fitted to each subspace. It is ideal to consider some methods to combine the generated multiple induced models. As expected, a more sophisticated prediction combination will result in performance improvements.

GPANDA consists of three components: dynamic DE-based clustering; dynamic PSO-based model induction approach and a GP to induce model trees. An analysis of the impact that each component has on the performance of GPANDA and the subsequent adaptation necessary to improve the overall performance of this technique can be considered. Future work could consider adapting each component as follows:

(a) The QR decomposition in dynamic PSO-based model induction approach can be replaced with NARX since QR decomposition proved to be computationally expensive compared to NARX. Also, to consider combining NARX-QPSO with empirical mode decomposition (EMD) to create an ensemble model that can capture the inherent nonlinearity in temporal data.

(b) In dynamic DE-based clustering, a centroid population whose context individual is assigned no data can be removed to eases the unnecessary computational effort.

(c) The individual's mutation, crossover and fitness evaluation is recursively performed in each GP generation and tends to severely affect the computational performance of the algorithm. As such, to improve the computational performance of a GP, a model tree can be expressed as an array which will then use indexing.

For many applications, especially those used in a dynamic environment, optimal parameter settings may vary throughout a run. More often than not, exploration is preferential at the beginning of the search process to ensure population coverage and diversity, while exploitation is best at the end of the search to ensure the convergence of the population to the global optimization. Future research will investigate the effects of adaptive parameters control during a GPANDA run.

The quality of the predictive model depends heavily on the fitness measure. However, it may not be clear how to select a fitness measure for a particular problem. It may be that a single measure performs well under a certain condition, but badly in others. Future work will use a multi-objective fitness function in GPANDA, which can be attributed to further improve the performance of the generated model.

Bibliography

[1] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, pp. 58., 2004.

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

[3] 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.

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

[5] G. Potgieter and A.P. Engelbrecht, "Genetic Algorithm for Structurally Optimisation of Learned Polynomial Expressions," *Applied Mathematics and Computation*, vol. 186, pp. 1441-1466, 2007.

[6] I. Ẑliobaitė, M. Pechenizkiy and, J. Gama," An overview of concept drift applications,"In *Big data analysis: new algorithms for a new society. Springer, Cham*, pp. 91-114, 2016.

[7] I. Ẑliobaitė, "Learning Under Concept Drift: An Overview," arXiv preprint *arXiv:1010.4784.,* 2010.

[8] J. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81-106, 1986.

[9] J. Quinlan, "Simplifying decision trees," *International Journal of man-machine studies,* vol. 27, no. 3, pp. 221-234, 1987.

[10] L. Rokach, and O.Z. Maimon, Data mining with decision trees: theory and applications, vol. 69, World Scientific, 2008.

[11] J. Quinlan, "Decision trees as probabilistic classifiers," In *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 31-37, 1987.

[12] H. Blockeel, and L. De Raedt, "Top-down induction of first-order logical decision trees," *Artificial intelligence*, vol. 101, no. 1-2, pp. 285-297, 1998.

[13] C.F. Lin, Y.C. Yeh, Y.H. Hung, and R.I. Chang, "Data mining for providing a personalized learning path in creativity: An application of decision trees," *Computers & Education*, vol. 68, pp. 199-210, 2013.

[14] R.T. Clemen, and T. Reilly, Making Hard Decision with Decision Tools Suite, 1, Ed., Boston, MA: Thomson: Duxbury, 2001.

[15] J. R. Quinlan, Rulequest research data mining tools, Australia: RuleQuest Research Pty Ltd, 2002.

[16] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression, Wadsworth Int. Group, 1984.

[17] W. Buntine, "Learning classification trees," *Statistics and Computing*, vol. 2, no. 2, pp. 63-73, 1992.

[18] R.A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179-188, 1936.

[19] D. Malerba, "Stepwise Induction of Model Trees," *In Congress of the Italian Association for Artificial Intelligence, LNCS*, pp. 20-32, Springer, Berlin, Heidelberg, 2001

[20] R. Setiono, W.K. Leow, and J.M. Zurada, "Extraction of rules from artificial neural networks for nonlinear regression," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 564-577, 2002.

[21] W. Loh, "Classification, and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp, 14-23, 2011.

[22] Y. Wang, and I.H. Witten, "Induction of model trees for predicting continuous classes," 1996.

[23] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I.H. Witten, "Using model trees for classification," *Machine learning*, vol. 32, no. 1, pp. 63-76, 1998.

[24] E. Ikonomovska, J. Gama, and S. Deroski, "Learning model trees from evolving data streams," *Data mining and knowledge discovery*, vol. 23, no. 1, pp. 128-168, 2011.

[25] J. Quinlan, C4.5: Programs for Machine learning, Elsevier, 2014.

[26] I.H. Osman, and J.P. Kelly, "Meta-heuristics theory and applications," *Journal of the Operational Research Society*, vol. 48, no. 6, pp. 657-657, 1997.

[27] F.W. Glover, and G.A. Kochenberger, Ed., Handbook of metaheuristics, vol. 57, *Springer Science & Business Media*, 2006.

[28] F. Glover,"Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research*, vol. 13, no. 5, pp. 533-549, 1986.

[29] R.C. Eberhart, and J. Kennedy, "A New Optimiser using Particle Swarm Theory," In *6th International Symposium on Micro Machine and Human Science*, Nagoya-Japan. IEEE Service Centre, 1995.

[30] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087, 1953.

[31] J. Holland, Adaptation in Natural and Artificial Systems, A. Arbor, Ed., MI: University of Michigan Press, 1975.

[32] J. Holland, "Genetic Algorithms," *Scientific American*, vol. 267, no. 1, pp. 66-73, 1992.

[33] B. Leonora, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing: an international journal*, vol. 8, no. 2, pp. 239-287, 2009.

[34] C. Blum, and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268-308, 2003.

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

[36] Y.S. Lee, and L.I. Tong, "Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming, " *Knowledge-Based Systems*, Elsevier, vol. 24, no. 1, pp. 66-72, 2011.

[37] J. Branke, Evolutionary optimization in dynamic environments, vol. 3, *Springer Science & Business Media*, 2012.

[38] C. Cruz, J.R. Gonzàlez, and D.A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing*, vol. 15, no. 7, pp. 1427-1448, 2011.

[39] A.P. Engelbrecht, Computational Intelligence: An Introduction, 2nd Edition ed., South Africa: John Wiley and Sons, 2007.

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

[41] J.G.O.L. Duhain, and A.P. Engelbrecht, Particle Swarm Optimisation in Dynamically Changing Environment-An Empirical Study, Pretoria: Department of Computer Science, University of Pretoria, South Africa, 2011.

[42] M. M. Salvado, "Handling concept drift in data stream mining," [Online Accessed 1 October 2018]. Available:
`https://www.slideshare.net/draxus/handling-concept-drift-in-data-stream-mining`.

[43] R. P. Chandra_Bose, W.M.P. van der Aalst, I. Ẑliobaitė, and M. Pechenizkiy, "Handling Concept Drift in Process Mining," In *International Conference on Advanced Information Systems Engineering*, Springer, Berlin, Heidelberg, pp. 391-405, 2011.

[44] M. Last, "Online classification of nonstationary data streams," *Intelligent Data Analysis*, vol. 6, no. 2, pp. 129-147, 2002.

[45] S. Garci , J. Luengo, and F. Herrera, "Data preprocessing in data mining," *Springer*, 2015.

[46] 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.

[47] 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.

[48] 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.

[49] J. Gama, I. Ẑliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1-44:37, 2014.

[50] 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.

[51] C. Alippi, G. Boracchi, and M. Roveri, “Just in time classifiers: Managing the slow drift case” *Proceedings of International Joint Conference of Neural Networks*, pp. 114-120, 2009.

[52] L. Torrey, and J. Shavlik, “Transfer Learning,” *In Handbook of Research on Machine Learning Applications: Algorithms, Methods, and Techniques*, Soria, Ed., IGI Global, 2009.

[53] E. Hüllermeier, T. Fober, and M. Mernberger, “Inductive Bias,” In *Encyclopedia of Systems Biology, O.W.K.C.a.H.Y.W. Dubitzky*, Ed., New York, NY., Springer, 2013.

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

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

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

[57] L.I. Kuncheva, “Classifier ensembles for changing environments,” In *Proceedings of 5th International Workshop of Multiple Classifier Systems*, Vols., pp. 1-15, 2004.

[58] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, “Dynamic integration of classifiers for handling concept drift,” *Information Fusion*, vol. 9, no. 1, pp. 56-68, 2008.

[59] G.E. Goldberg, and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” *Foundations of Genetic Algorithms*, pp. 69-93, 1991.

[60] 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.

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

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

[63] J. Li, FGP: A Genetic Programming-based Tool for Financial Forecasting. PhD Thesis, University of Essex, 2000.

[64] 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.

[65] J.F. Miller, and P. Thomson, "Cartesian genetic programming," In *European Conference on Genetic Programming*, pp. 121-132, 2000.

[66] M.F. Brameier, and W. Banzhaf, "Linear genetic programming," *Springer Science & Business Media*, 2007.

[67] J.R. Koza, "Concepts Formation and Tree Induction using the Genetic Programming," *Stanford University*, Department of Computer Science.

[68] Y.L. Geom, "Genetic recursive regression for modelling and forecasting real-world," *Advances in Genetic Programming*, pp. 401-423, 1999.

[69] M. Kaboudan, "Genetic evolution of regression models for business and economic," *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 1260-1268, 1999.

[70] H.A. Abass, R.A. Saker, and C.S. Newton, Data Mining: A Heuristic Approach, Idea Publishing Group, 2002.

[71] L. Vanneschi, and G. Cuccu, "A study of genetic programming variable population size for dynamic optimization problems," 2004.

[72] 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, IEEE.

[73] 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*, 2007.

[74] K. Price, R.M. Storn, and J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization," *Springer*, 2005.

[75] R. Storn, and K. Price, "Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimisation over Continuous Spaces," *Science*, vol. 11, no. 4, pp. 1-15, 1995.

[76] R. Storn, and K. Price, “Differential Evolution-A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 331-359, 1997.

[77] N.K. Madavan, “Multiobjective optimization using a Pareto differential evolution approach,” In *Proceedings of Congress on Evolutionary Computation*, vol. 2, pp. 1145-1150, 2002.

[78] S. Das, A. Abraham, K. Chakraborty, and A. Konar, “Differential evolution using a neighborhood-based mutation operator,”*IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526-553, 2009.

[79] A.K. Qin, and P.N. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” In *Proceedings of The 2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785-1791, 2005.

[80] D. Zaharie, “A comparative analysis of crossover variants in differential evolution,” In *Proceedings of 2nd International Symposium Advances in Artificial Intelligence and Applications*, pp. 171-181.

[81] I.L. Lòpez Cruz, L.G. van Willigenburg, and G. van Straten, “Efficient Differential Evolution algorithms for multimodal optimal control problems,” *Applied Soft Computing*, vol. 3, no. 2, pp. 97-122, 2003.

[82] R. Joshi, and A.C. Sanderson, “Minimal Representation Multisensor Fusion using Differential Evolution,” In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation*, pp. 255-273, 1997.

[83] A.W. Iorio, and X. Li, “Solving rotated multi-objective optimization problems using differential evolution,” In *Australasian Joint Conference on Artificial Intelligence. Springer*, Berlin, Heidelberg, pp. 861-872, 2004.

[84] A.K. Qin, V.L. Huang, and P.N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398-417, 2009.

[85] M. Varadarajan, and K.S. Swarup, “Solving multi-objective optimal power flow using differential evolution,” *IET Generation, Transmission & Distribution*, vol. 2, no. 5, pp. 720-730, 2008.

[86] J. Vesterstrom, and R. Thomsen, “A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems,” In *Evolutionary Computation, 2004. CEC2004*, pp. 1980-1987, 2004.

[87] T. Blackwell, “Particle Swarm Optimisation in Dynamic Environments,” *Evolutionary Computation in Dynamic and Uncertain Environments*, vol. 51, pp. 29-49, 2007.

[88] H. Raza, P Nandal, and S. Makker, “Selection of cluster-head using PSO in CGSR protocol,” *International Conference on Methods and Models in Computer Science (ICM2CS-2010)*, 2010.

[89] R.M. Satheesh, P. Asokan, and S. Kumanan, "Design of Loop Layout in Flexible Manufacturing System using Non-traditional Optimisation Technique," London, Springer-Verlag London Limited, 2007.

[90] R.S. Kadadevaramath, "Production and Distribution Scheduling of Supply Chain Structure using Intelligent Particle Swarm Optimisation Algorithm," *International Journal of Intelligent Systems Technologies and Applications*, 2009.

[91] M. Clerc, and J. Kennedy, "The Particle Swarm-explosion Stability and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[92] I. Trelea, "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317-325, 2003.

[93] F. van den Bergh, "An Analysis of Particle Swarm Optimizers," *PhD thesis*, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2012.

[94] F. van den Bergh, and A.P Engelbrecht, "Effects of Swarm Size on Cooperative Particle Swarm Optimizers," In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 892-899, 2001.

[95] A. Ismail, and A.P. Engelbrecht, "Training Product Units in Feedforward Neural Networks using Particle Swarm Optimization," In *Proceedings of the International Conference on Artificial Intelligence*, pp. 36-40, 1999.

[96] J.J. Liang, A.K. Qin, P.N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," IEEE transactions on evolutionary computation, vol. 10, no. 3, pp. 281-295, 2006.

[97] C.M. Huang, C.J. Huang, and M.L. Wang, "A particle swarm optimization to identifying the ARMAX model for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1126-1133, 2005.

[98] S. Selvan, C. Xavier, N. Karssemeijer, J. Sequeira, R. Cherian, and B. Dhala, "Parameter estimation in stochastic mammogram model by heuristic optimization techniques," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 4, pp. 685-695, 2006.

[99] A.P. Engelbrecht, "Particle Swarm Optimisation," In *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion-GECCO*, 2014.

[100] T.M. Blackwell, and P.J Bentley, "Dynamic Search with Charged Swarms," In *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, pp. 19-26, 2002.

[101] A. Carlisle, and G. Dozler, "Tracking Changing Extrema with Adaptive Particle Swarm Optimiser," In *Proceedings of the 5th Biannual World Automation Congress*, 2002.

[102] S. Yang, and X. Yao, "A Comparative Study on Particle Swarm Optimization in Dynamic Environments," In *Evolutionary Computation for DOPs*, Berlin Heidelberg, Springer-Verlag, 2013, pp. 109-136.

[103] T. Blackwell and J. Branke, "Multi-swarm Optimisation in Dynamic Environments," *Applications of Evolutionary Computing*, vol. 3005, pp. 489-500, 2004.

[104] T.M. Blackwell, and P.J Bentley, "Don't Push Me Collision-Avoidance Swarms," In *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1691-1696, 2002.

[105] T. Blackwell, and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459-472, 2006.

[106] G. Gan, C. Ma, and J. Wu, Data clustering: theory, algorithms, and applications, vol. 20, Siam, 2007.

[107] F.E. Clements, "Use of cluster analysis with anthropological data," *American Anthropologist*, vol. 56, no. 2, pp. 180-199, 1954.

[108] R.C. Eberhart and Y. Shi, Computational intelligence: concepts to implementations, chapter 1, Kaufmann Publishers, 2007.

[109] A.K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651-666, 2010.

[110] A.K Jain, and D.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, 1998.

[111] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141-182, 1997.

[112] M.E. Celebi, ed., Partitional clustering algorithms, Springer, 2014.

[113] S. Ray, and R.H. Turi, "Determination of the Number of Clusters in K-means Clustering Application in Colour Image Segmentation," In *Proceedings of the 4th International Conference on Pattern Recognition and Digital Techniques*, 1999.

[114] D.L. Davies, and D.W. Bouldin, "A Clustering Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224-227, 1979.

[115] J.C. Dunn, "Well Separated Clusters and Optimal Fuzzy Partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95-104, 1974.

[116] L. Kaufman, and P. J. Rousseeuw, "Partitioning Around Medoids," In *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 2009, pp. 86-89.

[117] C.H. Chou, M.C. Su, and E. Lai, "A New Cluster Validity Measure and its Application to Image Compression," *Pattern Anal Application*, Springer Verlag London, vol. 7, pp. 205-220, 2004.

[118] Y. Xiong, and D. Yeung, "Time Series Clustering with ARMA Mixtures," *Pattern Recognition* - Science Direct, 2004.

[119] L. Rokach, and O. Maimon, "Clustering Methods," In *Data mining and knowledge discovery handbook*, Boston, MA., Springer, 2005, pp. 321-352.

[120] M.B. Al-Zoubi, and M.A. Rawi, "An Efficient Approach for Computing Silhouette Coefficients, " *Journal of Computer Science*, vol. 4, no. 1, pp. 252-255, 2008.

[121] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys* (CSUR), vol. 31, no. 3, pp. 264-323, 1999.

[122] D.W. van der Merwe, and A.P. Engelbrecht, "Data Clustering Using Particle Swarm Optimisation," *Department of Computer Science*, University of Pretoria, 2003.

[123] C.Y. Lee, and E.K. Antonsson, "Dynamic partitional clustering using evolution strategies," In *Proceedings of 26th Annual Conference of the IEEE Industrial Electronics Society*, vol. 4, pp. 2716-2721, 2000.

[124] G. Das, H.Mannila, and P. Smyth, "Rule Discovery from Time Series," *KDD*, pp. 16- 22, 1998.

[125] R. Dajani, M. Miquel, M.C. Forilini, and P. Rubel, "Modeling of Ventricular Repolarization Time Series by Multi-layer Perceptrons," In *8th Conference on Artificial Intelligence in Medicine in Europe, AIME.*, pp. 152-155, 2001.

[126] E. Keogh, and P. Smyth, "A probabilistic Approach to Fast Pattern Matching in Time Series Databases," *KDD*, pp. 24-30, 1997.

[127] T. Oates, "Identifying Distinctive Subsequences in Multivariate Time Series by Clustering, " *KDD*, pp. 322-326, 1999.

[128] P. Kalnis, N. Mamoulis, and S. Bakiras, "On the Discovering Moving Clusters in Spatio-temporal Data, " In *Proceedings of 9th International Symposium on Spatial and Temporal Databases*, 2005.

[129] M.G.H. Omran, A.P. Engelbrecht, and A. Salman, "Dynamic clustering using particle swarm optimization with application in unsupervised image classification," *Transactions on Engineering, Computing and Technology*, vol. 9, pp. 199-204, 2005.

[130] C. Li, and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," In *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 439-446, 2009.

[131] S.C.M. Cohen, and L.N. de Castro, "Data clustering with particle swarms," In *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1792-1798, 2006.

[132] R. Mendes and A. S. Mohais, "Dynde: A Differential Evolution for Dynamic Optimization Problems," In *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2808-2815, 2005.

[133] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with the external archive for optimization in dynamic environments," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 881-897, 2013.

[134] S. Paterlini, and T. Krink, "High-performance clustering with differential evolution," In *Proceedings of Congress on Evolutionary Computation*, vol. 2, pp. 2004-2011, 2004.

[135] 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.

[136] S. Wu, A.W.C Liew, H. Yan, and M. Yang, "Cluster analysis of gene expression data based on self-splitting and merging competitive learning," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, no. 1, pp. 5-15, 2004.

[137] C.J. Veenman, M.J.T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1273-1280, 2002.

[138] H. Sun, S. Wang, and Q. Jiang, "Fcm-based model selection algorithms for determining the number of clusters," *Pattern Recognition*, vol. 37, no. 10, pp. 2027-2037, 2004.

[139] K. Do-Jong, "A novel validity index for determination of the optimal number of clusters," *IEICE Transactions on Information and Systems*, vol. 84, no. 2, pp. 281-285, 2001.

[140] K. Georgieva, and A.P. Engelbrecht, "Cooperative DynDE for Temporal Data Clustering," *IEEE World Congress on Computational Intelligence*, 2014.

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

[142] E. Mohammed, and K. Mohamed, "A Taxonomy of Cooperative Search Algorithms," *Dept. of Electrical and Computer Engineering*, University of Waterloo.

[143] L. Dio san, and M. Oltean, "Evolutionary design of evolutionary algorithms," *Genetic Programming and Evolvable Machines*, vol. 10, no. 3, pp. 263-306, 2009.

[144] H.G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," *Technical Report AIC-90-001, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, D.C.*, 1990.

[145] X. Li, and H.D. Khanh, "Comparing particle swarms for tracking extrema in dynamic environments," In *Proceedings of the Congress on Evolutionary Computation*, vol. 3, pp. 1772-1779, 2003.

[146] D.P. Solomatine, and K.N. Dulal, "Model trees as an alternative to neural networks in rainfall-runoff modelling," *Hydrological Sciences Journal*, vol. 48, no. 3, pp. 399-411, 2003.

[147] H. Drucker, C.J. Burges, L. Kaufman, A.J. Smola, and V. Vapnik, "Support vector regression machines," *In Advances in neural information processing systems*, pp. 155-161, 1997.

[148] M. Awad, and R. Khanna, "Support vector regression," *In Efficient Learning Machine*, Berkeley, CA, Apress, 2015, pp. 67-80.

[149] S.M. Abdullah, A.I.M. Yassin, and N.M. Tahir, "Particle Swarm Optimization and Least Squares Estimation of NARMAX," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 22, 2015.

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

[151] Z. Lu, C. Yang, D. Qin, Y. Luo, and M. Momayez, "Estimating ultrasonic time-of-flight through echo signal envelope and modified Gauss-Newton method," *Measurement*, vol. 94, pp. 355-363, 2016.

[152] P. Erdoĝmus, and S. Ekiz "Nonlinear Regression using Particle Swarm Optimization and Genetic Algorithm," *International Journal of Computer Applications*, vol. 153, no. 6, 2016.

[153] T. Özel, and Y. Karpat, "Identification of constitutive material model parameters for high-strain rate metal cutting conditions using evolutionary computational algorithms," *Materials and manufacturing processes*, vol. 22, no. 5, pp. 659-667, 2007.

[154] S. Cheng, C. Zhao, J. Wu, and Y. Shi, "Particle Swarm Optimization in Regression Analysis: A Case Study," In *International Conference in Swarm Intelligence*, pp. 55-63, Springer, Berlin, Heidelberg, 2013.

[155] S. Wilson, "Function approximation with a classifier system," In *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, pp. 974-98, 2001.

[156] P.J. Angeline, "Evolving predictors for chaotic time series," In *Proceedings of SPIE: Application and Science of Computational Intelligence*, vol. 3390, pp. 170-180, 1998.

[157] N. Nikolaev, and H. Iba, "Genetic programming using Chebyshev polynomials," In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 89-96, 2001.

[158] 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.

[159] 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, p. 499-510, 2016.

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

[161] R. Schwaerzel, and T. Bylander, "Predicting Financial Time Series by Genetic Programming with Trigonometric Functions and High-Order Statistics," *GECCO*, 2006.

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

[163] A. Hui, "Using genetic programming to perform time-series forecasting of stock prices," *Genetic Algorithms and Genetic Programming at Stanford*, pp. 83-90, 2003.

[164] M. Kaboudan, "Forecasting with computer-evolved model specifications: a genetic programming application," *Computer and Operations Research*, vol. 30, pp. 1661-81, 2003.

[165] N. Wagner, and Z. Michalewicz, "Genetic programming with efficient population control for financial times series prediction," *Genetic and Evolutionary Computation Conference Late Breaking Papers*, San Francisco, CA., vol. 1, pp. 458-62, 2001.

[166] S. Demeyer, "Research Methods in Computer Science," In *ICSM*, pp. 600, 2011.

[167] G. Dodig-Crnkovic, "Scientific methods in computer science," In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, Skovde, Suecia, pp. 126-130, 2002.

[168] B.J. Oates, "Researching information systems and computing," *Sage*, 2005.

[169] C. Johnson, "What is Research in Computing Science," 2006a. [Online: 6 August 2020]. Available:
`http://www.dcs.gla.ac.uk/j̃ohnson/teaching/research_skills/research.html`.

[170] C. Johnson, "What is research in computing science," Computer Science Dept, Glasgow University, 2006b. [Online: Accessed 6 August 2020]. Available:
`http://www.dcs.gla.ac.uk/j̃ohnson/teaching/research_skills/research.html`.

[171] T. Nyathi, Automated Design of Genetic Programming Classification Algorithms, PhD Thesis: University of Kwazulu-Natal, 2018.

[172] G. Susman, and R. Evered, "An assessment of the scientific merits of action research," *Administrative science quarterly*, pp. 582-603, 1978.

[173] R. Baskerville, and T. Wood-Harper, "A critical perspective on action research as a method for information systems research," *Journal of Information Technology*, vol. 11, no. 3, pp. 235-246, 1996.

[174] M. Lòpez-Ibàñez, J. Dubois-Lacoste, L. Pèrez Càceres, T. Stutzle, and M. Birattari, "The irace package: Iterated Racing for Automatic Algorithm Configuration, " *Operations Research Perspectives*, vol. 3, pp. 43-58, 2016.

[175] T. Benkedjouh, K. Medjaher, N. Zerhouni, and S. Rechak, "Health assessment and life prediction of cutting tools based on support vector regression," *Journal of Intelligent Manufacturing*, vol. 26, no. 2, pp. 213-223, 2015.

[176] K. Mohammadi, S. Shamshirband, M.H. Anisi, K.A. Alam, and D. Petkovic, "Support vector regression-based prediction of global solar radiation on a horizontal surface," *Energy Conversion and Management*, vol. 91, pp. 433-441, 2015.

[177] Y.-H. Pao, S.M. Phillips, and D.J. Sobajic, "Neural-net computing and the intelligent control of systems," *International Journal of Control*, vol. 56, pp. 263-289, 1992.

[178] Y. Ren, P.N. Suganthan, N. Srikanth, and G. Amaratunga, "Random vector functional link network for short-term electricity load demand forecasting," *Information Sciences*, vol. 367, pp. 1078-1093, 2016.

[179] X. Qiu, P. N. Suganthan, and G. A. Amaratunga, "Ensemble incremental learning random vector functional link network for short-term electric load forecasting," *Knowledge-Based Systems*, vol. 145, pp. 182-196, 2018.

[180] J. Che, and J. Wang, "Short-term electricity prices forecasting based on support vector regression and auto-regressive integrated moving average modeling," *Energy Conversion and Management*, vol. 51, no. 10, pp. 1911-1917, 2010.

[181] 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.

[182] J. Stock and M. Watson, "Forecasting Inflation," *Journal of Monetary Economics*, vol. 44, pp. 293-335, 1999.

[183] Data-Feed ToolBox, April 2018. [Online]. Available: www.mathworks.com.

[184] L. Bennett, L. Swartzendruber, and H. Brown, "Superconductivity Magnetization Modeling," *NIST*, 1994.

[185] 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.

[186] R.W. Morrison, and K.A. De Jong "A test problem generator for non-stationary environments," In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). IEEE.*, pp. 2047-2053, 1999.

[187] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, pp. 1875-1882, 1999.

[188] Y. Jin, and B. Sendhoff, "Constructing dynamic optimization test problems using the multiobjective optimization concept," *Workshops on Applications of Evolutionary Computation 2004, LNCS 3005*, pp. 526-536, 2004.

[189] C. Li, M. Yang, and L. Kang, "A new approach to solving dynamic TSP," In *Proceedings of the 6th International Conference on Simulated Evolution and Learning*, pp. 236-243, 2006.

[190] C. Li, and S. Yang "A Generalized Approach to Construct Benchmark Problems for Dynamic Optimization," In *Proceedings of the 8th International Conference on Simulated Evolution and Learning*, 2008.

[191] 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.

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

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

[194] A. Bjorck, Numerical methods for least squares problems, vol. 51, Siam, 1996.

[195] D. Parrott, and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 440-458, 2006.

[196] D. Gumus, E. Ozcan, and J. Atkin, "An investigation of tuning a memetic algorithm for cross-domain search," In *Evolutionary Computation (CEC) 2016 IEEE Congress*, pp. 135-142, 2016.

[197] Australian energy market operator. (2019). Retrieved from `http://www.aemo.com.au/`

[198] Australian bureau of meteorology. (2019). Retrieved from `http://www.bom.gov.au/`

[199] A. Fentis, L. Bahatti, M. Tabaa, and M. Mestari, "Short-term nonlinear autoregressive photovoltaic power forecasting using statistical learning approaches and insitu observations, " *International Journal of Energy and Environmental Engineering*, 10(2), pp. 189-206, 2019.

[200] J. Clegg, J. F. Dawson, S. J. Porter, & M. H. Barley, "The use of a genetic algorithm to optimize the functional form of a multi-dimensional polynomial fit to experimental data."In *IEEE Congress on Evolutionary Computation* vol. 1, pp. 928-934, 2005.

Appendix 1. P-values

Table 8.1: P-values for DynPSO Experiments on Frequencies

		$R^2_{a_1}$	$R^2_{a_2}$	$R^2_{a_3}$	$R^2_{a_4}$	$R^2_{a_5}$	E_{MS_1}	E_{MS_2}	E_{MS_3}	E_{MS_4}	E_{MS_5}
QR-QPSO vs QPSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0218	**0.0857**	0.0396	0.0284	0.0492
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0131	0.0169	0.0411	0.0052	0.0394
QR-QPSO vs QR-rPSO	$E_{MS}T$	0.0045	0.0492	0.0341	0.0056	0.0013	0.0449	0.0490	0.0093	0.0124	0.0311
	$E_{MS}G$	0.0467	0.0112	0.0098	0.0007	0.0167	0.0001	0.0292	0.0058	0.0008	0.0139
QR-QPSO vs rePSO	$E_{MS}T$	0.0496	0.0004	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0227	0.0437
	$E_{MS}G$	0.0001	0.0001	0.0474	0.0069	0.0009	0.0001	0.0015	0.0001	0.0011	0.0001
QR-QPSO vs mPSO	$E_{MS}T$	0.0404	0.0357	0.0001	0.0372	0.0265	0.0001	0.0001	0.0001	0.0009	0.0001
	$E_{MS}G$	0.0050	0.0001	0.0362	0.0038	0.0067	0.0001	0.0453	0.0084	0.0001	0.0380
QR-QPSO vs QR-mPSO	$E_{MS}T$	0.0273	0.0173	0.0284	0.0418	0.0161	**0.0952**	0.0481	0.0098	**0.8107**	0.0311
	$E_{MS}G$	0.0383	0.0398	0.0024	0.0105	0.0009	0.0001	0.0073	0.0308	0.0273	0.0414
QPSO vs QR-rPSO	$E_{MS}T$	0.0001	0.0001	0.0394	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QPSO vs rePSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QPSO vs QR-mPSO	$E_{MS}T$	0.0001	0.0374	0.0168	0.0001	0.0001	0.0012	0.0001	**0.0818**	0.0023	0.0011
	$E_{MS}G$	0.0001	0.0260	0.0023	0.0006	0.0001	0.0001	0.0001	0.0005	0.0001	0.0001
QPSO vs mPSO	$E_{MS}T$	0.0076	0.0374	0.0315	0.0005	0.0001	0.0045	0.0317	0.0066	0.0227	0.0038
	$E_{MS}G$	0.0001	0.0083	0.0061	0.0001	0.0001	0.0001	0.0358	0.0001	0.0062	0.0004
QR-rePSO vs rePSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QR-rePSO vs QR-mPSO	$E_{MS}T$	0.0101	0.0260	0.0438	0.0380	0.0001	0.0231	0.0208	0.0001	0.0004	0.0004
	$E_{MS}G$	0.0001	0.0091	0.0071	0.0149	0.0001	0.0163	0.0058	0.0001	**0.0519**	0.0058
rePSO vs QR-mPSO	$E_{MS}T$	0.0001	0.0001	0.0104	0.0001	0.0001	0.0047	0.0023	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0210	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0004	0.0001	0.0041
rePSO vs mPSO	$E_{MS}T$	0.0363	0.0304	0.0069	0.0326	0.0089	0.0036	0.0074	**0.0785**	0.0359	0.0001
	$E_{MS}G$	0.0112	0.0366	0.0147	0.0001	0.0001	0.0001	**0.0884**	0.0471	0.0001	0.0001
QR-mPSO vs mPSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0014	0.0001	0.0051	0.0056	0.0001	0.0001	0.0018	0.0211	0.0001	0.0037
QR-rPSO vs mPSO	$E_{MS}T$	0.0337	0.0001	0.0001	0.0456	0.0001	0.0005	0.0164	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0046	0.0001	0.0001	0.0168	0.0001	0.0001	0.0049	0.0001	0.0001	0.0007

Table 8.2: P-values for DynPSO Experiments on Severity

		$R^2_{a_1}$	$R^2_{a_2}$	$R^2_{a_3}$	$R^2_{a_4}$	$R^2_{a_5}$	E_{MS_1}	E_{MS_2}	E_{MS_3}	E_{MS_4}	E_{MS_5}
QRP-QSO vs QPSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0218	0.0457	0.0396	**0.9284**	0.0492
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0131	0.0169	0.0411	0.0052	0.0394
QR-QPSO vs QR-rPSO	$E_{MS}T$	0.0045	0.0492	0.0341	0.0056	0.0013	0.0449	0.0490	0.0093	0.0124	**0.0711**
	$E_{MS}G$	0.0467	0.0112	0.0098	0.0007	0.0167	0.0001	0.0292	0.0058	0.0008	0.0139
QR-QPSO vs rePSO	$E_{MS}T$	0.0496	0.0004	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0227	0.0437
	$E_{MS}G$	0.0001	0.0001	0.0474	0.0069	0.0009	0.0001	0.0015	0.0001	0.0011	0.0001
QR-QPSO vs mPSO	$E_{MS}T$	0.0404	0.0357	0.0001	0.0372	0.0265	0.0001	0.0001	0.0001	0.0009	0.0001
	$E_{MS}G$	0.0050	0.0001	0.0362	0.0038	0.0067	0.0001	0.0453	0.0084	0.0001	0.0380
QR-QPSO vs QR-mPSO	$E_{MS}T$	0.0273	0.0173	0.0284	0.0418	0.0161	0.0452	0.0481	0.0098	0.0107	0.0311
	$E_{MS}G$	0.0383	0.0398	0.0024	0.0105	0.0009	0.0001	0.0073	0.0308	0.0273	0.0414
QPSO vs QR-rPSO	$E_{MS}T$	0.0001	0.0001	0.0394	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QPSO vs rePSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QPSO vs QR-mPSO	$E_{MS}T$	0.0001	0.0374	0.0168	0.0001	0.0001	0.0012	0.0001	**0.0818**	0.0023	0.0011
	$E_{MS}G$	0.0001	0.0060	0.0023	0.0006	0.0001	0.0001	0.0001	0.0005	0.0001	0.0001
QPSO vs mPSO	$E_{MS}T$	0.0076	0.0374	0.0315	0.0005	0.0001	0.0045	0.0317	0.0066	0.0227	0.0038
	$E_{MS}G$	0.0001	0.0083	0.0061	0.0001	0.0001	0.0001	0.0358	0.0001	0.0062	0.0004
QR-rePSO vs rePSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
QR-rePSO vs QR-mPSO	$E_{MS}T$	0.0101	**0.0960**	0.0380	**0.0638**	0.0001	0.0231	0.0208	0.0001	**0.6304**	0.0004
	$E_{MS}G$	0.0001	0.0071	0.0791	**0.2749**	0.0001	0.0163	0.0058	0.0001	0.0519	0.0042
rePSO vs QR-mPSO	$E_{MS}T$	0.0001	0.0001	0.0104	0.0001	0.0001	0.0047	0.0023	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0210	0.0001	0.0106	0.0063	0.0001	0.0001	0.0001	0.0004	0.0001	0.0041
rePSO vs mPSO	$E_{MS}T$	0.0363	0.0304	0.0069	0.0704	0.0089	0.0036	0.0074	0.0185	0.0359	0.0008
	$E_{MS}G$	0.0112	0.0241	**0.0647**	0.0001	**0.5628**	0.0001	0.0884	0.0001	**0.0971**	0.0001
QR-mPSO vs mPSO	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0014	0.0001	0.0051	0.0056	0.0001	0.0001	0.0018	0.0211	0.0001	0.0037
QR-rPSO vs mPSO	$E_{MS}T$	0.0337	0.0001	0.0001	0.0456	0.0001	0.0005	0.0164	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0046	0.0001	0.0001	0.0168	0.0001	0.0001	0.0049	0.0001	0.0001	0.0007

Table 8.3: P-values for Comparative Experiments of DynPSO with Benchmarks

	QR-QPSO vs SVR	QR-QPSO vs RVFL	QRrePSO vs SVR	QRrePSO vs RVFL	QRmPSO vs SVR	QRmPSO vs RVFL	SVR vs RVFL
RMSE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MAPE	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Table 8.4: P-values for DynPSO Hybrid Experiments

	RMSE	MAPE
QR vs. NARX	0.0402	0.0316
QR vs. QR-QPSO	0.0001	0.0001
QR vs. NARX-QPSO	0.0001	0.0001
QR vs. QPSO	0.0194	0.0049
NARX vs. QR-QPSO	0.0001	0.0001
QPSO vs. QR-PSO	0.0001	0.0001
NARX vs. QPSO	0.0158	0.0261
NARX vs. NARX-QPSO	0.0001	0.0001
QPSO vs. NARX-QPSO	0.0001	0.0001
QR-QPSO vs. NARX-QPSO	0.0026	0.0175

Table 8.5: P-values for Comparative Experiments of GPANDA with Benchmarks

		DynGP vs. DyFor GP (Progressive)					DynGP vs. DyFor GP (Abrupt)				
		1	2	3	4	5	1	2	3	4	5
A	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	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	0.0001
	$E_{MS}G$	0.0001	0.0105	0.0163	0.0001	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	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	**0.1068**
D	$E_{MS}T$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	$E_{MS}G$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	**0.2413**
		DynGP vs. DyFor GP (Random)					DynGP vs. DyFor GP (Elect)				
A	$E_{MS}T$	0.0001	0.0001	0.0001			**0.0511**	**0.2175**	**0.1217**	**0.2458**	0.0351
	$E_{MS}G$	**0.0853**	0.0047	0.0009			**0.1094**	**0.3445**	**0.0762**	**0.0923**	0.0014
B	$E_{MS}T$	0.0001	0.0001	0.0001			**0.6284**	**0.9582**	**0.0812**	**0.0681**	0.0007
	$E_{MS}G$	**0.3610**	**0.7731**	0.0218			**0.6396**	**0.1264**	**0.0515**	**0.0893**	**0.0719**
C	$E_{MS}T$	0.0001	0.0001	0.0001			0.0250	0.0374	0.0116	0.0317	0.0428
	$E_{MS}G$	**0.0962**	0.0453	**0.2193**			**0.4475**	**0.2856**	**0.0886**	**0.1432**	**0.3822**
D	$E_{MS}T$	0.0001	0.0001	0.0001			0.0001	0.0001	0.0001	0.0379	0.0411
	$E_{MS}G$	0.0001	0.0001	0.0001			0.0001	0.0001	0.0001	**0.3140**	**0.1247**
		GPANDA vs. DyFor GP (GDP)					DynGP vs. GPANDA (GDP)				
A	$E_{MS}T$	**0.7848**	**0.0589**	**0.1738**			0.0001	0.0001	0.0001		
	$E_{MS}G$	0.0001	0.0001	0.0001			0.0001	0.0001	0.0001		