Time Series Forecasting using Dynamic Particle Swarm Optimizer Trained Neural Networks

by

Salihu Aish Abdulkarim

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

December 2018

Publication data:

Salihu Abdulkarim Bloggs. Time Series Forecasting using Dynamic Particle Swarm Optimizer Trained Neural Networks. Doctoral thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, December 2018.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

http://upetd.up.ac.za/UPeTD.htm

Time Series Forecasting using Dynamic Particle Swarm Optimizer Trained Neural Networks

by

Salihu Abdulkarim Bloggs

E-mail: sakwami@gmail.com

Abstract

Time series forecasting is a very important research area because of its practical application in many fields. Due to the importance of time series forecasting, much research effort has gone into the development of forecasting models and in improving prediction accuracies. The interest in using artificial neural networks (NNs) to model and forecast time series has been growing. The most popular type of NN is arguably the feedforward NN (FNN). FNNs have structures capable of learning static input-output mappings, suitable for prediction of non-linear stationary time series. To model non-stationary time series, recurrent NNs (RNNs) are often used. The recurrent/delayed connections in RNNs give the network dynamic properties to effectively handle temporal sequences. These recurent/delayed connections, however, increase the number of weights that are required to be optimized during training of the NN.

Particle swarm optimization (PSO) is an efficient population based search algorithm based on the social dynamics of group interactions in bird flocks. Several studies have applied PSO to train NNs for time series forecasting, and the results indicated good performance on stationary time series, and poor performance on non-stationary and highly noisy time series. These studies have assumed static environments, making the original PSO, which was designed for static environments, unsuitable for training NNs for forecasting many real-world time series generated by non-stationary processes.

In dealing with non-stationary data, modified versions of PSOs for optimization in dynamic environments are used. These dynamic PSOs are yet to be applied to train NNs on forecasting problems. The first part of this thesis formulates training of a FNN forecaster as a dynamic optimization problem, to investigate the application of a dynamic PSO algorithm to train FNNs in forecasting time series in non-stationary environments. For this purpose, a set of experiments were conducted on ten forecasting problems under nine different dynamic scenarios. Results obtained are compared to the results of FNNs trained using a standard PSO and resilient backpropagation (RPROP). The results show that the dynamic PSO algorithm outperform the PSO and RPROP algorithms. These findings highlight the potential of using dynamic PSO in training FNNs for real-world forecasting applications.

The second part of the thesis tests the hypothesis that recurrent/delayed connections are not necessary if a dynamic PSO is used as the training algorithm. For this purpose, set of experiments were carried out on the same problems and under the same dynamic scenarios. Each experiment involves training a FNN using a dynamic PSO algorithm, and comparing the result to that obtained from four different types of RNNs (i.e. Elman NN, Jordan NN, Multi-Recurrent NN and Time Delay NN), each trained separately using RPROP, standard PSO and the dynamic PSO algorithm. The results show that the FNNs trained with the dynamic PSO significantly outperform all the RNNs trained using any of the algorithms considered. These findings show that recurrent/delayed connections are not necessary in NNs used for time series forecasting (for the time series considered in this study) as long as a dynamic PSO algorithm is used as the training method.

Keywords: Time Series, Feedforward Neural Networks, Recurrent Neural Networks, Particle Swarm Optimization, Cooperative Quantum Particle Swarm Optimization, Dynamic Environment.

Supervisors	**:** Prof. A. P. Engelbrecht
Department	**:** Department of Computer Science
Degree	**:** Doctor of Philosophiae

"Our world is increasingly complex, often chaotic, and always fast-flowing. This makes forecasting something between tremendously difficult and actually impossible, with a strong shift toward the latter as timescales get longer."

Andrew McAfee

"When the number of factors coming into play in a phenomenological complex is too large scientific method in most cases fails. One need only think of the weather, in which case the prediction even for a few days ahead is impossible."

Albert Einstein

Acknowledgements

I would like to express my sincere gratitude to the following people for their assistance during the production of this thesis:

- Prof Andries P. Engelbrecht for his mentorship, invaluable guidance and support throughout my research journey;
- The members of CIRG who helped with their opinions and technical knowledge;
- My family for their unwavering support and love.

Contents

List of Figures **v**

List of Algorithms **viii**

List of Tables **ix**

1 Introduction **1**

1.1 Motivation 2

1.2 Objectives 4

1.3 Contributions 4

1.4 Thesis Outline 5

2 Time Series Forecasting **7**

2.1 Time Series 8

2.1.1 Characteristics of Time Series 8

2.1.2 Time Series Analysis 12

2.1.3 Stochastic Processes in Time Series 13

2.2 Time series forecasting methods 13

2.2.1 Statistical Methods 14

2.2.2 k-Nearest Neighbours Technique 15

2.2.3 Support Vector Machines 16

2.2.4 Artificial Neural Networks 18

2.3 Summary 19

3 Artificial Neural Networks **20**
3.1 Neural Network Forecasters 21
3.1.1 Feedforward Neural Networks 21
3.1.2 Time Delay Neural Networks 23
3.1.3 Simple Recurrent Neural Networks 23
3.2 Neural Network Training 25
3.2.1 Backpropagation Algorithm 26
3.2.2 Resilient Propagation Algorithm 26
3.2.3 Particle Swarm Optimization 27
3.3 Issues in Modeling Neural Network Forecaster 28
3.3.1 Data Collection and Preparation 28
3.3.2 Selection of Network Type and Architecture 29
3.3.3 Weight Initialization 32
3.4 Summary 32

4 Particle Swarm Optimization **33**
4.1 Basic PSO Algorithm 34
4.2 Control Parameters of Particle Swarm Optimization Algorithms 36
4.3 Applications of Particle Swarm Optimization 38
4.4 Summary 39

5 Particle Swarm Optimization in Dynamic Environments **40**
5.1 Dynamic Optimization Problems 41
5.1.1 Characteristics of Dynamic Environments 42
5.1.2 Types of Dynamic Environments 43
5.2 Standard PSO in Dynamic Environments 45
5.3 Change Detection and Response 46
5.3.1 Change Detection 46
5.3.2 Response to Change 47
5.4 Particle Swarm Optimizers for Dynamic Environments 48
5.4.1 Reinitialising Particle Swarm Optimization 49
5.4.2 Charged Particle Swarm Optimization 49

5.4.3 Quantum Particle Swarm Optimization . . . 50
5.4.4 Multiswam Particle Swarm Optimization . . . 50
5.4.5 Cooperative Particle Swarm Optimization . . . 51
5.5 Summary . . . 52

6 Training Feedforward Neural Network Forecasters using Dynamic Particle Swarm Optimizer 53
6.1 Experimental Procedure . . . 54
6.1.1 Datasets . . . 55
6.1.2 Dataset Preparation . . . 59
6.1.3 Simulating Dynamic Environments . . . 60
6.1.4 Parameter Selection . . . 62
6.1.5 Performance Measure . . . 64
6.2 Results . . . 65
6.2.1 SAM Time Series . . . 65
6.2.2 HIT Time Series . . . 70
6.2.3 DMT Time series . . . 71
6.2.4 Mackay Glass . . . 75
6.2.5 Lorenz Time Series . . . 79
6.2.6 IAP Time Series . . . 83
6.2.7 S&P 500 Time Series . . . 88
6.2.8 AWS Time Series . . . 91
6.2.9 USD Time Series . . . 96
6.2.10 LM Time Series . . . 101
6.3 Summary . . . 107

7 Recurrent Connections: Are They Necessary? 109
7.1 Methodology . . . 109
7.1.1 Datasets . . . 110
7.1.2 Parameter Selection . . . 110
7.2 Results . . . 112
7.2.1 SAM Time Series . . . 112

7.2.2 HIT Time Series 121
7.2.3 DMT Time Series 129
7.2.4 MG Time Series 137
7.2.5 Lorenz Time Series 144
7.2.6 IAP Time Series 153
7.2.7 S&P Time Series 160
7.2.8 AWS Time Series 167
7.2.9 USD Time Series 175
7.2.10 LM Time Series 182
7.3 Summary 193

8 Conclusions **194**
8.1 Summary of Conclusions 194
8.2 Future Work 196

Bibliography **199**

A Acronyms **215**

B Symbols **217**
B.1 Chapter 2 217
B.2 Chapter 3 218
B.3 Chapter 4 218
B.4 Chapter 5 219
B.5 Chapter 6 219

C Derived publications **220**

List of Figures

2.1 Linear trend 9
2.2 Seasonal variation 10
2.3 Cyclical variation 10
2.4 Random variation 11
2.5 SVM sepration of data points 17

3.1 A feedforward neural network forecaster 22
3.2 Time-Delay neuron 23
3.3 Simple recurrent neural networks 24
3.4 Multirecurrent neural network 25

6.1 Time plot of four datasets used 56
6.2 Time plot of six datasets used 58
6.3 Sliding Time Window 60
6.4 Training and generalization error results for SAM time series, scenarios B1 to B3 69
6.5 Training and generalization error Results for HIT Time Series, Scenarios A1 to A3 74
6.6 Training and generalization error Results for DMT Time Series, Scenarios C1 to C3 78
6.7 Training and generalization error Results for MG Time Series, Scenatios A1 to A3 82
6.8 Training and generalization error Results for Lorenz Time Series, Scenarios C1 to C3 86

6.9 Training and generalization error results for IAP time series, scenarios C1 to C3 90
6.10 Training and generalization error results for S&P time series, scenarios C1 to C3 94
6.11 Training and generalization error Result for AWS Time Series, scenarios A1 to A3 98
6.12 Training and generalization error results for USD time series, scenarios B1 to B3 102
6.13 Training and generalization error Results for LM Time Series, Scenarios B1 to B3 106

7.1 Training and generalization error progression in predicting SAM time series under scenario A1 114
7.2 Training and generalization error results for SAM time series, scenario B2 117
7.3 Training and generalization error results for SAM time series, scenario C3 119
7.4 Training and generalization error results for HIT time series, scenario A2 123
7.5 Training and generalization error results for HIT time series, scenario B2 125
7.6 Results of HIT time series, scenarios C1 to C3 127
7.7 Training and generalization error results for HIT time series, scenario C2 128
7.8 Training and generalization error results for DMT time series, scenario A1 131
7.9 Training and generalization error results for DMT time series, scenario B2 132
7.10 Training and generalization error results for DMT time series, scenario C3 134
7.11 Training and generalization error results for MG time series, scenario A1 139
7.12 Training and generalization error results for MG time series, scenario B2 141
7.13 Training and generalization error results for MG time series, scenario C2 143
7.14 Training and generalization error results for Lorenz time series, scenario A1 145
7.15 Training and generalization error results for Lorenz time series, scenario B1 149
7.16 Training and generalization error results for Lorenz time series, scenario C1 152
7.17 Training and generalization error results for AIP time series, scenario A1 153
7.18 Training and generalization error results for IAP time series, scenario B1 157
7.19 Training and generalization error results for IAP time series, scenario C1 159
7.20 Training and generalization error results for SP time series, scenario A3 . 162

7.21 Training and generalization error results for SP time series, scenario B3 . 163
7.22 Training and generalization error results for SP time series, scenario C3 . 167
7.23 Training and generalization error results for AWS time series, scenario A3 170
7.24 Training and generalization error results for AWS time series, scenario B3 172
7.25 Training and generalization error results for AWS time series, scenario C3 173
7.26 Training and generalization error results for USD time series, scenario A2 178
7.27 Training and generalization error results for USD time series, scenario B2 180
7.28 Training and generalization error results for USD time series, scenario C2 183
7.29 Training and generalization error results for LM time series, scenario A1 184
7.30 Training and generalization error results for LM time series, scenario B2 . 188
7.31 Training and generalization error results for LM time series, scenario C3 192

List of Algorithms

1 Basic PSO 35

List of Tables

6.1 Dynamic scenarios settings for each dataset 61
6.2 Input nodes for synthetic datasets adopted from previous studies 63
6.3 PSO parameter ranges considered 64
6.4 SAM Time Series Results 66
6.5 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 67
6.6 SAM time series Algorithm Ranking for Scenarios A to C 68
6.7 HIT Time Series Results 71
6.8 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 72
6.9 HIT Time Series Algorithm Ranking for Scenarios A to C 73
6.10 DMT Time Series Results 75
6.11 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%, for scenarios A1 to A3 76
6.12 DMT Time Series Algorithm Ranking for Scenarios A to C 77
6.13 MG Time Series Results 79
6.14 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 80

6.15 MG Time Series Algorithm Ranking for Scenarios A to C 81
6.16 Lorenz Time Series Results 83
6.17 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 84
6.18 Lorenz Time Series Algorithm Ranking for Scenarios A to C 85
6.19 IAP Time series Results 87
6.20 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 88
6.21 AIP Algorithm Ranking for scenarios A to C 89
6.22 S&P Time Series Results 91
6.23 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 92
6.24 S&P Time Series Algorithm Ranking for Scenarios A to C 93
6.25 AWS Time Series Results 95
6.26 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 96
6.27 AWS Algorithm Ranking for scenarios A to C 97
6.28 USD Time Series Results 99
6.29 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 100
6.30 USD Time Series Algorithm Ranking for Scenarios A to C 101
6.31 LM Time Series Results 103
6.32 Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95% 104
6.33 LM Time Series Algorithm Ranking for Scenarios A to C 105

6.34 Average Algorithm Ranking Overall Time Series 107

7.1 Neural network parameters for each dataset 111
7.2 Results of SAM time series, scenarios A1 to A3 113
7.3 Models ranking in forecasting SAM time series, scenarios A1-A3 114
7.4 Results of SAM time series, scenario B1 to B3 116
7.5 Models ranking in forecasting the SAM time series, scenarios B1 to B3 . 117
7.6 Results of SAM time series, scenario C1 to C3 118
7.7 Models ranking in forecasting the SAM time series, scenarios C1 to C3 . 120
7.8 Overall models ranking in forecasting the SAM time series, scenarios A to C 121
7.9 Results of HIT time series, scenario A1 to A3 122
7.10 Models ranking in forecasting the HIT time series, scenarios A1 to A3 . . 123
7.11 Results of HIT time series, scenarios B1 to B3 124
7.12 Models ranking in forecasting the HIT time series, scenarios B1 to B3 . . 126
7.13 Models ranking in forecasting the HIT time series, scenarios C1 to C3 . . 128
7.14 Models ranking in forecasting the HIT time series, scenarios A to C . . . 129
7.15 Result of DMT time series, scenarios A1 to A3 130
7.16 Models ranking in forecasting the DMT time series, scenarios A1 to A3 . 132
7.17 Results of DMT time series, scenario B1 to B3 133
7.18 Models ranking in forecasting the DMT time series, scenarios B1-B3 . . . 134
7.19 Results of DMT time series, scenario C1 to C3 135
7.20 Models ranking in forecasting the DMT time series, scenarios C1 to C3 . 136
7.21 Overall ranking of the models in predicting the DMT problem, scenarios A to C 137
7.22 Results of MG time series, scenario A1-A3 138
7.23 Models ranking in forecasting the MG time series, scenarios A1 to A3 . . 139
7.24 Results of MG time series, scenario B1 to B3 140
7.25 Models ranking in forecasting the MG time series, scenarios B1 to B3 . . 141
7.26 Results of MG time series, scenario C1 to C3 142
7.27 Models ranking in forecasting the MG time series, scenarios C1-C3 144
7.28 Overall ranking of the models in predicting the MG problem, scenarios A to C 145

7.29 Results of Lorenz time series, scenario A1 to A3 146
7.30 Models ranking in forecasting the Lorenz time series, scenarios A1 to A3 147
7.31 Results of Lorenz time series, scenario B1 to B3 148
7.32 Models ranking in forecasting the Lorenz time series, scenarios B1 to B3 149
7.33 Results of Lorenz time series, scenario C1 to C3 150
7.34 Models ranking in forecasting the Lorenz time series, scenarios C1 to C3 151
7.35 Overall models ranking in forecasting the Lorenz time series, scenario error A-C 152
7.36 Results of IAP time series, scenario A1 to A3 154
7.37 Models ranking in forecasting the IAP time series, scenario A1 to A3 . . 155
7.38 IAP time series results for scenarios B1 to B3 156
7.39 Models ranking in forecasting the IAP time series, scenario error B1-B3 . 157
7.40 IAP Results for scenarios C1-C3 158
7.41 Models ranking in forecasting the IAP time series, scenario C1-C3 159
7.42 Overall models ranking in forecasting the IAP time series, scenario error A-C 160
7.43 Results of S&P time series, scenario A1 to A3 161
7.44 Models ranking in forecasting the S&P time series, scenarios A1 to A3 . 162
7.45 Results for S&P time series, scenarios B1-B3 164
7.46 Models ranking in forecasting the S&P time series, scenarios B1 to B3 . 165
7.47 Results of S&P time series, scenario C1 to C3 166
7.48 Models ranking in forecasting the S&P time series, scenarios C1 to C3 . 167
7.49 Overall models average ranking in forecasting the S&P time series, scenarios A to C 168
7.50 Results of AWS time series, scenarios A1 to A3 169
7.51 Models ranking in forecasting the AWS time series, scenarios A1 to A3 . 170
7.52 Results of AWS time series, scenarios B1 to B3 171
7.53 Models ranking in forecasting the AWS time series, scenarios B1 to B3 . 172
7.54 Results of AWS time series, scenarios C1 to C3 174
7.55 Models ranking in forecasting the AWS time series, scenarios C1 to C3 . 175

7.56 Overall models' ranking in forecasting the AWS time series, scenarios A to C . . . 176
7.57 Results of USD time series, scenarios A1 to A3 . . . 177
7.58 Models ranking in forecasting the USD time series, scenarios A1 to A3 . . . 178
7.59 Results of USD time series, scenarios B1 to B3 . . . 179
7.60 Models ranking in forecasting the USD time series, scenarios B1 to B3 . . . 181
7.61 Results of USD time series, scenarios C1 to C3 . . . 182
7.62 Models ranking in forecasting the USD time series, scenarios C1 to C3 . . . 183
7.63 Overall models ranking in forecasting the USD time series, scenarios A to C 184
7.64 Results of LM time series, scenario A1 to A3 . . . 185
7.65 Models ranking in forecasting the LM time series, scenarios A1 to A3 . . . 186
7.66 Results of LM time series, scenario B1 to B3 . . . 187
7.67 Models ranking in forecasting the LM time series, scenarios B1 to B3 . . . 189
7.68 Results of LM time series, scenario C1 to C3 . . . 190
7.69 Models ranking in forecasting the LM time series, scenarios C1 to C3 . . . 191
7.70 Overall model's average ranking in forecasting the LM time series, scenarios A to C . . . 192

Chapter 1

Introduction

Forecasting is an essential task in dealing with uncertainties. Most decisions taken by businesses, governments, and people are based on forecasts. Forecasting in general, involves estimating or predicting future events (values) of a time based sequence of historical data (time series). The oldest and simplest method used in forecasting is human judgment, and is often subject to errors. These errors may be within acceptable limits when the data to be predicted is well understood. When the forecasting problem is relatively more complex and less understood, other methods that require little or no human judgment become more appropriate. A time series forecasting problem becomes increasingly more complex in an environment with continuously shifting conditions. In such conditions, a method that can adapt to the changing environment is required.

Due to the importance of time series forecasting, much research effort has gone into the development of forecasting models and in improving prediction accuracies. Statistical methods were the largely used time series forecasting tools [30]. Statistical methods are, however, known to have two main drawbacks, namely, high mathematical complexity and dependence on prior knowledge of how the time series was generated [76, 101]. Neural networks (NNs), as an alternative approach to time series modeling, address these issues and have shown to be more efficient in terms of prediction accuracy [4, 5].

The interest in using NNs to model and forecast time series has been growing. The most popular type of NN is arguably the feedforward NN (FNN). FNNs have structures capable of learning static input-output mappings [49], suitable for prediction of non-linear stationary time series. A stationary time series is one whose unconditional joint probability distribution does not change when shifted in time. To model non-stationary time series, NN forecasters are often implemented using recurrent/delayed connections to give the network dynamic properties to effectively handle temporal sequences. The NNs implimented with recurrent connections are generally referred to as recurrent NNs (RNN).

Even though NNs have done well in many time series forecasting problems, their performance depends on the selection of the right network architecture and an appropriate training algorithm [3, 153].

Particle swarm optimization (PSO) is an efficient population based search algorithm based on the social dynamics of group interactions in bird flocks. PSO is now an established method for training NNs, and was shown to outperform the classical back-propagation (BP) training algorithm in a number of specific applications [128, 129, 151]. The original PSO was, however, designed for static environments. In dealing with non-stationary data, modified versions of the PSOs designed for optimization in dynamic environments are used.

This thesis applies dynamic PSO to training NN forecasters for non-stationary time series modelling.

Motivation for this thesis is given in Section 1.1. Section 1.2 highlight the research objectives. The contributions of the study are given in Section 1.3. Outline of the thesis is given in Section 1.4.

1.1 Motivation

A study of the NN literature reveals that researchers and analysts arbitrarily choose the type of NN they use in forecasting applications. While the FNN is the type dominantly used, recent trends indicate that RNNs are now the widely used architecture because of their ability to model dynamic systems implicitly. The recurent/delayed connections in RNNs, however, increase the number of weights that are required to be optimized during training of the NN.

To build a NN forecaster using RNNs, one has to choose among the different special cases of RNNs (such as Elman NN, Jordan NN, or Multirecurrent NN). It is, however, very difficult to single out a particular special case of the RNNs that is better in modeling different types of time series [1]. Thus, selecting which RNN to use becomes a problem. Another issue that needs to be resolved is how to decide on the number of context layers if an Elman NN is used, or the number of state layers if a Jordan NN is used, or how to decide on the number of previous values to remember if a time delay NN is used.

Even though several studies have applied PSO to the task of training NN forecasters, producing favorable results, these studies have assumed static environments, making them unsuitable for many real-world time series which are generated by varying processes. Since PSO was developed for optimization in a stationary environment, it seems more appropriate to use PSO variants designed for solving problems in dynamic environments in order to train NN forecasters under non-static environments. For example, the performance of PSO as a training algorithm for FNNs in forecasting non-stationary time series was evaluated in [3], where the authors considered the problem as a static optimization problem. The results showed that PSO performed better than BP, but concluded that the result of PSO could significantly be improved by using a suitable NN structure (that can handle dynamic data). Jar *et al* [76] compared the performance of PSO and BP in training FNNs for time series forecasting. The results indicate superiority of the PSO over BP, but Jar *et al* concluded that the PSO trained FNNs were unable to track non-stationary data.

Dynamic PSOs have been used to train NNs on classification problems under non-stationary environments [42, 108, 110]. These studies showed that dynamic PSOs are indeed applicable to NN training on classification problems for dynamic environments, by yielding significantly better or similar results compared to the classic BP algorithm and the original PSO.

To the knowledge of the author, there is no study yet that applies a dynamic PSO algorithm to train NN forecasters for non-static environments. Thus, if dynamic PSO is efficiently applicable to training FNN forecasters for non-stationary evironments, then it is possible that the recurrent/delayed connections (required for handling the temporal component of a non-stationary time series) are no longer necessary in NN forecasters for dynamic environments. If this holds true, then the problems associated with using

RNNs to model time series under non-stationary environments are resolved by using FNN trained with dynamic PSO.

1.2 Objectives

The primary objectives of this thesis can be summarized as follows:

- To formulate training of a NN forecaster as a dynamic optimization problem.
- To investigate the applicability and performance of a dymanic PSO as a training algorithm for FNN forecasters for non-stationary environments.
- To investigate if recurrent/delayed connections are necessary for a NN time series forecaster when a dynamic PSO is used as the training algorithm.

1.3 Contributions

The main contributions of the this thesis are:

- The formulation of training a NN forecaster for non-stationary time series as a dynamic optimization problem.
- The first analysis of the applicability of a dynamic PSO algorithm to the training of NN forecasters under different kinds of dynamic environments.
- The implementation of the RPROP algorithm, and addition of the algorithm to CIlib [102].
- An empirical analysis comparing the performance of a cooperative quantum PSO to the RPROP and the standard PSO algorithms in training FNNs and simple recurrent NN forecasters under different dynamic environmental scenarios.
- An empirical analysis showing that recurrent/delayed connections are not necessary for NN forecasters if a dynamic PSO is used as the training method.

1.4 Thesis Outline

Chapter two provides an overview of time series forecasting. The chapter starts by defining a time series, and discussions on the characteristic components of a time series. This is followed by discussions on time series analysis and applications, which includes forecasting, classification, explanation, and transformation. In this context, discussions on the most important and widely used time series forecasting methods are provided, which includes autoregressive integrated moving average, k-nearest neighbour, support vector regression, and NNs.

Chapter three reviews different types of NNs used in modeling time series. This includes FNNs, time delay NNs, Elman NNs, Jordan NNs and multi-recurrent NNs. The chapter also discusses a number of optimization methods applied to training NNs, namely BP, resilient propagation, and PSO. This is followed by discussions on the major performance issues in designing NN forecasters.

Chapter four provides a review of the basic PSO algorithm. The chapter also describes the application of PSO to the training of NNs.

Chapter five briefly discusses dynamic optimization problems, and the different characteristics of dynamic environments. This is followed by a discussion of the challenges faced by standard PSO when applied to dynamic environments, and methods of addressing these challenges. Existing PSO approaches to dynamic optimization are then discussed.

Chapter six formulates training of a NN as a dynamic optimization problem to investigate the applicability and efficiency of a dynamic PSO algorithm in training FNN forecasters for non-stationary environments. The chapter provides a detailed description of the experimental procedure followed, which includes detailed descriptions of the datasets, data pre-processing methods, dynamic environments considered, parameter optimization process, and performance measures. Experimental results are then presented and analysed, and conclusions are given.

Chapter seven investigates the need for recurrent/delayed connections for NNs used for non-stationary time series forecasting if a dynamic PSO is used as the training algorithm. Results from the experiments carried out are presented and discussed, and conclusions are given.

Chapter eight summarises the findings of this thesis, followed by a discussion of future research directions.

The appendices provide a list of acronyms used throughout the text along with their definitions, a list of symbols used in this thesis, and a list of publications derived from the work discussed in this thesis.

Chapter 2

Time Series Forecasting

"It is difficult to make predictions, especially about the future."

– NEILS BOHR

Forecasting is the process of estimating or predicting future events (or values) of a time based sequence of historical data (called a time series) in order to deal with future uncertainties. Most decisions taken by businesses, governments, and people are based on forecasts, and the quality of those decisions largely depends on the prediction errors. The oldest and simplest method used in forecasting is human judgment, and is often subject to errors. These errors may be within acceptable limits when the data to be predicted is well understood. When the time series problem is relatively more complex and less understood, other methods that require little or no human judgment become more appropriate. The time series problem becomes increasingly more complex in an environment with continuously shifting conditions.

This chapter looks at time series forecasting methods from a more general point of view in order to provide a basis for the methods studied in this thesis. To effectively do that, some background information on what a time series is, its characteristics, and its analysis are discussed.

In the remainder of the chapter, Section 2.1 gives an overview on time series and Section 2.2 discusses time series forecasting methods.

2.1 Time Series

A time series is a collection of observations measured at discrete times. The order in which time series observations are taken is very crucial. Mathematically, a time series is defined as a set of vectors, $\mathbf{x}(t)$, $t = 1, 2, \cdots, n$, where t is the time interval and n is the number of observations. Some examples of time series are records of the annual numbers of students enrolled at the university of Pretoria between 1990 and 2017, and records of the daily minimum temperature in Johannesburg between 1st January, 2016 and 31st December, 2016.

A time series is said to be discrete if observations are measured at discrete points of time, usually at an equally spaced interval such as monthly, quarterly, or yearly. If the observations are measured at every instance of time, the time series becomes continuous. A continuous time series can be easily transformed to a discrete one by merging data together over a specified time interval. The selected time interval determines the number of data points in the time series, referred to as the sampling frequency. The sampling frequency has an influence on the main characteristics of the resulting time series.

A time series containing a single variable is called univariate. When a time series contains more than one variable, it is called multivariate. Univariate time series processing forms the basis of this work, and as such, all discussions henceforth, are limited to it.

In the remainder of this section, Section 2.1.1 describes the characteristic features of time series, and Section 2.1.2 discusses time series analysis. Stochastic processes in time series is are discussed in Section 2.1.3.

2.1.1 Characteristics of Time Series

To process a time series for any application, certain components (features) that characterize the series have to be considered. These features are:

- The trend, which indicates an average increase or decline in the values of a time series over a long-term period. Figure 2.1 shows an example of what a trend looks like, where the dotted line indicates a long-term linear trend. A nonlinear trend may also be observed in a time series. An upward trend is commonly seen in

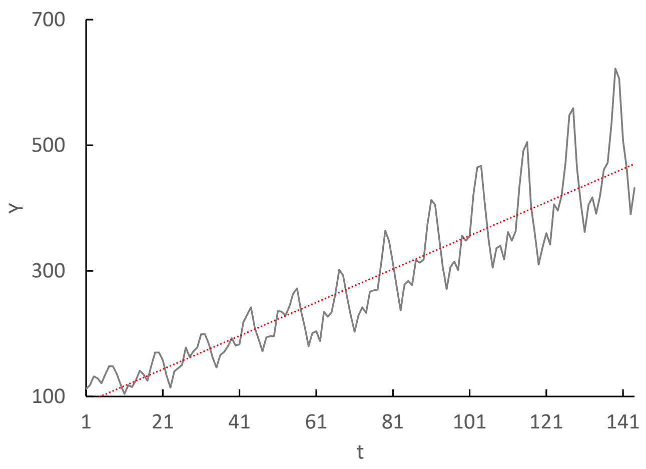


Figure 2.1: Linear trend

time series relating to population growth and water consumption in a community, whereas a downward trend may be observed in series related to epidemics and illiteracy rate.

- Seasonality, which is a regular fluctuation in time series values over fixed and known periods. Seasonal movement may be observed quarterly, monthly, weekly or even daily, and the seasonality is either deterministic or stochastic. The appearance of seasonal variation is exemplified in Figure 2.2. Seasonal fluctuations may be due to customs, preferences, weather, and climate changes and technologies.

- Cyclical variations, which are regular changes in time series values over an unknown variable period (typically more than one year period). Cyclical variations are caused by events which occur in cycles. Figure 2.3 shows an example of a time series with cycles of approximately 10 years (some lasting eight, nine or 10 years). In contrast to seasonality, cyclical variation covers a longer time period and has a systematic pattern that is not easily predictable. Cyclical variations may be seen in most financial and economic series.

- Random variations, also called residual errors, are irregular movements in a time series caused by unpredictable circumstances that are not regular and do not repeat in a particular pattern. They are caused by events such as outbreak of disease,

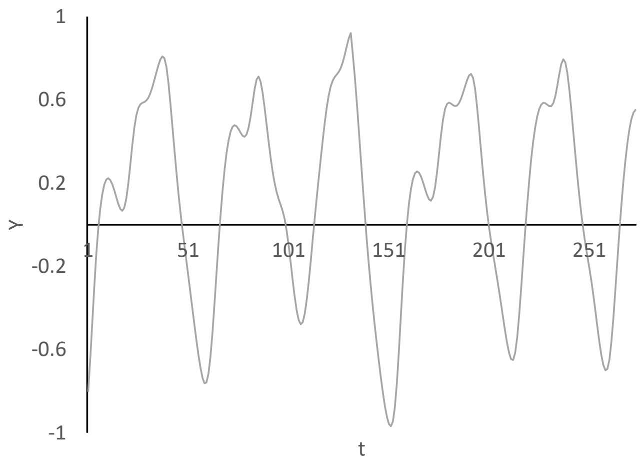


Figure 2.2: Seasonal variation

revolution, war, or strikes. Figure 2.4 illustrates an example of a series with random variations. Random variations are not predictable.

Other features such as outliers and missing observations are often observed in a time series. Outliers are data observations that deviate substantially from the data distribution, causing large errors on models constructed from the data. Outliers are normally removed using statistical techniques during data preparation before model construction. Even though such action will alleviate the problems of outliers, other important informa-

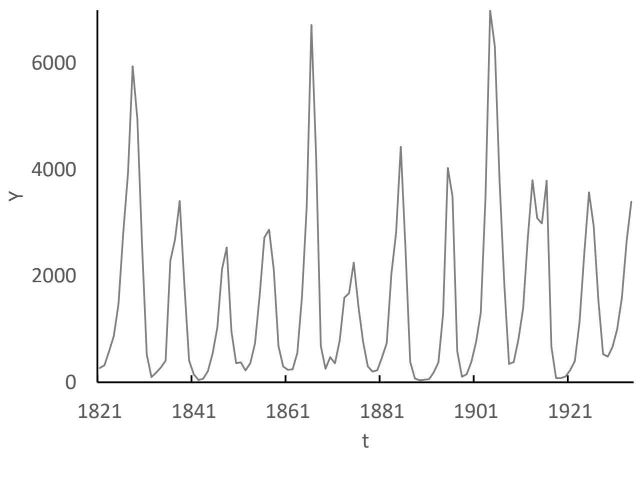


Figure 2.3: Cyclical variation

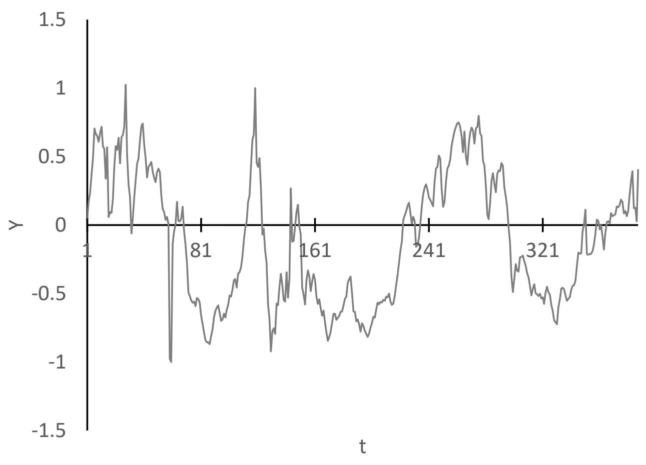


Figure 2.4: Random variation

tion may be lost at the same time. Chatfield [30] noted that treatment of some specific features such as outliers, missing observations, or possible errors can be more important than the choice of forecasting method.

A plot of a time series reveals the presence of different features contained in the series, and is generally used in studying and assessing the properties of a time series.

Since time series usually have different combinations of characteristic components, decomposition is employed in classical time series analysis to separately describe these components. Based on the way that these components are combined, linear modeling can be classified in additive, multiplicative, and pseudo additive compositions [29].

An additive model,

$$Y_t = T + S + R \tag{2.1}$$

where T, S and R are the trend, seasonal and random variations respectively, assumes that the characteristic components are independent of one another.

A multiplicative model,

$$Y_t = T \times S \times R \tag{2.2}$$

on the other hand assumes that the characteristic components are not necessarily independent of one another. Pseudo additive decomposition has features of both the additive and the multiplicative models.

2.1.2 Time Series Analysis

Time series analysis is the process of using a mathematical model to extract meaningful statistics or other characteristics from a time series in order to understand its underlying structure and functions that produces it. Some typical applications in time series analyses are:

- forecasting, which is the process of estimating future values of a time series based on previous ones;
- classification, which is the process of grouping time series into a number of classes;
- explanation, which is the process of describing a time series in terms of the parameters of a model; and
- transformation, which is the process of mapping one time series onto another.

Forecasting applications are the most widely spread and imminent in literature due to their huge importance in many fields. Most strategic decisions are often based on forecast results. Therefore, fitting a model that can adequately model a time series is quite important.

In modeling a time series forecaster, the series values are analyzed to develop a mathematical model that captures the underlying data generating process of the series. The mathematical model developed is then used to predict future values of the series. The analysis is based on the fact that successive observations are not independent and their chronological order must be taken into account.

As described in [40], time series forecasting is the problem of finding a function $f : \mathbb{R}^n \to \mathbb{R}$ such as to obtain an estimate $\mathbf{y}(t+d)$ of $\mathbf{y}$ at time $(t+d)$, given n observations of $\mathbf{y}$ up to time t:

$$\mathbf{y}(t+d) \approx f(\mathbf{y}(t), \mathbf{y}(t-1), \ldots, \mathbf{y}(t-n)) \tag{2.3}$$

where d is the prediction lag (also called the forecasting horizon). This implies that time series forecasting is a function approximation problem.

2.1.3 Stochastic Processes in Time Series

A time series is said to be deterministic if exact future values can be determined. However, in most time series applications, future values can not be predicted with certainty due to residual error, ϵ, which is as the result of noise processes (produced randomly from uncontrollable influencing factors). Future values of a series have a probability distribution which is conditioned by knowledge of past values,

$$\mathbf{y}(t+d) = f(\mathbf{y}(t), \mathbf{y}(t-1), \cdots) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{2.4}$$

where $\mathcal{N}$ is a zero-mean normal distribution with a variance of σ^2.

A mathematical expression that describes the probability structure of a time series is called a stochastic process [66]. Therefore, a time series is actually a sample realization of a stochastic process that produces the series [30]. Observations of a time series are usually assumed to be independent and identically distributed (i.i.d), following a normal distribution. However, this assumption is not exactly correct, because a time series exhibits a more or less regular pattern in the long run [30], except for a chaotic time series.

2.2 Time series forecasting methods

In over a half century's active research in the area of time series analysis, many time series forecasting methods were developed. These methods range from simple techniques such as naive forecast (which involves setting the forecast to the last time series observation) to more complex computational intelligence (CI) methods such as artificial neural networks (NNs). This section provides an overview of the time series forecasting methods most relevant to this thesis, widely used and often cited in the literature, in order to provide a basis for the remainder of the thesis.

Section 2.2.1 discusses statistical methods used in modelling time series. Section 2.2.2 describes the k-nearest neighbour technique and its application in time series forecasting. Support vector regression is discussed in Section 2.2.3. Section 2.2.4 discusses artificial neural networks for time series processing.

2.2.1 Statistical Methods

There are several statitistical techniques used in time series modelling. The prominent among these techniques are the moving average (MA) [20], exponential smoothing [55], auroregressive (AR) [20], autoregressive moving average (ARMA) [91], and autoregressive integrated moving average (ARIMA) [20, 21]. Chartfield [30] observed that statistical methods were the largely used time series forecasting tools. Among all the statistical techniques, ARIMA is more popular and the most widely used [21].

The ARIMA model was developed by Box and Jenkins in 1970 [20], and became very popular and important in forecasting due to the practical approach (called Box-Jenkin's methodology) developed for building the model. In ARIMA, the future value of a time series is assumed to be a linear function of the series' past observations plus a random error. Hence, the underlying function that produces the time series is expressed as

$$\hat{y}_t = \phi_0 + \epsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \tag{2.5}$$

where $\hat{y}_t$ is the forecasted value, ϵ_t is the random error with the property of zero mean and constant variance; ϕ_i and θ_j are the model parameters.

From Equation (2.5), if the value of $q = 0$, the model reduces to a pure AR model of order p. If $p = 0$, the model becomes an MA model of order q. Therefore, in building an ARIMA model, the main task is to find the appropriate order, (p, q), of the model. Box-Jenkins' methodology describes three iterative steps in order to select a satisfactory ARIMA model:

- *Model identification step*, which entails preprocessing the data series for better modeling and the determination of the order (p, q) of the ARIMA model. If the time series is non-stationary, it is reduced to a stationary series via simple differencing of degree, $\eth$, until there are no obvious patterns such as a trend or seasonality left in the series. Differencing means taking the difference between consecutive observations. Once the data is stationary, the order (p, q) of the model is determined.

- *Parameter estimation step*, which deals with optimizing the model parameters, ϕ_i and θ_j, such that the overall measure of error is minimized using nonlinear optimization techniques such as maximum likelihood estimation and non-linear least square methods.

- *Diagnostic checking step*, which involves checking the goodness of fit of the model to the series. The model is then used for forecasting if the model is adequate. If the model is found to be inadequate, the three Box-Jenkins's steps are repeated until a better alternative model is identified.

Even though ARIMA models are very popular, they are constrained to handling only linear time series. To overcome this issue, many non-linear statistical models such as the threshold autoregressive (TAR) model [134], the auto regressive conditional heteroscedasticity (ARCH) model [51], the generalized auto regressive conditional heteroscedasticity (GARCH) model [18], and the non-linear autoregressive (NAR) model [154] were proposed. These models are capable of handling non-linear time series, but their mathematical complexities is high and they also largely depend on specific knowledge of how the time series is generated [76, 101].

2.2.2 k-Nearest Neighbours Technique

The k-nearest neighbour (k-NN) [52] is a simple nonparametric learning method used in classification and forecasting. The k-NN algorithm uses local neighbourhoods to make a decision, based on the assumption that objects that are near each other share similar characteristics [34]. Therefore, the training data is considered a feature metric space. In k-NN, the training set is memorized and no model is associated to the learning concept.

To forecast a time series using the k-NN technique, the series is organised into k subsequences that are similar to the most recent subsequence before the point of forecast. Thus, histories with similar dynamic behavior are detected in the past series and used later in forecasting the next point at the end of the series.

Three pre-determined parameters are required in k-NN forecasting, namely the embedding dimension, m, the sampling frequency (i.e. delay time), τ, and the number of nearest neighbours, k. Once these parameters are determined, the given time series, say $(y_1, y_2, \ldots, y_t, \ldots, y_T)$, is transformed into subsequences of equal lengths as vectors, containing m observations sampled from the series at intervals of τ, i.e.

$$\mathbf{y}_t^{m,T} = (y_t, y_{t-\tau}, \ldots, y_{t-(m-1)\tau}); \quad t = m, m+1, \ldots, T \tag{2.6}$$

These vectors are often called m-history vectors [118]. The k-NN employ k m-history vectors that have similar dynamic behavior as the delay vector, $\mathbf{y}_T^{m,T}$, to predict the time series value at the next time step, $t = T+1$, i.e.

$$\mathbf{y}_T^{m,T} = (y_T, y_{T-\tau}, y_{T-2\tau}, \ldots, y_{T-(m-1)\tau}) \tag{2.7}$$

Selection of the k m-histories similar to the delay vector is based on a similarity measure, typically a distance function such as the Euclidean distance. Recently, Durbin and Rumelhart [104] showed that the Mahalanobis distance is more appropriate because of possible correlation among vectors at different time frames, compared to distances based on deterministic vectors, such as the Euclidean distance. Thus, the closest k history vectors that minimizes the distance function, $\mho(y_T^{m,T}, y_t^{m,T})$, are selected.

The value at $t = T+1$ is then predicted as the average (if there are no outliers, otherwise the median) of the target output values on the closest k history vectors. Other simple methods, such as simple weighted averaging [64], and local linear regression [90] are also used.

The cost of learning in k-NN is zero and no assumptions are required about the characteristics of the concepts to learn. The k-NN also employs simple local approximation procedures to learn complex concepts. However, the learned concepts have no description (which means that the model can not be interpreted). Another drawback of k-NN is that finding k nearest neighbours becomes computationally expensive for large data sets. Also, the technique suffers from the curse of dimensionality (i.e. the non-intuitive properties of data observed when working in high-dimensional space).

2.2.3 Support Vector Machines

A support vector machine (SVM) [19, 33] is a supervised learning method use for classification purposes. When SVM is used for forecasting, it is referred to as support vector regression (SVR) [41]. SVMs are based on the idea of constructing a hyperplane that best separates a dataset into two distinct classes as shown in Figure 2.5.

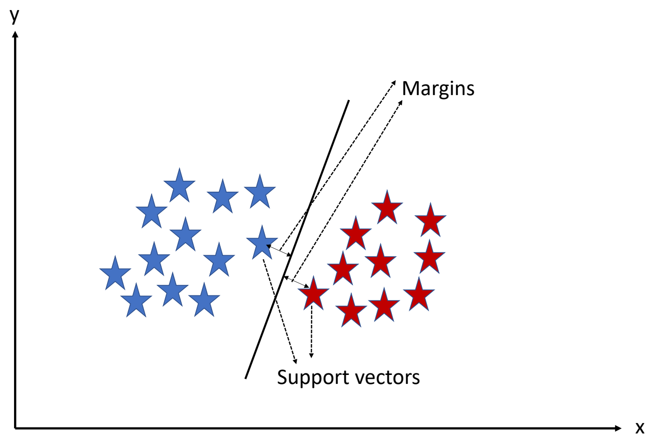


Figure 2.5: SVM sepration of data points

The data points closer to the either side of the separating hyperplane are called *support vectors* and are considered critical elements in the dataset. To have a greater chance of correctly classifying new data, the hyperplane with the greatest possible margin (i.e. the distance between support vectors of opposing classes) is chosen. This type of linear classifier with a maximum-margin hyperplane is called a maximum-margin classifier.

In order to classify linearly non-separable data, the kernel trick is applied to the maximum-margin hyperplane. Kerneling results in mapping data into a higher dimensional space where the data is linearly separable. Thus, to classify non-linear data, a maximum-margin hyperspace is fitted in the transformed feature space.

A similar process is followed in SVR. The data series is mapped into a high dimensional feature space via nonlinear mapping, and linear regression is carried out in this space [99]. Carrying out linear regression in a high dimensional feature space corresponds to the nonlinear regression in the low dimensional input space.

SVRs have been used to forecast time series data when the underlying processes are typically nonlinear, non-stationary and not defined a-priori [82]. The main advantage of SVM is that it provides unique and globaly optimal solution, unlike other methods such as NNs [25]. SVM also provides accurate model even with small dataset. However, the model has high algorithmic complexity and requires extensive model selection process.

For a comprehensive survey on SVMs used in forecasting, please refer to [119].

2.2.4 Artificial Neural Networks

Artificial neural networks (NNs) are the most successful and frequently used computational intelligence model in time series forecasting [86]. A traditional NN has an architecture composed of interconnected artificial neurons (nodes) organised in three layers, i.e. input, hidden and output layers. NNs are data driven models that can represent any type of nonlinear function [68]. Basically, a NN is a complex function representing a nonlinear mapping from a given input space to an output space. For accurate approximation in the mapping, the NN is required to be trained to learn patterns contained in a data set. Even though NNs have done well in many time series forecasting problems, their performance depends on the selection of the right NN architecture and appropriate training algorithm [3, 153].

In forecasting problems, a NN carries out the following mapping:

$$y_{t+1} = f(y_1, y_2, \ldots, y_t) \tag{2.8}$$

where y_{t+1} is the forecasted observation produced as the NN output and $y_1, y_2, \ldots, y_t$ are the historical values of the series fed into the NN as input. For example, a NN with nine input nodes and one output uses the series values $y_1, y_2, \ldots, y_9$ to forecast the value y_{10}. The training window then shifts one step to extract $m - 9$ training examples from the series with m observations. The NN learns to forecast y_t using $y_{t-9}, \ldots, y_{t-1}$ as input, and the process continues for several training steps until the prediction accuracy is satisfactory.

For explicit handling of the order between observations when learning a mapping function from input to output, *recurrent* NNs (RNNs) [59] are used. The addition of the temporal component of the series adds a new dimension to the function being approximated. Thus, instead of mapping input and output alone, the network gain the capability of learning the mapping function for the inputs over time to an output. This capability unlocks the time series for the neural networks.

For an extensive review on NNs applied to time series prediction, please refer to [6, 146, 153].

2.3 Summary

This chapter provided a brief overview of time series forecasting. Section 2.1 defined a time series, giving some examples, and discussed the characteristic components of a time series. These components are trend, seasonality, cyclic and random variations. A time series may have these components combined in different ways. To assess and study the properties of a given time series, a plot of the time series was used.

Time series analysis is performed in order to fit a model that adequately describes a time series. Some applications of time series analysis includes forecasting, classification, explanation, and transformation. In forecasting, historical time series values are analyzed to develop a mathematical model that captures the underlying data generating process of the series, which is then used to predict future values of the series. The future values, however, can not be predicted with certainty in most time series due to residual errors.

Section 2.2 discussed the most important and widely used time series forecasting methods, which includes ARIMA, k-NN, SVM and NNs. The next chapter reviews NN architectures and NN training algorithms.

Chapter 3

Artificial Neural Networks

"There are billions of neurons in our brains, but what are neurons? Just cells. The brain has no knowledge until connections are made between neurons. All that we know, all that we are, comes from the way our neurons are connected."
– Tim Berners-Lee, 1999

NNs are computational models inspired from the structural/functional studies of the human brain. The human brain is a highly sophisticated parallel and non-linear system that is responsible for the emergence of consciousness and complex behaviors such as perception, pattern recognition, and motor control [63]. A huge number of interconnected elementary cells called neurons constitute the human brain, and the main essence of this neural ensemble is "control through communication" [24]. The neurons communicate by sending signals via interconnections called synapses. A single neuron fires or sends a signal only if a threshold local to the neuron is exceeded, strengthening the connections of actively communicating neurons and weakening idle connections. The brain, via this neural communication, learns the mapping between the sensed input and the desired output. Thus, NNs were designed to mimic the learning process of the human brain described above.

Because of a NN's ability to learn and to generalize from experience, they have been used with success in many fields to carry out complex tasks such as classification, optimization, pattern recognition, and time series modeling [49].

In the remainder of this chapter, Section 3.1 provides an overview on NNs for forecasting. Section 3.2 describes NN training algorithms. Performance issues in designing NN forecasters are discussed in Section 3.3. Finally, the chapter is summarized in Section 3.4.

3.1 Neural Network Forecasters

NNs are mathematical models built up from weighted connections among simple processing units or nodes, called artificial neurons. In a traditional NN, these nodes are organised in three layers: input, hidden and output. A NN with enough nodes in the hidden layer is capable of approximating any non-linear function to any required accuracy [68], without the need of prior knowledge on the underlying data generation process.

NNs are non-linear, non-parametric and data driven models. Because of these salient features, NNs are now widely applied in time series processing. This section reviews NNs applied in time series forecasting.

3.1.1 Feedforward Neural Networks

Feedforward NN (FNN) is arguably the most popular type of NNs. A FNN has an acyclic structure, where information flows through the network in only one direction. FNNs were shown to be universal function approximators [36]. This implies that a FNN can approximate any function, $f\colon \mathcal{R}^n \longrightarrow \mathcal{R}^m$, i.e.

$$f(\vec{y}) = \varphi\left(\sum_{j=1}^{k} \upsilon_{j,l}\lambda\left(\sum_{i=1}^{n} w_{i,j}y_i - \beta_j\right) - \beta_l\right), \quad l = 1\dots m \tag{3.1}$$

where φ and λ are the activation functions, k is the number of hidden neurons, $\upsilon_{j,l}$ and $w_{i,j}$ are weights, and β_i, β_j are biases (thresholds). A FNN carries out approximation of non-linearity through superposition of several instances of the activation functions.

In modeling a time series using FNNs, a *time window* mechanism is employed. The window, which is a restricted part of the series, is presented to the network. After analyzing the observations in the window, the window is shifted by one or more steps further in time. Thus, the mapping of a FNN from an input window to an output can be written as

$$\hat{y}_t = f(y(t-1), y(t-2), \ldots, y(t-p)) \tag{3.2}$$

where p is size of the window, f is an arbitrary non-linear function, and $(y(t-1), y(t-2), \ldots, y(t-p))$ is sequence of observations in the series. Making f dependent on p previous observations is similar to having p input units fed with p adjacent observations of the series as shown in Figure 3.1. The time window approach reduced the temporal dimension of the series to zero by parallelizing the observations [136].

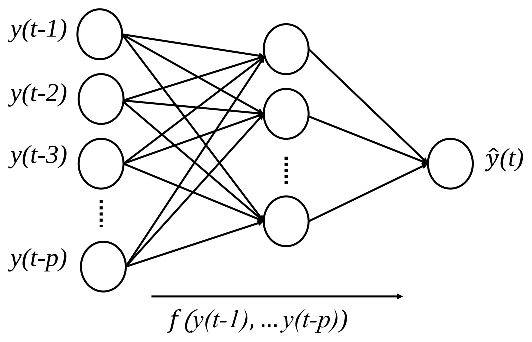


Figure 3.1: A feedforward neural network forecaster

The main advantage of using the window approach is that the underlying network architecture is not affected. The window approach may, however, result in relatively large networks, since the window dictates the number of input units and connections used in the network.

A FNN is comparable with the classical time series model, i.e. the linear autoregressive of order p (AR[p]). The AR[p] assumes linear combination of a fixed number of previous time series observations, i.e.

$$\hat{y}_t = f_L(\mathbf{y}(t-1), \ldots, \mathbf{y}(t-p)) \tag{3.3}$$

where f_L is a linear function. Apart from the non-linearity, the FNN and AR[p] models are identical [135].

3.1.2 Time Delay Neural Networks

Time delay NNs (TDNNs) [145] are a special case of FNNs designed with extra time delay connections (which is a form of memory mechanism) for effective handling of temporal data. Input patterns to the network are delayed to arrive at hidden units at different points in time. A special type of neurons with n_t delayed patterns, illustrated in Figure 3.2 are used in the input layer of a TDNN.

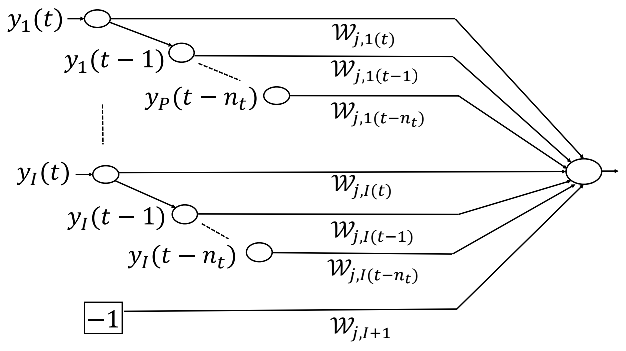


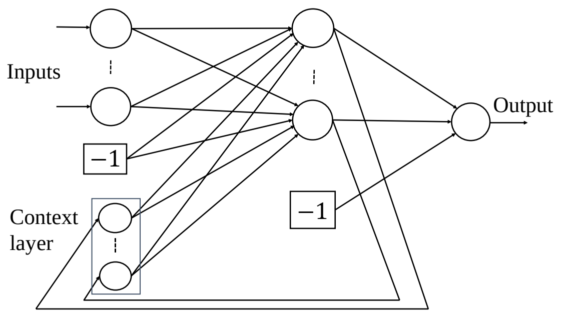
Figure 3.2: Time-Delay neuron

The TDNN is identical to a time window and can also be viewed as an autoregressive model. The major disadvantage of the TDNN is that the time period processed is strictly limited by the number and arrangement of the time delays.

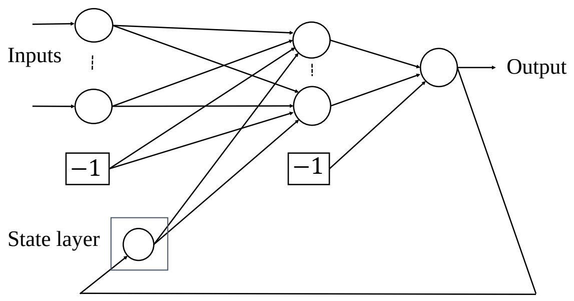
3.1.3 Simple Recurrent Neural Networks

Recurrent NNs (RNNs) are bidirectional networks specifically designed for learning sequential or time varying patterns. RNNs have feedback connections (which serves as internal memories) that allow the networks to incorporate a trace of what has been processed by the networks (i.e. past experience). The reason for using this recurrent mechanism is that, theoretically, any past observation in a time series can have an influence on the predicted future values of the series. Also, the memory capacity of networks not employing the recurrent mechanisms is limited by the size of the window or the number of delay connections.

The class of RNN models that are updated with a single update cycle (as in the FNNs) are called simple recurrent NNs (SRNNs). Among the different types of SRNNs developed, the Elman NN [48] and the Jordan NN [77] are the most common types applied in time series processing. While the Elman NN extends a FNN by adding feedback connections from the hidden layer to a *context layer* as shown in Figure 3.3a, the Jordan NN have its feedback connections from the output layer to a *state layer* as illustrated in Figure 3.3b. The *context/state* layers serve as extensions of the input layer, and are used to keep the previous state of the hidden/output layers respectively.



(a) Elman neural network



(b) Jordan neural network

Figure 3.3: Simple recurrent neural networks

It is clear from the above that the output of the Elman or the Jordan SRNN is dependent on the input, $z(t)$, and the state of the network, $s(t)$, i.e.

$$y(t) = f(z(t), s(t)) \tag{3.4}$$

Typically, the input consists of an observation from a time series

$$y(t) = f(y(t-1), s(t)) \tag{3.5}$$

where the state $s(t)$ is a function of the input $z(t-1)$ and the previous state $s(t-1)$, i.e.

$$s(t) = g(z(t-1), s(t-1)) \tag{3.6}$$

By replacing $s(t)$ in Equation 3.4 $(n-1)$ times, the output y becomes dependent on the inputs $z(t), z(t-1), \ldots, z(t-n)$,

$$y(t) = h(z(t), z(t-1), \ldots, z(t-n), s(t-n)) \tag{3.7}$$

where h is an unknown function. As part of the network, Equation 3.4 represents feed-forward mapping of all the available information ($z(t)$ and $s(t)$) to the output $y(t)$. Equation 3.6 represents the next-state function, modeling the mapping from the past state $s(t-1)$ and additional information to the current state $s(t)$.

The weights of the feedforward connections in SRNNs represent the long term memory holding all the required information, while the state provides context for the new input (as required in processing time series). The introduction of recurrence generally improves the performance of NNs in sequence prediction.

The Multi-recurrent NN (MRNN) is another type of SRNN often applied in processing time series [40]. The MRNN has feedback connections from both the hidden and the output layers to the state layer, as shown in Figure 3.4. The MRNN is simply a hybrid of Elman and Jordan SRNNs, and shares the properties of the two networks.

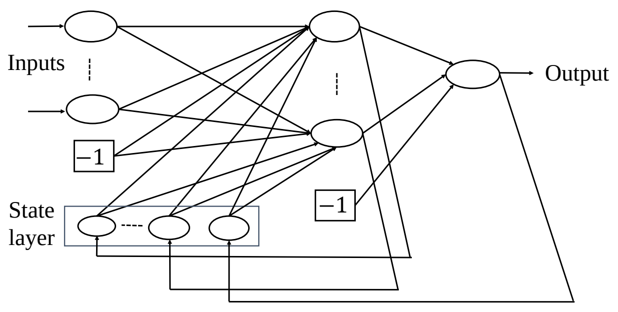


Figure 3.4: Multirecurrent neural network

3.2 Neural Network Training

Even though NNs are capable of mapping a given input pattern to an output value, they are required to be trained to learn patterns contained in a data set for accurate approximation in the mapping. The objective of training a NN is to find an optimal set of weights (including biases) that minimizes the NN error. The training can be via a supervised, unsupervised or re-enforcement process. The supervised method of training is used in NNs designed for time series prediction. In supervised training, a set of input and associated output examples (i.e. training set) is presented to a network with the aim to adjust the weight values such that the measure of difference between the actual output and the target outputs (called the error function) is minimized.

A number of optimization algorithms have been developed for training NNs [13, 14, 44, 149]. This section provides an overview on some algorithms commonly applied in training NN forecasters.

Section 3.2.1 describes the backpropagation algorithm. Section 3.2.2 discusses the resilient propagation algorithm. The particle swarm optimization algorithm is described in Section 3.2.3.

3.2.1 Backpropagation Algorithm

Backpropagation (BP) is a classical optimization algorithm for training NNs. The BP algorithm uses gradient descent to minimize the overall NN error by calculating the gradient of the training error, $\mathcal{E}$, in weight space and moving the weight vector along a negative gradient, i.e.

$$w_{i,j}(t+1) = w_{i,j}(t) + \eta \frac{\partial \mathcal{E}}{\partial w_{i,j}}(t) \tag{3.8}$$

where $w_{i,j}$ is the weight on connection from neuron i to j; η is the learning rate, a positive constant that controls the size of steps taken in the weight space. It is important to note that η has huge impact on the convergence of the algorithm: A too small value may lead to many steps before an acceptable solution is reached, making convergence extremely slow. On the other hand, if η is too large, it may lead to successive steps in the gradient descent oscillating back and forth along the gradient, making convergence difficult.

Since the weight update is not only dependent on the size of η, but also on $\frac{\partial \mathcal{E}}{\partial w_{i,j}}$, unforeseen influence of the derivative size may seriously disturb the effect of a carefully chosen η. To avoid this problem, the resilient propagation training method was developed [116]. Resilient propagation is an efficient training algorithm and is used in this work.

3.2.2 Resilient Propagation Algorithm

Resilient propagation (RPROP) [116] is an efficient supervised batch learning scheme that performs a direct adaption of the weight step based on local gradient information.

During the learning process, the sign of the partial derivative, $\frac{\partial \mathcal{E}}{\partial w_{i,j}}$, is used to dictate the direction of the weight update. The size of the weight update is exclusively determined by a weight specific update value, $\Delta_{i,j}(t)$, instead of using the size of $\frac{\partial \mathcal{E}}{\partial w_{i,j}}$ as in BP.

For each weight, $w_{i,j}$, the change in weight is determined as;

$$\Delta w_{i,j}(t) = \begin{cases} -\Delta_{i,j} & \text{if } \frac{\partial \mathcal{E}}{\partial w_{i,j}}(t) > 0 \\ +\Delta_{i,j} & \text{if } \frac{\partial \mathcal{E}}{\partial w_{i,j}}(t) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

where $\frac{\partial \mathcal{E}}{\partial w_{i,j}}(t)$ is the sum gradient information over all patterns of the training set and the new update values, $\Delta_{i,j}(t)$ are determined as follows;

$$\Delta_{i,j}(t) = \begin{cases} \eta^{+}\Delta_{i,j}(t-1) & \text{if } \frac{\partial \mathcal{E}}{\partial w_{i,j}}(t-1)\frac{\partial E}{\partial w_{i,j}}(t) > 0 \\ \eta^{-}\Delta_{i,j}(t-1) & \text{if } \frac{\partial \mathcal{E}}{\partial w_{i,j}}(t-1)\frac{\partial E}{\partial w_{i,j}}(t) < 0 \\ \Delta_{i,j}(t) & \text{otherwise} \end{cases} \tag{3.10}$$

where $0 < \eta^{-} < 1 < \eta^{+}$. Using the equations above, a weight is adjusted as

$$w_{i,j}(t+1) = w_{i,j}(t) + \Delta w_{i,j}(t) \tag{3.11}$$

Since BP and RPROP are based on gradient descent, they are both susceptible to premature convergence on a local optimum. Another disadvantage of these methods is their dependence on search starting points (i.e. initial weights of a NN).

3.2.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based search algorithm inspired by the social behaviour of birds in a flock. A PSO algorithm manipulates a swarm of particles, where each particle is a potential solution to an optimization problem. The swarm traverses a search space searching for an optimum solution. As the particles move around, each particle is attracted to both the best position it has found so far, as well as the overall best position found within its neighborhood. Due to these dynamics, a swarm that was uniformly distributed in the search space converges on a small area around the optimum position [32, 138]. Refer to [44, 49] for a detailed description of PSO.

To train a NN using PSO, each particle in the swarm represents a weight vector of the NN, and the quality of each particle is computed using the sum square error (SSE) over the training set. Unlike gradient based methods where weights and biases are updated per training pattern, PSO uses position and velocity update equations to adjust weights and biases, after which the training set is used to evaluate the quality of the particle (or NN).

3.3 Issues in Modeling Neural Network Forecaster

Designing a NN time series forecaster for a particular problem is a non-trivial task. Researchers often face problems at different design stages that affect the performance of a NN [71]. This section reviews the key issues in the design/implementation of NN forecasters.

Section 3.3.1 describes data collection and preparation issues in designing NN forecasters. Selection of network type and architecture is discussed in Section 3.3.2. Section 3.3.3 discusses issues in NN weight initialization.

3.3.1 Data Collection and Preparation

There is no explicit way of determining the size of data sample required for modeling a time series using NNs. The size usually depends on characteristics of the data series and the number of hidden layer nodes employed. Empirical analysis has shown that the accuracy of a NN forecaster improves with increased training data [100]. However, to avoid under/over fitting problems, increasing the size of the data series has to be done while considering the number of hidden layer nodes.

Preprocessing the data set is important for achieving good prediction performance. Thus, the data series should be scaled to an active range and domain of the activation functions used in the output layer. For faster convergence in backpropagation training, the data is required to be normalized after scaling by setting the mean of the dataset to be close to zero.

As in most NN applications, the dataset is normally divided into independent subsets for training and testing. The training set is used to estimate the unknown connection weights, while the test set is used to assess the generalisation ability of the trained NN. In time series applications, the data series is split such that the first subset is used for training, and the second subset is reserved for testing. The common training and testing split ratios found in the literature are 90/10, 80/20, 70/30 and 50/50, with the 70/30 being the most popular. It is a generally accepted fact that the training set should be larger than the testing set [35]. Shuffling the dataset before the split (as recommended in other NN applications) will lead to poor performance of a NN forecaster. The sequential order of the data series must be maintained.

3.3.2 Selection of Network Type and Architecture

In modeling a NN forecaster, determining the right NN architecture to use is very important and yet a difficult task. Choosing the right NN architecture involves deciding on the number of nodes in each layer, the activation functions, and the way that information flows through the network. A study of the literature reveals that researchers and analysts arbitrarily choose the type of NN they use in forecasting applications. While the FNN is the type dominantly used, recent trends indicate that RNNs are now the widely used architecture because of their ability to model dynamic systems implicitly.

To build a time series forecaster using either a FNN or RNNs, some issues have to be resolved. For example, how to make use of temporal dependencies if a FNN is used. How to decide on the number of context layers if Elman NN used, or how to decide on the number of previous values to remember if a TDNN is used. Another issue with using RNNs is that, among the different special cases of the RNNs (such as Elman NN, Jordan NN, and Multirecurrent NN), it is very difficult to single out a particular one that is better in modeling different types of time series [1]. Thus, selecting which one to use becomes a problem. This thesis hypothesizes that recurrent connections are not necessary if a dynamic particle swarm optimizer is used to train a FNN in time series forecasting. Therefore, if the hypothesis holds true, then the problem of deciding the type of NN to use as above is resolved.

Number of Input Neurons

There is no systematic procedure to find the number of input nodes for time series modeling. The number of input nodes is mostly determined via trial and error, empirically, or based on experience. Some heuristics are to use [84, 121, 131]

- four input neurons for quarterly data, and
- twelve input neurons for monthly data.

Some authors directly adopt the number of input nodes used in prior studies. There has been no consistent result in determining this parameter in the literature. Many researchers have reported the advantage of using a large number of input nodes while others report the contrary [153].

Fixing the Number Hidden Neurons

A single hidden layer is generally used because it is capable of approximating any continuous function, given enough number of neurons [68]. One of the major problems faced by researchers in implementing a NN, is the selection of the number of hidden neurons [122]. This parameter has a huge influence on the behaviour and final output of a NN and as such, must be selected carefully. Choosing a too small number of neurons in the hidden layer may result in underfitting. Underfitting happens when the neurons in the hidden layer are not enough to adequately learn the functional mapping between the input space and the output spaces. On the other hand, choosing too many neurons in the hidden layer may cause overfitting. Overfitting happens when a NN overestimates the complexity of the target problem, causing the NN to lose its generalization ability by learning the noise. Another problem of choosing too many hidden layer neurons is the increase in training time. Hence, an optimal number of neurons has to be selected.

A number of methods for selecting the optimal number of hidden neurons have been proposed over the years by several researchers. These methods can be classified into pruning, constructive, and regularization/penalty approaches. Pruning approaches start with a large enough network that is capable of capturing the functional input-output relationship, and then dispensable neurons/weights are removed. Construction approaches work in an opposite way, where a minimal possible network is optimized by adding neurons/weight to the network. Please refer to [122] for an excellent review on methods proposed in the last few decades for fixing the number of hidden neurones.

There is no generally accepted theory for determining the optimal number of hidden neurons. Several rules of thumb were proposed, however, none of them works well for all problems [153]. A simple approach is to construct a number of NNs with different architectures, compare their training and generalization performance on a given problem, and then to select the architecture with the best generalization error [49].

Deciding the Number of Previous Values to Remember

As in the case of selecting the number of hidden layer nodes, there is no generally accepted theory for determining the number of delay connections to use if TDNNs are employed or the number of context/state layers to use if SRNNs are employed. Mostly, pruning or constructive approaches (discussed in Section 3.3.2) are used to select these important parameters.

Number of Output Nodes

Deciding on the number of output nodes is relatively simple since it often corresponds to the forecasting horizon of the problem. The forecasting horizon is said to be one-step-ahead when the objective is to predict the next future value in a time series. In a one-step-ahead problem, the obvious number of output nodes required is one.

In the case of a multi-step-ahead forecasting problem, where the objective is to predict a sequence of values into the future, two strategies are often employed. The first strategy involves recursive use of one-step-ahead forecasting. In this approach, the model is constructed with a single output node, and the current predicted value is used as one of the inputs in the next forecasting iteration, replacing one of the true observations in the time series. The second approach, called the direct method, uses several (horizon-specific) output nodes to directly predict each step into the future.

The problem of selecting the best implementation strategy for the multi-step-ahead forecasting becomes an issue. Due to error accumulation (i.e. propagation of past prediction errors into future predictions) in recursive methods, the direct method is generally considered to be better. Some experimental results [11, 31] confirm the superiority of the direct method against the recursive method.

3.3.3 Weight Initialization

The connection links of a NN are first initialized with weight values which are subsequently adjusted during training to learn the functional mapping between the input and output spaces. For a time series forecaster, the function mapping does a non-linear regression. The weights (i.e. the weight matrix between each pair of layers) has an influence on the performance of a NN. When a gradient-based training algorithm is used, the initial weights serve as the starting point of the hill climbing search for an optimum on the error surface. If the initial search position is close the optimum, fast convergence is observed. However, if the initial position is on a flat region, slow convergence is experienced. Also, setting the initial weights to large values prematurely saturates neurons because large weights result in large net input signals [72]. For population based training algorithms such as genetic algorithm and PSO, the initialization should be uniform over the entire search space to ensure that different parts of the search space are covered.

A good strategy to initialize weights is to randomly generate small weight values centered around zero. This will ensure that the net input signal to a neuron is within the active domain of the activation function, so that the activation function will produce midrange output values without bias to any solution regardless of the input values. Generating the random weights in the range $(\frac{-1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}})$ was shown to be effective, where $fanin$ is the number of connections into a neuron [148].

3.4 Summary

This chapter provided an overview on how different types of NNs are used in modeling time series. The NNs discussed are FNNs, TDNNs, Elman NNs, Jordan NNs and MRNNs. A number of optimization methods applied in training NNs were discussed, namely BP, RPROP, and PSO. Performance issues in designing NN forecasters, such as data preprocessing, architecture selection and weight initialisation, were also discussed. The next chapter provides an overview of PSO.

Chapter 4

Particle Swarm Optimization

"It is the long history of humankind (and animal kind, too) that those who learned to collaborate and improvise most effectively have prevailed."

– Charles Darwin

PSO is a simple and efficient population-based search algorithm based on the social dynamics of group interactions in bird flocks. It was first introduced by Eberhart and Kennedy in 1995 [44].

In PSO, individuals called particles are initialized to move through a hyper-dimensional search space to find an optimum position (i.e. the solution). As the particles move around, each particle keeps track of the best solution it found so far. The particles also have social contact with other particles in the neighborhood, where the particles can query for the best solution found within the neighborhood. The search behaviour of each particle is therefore partly influenced by its personal experience, the knowledge of its neighbours, and also its previous search direction. Through this simple social behavior, collective effort resulted in the emerged behaviour of finding an optimal solution in the search space. Observations from natural swarms have shown that collective effort by a group is usually more rewarding than individual effort [67].

This chapter discusses the PSO algorithm, and its application in training NNs. The remainder of the chapter is structured as follows: Section 4.1 provides an overview of the basic PSO, Section 4.2 discusses the parameters of the basic PSO algorithm, Section 4.3 discusses PSO application to NNs, and Section 4.4 summarizes the chapter.

4.1 Basic PSO Algorithm

PSO manipulates a swarm of particles that share information about the best position found in the search space. Each particle in the swarm represents a potential solution to an optimization problem. For an n variable problem, the swarm operates in an n-dimensional search space. Each particle i in the swarm is represented by its position vector, $\mathbf{x}_i$ and the quality of the solution represented by the particle is evaluated using an objective function. Depending on the type of problem, the quality of the solution represented by a particle can be computed in different ways: In the case of a function minimization problem, a particle that yields a smaller value when its position is passed as a parameter to the objective function is regarded as being of better quality than a particle that yields a larger value.

The position of each particle i, at time step t, is changed by adding a velocity vector, $\mathbf{v}_i(t)$, to the current position as follows:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{4.1}$$

The entire optimization process is driven by the velocity vector, which is determined by a combination of experiential knowledge of the particle (often referred to as the cognitive component), the information shared in the neighbourhood of the particle (referred to as the social component), and the momentum term. The velocity vector is updated using

$$\mathbf{v}_i(t+1) = \omega\mathbf{v}_i(t) + c_1\mathbf{r}_1(t) \otimes (\mathbf{y}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2(t) \otimes (\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)) \tag{4.2}$$

where ω is the inertia weight [125], used to control the influence of the previous velocity on the current one; $\mathbf{v}_i(t)$ is the step size and direction; c_1 and c_2 are positive acceleration constants used to scale the influence of the cognitive component (i.e. second term) and the social component (i.e. third term) respectively; $\mathbf{r}_1(t)$ and $\mathbf{r}_2(t)$ are vectors of random values (introducing the stochastic element to the equation) sampled from a uniform distribution, $U(0,1)$; $\otimes$ is a component-wise multiplication; $\mathbf{y}_i(t)$ is the best position so far visited by particle i (known as the personal best or *pbest*); and $\hat{\mathbf{y}}_i(t)$ is the best position found by any of the particles within the neighborhood of particle i (referred to as the neighbourhood best or *nbest*).

Algorithm 1 Basic PSO

```
Create and randomly initialize an $n$-dimensional swarm;
while stopping condition(s) is (are) not true do
  for each particle do
    if position is better than pbest then
      Update pbest using equation (4.3)
    end if
    if position is better than nbest then
      Update nbest using equation (4.4)
    end if
  end for
  for each particle do
    Update velocity using equation (4.2)
    Update position using equation (4.1)
  end for
end while
```

The PSO algorithm is summarized in Algorithm 1. All particles are randomly positioned in the search space, and the velocity of each particle is set to the vector $\vec{0}$. The quality of each particle is evaluated at every time step and the corresponding *pbest* and *nbest* updated as necessary. The *pbest* of each particle is updated as (assuming minimization)

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \leq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) > f(\mathbf{y}_i(t)) \end{cases} \tag{4.3}$$

and the *nbest* for particle i, in the neighborhood $\mathcal{N}_i$, is updated using [49]

$$\hat{\mathbf{y}}_i(t) \in \{\mathcal{N}_i \mid \{f\left(\hat{\mathbf{y}}_i(t)\right) = min\{f(\mathbf{x})\}, \quad \forall \quad \mathbf{x} \in \mathcal{N}_i\} \tag{4.4}$$

where $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function. The algorithm terminates after meeting certain stopping conditions, such exceeding a fixed number of iterations, no improvement observed over number of iterations, or discovering an acceptable solution.

4.2 Control Parameters of Particle Swarm Optimization Algorithms

The dynamics and efficiency of PSO algorithms are affected by a number of control parameters [123, 124, 126, 140, 143]. Values for these parameters have to be chosen carefully, because best values are problem dependent and have interconnected influences on performance. These parameters are:

- The swarm size, which is the number of particles in a swarm. An adequate swarm size is required for good initial swarm diversity that will cover the entire search space. A small number of particles may not give the required initial swarm diversity even with a good uniform initialization scheme. A high number of particles allows a larger part of the search space to be covered, but at the expense of increased computational complexity per iteration and the likelihood of degrading the search into a parallel random search. A too large swarm size may degrade the performance of PSO [50, 92].

- The neighbourhood size, which determines the degree of interaction among the swarm particles. When the neighbourhood size is small, the degree of interaction among the particles is low. This imply that small neighbourhoods facilitate exploitation (i.e. the concentration search effort around a particular region). On the other hand, when the neighbourhood size is large, the degree of interaction among the particles is high, thereby facilitating exploration (i.e scouting different regions of the search space in order to locate a good optimum). Different social network structures (topologies) that dictates neighbourhood size have been developed [78, 79, 80]. For example, a star topology has a fully interconnected structure, where each particle can communicate with every other particle within the swarm. The ring topology has a loosely connected structure, where each particle is connected to two others in ring shape fashion. Von Neumann topology has a grid-like structure where each particle is connected to its four immediate neighbours.

- The acceleration coefficients, which determines the influence of the cognitive and the social components on the trajectory of a particle, respectively. When the value of $c_2 >> c_1$, particles are more strongly attracted to the *nbest* than their *pbest* positions. This facilitates exploration. When the value for $c_1 >> c_2$, each particle becomes more attracted to its *pbest* position, decreasing the social interaction within the swarm. This result in more exploitation.

- The inertia weight, which is used to scale the influence of the previous velocity on the new velocity. Setting it to a large value facilitates exploration. However, when $\omega > 1$, the velocity grows with time, causing the swarm to diverge. A small value for ω promotes exploitation. A too small value will, however, limit the swarm exploration ability. Thus, the trade-off between exploration and exploitation depends on both ω and c_1 and c_2.

- **Maximum velocity**, V_{max}, is used to prevent a likely scenario of particles leaving the search space. In the velocity clamping strategy [45], escalating velocity values that exceed a certain maximum velocity are clamped down, to force the particles stay within boundary limits. Care must be taken in selecting the value of V_{max} because it affects the dynamic behaviour of the swarm. Choosing a large value promotes global exploration while a small value favours local exploitation. If the value is too large, particles may leap over good solutions, while a too small value may lead to particles trapped in local optima. Even though velocity clamping address the issue of escalating velocities, it has some drawbacks. During step size adjustment, unintended change in particle's flight direction occurs. Also, with velocity clamping, it is possible to have all velocities equal to V_{max}, where all particles searches only on the boundaries of a hypercube define by $[x_i(t) - V_{max}, x_i(t) + V_{max}]$.

- The number of iterations, often used as a terminating condition in PSO. When the number of iterations specified is too small, the algorithm terminates prematurely. On the other hand, when it is too large, it results in unnecessary computational cost.

4.3 Applications of Particle Swarm Optimization

PSO has a wide and diverse range of application areas, including scheduling [94], clustering [56] and classification [12], signal analysis [150], robotics [37], process control [8], NNs [44], and many more [106]. This study deals with PSO application in NNs, and therefore provides a review of PSO application to NN training.

One of the first applications of PSO was to train FNNs [44, 81], and the results from these applications have shown PSO to be an efficient training alternative. Since then, many researchers have successfully used PSO to train different types of NNs, including FNNs [95], RNNs [65, 73], radial bases function networks (RBFs) [9, 105], self-organising map NNs [89, 120] and product unit NNs [75]. It has also been applied successfully to train support vector machines [103]. PSO was shown to provide more accurate results compared to other learning algorithms in some specific applications [49].

To train a NN using PSO, particles are randomly initialized in an n-dimensional search space, where n is the total number of weights and biases of the NN. The position vector of each particle represents a solution (weight vector) of the NN, and the quality of each particle's position is determined by constructing the NN from the weight vector and calculating the sum square error (SSE) over the training set. Using this representation and fitness function, particles follow the PSO procedure described above to determine their direction and step size as they travel searching to find the best set of weights that minimizes the NN's error.

The performance of NNs trained using PSO was recently shown to be hindered by saturation of the activation functions in the hidden layer [111]. Saturation happens when the nodes in the hidden layer of a NN mostly output values near the asymptotic ends of the activation function range, thereby restricting the overall information capacity of the NN. The hidden unit saturation problem can be avoided by using unbounded activation functions. Van Wyk and Engelbrecht [144] have shown that the rectified linear function, which is left unbounded, has performance equal to that of the sigmoid or tanh functions.

Due to the successful application of PSO to train NNs in a number of problems, many researchers and analysts have applied PSO to time series forecasting [2, 57, 60, 97, 152]. PSO was also combined with other learning algorithms in order to improve forecasting accuracy. For example, [7] and [96] applied a hybrid of a GA and a PSO to train a feedforward NN to predict the power of solar stirling heat engines, and reservoir permeability, respectively.

The forecasting accuracy of NNs trained using PSO was evaluated in several studies, such as in [3, 76], and the results showed that PSO outperformed BP. Jha et al [76], however, concluded that PSO was not able to track non-stationary data. In a separate study, PSO was also shown to outperform a GA for a load forecasting problem [58]. Even though several studies have applied PSO to the task of training NN forecasters, producing favorable results, these studies have assumed a static environment, making them unsuitable for many real-world time series which are generated by varying processes.

Comparing the advantages of PSO to the classical BP, PSO is less dependent on initial weight values (since multiple starting points are used), PSO is less susceptible to premature convergence on a local optimum, and do not need derivatives of the NN to be computed. The main drawbacks of PSO in comparison with BP are slower speed of convergence [108], and a higher number of parameters are required to be tuned in order to achieve best performance.

4.4 Summary

This chapter provided an overview of the basic PSO algorithm for static optimization problems, and the various control parameters that have an influence on the algorithm's performance. Also, it describes the application of PSO in training NNs. Since basic PSO was designed to handle static optimization problems, the next chapter discusses the challenges of applying PSO to dynamic optimization problems. The chapter also describes various PSO algorithms modified to deal with optimization in dynamic environments.

Chapter 5

Particle Swarm Optimization in Dynamic Environments

"Adapt or perish, now as ever, is natures inexorable imperative."

– H. G. Wells

Basic PSO introduced in Chapter 4 assumes a static environment. In such environment, the objective function (i.e. the search landscape) does not change over time. This assumption is, however, not true for most real-world problems, because changes in the environment are observed, resulting in a dynamic environment (DE). Given an optimization problem in a DE (also referred to as a dynamic optimization problem), a solution once found may become suboptimal as the environment changes over time. This temporal property adds extra complexity to the problem since the optimization algorithm must not only find the optimal solution, but must also keep track of the optimum as it moves through the search space, as well as detect new optimal solutions as they appear.

The objectives of this chapter are to provide an overview on the challenges faced by PSO when applied to dynamic optimization problems, and to review some PSO algorithms modified to suit DEs. Before that, it is important to define dynamic optimization problems and to discuss the characteristics of different types of DEs.

The chapter is structured as follows: Section 5.1 briefly discusses dynamic optimization problems. The challenges faced by basic PSO in DEs are discussed in Section 5.2. Section 5.3 describes the mechanisms for change detection and response in a DE. Section 5.4 gives an overview of popular PSO algorithms designed for DEs. Section 5.5 summarizes the chapter.

5.1 Dynamic Optimization Problems

In computational sciences, an optimization problem refers to the problem of finding the best solution from a set of all feasible solutions. Formally, an optimisation problem is defined as the problem of either minimising or maximising an objective function within a search space, where the search space is the set of all feasible and infeasible solutions. Training a NN is a classical example of an optimisation problem, since the aim is to find a set of weights that minimizes the error produced by the NN, where the error function is the objective function of the NN.

When the objective function of an optimization problem changes over time, the problem becomes a *dynamic optimization problem* (DOP). Assuming a minimization problem, a boundary constrained DOP is mathematically defined as [49]

$$\begin{aligned} \text{minimize} \quad & f(\vec{x}, \vec{\varphi}(t)) \\ \text{subject to} \quad & x_d \in dom(x_d) \end{aligned} \tag{5.1}$$

where $\vec{\varphi}(t)$ is a vector of time-dependent control parameters of the objective function, and $dom(x_d)$ is the domain of x_d for dimension d. The objective of a DOP is to find $\vec{x}^*(t) = min_{\vec{x}}\, f(\vec{x}, \vec{\varphi}(t))$, where $\vec{x}^*(t)$ is the minimum found at time step t.

Considering the landscape of the objective function as a hypersurface, it can then be said that environmental changes are basically disturbances of this hypersurface. Thus, changes in an environment leads to volatile optima of the objective function, where existing optima may disappear and a new optima may appear elsewhere in the hypersurface.

For a DOP, the objective is then not just to locate an optimum of the objective function (as for static optimization problems), but also to track the changing optima, as well as locating new emerging optima, and detecting disappearing optima.

Section 5.1.1 describes characteristics specific to dynamic optimization problems and Section 5.1.2 briefly discusses the types of DEs.

5.1.1 Characteristics of Dynamic Environments

Dynamic environments are characterized by the nature of the modifications that the objective function undergoes over time. A DE is characterized by,

- the *direction of change*, which specifies if a change can alter the objective function value of the optimum, the location of the optimum, or both.
- the *homogeneity of movement*, which involves systematic change over the search space. When a change takes place in a homogeneous environment, all the optima enjoy uniform transformations, and the peaks show the same behaviour at any given time. However, when the change happens in a heterogeneous environment, every optimum may experience different transformations, and different peaks can exhibit completely different bahaviours.
- the *presence or absence of patterns in the changes* to the landscape of the objective function. If there are patterns in the changes, the shape of these patterns (i.e. the trajectory of the optima) defines the dynamics of the environment, which can also be used to classify the DE. These trajectories could lead to environments with linear, circular or random dynamics. For a pattern that repeats itself over time, the *cycle length* (which is the number of environmental states between two consecutive occurrence of the same states) is another characteristic of DEs.
- *temporal severity*, which refers to the number of times an environment changes in a unit of time. The changes can occur continuously, at irregular intervals, or periodically. Changes due to regular periodic modifications can be defined in terms of a *frequency update* [10], which is usually measured by the number of iterations, the number of function evaluations, or the time between successive changes. The landscape of an environment is said to have the lowest possible temporal severity when changes take place only once. When the modification to the environment occurs at each time step (causing the environment to change continuously), the landscape is said to have the highest temporal severity. The delay between any two changes due to the low temporal severity of a landscape gives an optimization algorithm a chance to find solutions. However, frequent changes in the landscape

due to higher temporal severity compels the optimization algorithm to continually adjust the solution(s) found in response to the changes that occur.

- *spacial severity*, which refers to the magnitude of the changes in the environment. When the intensity of modification to the optimum is invariably similar after every change, the spatial severity is said to be *regular*. Otherwise, the spatial severity may *follow a pattern* when the intensity of alterations is predictable after every change. The severity of the changes can also vary randomly within a certain range. For environments with non-regular spatial severity, spatial severity is determined as the average spatial severity over the entire time period. For environments with multiple optima, spatial severity can be evaluated by either considering only the most extensive alteration in objective function value or location, or by considering changes to a randomly selected peak [22]. Alternatively, the spatial severity can be measured by averaging the changes experienced by all optima. Generally, the level of severity of change affects the performance of an optimization algorithm. A large change makes it difficult for the optimization algorithm to recover from the change, because the new optimum may have been displaced to a location far away from the old optimum. But in the case of a small change, the new optimum may be within the area surrounding the old optimum.

5.1.2 Types of Dynamic Environments

A number of DE classification schemes have been proposed in the literature, and they are all based on one specific, or combination of, DE characteristic discussed in Section 5.1.1.

Eberhart *et al* identified the following three types of DEs based on the direction of change [47, 69]:

- **Type I** environments, where the position of an optimum in the problem space changes, while the objective function value remains the same. For this type of environment, an optimization algorithm is required to first find the optimum, and then keep track of the optimum as it moves in the search space. This is required because a new optimum can not appear in the search space, nor a local optimum become a global optimum.

- **Type II** environments, where the value of an optimum changes with time, but the position of the optimum remains unchanged. This kind of environment can be realised through fitness rescaling of the landscape components [147]. In a type II environment, where there are multiple peaks, a decrease in the value of a local optimum or an increase in the value of a global optimum may result in a change of the global optimum, if minimization is assumed. An optimization algorithm applied to a type II environment must aim to detect the emergence of a new optimum and disappearance of existing optima.

- **Type III** environments combine the properties of type I and type II environments, where both the value of an optimum and its location are subject to changes.

Angeline proposed another scheme for classifying DEs into distinct categories, based on the trajectories of the optima over a number of iterations [10]. Three types of DEs are defined, namely, environments with linear peak trajectories, environments with circular trajectories, and environments with random trajectories.

Even though the Eberhart *et al* and the Angeline classifications provide information on different features of the changes that take place in the environment, none of the two classifications provide explicit information about the temporal or spatial severity of the changes.

Duhain and Engelbrecht [43] proposed a DE classification scheme based on the temporal and spatial severity of changes an environment undergo:

- Quasi-static environments, where changes in the environment are negligible compared to the scale of the problem, such that the performance of the algorithm is not affected for the duration of the simulation.

- Progressively changing environments, where the frequency of modification to the environment is high, and the magnitude is small, resulting in gradually changing environments. For efficient optimization in the progressively changing environments, an algorithm needs to re-optimize fast and to inject some diversity.

- Abruptly changing environments, where the changes are severe, but do not occur often. This allows the environment to remain static for some time, enabling the algorithm to find a good solution before the severe change that occurs abruptly, which may significantly change the objective function value and/ or location of the optimum. Therefore, an optimization algorithm needs to inject much diversity.

- Chaotic environments, where the environment is frequently and severely changing due to the high spatial and temporal severity. In an environment with a chaotic nature of behaviour, the algorithm must be able to adapt to the frequent and severe changes simultaneously. The optimization process in a chaotically changing environment may be highly difficult if, after every iteration, the optimum is randomly displaced to a different area of the search space. Thus, an optimization algorithm needs to re-optimize fast and to inject much diversity.

The DE classes proposed by Duhain and Engelbrecht require defining levels of spatial and temporal severity. However, there is no absolute value that can be used to distinguish between severely and non-severely changes or frequently and infrequently changing environments. Hence, measuring the level of spatial and temporal severity is considered problem dependent. Thus, in measuring spatial severity, the size of the entire search space should be considered. Similarly, in differentiating between frequently and infrequently changing environments, the duration of the entire simulation should be considered.

To analyse the influence of the behaviours of different environments on the performance of an algorithm, experimental test cases can be simulated with varying spatial and temporal severity to test the algorithm. Duhain and Engelbrecht [43] proposed that the three classification schemes can be combined to produce 27 different dynamic optimization problem classes.

5.2 Standard PSO in Dynamic Environments

Standard PSO was shown to have an inherent ability to adapt to minor changes in the environmental landscape [26, 47, 69]. Consider a converging swarm, where the particles are oscillating around the global optimum, moving progressively closer to the global best position. If there is a shift in the position of an optimum to a nearby location, it is likely that one of the oscillating particles may discover the new optimum, and attract other particles to the new optimum. However, if the shift in the position of an optimum is large, displacing the optimum to a location outside the radius of the contracting swarm, PSO will fail to detect the new optimum since there are no particles exploring that part of the search space. The PSO will also fail to discover a peak appearing outside the region of the contracting swarm due to *loss of diversity*. Loss of swam diversity is one of the major challenges faced by PSO when applied to DEs. To address the loss of diversity problem, diversity need to be re-injected or maintained at all times.

The issue of *outdated memory* is the other major challenge faced by PSO when applied to DEs. When a change occurs, the swarm attractors (i.e. the personal and neighborhood best positions) may no longer be representative of the new environment, as the optimum may have moved to a new position. Thus, the attractors may lead the swarm to part of the search space that contains poor solutions. To avoid this issue, this personal best positions have to be re-evaluated.

5.3 Change Detection and Response

To address the challenges faced by PSO in dynamic environments, mechanisms need to be put in place that will ensure any change in the environment is timely and correctly detected. When a change in the environment is detected, an appropriate response strategy must be triggered to enable the optimization algorithm to adapt to these changes. This section discusses change detection and response strategies for adapting PSO to DEs in Sections 5.3.1 and 5.3.2 respectively.

5.3.1 Change Detection

Change in a landscape can occur at a predefined interval or known frequency. In such case, the task of detecting the change becomes easy. However, in most cases, prior knowledge about when changes occur is not known. Thus, some mechanisms are required to detect changes based on feedback from the environment.

In PSO literature, a change detection mechanism is commonly implemented by using sentry particles, as proposed by Carlisle and Dozier [26]. A sentry is chosen to be any particle in the swarm or a randomly selected fixed point in the search space [28]. At the beginning of each iteration, the quality of each sentry point is evaluated. If the quality of any of the sentry points differ between the current and the previous iterations, a change is assumed to have occurred.

When one particle is used as the sentry for all iterations, it is likely that changes in the environment may not be detected if the area around the sentry remains static while changes occur elsewhere. To avoid this issue, Carlisle and Dozier [26] used randomly selected particles as the sentries for each iteration in order to get feedback from different parts of the environment. As an alternative, Hu and Eberhart [69] proposed using the global best position as the sentry. This approach ensures that global information is used in detecting changes, and is based on the assumption that if the position of an optimum changes, then the optimum value of the current position also changes. However, only changes around the global best position will be detected. Thus, using the global best position as the sentry will only work for quasi-static and progressive environments (to some extent). To improve the change detection accuracy, Hu and Eberhart [70] further used the global best position and the global second-best position as the sentries. Even with this improvement, the approach still suffers from the same issue mentioned above.

Swarm particles used as sentries are effective only before the swarm converges, because the sentry particles that are supposed to be scouting the search space are now converged at a point. To avoid this problem, fixed point sentries uniformly distributed in the search space can be used. Selecting the number of sentries to use is problem dependent. When a large number of sentries are used, the chance of detecting the change increases, as well as the overall computational complexity in evaluating sentry particles.

5.3.2 Response to Change

Once a change in an environment is detected, the problems of outdated memory and diversity loss are addressed using the following strategies:

Memory Update

To address the issue of outdated memory, the swarm memory has to be reset. Carlisle and Dizier [26] and Hu and Eberhart [69] suggested that once change is detected, the personal best positions of all the particle to be reset to their current positions in order to prevent the particles returning to staled positions. However, if the swarm has converged to a solution, resetting the personal best positions of all the particles to their current positions will introduce the diversity problem. This problem can be addressed by carrying out partial reinitialization of the swarm (to increase diversity) in combination with the resetting of the personal best position of the particles [70]. Forgetting the experience gained by the swarm about the search space by resetting all the particles may not be ideal if changes are less severe or gradual, since it is likely that, after a change, the personal best position may be closer to the new optimum than the current positions. Carlisle and Dozier [27] proposed resetting the personal best positions only if the fitness of the current positions for the new environment is better than the personal best positions.

Also, to prevent the global (or the neighbourhood) best position to lead the swarm to out-of-date positions, the global best position should be selected from the most recent personal best positions obtained after change detection. An alternative strategy is to recalculate the global best position only if it is worse under the new environment.

Diversity Enhancement

Loss of diversity results from convergence, where swarm particles remain in a stable state and can no longer explore the search space, even if the optimum does change. Loss of diversity is the most critical problem for algorithms in dynamic environments. This problem is solved by enhancing the swarm diversity using the concepts of re-diversification, repulsion, multi-population or dynamic neighbourhood topology. These diversity enhancement schemes are described in detail in the following section where some PSO algorithms modified for dynamic environments are discussed.

5.4 Particle Swarm Optimizers for Dynamic Environments

A number of modified PSO algorithms have been developed to track optima in dynamic environments. This section reviews the most popular versions of these algorithms.

5.4.1 Reinitialising Particle Swarm Optimization

The reinitialising PSO algorithm is basically a standard PSO augmented with a change detection and response strategy. When no environment change is detected, the algorithm behaves like a standard PSO. When a change is detected, a percentage of the swarm particles are randomly repositioned in the search space to reintroduce diversity into the search space. To prevent these particles from being immediately pulled back to their previous positions, their personal best positions are set to the current positions. The percentage of particles repositioned is problem dependent and should be determined empirically.

Reinitialisation of PSO heavily relies on the detection mechanism employed, and if some changes in the search space pass undetected, the algorithm may experience the limitations of a standard PSO. Also, frequent reinitialisations due to high temporal severity may reduce the exploitation capacity of the algorithm. Another disadvantage is partial loss of knowledge about the search space due to particle reinitialisation [70].

5.4.2 Charged Particle Swarm Optimization

The charged PSO (CPSO), proposed by Blackwell and Bentley [17] use Coulomb repulsion among particles to maintain diversity. All the particles in CPSO store a charge. These charged particles repel each other if the distance between them is too small. This repelling property prevents the particles from converging to a single point in the search space, thus promoting exploration. A slight variant, called the *atomic* PSO, has a mix of charged and neutral particles. The neutral particles behave according to the position and velocity update rules of a standard PSO. The neutral particles fine-tune the solutions found, thereby facilitating exploitation. The interaction between charged and neutral particles helps the algorithm to maintain a balance between exploration and exploitation.

The atomic PSO was shown to be superior to the CPSO and the standard PSO in solving some DOPs [17]. The study also showed that a swarm with a 50/50 split between charged and neural particles are more efficient on some problems. The main disadvantage of CPSO is its computational complexity of $O(n^2)$ (where n is the number of charged particles), since the distance between all charged particles is required to be computed for calculating inter-particle repulsion. Also, CPSO introduces additional control parameters that have to be tuned.

5.4.3 Quantum Particle Swarm Optimization

The quantum PSO algorithm (QPSO) [16] is inspired by the quantum model of an atom. A portion of the particles implement the standard PSO velocity and position updates. The rest of the particles, referred to as quantum particles, have their positions sampled from a probability distribution centered on the global best position. Positions of the quantum particles are calculated as,

$$\mathbf{x}_i \sim d(\hat{\mathbf{y}}, \mathbf{r}_{cloud}) \tag{5.2}$$

where d is the probability distribution and $\mathbf{r}_{cloud}$ is the quantum radius. The control parameter, r_{cloud}, is problem and environment dependent [16, 42].

Recently, Harrison *et al* [62] proposed using a parent centric crossover (PCX) operator [38] to generate the positions of quantum particles instead of using the radius and probability distribution parameters used in the standard QPSO. In an empirical study using single-peak dynamic environment types, the new variant (i.e. QPSO-PCX) was shown to be superior to QPSO on progressive and chaotic environments, while no superior approach on quasi-static and abrupt environments was found [62]. The performance of the QPSO-PCX algorithm was, however, not evaluated on real-world and multi-peak environments.

Compared to charged particles, quantum particles behave better due to lower complexity. The main issue with QPSO is that it can not detect changes outside the $\mathbf{r}_{cloud}$.

5.4.4 Multiswam Particle Swarm Optimization

Blackwell and Branke [15] proposed employing multiple swarms for optimization in dynamic environments. The multiswarm PSO (MPSO) uses independent sub-swarms where each subswarm tracks a different peak in the search space. The sub-swarms interact locally through *exclusion* to prevent more than one sub-swarm from settling on a single peak. This process ensures that swarm diversity is maintained. The process involves centering an exclusion radius r_{excl} around the best particle of each sub-swarm (also called the swarm attractor). If any sub-swarm enters the r_{excl} of another, the sub-swarm with the worst swarm attractor's quality is reinitialized.

Due to exclusion, different swarms may converge on different local optima. A swarm is considered to have converged if its radius (i.e half of the maximum distance between two particles in the swarm) is smaller than the convergence radius, $r_{conv.}$. In a situation where a new optimum appears while the swarms have started contracting around their local optima, detecting a new emerging optimum becomes a challenge due to loss of swarm diversity. To address loss of swarm diversity, MPSO employs another global interaction mechanism, called *anti-convergence*, to ensure that the algorithm maintains its exploration capacity. When all the swarms have converged, anti-convergence reinitializes the weak sub-swarm (i.e. the one with the worst swarm attractor's quality). This process compels the weak swarm to patrol the search space for a new optimum, which helps the algorithm to regain its diversity.

In order to maintain diversity within the sub-swarms (which is a requirement for effective tracking of moving optima), the CPSO (discussed in section 5.4.2) or the QPSO (discussed in section 5.4.3) are used in the sub-swarms. The main disadvantage of the MPSO algorithm is the large number of control parameters required to be set.

5.4.5 Cooperative Particle Swarm Optimization

Cooperative PSO was introduced by Van den Bergh and Engelbrecht [139, 142] to improve the performance of standard PSO for large dimensional problems. A divide-and-conquer approach is followed to break the n dimensional problem into a number of smaller-dimensional problems that are easier to solve. Results presented in [141] showed that, as the dimensionality of a problem increases, cooperative PSO performs better than standard PSO.

The cooperative PSO partitions the search space dimension-wise into k disjoint groups where each group is assigned to an individual sub-swarm to optimize. The algorithm maintains a complete solution (or context) vector that individual sub-swarms use to evaluate the quality of its particles. Quality is evaluated for each particle in a sub-swarm by substituting values of the decision variables the sub-swarm is accountable for into the context vector, while keeping constant the remaining values in the vector (which are the best values from other sub-swarms). The particle that, when substituted into the context vector, had the best quality is returned as the best solution in the sub-swarm. After all sub-swarms have evaluated their particles' quality, the context vector contains the optimal solution discovered so far.

The main drawback of the cooperative PSO algorithm is its likelihood of converging onto a pseudominimum that is created due the decomposition of the search space. The effectiveness of this decomposition is also affected by the level of correlation between the subproblems. Despite these potential problems, the cooperative PSO algorithms showed significantly better performance on many of the problems tested [142].

Rakitianskaia and Engelbrecht [109] proposed the cooperative charged PSO (CCPSO) for optimization in dynamic environments. The CCPSO simply use charged PSO as the sub-swarms in a cooperative PSO. Unger *et al* [137] introduced another cooperative PSO strategy for dynamic environments called the cooperating quantum PSO (CQSO). The CQSO employs QPSO in the sub-swarms of a cooperative PSO. Experimental results from [137] suggested the superiority of CQSO over a number of other PSO variants for dynamic environments.

5.5 Summary

This chapter briefly discussed dynamic optimization problems and the different characteristics of dynamic environments. The challenges of diversity loss and outdated memory faced by standard PSO when applied to DEs were presented. Mechanisms for environmental change detection and appropriate response strategies to the problems faced by PSO in DEs were discussed. An overview of some popular PSO algorithms modified to suit DEs is given. The next chapter formulates training of a NN forecaster as a dynamic optimization problem to investigate the applicability and efficiency of a dynamic PSO algorithm as training algorithm for FNN forecasters under non-stationary environments.

Chapter 6

Training Feedforward Neural Network Forecasters using Dynamic Particle Swarm Optimizer

"We must never make experiments to confirm our ideas, but simply to control them."

– Claude Bernard -1865

PSO is now an established method for training NNs, and was shown in several studies to outperform the classical BP training algorithm [113, 133, 141]. The effectiveness of standard PSO as a training algorithm for FNNs in forecasting non-stationary time series was evaluated in [3], where the authors considered the problem as a static optimization problem. The results showed that PSO performed better than BP, but concluded that the result of PSO could significantly be improved by using a suitable NN structure that can handle dynamic data. Jar *et al* [76] compared the performance of PSO and BP in training FNNs for time series forecasting. The results indicated superiority of PSO over BP, but concluded that the PSO trained FNNs were unable to track the non-stationary data.

The original PSO was, however, designed for static environments. In dealing with non-stationary data, modified versions of PSO for optimization in dynamic environments are used (refer to Chapter 5). Dynamic PSOs have been used to train NNs on classification problems under non-stationary environments [42, 109, 112], as discussed in Chapter 5. These studies showed that dynamic PSOs are indeed applicable to NN training in dynamic environments, producing significantly better or similar results compared to the classic BP.

This chapter formulates training of a NN forecaster as a dynamic optimization problems, and investigates the applicability and efficiency of a dymanic PSO algorithm as training algorithm for FNN forecasters under non-stationary environments. The results obtained for the dynamic PSO are then compared against standard PSO and backpropagation (RPROP) algorithms.

The rest of the chapter is organised as follows: Section 6.1 describes the experimental procedure followed. Experimental results are presented and discussed in Section 6.2. Section 6.3 summarizes the chapter.

6.1 Experimental Procedure

This section describes the general methodology followed. In the study, sets of experiments were conducted on ten different forecasting problems. Each set of experiments entails separately training a FNN using a dynamic PSO, a standard PSO, and a BP under nine different dynamic scenarios to forecast one of the problems. For each problem, the performance of the FNN trained with a dynamic PSO algorithm was compared to the performance of the FNN trained using BP and standard PSO. For effective evaluation of the performance, 30 independent runs for each experiment were carried out and averages with confidence interval over these 30 runs were computed.

All algorithms used were implemented in the Computational Intelligence Library (CIlib), version 0.9. CIlib is an open source collective project aimed at building a generic framework, designed to accommodate the implementation and execution of computational intelligence algorithms [102]. CIlib is available at http://github.com/cirg-up/cilib.

Section 6.1.1 describes the various datasets used. Section 6.1.2 discusses the data preparation process employed. The process of simulating a dynamic environment is described in section 6.1.3. Section 6.1.4 describes the assignment of values to the control parameters of the algorithms. The performance metrics and statistical methods used in the study are discussed in Section 6.1.5.

6.1.1 Datasets

A set of ten different well studied time series with varying complexities were used in the study. Seven are real-world time series obtained online from the Time Series Data Library at http://robjhyndman.com/TSDL, and the other three are artificially generated. The time series studied are:

- Sunspot Annual Measure Time Series

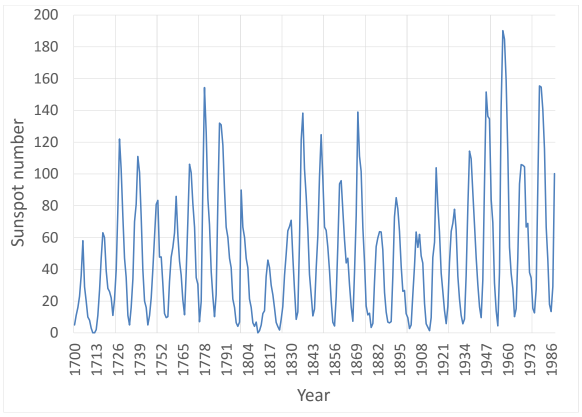
  The Sunspot Annual Measure (SAM) dataset consists of 289 points representing the total annual measure of sunspots from 1770 to 1988. Illustrated in Figure 6.1a, the SAM series has a strong seasonal pattern with somewhat constant trend. The SAM time series has been extensively studied in statistical literature, and is often used as a standard for evaluating and comparing new forecasting methods. The data is known to be non-linear, non-stationary and non-Gaussian [153].

- Airline Passengers Time Series

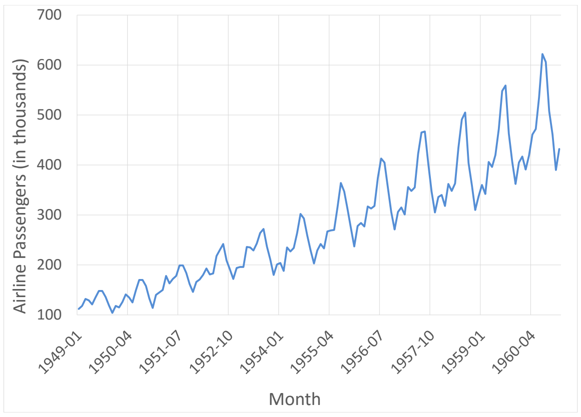
  The International Airline Passengers (IAP) dataset has a total of 144 observations representing the monthly number of international airline passengers from January 1949 to December 1960. The IAP time series is a well-known dataset and was used in the classical work by Box and Jenkins [21]. As shown in Figure 6.1b, the series follows a multiplicative seasonal pattern with an upward trend and no outliers. The IAP dataset is non-stationary due to the presence of strong seasonal variation.

- Australian Wine Sales Time Series

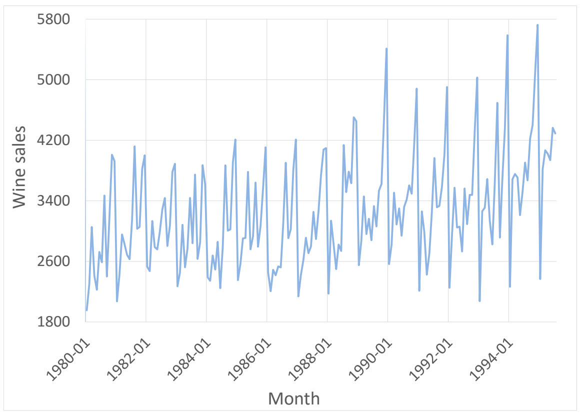
  The Australian Wine Sales (AWS) series contains 187 observations of monthly wine sales (in thousands of liters) in Australia from January 1980 to July 1995. Illustrated in Figure 6.1c, the AWS time series has a slightly increasing trend and a strong seasonal pattern (peaks in July, and troughs in January) with no obvious outliers. Another distinct feature of the series is an increase in variability.



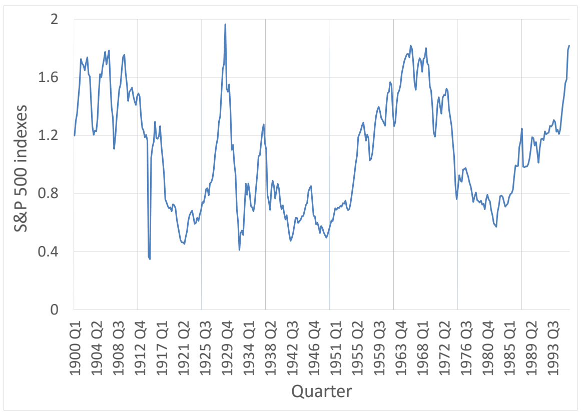
(a) SAM Time Series



(b) IAP Time Series



(c) AWS Time Series



(d) S&P Time Series

Figure 6.1: Time plot of four datasets used

- Standard and Poor 500 Indexes

  The Standard and Poor (S&P) series has 388 values of historical quarterly S&P 500 indexes from 1990 to 1996. It is a very common financial time series used as a benchmark by many researchers in testing forecasting models [30, 107, 127]. A plot of the dataset is given in Figure 6.1d, which reveals a constant trend with long-run cycles.

- US Death Time Series

  The US Death (USD) time series contains the total monthly deaths caused by accidents in the USA between January 1973 and December 1978, making a total of

72 observations. Shown in Figure 6.2a, the USD series reveals a slightly constant trend with seasonal variations (peaks in January).

- Hourly Internet Traffic Time Series

  The Hourly Internet Traffic (HIT) time series has 1657 data points representing aggregated hourly Internet traffic data (in bits) from an Internet service provider in the United Kingdom academic network backbone. It was collected between 19 November 2004, at 09:30 hours and 27 January 2005, at 11:11 hours. As shown in Figure 6.2b, the HIT time series is non-stationary and with a slightly downward trend.

- Daily Minimum Temperature Time Series

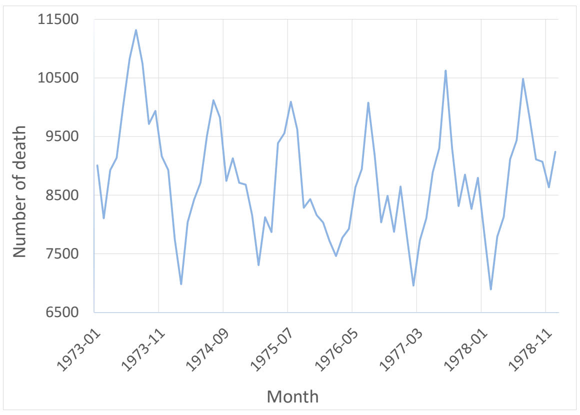
  The Daily Minimum Temperature (DMT) time series contains a total of 3650 observations representing daily minimum temperatures in Melbourne, Australia, 1981-1990. Figure 6.2c displays the DMT time series graphically. The series is highly noisy with a constant trend over the entire time span.

- Lorenz Time Series

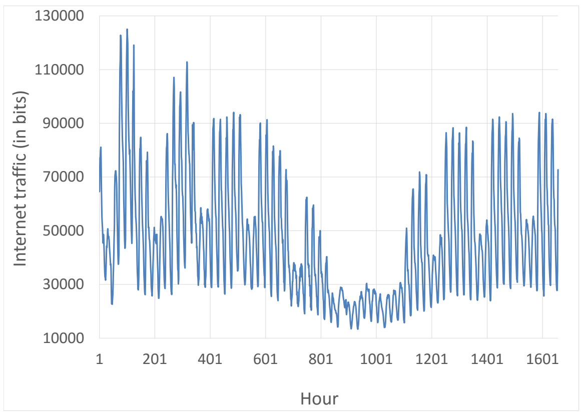
  The Lorenz time series is a widely studied chaotic series generated from a Lorenz system. Described in the equations below, the Lorenz system is known to be a simplified model of various physical systems [87]:

$$\begin{aligned} x(t+1) &= x(t) + K\sigma[y(t) - x(t)] \\ y(t+1) &= y(t) + K(x(t)[r - z(t)] - y(t)) \\ z(t+1) &= z(t) + K(x(t)y(t) - bz(t)) \end{aligned} \tag{6.1}$$

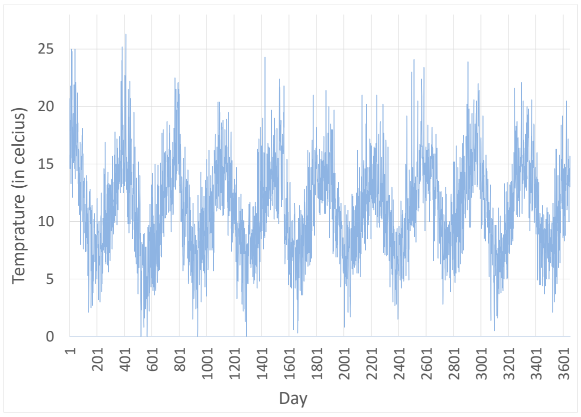
  where $x(t), y(t), z(t)$ are states of the Lorenz system, r, σ, b are constant parameters of the system, and K is the sampling time. A total of 5000 samples were generated using $\sigma = 10.500001$, $r = 28.200001$, $b = 2.700001$, $K = 0.0130001$, $x(0) = 1.200001$, $y(0) = 1.500001$, and $z(0) = 1.600001$. The $y(t)$ state data was chosen for the study, and only the last 4000 samples were used to avoid a transient mode (i.e. prevailing influence of parameters used in generating the series). A plot of the time series is given in Figure 6.2d.



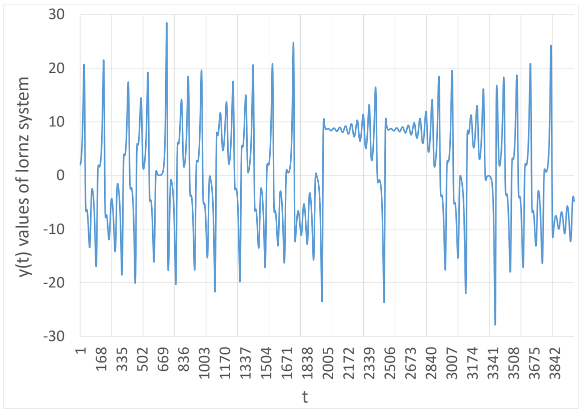
(a) USD Time Series



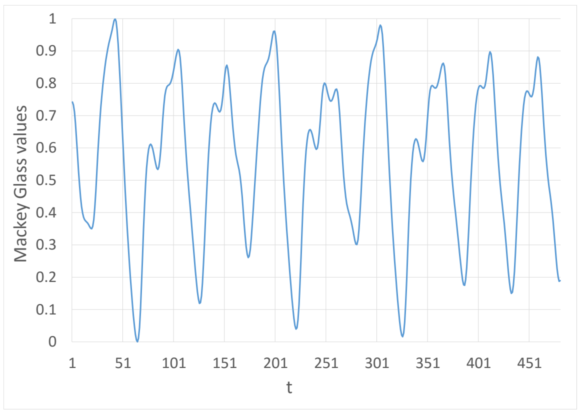
(b) HIT Time Series



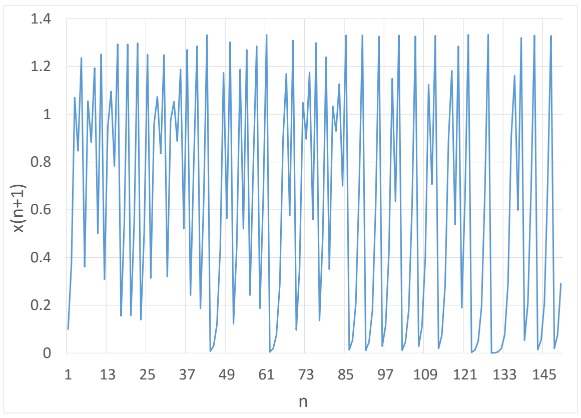
(c) DMT Time Series



(d) Lorenz Time Series



(e) MG Time Series



(f) LM Time Series

Figure 6.2: Time plot of six datasets used

- Mackay Glass Time Series

  The Mackay Glass (MG) time series is a chaotic series generated from a solution of the Mackay-Glass delay-differential equation [83],

  $$\frac{dx}{dt} = \frac{ax(t-\tau)}{1-x^c(t-\tau)} - bx(t) \tag{6.2}$$

  using $\tau = 30$, $a = 0.2$, $b = 0.1$, $c = 10$, and initial condition $x(t) = 0.9$ for $0 \leq t \leq \tau$. A 500 points dataset was generated for this study, where 480 data points after the initial transients were used for training and testing. A plot of the MG series is given in Figure 6.2e.

- Logistic Map Time Series

  The Logistic Map (LM) time series is a chaotic series generated from logistic map equation [74],

  $$x(n+1) = x(n) + Gx(n)(1-x(n)) \tag{6.3}$$

  For this study, data points were generated by iterating the equation 150 times starting from a random initial value set to 0.1 and $G = 3$. The equation behaves chaotically when G is set to 3. The LM time series as illustrated in Figure 6.2f is chaotic, non-linear and non-stationary.

6.1.2 Dataset Preparation

All the datasets were scaled to the range [-1, 1] and normalised such that the mean of each input variable over the training set is close to zero (to promote faster convergence) as suggested in [85]. The normalization was done using,

$$x'_n = \frac{x_n}{\sqrt{N}} \tag{6.4}$$

where x_n is an observation, x'_n is the normalised observation, and N is the number of observations in the dataset. Each of the datasets was divided into two independent subsets. The first 80% of the dataset was used for training and the remaining 20% for testing. Since the work deals with time series, the datasets were not shuffled. For parameter optimization purposes only, the training subset was split further into a 70:30 ratio for training and validation respectively.

6.1.3 Simulating Dynamic Environments

For each dataset, performance was investigated under nine different dynamic environmental scenarios. The scenarios were simulated using a sliding time window technique. This involves choosing a window of size w and a step value s for sliding the window over the dataset. The window is used to train the NNs for f number of iterations (i.e change frequency) before the window slides over the dataset. The sliding process involves throwing away the s values at the beginning of the window and adding the next s values in the data series to the end of the window. The training and sliding process is repeated until all of the datasets are used. An example to illustrate this process with $w = 6$ and $s = 4$ is given in Figure 6.3. When the window slides, four values, $\{x_1, x_2, x_3, x_4\}$, are discarded and new values, $\{x_7, x_8, x_9, x_{10}\}$, are added, while x_5 and x_6 remain in the window.

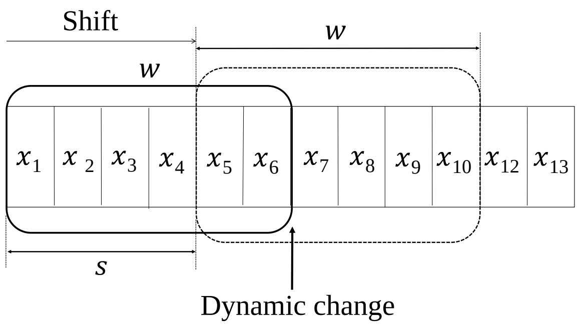


Figure 6.3: Sliding Time Window

The step size determines the spatial severity of the change. A small value for s implies a slight change, while a large value implies a drastic change. An algorithm runs on a window for f iterations before the window slides, controlling the temporal severity. Table 6.1 presents the parameter setup used to simulate the nine different dynamic scenarios for all the problems. As shown in the table, the combination of s and f differs under different scenarios, and therefore requires different numbers of iterations to traverse the entire dataset, calculated as

$$T = f * \frac{N - w}{s} + f \tag{6.5}$$

Table 6.1: Dynamic scenarios settings for each dataset

Dataset	Parameters	Scenarios								
		A1	A2	A3	B1	B2	B3	C1	C2	C3
SAM	Window size	60	60	60	60	60	60	60	60	60
	Step size	20	40	60	20	40	60	20	40	60
	change frequency	50	50	50	100	100	100	150	150	150
HIT	Window size	584	584	584	584	584	584	584	584	584
	Step size	100	250	528	100	250	528	100	250	528
	Change frequency	50	50	50	100	100	100	150	150	150
DMT	Window size	510	510	510	510	510	510	510	510	510
	Step size	200	400	510	200	400	510	200	400	510
	Change frequency	50	50	50	100	100	100	150	150	150
MG	Window size	84	84	84	84	84	84	84	84	84
	Step size	30	60	84	30	60	84	30	60	84
	change frequency	50	50	50	100	100	100	200	200	200
Lorenz	Window size	330	330	330	330	330	330	330	330	330
	Step size	100	250	330	100	250	330	100	250	330
	change frequency	50	50	50	100	100	100	150	150	150
IAP	Window size	32	32	32	32	32	32	32	32	32
	Step size	10	25	32	10	25	32	10	25	32
	Change frequency	50	50	50	100	100	100	150	150	150
AWS	Window size	42	42	42	42	42	42	42	42	42
	Step size	20	35	42	20	35	42	20	35	42
	Change frequency	50	20	50	100	100	100	150	150	150
S&P	Window size	58	58	58	58	58	58	58	58	58
	Step size	20	40	58	20	40	58	20	40	58
	Change frequency	50	50	50	100	100	100	150	150	150
USD	Window size	20	20	20	20	20	20	20	20	20
	Step size	8	16	20	8	16	20	8	16	20
	change frequency	50	50	50	100	100	100	150	150	150
LM	Window size	31	31	31	31	31	31	31	31	31
	Step size	10	25	31	10	25	31	10	25	31
	Change frequency	50	50	50	100	100	100	150	150	150

6.1.4 Parameter Selection

In order to ensure that all the algorithms exhibited efficient performance and to facilitate fair comparison among the training algorithms, the relevant algorithm parameters were optimised and set up as follows:

1. **NN Configuration**

   The NN for each dataset was determined as follows:

   - Input layer: For datasets collected annually, 10 input nodes were used, each representing a year in the decade. For monthly datasets, 12 input nodes were used, each representing a month of the year. For quarterly/weekly datasets, four input nodes were used, each representing a quarter of the year or a week of the month respectively. For data collected hourly, 24 input nodes were used. This intuitive method was used by a number of analysts such as [61, 84, 121, 130, 132], and has been effective in constructing optimal NN structures. For the synthetic datasets, the number of input nodes where adopted from previous studies as shown in Table 6.2.
   - Hidden layer: A single hidden layer was used for all the NNs and the number of hidden nodes was iteratively optimized on the validation set, where discrete numbers in the range [2, 50] were considered. For every value within the range, 30 independent runs were conducted and the value that yielded the minimum average validation error was chosen as optimal. Due to the saturation problem caused by bounded functions when PSO is used to train FNNs (refer to section 4.3), linear activation functions were used in the hidden units.
   - Output layer: A single output node with a modified hyperbolic tangent function was used for all the NNs (one step ahead forecasting was considered).

   All NN weights were randomly initialized in the range $[-\frac{1}{\sqrt{F}}, \frac{1}{\sqrt{F}}]$, where F is the number of incoming connections for a specific node. Wessels and Barnard [148] have shown this to be a good initialization range.

Table 6.2: Input nodes for synthetic datasets adopted from previous studies

Dataset	Input Nodes	Researchers
MG	4	[23]
Lorenz	5	[53]
LM	3	[54]

2. **RPROP Setup**

   Default RPROP parameters were used in this study, since RPROP does not require optimizing parameters to obtain optimal convergence times on many problems [115]. Thus, for all experiments, $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_0 = 0.0125$ (initial value of $\Delta_{i,j}$), $\Delta_{min} = 0$ and $\Delta_{max} = 50$, see [114, 116].

3. **PSO Setup**

   For all the experiments, a linearly decreasing inertia weight was used, with an initial value of 0.9 and a final value of 0.5. Acceleration coefficients values were fixed at $c_1 = c_2 = 1.49$ to ensure convergence, based on [32, 46]. Velocity was not constrained and the Von Neumann topology was used since it facilitates diversity [79, 88], which is good for dynamic problems. For each experiment, the swarm size was determined as equal to the total number of particles used in the CQSO in order to facilitate fair comparison.

4. **CQSO Setup**

   In addition to the PSO parameter setup, the radius of the quantum cloud and the percentage of quantum particles per swarm were iteratively selected from the ranges given in Table 6.3, as suggested in [15].

   For each dataset, the number of sub-swarms, k, in the CQSO was determined as the ratio $[N_w/d]$, where N_w is the total number of weights and biases in the NN and d is the number of weights grouped together. The value of parameter d was iteratively optimized from the range of values given in Table 6.3. The size of each sub-swarm was set to 10 particles based on [140, 142].

Table 6.3: PSO parameter ranges considered

Parameter	Range
Quantum radius, r	[0.2, 0.5, 0.8, 1, 2]
% of quantum particles	[10, 20, 30, 40, 50]
Number of dimensions per group	[4, 6, 8, 10,12]

6.1.5 Performance Measure

In this study, the collective mean fitness (CMF) proposed by Morrison [98] was employed as the performance measure for all experiments. CMF reflects algorithm performance across the entire range of landscape dynamics, and is given as:

$$CMF = \frac{\sum_{t=1}^{T} F(t)}{T} \tag{6.6}$$

where $F(t)$ is a measure of quality of the solution and T is the total number of iterations. CMF is quite interesting for dynamic problems because it captures an algorithm's entire performance history. This measure allows for convenient statistical comparison between algorithms [108]. The mean square error (MSE) calculated over the dataset during each epoch was used as the algorithm performance at each iteration.

The generalization factor ρ proposed in [117] was used to check the overfitting behavior of the algorithms used in this study. Overfitting is a phenomenon where NN performs well on training data but poorly on generalization data. The generalization factor is defined as $\rho = G_E/T_E$, where G_E and T_E are the generalization and training errors, respectively. A $\rho < 1$ is an indication of good generalization performance, while $\rho > 1$ is an indication of overfitting. The ρ was calculated in the same way as the CMF (equation (6.6)). Thus, all reported values of ρ reflect the generalization factor across the entire algorithm run.

A two-tailed non-parametric Mann-Whitney U test [93] was used to determine whether the difference in performance between two algorithms is statistically significant or not. The choice of the significance test is based on [39], where the authors showed that the Mann-Whitney U test is safer than parametric tests such as the t-test, since the Mann-Whitney U test assumes neither normal distributions of data nor homogeneity of variance. The null hypothesis, $H_0 : \mu_1 = \mu_2$, where μ_1 and μ_2 are the means of the two samples being compared, was evaluated at a significance level of 95%. The alternative hypothesis was defined as $H_1 : \mu_1 \neq \mu_2$. Thus, any p-value less than 0.05 corresponds to rejection of the null hypothesis that there was no statistically significant difference between the sample means. For the sake of convenience, all p-values were bounded below by 0.0001.

6.2 Results

This section presents and discusses the results obtained from the experiments carried out and the conclusion arrived at based on the overall findings from the experiments.

For each problem, the results are presented in three tables. The first table summarizes the CMF T_E and G_E with their confidence interval obtained in forecasting the problem under the nine dynamic scenarios considered. Also reported in the table are the ρ values. The p-values of the Mann Whitney U test between the algorithms are listed in the second table. The second table is given only when many of the pairs have have p-values greater than the threshold. The third table presents the performance ranking of the algorithms based on the CMF T_E and G_E values, taking into account the p-values.

To investigate performance of the training algorithms over time, both training and generalization errors were plotted against the iteration of the algorithms. The figures shown for each problem are only the results considered as representative or interesting because it is infeasible to show the results for all the nine dynamic scenarios of the ten problems used in the entire experiments.

6.2.1 SAM Time Series

Table 6.4 summarises the CMF T_E, G_E and ρ obtained by the algorithms in forecasting the SAM problem for the nine dynamic scenarios considered. Table 6.4 shows that CQSO produced the lowest training and generalization errors for all the scenarios. The CQSO algorithm also showed no sign of overfitting, with $\rho < 1$ for all the scenarios. The p-values indicates that there is a significant difference in performance between CQSO and the other two algorithms.

Table 6.4: SAM Time Series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:20)*			**A2** *(f:50, s:40)*			**A3** *(f:50, s:60)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.80E-04	3.03E-04	1.10	3.84E-04	3.85E-04	1.00	6.69E-04	6.48E-04	0.96
	±3.48E-05	±3.31E-05	±0.03	±4.29E-05	±4.55E-05	±0.01	±1.48E-04	±1.49E-04	±0.02
PSO	2.78E-04	3.76E-04	1.35	3.97E-04	2.74E-04	0.69	3.33E-04	3.29E-04	0.99
	±1.30E-05	±2.85E-05	±0.07	±1.81E-05	±2.02E-05	±0.03	±2.00E-05	±2.17E-05	±0.03
CQSO	1.45E-04	1.27E-04	0.88	1.59E-04	8.41E-05	0.53	1.39E-04	1.02E-04	0.73
	±2.95E-06	±2.36E-06	±0.01	±3.55E-06	±1.57E-06	±0.01	±3.30E-06	±2.75E-06	±0.00
	B1 *(f:100, s:20)*			**B2** *(f:100, s:40)*			**B3** *(f:100, s:60)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.77E-04	2.05E-04	1.18	2.68E-04	2.96E-04	1.12	3.87E-04	3.92E-04	1.02
	±2.49E-05	±2.36E-05	±0.02	±4.00E-05	±3.94E-05	±0.02	±5.56E-05	±5.42E-05	±0.02
PSO	2.37E-04	2.78E-04	1.17	3.34E-04	1.95E-04	0.58	2.62E-04	2.26E-04	0.86
	±7.00E-06	±2.00E-05	±0.07	±1.68E-05	±1.82E-05	±0.04	±9.74E-06	±1.51E-05	±0.04
CQSO	1.32E-04	1.19E-04	0.91	1.49E-04	8.13E-05	0.55	1.29E-04	9.36E-05	0.73
	±2.76E-06	±2.18E-06	±0.00	±2.99E-06	±1.03E-06	±0.01	±1.97E-06	±1.56E-06	±0.00
	C1 *(f:150, s:20)*			**C2** *(f:150, s:40)*			**C3** *(f:150, s:60)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.50E-04	1.91E-04	1.29	2.17E-04	2.73E-04	1.29	2.63E-04	2.74E-04	1.05
	±1.40E-05	±1.39E-05	±0.03	±3.45E-05	±3.32E-05	±0.03	±3.77E-05	±3.64E-05	±0.02
PSO	2.16E-04	2.05E-04	0.95	2.77E-04	1.44E-04	0.52	2.40E-04	2.10E-04	0.87
	±9.35E-06	±1.26E-05	±0.05	±1.17E-05	±1.11E-05	±0.03	±1.11E-05	±1.43E-05	±0.03
CQSO	1.29E-04	1.18E-04	0.91	1.44E-04	8.09E-05	0.56	1.24E-04	8.93E-05	0.72
	±2.48E-06	±1.93E-06	±0.00	±2.01E-06	±6.72E-07	±0.01	±1.90E-06	±1.57E-06	±0.00

The results showed that all the three algorithms performed better in scenarios with the lowest temporal severity (i.e. scenarios C1, C2 and C3). The errors produced by the algorithms increased with increase in temporal severity (i.e. the errors produced by the algorithms in B scenarios is lower than the errors obtained the A scenarios). For the scenarios with lowest spatial severity (i.e. A1, B1, and C1), the three algorithms again

Table 6.5: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.2675	0.2612	0.0001	0.0014	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0005	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0002	0.6789	0.0948	0.6789	0.0030
RROP vs CQSO	0.0246	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

produced the lowest errors compared to the scenarios with higher spatial severity.

For all scenarios, PSO showed no sign of overfitting except under scenarios A1 and B1, which indicated some slight overfitting behaviour. Considering that scenarios A1, B1 and C1 share the same step size with varying value of f (i.e. 50, 100 and 150 respectively), and that the level of overfitting decreased with increase in the value of f, it can be concluded that the overfitting was caused by too many iterations before the change (i.e. window sliding). The RPROP algorithms also showed some slight signs of overfitting behaviour under all scenarios except scenarios A2 and A3. Contrary to PSO, the overfitting behaviour shown by RPROP increased with increase in the value of f.

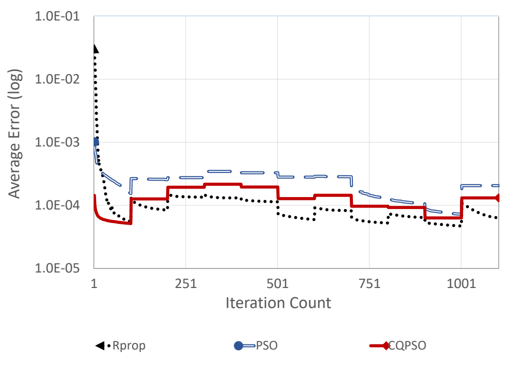
Table 6.6 lists the algorithm ranks based on the CMF T_E and G_E values in forecasting the SAM problem. Table 6.6 shows that CQSO had the highest average ranks under scenarios A, B and C. Hence, CQSO emerged as the overall winner among the training algorithms. There is no clear winner in the overall ranking between RPROP and the PSO

Table 6.6: SAM time series Algorithm Ranking for Scenarios A to C

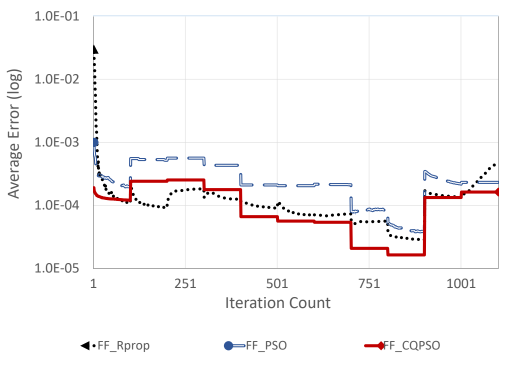
Algorithm	A1		A2		A3		**Average Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2.5	2	2.5	3	3	3	2.67	2.67
PSO	2.5	3	2.5	2	2	2	2.33	2.33
CQSO	1	1	1	1	1	1	1	1
	B1		B2		B3		**Average Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	2	2	3	3	3	2.33	2.67
PSO	3	3	3	2	2	2	2.67	2.33
CQSO	1	1	1	1	1	1	1	1
	C1		C2		C3		**Average Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	2.5	2	2.5	2.5	3	2.16	2.67
PSO	3	2.5	3	2.5	2.5	2	2.83	2.33
CQSO	1	1	1	1	1	1	1	1
	Average R(A)		Average R(B)		Average R(C)		**Overall Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2.67	2.67	2.33	2.67	2.16	2.67	2.38	2.67
PSO	2.33	2.33	2.67	2.33	2.83	2.33	2.61	2.33
CQSO	1	1	1	1	1	1	1	1

algorithm. Thus, in terms of training performance, RPROP achieved 2^{nd} and PSO 3^{rd} ranking positions, while in terms of generalization, PSO ranked overall 2^{nd} and RPROP 3^{rd}.

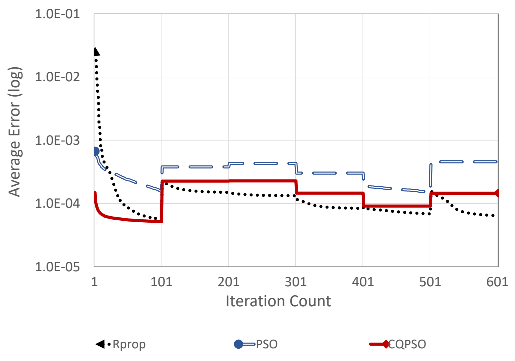
Figure 6.4 illustrates the progression of the error over time, obtained for the SAM problem under scenarios B1, B2, and B3. As visualized in Figure 6.4, all the training algorithms showed stable performance progression with initial increases in peaks due to environment changes, and later adapted well to environmental changes toward the middle to the end. The figure shows that CQSO adapted better by producing the lowest generalization errors during almost the entire algorithm run. The CQSO produced the lowest initial errors and took about ten algorithm iterations to locate a minimum



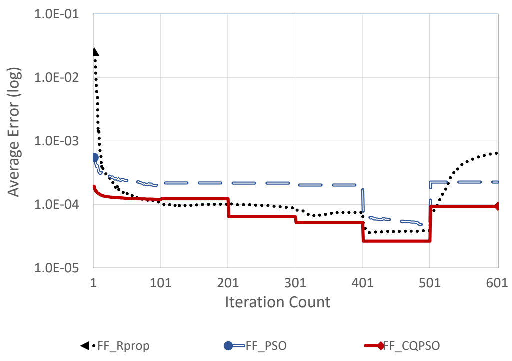
(a) T_E for B1



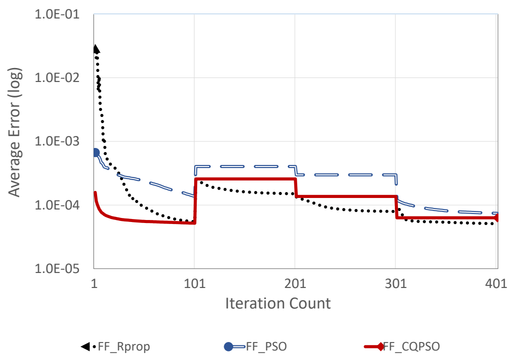
(b) G_E for B1



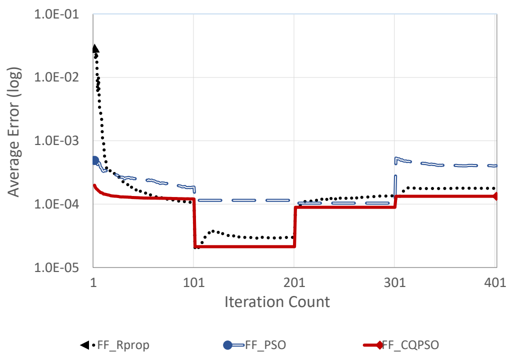
(c) T_E for B2



(d) G_E for B2



(e) T_E for B3



(f) G_E for B3

Figure 6.4: Training and generalization error results for SAM time series, scenarios B1 to B3

under all scenarios, while the other algorithms took about 100 algorithm iterations. This indicates that CQSO benefited from the component wise optimization. Once the algorithms located a minimum, they constantly tracked it and successfully recovered from changes throughout the experiment.

6.2.2 HIT Time Series

Table 6.7 presents the CMF T_E, G_E, and ρ values obtained in forecasting the HIT time series. Table 6.8 lists the p-values obtained from the pairwise comparisons between the algorithms.

Table 6.7 shows that RPROP produced the worst errors, while CQSO yielded the best errors under all the scenarios. The p-values in Table 6.8 indicate that the difference in training and generalization performance between any two of the training algorithms were significant for all scenarios.

The ρ values reported in Table 6.7 indicate that RPROP did not overfit under any of the nine scenarios. The PSO algorithm exhibited slight overfitting behaviour under scenarios B1 and C1. This slight overfitting behaviour was due to training on the sliding window for too long before the window slides, since PSO did not show any sign of overfitting under scenario A1 (where $f = 50$), and the ρ value under scenario B1 (where $f = 100$) is lower than under scenario C1 (where $f = 150$). The CQSO overfitted under the scenarios A1, B1 and C1. Since these scenarios have smaller step sizes, the sliding window retains a larger amount of stale data, which the algorithm fitted too well.

Table 6.9 presents the performance ranking of the training algorithms in forecasting the HIT problem. As shown in Table 6.9, the CQSO algorithm had the best average training and generalization ranks for the A, B and C scenarios. Thus, CQSO achieved the overall first ranking position. The PSO and RPROP algorithms achieved the overall second and third positions, respectively.

Figure 6.5 illustrates the progression of T_E and G_E over time, obtained by the training algorithms under scenarios A1 to A3. Scenarios A1 to A3 simulated frequent changes (after every 50 iterations). As shown in Figure 6.5, all the algorithms exhibited stable performance progression. The CQSO took about five algorithm iterations to locate a minimum while the PSO and the RPROP took about 20 and 30 algorithm iterations, respectively. The figure shows that after about 30 initial iterations, RPROP and CQSO produced similar performance throughout the algorithm runs. Figure 6.5 also show that

PSO had the worst performance throughout the algorithm runs.

6.2.3 DMT Time series

Table 6.10 presents the CMF T_E and G_E, and also the ρ results obtained by the training algorithms in forecasting the DMT time series. Table 6.10 reports the p-values of the

Table 6.7: HIT Time Series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:100)*			**A2** *(f:50, s:250)*			**A3** *(f:50, s:584)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.42E-04	2.41E-04	1.00	4.19E-04	4.18E-04	0.99	1.56E-03	1.55E-03	0.99
	±8.36E-05	±8.33E-05	±0.00	± 1.08E-04	± 1.09E-04	± 0.01	± 4.59E-04	± 4.58E-04	± 0.01
PSO	2.83E-05	2.78E-05	0.99	5.15E-05	4.85E-05	0.93	1.21E-04	1.18E-04	0.94
	±4.38E-06	±4.36E-06	± 0.02	± 7.41E-06	± 7.68E-06	± 0.02	± 5.38E-05	± 5.62E-05	± 0.03
CQSO	4.59E-06	5.62E-06	1.22	5.55E-06	5.17E-06	0.93	1.35E-05	1.24E-05	0.93
	±2.25E-07	±3.05E-07	±0.001	± 3.69E-07	± 3.21E-07	± 0.01	± 1.20E-06	± 1.04E-06	± 0.02
	B1 *(f:100, step s:100)*			**B2** *(f:100, step s:250)*			**B3** *(f:100, step s:584)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.80E-04	1.77E-04	0.99	1.88E-04	1.84E-04	0.99	6.06E-04	6.06E-04	1.00
	±5.67E-05	±5.58E-05	±0.09	± 4.78E-05	± 4.59E-05	± 0.01	± 1.94E-04	± 1.95E-04	± 0.00
PSO	1.85E-05	1.89E-05	1.03	3.11E-05	2.70E-05	0.86	5.24E-05	4.69E-05	0.89
	±2.45E-06	±2.48E-06	±0.02	± 5.90E-06	± 5.44E-06	± 0.02	± 5.85E-06	± 5.63E-06	± 0.02
CQSO	1.20E-05	1.35E-05	1.16	5.01E-06	4.67E-06	0.93	6.75E-06	6.38E-06	0.95
	±1.82E-06	±1.64E-06	±0.04	± 2.11E-07	± 1.86E-07	± 0.00	± 5.10E-07	± 4.26E-07	± 0.01
	C1 *(f:150, step s:100)*			**C2** *(f:150, step s:250)*			**C3** *(f:150, step s:584)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	8.92E-05	8.74E-05	0.98	1.45E-04	1.44E-04	0.99	4.11E-04	4.07E-04	1.00
	±3.04E-05	±2.96E-05	±0.01	± 4.94E-05	± 5.00E-05	± 0.01	± 1.43E-04	± 1.40E-04	± 0.00
PSO	1.48E-05	1.57E-05	1.07	2.06E-05	1.73E-05	0.84	4.42E-05	3.65E-05	0.82
	±1.93E-06	±1.80E-06	±0.02	± 1.44E-06	± 1.29E-06	± 0.01	± 5.48E-06	± 5.31E-06	± 0.02
CQSO	3.88E-06	4.69E-06	1.21	5.08E-06	4.73E-06	0.93	6.18E-06	5.65E-06	0.92
	±1.33E-07	±1.59E-07	±0.00	± 2.09E-07	± 1.78E-07	± 0.01	± 3.93E-07	± 3.40E-07	± 0.01

Table 6.8: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

pairwise comparisons between errors produced by the algorithms.

As shown in Table 6.10, the CQSO yielded the lowest cumulative mean T_E and G_E compared to the PSO and the Rprop for all nine dynamic scenarios. Rprop produced the worst performance among the algorithms under all scenarios.

Table 6.11 lists the p-values of the Mann Whitney U test between the algorithms in forecasting the DMT problem. The p–values in Table 6.11 indicate that the difference in performance between the algorithms were significant.

The ρ values given in Table 6.10 show that under all the scenarios, the training algorithms exhibited just slight or no overfiiting behaviour. The slight overfitting was due to training on the sliding window for too long before the window slides, as seen in Section 6.2.2.

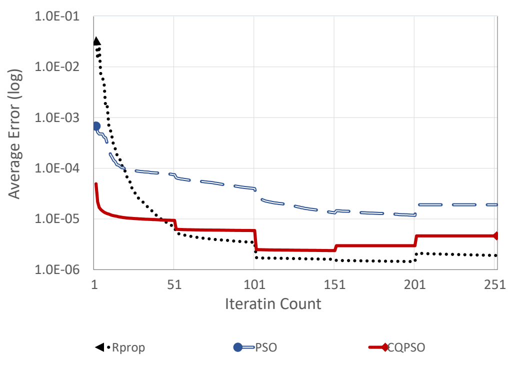
Table 6.12 lists the performance ranking of the algorithms in forecasting the DMT problem under all scenarios, based on the CMF G_E and T_E values given in Table 6.10.

Table 6.9: HIT Time Series Algorithm Ranking for Scenarios A to C

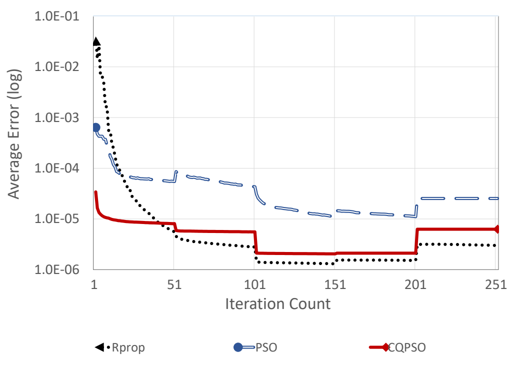
Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	Average R(A)		Average R(B)		Average R(C)		**Overall Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1

Table 6.12 illustrates that CQSO obtained the highest rank for all scenarios. The PSO achieved the second highest average rank, while RPROP had the lowest average rank. Thus, CQSO emerged as the overall winner.

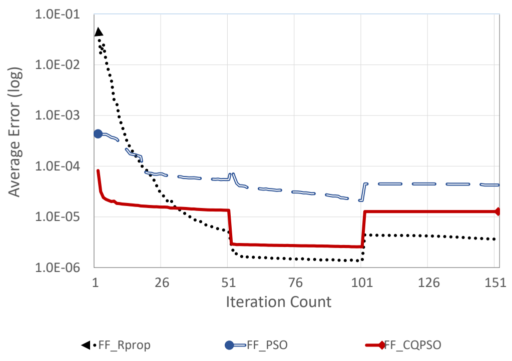
Figure 6.6 illustrates the progression of T_E and G_E over time, obtained by the algorithms under scenarios C1, C2 and C3. Scenarios C1 to C3 simulated low change frequency (after every 150 iterations). As shown in Figure 6.6, all the algorithms exhibited stable and similar performance progression after about 100 iterations. The CQSO achieved best accuracy faster (within the first five iterations) than the PSO and RPROP, which took about 50 algorithm iterations. This shows that CQSO must have benefited



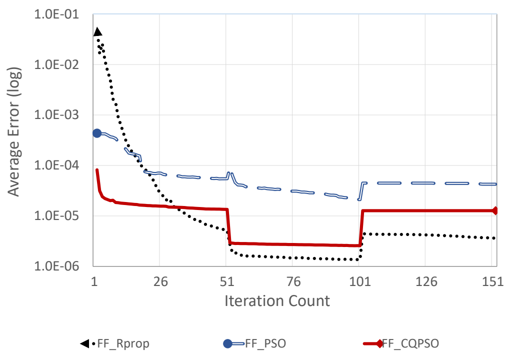
(a) T_E for A1



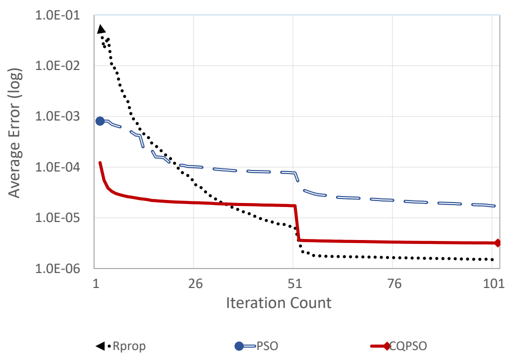
(b) G_E for A1



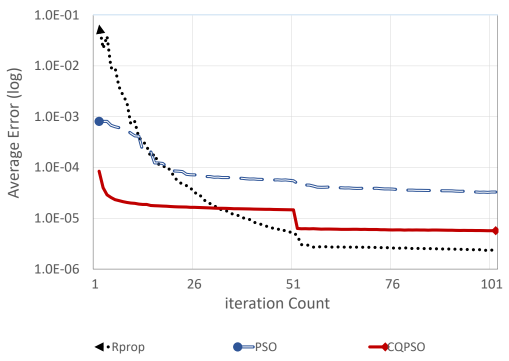
(c) T_E for A2



(d) G_E for A2



(e) T_E for A3



(f) G_E for A3

Figure 6.5: Training and generalization error Results for HIT Time Series, Scenarios A1 to A3

from the component wise optimization strategy or had better exploration due to the QSO used in the subswarms, which maintains diversity.

Table 6.10: DMT Time Series Results

Algorithm	**Scenario**								
	A1 *(f:10, s:200)*			**A2** *(f:10, s:400)*			**A3** *(f:10, s:510)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	5.65E-04	5.66E-04	1.01	1.50E-03	1.51E-03	1.01	1.78E-03	1.77E-03	0.99
	±1.61E-04	±1.60E-04	±0.00	±3.80E-04	±3.80E-04	±0.01	±4.59E-04	±4.56E-04	±0.01
PSO	2.93E-05	3.05E-05	1.04	5.34E-05	5.84E-05	1.17	4.37E-05	4.27E-05	0.98
	±6.34E-06	±6.78E-06	±0.02	±1.76E-05	±1.76E-05	±0.08	±1.26E-05	±1.22E-05	±0.03
CQSO	1.19E-05	1.24E-05	1.04	1.19E-05	1.39E-05	1.17	1.20E-05	1.30E-05	1.09
	±2.43E-07	±2.75E-07	±0.00	±3.12E-07	±2.81E-07	±0.01	±2.93E-07	±2.40E-07	±0.01
	B1 *(f:50, s:200)*			**B2** *(f:50, s:400)*			**B3** *(f:50, s:510)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.46E-04	1.47E-04	1.01	3.01E-04	3.02E-04	1.01	2.76E-04	2.75E-04	1.00
	±3.82E-05	±3.80E-05	±0.01	±8.28E-05	±8.25E-05	±0.01	±9.55E-05	±9.49E-05	±0.01
PSO	1.59E-05	1.61E-05	1.02	2.00E-05	2.32E-05	1.19	2.01E-05	2.06E-05	1.03
	±2.89E-06	±2.62E-06	±0.01	±3.50E-06	±3.16E-06	±0.03	±2.45E-06	±2.46E-06	±0.02
CQSO	1.10E-05	1.17E-05	1.06	1.09E-05	1.30E-05	1.20	1.09E-05	1.19E-05	1.10
	±1.29E-07	±1.39E-07	±0.00	±1.30E-07	±1.59E-07	±0.00	±1.60E-07	±2.10E-07	±0.00
	C1 *(f:100, s:200)*			**C2** *(f:100, s:400)*			**C3** *(f:100, s:510)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	5.40E-05	5.51E-05	1.03	8.54E-05	8.77E-05	1.04	1.69E-04	1.69E-04	1.01
	±1.36E-05	±1.37E-05	±0.01	±2.07E-05	±2.07E-05	±0.01	±4.64E-05	±4.60E-05	±0.01
PSO	1.38E-05	1.43E-05	1.03	1.75E-05	2.00E-05	1.16	1.62E-05	1.70E-05	1.05
	±8.17E-07	±8.69E-07	±0.01	±2.94E-06	±2.91E-06	±0.02	±1.36E-06	±1.39E-06	±0.02
CQSO	1.08E-05	1.15E-05	1.06	1.07E-05	1.28E-05	1.20	1.08E-05	1.17E-05	1.09
	±1.13E-07	±1.31E-07	±0.00	±1.29E-07	±1.62E-07	±0.00	±2.97E-07	±2.90E-07	±0.00

6.2.4 Mackay Glass

Table 6.13 show that, for all nine dynamic scenarios, CQSO consistently produced the lowest training and generalization errors, while RPROP yielded the worst errors. The p-values in Table 6.14 indicate that the difference in performance between the algorithms

Table 6.11: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%, for scenarios A1 to A3

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

were statistically significant.

Table 6.13 also illustrates that increased spatial severity of changes (i.e. from A1 to A3, B1 to B3, and C1 to C3) made adaption to the changes more difficult for the training algorithms, since both training and generalization errors increased with an increase in spatial severity. The table also indicates that a decrease in temporal severity (such as from A1 to B1 and to C1) improved the performance of the training algorithms.

All the three training algorithms showed no sign of overfitting, as indicated by the ρ values given in Table 6.13. However, RPROP had the largest ratio between the generalization and training error.

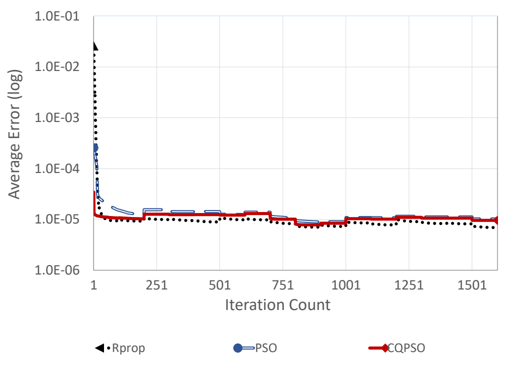
Figure 6.7 illustrates the progression of the algorithms over time on the MG problem under scenarios A1, A2, and A3. As visualized in the figure, CQSO and RPROP clearly outperformed the PSO throughout the algorithm runs. Within the first 100 iterations,

Table 6.12: DMT Time Series Algorithm Ranking for Scenarios A to C

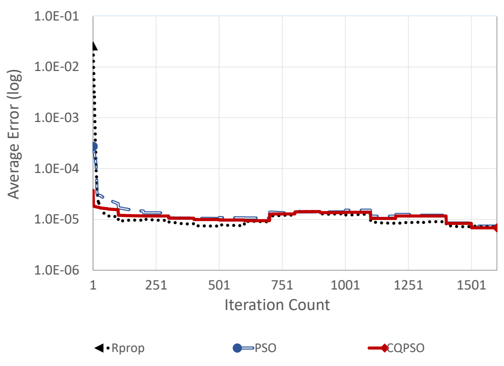
Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	Average R(A)		Average R(B)		Average R(C)		**Overall Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1

CQSO yielded the lowest errors, but after that, RPROP improved, and produced similar performance to CQSO throughout the remaining training time.

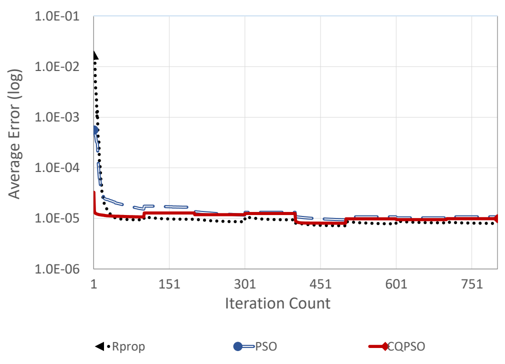
Table 6.15 shows that, for the A, B and C scenarios, the CQSO algorithm obtained the highest average rank, both in terms of T_E and G_E, while RPROP obtained the lowest average ranks. Thus, CQSO emerged as the overall winner.



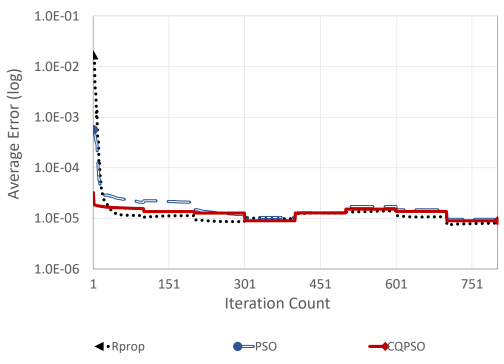
(a) T_E for C1



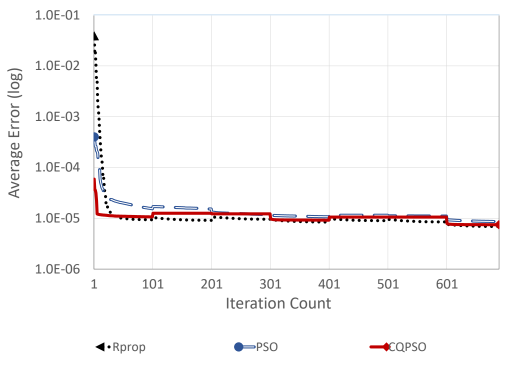
(b) G_E for C1



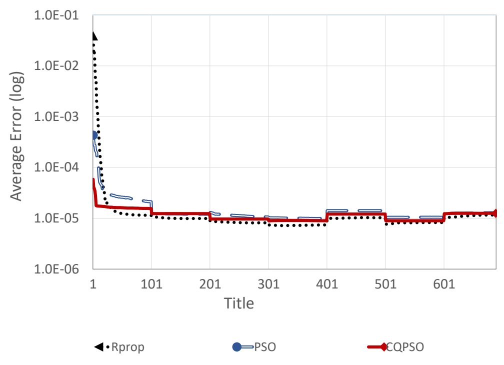
(c) T_E for C2



(d) G_E for C2



(e) T_E for C3



(f) G_E for C3

Figure 6.6: Training and generalization error Results for DMT Time Series, Scenarios C1 to C3

Table 6.13: MG Time Series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:30)*			**A2** *(f:50, s:60)*			**A3** *(f:50, s:84)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.50E-04	2.40E-04	0.96	4.43E-04	4.28E-04	0.96	6.21E-04	5.94E-04	0.95
	±2.76E-05	±2.60E-05	±0.01	±5.29E-05	±5.21E-05	±0.01	±7.58E-05	±7.54E-05	±0.02
PSO	7.85E-05	3.87E-05	0.50	1.01E-04	5.44E-05	0.56	1.27E-04	5.41E-05	0.44
	±5.21E-06	±2.92E-06	±0.03	±9.72E-06	±4.31E-06	±0.04	±1.17E-05	±4.49E-06	±0.03
CQSO	5.69E-06	3.50E-06	0.62	8.66E-06	5.80E-06	0.67	1.03E-05	8.31E-06	0.82
	±4.29E-07	±2.33E-07	±0.02	±9.69E-07	±6.63E-07	±0.02	±9.17E-07	±7.09E-07	±0.03
	B1 *(f:100, s:30)*			**B2** *(f:100, s:60)*			**B3** *(f:100, s:84)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.55E-04	2.46E-04	0.96	4.32E-04	4.19E-04	0.97	2.84E-04	2.73E-04	0.96
	±2.58E-05	±2.64E-05	±0.01	±4.25E-05	±4.24E-05	±0.01	±3.01E-05	±2.97E-05	±0.01
PSO	3.89E-05	1.88E-05	0.48	6.95E-05	4.33E-05	0.62	8.47E-05	3.97E-05	0.47
	±3.93E-06	±2.60E-06	±0.04	±6.95E-06	±5.86E-06	±0.04	±6.22E-06	±3.80E-06	±0.03
CQSO	4.22E-06	2.74E-06	0.65	5.29E-06	3.84E-06	0.73	5.93E-06	5.22E-06	0.89
	±2.65E-07	±1.53E-07	±0.02	±5.17E-07	±3.51E-07	±0.02	±5.81E-07	±4.58E-07	±0.02
	C1 *(f:200, s:30)*			**C2** *(f:200, s:60)*			**C3** *(f:200, s:84)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	6.43E-05	6.23E-05	0.97	1.09E-04	1.05E-04	0.96	1.37E-04	1.31E-04	0.96
	±8.34E-06	±7.98E-06	±0.01	±9.43E-06	±9.53E-06	±0.02	±1.36E-05	±1.32E-05	±0.02
PSO	2.10E-05	9.41E-06	0.44	3.82E-05	2.24E-05	0.58	4.60E-05	2.28E-05	0.50
	±2.13E-06	±1.36E-06	±0.03	±3.68E-06	±2.78E-06	±0.05	±4.24E-06	±2.46E-06	±0.04
CQSO	3.32E-06	2.53E-06	0.77	4.01E-06	3.12E-06	0.78	4.34E-06	4.42E-06	1.05
	±1.53E-07	±8.79E-08	±0.02	±1.98E-07	±1.23E-07	±0.02	±3.51E-07	±1.77E-07	±0.06

6.2.5 Lorenz Time Series

Table 6.16 clearly shows that, under all nine scenarios for the Lorenz time series, CQSO outperformed the PSO and the RPROP algorithms by producing the lowest cumulative mean training and generalization errors. It is also clearly illustrated in the table that

Table 6.14: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

PSO produced lower errors in terms of both training and generalization, when compared to RPROP under all scenarios. The p-values given in Table 6.17 confirmed that the differences in performance between the algorithms were significant for all the scenarios.

Table 6.16 also shows that the training and generalization errors produced by the algorithms increased with increase in spatial severity, a phenomenon that indicates that adaptation became more difficult for the algorithms as spatial severity of changes increased. Performance of the algorithms also improved with a decrease in temporal severity.

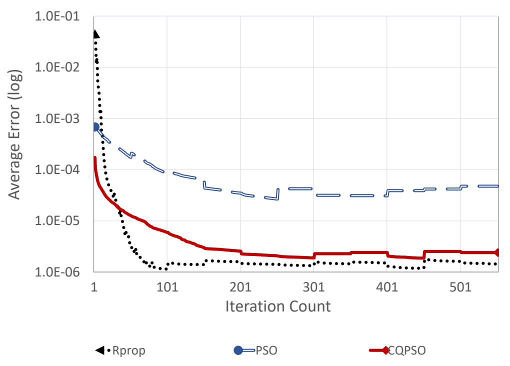
The ρ values in Table 6.16 indicate that none of the training algorithms overfitted. RPROP, however, produced the highest ρ value (equal to 1 under all the scenarios), which is an indication that RPROP produced the largest ratio between generalization and training error, than the PSO and CQSO (as seen in Section 6.2.4).

Table 6.15: MG Time Series Algorithm Ranking for Scenarios A to C

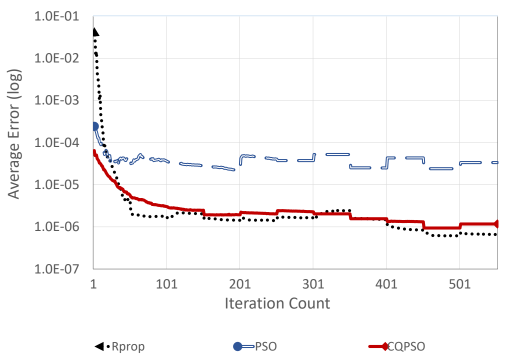
Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	Average R(A)		Average R(B)		Average R(C)		**Overall Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1

Figure 6.8 shows for scenarios C1, C2, and C3 that the CQSO produced the lowest errors within the initial 50 iterations, while RPROP had the worst errors. Thereafter, RPROP's performance improved and outperformed the CQSO for the remaining training time. After the initial 50 iterations, PSO clearly produced the worst performance throughout the remaining training time.

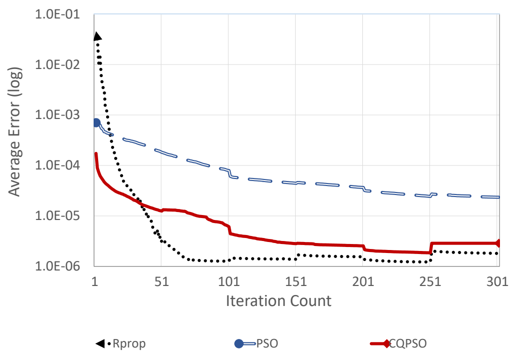
Table 6.18 shows that the CQSO and the RPROP algorithms obtained the overall highest and lowest average ranks, respectively. Thus, CQSO emerged as the overall winner.



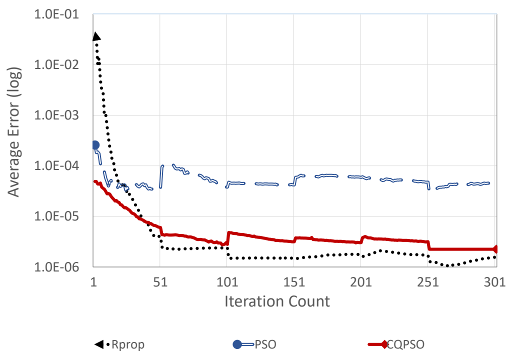
(a) T_E for A1



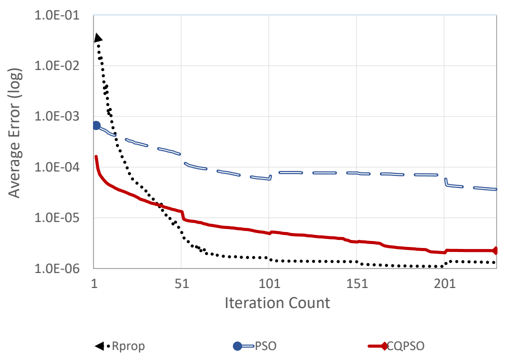
(b) G_E for A1



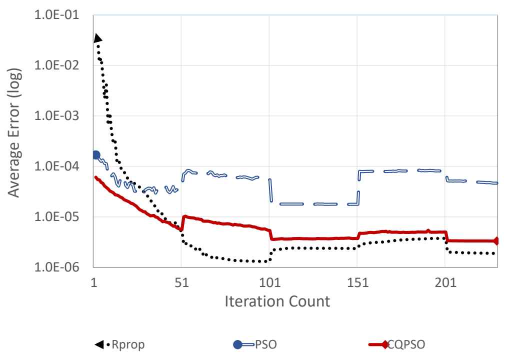
(c) T_E for A2



(d) G_E for A2



(e) T_E for A3



(f) G_E for A3

Figure 6.7: Training and generalization error Results for MG Time Series, Scenatios A1 to A3

Table 6.16: Lorenz Time Series Results

Algorithm	**Scenario**								
	A1 *(f:10, s:100)*			**A2** *(f:10, s:250)*			**A3** *(f:10, s:330)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	3.66E-04 ±7.92E-05	3.65E-04 ±7.85E-05	1.00 ±0.01	7.77E-04 ± 1.17E-04	7.76E-04 ± ±1.16E-04	1.00 ±0.01	9.69E-04 ±1.52E-04	9.66E-04 ±1.51E-04	1.00 ±0.01
PSO	2.42E-05 ±5.75E-06	2.38E-05 ±5.29E-06	0.98 ±0.05	4.55E-05 ±1.24E-05	4.50E-05 ±1.18E-05	1.00 ±0.03	5.31E-05 ±1.66E-05	5.05E-05 ±1.51E-05	0.98 ±0.06
CQSO	3.03E-06 ±3.78E-07	1.99E-06 ±2.61E-07	0.66 ±0.03	4.72E-06 ±7.44E-07	3.80E-06 ±7.02E-07	0.78 ±0.03	6.86E-06 ±1.22E-06	5.81E-06 ±1.07E-06	0.85 ±0.04
	B1 *(f:50, s:100)*			**B2** *(f:50, s:250)*			**B3** *(f:50, s:330)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	6.71E-05 ±1.28E-05	6.72E-05 ±1.30E-05	1.00 ±0.01	1.64E-04 ±3.69E-05	1.63E-04 ±3.63E-05	1.00 ±0.01	1.97E-04 ±3.26E-05	1.97E-04 ±3.32E-05	1.00 ±0.01
PSO	1.03E-05 ±3.20E-06	8.73E-06 ±3.06E-06	0.80 ±0.06	1.68E-05 ±7.25E-06	1.63E-05 ±7.64E-06	0.94 ±0.04	2.94E-05 ±1.06E-05	2.92E-05 ±1.17E-05	0.94 ±0.06
CQSO	8.40E-07 ±1.60E-07	4.47E-07 ±8.65E-08	0.53 ±0.02	1.62E-06 ±1.83E-07	1.18E-06 ±1.38E-07	0.73 ±0.01	2.08E-06 ±2.32E-07	1.53E-06 ±1.61E-07	0.74 ±0.03
	C1 *(f:100, s:100)*			**C2** *(f:100, s:250)*			**C3** *(f:100, s:330)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	3.13E-05 ±5.20E-06	3.13E-05 ±5.29E-06	1.00 ±0.01	7.13E-05 ±8.60E-06	7.09E-05 ±8.61E-06	1.00 ±0.01	1.16E-04 ±1.71E-05	1.16E-04 ±1.71E-05	1.00 ±0.01
PSO	5.09E-06 ±1.25E-06	3.67E-06 ±1.15E-06	0.67 ±0.05	1.58E-05 ±4.96E-06	1.46E-05 ±4.77E-06	0.90 ±0.05	1.20E-05 ±4.41E-06	1.03E-05 ±3.73E-06	0.85 ±0.05
CQSO	5.15E-07 ±6.51E-08	3.21E-07 ±3.71E-08	0.63 ±0.02	1.02E-06 ±1.31E-07	6.87E-07 ±8.21E-08	0.68 ±0.01	1.12E-06 ±1.10E-07	7.25E-07 ±7.43E-08	0.65 ±0.02

6.2.6 IAP Time Series

Table 6.19 and Table 6.20 show that the PSO produced the worst errors compared to CQSO and RPROP under all scenarios, with statistical significance.

For the gradually changing scenarios A1, B1 and C1, RPROP significantly outperformed CQSO, except for B1, where CQSO produced a lower G_E. For the severely

Table 6.17: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

changing scenarios (i.e. A2, B2 and C2), CSQO produced significantly lower G_E compared to RPROP. The CQSO also outperformed RPROP in terms of T_E for scenario B2. The two algorithms, however, had similar T_E for scenarios A2 and C2. For the abruptly changing scenarios A3, B3 and C3, CQSO produced significantly lower errors than RPROP, except for A3, where the two algorithms produced similar performance. All these show that RPROP performed better than CQSO for the gradually changing scenarios, while CQSO outperformed the RPROP for the severely and abruptly changing scenarios.

It is observed that the performance of both RPROP and PSO improved for the C scenarios, while that of CQSO deteriorated compared to the corresponding performance for the B scenarios. Since the temporal severity of the C scenarios is lower, the improvement or deterioration in performance was caused by the higher number of iterations allowed

Table 6.18: Lorenz Time Series Algorithm Ranking for Scenarios A to C

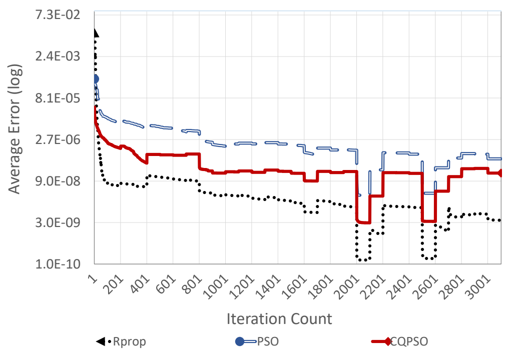
Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1
	Average R(A)		Average R(B)		Average R(C)		**Overall Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	3	3	3	3	3	3	3	3
PSO	2	2	2	2	2	2	2	2
CQSO	1	1	1	1	1	1	1	1

before any change (i.e. window sliding).

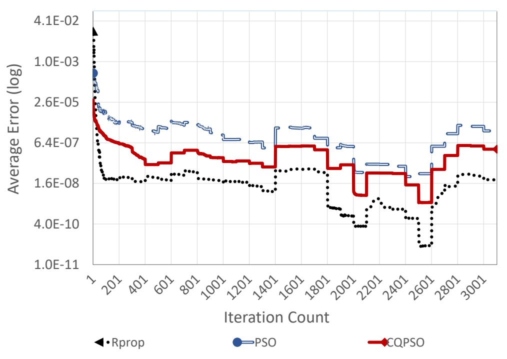
The ρ values in Table 6.19 illustrate that all training algorithms exhibited overfitting behaviour, which was due to training for too long on a small dataset (since the NNs parameters were optimized).

Table 6.21 shows that the CQSO achieved the overall best rank, RPROP came second, and the PSO came last.

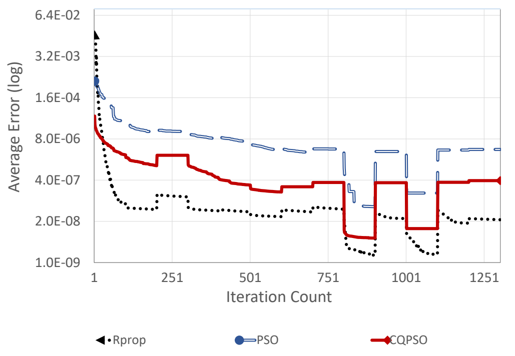
Figure 6.9 shows the error progression for the three algorithms throughout the algorithms runs. The errors produced by the algorithms peak out at each environmental



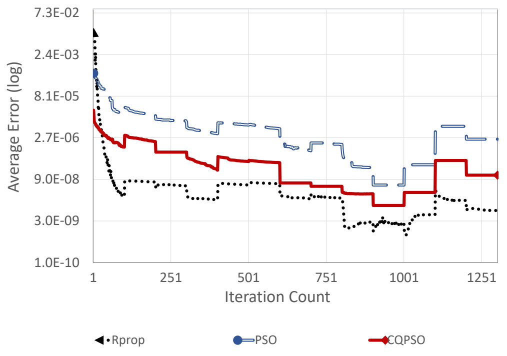
(a) T_E for C2



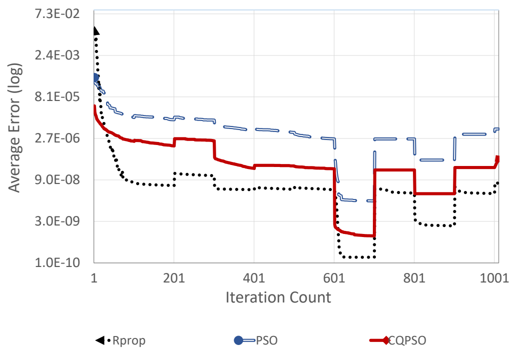
(b) G_E for C2



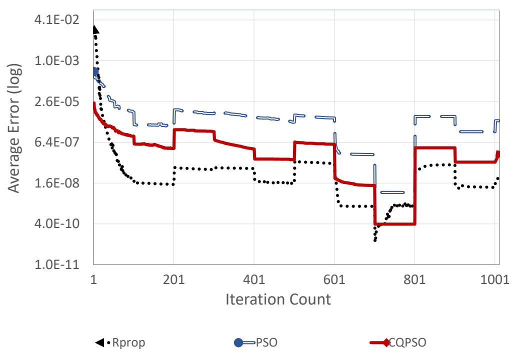
(c) T_E for C3



(d) G_E for C3



(e) T_E for C4



(f) G_E for C4

Figure 6.8: Training and generalization error Results for Lorenz Time Series, Scenarios C1 to C3

change. RPROP produced the highest peaks after the changes, but it recovered before another change occurred. CQSO produced the lowest peaks after all of the changes.

Table 6.19: IAP Time series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:10)*			**A2** *(f:50, s:25)*			**A3** *(f:50, s:32)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.82E-04 ±3.51E-05	2.28E-04 ±3.08E-05	1.36 ±0.09	3.47E-04 ±4.47E-05	4.73E-04 ±5.59E-05	1.39 ±0.07	4.59E-04 ±1.32E-04	5.55E-04 ±1.24E-04	1.29 ±0.09
PSO	2.52E-03 ±4.36E-04	2.75E-03 ±4.90E-04	1.08 ±0.02	1.95E-03 ±2.35E-04	2.75E-03 ±3.15E-04	1.42 ±0.05	1.73E-03 ±2.21E-04	3.39E-03 ±3.99E-04	1.98 ±0.03
CQSO	3.10E-04 ±2.03E-04	3.64E-04 ±2.38E-04	1.21 ±0.06	3.89E-04 ±1.32E-04	4.30E-04 ±1.36E-04	1.30 ±0.13	2.76E-04 ±1.07E-04	5.57E-04 ±2.13E-04	1.98 ±0.12
	B1 *(f:100, s:10)*			**B2** *(f:100, s:25)*			**B3** *(f:100, s:32)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	9.80E-05 ±2.27E-05	1.36E-04 ±2.10E-05	1.53 ±0.11	2.02E-04 ±3.87E-05	2.78E-04 ±3.84E-05	1.45 ±0.08	2.61E-04 ±5.72E-05	3.58E-04 ±5.60E-05	1.52 ±0.12
PSO	1.44E-03 ±2.03E-04	1.57E-03 ±2.30E-04	1.08 ±0.02	1.16E-03 ±1.69E-04	1.76E-03 ±2.40E-04	1.55 ±0.06	1.16E-03 ±1.56E-04	2.33E-03 ±2.88E-04	2.02 ±0.03
CQSO	8.84E-05 ±2.50E-05	9.78E-05 ±2.97E-05	1.08 ±0.06	1.30E-04 ±4.91E-05	1.70E-04 ±5.01E-05	1.54 ±0.13	8.91E-05 ±2.04E-05	1.51E-04 ±4.75E-05	1.59 ±0.18
	C1 *(f:150, s:10)*			**C2** *(f:150, s:25)*			**C3** *(f:150, s:32)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	6.07E-05 ±1.03E-05	1.05E-04 ±1.11E-05	1.86 ±0.12	1.41E-04 ±3.53E-05	2.11E-04 ±3.12E-05	1.71 ±0.17	1.75E-04 ±3.60E-05	2.75E-04 ±3.22E-05	1.80 ±0.19
PSO	9.41E-04 ±2.04E-04	1.04E-03 ±2.29E-04	1.09 ±0.01	9.54E-04 ±1.76E-04	1.33E-03 ±2.41E-04	1.45 ±0.09	9.64E-04 ±1.64E-04	1.95E-03 ±3.07E-04	2.06 ±0.04
CQSO	1.43E-04 ±4.59E-05	1.61E-04 ±5.49E-05	1.14 ±0.08	1.43E-04 ±4.59E-05	1.61E-04 ±5.49E-05	1.14 ±0.08	1.00E-04 ±2.43E-05	1.28E-04 ±3.02E-05	1.33 ±0.16

PSO produced the worst performance throughout the remaining training time.

Table 6.20: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.1984	0.0003	0.0001	0.0002	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0385	0.3292	0.0010	0.0101	0.0413	0.0625
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.3750	0.0002	0.5250	0.0002	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

6.2.7 S&P 500 Time Series

Table 6.22 shows that the CQSO yielded the lowest T_E and G_E for all scenarios. The p-values in Table 6.23 confirmed that the performance of CQSO was significantly superior to RPROP and PSO, except for scenario B2 where PSO produced similar G_E. In general, the errors produced by all three algorithms worsen as temporal severity increased.

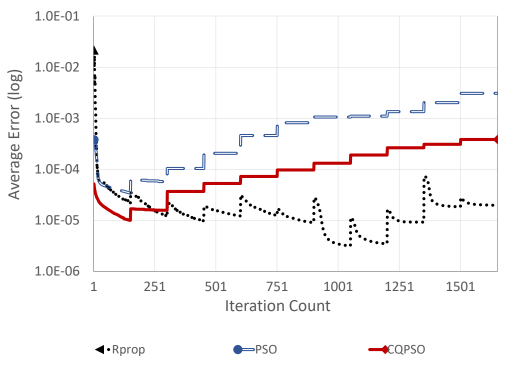
The ρ values reported in Table 6.22 indicate that all the training algorithms overfitted. Table 6.24 shows that, the CQSO obtained the highest average rank in A, B and C scenarios. Thus, the CSQO achieved the overall best rank, the RPROP came second, and the PSO came last.

Figure 6.10 shows the progression of the algorithms over time in forecasting the S&P problem for scenarios C1, C2, and C3. As visualized in the figure, the CQSO

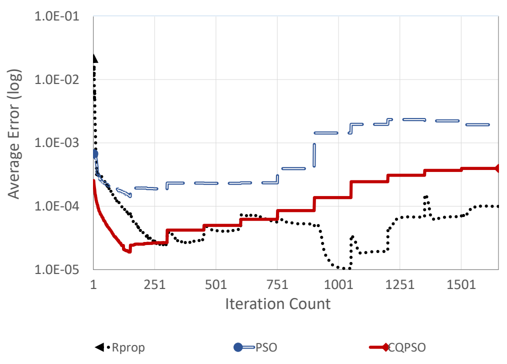
Table 6.21: AIP Algorithm Ranking for scenarios A to C

Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	1	1.5	2	2	1.5	1.5	1.5
PSO	3	3	3	3	3	3	3	3
CQSO	2	2	1.5	1	1	1.5	1.5	1.5
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1.5	2	2	2	2	2	1.83	2
PSO	3	3	3	3	3	3	3	3
CQSO	1.5	1	1	1	1	1	1.75	1
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	1	1.5	2	2	2	1.5	1.67
PSO	3	3	3	3	3	3	3	3
CQSO	2	2	1.5	1	1	1	1.5	1.33
	Average R(A)		Average R(B)		Average R(C)		**Overall Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1.5	1.5	1.83	2	1.5	1.67	1.61	1.72
PSO	3	3	3	3	3	3	3	3
CQSO	1.5	1.5	1.75	1	1.5	1.33	1.58	1.28

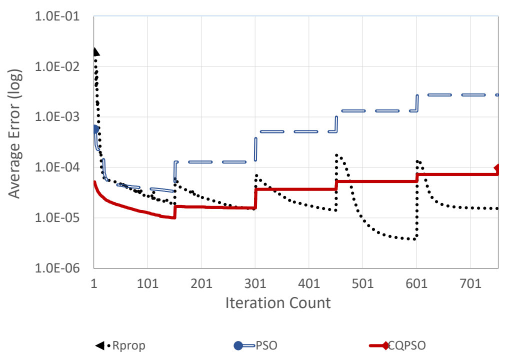
produced the lowest errors within the initial 50 iterations, while RPROP had the worst errors. After the 50 iterations, PSO produced the worst performance throughout the remaining algorithm run. The CQSO benefitted from its component wise optimization strategy and the QSO used in the subswarms, since it outperformed PSO before the first environmental change for all scenarios. Also, the CSQO adapted better than PSO after all environmental changes. However, RPROP tracked the changing minima better than the CQSO throughout the algorithm runs. The figures also illustrate that CQSO's adaption to dynamic changes improved with increase in spatial severity.



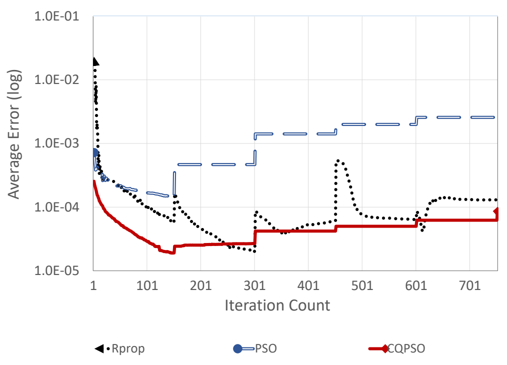
(a) T_E for C1



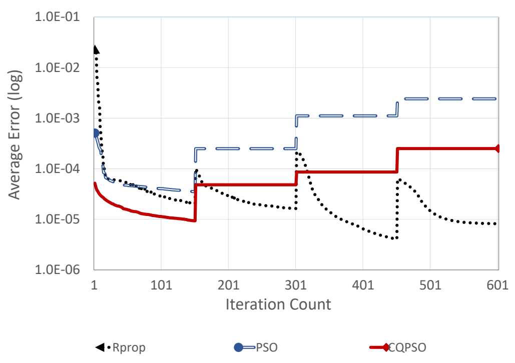
(b) G_E for C1



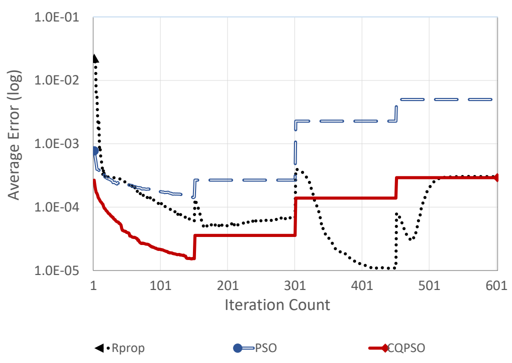
(c) T_E for C2



(d) G_E for C2



(e) T_E for C3



(f) G_E for C3

Figure 6.9: Training and generalization error results for IAP time series, scenarios C1 to C3

Table 6.22: S&P Time Series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:20)*			**A2** *(f:50, s:40)*			**A3** *(f:50, s:58)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	3.62E-04	5.15E-04	1.46	6.54E-04	8.43E-04	1.31	8.97E-04	1.10E-03	1.24
	± 3.22E-05	± 3.28E-05	± 0.07	± 6.00E-05	± 6.39E-05	± 0.04	± 6.07E-05	± 7.40E-05	± 0.04
PSO	7.88E-04	9.68E-04	1.25	8.11E-04	1.18E-03	1.47	5.42E-04	6.62E-04	1.38
	± 9.42E-05	± 1.04E-04	± 0.03	± 9.15E-05	± 1.23E-04	± 0.02	± 9.98E-05	± 9.59E-05	± 0.16
CQSO	2.47E-04	3.78E-04	1.70	4.18E-04	6.71E-04	1.63	1.83E-04	2.77E-04	2.36
	± 4.13E-05	± 4.49E-05	± 0.16	± 5.20E-05	± 7.54E-05	± 0.03	± 5.98E-05	± 5.92E-05	± 0.52
	B1 *(f:100, s:20)*			**B2** *(f:100, s:40)*			**B3** *(f:100, s:58)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.91E-04	3.31E-04	1.76	3.63E-04	5.08E-04	1.43	4.26E-04	5.53E-04	1.32
	1.47E-05	1.80E-05	0.07	3.41E-05	3.44E-05	0.06	4.09E-05	4.51E-05	0.04
PSO	6.06E-04	7.68E-04	1.29	5.61E-04	8.53E-04	1.53	4.63E-04	6.00E-04	1.66
	7.21E-05	8.13E-05	0.03	5.20E-05	7.18E-05	0.02	9.37E-05	9.00E-05	0.31
CQSO	8.82E-05	2.21E-04	2.66	2.34E-04	4.17E-04	1.94	6.91E-05	1.75E-04	3.67
	1.09E-05	1.31E-05	0.18	4.20E-05	5.89E-05	0.13	1.75E-05	8.25E-06	0.66
	C1 *(f:150, s:20)*			**C2** *(f:150, s:40)*			**C3** *(f:150, s:58)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.32E-04	2.48E-04	1.95	2.49E-04	3.76E-04	1.54	3.25E-04	4.52E-04	1.42
	1.25E-05	1.20E-05	0.12	2.29E-05	2.79E-05	0.06	3.66E-05	3.87E-05	0.04
PSO	4.54E-04	5.96E-04	1.34	4.74E-04	7.38E-04	1.56	3.35E-04	4.46E-04	1.78
	5.23E-05	5.96E-05	0.03	3.96E-05	5.61E-05	0.02	7.18E-05	6.62E-05	0.39
CQSO	5.60E-05	1.85E-04	3.33	1.08E-04	2.45E-04	2.39	4.20E-05	1.73E-04	4.78
	2.73E-06	5.57E-06	0.09	2.24E-05	3.01E-05	0.11	7.97E-06	5.96E-06	0.53

6.2.8 AWS Time Series

Table 6.25 show that, for scenarios A1, A2, and A3 (where $f = 50$), CQSO obtained the lowest training and generalization errors. The p-values in Table 6.26 indicate that CQSO produced significantly superior performance compared to the RPROP and PSO, except for scenario A1, where RPROP produced similar generalization performance. Comparing the performance of RPROP and PSO algorithms, RPROP outperformed PSO under the

Table 6.23: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0021	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0003	0.0001	0.0001	0.0001	0.0022	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.1646	0.0001	0.0001	0.2311
RROP vs CQSO	0.0001	0.0002	0.0001	0.0001	0.0584	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.7117	0.0001	0.0001	0.9764
RROP vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

scenario A1, which has the lowest spatial severity. For scenarios with higher spatial severity, A2 and A3, PSO produced superior results.

For scenarios B1, B2 and B3 where the temporal severity ($f = 100$) is lower compared to the A scenarios, the performance of RPROP and PSO improved while CQSO did not. This shows that RPROP and PSO requires more iterations before environmental change than the CQSO. Even though the performance of RPROP and PSO improved, CQSO still produced significantly lower errors for scenarios B2 and B3. For scenario B1, RPROP produced the lowest errors, but the difference in training performance with CQSO was insignificant.

For the C scenarios (where $f = 150$), the performance of RPROP and PSO improved compared to their performance on the B scenarios. RPROP produced significantly superior results for scenarios C1 and C3, while CQSO yielded the best performance for

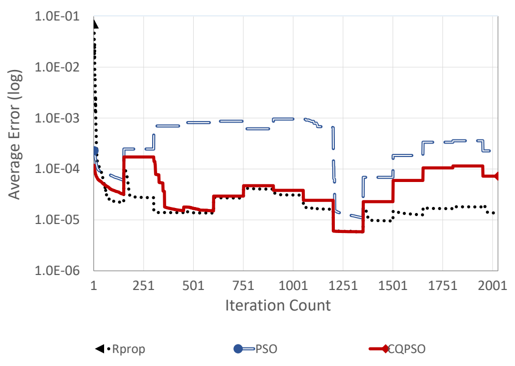
Table 6.24: S&P Time Series Algorithm Ranking for Scenarios A to C

Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	2	2	2	3	3	2.33	2.33
PSO	3	3	3	3	2	2	2.67	2.67
CQSO	1	1	1	1	1	1	1	1
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	2	2	1.5	2.5	2.5	2.17	2.83
PSO	3	3	3	3	2.5	2.5	2.83	2.83
CQSO	1	1	1	1.5	1	1	1	1.17
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	2	2	2	2.5	2.5	2.16	2.16
PSO	3	3	3	3	2.5	2.5	2.83	2.83
CQSO	1	1	1	1	1	1	1	1
	Average R(A)		Average R(B)		Average R(C)		**Overall Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2.33	2.33	2.17	2.83	2.16	2.16	2.22	2.44
PSO	2.67	2.67	2.83	2.83	2.83	2.83	2.78	2.78
CQSO	1	1	1	1.17	1	1	1	1.06

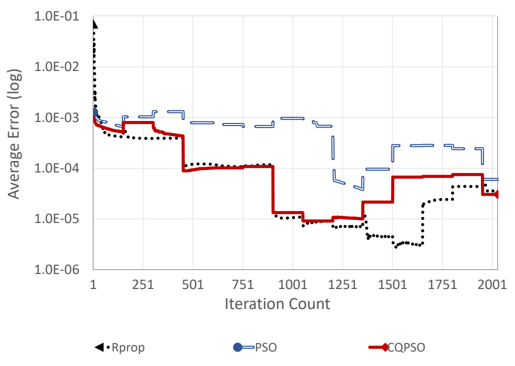
scenario C2.

The ρ values reported in Table 6.25 indicate that all the algorithm's overfitted. The ρ values obtained by the algorithms increased with a decrease in temporal severity (higher value of f). This implies that the overfitting behaviours exhibited by the algorithms were due to training for a long time on a dataset that has a small number of observations.

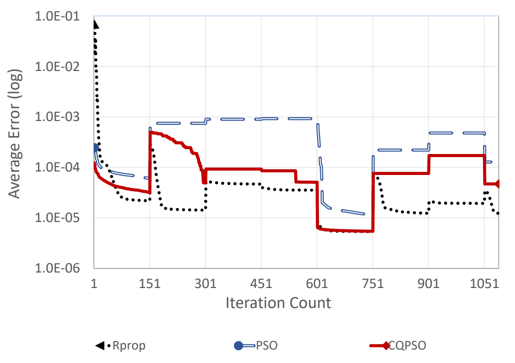
Table 6.27 indicated that the CQSO obtained the highest average rank for the A and B scenarios, while RPROP obtained the highest average rank for the C scenarios. Thus, CSQO achieved the overall best rank, the RPROP came second, and the PSO came last.



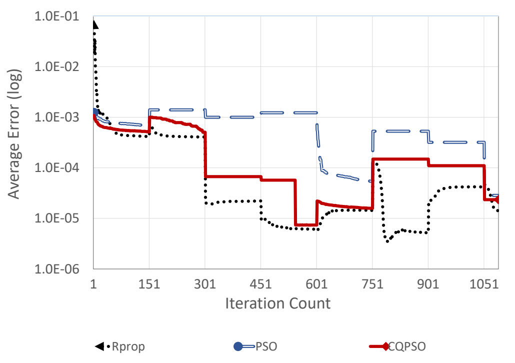
(a) T_E for C1



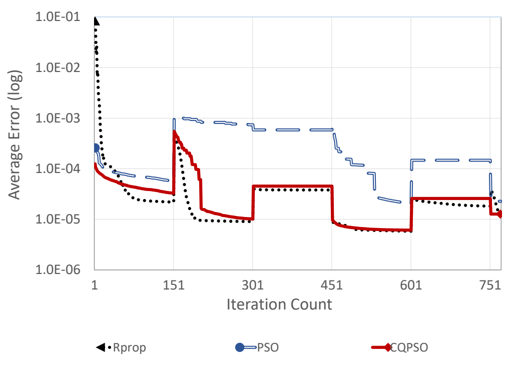
(b) G_E for C1



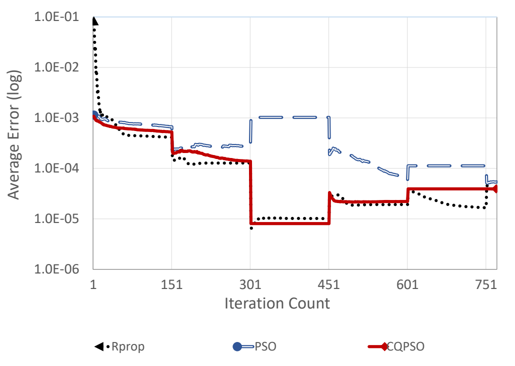
(c) T_E for C2



(d) G_E for C2



(e) T_E for C3



(f) G_E for C3

Figure 6.10: Training and generalization error results for S&P time series, scenarios C1 to C3

Figure 6.11 shows the performance progression of the algorithms over time in fore-

Table 6.25: AWS Time Series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:20)*			**A2** *(f:50, s:45)*			**A3** *(f:50, s:42)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	5.53E-04 ± 2.58E-05	6.77E-04 ± 2.62E-05	1.23 ± 0.02	8.46E-04 ± 6.54E-05	1.13E-03 ± 6.86E-05	1.35 ± 0.03	9.56E-04 ± 7.35E-05	1.31E-03 ± 7.49E-05	1.39 ± 0.03
PSO	6.34E-04 ± 2.35E-05	9.96E-04 ± 8.09E-05	1.56 ± 0.08	7.55E-04 ± 3.74E-05	7.76E-04 ± 5.27E-05	1.03 ± 0.05	8.04E-04 ± 4.53E-05	1.27E-03 ± 9.98E-05	1.57 ± 0.07
CQSO	3.92E-04 ± 2.29E-05	6.71E-04 ± 5.27E-05	1.70 ± 0.04	3.83E-04 ± 2.29E-05	5.98E-04 ± 5.79E-05	1.54 ± 0.07	4.12E-04 ± 3.21E-05	8.06E-04 ± 8.04E-05	1.94 ± 0.05
	B1 *(f:100, s:20)*			**B2** *(f:100, s:35)*			**B3** *(f:100, s:42)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	3.85E-04 ± 1.98E-05	5.44E-04 ± 1.95E-05	1.42 ± 0.03	5.00E-04 ± 3.37E-05	8.16E-04 ± 3.37E-05	1.65 ± 0.05	4.95E-04 ± 3.02E-05	8.67E-04 ± 3.18E-05	1.77 ± 0.04
PSO	5.45E-04 ± 1.49E-05	9.02E-04 ± 4.70E-05	1.65 ± 0.07	6.61E-04 ± 2.21E-05	6.78E-04 ± 3.24E-05	1.03 ± 0.04	6.99E-04 ± 3.57E-05	1.15E-03 ± 8.83E-05	1.64 ± 0.06
CQSO	4.07E-04 ± 1.88E-05	7.05E-04 ± 4.08E-05	1.73 ± 0.02	3.59E-04 ± 1.45E-05	6.06E-04 ± 3.42E-05	1.68 ± 0.04	4.10E-04 ± 2.22E-05	8.35E-04 ± 5.73E-05	2.02 ± 0.03
	C1 *(f:150, s:20)*			**C2** *(f:150, s:35)*			**C3** *(f:150, s:42)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	3.20E-04 ± 9.21E-06	4.87E-04 ± 9.35E-06	1.53 ± 0.02	3.89E-04 ± 1.89E-05	7.29E-04 ± 1.70E-05	1.89 ± 0.05	3.61E-04 ± 1.68E-05	7.48E-04 ± 1.84E-05	2.09 ± 0.05
PSO	5.20E-04 ± 2.41E-05	8.98E-04 ± 5.79E-05	1.72 ± 0.05	6.48E-04 ± 2.81E-05	6.75E-04 ± 4.24E-05	1.04 ± 0.04	6.38E-04 ± 3.17E-05	1.09E-03 ± 7.96E-05	1.70 ± 0.05
CQSO	3.84E-04 ± 1.35E-05	6.56E-04 ± 3.04E-05	1.70 ± 0.02	3.51E-04 ± 1.40E-05	6.10E-04 ± 2.89E-05	1.73 ± 0.02	4.11E-04 ± 1.43E-05	8.51E-04 ± 3.34E-05	2.07 ± 0.02

casting the AWS problem for scenarios A1, A2, and A3. As shown in the figure, all the algorithms exhibited a stable performance progression. The CQSO located a minimum faster (within the first five iterations) than the PSO and RPROP, which took about 50 algorithm iterations. Training errors generated by the algorithms peaked out at each environmental change. For all scenarios, RPROP recovered from the change before another one occurs, unlike the CQSO and the PSO. The figure clearly illustrate that the PSO

Table 6.26: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0459	0.0021	0.0001	0.0001	0.3516
RROP vs CQSO	0.0001	0.0001	0.0001	0.4333	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0399	0.0001	0.0001	0.0001
RROP vs CQSO	0.1103	0.0001	0.0001	0.0001	0.0001	0.0105
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0052	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0228	0.0001
RROP vs CQSO	0.0001	0.0030	0.0002	0.0001	0.0001	0.0001
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0276	0.0001

produced the worst training performance. The figure also show that the generalization performance of RPROP deteriorated and that of CQSO improved with an increase in spatial severity.

6.2.9 USD Time Series

Table 6.28 and Table 6.29 show that, for the scenarios A1, A2 and A3, the CQSO significantly outperformed the PSO and the RPROP in forecasting the USD time series, except for A1, where RPROP produced a similar T_E.

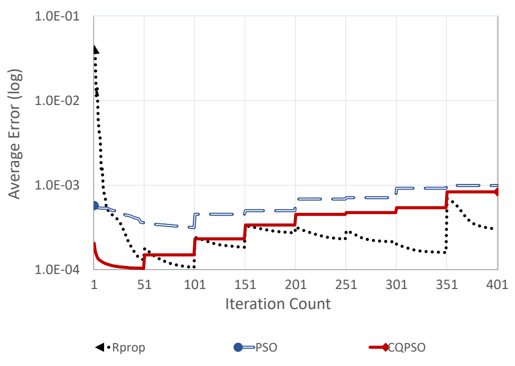
For scenarios B1, B2 and B3, the performance of all the training algorithms improved compared to their performance for scenarios A1, A2 and A3. This is attributed to the increase in the value of f from 50 to 100. Examining the performance of the algorithms

Table 6.27: AWS Algorithm Ranking for scenarios A to C

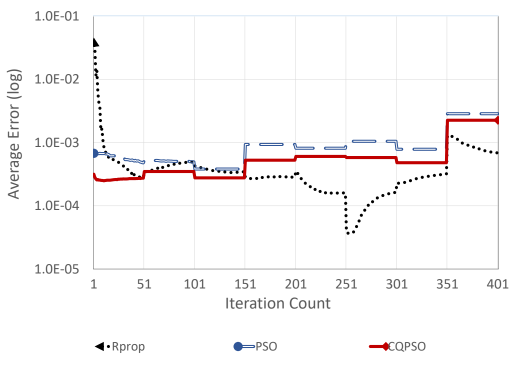
Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	1.5	3	3	3	2.5	2.67	2.33
PSO	3	3	2	2	2	2.5	2.33	2.5
CQSO	1	1.5	1	1	1	1	1	1.17
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1.5	1	2	3	2	2	1.83	2
PSO	3	3	3	2	3	3	3	2.67
CQSO	1.5	2	1	1	1	1	1.17	1.33
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	1	2	3	1	1	1.33	1.67
PSO	3	3	3	2	3	3	3	2.67
CQSO	2	2	1	1	2	2	1.67	1.67
	Average R(A)		Average R(B)		Average R(C)		**Overall Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2.67	2.33	2.83	2	1.33	1.67	2.28	2
PSO	2.33	2.5	3	2.67	3	2.67	2.78	2.61
CQSO	1	1.17	1.17	1.33	1.67	1.67	1.28	1.39

relative to each other revealed that, for scenarios B2 and B3, the CQSO produced the lowest errors, similar to the performance seen for A2 and A3. These errors are significantly different compared to the errors obtained by PSO. The errors produced by CQSO are, however, not significantly different compared to errors obtained by RPROP, except for G_E for B2. Even though RPROP had the best training error for scenario B1, the CQSO had a superior generalization error.

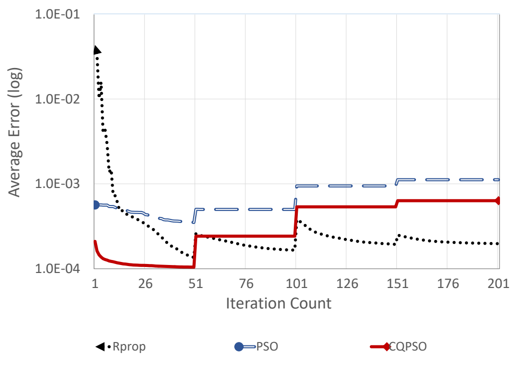
For scenarios C1, C2 and C3, the performance of all the training algorithms improved compared to their performance for scenarios B1, B2 and B3. This is also attributed



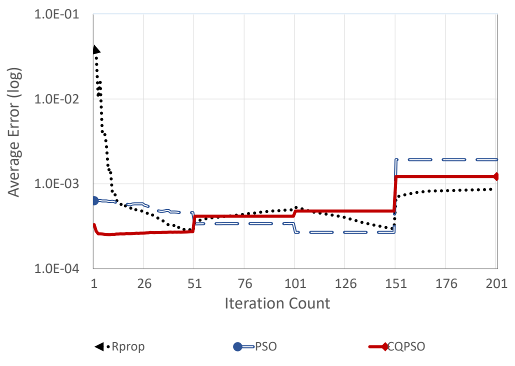
(a) T_E for A1



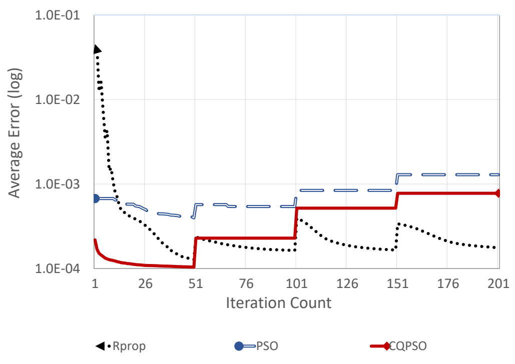
(b) G_E for A1



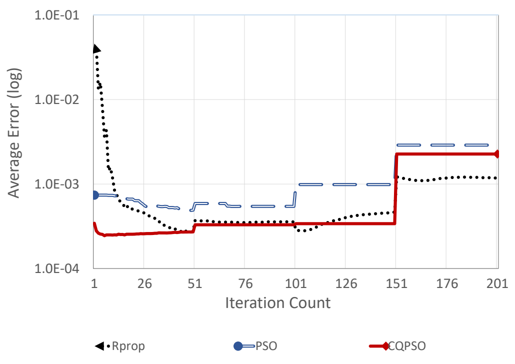
(c) T_E for A2



(d) G_E for A2



(e) T_E for A3



(f) G_E for A3

Figure 6.11: Training and generalization error Result for AWS Time Series, scenarios A1 to A3

to the increase in the value of f from 100 to 150. The improvement was, however, minor for CQSO. For scenario C1, while RPROP had the best training performance,

Table 6.28: USD Time Series Results

Algorithm	**Scenario**								
	A1 *(f:50, s:8)*			**A2** *(f:50, s:16)*			**A3** *(f:50, s:20)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	4.71E-04	1.20E-03	3.11	1.07E-03	1.99E-03	2.25	9.62E-04	1.65E-03	1.99
	± 1.23E-04	± 1.17E-04	± 0.34	± 2.54E-04	± 2.87E-04	± 0.26	± 2.63E-04	± 3.09E-04	± 0.19
PSO	6.49E-04	1.29E-03	2.00	6.98E-04	1.73E-03	2.50	8.74E-04	1.52E-03	1.75
	± 2.70E-05	± 6.60E-05	± 0.10	± 4.00E-05	± 8.12E-05	± 0.10	± 3.74E-05	± 8.11E-05	± 0.08
CQSO	4.90E-04	1.01E-03	2.11	4.12E-04	1.16E-03	2.93	5.66E-04	1.10E-03	1.95
	± 2.37E-05	± 5.77E-05	± 0.16	± 3.64E-05	± 4.50E-05	± 0.21	± 3.08E-05	± 4.18E-05	± 0.07
	B1 *(f:100, s:8)*			**B2** *(f:100, s:16)*			**B3** *(f:100, s:20)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	3.18E-04	1.19E-03	4.25	6.21E-04	1.02E-03	1.94	5.64E-04	1.19E-03	2.69
	± 5.05E-05	± 5.67E-05	± 0.46	± 1.70E-04	± 1.51E-04	± 0.17	± 1.43E-04	± 1.37E-04	± 0.34
PSO	6.02E-04	1.01E-03	1.71	6.29E-04	1.04E-03	1.66	6.42E-04	1.26E-03	1.97
	± 4.50E-05	± 7.57E-05	± 0.11	± 2.65E-05	± 5.82E-05	± 0.05	± 2.96E-05	± 5.19E-05	± 0.07
CQSO	4.61E-04	9.05E-04	1.98	3.96E-04	6.28E-04	1.59	4.97E-04	1.05E-03	2.13
	± 1.92E-05	± 5.39E-05	± 0.12	± 1.77E-05	± 2.46E-05	± 0.04	± 2.47E-05	± 2.81E-05	± 0.09
	C1 *(f:150, s:8)*			**C2** *(f:150, s:16)*			**C3** *(f:150, s:20)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.60E-04	1.21E-03	5.39	4.60E-04	9.25E-04	2.22	3.44E-04	9.78E-04	3.16
	± 4.25E-05	± 4.03E-05	± 0.62	± 8.59E-05	± 9.42E-05	± 0.17	± 5.49E-05	± 5.04E-05	± 0.30
PSO	5.42E-04	9.39E-04	1.74	5.45E-04	8.79E-04	1.61	5.44E-04	1.18E-03	2.19
	± 2.02E-05	± 3.72E-05	± 0.08	± 2.34E-05	± 4.22E-05	± 0.03	± 2.15E-05	± 3.86E-05	± 0.07
CQSO	4.42E-04	9.05E-04	2.05	3.84E-04	6.15E-04	1.61	4.53E-04	1.02E-03	2.28
	± 1.27E-05	± 5.31E-05	± 0.11	± 1.80E-05	± 2.40E-05	± 0.03	± 2.22E-05	± 2.39E-05	± 0.08

it produced the worst generalization errors. This clearly shows that an increase in the value of f caused the algorithm to overfit. However, PSO benefited, as it produced similar generalization performance with CQSO (which produced the best generalization performance in A1 and B1). For scenario C2, CQSO produced the lowest training and generalization errors as it did for A2 and B2. For scenario C3, RPROP had the lowest errors.

Table 6.29: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.2939	0.1039	0.0081	0.8476	0.1171
RROP vs CQSO	0.1515	0.0001	0.0052	0.0228	0.0001	0.0005
PSO vs CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0024	0.0256	0.0001	0.0120	0.0134
RROP vs CQSO	0.0001	0.7227	0.1515	0.0001	0.0001	0.9058
PSO vs CQSO	0.0001	0.0001	0.0001	0.0139	0.0001	0.0001
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.0047	0.0001	0.0001	0.6361	0.0001
RROP vs CQSO	0.0001	0.9411	0.0002	0.0001	0.0001	0.0017
PSO vs CQSO	0.0001	0.0001	0.0001	0.1738	0.0001	0.0001

The ρ values reported in Table 6.28 indicate that all the algorithms overfitted, which is likely caused by using too many iterations on a small dataset.

As illustrated in Table 6.30, CSQO obtained the highest average training and generalization rank for the A and B scenarios. For the C scenarios, RPROP obtained the highest average rank in training, while CQSO had the highest average rank in generalization. Thus, CQSO achieved the overall best rank, the RPROP came second, and the PSO came last.

Figure 6.12 shows the performance progression of the algorithms over time for the scenarios B1, B2, and B3, in forecasting the USD problem. The figures clearly show that the training errors generated by the algorithms peaked out at each environmental change. RPROP, unlike CQSO and PSO, recovered from the change before another one occurred. The figure also illustrates that training performance of the algorithms

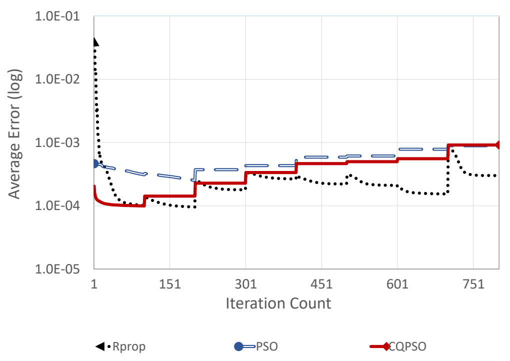
Table 6.30: USD Time Series Algorithm Ranking for Scenarios A to C

Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	2	2.5	2.5	2.5	2.5	2	2.33
PSO	3	3	2.5	2.5	2.5	2.5	2.67	2.67
CQSO	2	1	1	1	1	1	1.33	1
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	3	1.5	3	2.25	2.25	1.58	2.75
PSO	3	2	3	2	2.25	2.25	2.75	2.08
CQSO	2	1	1.5	1	1.5	1.5	1.67	1.16
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	3	1.5	2.5	1	1	1.17	2.17
PSO	3	1.5	3	2.5	3	3	3	2.33
CQSO	2	1.5	1.5	1	2	2	1.83	1.5
	Average R(A)		Average R(B)		Average R(C)		**Overall Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2	2.33	1.5	2.17	1.58	2.75	1.67	2.42
PSO	2.67	2.67	3	2.67	2.75	2.08	2.81	2.47
CQSO	1.33	1	1.5	1.17	1.67	1.16	1.5	1.11

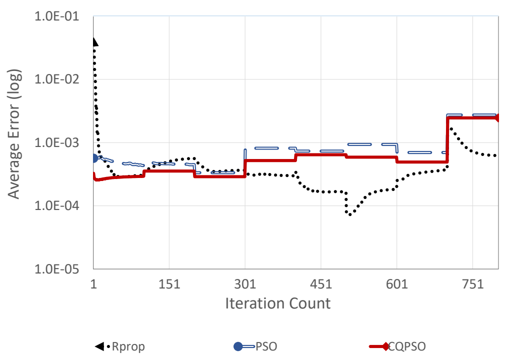
improved with increase in spatial severity. It is also clearly shown in the figure that the algorithms produced higher generalization errors, which indicates overfitting behaviour.

6.2.10 LM Time Series

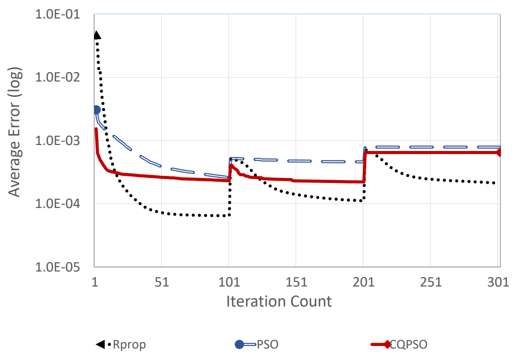
Table 6.31 shows that RPROP produced the lowest training error for scenario A1, while PSO produced the worst error. However, the p-values in Table 6.32 indicate that the difference in performance between RPROP and CQSO was insignificant. In terms of generalization, RPROP produced significantly the highest error, while PSO and CQSO



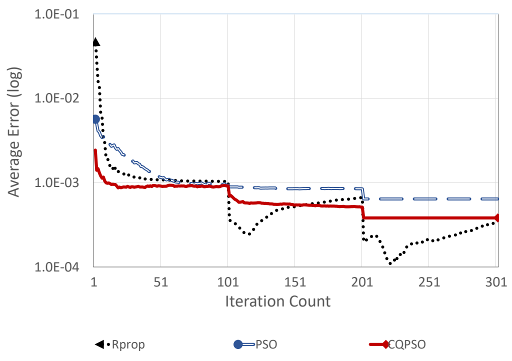
(a) T_E for B1



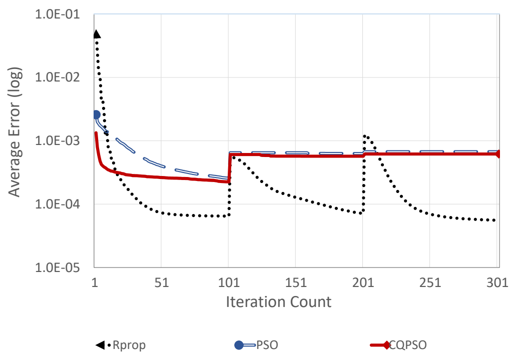
(b) G_E for B1



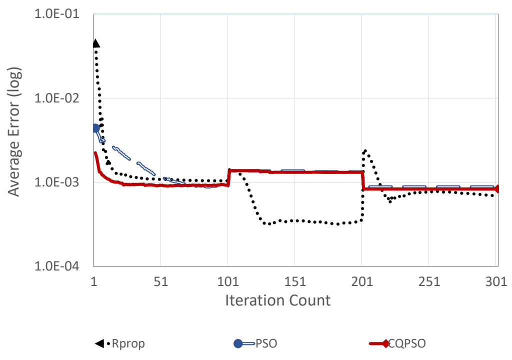
(c) T_E for B2



(d) G_E for B2



(e) T_E for B3



(f) G_E for B3

Figure 6.12: Training and generalization error results for USD time series, scenarios B1 to B3

produced lower errors that are not significantly different. For scenario A2, CQSO yielded the lowest training and generalization erros while RPROP yielded the highest errors. The

Table 6.31: LM Time Series Results

Algorithm	Scenario								
	A1 *(f:50, s:10)*			**A2** *(f:50, s:25)*			**A3** *(f:50, s:31)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.21E-03 ±4.27E-05	3.81E-03 ±4.74E-05	1.72 ±0.02	2.94E-03 ±9.41E-05	3.12E-03 ±9.61E-05	1.06 ±0.01	3.13E-03 ±1.24E-04	3.00E-03 ±1.21E-04	0.96 ± 0.01
PSO	2.54E-03 ± 1.32E-04	3.58E-03 ± 7.46E-05	1.43 ± 0.07	2.40E-03 ± 1.09E-05	2.57E-03 ± 1.61E-05	1.07 ± 0.01	2.39E-03 ± 1.25E-05	2.29E-03 ± 3.36E-05	0.96 ± 0.01
CQSO	2.29E-03 ± 7.76E-05	3.67E-03 ± 7.04E-05	1.61 ± 0.06	2.38E-03 ± 1.70E-06	2.56E-03 ± 4.53E-06	1.07 ± 0.00	2.37E-03 ± 1.38E-05	2.26E-03 ± 3.65E-05	0.95 ± 0.02
	B1 *(f:100, s:10)*			**B2** *(f:100, s:25)*			**B3** *(f:100, s:31)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	2.06E-03 ± 2.06E-05	3.72E-03 ± 2.06E-05	1.81 ± 0.01	2.63E-03 ± 2.70E-05	2.92E-03 ± 2.81E-05	1.11 ± 0.00	2.73E-03 ± 5.68E-05	2.68E-03 ± 5.65E-05	0.98 ± 0.00
PSO	2.75E-03 ± 8.53E-05	3.58E-03 ± 2.67E-05	1.31 ± 0.04	2.40E-03 ± 8.90E-06	2.56E-03 ± 1.99E-05	1.07 ± 0.01	2.39E-03 ± 1.30E-05	2.23E-03 ± 1.89E-05	0.93 ± 0.01
CQSO	2.39E-03 ± 8.84E-05	3.78E-03 ± 4.56E-05	1.60 ± 0.06	2.38E-03 ± 7.10E-07	2.55E-03 ± 3.54E-06	1.07 ± 0.00	2.41E-03 ± 1.14E-05	2.26E-03 ± 1.77E-05	0.94 ± 0.01
	C1 *(f:150, s:10)*			**C2** *(f:150, s:25)*			**C3** *(f:150, s:31)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
RPROP	1.97E-03 ± 1.16E-05	3.70E-03 ± 1.15E-05	1.87 ± 0.01	2.58E-03 ± 3.86E-05	2.98E-03 ± 3.98E-05	1.16 ± 0.00	1.00E+00 ± 3.18E-05	2.56E-03 ± 3.03E-05	1.00 ± 0.00
PSO	2.99E-03 ± 1.18E-04	3.68E-03 ± 2.36E-05	1.25 ± 0.06	2.39E-03 ± 5.24E-06	2.53E-03 ± 1.11E-05	1.06 ± 0.00	2.41E-03 ± 7.96E-06	2.27E-03 ± 1.46E-05	0.94 ± 0.01
CQSO	2.58E-03 ± 1.34E-04	3.74E-03 ± 4.64E-05	1.48 ± 0.07	2.38E-03 ± 5.57E-06	2.54E-03 ± 1.15E-05	1.07 ± 0.01	2.42E-03 ± 7.85E-06	2.26E-03 ± 1.67E-05	0.93 ± 0.01

p-values in Table 6.32 indicate that CQSO and PSO produced similar generalization errors. For scenario A3, RPROP produced the worst training and generalization errors, while CQSO produced lower errors that do not differ significantly from that of PSO. The generalization error produced by the three algorithms, i.e. CQSO, PSO and RPROP, improved as spatial severity increased for the A scenarios.

Table 6.31 shows that, for scenarios B1, B2, and B3 (where $f = 100$), the performance

Table 6.32: Mann-Whitney U p-values obtained for the average training and generalization error comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Algorithm	**Training**			**Generalization**		
	A1	A2	A3	A1	A2	A3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.3440	0.0001	0.0001	0.0010	0.0001	0.0001
PSO vs CQSO	0.0018	0.0001	0.1474	0.0546	0.0604	0.1691
	B1	B2	B3	B1	B2	B3
RROP vs PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.0068	0.0001	0.0001
PSO vs CQSO	0.0001	0.0002	0.1039	0.0001	0.1474	0.0493
	C1	C2	C3	C1	C2	C3
RROP vs PSO	0.0001	0.001	0.0001	0.3440	0.001	0.0001
RROP vs CQSO	0.0001	0.0001	0.0001	0.3077	0.0001	0.0001
PSO vs CQSO	0.0001	0.0371	0.1515	0.0863	0.2200	0.2675

of CQSO and PSO deteriorated compared to their performance for the corresponding scenarios A1, A2 and A3 (where $f = 50$). Even though the performance of RPROP improved, it still produced the worst training and generalization errors for scenarios B2 and B3. For scenario B1, RPROP produced the best training error and PSO the worst. In terms of generalization, PSO produced the lowest error and CQSO the highest. For scenarios B1 and B2, both PSO and CQSO yielded similar training and generalization errors, except for B1, where CQSO produced a significantly lower training error.

For scenario C1, while RPROP yielded the lowest and PSO the highest training errors, all the three algorithms produced similar generalization performance. For scenarios C2 and C3, RPROP produced the worst errors, while CQSO and PSO showed similar performance.

The ρ values in Table 6.31 showed that for scenarios A1, B1 and C1, all the train-

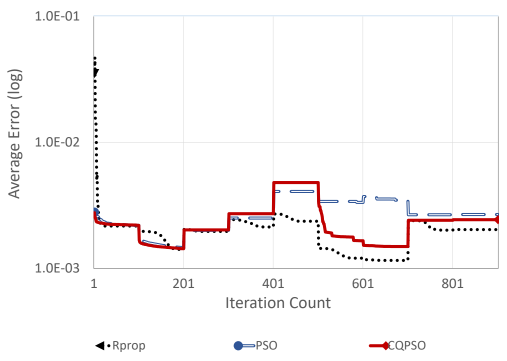
Table 6.33: LM Time Series Algorithm Ranking for Scenarios A to C

Algorithm	A1		A2		A3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1.5	3	3	3	3	3	2.5	3
PSO	3	1.5	2	1.5	1.5	1.5	2.17	1.5
CQSO	1.5	1.5	1	1.5	1.5	1.5	1.33	1.5
	B1		B2		B3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	2	3	3	3	3	2.33	2.67
PSO	3	1	2	1.5	1.5	1.5	2.17	1.33
CQSO	2	3	1	1.5	1.5	1.5	1.5	2
	C1		C2		C3		**Average Rank**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	1	2	3	3	3	3	2.33	2.67
PSO	3	2	1.5	1.5	1.5	1.5	2	1.67
CQSO	2	2	1.5	1.5	1.5	1.5	1.67	1.67
	Average R(A)		Average R(B)		Average R(C)		**Overall Ranking**	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
RPROP	2.5	3	2.33	2.67	2	2.63	2.27	2.77
PSO	2.17	1.5	2.17	1.33	2.75	2.25	2.36	1.69
CQSO	1.33	1.5	1.5	2	1.25	1.38	1.36	1.62

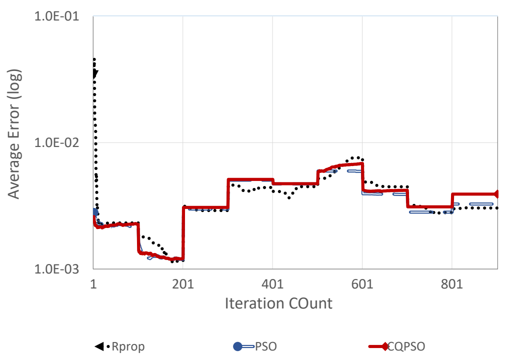
ing algorithms overfitted. For the more spatially severe scenarios A2, B2 and C2, the algorithms only showed minor signs of overfitting. For scenarios A3, B3 and C3, where the whole window is discarded during change, none of the training algorithms showed any sign of overfitting. This shows good adaption by the algorithms as spacial severity increased.

Table 6.33 show that CQSO achieved the overall best rank.

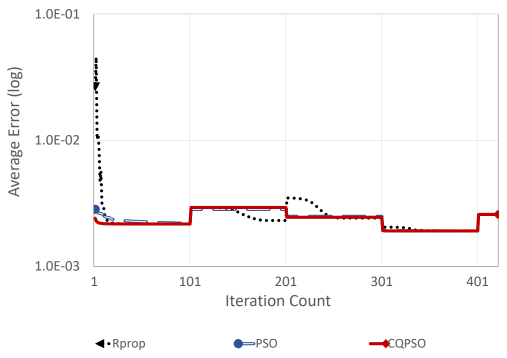
Figure 6.13 shows that all the algorithms exhibited similar performance progression throughout the experiment runs, except during the initial few iterations.



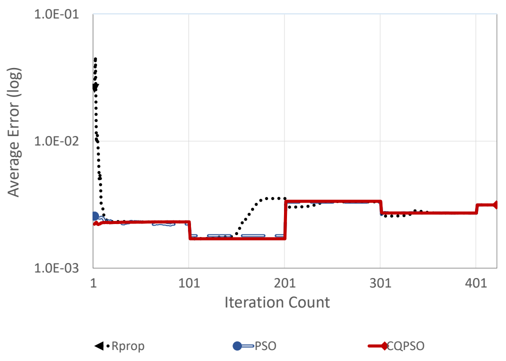
(a) T_E for B1



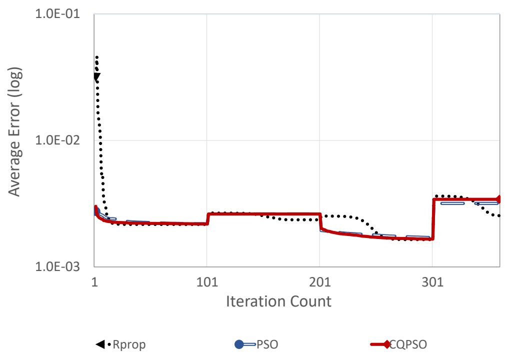
(b) G_E for B1



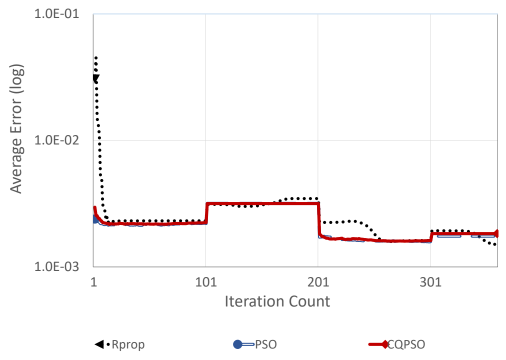
(c) T_E for B2



(d) G_E for B2



(e) T_E for B3



(f) G_E for B3

Figure 6.13: Training and generalization error Results for LM Time Series, Scenarios B1 to B3

Table 6.34: Average Algorithm Ranking Overall Time Series

Problem	RPROP		PSO		CQSO	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
SAM	2.38	2.67	2.61	2.33	**1**	**1**
HIT	3	3	2	2	**1**	**1**
DMT	3	3	2	2	**1**	**1**
MG	3	3	2	2	**1**	**1**
Lorenz	3	3	2	2	**1**	**1**
IAP	1.67	1.72	3	3	**1.58**	**1.28**
S&P	2.22	2.44	2.78	2.78	**1**	**1.06**
AWS	2.28	**2**	2.78	2.61	**1.28**	2.39
USD	1.67	2.42	2.81	2.47	**1.5**	**1.11**
LM	2.27	2.77	2.36	1.69	**1.36**	**1.62**
Overall average rank	2.45	2.60	2.43	2.29	1.17	1.25

6.3 Summary

The aim of this chapter was to investigate the applicability and efficiency of a dynamic PSO algorithm in training FNN forecasters under non-static environments, and to compare the performance against standard PSO and back-propagation (RPROP). Experiments were conducted by training a FNN using the algorithms investigated to forecast ten problems under nine different dynamic scenarios.

The chapter provided a detailed description of the experimental procedure followed, which includes descriptions of the datasets used and the data pre-processing employed, a discussion on simulating dynamic environments, the parameter optimization process employed, and the performance measures used. The rest of the chapter was dedicated to the experimental results obtained.

Table 6.34 summarizes the overall average ranks obtained by the algorithms for the ten different forecasting problems considered. The highest rank for each problem is given in bold. The overall average algorithm ranks for the ten problems are also listed in Table 6.34. Table 6.34 shows that the dynamic PSO (CQSO) outperformed both the standard PSO and the RPROP algorithms in terms of T_E and G_E. Thus, CQSO was shown to be very efficient in training FNN forecasters, and is a viable alternative to both the standard PSO and RPROP.

It was observed throughout the empirical analysis carried out in this chapter that the CQSO and the PSO were more successful under the severely and abruptly changing scenarios, while RPROP was more successful under the gradually changing scenarios. The CQSO also converged faster than PSO and RPROP. RPROP was, however, more sensitive to stale data.

The CQSO involved more parameter optimization than both PSO and RPROP, since, in addition to the standard PSO parameters, other parameters specific to the CQSO also require optimization. Hence, the CQSO required more fine-tuning than PSO and RPROP, but have a higher potential to outperform the other two algorithms.

RPROP overfitted whenever the temporal severity allowed RPROP to train for too long. The CQSO, however, exhibited minor or no overfitting for most scenarios considered. A multiple number of factors may be responsible for such behaviour by CQSO, which remains a topic for future research.

The next chapter investigates if recurrent connections in NN forecasters provide any benefit if the NN is trained using a dynamic PSO algorithm.

Chapter 7

Recurrent Connections: Are They Necessary?

"A central lesson of science is that to understand complex issues (or even simple ones), we must try to free our minds of dogma and to guarantee the freedom to publish, to contradict, and to experiment. Arguments from authority are unacceptable. "

– Carl Sagan -1998

Chapter 6 evaluated the applicability and performance of a dynamic PSO algorithm as a training algorithm for FNN forecasters under non-stationary environments. This chapter tests the hypothesis that recurrent/delayed connections are not necessary in a NN trained for non-stationary time series forecasting if a dynamic PSO algorithm is used as the training algorithm.

The experimental methodology used to test the hypothesis is given in Section 7.1. The results and discussion of the findings are given in Section 7.2. Section 7.3 summarises the chapter.

7.1 Methodology

This chapter investigates the necessity of recurrent/delayed connections in NN forecasters if a dynamic PSO algorithm is used as the training method. For this purpose, a set of experiments under the nine different dynamic environmental scenarios described in Section 6.1.3 were carried out on the same ten forecasting problems. Each experiment involves training a FNN using a dynamic PSO algorithm, and comparing the result to that obtained from four different types of RNNs (i.e. Elman NN, Jordan NN, Multi-Recurrent NN and Time Delay NN), each trained separately using RPROP, standard PSO and the dynamic PSO algorithm. For effective performance evaluation, 30 independent runs for each experiment were carried out and the average with confidence interval over these 30 runs was computed. The performance metrics described in Section 6.1.5 were used in all the experiments. All algorithms used were implemented in the Computational Intelligence library (CIlib) version 0.9 [102].

The remainder of this section discusses the problems used in the experiments and the process used to assign values to the control parameters of the algorithms used.

7.1.1 Datasets

In keeping with the work of the previous chapter, the datasets described in Section 6.1.1 were used for the experimental work in this chapter. Similarly, the datasets were preprocessed using the methods described in Section 6.1.2.

7.1.2 Parameter Selection

All the relevant algorithm parameters were optimised and setup as follows:

NN Configuration

For each problem, the number of input, hidden, and output layer nodes of the NNs were selected as described in Section 6.1.4. The number of time steps (or delayed patterns) in the TDNNs were also iteratively optimized in a similar way that the optimal number of hidden nodes were selected. The parameters selected for the NNs per problem are listed in Table 7.1.

Linear activation functions were used in the hidden units of the FNNs due to saturation issues, as discussed in Section 4.3. However, modified hyperbolic tangent functions suggested in [85], were used in the hidden layer nodes of the RNNs and in the output

Table 7.1: Neural network parameters for each dataset

Problem	Input nodes	Delays for TDNN	Hidden nodes					Output nodes
			FNN	Elman NN	Jordan NN	MRNN	TDNN	
MG	4	1	11	3	6	2	4	1
Lorenz	5	6	6	2	3	2	2	1
LM	3	4	6	13	6	10	4	1
S&P	4	2	14	3	8	3	4	1
HIT	12	2	3	3	8	3	2	1
AWS	12	2	13	4	6	3	2	1
USD	12	1	2	2	7	2	2	1
SAM	10	1	4	3	11	5	2	1
HIT	24	1	2	3	3	2	2	1
DMT	30	6	3	3	3	3	3	1

nodes of all the NN models. The modified hyperbolic tangent function has a softer slope and wider activation range compared to the sigmoid function. Bounded activation functions were used in the RNNs in order to avoid passing blown-up activations (i.e. large outputs) from the unbounded functions to the context/state layer, since large context/state layer inputs may dominate the real NN inputs (which were normalized to be close to zero).

All NN weights were initialized randomly in the range $[-\frac{1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}]$, where *fanin* is the number incoming connections to a node.

Training algorithms setup

The control parameters of the training algorithms were assigned values as follows:

- Default parameters were used for the RPROP algorithm, since the algorithm does not require optimizing parameters to obtain optimal convergence times on most problems [116].

- The PSO setup described in section 6.1.4 was used.
- In addition to the parameter values used for the standard PSO, the optimal CQSO parameter values obtained in section 6.1.4 were used in setting up the CQSOs used for the purpose of this chapter.

7.2 Results

This section presents and discusses the results obtained from the experiments carried out in this chapter. The conclusions arrived at, based on the overall findings from the experiments, are also discussed.

For convenience, the naming convention Y-X was employed, where Y refers to either Elman, Jordan, MRNN, or TDNN, and X refers to either RPROP, PSO, or CQSO.

7.2.1 SAM Time Series

Scenarios A1 to A3: Table 7.2 summarizes the CMF T_E, G_E, and ρ values obtained by the forecasting models in predicting the SAM problem. Table 7.3 presents the performance ranking of the models based on their CMF T_E and G_E values.

The error values in Table 7.2 show that the FNN-CQSO outperformed the other models by yielding the lowest CMF T_E and G_E values, except for scenario A3, where the Jordan-CQSO yielded a slightly lower generalization error.

All the p-values for the pairwise comparisons between the FNN-CQSO and the other models are less than the 0.0001 threshold, except for the FNN-CQSO vs Jordan-CQSO comparison, where the models produced statistically similar G_E values for the three scenarios, and similar T_E values for scenario A1.

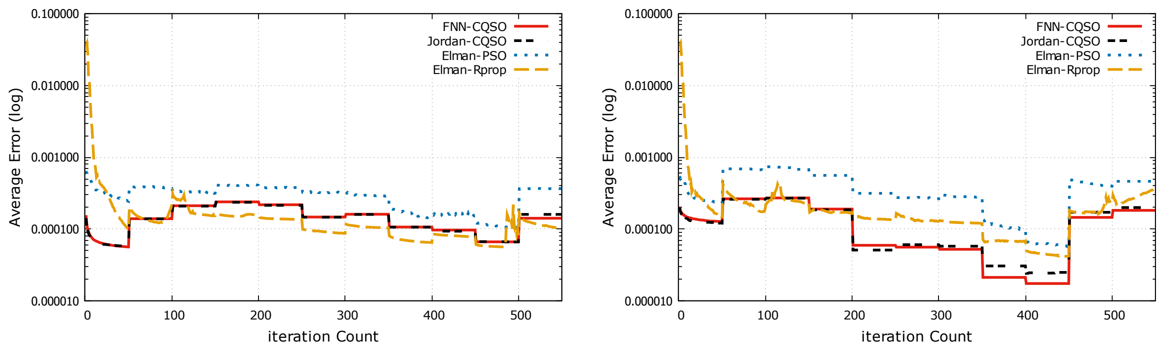
The ρ values obtained by the models illustrate that all the models had good generalization behaviour, except for scenario A1, where the RPROP and the standard PSO trained models overfitted. It was observed that the generalisation behaviour of all the models improved for scenario A2, compared to scenario A1.

Figure 7.1 illustrates the progression of T_E and G_E values over time for the FNN-CQSO and the three top performing models, one from each of the RPROP, PSO and

Table 7.2: Results of SAM time series, scenarios A1 to A3

Scenario / Model	**A1** *(f:50, s:20)*			**A2** *(f:50, s:40)*			**A3** *(f:50, s:60)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	1.45E-04 ± 2.95E-06	1.27E-04 ± 2.36E-06	0.88 ± 0.01	1.59E-04 ± 3.55E-06	8.41E-05 ± 1.57E-06	0.53 ± 0.01	1.39E-04 ± 3.30E-06	1.01E-04 ± 2.75E-06	0.73 ± 0
Elman-RPROP	3.77E-04 ± 7.47E-05	4.18E-04 ± 7.37E-05	1.14 ± 0.04	6.55E-04 ± 1.26E-04	6.45E-04 ± 1.30E-04	0.98 ± 0.03	9.37E-04 ± 2.10E-04	1.05E-03 ± 3.39E-04	1.08 ± 0.2
Elman-PSO	2.96E-04 ± 1.30E-05	3.81E-04 ± 2.78E-05	1.29 ± 0.04	4.39E-04 ± 2.44E-05	2.84E-04 ± 2.03E-05	0.65 ± 0.03	3.89E-04 ± 1.85E-05	3.58E-04 ± 1.99E-05	0.92 ± 0.2
Elman-CQPSO	2.41E-04 ± 1.27E-05	2.49E-04 ± 3.17E-05	1.02 ± 0.1	2.72E-04 ± 1.35E-05	1.31E-04 ± 1.30E-05	0.48 ± 0.03	2.32E-04 ± 1.42E-05	1.90E-04 ± 1.22E-05	0.83 ± 0.04
Jordan-RPROP	1.89E-02 ± 3.41E-02	1.88E-02 ± 3.36E-02	1.12 ± 0.05	4.28E-02 ± 7.92E-02	4.21E-02 ± 7.73E-02	1.04 ± 0.03	2.36E-03 ± 2.24E-03	2.54E-03 ± 2.47E-03	1.01 ± 0.03
Jordan-PSO	4.90E-04 ± 3.79E-05	6.40E-04 ± 8.48E-05	1.31 ± 0.11	7.56E-04 ± 7.89E-05	5.80E-04 ± 9.11E-05	0.74 ± 0.04	5.80E-04 ± 7.70E-05	5.79E-04 ± 8.01E-05	1 ± 0.04
Jordan-CQPSO	1.45E-04 ± 5.40E-06	1.31E-04 ± 5.19E-06	0.91 ± 0.04	1.64E-04 ± 4.87E-06	8.16E-05 ± 2.04E-06	0.5 ± 0.01	1.37E-04 ± 5.45E-06	1.02E-04 ± 4.88E-06	0.74 ± 0.01
MRNN-RPROP	5.36E-02 ± 8.63E-02	5.60E-02 ± 8.94E-02	1.11 ± 0.08	1.40E-02 ± 8.37E-03	1.44E-02 ± 8.70E-03	1.03 ± 0.02	3.95E-03 ± 2.07E-03	3.92E-03 ± 2.03E-03	0.97 ± 0.03
MRNN-PSO	3.91E-04 ± 2.32E-05	4.97E-04 ± 4.12E-05	1.28 ± 0.09	6.02E-04 ± 3.46E-05	4.03E-04 ± 2.98E-05	0.67 ± 0.03	5.06E-04 ± 4.09E-05	4.88E-04 ± 4.02E-05	0.97 ± 0.05
MRNN-CQPSO	2.20E-04 ± 1.00E-05	1.89E-04 ± 1.88E-05	0.86 ± 0.06	2.45E-04 ± 1.79E-05	1.10E-04 ± 8.22E-06	0.45 ± 0.02	2.04E-04 ± 7.26E-06	1.64E-04 ± 8.52E-06	0.8 ± 0.03
TDNN-RPROP	4.03E-04 ± 7.96E-05	4.36E-04 ± 7.95E-05	1.11 ± 0.03	6.58E-04 ± 1.54E-04	6.69E-04 ± 1.44E-04	1.04 ± 0.02	1.06E-03 ± 2.05E-04	1.00E-03 ± 1.95E-04	0.95 ± 0.02
TDNN-PSO	3.16E-04 ± 1.54E-05	3.89E-04 ± 2.47E-05	1.24 ± 0.07	4.22E-04 ± 2.09E-05	2.65E-04 ± 2.34E-05	0.62 ± 0.03	3.49E-04 ± 1.68E-05	3.20E-04 ± 2.07E-05	0.91 ± 0.03
TDNN-CQPSO	2.55E-04 ± 1.19E-05	2.31E-04 ± 2.08E-05	0.9 ± 0.05	2.93E-04 ± 1.45E-05	1.41E-04 ± 1.22E-05	0.48 ± 0.03	2.33E-04 ± 7.36E-06	1.96E-04 ± 1.29E-05	0.84 ± 0.04

CQSO trained models. As visualized in Figure 7.1, the two CQSO based models (i.e. FNN-CQSO and Jordan-CQSO) had the lowest initial T_E and G_E values, and took about 10 epochs to reach the lowest value, while the other two models, the Elman-RPROP and the Elman-PSO took about 50 epochs. The figure also shows that, after



(a) T_E (b) G_E

Figure 7.1: Training and generalization error progression in predicting SAM time series under scenario A1

the first environmental change, the Elman-PSO consistently produced the worst T_E and G_E values. The Elman-RPROP, however, had the best training performance after the initial 50 epochs, while the FNN-CQSO and the Jordan-CQSO provided the best generalization performance throughout the experiments.

Table 7.3: Models ranking in forecasting SAM time series, scenarios A1-A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	2	1	1	1	1.33
Elman-RPROP	8	8	9	10	10	11	9	9.67
Elman-PSO	6	6	7	7	7	7	6.67	6.67
Elman-CQPSO	4	5	4	4	4	4	4	4.33
Jordan-RPROP	12	12	13	13	12	12	12.33	12.33
Jordan-PSO	11	11	11	9	9	9	10.33	9.67
Jordan-CQPSO	2	2	2	1	2	2	2	1.67
MRNN-RPROP	13	13	12	12	13	13	12.67	12.67
MRNN-PSO	9	10	8	8	8	8	8.33	8.67
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	10	9	10	11	11	10	10.33	10
TDNN-PSO	7	7	6	6	6	6	6.33	6.33
TDNN-CQPSO	5	4	5	5	5	5	5	4.67

Table 7.3 shows that the FNN-CQSO model achieved the highest average training and generalization ranks over the three scenarios. The table also reveals that all the CQSO based models achieved higher ranks compared to the PSO and the RPROP based models. The RPROP based models achieved the lowest average ranks.

Scenarios B1 to B3: Table 7.4 shows that the FNN-CQSO model produced the lowest T_E and G_E values compared to all the other models, except for scenario B2, where the Jordan-CQSO yielded the lowest G_E value. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold, except for the FNN-CQSO vs Jordan-CQSO comparison. The FNN-CQSO and the Jordan-CQSO produced statistically similar T_E values for the three scenarios, and similar G_E values for scenario B3. It is observed that all the models yielded lower errors compared to their performance for the corresponding A scenarios (i.e where the models have the same spatial severity). The improvement in performance was due to an increase in the value of the change frequency f (i.e the number of iterations before environmental change).

The ρ values in Table 7.4 indicate that, for all three scenarios, none of the CQSO trained models overfitted. For scenario B1, both the PSO and the RPROP trained models overfitted. However, for scenarios B2 and B3, the PSO trained models did not overfit. The ρ obtained by the RPROP trained models showed slight overfitting for scenario B2, and no sign of overfitting for scenario B3. It is observed that the ρ values for the RPROP models decreased with increase in spatial severity.

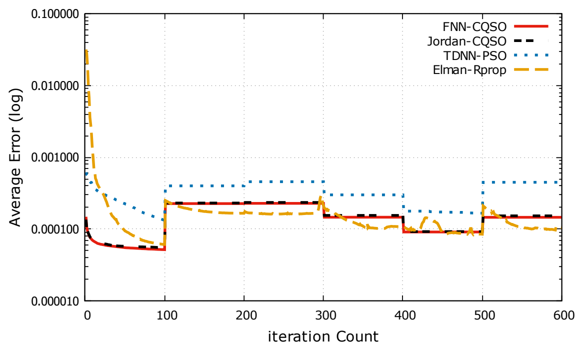
Figure 7.2 illustrates the performance progression over time for the four best performing models for scenario B2. As visualized in the figure, the FNN-CQSO and the Jordan-CQSO located the lowest value faster, in about 10 epochs during training, while the other models took about 100 epochs. The generalization performance over time for the two CQSO trained models also took about only 10 epochs to achieve the lowest value, while the other models took about 50 epochs. All four models kept track of the lowest value once located. During training, the TDNN-PSO had the worst performance throughout the model's run, and the CQSO based models had the best generalization performance. It is observed that, after the initial 100 epochs, all four models produced lower generalization errors, adapting well to the environmental changes. Thus, the progression of the errors show that the FNN-CQSO model had at least similar or better

Table 7.4: Results of SAM time series, scenario B1 to B3

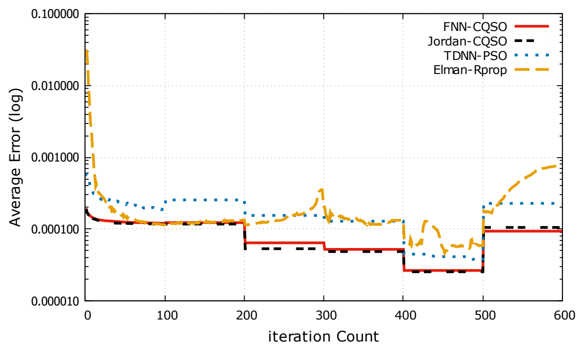
Scenario / Model	**B1** *(f:100, s:20)*			**B2** *(f:100, s:40)*			**B3** *(f:100, s:60)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	1.32E-04 ± 2.76E-06	1.19E-04 ± 2.18E-06	0.91 ± 0	1.49E-04 ± 2.99E-06	8.13E-05 ± 1.03E-06	0.55 ± 0.01	1.29E-04 ± 1.97E-06	9.36E-05 ± 1.56E-06	0.73 ± 0
Elman-RPROP	3.44E-04 ± 1.54E-04	4.11E-04 ± 1.69E-04	1.22 ± 0.04	3.14E-04 ± 4.94E-05	3.62E-04 ± 5.31E-05	1.17 ± 0.06	5.94E-04 ± 3.45E-04	5.97E-04 ± 3.52E-04	1.01 ± 0.02
Elman-PSO	2.68E-04 ± 1.43E-05	3.18E-04 ± 3.61E-05	1.17 ± 0.04	3.52E-04 ± 1.83E-05	1.88E-04 ± 1.69E-05	0.53 ± 0.06	3.03E-04 ± 1.66E-05	2.59E-04 ± 1.49E-05	0.86 ± 0.02
Elman-CQPSO	2.46E-04 ± 1.35E-05	2.35E-04 ± 2.37E-05	0.95 ± 0.08	2.63E-04 ± 1.29E-05	1.37E-04 ± 1.14E-05	0.53 ± 0.06	2.13E-04 ± 1.02E-05	1.93E-04 ± 8.75E-06	0.91 ± 0.04
Jordan-RPROP	2.78E-03 ± 2.97E-03	2.87E-03 ± 2.93E-03	1.19 ± 0.07	4.32E-03 ± 3.94E-03	4.52E-03 ± 3.99E-03	1.09 ± 0.03	1.14E-03 ± 4.90E-04	1.19E-03 ± 5.10E-04	1.04 ± 0.03
Jordan-PSO	5.02E-04 ± 5.58E-05	6.98E-04 ± 9.38E-05	1.39 ± 0.07	6.59E-04 ± 8.29E-05	4.78E-04 ± 7.48E-05	0.71 ± 0.05	5.17E-04 ± 5.75E-05	5.57E-04 ± 6.21E-05	1.08 ± 0.06
Jordan-CQPSO	1.34E-04 ± 3.58E-06	1.24E-04 ± 3.98E-06	0.93 ± 0.03	1.54E-04 ± 3.87E-06	7.91E-05 ± 2.59E-06	0.51 ± 0.01	1.30E-04 ± 4.50E-06	9.71E-05 ± 4.09E-06	0.75 ± 0.02
MRNN-RPROP	5.15E-03 ± 2.22E-03	5.75E-03 ± 2.62E-03	1.15 ± 0.07	7.45E-03 ± 4.11E-03	7.61E-03 ± 4.42E-03	1.02 ± 0.03	6.52E-03 ± 3.94E-03	6.48E-03 ± 4.03E-03	1.01 ± 0.03
MRNN-PSO	3.24E-04 ± 1.07E-05	4.39E-04 ± 2.75E-05	1.36 ± 0.08	5.03E-04 ± 3.84E-05	3.15E-04 ± 1.99E-05	0.64 ± 0.04	4.29E-04 ± 2.35E-05	4.11E-04 ± 3.39E-05	0.95 ± 0.05
MRNN-CQPSO	2.19E-04 ± 8.62E-06	1.88E-04 ± 1.30E-05	0.86 ± 0.04	2.38E-04 ± 1.28E-05	1.10E-04 ± 8.67E-06	0.46 ± 0.02	1.96E-04 ± 9.33E-06	1.61E-04 ± 1.17E-05	0.82 ± 0.04
TDNN-RPROP	2.31E-04 ± 3.21E-05	2.80E-04 ± 3.10E-05	1.24 ± 0.05	4.07E-04 ± 9.01E-05	4.67E-04 ± 8.75E-05	1.21 ± 0.05	5.58E-04 ± 1.28E-04	5.42E-04 ± 1.22E-04	0.99 ± 0.02
TDNN-PSO	2.74E-04 ± 1.01E-05	2.81E-04 ± 1.70E-05	1.03 ± 0.06	3.38E-04 ± 1.63E-05	1.76E-04 ± 1.62E-05	0.52 ± 0.03	2.80E-04 ± 1.13E-05	2.47E-04 ± 1.54E-05	0.88 ± 0.03
TDNN-CQPSO	2.35E-04 ± 6.15E-06	2.08E-04 ± 1.27E-05	0.88 ± 0.04	2.59E-04 ± 9.16E-06	1.25E-04 ± 7.63E-06	0.48 ± 0.02	2.25E-04 ± 7.43E-06	1.86E-04 ± 1.01E-05	0.83 ± 0.04

performance compared to the other models.

The performance ranking of the models given in Table 7.5 shows that the FNN-CQSO model obtained the highest average training and generalization ranks, and the remaining CQSO based models achieved higher average ranks compared to the PSO and



(a) T_E



(b) G_E

Figure 7.2: Training and generalization error results for SAM time series, scenario B2

the RPROP based models.

Scenarios C1 to C3: Table 7.6 shows that the FNN-CQSO model produced the lowest training and generalization errors compared to the remaining models, except for scenario C2, where the Jordan-CQSO produced a slightly lower generalization error. All the p-

Table 7.5: Models ranking in forecasting the SAM time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	2	1	1	1	1.33
Elman-RPROP	10	9	6	9	11	11	9	9.67
Elman-PSO	7	8	8	7	7	7	7.33	7.33
Elman-CQPSO	6	5	5	5	4	5	5	5
Jordan-RPROP	12	12	12	12	12	12	12	12
Jordan-PSO	11	11	11	11	9	10	10.33	10.67
Jordan-CQPSO	2	2	2	1	2	2	2	1.67
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	9	10	10	8	8	8	9	8.67
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	4	6	9	10	10	9	7.67	8.33
TDNN-PSO	8	7	7	6	6	6	7	6.33
TDNN-CQPSO	5	4	4	4	5	4	4.67	4

Table 7.6: Results of SAM time series, scenario C1 to C3

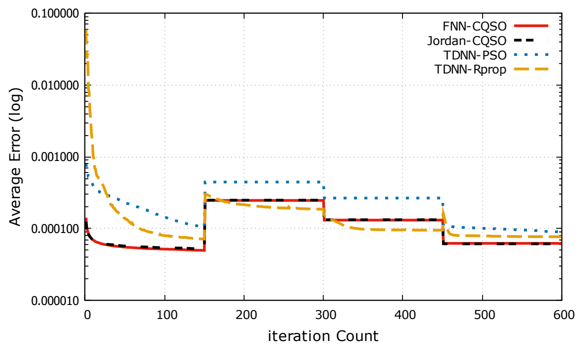
Scenario / Model	**C1** *(f:150, s:20)*			**C2** *(f:150, s:40)*			**C3** *(f:150, s:60)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	1.29E-04 ± 2.48E-06	1.18E-04 ± 1.93E-06	0.91 ± 0	1.44E-04 ± 2.01E-06	8.09E-05 ± 6.72E-07	0.56 ± 0.01	1.24E-04 ± 1.90E-06	8.93E-05 ± 1.57E-06	0.72 ± 0
Elman-RPROP	2.99E-04 ± 1.35E-04	3.77E-04 ± 1.58E-04	1.34 ± 0.07	4.16E-04 ± 1.36E-04	4.60E-04 ± 1.27E-04	1.19 ± 0.07	5.02E-04 ± 2.02E-04	5.13E-04 ± 2.01E-04	1.04 ± 0.02
Elman-PSO	2.45E-04 ± 1.22E-05	2.40E-04 ± 2.53E-05	0.97 ± 0.07	2.89E-04 ± 1.07E-05	1.38E-04 ± 9.19E-06	0.48 ± 0.07	2.71E-04 ± 2.31E-05	2.16E-04 ± 1.75E-05	0.8 ± 0.02
Elman-CQPSO	2.35E-04 ± 1.26E-05	2.13E-04 ± 1.91E-05	0.9 ± 0.05	2.51E-04 ± 1.04E-05	1.22E-04 ± 1.17E-05	0.49 ± 0.05	2.13E-04 ± 9.74E-06	2.04E-04 ± 9.73E-06	0.96 ± 0.03
Jordan-RPROP	1.84E-01 ± 2.01E-01	1.80E-01 ± 1.96E-01	1.14 ± 0.05	2.33E-03 ± 1.45E-03	2.42E-03 ± 1.48E-03	1.12 ± 0.05	2.89E-03 ± 1.45E-03	3.08E-03 ± 1.55E-03	1.09 ± 0.05
Jordan-PSO	4.35E-04 ± 4.02E-05	6.18E-04 ± 8.99E-05	1.39 ± 0.11	5.69E-04 ± 4.96E-05	4.26E-04 ± 5.54E-05	0.73 ± 0.04	6.53E-04 ± 1.47E-04	6.75E-04 ± 1.51E-04	1.06 ± 0.05
Jordan-CQPSO	1.33E-04 ± 4.41E-06	1.22E-04 ± 4.34E-06	0.92 ± 0.03	1.47E-04 ± 4.12E-06	7.59E-05 ± 2.11E-06	0.52 ± 0.01	1.25E-04 ± 3.56E-06	9.01E-05 ± 2.89E-06	0.72 ± 0
MRNN-RPROP	5.16E-03 ± 2.52E-03	5.35E-03 ± 2.62E-03	1.14 ± 0.07	5.25E-03 ± 2.02E-03	5.37E-03 ± 2.02E-03	1.08 ± 0.06	4.65E-03 ± 2.10E-03	5.30E-03 ± 2.42E-03	1.17 ± 0.09
MRNN-PSO	3.08E-04 ± 1.44E-05	4.08E-04 ± 2.61E-05	1.33 ± 0.07	3.99E-04 ± 2.39E-05	2.73E-04 ± 2.06E-05	0.68 ± 0.03	4.00E-04 ± 2.69E-05	3.61E-04 ± 2.21E-05	0.91 ± 0.05
MRNN-CQPSO	2.13E-04 ± 1.11E-05	1.80E-04 ± 1.77E-05	0.84 ± 0.07	2.27E-04 ± 1.05E-05	1.04E-04 ± 7.51E-06	0.46 ± 0.02	1.85E-04 ± 1.06E-05	1.52E-04 ± 8.78E-06	0.83 ± 0.04
TDNN-RPROP	2.28E-04 ± 3.01E-05	2.88E-04 ± 3.07E-05	1.3 ± 0.06	3.16E-04 ± 4.69E-05	4.13E-04 ± 4.14E-05	1.37 ± 0.08	3.65E-04 ± 8.38E-05	3.72E-04 ± 7.81E-05	1.05 ± 0.03
TDNN-PSO	2.49E-04 ± 7.21E-06	2.26E-04 ± 9.96E-06	0.91 ± 0.04	2.94E-04 ± 9.26E-06	1.41E-04 ± 7.39E-06	0.48 ± 0.02	2.55E-04 ± 9.63E-06	2.07E-04 ± 1.08E-05	0.81 ± 0.03
TDNN-CQPSO	2.35E-04 ± 9.37E-06	2.06E-04 ± 1.42E-05	0.87 ± 0.03	2.64E-04 ± 1.02E-05	1.20E-04 ± 6.27E-06	0.46 ± 0.01	2.14E-04 ± 5.85E-06	1.70E-04 ± 9.03E-06	0.8 ± 0.04

values for the pairwise comparisons between the FNN-CQSO and the other models are less than the 0.0001 threshold, except for the FNN-CQSO vs Jordan-CQSO comparison. The difference in G_E performance between the FNN-CQSO and the Jordan-CQSO was statistically significant for scenario C2 (where the Jordan-CQSO yielded a lower G_E

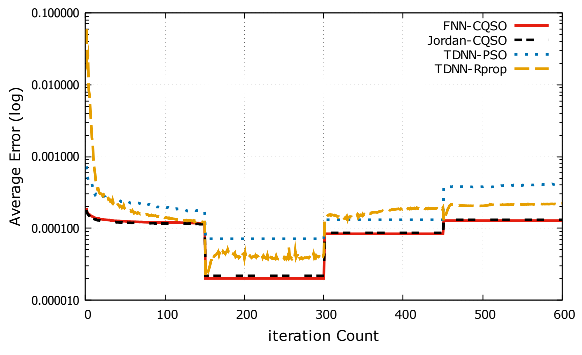
value). It is observed that the performance of the FNN-CQSO model improved with increase in spatial severity, from C1 to C3, while the reverse is the case for the RPROP trained models. This means that the RPROP based models did better for scenario C1, where spatial changes were gradual. For the PSO and the other CQSO based models, the performance deteriorated from scenario C1 to C2 due to an increase in the spatial severity. However, the performance improved for scenario C3 compared to C2. This shows that the number of iterations ($f = 150$) was enough for the training algorithms to learn the new patterns in the sliding window.

The ρ values listed in Table 7.6 show that none of the CQSO trained NNs overfitted. The NN models trained using RPROP, however, overfitted. It is observed that the ρ values obtained by the RPROP based models decreased as the change severity increased. Thus, for scenario C3, where the spatial severity is abrupt, the models' overfitting behaviour was only slight. All these indicate that the generalization behaviour of the CQSO and the RPROP based models were not affected by an increase in the value of the change frequency. The models trained using PSO also showed no or slight signs of overfitting for all the scenarios except for scenario C1, where the Jordan-PSO and the MRNN-PSO overfitted. This shows that an increase in the value of f improved the generalization behaviour of the Elman-PSO and the TDNN-PSO models.

Figure 7.3 illustrates the performance progression over time for the FNN-CQSO and three other top performing models, which includes the Jordan-CQSO, TDNN-RPROP and TDNN-PSO, for scenario C3. As shown in Figure 7.3, the TDNN-RPROP model



(a) T_E



(b) G_E

Figure 7.3: Training and generalization error results for SAM time series, scenario C3

had the worst initial errors. The two CQSO based models (i.e. the FNN-CQSO and the Jordan-CQSO) produced the lowest initial errors. The figure also illustrates that, during training, all four models had an initial increase in peak after the first environment change, but later adapted to the environmental changes. The progression of the G_E values shows that the two CQSO based models produced the best performance throughout the training process.

Table 7.7 shows that, for the three chaotic scenarios, the CQSO trained models achieved the highest average ranks compared to the PSO and the RPROP trained models. The RPROP trained models obtained the lowest average ranks.

The overall ranking of the models trained to predict the SAM time series for the nine scenarios is presented in Table 7.8. The table shows that the FNN-CQSO model emerged as the overall winner. The table also shows that all the CQSO based models achieved higher ranks compared to the PSO and the RPROP models. All these show that the CQSO training algorithm generally improved the performance of all the NNs in predicting the SAM problem under non-stationary environments.

Table 7.7: Models ranking in forecasting the SAM time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	2	1	1	1	1.33
Elman-RPROP	9	9	10	11	10	10	9.67	10
Elman-PSO	7	7	6	6	7	7	6.67	6.67
Elman-CQPSO	6	5	4	5	4	5	4.67	5
Jordan-RPROP	13	13	12	12	12	12	12.33	12.33
Jordan-PSO	11	11	11	10	11	11	11	10.67
Jordan-CQPSO	2	2	2	1	2	2	2	1.67
MRNN-RPROP	12	12	13	13	13	13	12.67	12.67
MRNN-PSO	10	10	9	8	9	8	9.33	8.67
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	4	8	8	9	8	9	6.67	8.67
TDNN-PSO	8	6	7	7	6	6	7	6.33
TDNN-CQPSO	5	4	5	4	5	4	5	4

Table 7.8: Overall models ranking in forecasting the SAM time series, scenarios A to C

Model	Scenario A		Scenario B		Scenario C		Overall Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1.5	1	1.33	1	1.33	1	1.39
Elman-RPROP	9	9.67	9	9.67	9.67	10	9.22	9.78
Elman-PSO	6.67	6.67	7.33	7.33	6.67	6.67	6.89	6.89
Elman-CQPSO	4	4.33	5	5	4.67	5	4.56	4.78
Jordan-RPROP	12.33	12.33	12	12	12.33	12.33	12.22	12.22
Jordan-PSO	10.33	9.67	10.33	10.67	11	10.67	10.55	10.34
Jordan-CQPSO	2	1.5	2	1.67	2	1.67	2	1.61
MRNN-RPROP	12.67	12.67	13	13	12.67	12.67	12.78	12.78
MRNN-PSO	8.33	8.67	9	8.67	9.33	8.67	8.89	8.67
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	10.33	10	7.67	8.33	6.67	8.67	8.22	9
TDNN-PSO	6.33	6.33	7	6.33	7	6.33	6.78	6.33
TDNN-CQPSO	5	4.67	4.67	4	5	4	4.89	4.22

7.2.2 HIT Time Series

Scenarios A1 to A3: Table 7.9 shows that the FNN-CQSO produced the lowest CMF T_E and G_E values for scenario A1. For scenarios A2 and A3, the Jordan-CQSO model yielded the lowest errors. The FNN-CQSO, however, outperformed the remaining models by producing lower errors. It is observed that the errors produced by all the models worsen as the spatial severity increased, similar to the observation made in Section 7.2.1.

All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were less than the 0.0001 threshold, except for the FNN-CQSO vs MRNN-CQSO comparison for scenarios A2 and A3, and for the FNN-CQSO vs TDNN-CQSO comparison for scenario A3.

All the models showed good generalisation behaviour, as indicated by the ρ values given in Table 7.9.

Figure 7.4 illustrates the progression of the T_E and G_E values over time for the FNN-CQSO and three other top performing models for scenario A2. Figure 7.4a shows

Table 7.9: Results of HIT time series, scenario A1 to A3

Scenario Model	**A1** *(f:50, s:250)*			**A2** *(f:50, s:500)*			**A3** *(f:50, s:528)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	5.55E-06 ± 3.69E-07	5.17E-06 ± 3.21E-07	0.93 ± 0.01	1.11E-05 ± 1.25E-06	1.10E-05 ± 9.58E-07	1.01 ± 0.02	1.35E-05 ± 1.20E-06	1.24E-05 ± 1.04E-06	0.93 ± 0.02
Elman-RPROP	4.73E-04 ± 1.09E-04	4.77E-04 ± 1.10E-04	1.01 ± 0.01	8.04E-04 ± 2.44E-04	8.04E-04 ± 2.45E-04	1 ± 0.01	1.08E-03 ± 2.58E-04	1.08E-03 ± 2.57E-04	1 ± 0.01
Elman-PSO	8.26E-05 ± 1.04E-05	8.37E-05 ± 1.14E-05	1.01 ± 0.01	1.39E-04 ± 3.41E-05	1.29E-04 ± 3.78E-05	0.88 ± 0.01	1.58E-04 ± 5.61E-05	1.43E-04 ± 5.12E-05	0.9 ± 0.01
Elman-CQPSO	1.21E-05 ± 8.75E-07	1.09E-05 ± 7.35E-07	0.91 ± 0.01	1.37E-05 ± 1.09E-06	1.32E-05 ± 1.01E-06	0.97 ± 0.01	1.67E-05 ± 1.37E-06	1.59E-05 ± 1.22E-06	0.95 ± 0.02
Jordan-RPROP	9.57E-03 ± 5.05E-03	9.85E-03 ± 5.59E-03	0.99 ± 0.01	1.13E-02 ± 4.28E-03	1.12E-02 ± 4.25E-03	1 ± 0.01	1.52E-02 ± 5.87E-03	1.51E-02 ± 5.83E-03	1 ± 0.01
Jordan-PSO	8.23E-05 ± 1.57E-05	8.07E-05 ± 1.63E-05	0.97 ± 0.03	1.37E-04 ± 3.93E-05	1.20E-04 ± 3.66E-05	0.86 ± 0.04	1.07E-04 ± 1.88E-05	9.30E-05 ± 1.66E-05	0.88 ± 0.04
Jordan-CQPSO	8.28E-06 ± 4.84E-07	7.77E-06 ± 4.26E-07	0.94 ± 0.01	9.34E-06 ± 5.85E-07	9.44E-06 ± 6.08E-07	1.01 ± 0.01	1.21E-05 ± 3.13E-06	1.16E-05 ± 3.03E-06	0.96 ± 0.01
MRNN-RPROP	1.09E-02 ± 4.14E-03	1.08E-02 ± 4.07E-03	0.99 ± 0	1.19E-02 ± 3.62E-03	1.19E-02 ± 3.61E-03	1 ± 0	2.69E-02 ± 8.38E-03	2.68E-02 ± 8.34E-03	1 ± 0
MRNN-PSO	5.46E-05 ± 7.59E-06	5.09E-05 ± 7.06E-06	0.93 ± 0.02	7.24E-05 ± 9.23E-06	6.12E-05 ± 9.84E-06	0.83 ± 0.02	7.58E-05 ± 9.01E-06	6.82E-05 ± 9.93E-06	0.89 ± 0.03
MRNN-CQPSO	1.01E-05 ± 1.04E-06	9.35E-06 ± 8.81E-07	0.93 ± 0.01	1.15E-05 ± 1.45E-06	1.15E-05 ± 1.19E-06	1.02 ± 0.02	2.55E-05 ± 2.67E-05	2.51E-05 ± 2.67E-05	0.96 ± 0.01
TDNN-RPROP	5.57E-04 ± 1.50E-04	5.54E-04 ± 1.48E-04	1 ± 0.01	1.08E-03 ± 2.96E-04	1.07E-03 ± 2.92E-04	1 ± 0.01	1.70E-03 ± 4.89E-04	1.64E-03 ± 4.65E-04	0.98 ± 0.02
TDNN-PSO	1.20E-04 ± 3.23E-05	1.16E-04 ± 2.90E-05	0.99 ± 0.04	4.26E-04 ± 2.39E-04	4.19E-04 ± 2.36E-04	0.95 ± 0.05	2.00E-04 ± 9.71E-05	1.85E-04 ± 9.16E-05	0.91 ± 0.04
TDNN-CQPSO	3.10E-05 ± 3.76E-05	2.99E-05 ± 3.70E-05	0.95 ± 0.02	1.60E-05 ± 3.13E-06	1.52E-05 ± 1.95E-06	0.99 ± 0.03	1.04E-04 ± 1.25E-04	9.18E-05 ± 1.10E-04	0.93 ± 0.02

that the models trained using either of the PSO variants produced the lowest initial training errors, and also had similar training error progression throughout the models' runs. However, Figure 7.4b shows that the CQSO trained models generalized better than the PSO trained models. This obviously indicates that using CQSO improved the

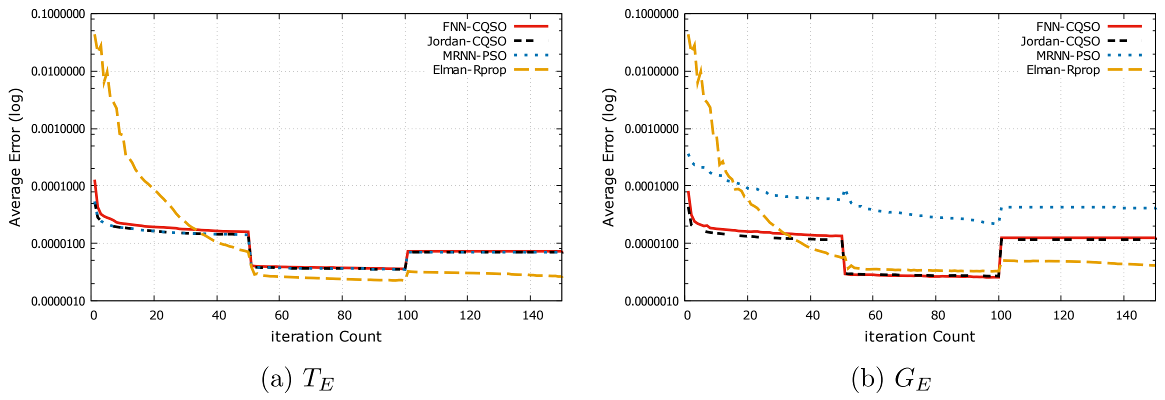


Figure 7.4: Training and generalization error results for HIT time series, scenario A2

performance of the NN models. Figure 7.4 also shows that the FNN trained using CQSO performed on par or even better than the RNN models.

The average ranking of the models over the three scenarios shown in Table 7.10 indicates that the Jordan-CQSO and the FNN-CQSO achieved the first and second

Table 7.10: Models ranking in forecasting the HIT time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	2	2	2	2	1.67	1.67
Elman-RPROP	10	10	10	10	10	10	10	10
Elman-PSO	8	8	8	8	8	8	8	8
Elman-CQPSO	4	4	4	4	3	3	3.67	3.67
Jordan-RPROP	12	12	12	12	12	12	12	12
Jordan-PSO	7	7	7	7	7	7	7	7
Jordan-CQPSO	2	2	1	1	1	1	1.33	1.33
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	6	6	6	6	5	5	5.67	5.67
MRNN-CQPSO	3	3	3	3	4	4	3.33	3.33
TDNN-RPROP	11	11	11	11	11	11	11	11
TDNN-PSO	9	9	9	9	9	9	9	9
TDNN-CQPSO	5	5	5	5	6	6	5.33	5.33

Table 7.11: Results of HIT time series, scenarios B1 to B3

Scenario / Model	**B1** *(f:100, s:250)*			**B2** *(f:100, s:500)*			**B3** *(f:100, s:528)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	5.01E-06 ± 2.11E-07	4.67E-06 ± 1.86E-07	0.93 ± 0	6.14E-06 ± 3.87E-07	6.36E-06 ± 3.83E-07	1.04 ± 0.01	6.75E-06 ± 5.10E-07	6.38E-06 ± 4.26E-07	0.95 ± 0.01
Elman-RPROP	2.07E-04 ± 5.33E-05	2.08E-04 ± 5.32E-05	1 ± 0.01	3.68E-04 ± 9.45E-05	3.66E-04 ± 9.37E-05	1 ± 0.01	6.63E-03 ± 2.41E-03	6.59E-03 ± 2.39E-03	1 ± 0.01
Elman-PSO	6.94E-05 ± 2.23E-05	6.29E-05 ± 1.97E-05	0.91 ± 0.01	7.42E-05 ± 1.02E-05	6.25E-05 ± 9.03E-06	0.84 ± 0.01	9.79E-05 ± 2.17E-05	9.13E-05 ± 2.39E-05	0.92 ± 0.01
Elman-CQPSO	9.62E-06 ± 5.95E-07	8.66E-06 ± 4.96E-07	0.9 ± 0.01	1.19E-05 ± 7.57E-07	1.16E-05 ± 6.27E-07	0.98 ± 0.02	1.59E-05 ± 4.37E-06	1.51E-05 ± 4.53E-06	0.94 ± 0.01
Jordan-RPROP	4.82E-03 ± 1.85E-03	4.78E-03 ± 1.82E-03	0.99 ± 0	5.62E-03 ± 2.36E-03	5.59E-03 ± 2.35E-03	1 ± 0.01	6.63E-03 ± 2.41E-03	6.59E-03 ± 2.39E-03	1 ± 0.01
Jordan-PSO	5.98E-05 ± 2.20E-05	5.70E-05 ± 2.44E-05	0.92 ± 0.03	7.20E-05 ± 1.01E-05	6.41E-05 ± 1.09E-05	0.87 ± 0.03	8.22E-05 ± 1.25E-05	7.50E-05 ± 1.32E-05	0.9 ± 0.03
Jordan-CQPSO	6.91E-06 ± 4.11E-07	6.46E-06 ± 3.79E-07	0.93 ± 0.01	8.27E-06 ± 4.39E-07	8.50E-06 ± 4.41E-07	1.03 ± 0.01	1.02E-05 ± 9.26E-07	9.72E-06 ± 8.83E-07	0.95 ± 0.01
MRNN-RPROP	4.94E-03 ± 1.33E-03	4.86E-03 ± 1.31E-03	0.99 ± 0.01	8.72E-03 ± 3.26E-03	8.64E-03 ± 3.22E-03	1 ± 0.01	1.77E-02 ± 5.35E-03	1.77E-02 ± 5.34E-03	1 ± 0.01
MRNN-PSO	3.43E-05 ± 4.83E-06	3.03E-05 ± 4.84E-06	0.87 ± 0.03	5.23E-05 ± 9.93E-06	4.66E-05 ± 1.06E-05	0.87 ± 0.03	7.25E-05 ± 1.47E-05	6.47E-05 ± 1.49E-05	0.88 ± 0.02
MRNN-CQPSO	8.85E-06 ± 7.08E-07	8.22E-06 ± 5.99E-07	0.93 ± 0.01	5.23E-05 ± 9.93E-06	4.66E-05 ± 1.06E-05	0.87 ± 0.03	1.32E-05 ± 1.52E-06	1.26E-05 ± 1.36E-06	0.96 ± 0.01
TDNN-RPROP	2.36E-04 ± 5.84E-05	2.27E-04 ± 5.47E-05	0.98 ± 0.02	6.56E-04 ± 1.99E-04	6.46E-04 ± 1.95E-04	0.99 ± 0.01	7.71E-04 ± 2.18E-04	7.51E-04 ± 2.10E-04	0.99 ± 0.02
TDNN-PSO	8.08E-05 ± 1.80E-05	8.06E-05 ± 1.89E-05	0.99 ± 0.03	1.33E-04 ± 4.93E-05	1.16E-04 ± 4.47E-05	0.86 ± 0.03	7.71E-04 ± 2.18E-04	7.51E-04 ± 2.10E-04	0.99 ± 0.02
TDNN-CQPSO	9.59E-06 ± 7.72E-07	9.02E-06 ± 6.95E-07	0.94 ± 0.01	1.41E-05 ± 5.62E-06	1.37E-05 ± 3.95E-06	1.03 ± 0.02	1.28E-05 ± 1.01E-06	1.23E-05 ± 8.63E-07	0.96 ± 0.01

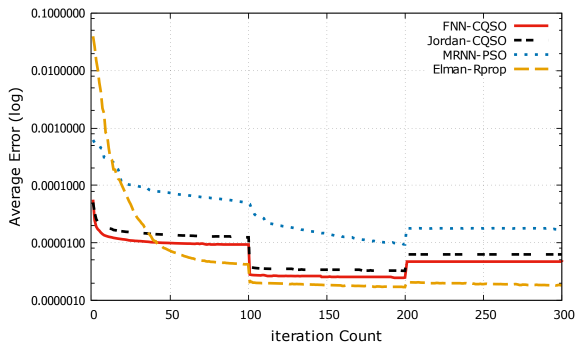
highest average ranks respectively. It is observed that the CQSO based models achieved higher average ranks than the other models.

Scenarios B1 to B3: Table 7.11 shows that the FNN-CQSO model produced superior performance compared to the other models. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold. This indicates that the difference in performance between the FNN-CQSO and the other models were statistically significant. The performance ranking of the models given in Table 7.12 shows that the FNN-CQSO model achieved the highest rank in terms of both training and generalization. The remaining CQSO trained models achieved higher ranks compared to the PSO and the RPROP trained models. The RPROP based models achieved the lowest ranks.

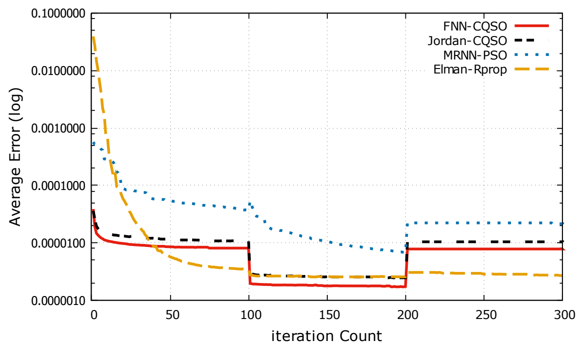
The ρ values obtained by all the models indicated good generalization behaviour.

Figure 7.5 visually illustrates the error progression over time for four selected top performing models for scenario B2. The figure shows that the MRNN-PSO model had the worst error progression. The figure also shows that the FNN-CQSO located the lowest value faster than the other models, and performed better than the Jordan-CQSO throughout the experiment. Figure 7.5a shows that the Elman-RPROP recovered from environmental changes faster than the other models.

It is observed that the performance of all the models, more especially that of the FNN-CQSO, improved compared to the results obtained for scenarios A1, A2 and A3 (where $f = 50$). This improvement in performance was due to an increase in the value of f (i.e the number of iterations allowed before a change). The errors in Table 7.11 illustrate that tracking of optima became more difficult for all the models as the spatial severity increased. Examination of the average rank achieved by all the models revealed that the CQSO trained models obtained the highest average ranks, while the RPROP



(a) T_E



(b) G_E

Figure 7.5: Training and generalization error results for HIT time series, scenario B2

Table 7.12: Models ranking in forecasting the HIT time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	10	10	10	10	12	12	10.5	10.5
Elman-PSO	8	8	8	7	8	8	8	7.67
Elman-CQPSO	5	4	3	3	5	5	4.33	4
Jordan-RPROP	12	12	12	12	12	12	11.83	11.83
Jordan-PSO	7	7	7	8	7	7	7	7.33
Jordan-CQPSO	2	2	2	2	2	2	2	2
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	6	6	6	6	6	6	5.83	5.83
MRNN-CQPSO	3	3	6	6	4	4	4.17	4.17
TDNN-RPROP	11	11	11	11	10	10	10.5	10.5
TDNN-PSO	9	9	9	9	10	10	9.17	9.17
TDNN-CQPSO	4	5	4	4	3	3	3.67	4

models achieved the lowest average ranks.

Scenarios C1 to C3: The FNN-CQSO model yielded the lowest training and generalization errors compared to all the other models, similar to the results obtained for scenarios B1, B2 and B3. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were less than the 0.0001 threshold. Thus, the FNN-CQSO achieved the highest performance rank in terms of training and generalization, as shown in Table 7.13.

The ρ values in Table 7.6 indicate that all the models showed a slight or no sign of overfitting.

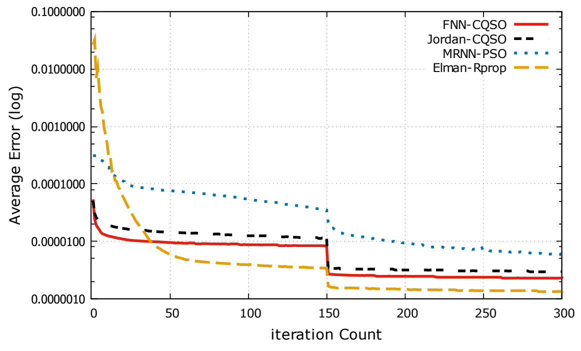
Figure 7.7 shows the performance progression over time for the FNN-CQSO and three other top performing models (i.e Jordan-CQSO, MRNN-PSO and Elman-RPROP) for scenario C2. Figure 7.7a shows that, after the initial few epochs, the Elman-RPROP produced lowest training errors throughout the experiments. Figure 7.7b, however, shows that both the Elman-RPROP and the FNN-CQSO produced similar generalization error

Figure 7.6: Results of HIT time series, scenarios C1 to C3

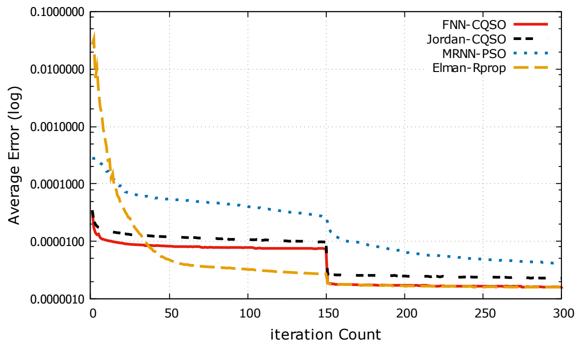
Scenario / Model	**C1** *(f:150, s:250)*			**C2** *(f:150, s:500)*			**C3** *(f:150, s:528)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	5.08E-06 ± 2.09E-07	4.73E-06 ± 1.78E-07	0.93 ± 0.01	5.58E-06 ± 2.07E-07	5.84E-06 ± 2.01E-07	1.05 ± 0.01	6.18E-06 ± 3.93E-07	5.65E-06 ± 3.40E-07	0.92 ± 0.01
Elman-RPROP	2.18E-04 ± 8.96E-05	2.18E-04 ± 8.80E-05	1 ± 0.01	1.99E-04 ± 5.67E-05	2.00E-04 ± 5.63E-05	1.01 ± 0.01	3.87E-04 ± 1.04E-04	3.88E-04 ± 1.05E-04	1 ± 0.01
Elman-PSO	4.10E-05 ± 9.28E-06	3.65E-05 ± 8.08E-06	0.89 ± 0.01	5.71E-05 ± 7.31E-06	4.85E-05 ± 7.42E-06	0.84 ± 0.01	7.02E-05 ± 1.02E-05	6.55E-05 ± 1.20E-05	0.92 ± 0.01
Elman-CQPSO	8.70E-06 ± 4.88E-07	7.91E-06 ± 4.17E-07	0.91 ± 0.01	1.08E-05 ± 7.58E-07	1.07E-05 ± 6.66E-07	0.99 ± 0.01	1.25E-05 ± 1.02E-06	1.15E-05 ± 8.95E-07	0.92 ± 0.01
Jordan-RPROP	3.74E-03 ± 1.14E-03	3.71E-03 ± 1.13E-03	0.99 ± 0	6.08E-03 ± 1.45E-03	6.01E-03 ± 1.42E-03	0.99 ± 0.01	1.07E-02 ± 4.15E-03	1.06E-02 ± 4.06E-03	0.99 ± 0
Jordan-PSO	3.47E-05 ± 4.79E-06	2.86E-05 ± 5.24E-06	0.8 ± 0.03	4.67E-05 ± 5.65E-06	3.97E-05 ± 5.10E-06	0.85 ± 0.02	8.50E-05 ± 2.53E-05	8.02E-05 ± 2.66E-05	0.92 ± 0.03
Jordan-CQPSO	6.00E-06 ± 4.67E-07	5.55E-06 ± 3.31E-07	0.93 ± 0.01	7.74E-06 ± 5.87E-07	7.92E-06 ± 5.17E-07	1.03 ± 0.01	8.21E-06 ± 4.50E-07	7.85E-06 ± 4.09E-07	0.96 ± 0.01
MRNN-RPROP	2.98E-03 ± 1.13E-03	2.91E-03 ± 1.10E-03	0.98 ± 0.01	7.03E-03 ± 2.02E-03	6.98E-03 ± 2.00E-03	1 ± 0	5.17E-03 ± 1.98E-03	5.15E-03 ± 1.97E-03	1 ± 0.01
MRNN-PSO	2.73E-05 ± 3.81E-06	2.29E-05 ± 3.54E-06	0.83 ± 0.02	3.39E-05 ± 2.70E-06	2.92E-05 ± 2.50E-06	0.86 ± 0.02	5.07E-05 ± 1.27E-05	4.46E-05 ± 1.18E-05	0.87 ± 0.02
MRNN-CQPSO	8.28E-06 ± 8.26E-07	7.76E-06 ± 7.33E-07	0.94 ± 0.01	9.59E-06 ± 9.82E-07	9.71E-06 ± 8.34E-07	1.02 ± 0.01	1.12E-05 ± 2.39E-06	1.07E-05 ± 2.10E-06	0.96 ± 0.01
TDNN-RPROP	1.91E-04 ± 5.94E-05	1.88E-04 ± 5.74E-05	1 ± 0.01	3.24E-04 ± 9.91E-05	3.12E-04 ± 9.36E-05	0.99 ± 0.02	4.50E-04 ± 1.37E-04	4.33E-04 ± 1.32E-04	0.98 ± 0.02
TDNN-PSO	8.06E-05 ± 2.20E-05	7.74E-05 ± 2.23E-05	0.95 ± 0.03	1.44E-04 ± 5.19E-05	1.36E-04 ± 5.37E-05	0.89 ± 0.03	9.87E-05 ± 2.53E-05	9.65E-05 ± 3.00E-05	0.93 ± 0.04
TDNN-CQPSO	1.23E-05 ± 5.16E-06	1.14E-05 ± 4.87E-06	0.93 ± 0.01	1.22E-05 ± 2.81E-06	1.24E-05 ± 2.63E-06	1.04 ± 0.01	1.59E-05 ± 7.42E-06	1.49E-05 ± 6.77E-06	0.95 ± 0.01

progression after the first environmental change. Other observations made are similar to that of Figure 7.5.

The performance ranking of the models presented in Table 7.13 shows that the CQSO trained models obtained the highest average ranks, and that the RPROP based models



(a) T_E



(b) G_E

Figure 7.7: Training and generalization error results for HIT time series, scenario C2

achieved the lowest average ranks.

Table 7.14 shows that the FNN-CQSO model obtained the overall highest average rank in predicting the HIT problem for all nine scenarios. The table also indicates that all the CQSO based models achieved higher ranks than the PSO and RPROP based models.

Table 7.13: Models ranking in forecasting the HIT time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	11	11	10	10	10	10	10.33	10.33
Elman-PSO	8	8	8	8	7	7	7.67	7.67
Elman-CQPSO	4	4	4	4	4	4	4	4
Jordan-RPROP	13	13	12	12	13	13	12.67	12.67
Jordan-PSO	7	7	7	7	8	8	7.33	7.33
Jordan-CQPSO	2	2	2	2	2	2	2	2
MRNN-RPROP	12	12	13	13	12	12	12.33	12.33
MRNN-PSO	6	6	6	6	6	6	6	6
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	10	10	11	11	11	11	10.67	10.67
TDNN-PSO	9	9	9	9	9	9	9	9
TDNN-CQPSO	5	5	5	5	5	5	5	5

Table 7.14: Models ranking in forecasting the HIT time series, scenarios A to C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1.67	1.67	1	1	1	1	1.22	1.22
Elman-RPROP	10	10	10.5	10.5	10.33	10.33	10.28	10.28
Elman-PSO	8	8	8	7.67	7.67	7.67	7.89	7.78
Elman-CQPSO	3.67	3.67	4.33	4	4	4	4	3.89
Jordan-RPROP	12	12	11.83	11.83	12.67	12.67	12.17	12.17
Jordan-PSO	7	7	7	7.33	7.33	7.33	7.11	7.22
Jordan-CQPSO	1.33	1.33	2	2	2	2	1.78	1.78
MRNN-RPROP	13	13	13	13	12.33	12.33	12.78	12.78
MRNN-PSO	5.67	5.67	5.83	5.83	6	6	5.83	5.83
MRNN-CQPSO	3.33	3.33	4.17	4.17	3	3	3.50	3.50
TDNN-RPROP	11	11	10.5	10.5	10.67	10.67	10.72	10.72
TDNN-PSO	9	9	9.17	9.17	9	9	9.06	9.06
TDNN-CQPSO	5.33	5.33	3.67	4	5	5	4.67	4.78

This implies that using CQSO as a training algorithm improved the performance of all the RNN models.

7.2.3 DMT Time Series

Scenarios A1 to A3: Table 7.15 shows that the FNN-CQSO model outperformed the other models by producing the lowest training and generalization errors. All the p-values for the comparisons between the FNN-CQSO and any of the other models were below the 0.0001 threshold. This implies that the FNN-CQSO produced errors that were statistically not similar to any of the remaining models. The ρ values given in Table 7.15 shows that all the models showed either a slight or no sign of overfitting for scenarios A1 and A3, except the MRNN-RPROP model which overfitted. For scenario A2, the CQSO and the PSO based models overfitted, while the RPROP based models did not.

It is observed that the G_E values produced by the CQSO and the RPROP trained models worsened as the spatial severity increased.

Table 7.15: Result of DMT time series, scenarios A1 to A3

Scenario **Model**	**A1** *(f:10, s:200)*			**A2** *(f:10, s:400)*			**A3** *(f:10, s:510)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FNN-CQSO	1.19E-05 ± 2.43E-07	1.24E-05 ± 2.75E-07	1.04 ± 0	1.19E-05 ± 3.12E-07	1.39E-05 ± 2.81E-07	1.17 ± 0.01	1.20E-05 ± 2.93E-07	1.30E-05 ± 2.40E-07	1.09 ± 0.01
Elman-RPROP	7.29E-04 ± 2.35E-04	7.29E-04 ± 2.36E-04	1 ± 0.01	1.57E-03 ± 3.53E-04	1.58E-03 ± 3.54E-04	1 ± 0.01	1.32E-03 ± 3.21E-04	1.32E-03 ± 3.17E-04	1 ± 0.01
Elman-PSO	4.42E-05 ± 1.31E-05	4.58E-05 ± 1.38E-05	1.04 ± 0.01	5.29E-05 ± 1.72E-05	5.74E-05 ± 1.90E-05	1.14 ± 0.01	6.63E-05 ± 2.43E-05	6.58E-05 ± 2.49E-05	1 ± 0.01
Elman-CQPSO	1.35E-05 ± 2.05E-06	1.38E-05 ± 1.94E-06	1.02 ± 0.01	1.25E-05 ± 3.16E-07	1.45E-05 ± 4.02E-07	1.16 ± 0.01	1.26E-05 ± 3.21E-07	1.36E-05 ± 3.91E-07	1.08 ± 0.01
Jordan-RPROP	1.47E-02 ± 5.10E-03	1.48E-02 ± 5.16E-03	1 ± 0	2.52E-02 ± 6.27E-03	2.51E-02 ± 6.26E-03	1 ± 0	3.91E-02 ± 1.34E-02	3.92E-02 ± 1.34E-02	1 ± 0
Jordan-PSO	2.91E-05 ± 5.71E-06	3.02E-05 ± 5.80E-06	1.05 ± 0.03	3.81E-05 ± 7.66E-06	4.32E-05 ± 7.50E-06	1.19 ± 0.07	7.30E-05 ± 3.91E-05	7.27E-05 ± 3.66E-05	1.02 ± 0.04
Jordan-CQPSO	1.23E-05 ± 4.29E-07	1.27E-05 ± 3.91E-07	1.03 ± 0.01	1.65E-05 ± 8.09E-06	1.88E-05 ± 8.49E-06	1.16 ± 0.02	1.23E-05 ± 4.11E-07	1.32E-05 ± 4.18E-07	1.07 ± 0.01
MRNN-RPROP	1.62E-02 ± 4.71E-03	1.63E-02 ± 4.72E-03	1.01 ± 0	2.26E-02 ± 8.14E-03	2.26E-02 ± 8.17E-03	1 ± 0.02	3.01E-02 ± 9.69E-03	3.02E-02 ± 9.68E-03	1.03 ± 0.04
MRNN-PSO	3.18E-05 ± 7.96E-06	3.19E-05 ± 7.33E-06	1.02 ± 0.03	7.93E-05 ± 3.50E-05	8.70E-05 ± 3.52E-05	1.17 ± 0.06	6.20E-05 ± 1.90E-05	6.40E-05 ± 2.01E-05	1.02 ± 0.04
MRNN-CQPSO	1.28E-05 ± 5.76E-07	1.31E-05 ± 5.28E-07	1.03 ± 0.01	1.26E-05 ± 3.88E-07	1.48E-05 ± 5.22E-07	1.18 ± 0.01	2.75E-05 ± 2.80E-05	3.05E-05 ± 2.82E-05	1.21 ± 0.03
TDNN-RPROP	7.99E-04 ± 2.40E-04	7.81E-04 ± 2.33E-04	0.98 ± 0.04	1.69E-03 ± 4.26E-04	1.60E-03 ± 4.02E-04	0.96 ± 0.01	1.31E-03 ± 3.79E-04	1.23E-03 ± 3.48E-04	0.95 ± 0.02
TDNN-PSO	4.90E-04 ± 2.44E-04	4.91E-04 ± 2.42E-04	1.04 ± 0.02	3.11E-04 ± 1.74E-04	3.29E-04 ± 1.80E-04	1.12 ± 0.05	1.88E-04 ± 1.02E-04	1.87E-04 ± 1.01E-04	0.99 ± 0.01
TDNN-CQPSO	1.19E-05 ± 2.59E-07	1.25E-05 ± 2.71E-07	1.05 ± 0.01	3.14E-05 ± 3.85E-05	3.33E-05 ± 3.80E-05	1.17 ± 0.01	3.49E-05 ± 4.46E-05	3.52E-05 ± 4.35E-05	1.08 ± 0.01

Figure 7.8 shows the performance progression over time for the FNN-CQSO and three other best performing models (which includes the Jordan-CQSO, TDNN-RPROP and TDNN-PSO) for scenario A1. As visualized in the figure, the Elman-RPROP model had the worst initial errors. The two CQSO based models (i.e. FNN-CQSO and TDNN-

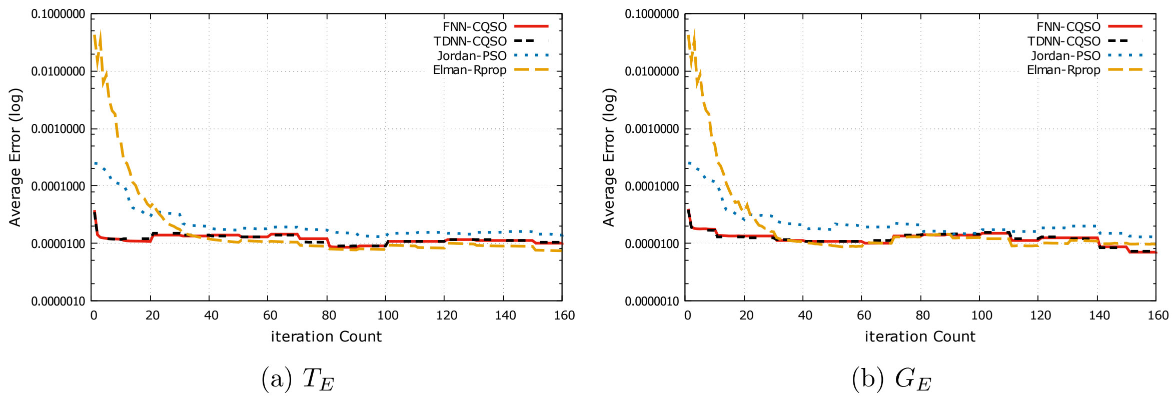


Figure 7.8: Training and generalization error results for DMT time series, scenario A1

CQSO) produced the lowest initial errors and took less than five epochs to locate the lowest error, while the other models took a minimum of about 35 epochs. This indicates that the CQSO based models benefited from the component wise optimization strategy, and enhanced diversity due to the QSO used in the subswarm. The figure also illustrates that, after the initial 35 epochs, all four models successfully recovered from the changes, producing similar performance throughout the remaining epochs, with the Jordan-PSO yielding slightly the worst errors.

The performance ranking of the models given in Table 7.16 shows that the FNN-CQSO model achieved the highest average ranks.

Scenarios B1 to B3: Table 7.17 shows that the FNN-CQSO obtained the lowest CMF T_E and G_E values. All the p-values obtained from the Mann Whitney U tests between the FNN-CQSO and the other models were below the 0.0001 threshold.

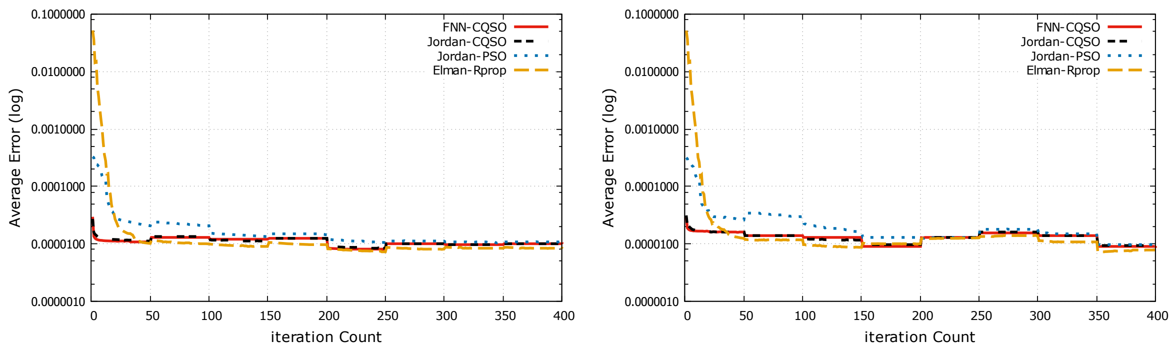
The ρ values in Table 7.17 show that all the models showed either a slight or no sign of overfitting for scenarios B1 and B3. For scenario B2, none of the RPROP trained models overfited, while the CQSO and the PSO trained models overfitted. These generalization behaviours exhibited by the models are similar to what was seen for scenarios A1 to A3.

Figure 7.9 illustrates the error progression over time achieved by the FNN-CQSO and three other selected top performing models for scenario B2. The observations made from the figure are similar to that of Figure 7.8 above.

Table 7.18 shows that the FNN-CQSO obtained the highest performance rank for

Table 7.16: Models ranking in forecasting the DMT time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	10	10	10	10	11	11	10.33	10.33
Elman-PSO	8	8	7	7	7	7	7.33	7.33
Elman-CQPSO	5	5	2	2	3	3	3.33	3.33
Jordan-RPROP	12	12	13	13	13	13	12.67	12.67
Jordan-PSO	6	6	6	6	8	8	6.67	6.67
Jordan-CQPSO	3	3	4	4	2	2	3	3
MRNN-RPROP	13	13	12	12	12	12	12.33	12.33
MRNN-PSO	7	7	8	8	6	6	7	7
MRNN-CQPSO	4	4	3	3	4	4	3.67	3.67
TDNN-RPROP	11	11	11	11	10	10	10.67	10.67
TDNN-PSO	9	9	9	9	9	9	9	9
TDNN-CQPSO	2	2	5	5	5	5	4	4



(a) T_E (b) G_E

Figure 7.9: Training and generalization error results for DMT time series, scenario B2

each of the scenarios, both in terms of training and generalization. Thus, the FNN-CQSO achieved the highest average rank over the three scenarios.

Scenarios C1 to C3: The results in Table 7.19 show that the FNN-CQSO model produced superior performance compared to the other models. All the p-values for the

Table 7.17: Results of DMT time series, scenario B1 to B3

Scenario / **Model**	**B1** *(f:50, s:200)*			**B2** *(f:50, s:400)*			**B3** *(f:50, s:510)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	1.10E-05 ± 1.29E-07	1.17E-05 ± 1.39E-07	1.06 ± 0	1.09E-05 ± 1.30E-07	1.30E-05 ± 1.59E-07	1.2 ± 0	1.09E-05 ± 1.60E-07	1.19E-05 ± 2.10E-07	1.1 ± 0
Elman-RPROP	2.07E-04 ± 5.24E-05	2.12E-04 ± 5.53E-05	1.02 ± 0.02	3.42E-04 ± 1.04E-04	3.44E-04 ± 1.03E-04	1.01 ± 0.01	4.02E-04 ± 1.06E-04	4.02E-04 ± 1.06E-04	1 ± 0.01
Elman-PSO	1.77E-05 ± 2.55E-06	1.83E-05 ± 2.79E-06	1.03 ± 0.02	2.54E-05 ± 6.05E-06	2.81E-05 ± 5.87E-06	1.14 ± 0.01	2.06E-05 ± 3.70E-06	2.17E-05 ± 4.14E-06	1.04 ± 0.01
Elman-CQPSO	1.14E-05 ± 1.25E-07	1.19E-05 ± 1.23E-07	1.05 ± 0	1.13E-05 ± 1.39E-07	1.34E-05 ± 1.74E-07	1.19 ± 0.01	1.14E-05 ± 2.15E-07	1.23E-05 ± 2.71E-07	1.08 ± 0.01
Jordan-RPROP	2.10E-03 ± 6.50E-04	2.12E-03 ± 6.55E-04	1.01 ± 0	5.11E-03 ± 1.91E-03	4.96E-03 ± 1.81E-03	0.99 ± 0.02	9.51E-03 ± 2.95E-03	9.51E-03 ± 2.93E-03	1 ± 0
Jordan-PSO	1.59E-05 ± 1.17E-06	1.62E-05 ± 1.21E-06	1.02 ± 0.01	2.10E-05 ± 3.21E-06	2.44E-05 ± 3.44E-06	1.18 ± 0.03	2.12E-05 ± 3.15E-06	2.13E-05 ± 2.89E-06	1.01 ± 0.02
Jordan-CQPSO	1.13E-05 ± 1.67E-07	1.19E-05 ± 1.86E-07	1.05 ± 0.01	1.10E-05 ± 1.56E-07	1.31E-05 ± 1.87E-07	1.18 ± 0.01	1.11E-05 ± 1.62E-07	1.20E-05 ± 2.11E-07	1.08 ± 0.01
MRNN-RPROP	3.85E-03 ± 2.14E-03	3.88E-03 ± 2.17E-03	1.01 ± 0.01	4.47E-03 ± 1.14E-03	4.44E-03 ± 1.14E-03	0.99 ± 0.01	5.62E-03 ± 2.17E-03	5.64E-03 ± 2.18E-03	1.01 ± 0.01
MRNN-PSO	1.66E-05 ± 1.40E-06	1.72E-05 ± 1.52E-06	1.04 ± 0.01	2.27E-05 ± 5.45E-06	2.60E-05 ± 5.10E-06	1.17 ± 0.03	2.26E-05 ± 5.04E-06	2.34E-05 ± 5.30E-06	1.03 ± 0.02
MRNN-CQPSO	1.14E-05 ± 1.45E-07	1.20E-05 ± 1.69E-07	1.05 ± 0	1.20E-05 ± 8.11E-07	1.40E-05 ± 7.08E-07	1.18 ± 0.01	1.13E-05 ± 2.10E-07	1.23E-05 ± 3.08E-07	1.08 ± 0.01
TDNN-RPROP	3.98E-04 ± 1.78E-04	3.99E-04 ± 1.79E-04	1 ± 0.02	4.85E-04 ± 2.14E-04	4.48E-04 ± 1.87E-04	0.97 ± 0.02	6.28E-04 ± 2.46E-04	6.34E-04 ± 2.60E-04	0.97 ± 0.03
TDNN-PSO	4.11E-04 ± 2.68E-04	4.13E-04 ± 2.70E-04	1.04 ± 0.02	4.32E-04 ± 2.65E-04	4.46E-04 ± 2.77E-04	1.06 ± 0.05	7.65E-05 ± 5.35E-05	7.56E-05 ± 5.12E-05	1 ± 0.04
TDNN-CQPSO	1.48E-05 ± 5.07E-06	1.54E-05 ± 4.85E-06	1.06 ± 0.01	1.30E-05 ± 2.31E-06	1.51E-05 ± 2.31E-06	1.18 ± 0.01	2.64E-05 ± 2.06E-05	2.66E-05 ± 1.94E-05	1.08 ± 0.02

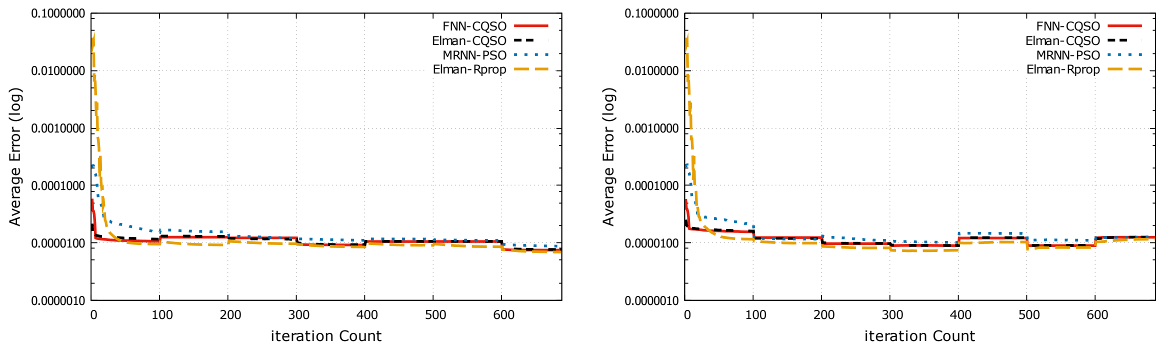
pairwise comparisons between the FNN-CQSO and the other models were less than the 0.0001 threshold, except for scenario C1, where the FNN-CQSO produced statistically similar errors compared to the TDNN-RPROP model. For scenario C2, the p-values indicate that the performance of the FNN-CQSO, compared to the Elman-CQSO, the

Jordan-RPROP, the Jordan-PSO, the MRNN-CQSO and the TDNN-RPROP, were statistically similar. For scenario C3, the difference in performance between the FNN-CQSO and the TDNN-RPROP was statistically insignificant.

Figure 7.10 shows the error progression over time for the FNN-CQSO and three

Table 7.18: Models ranking in forecasting the DMT time series, scenarios B1-B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	9	9	9	9	10	10	9.33	9.33
Elman-PSO	8	8	8	8	5	6	7	7.33
Elman-CQPSO	3	3	3	3	4	4	3.33	3.33
Jordan-RPROP	12	12	13	13	13	13	12.67	12.67
Jordan-PSO	6	6	6	6	6	5	6	5.67
Jordan-CQPSO	2	2	2	2	2	2	2	2
MRNN-RPROP	13	13	12	12	12	12	12.33	12.33
MRNN-PSO	7	7	7	7	7	7	7	7
MRNN-CQPSO	4	4	4	4	3	3	3.67	3.67
TDNN-RPROP	10	10	11	11	11	11	10.67	10.67
TDNN-PSO	11	11	10	10	9	9	10	10
TDNN-CQPSO	5	5	5	5	8	8	6	6



(a) T_E (b) G_E

Figure 7.10: Training and generalization error results for DMT time series, scenario C3

Table 7.19: Results of DMT time series, scenario C1 to C3

Scenario / **Model**	**C1** *(f:100, s:200)*			**C2** *(f:100, s:400)*			**C3** *(f:100, s:510)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	1.08E-05 ± 1.13E-07	1.15E-05 ± 1.31E-07	1.06 ± 0	1.07E-05 ± 1.29E-07	1.28E-05 ± 1.62E-07	1.2 ± 0	1.08E-05 ± 2.97E-07	1.17E-05 ± 1.16E-07	1.09 ± 0
Elman-RPROP	9.14E-05 ± 2.01E-05	9.25E-05 ± 2.00E-05	1.02 ± 0.01	1.68E-04 ± 5.76E-05	1.70E-04 ± 5.77E-05	1.02 ± 0.01	1.88E-04 ± 4.75E-05	1.88E-04 ± 4.72E-05	1 ± 0.01
Elman-PSO	1.46E-05 ± 1.08E-06	1.49E-05 ± 1.18E-06	1.03 ± 0.01	1.61E-05 ± 1.69E-06	1.86E-05 ± 1.86E-06	1.16 ± 0.01	1.91E-05 ± 2.78E-06	1.97E-05 ± 2.81E-06	1.04 ± 0.01
Elman-CQPSO	1.12E-05 ± 1.03E-07	1.17E-05 ± 1.23E-07	1.05 ± 0.01	1.09E-05 ± 2.32E-07	1.31E-05 ± 2.55E-07	1.2 ± 0.01	1.09E-05 ± 1.33E-07	1.17E-05 ± 1.81E-07	1.07 ± 0.01
Jordan-RPROP	1.22E-03 ± 3.67E-04	1.24E-03 ± 3.76E-04	1.02 ± 0.01	3.60E-03 ± 2.52E-03	3.51E-03 ± 2.31E-03	1.01 ± 0.01	3.79E-03 ± 9.82E-04	3.78E-03 ± 9.83E-04	1 ± 0.01
Jordan-PSO	1.53E-05 ± 1.96E-06	1.57E-05 ± 2.01E-06	1.03 ± 0.01	2.00E-05 ± 7.91E-06	2.25E-05 ± 7.71E-06	1.16 ± 0.02	1.73E-05 ± 2.05E-06	1.78E-05 ± 2.03E-06	1.03 ± 0.02
Jordan-CQPSO	1.22E-05 ± 1.64E-06	1.27E-05 ± 1.43E-06	1.04 ± 0.01	1.11E-05 ± 2.65E-07	1.33E-05 ± 2.41E-07	1.2 ± 0.01	1.10E-05 ± 3.71E-07	1.19E-05 ± 4.09E-07	1.08 ± 0.01
MRNN-RPROP	1.79E-03 ± 5.23E-04	1.77E-03 ± 5.36E-04	0.99 ± 0.04	2.78E-03 ± 9.93E-04	2.79E-03 ± 1.01E-03	1 ± 0.01	3.13E-03 ± 1.20E-03	3.13E-03 ± 1.21E-03	1 ± 0.01
MRNN-PSO	1.43E-05 ± 7.84E-07	1.47E-05 ± 8.81E-07	1.02 ± 0.01	1.78E-05 ± 2.63E-06	2.00E-05 ± 2.59E-06	1.14 ± 0.02	1.54E-05 ± 8.06E-07	1.62E-05 ± 8.61E-07	1.05 ± 0.02
MRNN-CQPSO	1.12E-05 ± 1.17E-07	1.19E-05 ± 1.50E-07	1.06 ± 0.01	1.09E-05 ± 1.49E-07	1.31E-05 ± 1.88E-07	1.2 ± 0.01	1.11E-05 ± 1.10E-07	1.19E-05 ± 1.65E-07	1.08 ± 0.01
TDNN-RPROP	1.50E-04 ± 6.51E-05	1.32E-04 ± 4.85E-05	0.96 ± 0.03	2.39E-04 ± 8.17E-05	2.17E-04 ± 6.41E-05	0.96 ± 0.03	2.97E-04 ± 9.35E-05	2.90E-04 ± 9.15E-05	0.99 ± 0.01
TDNN-PSO	1.82E-04 ± 8.13E-05	1.84E-04 ± 8.09E-05	1.03 ± 0.02	5.46E-04 ± 3.12E-04	5.35E-04 ± 3.05E-04	0.99 ± 0.05	3.41E-04 ± 1.45E-04	3.39E-04 ± 1.45E-04	0.98 ± 0.01
TDNN-CQPSO	1.19E-05 ± 4.79E-07	1.26E-05 ± 4.15E-07	1.06 ± 0.01	1.57E-05 ± 4.06E-06	1.77E-05 ± 3.90E-06	1.17 ± 0.02	1.17E-05 ± 8.94E-07	1.27E-05 ± 8.28E-07	1.09 ± 0.01

selected top performing models for scenario C3. All observations made are similar to that of Figure 7.8 above.

The ρ values in Table 7.19 indicate that all the models showed good generalisation behaviour for scenarios C1 and C3. For scenario C2, the CQSO and the PSO trained models overfitted, while the RPROP trained models exhibited either minor, or no overfitting baheviour.

Table 7.20 shows that the FNN-CQSO model obtained the highest average rank over the three scenarios. It is observed that the CQSO based models achieved higher ranks compared to the PSO and RPROP based models.

Table 7.20: Models ranking in forecasting the DMT time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	9	9	9	9	9	9	9	9
Elman-PSO	7	7	6	6	8	8	7	7
Elman-CQPSO	2	2	3	3	2	2	2.33	2.33
Jordan-RPROP	12	12	13	13	13	13	12.67	12.67
Jordan-PSO	8	8	8	8	7	7	7.67	7.67
Jordan-CQPSO	5	5	4	4	3	3	4	4
MRNN-RPROP	13	13	12	12	12	12	12.33	12.33
MRNN-PSO	6	6	7	7	6	6	6.33	6.33
MRNN-CQPSO	3	3	2	2	4	4	3	3
TDNN-RPROP	10	10	10	10	10	10	10	10
TDNN-PSO	11	11	11	11	11	11	11	11
TDNN-CQPSO	4	4	5	5	5	5	4.67	4.67

The values in Table 7.21 show that the FNN-CQSO obtained the overall highest average rank for all nine scenarios, and therefore emerged as the winner in predicting the DMT problem. The table also reveals that the CQSO trained models achieved higher average ranks compared to the PSO and the RPROP trained models. This implies that the CQSO generally improved the performance of the RNNs in predicting the DMT time series for the nine dynamic scenarios considered.

Table 7.21: Overall ranking of the models in predicting the DMT problem, scenarios A to C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	10.33	10.33	9.33	9.33	9	9	9.56	9.56
Elman-PSO	7.33	7.33	7	7.33	7	7	7.11	7.22
Elman-CQPSO	3.33	3.33	3.33	3.33	2.33	2.33	2.99	2.99
Jordan-RPROP	12.67	12.67	12.67	12.67	12.67	12.67	12.67	12.67
Jordan-PSO	6.67	6.67	6	5.67	7.67	7.67	6.78	6.67
Jordan-CQPSO	3	3	2	2	4	4	3	3
MRNN-RPROP	12.33	12.33	12.33	12.33	12.33	12.33	12.33	12.33
MRNN-PSO	7	7	7	7	6.33	6.33	6.78	6.78
MRNN-CQPSO	3.67	3.67	3.67	3.67	3	3	3.45	3.45
TDNN-RPROP	10.67	10.67	10.67	10.67	10	10	10.45	10.45
TDNN-PSO	9	9	10	10	11	11	10	10
TDNN-CQPSO	4	4	6	6	4.67	4.67	4.89	4.98

7.2.4 MG Time Series

Scenarios A1 to A3: The results presented in Table 7.22 show that the FNN-CQSO achieved the best performance by yielding the lowest training and generalization errors compared to all the other models. All the p-values for the Mann Whitney U test comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold. These p-values confirmed that the difference in performance between the FNN-CQSO and the other models were statistically significant. Therefore, the FNN-CQSO achieved the highest rank, both in terms of training and generalization, as shown in Table 7.23.

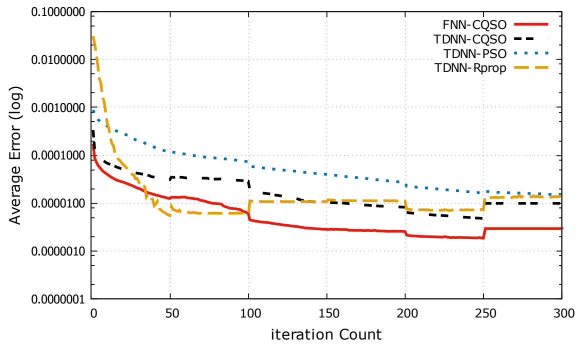
The ρ values in Table 7.22 illustrate that none of the CQSO and the PSO trained models overfitted. The RPROP trained models, however, either overfitted or showed slight signs of overfitting for scenario A1. For scenario A2, the RPROP trained models overfitted slightly. For scenario A3, none of the models trained using RPROP overfitted.

Figure 7.11 illustrates the performance progression over time for the FNN-CQSO and three other selected models for scenario A1. The three models selected obtained the best

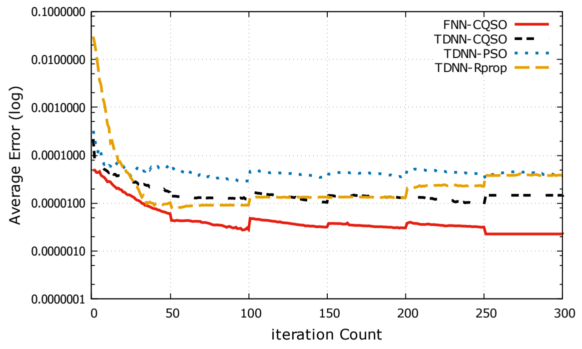
Table 7.22: Results of MG time series, scenario A1-A3

Scenario / Model	**A1** *(f:50, s:30)*			**A2** *(f:50, s:60)*			**A3** *(f:50, s:84)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	5.69E-06	3.50E-06	0.62	8.66E-06	5.80E-06	0.67	1.03E-05	8.31E-06	0.82
	± 4.29E-07	± 2.33E-07	± 0.02	± 9.69E-07	± 6.63E-07	± 0.02	± 9.17E-07	± 7.09E-07	± 0.03
Elman-RPROP	5.82E-04	6.86E-04	1.15	5.20E-04	5.54E-04	1.06	5.29E-04	5.27E-04	0.93
	± 4.42E-04	± 5.14E-04	± 0.1	± 1.55E-04	± 1.74E-04	± 0.17	± 1.89E-04	± 2.10E-04	± 0.04
Elman-PSO	7.71E-05	4.66E-05	0.61	9.73E-05	5.56E-05	0.57	1.44E-04	6.11E-05	0.45
	± 6.92E-06	± 6.07E-06	± 0.1	± 8.31E-06	± 5.56E-06	± 0.17	± 1.88E-05	± 4.88E-06	± 0.04
Elman-CQPSO	3.69E-05	3.55E-05	0.97	4.55E-05	4.38E-05	0.97	5.15E-05	3.79E-05	0.73
	± 5.28E-06	± 6.37E-06	± 0.11	± 5.17E-06	± 5.02E-06	± 0.05	± 5.49E-06	± 4.43E-06	± 0.03
Jordan-RPROP	1.99E-01	2.02E-01	1.06	1.18E-01	1.07E-01	1.01	4.23E-02	4.27E-02	0.99
	± 1.86E-01	± 1.89E-01	± 0.06	± 1.25E-01	± 1.15E-01	± 0.05	± 7.03E-02	± 7.03E-02	± 0.03
Jordan-PSO	7.59E-05	4.21E-05	0.55	8.77E-05	5.04E-05	0.59	1.18E-04	4.99E-05	0.44
	± 5.75E-06	± 4.03E-06	± 0.03	± 6.10E-06	± 4.39E-06	± 0.06	± 1.10E-05	± 4.46E-06	± 0.04
Jordan-CQPSO	1.36E-05	1.31E-05	0.94	2.06E-05	1.86E-05	0.9	2.46E-05	1.82E-05	0.73
	± 1.41E-06	± 2.22E-06	± 0.11	± 2.27E-06	± 2.35E-06	± 0.04	± 3.07E-06	± 2.44E-06	± 0.02
MRNN-RPROP	9.64E-03	9.90E-03	1.03	1.42E-02	1.66E-02	1.06	1.09E-02	1.11E-02	0.98
	± 2.84E-03	± 2.86E-03	± 0.02	± 6.73E-03	± 9.29E-03	± 0.07	± 6.11E-03	± 6.33E-03	± 0.03
MRNN-PSO	7.60E-05	4.81E-05	0.63	1.02E-04	5.59E-05	0.57	1.13E-04	4.81E-05	0.44
	± 5.99E-06	± 5.11E-06	± 0.04	± 1.17E-05	± 6.56E-06	± 0.06	± 8.38E-06	± 3.05E-06	± 0.03
MRNN-CQPSO	9.15E-05	5.45E-05	0.61	1.15E-04	9.66E-05	0.86	1.21E-04	6.69E-05	0.59
	± 1.13E-05	± 6.06E-06	± 0.05	± 1.38E-05	± 1.28E-05	± 0.06	± 1.52E-05	± 6.00E-06	± 0.06
TDNN-RPROP	1.33E-04	1.70E-04	1.29	2.97E-04	2.93E-04	1	2.88E-04	2.71E-04	0.93
	± 2.50E-05	± 3.12E-05	± 0.07	± 5.72E-05	± 5.56E-05	± 0.02	± 8.04E-05	± 7.76E-05	± 0.02
TDNN-PSO	6.83E-05	4.46E-05	0.64	8.19E-05	4.59E-05	0.56	1.05E-04	4.99E-05	0.48
	± 6.71E-06	± 6.34E-06	± 0.05	± 5.03E-06	± 5.28E-06	± 0.05	± 9.19E-06	± 4.42E-06	± 0.03
TDNN-CQPSO	1.81E-05	1.61E-05	0.91	2.14E-05	1.72E-05	0.81	2.80E-05	2.03E-05	0.73
	± 2.97E-06	± 2.48E-06	± 0.04	± 2.48E-06	± 1.91E-06	± 0.03	± 2.87E-06	± 2.15E-06	± 0.02

results from the RPROP, the PSO and the CQSO trained models. Figure 7.11 shows that all the models had a stable performance progression throughout the experiments, with a slight drop in training errors and a minor increase in the peak of the generalization errors after environmental changes. The figure also shows that the FNN-CQSO model



(a) T_E



(b) G_E

Figure 7.11: Training and generalization error results for MG time series, scenario A1

outperformed the other models right from the first epoch to the end.

Scenarios B1 to B3: Table 7.24 shows that the FNN-CQSO produced the lowest CMF T_E and G_E values compared to the other models. All the p-values for the pair-wise comparisons between the FNN-CQSO and the other models were below the 0.0001

Table 7.23: Models ranking in forecasting the MG time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	11	11	11	11	11	11	11	11
Elman-PSO	8	7	7	7	9	8	8	7.33
Elman-CQPSO	4	4	4	4	4	4	4	4
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	6	5	6	6	7	6	6.33	5.67
Jordan-CQPSO	2	2	2	3	2	2	2	2.33
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	7	8	8	8	6	5	7	7
MRNN-CQPSO	9	9	9	9	8	9	8.67	9
TDNN-RPROP	10	10	10	10	10	10	10	10
TDNN-PSO	5	6	5	5	5	7	5	6
TDNN-CQPSO	3	3	3	2	3	3	3	2.67

Table 7.24: Results of MG time series, scenario B1 to B3

Scenario / Model	**B1** *(f:100, s:30)*			**B2** *(f:100, s:60)*			**B3** *(f:100, s:84)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	4.22E-06 ± 2.65E-07	2.74E-06 ± 1.53E-07	0.65 ± 0.02	5.29E-06 ± 5.17E-07	3.84E-06 ± 3.51E-07	0.73 ± 0.02	5.93E-06 ± 5.81E-07	5.22E-06 ± 4.58E-07	0.89 ± 0.02
Elman-RPROP	9.85E-04 ± 8.32E-04	1.04E-03 ± 8.46E-04	1.17 ± 0.09	7.78E-04 ± 4.55E-04	8.62E-04 ± 5.19E-04	1.03 ± 0.04	3.90E-04 ± 2.12E-04	3.57E-04 ± 2.07E-04	0.89 ± 0.06
Elman-PSO	4.77E-05 ± 7.25E-06	3.80E-05 ± 6.17E-06	0.83 ± 0.09	6.95E-05 ± 6.95E-06	4.33E-05 ± 5.86E-06	0.62 ± 0.04	8.13E-05 ± 1.15E-05	3.94E-05 ± 3.73E-06	0.5 ± 0.06
Elman-CQPSO	2.50E-05 ± 2.77E-06	2.89E-05 ± 4.45E-06	1.16 ± 0.13	4.05E-05 ± 1.80E-05	2.72E-05 ± 3.34E-06	0.84 ± 0.07	3.54E-05 ± 7.08E-06	2.37E-05 ± 2.84E-06	0.72 ± 0.05
Jordan-RPROP	1.08E-01 ± 1.45E-01	1.17E-01 ± 1.59E-01	1.05 ± 0.03	9.91E-02 ± 1.32E-01	9.35E-02 ± 1.25E-01	1.04 ± 0.04	3.99E-02 ± 7.24E-02	4.18E-02 ± 7.58E-02	0.99 ± 0.02
Jordan-PSO	4.14E-05 ± 5.43E-06	2.42E-05 ± 3.79E-06	0.58 ± 0.05	5.74E-05 ± 4.36E-06	3.11E-05 ± 3.92E-06	0.55 ± 0.06	6.25E-05 ± 4.10E-06	3.29E-05 ± 2.94E-06	0.53 ± 0.04
Jordan-CQPSO	1.03E-05 ± 1.98E-06	1.26E-05 ± 4.09E-06	1.05 ± 0.18	1.37E-05 ± 1.88E-06	1.19E-05 ± 2.05E-06	0.85 ± 0.06	1.33E-05 ± 1.38E-06	9.66E-06 ± 9.18E-07	0.73 ± 0.03
MRNN-RPROP	7.71E-03 ± 2.83E-03	8.34E-03 ± 3.05E-03	1.08 ± 0.05	1.53E-02 ± 6.85E-03	1.61E-02 ± 7.02E-03	1.07 ± 0.04	8.79E-03 ± 4.22E-03	9.11E-03 ± 4.46E-03	0.99 ± 0.03
MRNN-PSO	4.77E-05 ± 6.73E-06	3.50E-05 ± 4.49E-06	0.78 ± 0.09	6.60E-05 ± 1.21E-05	3.84E-05 ± 4.51E-06	0.62 ± 0.06	6.67E-05 ± 4.69E-06	3.43E-05 ± 3.00E-06	0.51 ± 0.03
MRNN-CQPSO	4.58E-05 ± 5.19E-06	3.76E-05 ± 3.53E-06	0.86 ± 0.08	6.89E-05 ± 6.26E-06	4.93E-05 ± 5.22E-06	0.73 ± 0.06	8.99E-05 ± 1.40E-05	5.09E-05 ± 4.48E-06	0.61 ± 0.05
TDNN-RPROP	7.64E-05 ± 1.27E-05	1.12E-04 ± 1.60E-05	1.53 ± 0.14	1.22E-04 ± 2.40E-05	1.28E-04 ± 2.54E-05	1.05 ± 0.02	1.76E-04 ± 5.01E-05	1.71E-04 ± 4.93E-05	0.96 ± 0.02
TDNN-PSO	3.39E-05 ± 3.31E-06	2.39E-05 ± 3.28E-06	0.7 ± 0.04	5.65E-05 ± 4.29E-06	3.05E-05 ± 3.07E-06	0.54 ± 0.03	6.60E-05 ± 5.13E-06	3.33E-05 ± 3.23E-06	0.51 ± 0.04
TDNN-CQPSO	1.29E-05 ± 1.35E-06	1.26E-05 ± 1.13E-06	1 ± 0.06	1.43E-05 ± 1.26E-06	1.21E-05 ± 9.63E-07	0.86 ± 0.04	1.62E-05 ± 2.46E-06	1.29E-05 ± 1.52E-06	0.82 ± 0.03

threshold. This indicates that the errors produced by the FNN-CQSO are significantly different compared to the errors produced by the other models.

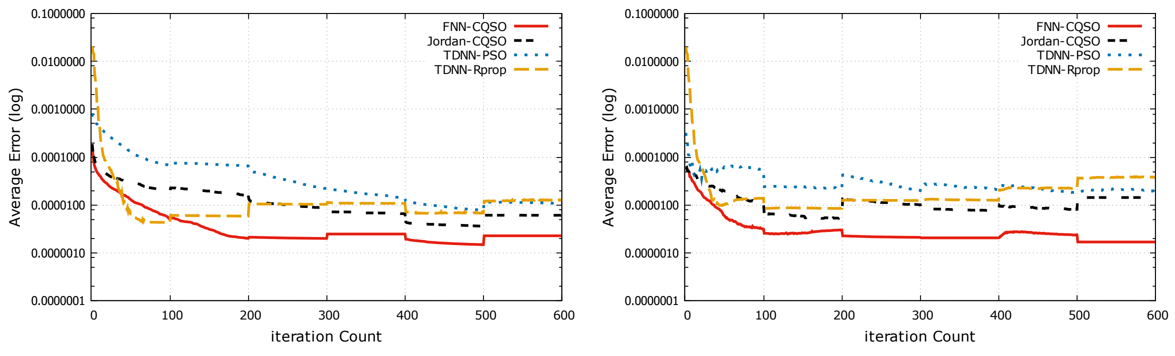
The ρ values in Table 7.24 indicate that all the models showed a slight or no sign of overfitting behaviour, except for scenario B1, where the Elman-RPROP, the TDNN-

Table 7.25: Models ranking in forecasting the MG time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	11	11	11	11	11	11	11	11
Elman-PSO	8	9	9	8	8	8	8.33	8.33
Elman-CQPSO	4	6	4	4	4	4	4	4.67
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	6	5	6	6	5	5	5.67	5.33
Jordan-CQPSO	2	3	2	2	2	2	2	2.33
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	9	7	7	7	7	7	7.67	7
MRNN-CQPSO	7	8	8	9	9	9	8	8.67
TDNN-RPROP	10	10	10	10	10	10	10	10
TDNN-PSO	5	4	5	5	6	6	5.33	5
TDNN-CQPSO	3	2	3	3	3	3	3	2.67

RPROP, and the Elman-CQSO overfitted.

Figure 7.12 illustrates performance progression over time, achieved by the FNN-CQSO and three selected top performing models for scenario B2. Observations from the figure are similar to that of Figure 7.11.



(a) T_E (b) G_E

Figure 7.12: Training and generalization error results for MG time series, scenario B2

The performance ranking of the models given in Table 7.25 shows that the FNN-CQSO achieved the highest average rank.

Scenarios C1 to C3: The T_E and G_E values in Table 7.26 show that the FNN-CQSO outperformed the other models. All the p-values for the pairwise comparisons

Table 7.26: Results of MG time series, scenario C1 to C3

Scenario / Model	**C1** *(f:200, s:30)*			**C2** *(f:200, s:60)*			**C3** *(f:200, s:84)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	3.32E-06 ± 1.53E-07	2.53E-06 ± 8.79E-08	0.77 ± 0.02	4.01E-06 ± 1.98E-07	3.12E-06 ± 1.23E-07	0.78 ± 0.02	4.34E-06 ± 3.51E-07	4.42E-06 ± 1.77E-07	1.05 ± 0.06
Elman-RPROP	1.35E-03 ± 1.57E-03	1.71E-03 ± 2.34E-03	1.31 ± 0.25	9.11E-04 ± 6.30E-04	1.11E-03 ± 8.40E-04	1.13 ± 0.12	1.52E-04 ± 7.57E-05	1.48E-04 ± 7.88E-05	0.93 ± 0.04
Elman-PSO	3.63E-05 ± 8.52E-06	3.15E-05 ± 6.72E-06	0.93 ± 0.25	4.42E-05 ± 5.96E-06	2.93E-05 ± 5.54E-06	0.65 ± 0.12	5.26E-05 ± 9.79E-06	2.66E-05 ± 3.71E-06	0.53 ± 0.04
Elman-CQPSO	1.70E-05 ± 2.32E-06	2.59E-05 ± 4.60E-06	1.57 ± 0.22	2.56E-05 ± 5.66E-06	2.02E-05 ± 3.43E-06	0.86 ± 0.07	1.96E-05 ± 2.24E-06	1.48E-05 ± 1.59E-06	0.76 ± 0.04
Jordan-RPROP	1.92E-01 ± 1.88E-01	1.90E-01 ± 1.84E-01	1.08 ± 0.06	8.40E-02 ± 1.19E-01	6.62E-02 ± 9.39E-02	1.03 ± 0.05	2.71E-01 ± 2.17E-01	2.77E-01 ± 2.20E-01	1.03 ± 0.05
Jordan-PSO	2.27E-05 ± 3.32E-06	1.75E-05 ± 3.64E-06	0.76 ± 0.12	3.45E-05 ± 3.19E-06	1.98E-05 ± 2.53E-06	0.57 ± 0.04	4.30E-05 ± 7.24E-06	2.03E-05 ± 2.50E-06	0.5 ± 0.05
Jordan-CQPSO	8.08E-06 ± 1.06E-06	1.40E-05 ± 3.52E-06	1.61 ± 0.26	9.44E-06 ± 1.38E-06	9.12E-06 ± 1.76E-06	0.93 ± 0.07	8.83E-06 ± 8.92E-07	6.85E-06 ± 4.77E-07	0.8 ± 0.03
MRNN-RPROP	3.43E-03 ± 1.25E-03	3.41E-03 ± 1.28E-03	1.01 ± 0.06	5.97E-03 ± 2.61E-03	6.86E-03 ± 3.49E-03	1.16 ± 0.09	9.58E-03 ± 5.22E-03	1.01E-02 ± 5.40E-03	1.13 ± 0.21
MRNN-PSO	4.35E-05 ± 1.56E-05	4.05E-05 ± 6.14E-06	1.15 ± 0.18	3.97E-05 ± 5.73E-06	3.17E-05 ± 5.38E-06	0.79 ± 0.04	4.41E-05 ± 6.33E-06	2.18E-05 ± 3.05E-06	0.51 ± 0.05
MRNN-CQPSO	3.88E-05 ± 1.28E-05	3.21E-05 ± 3.87E-06	0.97 ± 0.11	4.40E-05 ± 5.25E-06	3.32E-05 ± 2.78E-06	0.78 ± 0.05	4.91E-05 ± 5.14E-06	3.23E-05 ± 2.62E-06	0.69 ± 0.06
TDNN-RPROP	4.23E-05 ± 9.46E-06	7.35E-05 ± 1.57E-05	1.82 ± 0.23	7.08E-05 ± 1.48E-05	7.65E-05 ± 1.56E-05	1.09 ± 0.03	9.37E-05 ± 1.99E-05	8.84E-05 ± 2.00E-05	0.93 ± 0.02
TDNN-PSO	2.19E-05 ± 1.83E-06	1.75E-05 ± 1.71E-06	0.81 ± 0.06	3.28E-05 ± 2.51E-06	2.04E-05 ± 1.75E-06	0.63 ± 0.03	3.42E-05 ± 2.76E-06	1.79E-05 ± 1.37E-06	0.53 ± 0.02
TDNN-CQPSO	1.07E-05 ± 6.10E-07	2.01E-05 ± 1.35E-06	1.89 ± 0.08	1.09E-05 ± 1.05E-06	1.05E-05 ± 8.30E-07	0.99 ± 0.06	1.00E-05 ± 7.41E-07	8.42E-06 ± 5.85E-07	0.85 ± 0.04

between the FNN-CQSO and the other models were less than the 0.0001 threshold. This confirmed that the difference in performance between the FNN-CQSO and each of the other models was statistically significant.

Figure 7.13 shows the error progression over time for the FNN-CQSO and the three best performing models for scenario C2. The figure shows that the FNN-CQSO outperformed the other models throughout the experiment runs. The figure also shows that the G_E values of the FNN-CQSO model dropped after environmental changes, while at the same time the peak of the G_E of the other models increased. This indicates that the FNN-CQSO's adaptation to the changes was superior compared to the other models.

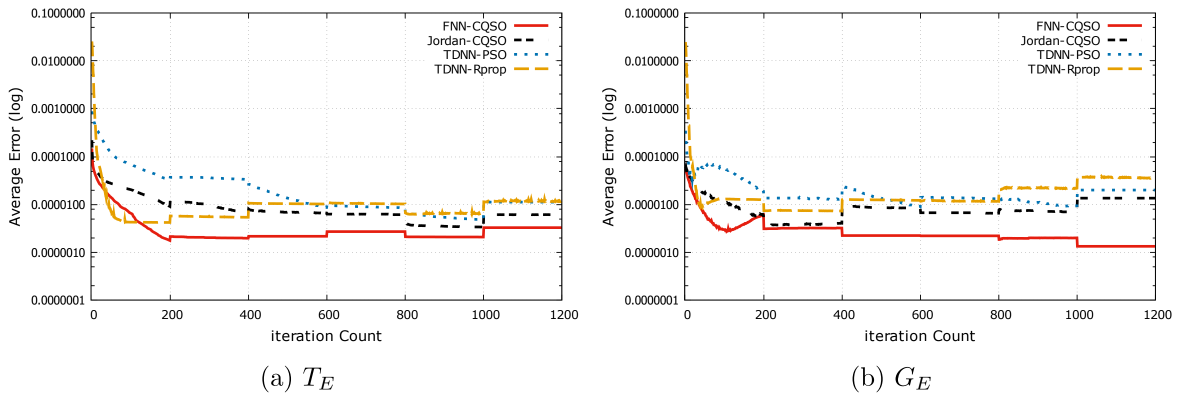


Figure 7.13: Training and generalization error results for MG time series, scenario C2

The ρ values presented in Table 7.26 show that the CQSO and RPROP trained models exhibited overfitting behaviours for scenario C1, except for the FNN-CQSO and the MRNN-CQSO. None of the PSO trained models overfitted, except for the MRNN-PSO model which showed minor signs of overfitting. For scenario C2, the CQSO and PSO trained models had good generalisation behaviour, while the RPROP trained models overfitted. For scenario C3, all the models exhibited minor or no overfitting behaviours, except for the MRNN-RPROP which overfitted.

The ranking of the models shown in Table 7.27 shows that the FNN-CQSO achieved the highest average rank in terms of both training and generalization. The ranking of the models shown in Table 7.27 shows that the FNN-CQSO achieved the highest average rank in terms of both training and generalization.

The overall performance ranking of the models for all nine scenarios, as given in

Table 7.27: Models ranking in forecasting the MG time series, scenarios C1-C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	11	11	11	11	11	11	11	11
Elman-PSO	7	7	9	7	9	8	8.33	7.33
Elman-CQPSO	4	6	4	5	4	4	4	5
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	6	3	6	4	6	6	6	4.33
Jordan-CQPSO	2	2	2	2	2	2	2	2
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	10	9	7	8	7	7	8	8
MRNN-CQPSO	8	8	8	9	8	9	8	8.67
TDNN-RPROP	9	10	10	10	10	10	9.67	10
TDNN-PSO	5	4	5	6	5	5	5	5
TDNN-CQPSO	3	5	3	3	3	3	3	3.67

Table 7.28, show that the FNN-CQSO achieved the highest average rank. This means that a FNN trained using a dynamic PSO (i.e CQSO) outperformed the SRNNs trained using either the RPROP, the PSO, or the CQSO algorithm in predicting the MG time series for the nine different dynamic scenarios considered.

7.2.5 Lorenz Time Series

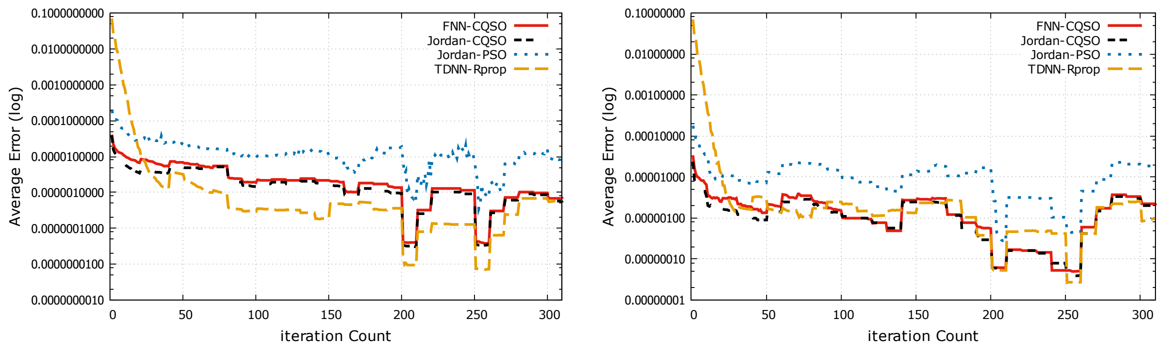
Scenarios A1 to A3: Table 7.29 shows that the Jordan-CQSO and the FNN-CQSO outperformed all the other models by yielding lower errors. The Jordan-CQSO, however, generated lower errors compared to the FNN-CQSO. Thus, the Jordan-CQSO and the FNN-CQSO achieved the 1^{st} and 2^{nd} performance ranking positions, as shown in Table 7.30. All the p-values for the pairwise comparisons between the FNN-CQSO and the remaining models were below the 0.0001 threshold, except for the FNN-CQSO vs MRNN-CQSO comparison. This indicates that the performance of the FNN-CQSO was significantly different compared to the other models, except when compared to the

Table 7.28: Overall ranking of the models in predicting the MG problem, scenarios A to C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	11	11	11	11	11	11	11	11
Elman-PSO	8	7.33	8.33	8.33	8.33	7.33	8.2	7.66
Elman-CQPSO	4	4	4	4.67	4	5	4	4.56
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	6.33	5.67	5.67	5.33	6	4.33	6	5.11
Jordan-CQPSO	2	2.33	2	2.33	2	2	2	2.22
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	7	7	7.67	7	8	8	7.6	7.33
MRNN-CQPSO	8.67	9	8	8.67	8	8.67	8.2	8.78
TDNN-RPROP	10	10	10	10	9.67	10	9.9	10
TDNN-PSO	5	6	5.33	5	5	5	5.1	5.33
TDNN-CQPSO	3	2.67	3	2.67	3	3.67	3	3

MRNN-CQSO.

Figure 7.14 shows the performance progression over time for the four best performing models for scenario A1. The figure shows that, after the initial few epochs, the Jordan-PSO model produced the worst errors throughout the experiments. The TDNN-RPROP



(a) T_E (b) G_E

Figure 7.14: Training and generalization error results for Lorenz time series, scenario A1

Table 7.29: Results of Lorenz time series, scenario A1 to A3

Scenario / **Model**	**A1** *(f:10, s:100)*			**A2** *(f:10, s:250)*			**A3** *(f:10, s:330)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	3.03E-06 ± 3.78E-07	1.99E-06 ± 2.61E-07	0.66 ± 0.03	4.72E-06 ± 7.44E-07	3.80E-06 ± 7.02E-07	0.78 ± 0.03	6.86E-06 ± 1.22E-06	5.81E-06 ± 1.07E-06	0.85 ± 0.04
Elman-RPROP	4.71E-04 ± 1.86E-04	4.94E-04 ± 1.83E-04	1.16 ± 0.16	1.30E-03 ± 5.29E-04	1.31E-03 ± 5.34E-04	1.01 ± 0.01	1.64E-03 ± 4.98E-04	1.65E-03 ± 4.99E-04	1.01 ± 0.01
Elman-PSO	2.05E-05 ± 5.86E-06	1.51E-05 ± 6.20E-06	0.69 ± 0.16	4.01E-05 ± 1.08E-05	3.26E-05 ± 8.69E-06	0.84 ± 0.01	4.21E-05 ± 1.00E-05	3.35E-05 ± 8.96E-06	0.8 ± 0.01
Elman-CQPSO	1.62E-05 ± 3.32E-06	1.59E-05 ± 3.65E-06	0.94 ± 0.04	2.04E-05 ± 3.73E-06	2.05E-05 ± 4.17E-06	0.98 ± 0.04	2.67E-05 ± 1.10E-05	2.51E-05 ± 9.90E-06	0.94 ± 0.05
Jordan-RPROP	6.02E-03 ± 1.83E-03	6.05E-03 ± 1.84E-03	1 ± 0	1.08E-02 ± 4.22E-03	1.10E-02 ± 4.28E-03	1 ± 0.01	1.55E-02 ± 5.48E-03	1.56E-02 ± 5.51E-03	1 ± 0
Jordan-PSO	1.56E-05 ± 2.24E-06	1.29E-05 ± 2.49E-06	0.81 ± 0.08	3.16E-05 ± 9.53E-06	2.81E-05 ± 9.74E-06	0.89 ± 0.07	3.56E-05 ± 8.73E-06	3.50E-05 ± 1.05E-05	0.94 ± 0.06
Jordan-CQPSO	2.22E-06 ± 3.73E-07	1.51E-06 ± 2.36E-07	0.7 ± 0.04	3.06E-06 ± 5.04E-07	2.29E-06 ± 4.64E-07	0.73 ± 0.04	3.26E-06 ± 6.25E-07	2.85E-06 ± 5.83E-07	0.86 ± 0.05
MRNN-RPROP	8.28E-03 ± 2.37E-03	8.51E-03 ± 2.56E-03	1.02 ± 0.01	1.67E-02 ± 6.13E-03	1.70E-02 ± 6.19E-03	1.01 ± 0.01	1.88E-02 ± 7.43E-03	1.89E-02 ± 7.47E-03	1 ± 0.01
MRNN-PSO	2.15E-05 ± 5.92E-06	1.45E-05 ± 2.96E-06	0.76 ± 0.07	3.94E-05 ± 7.98E-06	3.10E-05 ± 8.80E-06	0.74 ± 0.08	4.27E-05 ± 7.18E-06	3.65E-05 ± 7.37E-06	0.83 ± 0.07
MRNN-CQPSO	3.42E-06 ± 6.01E-07	2.31E-06 ± 3.26E-07	0.71 ± 0.06	7.80E-06 ± 3.34E-06	5.46E-06 ± 2.10E-06	0.77 ± 0.06	7.30E-06 ± 2.08E-06	6.19E-06 ± 1.92E-06	0.82 ± 0.05
TDNN-RPROP	4.83E-04 ± 1.66E-04	4.78E-04 ± 1.63E-04	1 ± 0.01	1.23E-03 ± 3.41E-04	1.22E-03 ± 3.32E-04	1.01 ± 0.01	1.44E-03 ± 4.37E-04	1.41E-03 ± 4.26E-04	0.99 ± 0.01
TDNN-PSO	5.90E-04 ± 3.47E-04	5.85E-04 ± 3.43E-04	0.98 ± 0.03	7.49E-04 ± 5.05E-04	7.48E-04 ± 5.07E-04	0.96 ± 0.03	4.32E-04 ± 1.73E-04	4.46E-04 ± 1.84E-04	0.95 ± 0.06
TDNN-CQPSO	2.36E-04 ± 3.10E-04	2.34E-04 ± 3.02E-04	1.03 ± 0.04	2.19E-04 ± 2.17E-04	2.21E-04 ± 2.19E-04	1.01 ± 0.03	2.90E-04 ± 2.26E-04	2.91E-04 ± 2.25E-04	0.99 ± 0.05

model yielded the highest initial errors, but improved subsequently, yielding the lowest T_E value. Figures 7.14a and 7.14b clearly show that the training performance of the TDNN-RPROP was slightly better than its generalization performance, which is an indication of minor overfitting by the model. The figures also show that the two CQSO

Table 7.30: Models ranking in forecasting the Lorenz time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	2	2	2	2	2	2	2	2
Elman-RPROP	9	10	11	11	11	11	10.33	10.67
Elman-PSO	6	6	7	7	6	5	6.33	6
Elman-CQPSO	5	7	4	4	4	4	4.33	5
Jordan-RPROP	12	12	12	12	12	12	12	12
Jordan-PSO	4	4	5	5	5	6	4.67	5
Jordan-CQPSO	1	1	1	1	1	1	1	1
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	7	5	6	6	7	7	6.67	6
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	10	9	10	10	10	10	10	9.67
TDNN-PSO	11	11	9	9	9	9	9.67	9.67
TDNN-CQPSO	8	8	8	8	8	8	8	8

based models (i.e. FNN-CQSO and Jordan-CQSO) had similar error progression profiles. Figure 7.14b shows that the CQSO based models adapted better to the changes than the other models.

The ρ values in Table 7.29 reveal that none of the CQSO or the PSO based models overfitted, except the TDNN-CQSO which exhibited minor overfitting for scenarios A1 and A2. The RPROP trained models exhibited either slight or no signs of overfitting, except for scenario A1, where the Elman-RPROP overfitted.

Table 7.30 shows that the FNN-CQSO achieved the second highest average rank over three scenarios in predicting the Lorenz time series.

Scenarios B1 to B3: Table 7.29 shows that the FNN-CQSO and the Jordan-CQSO outperformed all the other models by producing the lowest T_E and G_E values. However, the two models outperformed each other under different scenarios. While the FNN-CQSO had superior performance for scenario B1, the Jordan-CQSO outperformed the FNN-CQSO for scenario B2. For scenario B3, the Jordan-CQSO produced superior

Table 7.31: Results of Lorenz time series, scenario B1 to B3

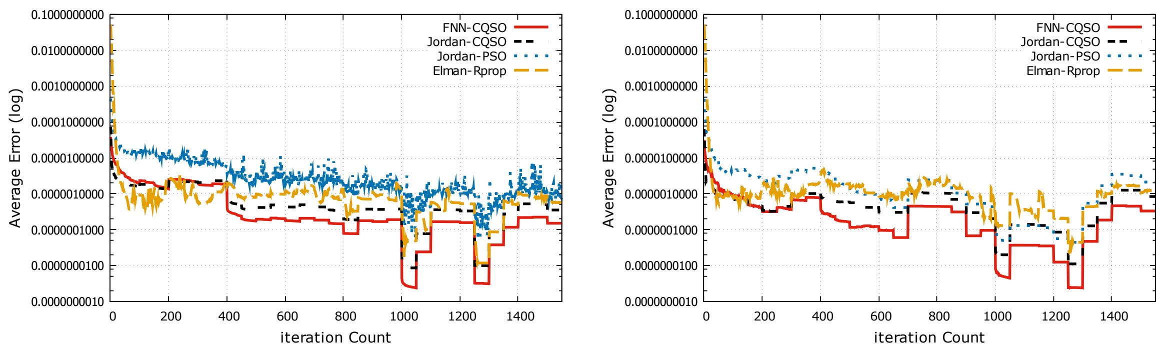
Scenario / **Model**	**B1** *(f:50, s:100)*			**B2** *(f:50, s:250)*			**B3** *(f:50, s:330)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	8.40E-07 ± 1.60E-07	4.47E-07 ± 8.65E-08	0.53 ± 0.02	1.62E-06 ± 1.83E-07	1.18E-06 ± 1.38E-07	0.73 ± 0.01	2.08E-06 ± 2.32E-07	1.53E-06 ± 1.61E-07	0.74 ± 0.03
Elman-RPROP	8.50E-05 ± 2.61E-05	8.55E-05 ± 2.60E-05	1.01 ± 0.01	3.20E-04 ± 8.77E-05	3.23E-04 ± 8.83E-05	1.01 ± 0.01	3.66E-04 ± 1.34E-04	3.66E-04 ± 1.34E-04	1 ± 0.01
Elman-PSO	8.20E-06 ± 2.41E-06	3.08E-06 ± 4.70E-07	0.49 ± 0.01	1.29E-05 ± 4.54E-06	7.68E-06 ± 4.42E-06	0.59 ± 0.01	1.31E-05 ± 2.30E-06	8.22E-06 ± 2.02E-06	0.63 ± 0.01
Elman-CQPSO	5.14E-06 ± 1.10E-06	3.39E-06 ± 9.10E-07	0.65 ± 0.05	7.49E-06 ± 1.06E-06	5.99E-06 ± 1.05E-06	0.78 ± 0.05	1.69E-05 ± 9.47E-06	1.50E-05 ± 8.66E-06	0.87 ± 0.04
Jordan-RPROP	1.29E-03 ± 4.48E-04	1.33E-03 ± 4.76E-04	1.01 ± 0.02	2.54E-03 ± 9.85E-04	2.55E-03 ± 9.86E-04	1.01 ± 0	3.18E-03 ± 1.33E-03	3.20E-03 ± 1.33E-03	1 ± 0.01
Jordan-PSO	5.42E-06 ± 9.05E-07	3.14E-06 ± 5.85E-07	0.61 ± 0.07	1.30E-05 ± 3.46E-06	1.01E-05 ± 3.20E-06	0.75 ± 0.07	1.25E-05 ± 5.29E-06	6.37E-06 ± 1.11E-06	0.65 ± 0.07
Jordan-CQPSO	8.64E-07 ± 1.52E-07	6.43E-07 ± 1.08E-07	0.76 ± 0.05	1.21E-06 ± 1.98E-07	9.21E-07 ± 1.74E-07	0.75 ± 0.04	1.87E-06 ± 6.23E-07	1.64E-06 ± 6.57E-07	0.83 ± 0.04
MRNN-RPROP	1.71E-03 ± 4.17E-04	1.78E-03 ± 4.41E-04	1.05 ± 0.04	4.87E-03 ± 1.62E-03	4.90E-03 ± 1.64E-03	1.01 ± 0.01	9.14E-03 ± 4.49E-03	9.02E-03 ± 4.30E-03	1 ± 0.01
MRNN-PSO	1.01E-05 ± 3.12E-06	3.97E-06 ± 9.55E-07	0.47 ± 0.07	1.90E-05 ± 6.03E-06	8.05E-06 ± 1.65E-06	0.57 ± 0.1	2.09E-05 ± 9.56E-06	8.79E-06 ± 2.53E-06	0.57 ± 0.08
MRNN-CQPSO	1.78E-06 ± 5.27E-07	1.12E-06 ± 1.59E-07	0.76 ± 0.09	2.80E-06 ± 8.81E-07	2.06E-06 ± 8.17E-07	0.72 ± 0.04	2.54E-06 ± 2.89E-07	1.99E-06 ± 2.33E-07	0.79 ± 0.04
TDNN-RPROP	9.53E-05 ± 2.56E-05	1.18E-04 ± 3.31E-05	1.26 ± 0.1	1.82E-04 ± 5.95E-05	1.84E-04 ± 5.99E-05	1.02 ± 0.01	2.72E-04 ± 7.93E-05	2.67E-04 ± 7.67E-05	0.99 ± 0.02
TDNN-PSO	3.13E-04 ± 2.10E-04	3.07E-04 ± 2.13E-04	0.97 ± 0.04	2.55E-04 ± 1.27E-04	2.52E-04 ± 1.27E-04	0.97 ± 0.02	2.08E-04 ± 1.06E-04	2.03E-04 ± 1.07E-04	0.94 ± 0.06
TDNN-CQPSO	4.52E-05 ± 4.31E-05	4.43E-05 ± 4.29E-05	0.91 ± 0.06	3.14E-04 ± 3.83E-04	3.20E-04 ± 3.89E-04	0.99 ± 0.04	3.64E-05 ± 2.40E-05	3.54E-05 ± 2.48E-05	0.93 ± 0.04

training performance and the FNN-CQSO had the best generalization error. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models are less than the 0.0001 threshold, except for scenarios B1 and B3, where the FNN-CQSO and the Jordan-CQSO produced statistically similar T_E and G_E values.

Table 7.32: Models ranking in forecasting the Lorenz time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	2	2	2	1	1.67	1.33
Elman-RPROP	9	9	11	11	11	11	10.33	10.33
Elman-PSO	6	4	5	5	5	5	5.33	4.67
Elman-CQPSO	4	6	4	4	6	7	4.67	5.67
Jordan-RPROP	12	12	12	12	12	12	12	12
Jordan-PSO	5	5	6	7	4	4	5	5.33
Jordan-CQPSO	2	2	1	1	1	2	1.33	1.67
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	7	7	7	6	7	6	7	6.33
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	10	10	8	8	10	10	9.33	9.33
TDNN-PSO	11	11	9	9	9	9	9.67	9.67
TDNN-CQPSO	8	8	10	10	8	8	8.67	8.67

Figure 7.15 illustrates the performance progression over time for the four selected best performing models for scenario B1. As visualized in the figure, the FNN-CQSO clearly outperformed the other models in terms of both training and generalization performance.



(a) T_E (b) G_E

Figure 7.15: Training and generalization error results for Lorenz time series, scenario B1

The performance ranking of the models presented in Table 7.32 shows that the FNN-

CQSO achieved the second and the first highest average training and generalization ranks, respectively.

Scenarios C1 to C3: Table 7.33 shows that the Jordan-CQSO produced the lowest training and generalization errors, except for scenario C1, where the FNN-CQSO yielded

Table 7.33: Results of Lorenz time series, scenario C1 to C3

Scenario / Model	**C1** *(f:100, s:100)*			**C2** *(f:100, s:250)*			**C3** *(f:100, s:330)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQSO	5.15E-07	3.21E-07	0.63	1.02E-06	6.87E-07	0.68	1.12E-06	7.25E-07	0.65
	± 6.51E-08	± 3.71E-08	± 0.02	± 1.31E-07	± 8.21E-08	± 0.01	± 1.10E-07	± 7.43E-08	± 0.02
Elman-RPROP	1.21E-04	1.25E-04	1.02	1.19E-04	1.32E-04	1.17	1.57E-04	1.57E-04	0.99
	± 7.41E-05	± 7.85E-05	± 0.04	± 3.35E-05	± 3.97E-05	± 0.33	± 4.70E-05	± 4.73E-05	± 0.02
Elman-PSO	1.31E-05	2.06E-06	0.37	1.20E-05	4.11E-06	0.5	1.06E-05	4.11E-06	0.58
	± 8.80E-06	± 4.37E-07	± 0.04	± 4.60E-06	± 6.35E-07	± 0.33	± 4.13E-06	± 5.77E-07	± 0.02
Elman-CQSO	3.22E-06	2.08E-06	0.65	5.65E-06	4.59E-06	0.8	5.80E-06	3.84E-06	0.72
	± 4.20E-07	± 2.86E-07	± 0.04	± 8.86E-07	± 8.30E-07	± 0.04	± 1.29E-06	± 6.62E-07	± 0.06
Jordan-RPROP	3.99E-02	4.00E-02	1.01	1.79E-03	1.79E-03	1	2.39E-03	2.40E-03	1.01
	± 7.70E-02	± 7.73E-02	± 0.01	± 6.02E-04	± 6.00E-04	± 0	± 1.24E-03	± 1.23E-03	± 0.01
Jordan-PSO	3.95E-06	1.57E-06	0.5	8.60E-06	4.38E-06	0.59	6.73E-06	4.31E-06	0.66
	± 1.16E-06	± 2.03E-07	± 0.07	± 2.91E-06	± 1.45E-06	± 0.08	± 1.06E-06	± 7.49E-07	± 0.07
Jordan-CQSO	4.80E-07	4.52E-07	0.97	7.76E-07	5.56E-07	0.71	9.30E-07	6.72E-07	0.73
	± 9.39E-08	± 8.22E-08	± 0.08	± 1.79E-07	± 1.38E-07	± 0.03	± 1.48E-07	± 1.07E-07	± 0.03
MRNN-RPROP	8.72E-04	9.45E-04	1.17	2.99E-03	3.01E-03	1.01	2.98E-03	3.02E-03	1.01
	± 2.83E-04	± 2.93E-04	± 0.3	± 1.59E-03	± 1.59E-03	± 0.01	± 1.52E-03	± 1.54E-03	± 0.01
MRNN-PSO	8.29E-06	1.90E-06	0.43	1.01E-05	4.02E-06	0.48	1.16E-05	4.99E-06	0.52
	± 3.73E-06	± 2.52E-07	± 0.08	± 2.23E-06	± 8.35E-07	± 0.09	± 2.79E-06	± 1.24E-06	± 0.08
MRNN-CQSO	1.28E-06	8.24E-07	0.79	1.76E-06	1.30E-06	0.75	2.12E-06	1.70E-06	0.81
	± 4.06E-07	± 1.03E-07	± 0.09	± 2.50E-07	± 1.76E-07	± 0.03	± 3.78E-07	± 2.93E-07	± 0.02
TDNN-RPROP	5.40E-05	1.11E-04	1.99	1.22E-04	1.22E-04	1.02	1.15E-04	1.15E-04	1.01
	± 1.78E-05	± 3.78E-05	± 0.4	± 3.55E-05	± 3.52E-05	± 0.01	± 3.43E-05	± 3.37E-05	± 0.01
TDNN-PSO	4.49E-04	4.45E-04	0.97	3.14E-04	3.11E-04	0.96	2.23E-04	2.26E-04	1
	± 2.69E-04	± 2.68E-04	± 0.03	± 1.86E-04	± 1.88E-04	± 0.03	± 1.67E-04	± 1.72E-04	± 0.06
TDNN-CQSO	1.10E-05	9.94E-06	0.9	9.02E-06	8.78E-06	0.96	3.22E-05	2.96E-05	0.8
	± 5.07E-06	± 4.51E-06	± 0.05	± 2.75E-06	± 2.93E-06	± 0.04	± 1.97E-05	± 1.97E-05	± 0.05

the lowest generalization error. The FNN-CQSO also outperformed the remaining models. All the p-values for the Mann Whitney U test comparisons between the FNN-CQSO and the other models are less than the 0.0001 threshold, except for the FNN-CQSO vs Jordan-CQSO comparison, where the two models had statistically similar T_E values for scenario C1, and similar G_E values for scenario C3.

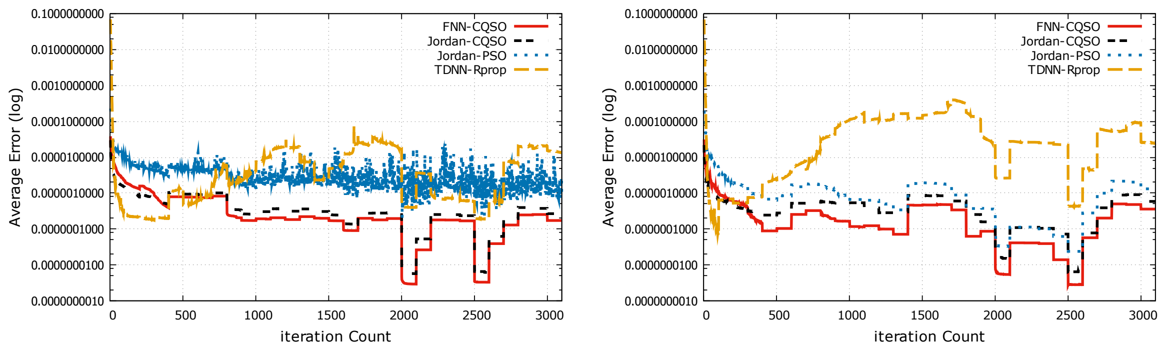
Figure 7.16 shows the performance progression over time achieved by the FNN-CQSO and three top performing models (which includes the Jordan-CQSO, TDNN-RPROP and the Jordan-PSO) for scenario C1. As visualized in Figure 7.16, the FNN-CQSO clearly outperformed the other models. Figure 7.16a illustrates that the Jordan-PSO fluctuated a lot in terms of T_E.

The ρ values in Table 7.33 shows that none of the CQSO and RPROP trained models overfitted. However, all the RPROP trained models exhibited minor signs of overfitting, except for scenario C1 where the MRNN-RPROP and the TDNN-RPROP overfitted, and for scenario C2 where the Elman-RPROP overfitted.

The performance ranking of the models for the three scenarios given in Table 7.34

Table 7.34: Models ranking in forecasting the Lorenz time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	2	1	2	2	2	2	2	1.67
Elman-RPROP	10	10	9	10	10	10	9.67	10
Elman-PSO	8	6	8	5	6	5	7.33	5.33
Elman-CQPSO	4	7	4	7	4	4	4	6
Jordan-RPROP	13	13	12	12	12	12	12.33	12.33
Jordan-PSO	5	4	5	6	5	6	5	5.33
Jordan-CQPSO	1	2	1	1	1	1	1	1.33
MRNN-RPROP	12	12	13	13	13	13	12.67	12.67
MRNN-PSO	6	5	7	4	7	7	6.67	5.33
MRNN-CQPSO	3	3	3	3	3	3	3	3
TDNN-RPROP	9	9	10	9	9	9	9.33	9
TDNN-PSO	11	11	11	11	11	11	11	11
TDNN-CQPSO	7	8	6	8	8	8	7	8



(a) T_E (b) G_E

Figure 7.16: Training and generalization error results for Lorenz time series, scenario C1

shows that the Jordan-CQSO emerged as the winner by achieving the highest average ranks. The FNN-CQSO model achieved the second highest average rank.

Table 7.35: Overall models ranking in forecasting the Lorenz time series, scenario error A-C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	2	2	1.67	1.33	2	1.67	1.89	1.67
Elman-RPROP	10.33	10.67	10.33	10.33	9.67	10	10.11	10.33
Elman-PSO	6.33	6	5.33	4.67	7.33	5.33	6.33	5.33
Elman-CQPSO	4.33	5	4.67	5.67	4	6	4.33	5.56
Jordan-RPROP	12	12	12	12	12.33	12.33	12.11	12.11
Jordan-PSO	4.67	5	5	5.33	5	5.33	4.89	5.22
Jordan-CQPSO	1	1	1.33	1.67	1	1.33	1.11	1.33
MRNN-RPROP	13	13	13	13	12.67	12.67	12.89	12.89
MRNN-PSO	6.67	6	7	6.33	6.67	5.33	6.78	5.89
MRNN-CQPSO	3	3	3	3	3	3	3.00	3
TDNN-RPROP	10	9.67	9.33	9.33	9.33	9	9.55	9.33
TDNN-PSO	9.67	9.67	9.67	9.67	11	11	10.11	10.11
TDNN-CQPSO	8	8	8.67	8.67	7	8	7.89	8.22

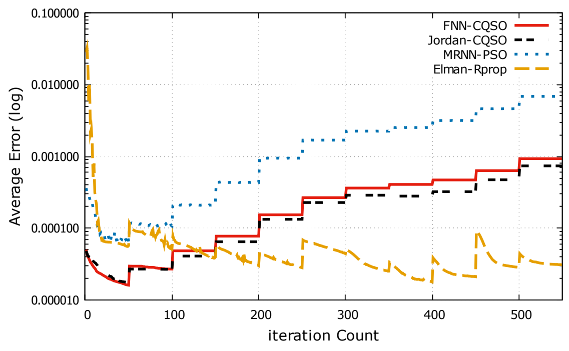
The overall ranking of the models for all nine scenarios, given in Table 7.35, shows that the FNN-CQSO achieved the second overall highest average rank. The Jordan-

CQSO, however, emerged as the overall winner. This indicates that the FNN-CQSO outperformed all the RNN models, except the Jordan-CQSO.

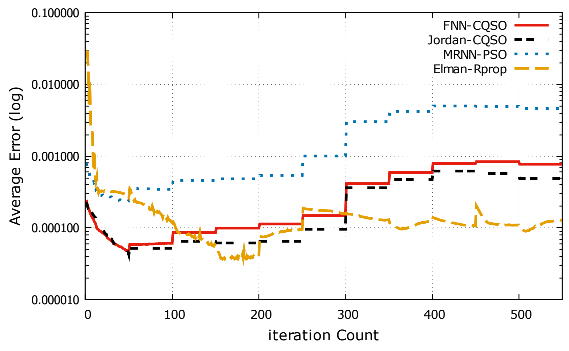
7.2.6 IAP Time Series

Scenarios A1 to A3: Table 7.36 shows that the Elman-RPROP produced the lowest CMF T_E for the gradually changing scenario A1, outperforming all the other models. However, in terms of generalization performance, the Jordan-CQSO model yielded the lowest error. For the severely and abruptly changing scenarios A2 and A3, the FNN-CQSO produced the lowest T_E and G_E values. All the p-values for the pairwise comparisons between the FNN-CQSO and the remaining models are less than the 0.0001 threshold, except for the FNN-CQSO vs Jordan-CQSO comparison for the three scenarios, and for the FNN-CQSO vs Elman-RPROP comparison for scenario A2 with reference to the training error. For these exceptions, the difference in performance between the models were statistically insignificant.

Figure 7.17 illustrates the performance progression over time for four of the top performing models for scenario A1. For the models shown in the figure, the peak of the T_E increased after every change (i.e. 50 epochs) without recovering, except for the Elman-RPROP which recovers immediately and kept track of the moving lowest value. The increase in the G_E peak after the changes, however, were smaller compared to those of the T_E as shown Figure 7.17b.



(a) T_E



(b) G_E

Figure 7.17: Training and generalization error results for AIP time series, scenario A1

Table 7.36: Results of IAP time series, scenario A1 to A3

Model	**A1** *(f:50, s:10)*			**A2** *(f:50, s:25)*			**A3** *(f:50, s:32)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	3.10E-04 ±2.00E-04	3.60E-04 ± 2.40E-04	1.21 ± 0.06	3.89E-04 ± 1.32E-04	4.30E-04 ± 1.36E-04	1.3 ± 0.13	2.76E-04 ± 1.07E-04	5.57E-04 ± 2.13E-04	1.98 ± 0.12
Elman-RPROP	2.30E-04 ± 4.00E-05	3.10E-04 ± 5.00E-05	1.46 ± 0.11	4.11E-04 ± 1.07E-04	5.41E-04 ± 1.05E-04	1.49 ± 0.12	5.48E-04 ± 2.06E-04	6.80E-04 ± 2.15E-04	1.33 ± 0.09
Elman-PSO	2.15E-03 ± 2.30E-04	2.32E-03 ± 2.60E-04	1.07 ± 0.11	2.16E-03 ± 2.21E-04	3.12E-03 ± 2.88E-04	1.46 ± 0.12	2.05E-03 ± 2.20E-04	3.95E-03 ± 3.94E-04	1.94 ± 0.09
Elman-CQPSO	1.75E-03 ± 2.30E-04	2.15E-03 ± 2.80E-04	1.23 ± 0.03	2.49E-03 ± 2.83E-04	2.11E-03 ± 2.84E-04	0.83 ± 0.03	2.07E-03 ± 2.37E-04	3.28E-03 ± 4.08E-04	1.57 ± 0.05
Jordan-RPROP	1.45E-01 ± 1.24E-01	1.47E-01 ± 1.25E-01	1.05 ± 0.02	3.84E-03 ± 3.03E-03	3.93E-03 ± 2.93E-03	1.2 ± 0.07	2.20E-03 ± 1.96E-03	2.88E-03 ± 2.87E-03	1.24 ± 0.08
Jordan-PSO	2.60E-03 ± 5.90E-04	2.82E-03 ± 6.40E-04	1.08 ± 0.03	2.41E-03 ± 4.81E-04	3.70E-03 ± 7.27E-04	1.56 ± 0.07	2.17E-03 ± 4.44E-04	4.18E-03 ± 8.08E-04	1.97 ± 0.08
Jordan-CQPSO	2.40E-04 ± 7.00E-05	2.70E-04 ± 8.00E-05	1.1 ± 0.05	5.03E-04 ± 2.39E-04	6.32E-04 ± 3.06E-04	1.33 ± 0.1	4.24E-04 ± 1.63E-04	8.42E-04 ± 3.15E-04	1.9 ± 0.13
MRNN-RPROP	1.47E-02 ± 1.11E-02	1.58E-02 ± 1.14E-02	1.13 ± 0.06	9.72E-03 ± 6.15E-03	1.09E-02 ± 6.94E-03	1.37 ± 0.12	8.28E-03 ± 8.29E-03	9.15E-03 ± 9.14E-03	1.31 ± 0.11
MRNN-PSO	2.09E-03 ± 2.20E-04	2.27E-03 ± 2.50E-04	1.08 ± 0.01	2.03E-03 ± 3.17E-04	2.94E-03 ± 4.22E-04	1.48 ± 0.05	1.92E-03 ± 2.01E-04	3.71E-03 ± 3.53E-04	1.94 ± 0.02
MRNN-CQPSO	1.36E-03 ± 3.50E-04	1.49E-03 ± 3.90E-04	1.09 ± 0.02	1.15E-03 ± 2.84E-04	1.62E-03 ± 4.00E-04	1.47 ± 0.09	1.12E-03 ± 2.78E-04	2.24E-03 ± 5.16E-04	2.04 ± 0.05
TDNN-RPROP	4.30E-04 ± 1.10E-04	8.60E-04 ± 3.10E-04	1.93 ± 0.2	7.49E-04 ± 1.66E-04	1.23E-03 ± 3.08E-04	1.7 ± 0.15	1.08E-03 ± 2.89E-04	1.55E-03 ± 4.17E-04	1.47 ± 0.14
TDNN-PSO	2.37E-03 ± 3.80E-04	2.58E-03 ± 4.10E-04	1.09 ± 0.01	2.37E-03 ± 3.01E-04	3.47E-03 ± 4.23E-04	1.48 ± 0.05	2.23E-03 ± 2.49E-04	4.25E-03 ± 4.37E-04	1.93 ± 0.06
TDNN-CQPSO	3.58E-03 ± 1.65E-03	4.01E-03 ± 1.83E-03	1.21 ± 0.07	3.85E-03 ± 2.05E-03	4.60E-03 ± 2.35E-03	1.55 ± 0.18	2.47E-03 ± 1.09E-03	4.41E-03 ± 1.91E-03	1.78 ± 0.1

The ρ values in Table 7.36 indicate that all the models overfitted. It is observed that the ρ values produced by the models increased with increase in spatial severity.

The performance ranking of the models shown in Table 7.37 illustrate that the FNN-CQSO obtained the highest average rank over the three scenarios.

Table 7.37: Models ranking in forecasting the IAP time series, scenario A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	3	3	1	1	1	1	1.67	1.67
Elman-RPROP	1	2	2	2	3	2	2	2
Elman-PSO	8	8	7	8	7	9	7.33	8.33
Elman-CQSO	6	6	10	6	8	7	8	6.33
Jordan-RPROP	13	13	11	11	10	6	11.33	10
Jordan-PSO	10	10	9	10	9	10	9.33	10
Jordan-CQSO	2	1	3	3	2	3	2.33	2.33
MRNN-RPROP	12	12	13	13	13	13	12.67	12.67
MRNN-PSO	7	7	6	7	6	8	6.33	7.33
MRNN-CQSO	5	5	5	5	5	5	5	5
TDNN-RPROP	4	4	4	4	4	4	4	4
TDNN-PSO	9	9	8	9	11	11	9.33	9.67
TDNN-CQSO	11	11	12	12	12	12	11.67	11.67

Scenarios B1 to B3: The T_E and G_E values shown in Table 7.38 indicate that the FNN-CQSO model outperformed the other models. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold, with a few exceptions. The exceptions indicate that the difference in errors produced by the FNN-CQSO and the Jordan-CQSO was statistically insignificant for scenarios B2 and B3. The p-values also show that the TDNN-PSO and the TDNN-CQSO produced similar training errors for scenario B1 compared to the FNN-CQSO.

The ρ values in Table 7.38 indicate that all the models overfitted, except for scenario B2 where the Elman-CQSO did not overfit.

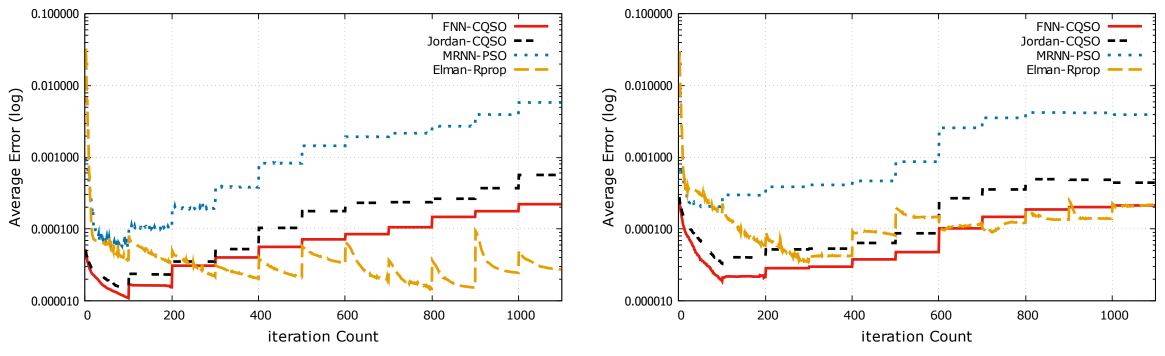
Figure 7.18 illustrates the performance progression over time for scenario B1. The models shown in the figure obtained the best results, one from each of the RPROP, PSO, and CQSO trained models. Observations made from the figure are similar to that of Figure 7.17.

The average performance ranks achieved by the models are shown in Table 7.39. The ranks in the table indicate that the FNN-CQSO outperformed all the other models.

Table 7.38: IAP time series results for scenarios B1 to B3

Model	**B1** *(f:50, s:10)*			**B2** *(f:50, s:25)*			**B3** *(f:50, s:32)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	8.84E-05 ± 2.50E-05	9.78E-05 ± 2.97E-05	1.08 ± 0.06	1.30E-04 ± 4.91E-05	1.70E-04 ± 5.01E-05	1.54 ± 0.13	8.91E-05 ± 2.04E-05	1.51E-04 ± 4.75E-05	1.59 ± 0.18
Elman-RPROP	1.10E-04 ± 2.11E-05	2.05E-04 ± 3.99E-05	2.01 ± 0.28	2.07E-04 ± 4.75E-05	3.18E-04 ± 5.71E-05	1.71 ± 0.15	4.11E-04 ± 1.25E-04	6.23E-04 ± 1.92E-04	1.60 ± 0.18
Elman-PSO	1.79E-03 ± 2.17E-04	1.93E-03 ± 2.44E-04	1.07 ± 0.28	2.03E-03 ± 2.34E-04	2.97E-03 ± 3.41E-04	1.47 ± 0.15	1.72E-03 ± 2.20E-04	3.32E-03 ± 3.89E-04	1.95 ± 0.18
Elman-CQPSO	1.72E-03 ± 2.54E-04	2.11E-03 ± 3.23E-04	1.21 ± 0.05	2.28E-03 ± 3.37E-04	1.94E-03 ± 3.32E-04	0.83 ± 0.04	8.68E-04 ± 2.52E-04	1.71E-03 ± 4.78E-04	1.99 ± 0.09
Jordan-RPROP	4.04E-02 ± 5.05E-02	4.22E-02 ± 5.25E-02	1.19 ± 0.07	1.09E-02 ± 1.00E-02	1.09E-02 ± 1.00E-02	1.23 ± 0.09	1.14E-01 ± 1.13E-01	1.32E-01 ± 1.37E-01	1.14 ± 0.07
Jordan-PSO	2.93E-03 ± 7.24E-04	3.22E-03 ± 8.12E-04	1.08 ± 0.03	2.72E-03 ± 5.75E-04	3.97E-03 ± 7.77E-04	1.51 ± 0.08	2.21E-03 ± 4.78E-04	4.20E-03 ± 8.21E-04	1.99 ± 0.09
Jordan-CQPSO	1.90E-04 ± 7.83E-05	2.21E-04 ± 9.29E-05	1.14 ± 0.05	1.65E-04 ± 6.76E-05	2.16E-04 ± 6.55E-05	1.60 ± 0.16	1.57E-04 ± 6.00E-05	2.93E-04 ± 1.28E-04	1.68 ± 0.18
MRNN-RPROP	6.44E-03 ± 2.78E-03	7.12E-03 ± 2.95E-03	1.18 ± 0.11	1.17E-02 ± 5.62E-03	1.42E-02 ± 7.57E-03	1.26 ± 0.13	1.35E-02 ± 6.27E-03	1.74E-02 ± 8.50E-03	1.29 ± 0.15
MRNN-PSO	1.79E-03 ± 3.25E-04	1.92E-03 ± 3.57E-04	1.06 ± 0.03	1.83E-03 ± 1.93E-04	2.70E-03 ± 2.72E-04	1.49 ± 0.05	1.67E-03 ± 2.01E-04	3.24E-03 ± 3.63E-04	1.95 ± 0.03
MRNN-CQPSO	1.09E-03 ± 3.47E-04	1.21E-03 ± 3.80E-04	1.18 ± 0.07	8.74E-04 ± 2.64E-04	1.17E-03 ± 3.58E-04	1.38 ± 0.10	8.92E-04 ± 2.71E-04	1.78E-03 ± 5.12E-04	2.08 ± 0.04
TDNN-RPROP	2.32E-04 ± 7.16E-05	5.60E-04 ± 2.56E-04	2.21 ± 0.19	6.53E-04 ± 2.57E-04	1.28E-03 ± 6.92E-04	1.88 ± 0.16	5.56E-04 ± 1.42E-04	8.34E-04 ± 1.97E-04	1.66 ± 0.21
TDNN-PSO	1.92E-03 ± 2.30E-04	2.11E-03 ± 2.47E-04	1.10 ± 0.02	2.16E-03 ± 2.29E-04	3.09E-03 ± 3.01E-04	1.45 ± 0.04	2.32E-03 ± 2.12E-04	4.40E-03 ± 3.71E-04	1.91 ± 0.02
TDNN-CQPSO	2.08E-03 ± 7.03E-04	2.36E-03 ± 7.67E-04	1.30 ± 0.10	2.33E-03 ± 1.30E-03	2.84E-03 ± 1.48E-03	1.57 ± 0.19	7.98E-04 ± 4.02E-04	1.45E-03 ± 7.29E-04	1.67 ± 0.17

Scenarios C1 to C3: The error values in Table 7.40 show that the FNN-CQSO outperformed the other models. All the p-values for the pairwise comparisons between the FNN-CQSO and the remaining models are less than the 0.0001 threshold, except for two cases: FNN-CQSO vs Jordan-CQSO for the three scenarios, and FNN-CQSO



(a) T_E (b) G_E

Figure 7.18: Training and generalization error results for IAP time series, scenario B1

vs ELMAN-RPROP with reference to the T_E for scenario C2. Thus, the FNN-CQSO achieved the highest performance rank as shown in Table 7.41.

Figure 7.19 illustrates the performance progression over time for the four best performing models for scenario C1. The models selected include the FNN-CQSO, Jordan-

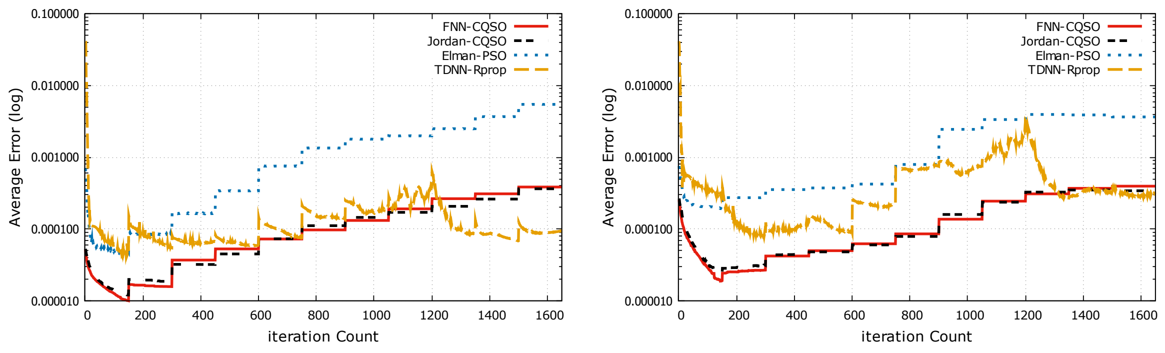
Table 7.39: Models ranking in forecasting the IAP time series, scenario error B1-B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1	1	1	1	1	1	1	1
Elman-RPROP	2	2	3	3	3	3	2.67	2.67
Elman-PSO	8	7	7	9	9	9	8	8.33
Elman-CQSO	6	9	9	6	6	6	7	7
Jordan-RPROP	13	13	12	12	13	13	12.67	12.67
Jordan-PSO	11	11	11	11	10	10	10.67	10.67
Jordan-CQSO	3	3	2	2	2	2	2.33	2.33
MRNN-RPROP	12	12	13	13	12	12	12.33	12.33
MRNN-PSO	7	6	6	7	8	8	7	7
MRNN-CQSO	5	5	5	4	7	7	5.67	5.33
TDNN-RPROP	4	4	4	5	4	4	4	4.33
TDNN-PSO	9	8	8	10	11	11	9.33	9.67
TDNN-CQSO	10	10	10	8	5	5	8.333	7.67

Table 7.40: IAP Results for scenarios C1-C3

Model	**C1** *(f:50, s:10)*			**C2** *(f:50, s:25)*			**C3** *(f:50, s:32)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	1.43E-04	1.61E-04	1.14	1.43E-04	1.61E-04	1.14	1.00E-04	1.28E-04	1.33
	± 4.59E-05	± 5.49E-05	± 0.08	± 4.59E-05	± 5.49E-05	± 0.08	± 2.43E-05	± 3.02E-05	± 0.16
Elman-RPROP	1.23E-03	2.46E-03	1.54	1.49E-04	2.46E-04	1.84	2.41E-04	4.04E-04	1.97
	± 1.66E-03	± 3.55E-03	± 0.25	± 3.07E-05	± 3.48E-05	± 0.19	± 6.30E-05	± 8.08E-05	± 0.29
Elman-PSO	1.65E-03	1.80E-03	1.08	1.48E-03	2.15E-03	1.49	1.90E-03	2.98E-03	1.56
	± 2.62E-04	± 2.85E-04	± 0.25	± 2.19E-04	± 2.88E-04	± 0.19	± 1.74E-04	± 2.95E-04	± 0.29
Elman-CQPSO	7.21E-04	8.08E-04	1.17	2.31E-03	2.02E-03	0.86	1.81E-03	2.90E-03	1.60
	± 3.32E-04	± 3.73E-04	± 0.08	± 2.97E-04	± 3.08E-04	± 0.03	± 1.88E-04	± 3.10E-04	± 0.01
Jordan-RPROP	2.30E-01	2.37E-01	1.14	7.04E-02	7.42E-02	1.15	2.07E-02	1.98E-02	1.09
.	± 1.56E-01	± 1.64E-01	± 0.06	± 1.08E-01	± 1.14E-01	± 0.04	± 2.72E-02	± 2.49E-02	± 0.06
Jordan-PSO	2.10E-03	2.27E-03	1.08	2.41E-03	3.45E-03	1.51	1.89E-03	3.70E-03	2.03
	± 4.06E-04	± 4.36E-04	± 0.02	± 5.07E-04	± 6.33E-04	± 0.07	± 3.40E-04	± 5.87E-04	± 0.06
Jordan-CQPSO	1.32E-04	1.58E-04	1.24	1.42E-04	1.96E-04	1.52	1.51E-04	2.56E-04	1.57
	± 7.60E-05	± 8.97E-05	± 0.11	± 4.68E-05	± 5.12E-05	± 0.14	± 6.37E-05	± 1.32E-04	± 0.21
MRNN-RPROP	8.98E-03	9.64E-03	1.15	1.40E-02	1.54E-02	1.22	1.56E-02	1.72E-02	1.36
	± 7.98E-03	± 8.26E-03	± 0.05	± 7.60E-03	± 8.39E-03	± 0.13	± 1.63E-02	± 1.78E-02	± 0.19
MRNN-PSO	1.90E-03	2.07E-03	1.08	1.67E-03	2.40E-03	1.46	1.55E-03	3.04E-03	1.99
	± 2.37E-04	± 2.61E-04	± 0.01	± 2.37E-04	± 3.29E-04	± 0.06	± 2.34E-04	± 4.23E-04	± 0.03
MRNN-CQPSO	7.02E-04	8.01E-04	1.18	8.66E-04	1.18E-03	1.44	6.99E-04	1.42E-03	2.11
	± 2.15E-04	± 2.44E-04	± 0.06	± 3.37E-04	± 4.49E-04	± 0.08	± 2.61E-04	± 4.83E-04	± 0.04
TDNN-RPROP	2.03E-04	5.50E-04	2.41	4.02E-04	7.86E-04	2.08	4.50E-04	8.71E-04	2.00
	± 5.61E-05	± 2.76E-04	± 0.25	± 9.27E-05	± 1.66E-04	± 0.18	± 9.44E-05	± 2.42E-04	± 0.26
TDNN-PSO	1.81E-03	1.99E-03	1.10	1.69E-03	2.50E-03	1.51	1.69E-03	3.27E-03	1.98
	± 2.72E-04	± 2.96E-04	± 0.01	± 2.19E-04	± 3.05E-04	± 0.06	± 2.87E-04	± 5.12E-04	± 0.04
TDNN-CQPSO	2.15E-03	2.50E-03	1.74	1.96E-03	2.34E-03	1.49	2.38E-03	4.21E-03	1.77
	± 1.62E-03	± 1.81E-03	± 0.32	± 7.46E-04	± 8.36E-04	± 0.19	± 1.45E-03	± 2.50E-03	± 0.13

CQSO, TDNN-Rprop and Elman-PSO. As visualized in Figure 7.19a, the peak of the models' T_E increased after every change without recovering, except for the TDNN-Rprop which immediately recovered from the changes, tracking the moving lowest value. The peak of the models' G_E also increased after the changes, but were smaller compared to



(a) T_E (b) G_E

Figure 7.19: Training and generalization error results for IAP time series, scenario C1

Table 7.41: Models ranking in forecasting the IAP time series, scenario C1-C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1	1	1	1	1	1	1	1
Elman-RPROP	6	10	3	3	3	3	4	5.33
Elman-PSO	7	6	6	7	10	7	7.67	6.67
Elman-CQSO	5	5	10	6	8	6	7.67	5.67
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	10	9	11	11	9	10	10	10
Jordan-CQSO	2	2	2	2	2	2	2	2
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	9	8	7	9	6	8	7.33	8.33
MRNN-CQSO	4	4	5	5	5	5	4.67	4.67
TDNN-RPROP	3	3	4	4	4	4	3.67	3.67
TDNN-PSO	8	7	8	10	7	9	7.67	8.67
TDNN-CQSO	11	11	9	8	11	11	10.33	10

the T_E peaks. Figure 7.19b shows that the FNN-CQSO generalized better than the other models by producing a lower G_E throughout the search process.

The overall ranking of the models presented in Table 7.42 shows that the FNN-CQSO achieved the highest average rank. Thus, the FNN-CQSO emerged as the overall winner. This indicates that a FNN trained with CQSO is sufficient in forecasting the AIP problem

Table 7.42: Overall models ranking in forecasting the IAP time series, scenario error A-C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1.67	1.67	1	1	1.33	1	1	1.22
Elman-RPROP	2	2	2.67	2.67	4	5.33	2.89	3.33
Elman-PSO	7.33	8.33	8	8.33	7.67	6.67	7.67	7.78
Elman-CQSO	8	6.33	7	7	7.67	5.67	7.56	6.33
Jordan-RPROP	11.33	10	12.67	12.67	13	13	12.33	11.89
Jordan-PSO	9.33	10	10.67	10.67	10	10	10.00	10.22
Jordan-CQSO	2.33	2.33	2.33	2.33	1.67	2	2.11	2.22
MRNN-RPROP	12.67	12.67	12.33	12.33	12	12	12.33	12.33
MRNN-PSO	6.33	7.33	7	7	7.33	8.33	6.89	7.55
MRNN-CQSO	5	5	5.67	5.33	4.67	4.67	5.11	5.00
TDNN-RPROP	4	4	4	4.33	3.67	3.67	3.89	4.00
TDNN-PSO	9.33	9.67	9.33	9.67	7.67	8.67	8.78	9.34
TDNN-CQSO	11.67	11.67	8.333	7.67	10.33	10	10.11	9.78

for the nine scenarios, and has performed better than the RNNs trained using either of the RPROP, PSO or CQSO algorithms.

7.2.7 S&P Time Series

Scenarios A1 to A3: The values in Table 7.43 show that the TDNN-RPROP and the FNN-CQSO models produced the lowest training and generalization errors for scenario A1. For scenario A2, where the spatial changes are more severe compared to scenario A1, the TDNN-RPROP model outperformed all the other models by yielding the lowest errors. For scenario A3, where the spatial changes are abrupt, the FNN-CQSO yielded the lowest errors. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold, except for a few cases. The exceptions are the FNN-CQSO vs Jordan-CQSO comparison where the two models produced statistically similar errors for the three scenarios, and the FNN-CQSO vs Elman-RPROP comparison where the two models produced similar T_E and G_E values

Table 7.43: Results of S&P time series, scenario A1 to A3

Scenario / **Model**	**A1** *(f:50, s:20)*			**A2** *(f:50, s:40)*			**A3** *(f:50, s:58)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	2.47E-04 ± 4.13E-05	3.78E-04 ± 4.49E-05	1.70 ± 0.16	4.18E-04 ± 5.20E-05	6.71E-04 ± 7.54E-05	1.63 ± 0.03	1.83E-04 ± 5.98E-05	2.77E-04 ± 5.92E-05	2.36 ± 0.52
Elman-RPROP	3.44E-04 ± 8.11E-05	5.10E-04 ± 7.98E-05	1.77 ± 0.21	1.29E-03 ± 1.01E-03	1.46E-03 ± 9.30E-04	1.53 ± 0.16	1.21E-03 ± 1.32E-03	1.50E-03 ± 1.44E-03	1.51 ± 0.11
Elman-PSO	8.06E-04 ± 1.06E-04	9.82E-04 ± 1.15E-04	1.24 ± 0.21	9.68E-04 ± 8.32E-05	1.37E-03 ± 1.14E-04	1.42 ± 0.16	7.45E-04 ± 1.21E-04	7.82E-04 ± 1.03E-04	1.15 ± 0.11
Elman-CQPSO	4.81E-04 ± 7.98E-05	6.37E-04 ± 8.88E-05	1.42 ± 0.10	7.81E-04 ± 7.84E-05	1.13E-03 ± 1.03E-04	1.47 ± 0.05	4.53E-04 ± 1.23E-04	5.58E-04 ± 1.11E-04	1.67 ± 0.34
Jordan-RPROP	1.53E-01 ± 1.77E-01	1.58E-01 ± 1.83E-01	1.42 ± 0.16	2.51E-03 ± 1.90E-03	2.67E-03 ± 1.82E-03	1.40 ± 0.12	4.73E-02 ± 7.47E-02	4.52E-02 ± 6.95E-02	1.27 ± 0.08
Jordan-PSO	7.76E-04 ± 9.42E-05	9.52E-04 ± 1.06E-04	1.25 ± 0.03	8.35E-04 ± 7.25E-05	1.22E-03 ± 9.90E-05	1.47 ± 0.02	6.14E-04 ± 1.16E-04	7.16E-04 ± 1.10E-04	1.31 ± 0.16
Jordan-CQPSO	2.55E-04 ± 4.86E-05	3.93E-04 ± 5.42E-05	1.74 ± 0.17	3.86E-04 ± 6.06E-05	6.32E-04 ± 8.77E-05	1.71 ± 0.08	1.85E-04 ± 6.82E-05	2.88E-04 ± 6.83E-05	2.43 ± 0.50
MRNN-RPROP	8.28E-03 ± 4.30E-03	8.75E-03 ± 4.31E-03	1.31 ± 0.16	1.25E-02 ± 4.37E-03	1.33E-02 ± 4.70E-03	1.20 ± 0.13	1.43E-02 ± 6.82E-03	1.62E-02 ± 8.38E-03	1.26 ± 0.14
MRNN-PSO	1.03E-03 ± 1.04E-04	1.23E-03 ± 1.15E-04	1.20 ± 0.01	9.77E-04 ± 7.62E-05	1.38E-03 ± 1.03E-04	1.41 ± 0.02	6.56E-04 ± 1.15E-04	7.02E-04 ± 1.03E-04	1.18 ± 0.12
MRNN-CQPSO	5.19E-04 ± 8.82E-05	6.85E-04 ± 1.00E-04	1.40 ± 0.09	8.60E-04 ± 2.93E-04	1.33E-03 ± 4.87E-04	1.53 ± 0.03	4.06E-04 ± 1.31E-04	4.87E-04 ± 8.02E-05	1.84 ± 0.40
TDNN-RPROP	1.79E-04 ± 3.42E-05	5.04E-04 ± 5.89E-05	3.04 ± 0.21	3.30E-04 ± 6.81E-05	5.68E-04 ± 8.00E-05	1.87 ± 0.14	4.92E-04 ± 1.29E-04	6.16E-04 ± 1.39E-04	1.34 ± 0.10
TDNN-PSO	1.20E-03 ± 1.04E-04	1.43E-03 ± 1.13E-04	1.20 ± 0.01	1.10E-03 ± 8.22E-05	1.61E-03 ± 1.16E-04	1.46 ± 0.02	1.06E-03 ± 1.06E-04	1.05E-03 ± 8.29E-05	1.01 ± 0.04
TDNN-CQPSO	8.76E-04 ± 7.00E-05	1.06E-03 ± 7.62E-05	1.22 ± 0.01	7.53E-04 ± 4.90E-05	1.21E-03 ± 7.52E-05	1.61 ± 0.02	7.62E-04 ± 1.27E-04	7.79E-04 ± 9.33E-05	1.16 ± 0.17

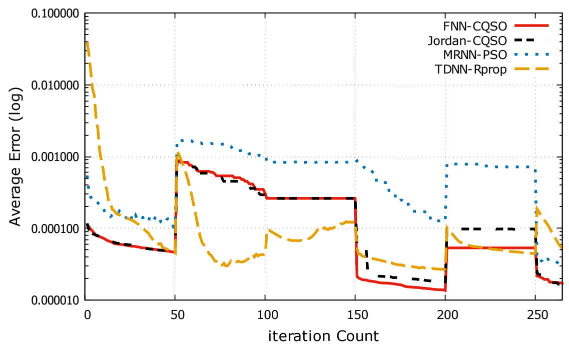
for scenario A1, and a similar G_E for scenario A2.

Figure 7.20 shows the performance progression over time achieved by the FNN-CQSO and three other top performing models (which includes the Jordan-CQSO, TDNN-RPROP and MRNN-PSO). As visualized in the figure, the peak of the T_E for the four

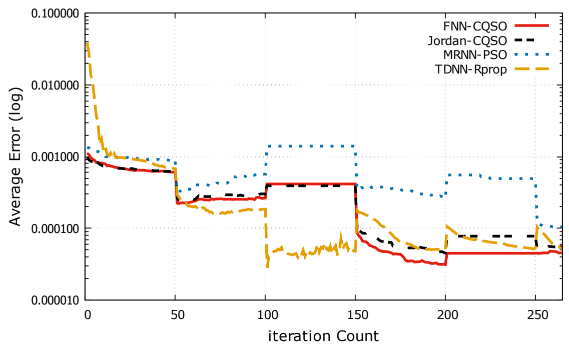
Table 7.44: Models ranking in forecasting the S&P time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	2	1	3	3	1	1	2	1.67
Elman-RPROP	4	4	11	10	11	11	8.67	8.33
Elman-PSO	8	8	8	8	8	9	8	8.33
Elman-CQPSO	5	5	5	4	4	4	4.67	4.33
Jordan-RPROP	13	13	12	12	13	13	12.67	12.67
Jordan-PSO	7	7	6	6	6	7	6.33	6.67
Jordan-CQPSO	3	2	2	2	2	2	2.33	2
MRNN-RPROP	12	12	13	13	12	12	12.33	12.33
MRNN-PSO	10	10	9	9	7	6	8.67	8.33
MRNN-CQPSO	6	6	7	7	3	3	5.33	5.33
TDNN-RPROP	1	3	1	1	5	5	2.33	3
TDNN-PSO	11	11	10	11	10	10	10.33	10.67
TDNN-CQPSO	9	9	4	5	9	8	7.33	7.33

models increased after the first environmental change, and the TDNN-RPROP recovered faster than the other models. However, for the third and subsequent changes, the FNN-CQSO produced better performance. Figure 7.20b shows that, on an average, the FNN-CQSO had a better G_E progression throughout the experiment runs compared to



(a) T_E



(b) G_E

Figure 7.20: Training and generalization error results for SP time series, scenario A3

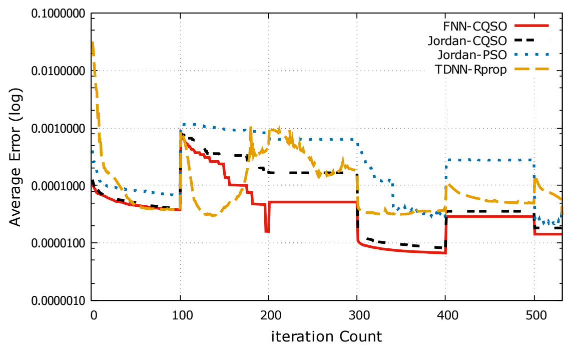
the other models.

The ρ values in Table 7.43 indicate that all the models overfitted. The performance ranking of the models given in Table 7.44 shows that the FNN-CQSO emerged as the winner, by achieving the highest average rank.

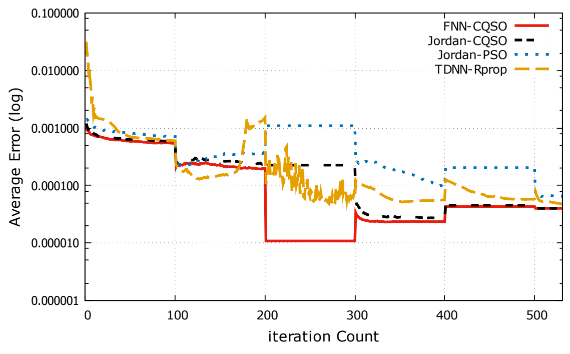
Scenarios B1 to B3: The cumulative mean training and generalization error values in Table 7.45 show that the FNN-CQSO outperformed the other models for scenario B1. For scenario B2, the TDNN-CQSO model yielded the lowest training and generalization errors compared to the other models. For scenario B3, the FNN-CQSO model yielded the best errors compared to the remaining models. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were less than the 0.0001 threshold, except for a few cases. The exceptions are the FNN-CQSO vs Jordan-CQSO comparison for scenarios B1 and B2, and the FNN-CQSO vs Elman-RPROP and FNN-CQSO vs TDNN-RPROP comparisons for scenario B2, where the models produced statistically similar results.

The performance ranking given in Table 7.46 shows that the TDNN-CQSO and the FNN-CQSO achieved the first and second highest ranking positions, respectively.

Figure 7.21 shows the performance progression over time achieved by the FNN-CQSO and three other best models (which includes the Jordan-CQSO, TDNN-RPROP and Jordan-PSO) in predicting the S&P time series for scenario B3. The FNN-CQSO had superior T_E and G_E progressions throughout the search, compared to the other models.



(a) T_E



(b) G_E

Figure 7.21: Training and generalization error results for SP time series, scenario B3

Table 7.45: Results for S&P time series, scenarios B1-B3

Scenario / Model	**B1** *(f:100, s:20)*			**B2** *(f:100, s:40)*			**B3** *(f:100, s:58)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	8.82E-05 ± 1.09E-05	2.21E-04 ± 1.31E-05	2.66 ± 0.18	2.34E-04 ± 4.20E-05	4.17E-04 ± 5.89E-05	1.94 ± 0.13	6.91E-05 ± 1.75E-05	1.75E-04 ± 8.25E-06	3.67 ± 0.66
Elman-RPROP	2.47E-04 ± 9.58E-05	4.28E-04 ± 1.11E-04	2.30 ± 0.31	1.59E-03 ± 2.51E-03	1.78E-03 ± 2.52E-03	1.77 ± 0.17	3.75E-04 ± 9.93E-05	5.02E-04 ± 1.03E-04	1.56 ± 0.16
Elman-PSO	6.94E-04 ± 8.53E-05	8.55E-04 ± 9.36E-05	1.25 ± 0.31	6.13E-04 ± 8.17E-05	9.17E-04 ± 1.10E-04	1.53 ± 0.17	4.56E-04 ± 9.57E-05	5.64E-04 ± 8.26E-05	1.55 ± 0.16
Elman-CQPSO	3.12E-04 ± 6.37E-05	4.84E-04 ± 6.81E-05	1.74 ± 0.13	5.71E-04 ± 1.00E-04	8.77E-04 ± 1.47E-04	1.56 ± 0.05	2.75E-04 ± 1.01E-04	3.87E-04 ± 9.77E-05	2.12 ± 0.44
Jordan-RPROP	7.97E-02 ± 1.06E-01	7.98E-02 ± 1.06E-01	1.49 ± 0.20	7.01E-02 ± 9.31E-02	6.77E-02 ± 8.93E-02	1.36 ± 0.10	4.63E-02 ± 8.57E-02	4.70E-02 ± 8.67E-02	1.37 ± 0.12
Jordan-PSO	4.55E-04 ± 8.23E-05	5.98E-04 ± 9.22E-05	1.37 ± 0.04	5.86E-04 ± 5.37E-05	8.82E-04 ± 7.28E-05	1.52 ± 0.02	3.97E-04 ± 8.67E-05	5.01E-04 ± 8.45E-05	1.45 ± 0.18
Jordan-CQPSO	1.16E-04 ± 3.56E-05	2.61E-04 ± 4.01E-05	2.57 ± 0.18	3.00E-04 ± 5.45E-05	5.07E-04 ± 7.62E-05	1.83 ± 0.13	1.26E-04 ± 4.72E-05	2.34E-04 ± 4.10E-05	3.09 ± 0.62
MRNN-RPROP	5.64E-03 ± 2.83E-03	6.80E-03 ± 3.11E-03	1.49 ± 0.23	6.03E-02 ± 1.08E-01	6.12E-02 ± 1.09E-01	1.26 ± 0.14	1.08E-02 ± 5.05E-03	1.17E-02 ± 5.24E-03	1.22 ± 0.12
MRNN-PSO	7.16E-04 ± 1.05E-04	8.86E-04 ± 1.15E-04	1.27 ± 0.03	6.71E-04 ± 7.29E-05	9.93E-04 ± 9.50E-05	1.50 ± 0.03	5.38E-04 ± 9.70E-05	6.13E-04 ± 9.04E-05	1.24 ± 0.14
MRNN-CQPSO	3.14E-04 ± 7.17E-05	4.68E-04 ± 7.80E-05	1.69 ± 0.14	4.66E-04 ± 7.40E-05	7.31E-04 ± 1.12E-04	1.61 ± 0.08	2.83E-04 ± 9.28E-05	4.01E-04 ± 8.88E-05	2.12 ± 0.45
TDNN-RPROP	1.15E-04 ± 1.59E-05	4.02E-04 ± 3.57E-05	3.74 ± 0.30	1.91E-04 ± 3.47E-05	3.96E-04 ± 4.28E-05	2.27 ± 0.19	3.11E-04 ± 1.37E-04	4.51E-04 ± 1.31E-04	1.73 ± 0.15
TDNN-PSO	1.01E-03 ± 8.09E-05	1.21E-03 ± 8.69E-05	1.20 ± 0.01	9.40E-04 ± 8.27E-05	1.41E-03 ± 1.11E-04	1.51 ± 0.03	1.10E-03 ± 1.35E-04	9.33E-04 ± 8.15E-05	0.89 ± 0.05
TDNN-CQPSO	6.48E-04 ± 9.41E-05	8.35E-04 ± 9.33E-05	1.41 ± 0.15	6.83E-04 ± 4.53E-05	1.12E-03 ± 6.87E-05	1.64 ± 0.01	4.74E-04 ± 9.54E-05	5.27E-04 ± 1.08E-04	1.18 ± 0.15

This indicates that the FNN-CQSO model handled the abrupt scenario B3 better than the RNNs trained with any of the three training algorithms.

The ρ values obtained by all the models show that all the models overfitted, except the TDNN-PSO for scenario B3. Table 7.46 shows that the FNN-CQSO emerged as the

Table 7.46: Models ranking in forecasting the S&P time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	2	2	1	1	1.33	1.33
Elman-RPROP	4	4	11	11	6	7	7	7.33
Elman-PSO	9	9	7	7	8	9	8	8.33
Elman-CQPSO	5	6	5	5	3	3	4.33	4.67
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	7	7	6	6	7	6	6.67	6.33
Jordan-CQPSO	3	2	3	3	2	2	2.67	2.33
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	10	10	8	8	10	10	9.33	9.33
MRNN-CQPSO	6	5	4	4	4	4	4.67	4.33
TDNN-RPROP	2	3	1	1	5	5	2.67	3
TDNN-PSO	11	11	10	10	11	11	10.67	10.67
TDNN-CQPSO	8	8	9	9	9	8	8.67	8.33

winner by obtaining the highest average rank over the three scenarios.

Scenarios C1 to C3: The results in Table 7.47 show that the FNN-CQSO model obtained the lowest cumulative training and generalization errors compared to all the remaining models. All the p-values for the pairwise comparisons between the FNN-CQSO and each of the other models are below the 0.0001 threshold, except for scenario C3, where the Jordan-CQSO produced statistically similar performance. The performance ranking of the models given in Table 7.48 shows that the FNN-CQSO obtained the highest average rank.

The performance progression over time for the four top performing models for scenario C3 is shown in Figure 7.22. The observations made from the figure are similar to that of Figure 7.21.

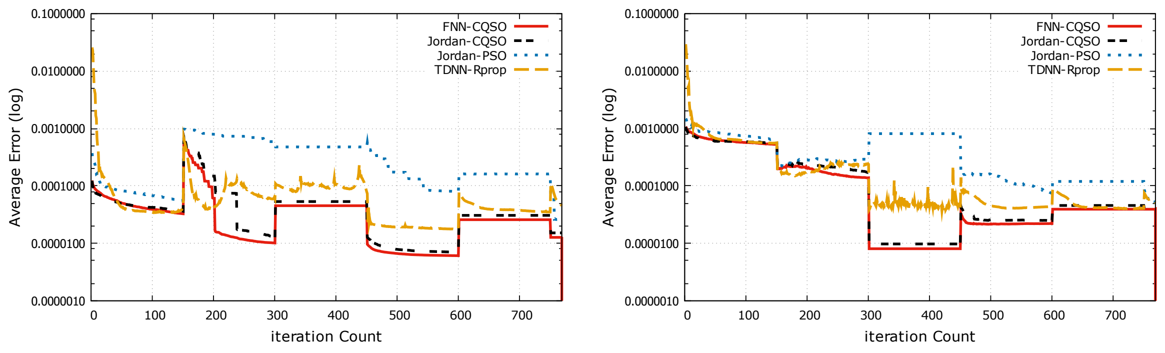
The ρ values given in Table 7.47 indicate that all the models overfitted, except the TDNN-PSO for scenario C3. The TDNN-RPROP model had the worst generalization performance for scenarios C1 and C2, while the FNN-CQSO had the worst generalization

Table 7.47: Results of S&P time series, scenario C1 to C3

Scenario / Model	**C1** *(f:150, s:20)*			**C2** *(f:150, s:40)*			**C3** *(f:150, s:58)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	5.60E-05 ± 2.73E-06	1.85E-04 ± 5.57E-06	3.33 ± 0.09	1.08E-04 ± 2.24E-05	2.45E-04 ± 3.01E-05	2.39 ± 0.11	4.20E-05 ± 7.97E-06	1.73E-04 ± 5.96E-06	4.78 ± 0.53
Elman-RPROP	2.59E-04 ± 1.46E-04	6.06E-04 ± 3.44E-04	2.63 ± 0.26	9.27E-04 ± 1.04E-03	1.11E-03 ± 1.06E-03	1.92 ± 0.23	1.18E-03 ± 1.33E-03	1.32E-03 ± 1.29E-03	1.81 ± 0.18
Elman-PSO	5.65E-04 ± 8.60E-05	7.18E-04 ± 9.61E-05	1.30 ± 0.26	5.97E-04 ± 6.19E-05	8.96E-04 ± 8.20E-05	1.52 ± 0.23	4.77E-04 ± 9.93E-05	5.32E-04 ± 7.90E-05	1.42 ± 0.18
Elman-CQPSO	2.73E-04 ± 8.45E-05	4.10E-04 ± 8.51E-05	1.86 ± 0.20	3.90E-04 ± 6.92E-05	6.28E-04 ± 9.95E-05	1.66 ± 0.05	1.42E-04 ± 4.29E-05	2.57E-04 ± 1.36E-05	2.60 ± 0.46
Jordan-RPROP	6.18E-02 ± 1.19E-01	6.02E-02 ± 1.16E-01	1.74 ± 0.24	2.79E-01 ± 2.28E-01	2.88E-01 ± 2.38E-01	1.31 ± 0.15	1.03E-01 ± 1.42E-01	1.05E-01 ± 1.41E-01	1.44 ± 0.14
Jordan-PSO	4.25E-04 ± 4.36E-05	5.65E-04 ± 4.89E-05	1.35 ± 0.03	4.53E-04 ± 6.26E-05	7.11E-04 ± 8.65E-05	1.59 ± 0.03	3.14E-04 ± 8.90E-05	4.08E-04 ± 7.50E-05	1.89 ± 0.38
Jordan-CQPSO	8.23E-05 ± 2.25E-05	2.17E-04 ± 2.51E-05	2.95 ± 0.19	1.48E-04 ± 3.37E-05	3.01E-04 ± 4.57E-05	2.20 ± 0.13	5.21E-05 ± 1.40E-05	1.77E-04 ± 6.67E-06	4.28 ± 0.53
MRNN-RPROP	3.38E-03 ± 1.38E-03	4.36E-03 ± 2.00E-03	1.41 ± 0.19	8.33E-02 ± 1.49E-01	8.41E-02 ± 1.48E-01	1.38 ± 0.15	7.81E-03 ± 5.40E-03	9.83E-03 ± 6.20E-03	1.43 ± 0.14
MRNN-PSO	6.28E-04 ± 8.52E-05	7.84E-04 ± 9.18E-05	1.27 ± 0.03	5.47E-04 ± 6.85E-05	8.13E-04 ± 8.73E-05	1.52 ± 0.04	4.03E-04 ± 8.82E-05	4.80E-04 ± 7.75E-05	1.38 ± 0.19
MRNN-CQPSO	2.91E-04 ± 8.09E-05	4.46E-04 ± 7.97E-05	1.86 ± 0.19	4.82E-04 ± 9.44E-05	7.55E-04 ± 1.37E-04	1.61 ± 0.05	1.11E-04 ± 2.40E-05	2.51E-04 ± 1.26E-05	2.92 ± 0.47
TDNN-RPROP	8.79E-05 ± 1.28E-05	3.62E-04 ± 3.71E-05	4.35 ± 0.35	1.36E-04 ± 1.92E-05	3.45E-04 ± 2.41E-05	2.70 ± 0.18	1.69E-04 ± 3.04E-05	3.19E-04 ± 3.57E-05	2.05 ± 0.15
TDNN-PSO	9.20E-04 ± 5.93E-05	1.10E-03 ± 6.11E-05	1.20 ± 0.01	8.10E-04 ± 7.33E-05	1.23E-03 ± 7.81E-05	1.54 ± 0.04	1.11E-03 ± 7.77E-05	9.11E-04 ± 4.76E-05	0.83 ± 0.03
TDNN-CQPSO	6.36E-04 ± 7.79E-05	8.05E-04 ± 7.56E-05	1.38 ± 0.14	6.15E-04 ± 3.00E-05	1.02E-03 ± 4.86E-05	1.65 ± 0.01	2.15E-04 ± 4.87E-05	2.66E-04 ± 4.07E-05	1.55 ± 0.27

performance for scenario C3.

The overall performance ranking of the models for all nine scenarios, given in Table 7.49, shows that the FNN-CQSO achieved the first ranking position in predicting the S&P time series.



(a) T_E (b) G_E

Figure 7.22: Training and generalization error results for SP time series, scenario C3

Table 7.48: Models ranking in forecasting the S&P time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	4	7	11	10	11	11	8.67	9.33
Elman-PSO	8	8	8	8	9	9	8.33	8.33
Elman-CQPSO	5	4	4	4	4	4	4.33	4
Jordan-RPROP	13	13	13	13	13	13	13	13
Jordan-PSO	7	6	5	5	7	7	6.33	6
Jordan-CQPSO	2	2	3	2	2	2	2.33	2
MRNN-RPROP	12	12	12	12	12	12	12	12
MRNN-PSO	9	9	7	7	8	8	8	8
MRNN-CQPSO	6	5	6	6	3	3	5	4.67
TDNN-RPROP	3	3	2	3	5	6	3.33	4
TDNN-PSO	11	11	10	11	10	10	10.33	10.67
TDNN-CQPSO	10	10	9	9	6	5	8.33	8

7.2.8 AWS Time Series

Scenarios A1 to A3: The CMF T_E and G_E values given in Table 7.50 indicate that the FNN-CQSO outperformed the other models for scenarios A1 and A2. For scenario A3, the TDNN-CQSO and MRNN-CQSO models obtained the lowest T_E and G_E values,

Table 7.49: Overall models average ranking in forecasting the S&P time series, scenarios A to C

Model	A		B		C		Overall Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	2	1.67	1.33	1.33	1	1	1.44	1.33
Elman-RPROP	8.67	8.33	7	7.33	8.67	9.33	8.11	8.33
Elman-PSO	8	8.33	8	8.33	8.33	8.33	8.11	8.33
Elman-CQPSO	4.67	4.33	4.33	4.67	4.33	4	4.44	4.33
Jordan-RPROP	12.67	12.67	13	13	13	13	12.89	12.89
Jordan-PSO	6.33	6.67	6.67	6.33	6.33	6	6.44	6.33
Jordan-CQPSO	2.33	2	2.67	2.33	2.33	2	2.44	2.11
MRNN-RPROP	12.33	12.33	12	12	12	12	12.11	12.11
MRNN-PSO	8.67	8.33	9.33	9.33	8	8	8.67	8.55
MRNN-CQPSO	5.33	5.33	4.67	4.33	5	4.67	5	4.78
TDNN-RPROP	2.33	3	2.67	3	3.33	4	2.78	3.33
TDNN-PSO	10.33	10.67	10.67	10.67	10.33	10.67	10.44	10.67
TDNN-CQPSO	7.33	7.33	8.67	8.33	8.33	8	8.11	7.89

respectively. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models are below the 0.0001 threshold, except for a few cases. The exceptions for scenario A1 are the FNN-CQSO vs Elman-RPROP and the FNN-CQSO vs Elman-CQSO comparisons, where the difference in G_E values were statistically insignificant. For scenario A2, the FNN-CQSO produced a statistically similar G_E compared to each one of the CQSO trained models. For scenario A3, the exceptions are the FNN-CQSO vs MRNN-CQSO and the FNN-CQSO vs TDNN-CQSO comparisons, where the models also produced similar G_E.

The ρ values in Table 7.50 show that all the models overfitted.

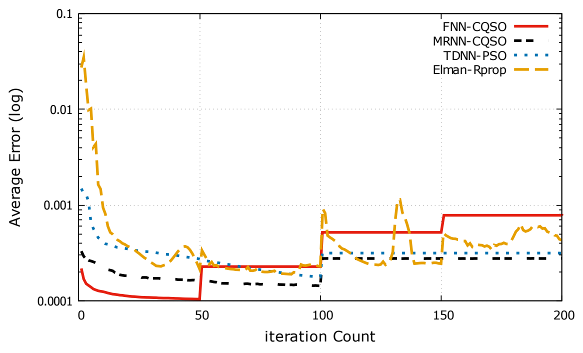
Figure 7.23 shows the performance of the four top performing models for scenario A3. The figure shows that the FNN-CQSO had the best initial T_E progression, but subsequently produced the worst performance throughout the remaining epochs, where the peak increased after every change. The MRNN-CQSO had, on average, the best T_E progression throughout the search. For the G_E, however, the FNN-CQSO had similar or better performance compared to the other models, adapting well to the changes, until

Table 7.50: Results of AWS time series, scenarios A1 to A3

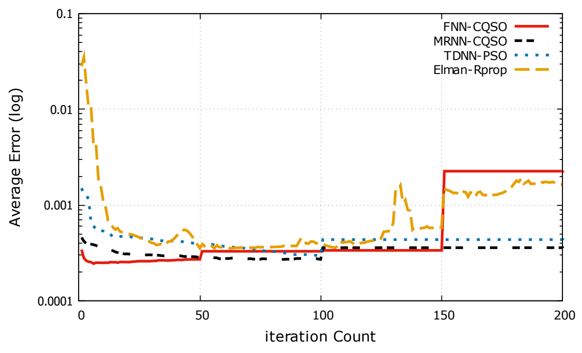
Model	**A1** *(f:50, s:20)*			**A2** *(f:50, s:35)*			**A3** *(f:50, s:42)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQSO	3.92E-04 ± 2.29E-05	6.71E-04 ± 5.27E-05	1.70 ± 0.04	3.83E-04 ± 2.29E-05	5.98E-04 ± 5.79E-05	1.54 ± 0.07	4.12E-04 ± 3.21E-05	8.06E-04 ± 8.04E-05	1.94 ± 0.05
Elman-RPROP	5.71E-04 ± 1.10E-04	7.27E-04 ± 1.07E-04	1.33 ± 0.07	7.49E-04 ± 1.44E-04	1.08E-03 ± 1.43E-04	1.55 ± 0.12	8.96E-04 ± 1.49E-04	1.29E-03 ± 1.60E-04	1.53 ± 0.10
Elman-PSO	5.98E-04 ± 2.78E-05	9.71E-04 ± 5.66E-05	1.63 ± 0.07	7.09E-04 ± 1.94E-05	7.25E-04 ± 3.24E-05	1.02 ± 0.12	6.89E-04 ± 2.37E-05	1.09E-03 ± 5.76E-05	1.58 ± 0.10
Elman-CQSO	4.31E-04 ± 2.37E-05	7.38E-04 ± 6.53E-05	1.70 ± 0.08	4.81E-04 ± 2.00E-05	6.65E-04 ± 5.37E-05	1.38 ± 0.09	4.89E-04 ± 4.14E-05	9.19E-04 ± 1.16E-04	1.84 ± 0.10
Jordan-RPROP	5.61E-03 ± 2.29E-03	6.08E-03 ± 2.49E-03	1.15 ± 0.05	2.80E-02 ± 3.87E-02	2.80E-02 ± 3.73E-02	1.27 ± 0.08	8.45E-02 ± 1.19E-01	8.85E-02 ± 1.19E-01	1.33 ± 0.10
Jordan-PSO	5.40E-04 ± 1.78E-05	9.03E-04 ± 5.65E-05	1.67 ± 0.07	6.88E-04 ± 2.34E-05	7.36E-04 ± 3.95E-05	1.07 ± 0.04	6.77E-04 ± 2.49E-05	1.14E-03 ± 6.49E-05	1.68 ± 0.06
Jordan-CQSO	4.42E-04 ± 2.74E-05	7.71E-04 ± 7.33E-05	1.72 ± 0.07	4.46E-04 ± 2.72E-05	6.36E-04 ± 6.07E-05	1.41 ± 0.08	4.79E-04 ± 3.78E-05	9.20E-04 ± 1.08E-04	1.89 ± 0.08
MRNN-RPROP	1.01E-02 ± 6.75E-03	1.04E-02 ± 6.59E-03	1.12 ± 0.05	3.09E-02 ± 2.19E-02	3.37E-02 ± 2.29E-02	1.16 ± 0.08	1.60E-02 ± 9.13E-03	1.63E-02 ± 9.16E-03	1.14 ± 0.06
MRNN-PSO	5.70E-04 ± 2.99E-05	9.58E-04 ± 5.09E-05	1.69 ± 0.06	6.70E-04 ± 2.69E-05	6.96E-04 ± 3.13E-05	1.04 ± 0.04	3.56E-04 ± 1.59E-05	4.29E-04 ± 1.54E-05	1.22 ± 0.05
MRNN-CQSO	4.87E-04 ± 2.82E-05	8.70E-04 ± 7.97E-05	1.78 ± 0.10	5.03E-04 ± 3.17E-05	6.14E-04 ± 6.30E-05	1.21 ± 0.07	2.25E-04 ± 9.17E-06	3.28E-04 ± 1.41E-05	1.47 ± 0.06
TDNN-RPROP	4.11E-03 ± 1.32E-03	5.91E-03 ± 1.78E-03	1.43 ± 0.12	6.10E-02 ± 6.15E-02	6.69E-02 ± 6.63E-02	1.13 ± 0.12	1.74E-03 ± 4.94E-04	1.95E-03 ± 5.70E-04	1.14 ± 0.05
TDNN-PSO	5.15E-04 ± 2.43E-05	9.09E-04 ± 6.87E-05	1.75 ± 0.07	6.52E-04 ± 2.36E-05	6.77E-04 ± 4.16E-05	1.03 ± 0.04	3.16E-04 ± 1.61E-05	4.34E-04 ± 1.62E-05	1.38 ± 0.05
TDNN-CQSO	4.96E-04 ± 3.58E-05	9.13E-04 ± 1.09E-04	1.80 ± 0.12	5.23E-04 ± 3.65E-05	6.06E-04 ± 7.23E-05	1.14 ± 0.07	2.11E-04 ± 8.00E-06	3.32E-04 ± 6.86E-06	1.59 ± 0.05

the last environmental change, where the performance deteriorated.

The ranking of the models in Table 7.51 shows that the FNN-CQSO is superior compared to the other models.



(a) T_E



(b) G_E

Figure 7.23: Training and generalization error results for AWS time series, scenario A3

Table 7.51: Models ranking in forecasting the AWS time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1	1	1	1	5	5	2.33	2.33
Elman-RPROP	9	2	10	10	10	10	9.97	7.3
Elman-PSO	10	10	9	8	9	8	9.33	8.67
Elman-CQSO	2	3	3	5	7	6	4	4.67
Jordan-RPROP	12	12	11	11	13	13	12	12
Jordan-PSO	7	6	8	9	8	9	7.67	8
Jordan-CQSO	3	4	2	4	6	7	3.67	5
MRNN-RPROP	13	13	12	12	12	12	12.3	12.3
MRNN-PSO	8	9	7	7	4	3	6.33	6.33
MRNN-CQSO	4	5	4	3	2	1	3.33	3
TDNN-RPROP	11	11	13	13	11	11	11.67	11.67
TDNN-PSO	6	7	6	6	3	4	5	5.67
TDNN-CQSO	5	8	5	2	1	2	3.67	4

Scenarios B1 to B3: The error values given in Table 7.52 show that the FNN-CQSO outperformed the other twelve models by yielding the lowest errors, except for scenario B2, where the Jordan-CQSO had lowest training error. All the p-values for the Mann Whitney U test between the FNN-CQSO and the other models are less than the 0.0001 threshold, except for the following pairs comparing the G_E values:

Table 7.52: Results of AWS time series, scenarios B1 to B3

Model	**B1** *(f:100, s:20)*			**B2** *(f:100, s:35)*			**B3** *(f:100, s:42)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQSO	4.07E-04	7.05E-04	1.73	3.59E-04	6.06E-04	1.68	4.10E-04	8.35E-04	2.02
	± 1.88E-05	± 4.08E-05	± 0.02	± 1.45E-05	± 3.42E-05	± 0.04	± 2.22E-05	± 5.73E-05	0.03
Elman-RPROP	8.13E-04	1.04E-03	1.39	9.53E-04	1.32E-03	1.7	2.50E-03	3.17E-03	1.68
	± 2.62E-04	± 2.78E-04	± 0.07	± 6.51E-04	± 6.27E-04	± 0.13	± 2.46E-03	± 2.75E-03	± 0.17
Elman-PSO	5.36E-04	8.93E-04	1.67	5.97E-04	6.42E-04	1.08	6.00E-04	1.02E-03	1.69
	± 2.39E-05	± 4.80E-05	± 0.07	± 2.66E-05	± 2.92E-05	± 0.13	± 2.77E-05	± 6.82E-05	± 0.17
Elman-CQSO	4.56E-04	8.03E-04	1.73	4.52E-04	6.28E-04	1.38	5.23E-04	9.80E-04	1.88
	± 3.71E-05	± 9.44E-05	± 0.08	± 2.76E-05	± 6.59E-05	± 0.09	± 1.16E-04	± 1.90E-04	± 0.06
Jordan-RPROP	4.98E-03	5.45E-03	1.2	6.22E-02	5.72E-02	1.42	7.30E-03	8.02E-03	1.27
	± 3.54E-03	± 3.81E-03	± 0.06	± 1.16E-01	± 1.05E-01	± 0.17	± 3.44E-03	± 3.56E-03	± 0.14
Jordan-PSO	5.01E-04	8.53E-04	1.7	5.83E-04	6.23E-04	1.07	5.74E-04	1.01E-03	1.75
	± 1.89E-05	± 5.09E-05	± 0.06	± 2.82E-05	± 3.48E-05	± 0.04	± 2.50E-05	± 5.94E-05	± 0.05
Jordan-CQSO	4.61E-04	8.47E-04	1.82	4.36E-04	6.49E-04	1.48	4.72E-04	9.12E-04	1.91
	± 3.45E-05	± 8.03E-05	± 0.06	± 2.29E-05	± 5.57E-05	± 0.07	± 3.15E-05	± 8.37E-05	± 0.06
MRNN-RPROP	9.47E-03	9.90E-03	1.14	1.32E-02	1.40E-02	1.24	1.79E-02	1.97E-02	1.17
	± 4.56E-03	± 4.55E-03	± 0.04	± 5.08E-03	± 5.27E-03	± 0.13	± 7.26E-03	± 8.23E-03	± 0.10
MRNN-PSO	5.23E-04	8.87E-04	1.69	5.81E-04	6.34E-04	1.09	5.69E-04	9.77E-04	1.72
	± 2.54E-05	± 6.24E-05	± 0.07	± 2.81E-05	± 4.07E-05	± 0.05	± 2.37E-05	± 5.13E-05	± 0.05
MRNN-CQSO	4.65E-04	8.37E-04	1.78	4.64E-04	5.91E-04	1.27	5.04E-04	9.66E-04	1.89
	± 2.49E-05	± 7.99E-05	± 0.09	± 1.91E-05	± 4.38E-05	± 0.07	± 3.80E-05	± 1.11E-04	± 0.09
TDNN-RPROP	1.69E-03	2.45E-03	1.48	4.83E-03	5.71E-03	1.16	2.92E-03	4.45E-03	1.55
	± 5.65E-04	± 8.40E-04	± 0.08	± 2.90E-03	± 3.22E-03	± 0.07	± 9.72E-04	± 1.56E-03	± 0.09
TDNN-PSO	4.76E-04	8.38E-04	1.75	5.82E-04	6.34E-04	1.09	5.41E-04	9.91E-04	1.82
	± 1.84E-05	± 5.61E-05	± 0.06	± 2.17E-05	± 3.52E-05	± 0.04	± 2.34E-05	± 5.94E-05	± 0.05
TDNN-CQSO	4.92E-04	9.62E-04	1.93	5.28E-04	6.51E-04	1.19	5.58E-04	1.15E-03	2.02
	± 3.59E-05	± 9.41E-05	± 0.08	± 4.50E-05	± 1.05E-04	± 0.09	± 5.62E-05	± 1.52E-04	± 0.09

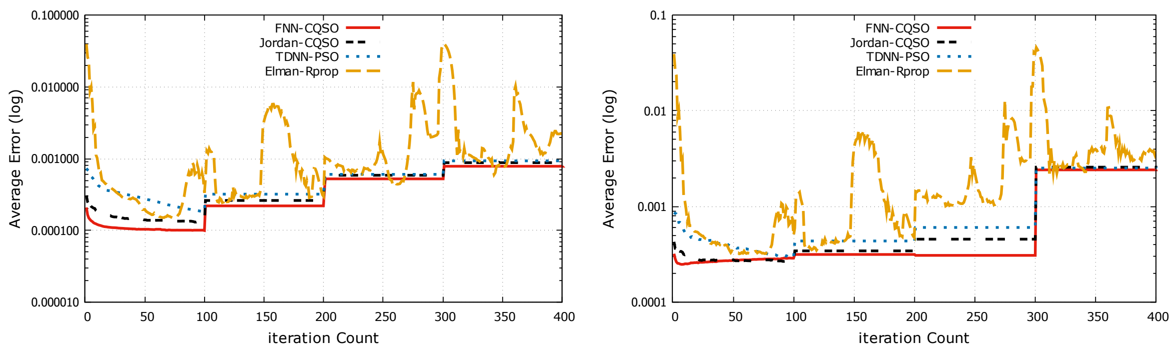
- FNN-CQSO vs Elman-RPROP and FNN-CQSO vs Elman-CQSO for scenario B1,
- FNN-CQSO vs any of the PSO or CQSO trained models for sceanrio B2, and
- FNN-CQSO vs Elman-RPROP, FNN-CQSO vs Elman-CQSO, FNN-CQSO vs

Jordan-CQSO and FNN-CQSO vs MRNN-CQSO for scenario B3.

The ρ values obtained by the models, given in Table 7.52, indicate that all the models overfitted.

Table 7.53: Models ranking in forecasting the AWS time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1	1	1	2	1	1	1	1.33
Elman-RPROP	10	10	10	10	12	12	10.67	10.67
Elman-PSO	9	8	9	7	9	9	9	8
Elman-CQSO	2	2	3	4	4	6	3	4
Jordan-RPROP	12	12	13	13	14	14	13	13
Jordan-PSO	7	6	8	3	8	8	7.67	5.67
Jordan-CQSO	3	5	2	8	2	2	2.33	5
MRNN-RPROP	13	13	12	12	15	15	13.33	13.33
MRNN-PSO	8	7	6	6	7	5	7	6
MRNN-CQSO	4	3	4	1	3	4	3.67	2.67
TDNN-RPROP	11	11	11	11	13	13	11.67	11.67
TDNN-PSO	5	4	7	5	5	7	5.67	5.33
TDNN-CQSO	6	9	5	9	6	10	5.67	9.33



(a) T_E (b) G_E

Figure 7.24: Training and generalization error results for AWS time series, scenario B3

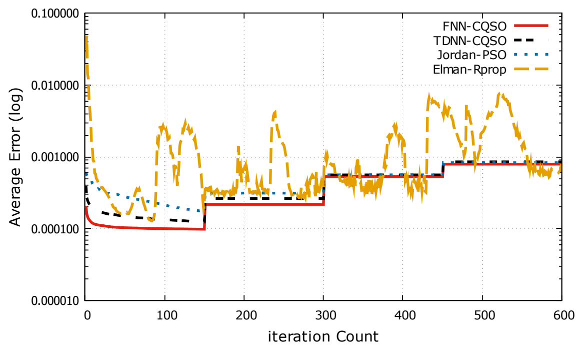
Figure 7.24 illustrates the performance progression over time for the four models that achieved the best results in predicting the AWS time series for scenario B3. The models selected include the FNN-CQSO, Jordan-CQSO, Elman-RPROP and TDNN-PSO. As visualized in Figure 7.24, the FNN-CQSO had the best T_E and G_E progression throughout the search, adapting well to the changes.

The performance ranking in Table 7.53 shows that the FNN-CQSO outperformed all the other models by achieving the highest average performance rank.

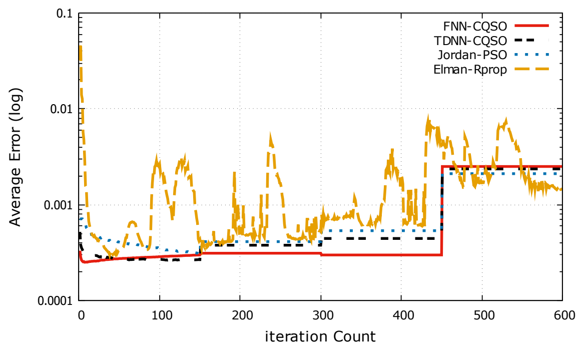
Scenarios C1 to C3: The cumulative mean error values in Table 7.54 show that the FNN-CQSO outperformed the remaining models, except for scenario C2, where the TDNN-CQSO models had the best generalization performance. All the p-values for the pairwise statistical comparisons between FNN-CQSO and each of the other models were less than the 0.0001 threshold, except for the following few cases: FNN-CQSO vs Elman-CQSO for scenario C1, and FNN-CQSO vs any of the RPROP and the CQSO trained models in terms of G_E for scenarios C2 and C3.

The ρ values given in Table 7.54 show that all the models exhibited overfitting behaviours.

Figure 7.25 shows the performance progression over time for the four models that achieved the best results for scenario C3. As visualized in Figure 7.25, while the Elman-RPROP model fluctuated a lot, the other three models had more stable error progression. Figure 7.25a shows that the FNN-CQSO had the best T_E progression throughout the



(a) T_E



(b) G_E

Figure 7.25: Training and generalization error results for AWS time series, scenario C3

Table 7.54: Results of AWS time series, scenarios C1 to C3

Model	**C1** *(f:150, s:20)*			**C2** *(f:150, s:35)*			**C3** *(f:150, s:42)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQSO	3.84E-04 ± 1.35E-05	6.56E-04 ± 3.04E-05	1.70 ± 0.02	3.51E-04 ± 1.40E-05	6.10E-04 ± 2.89E-05	1.73 ± 0.02	4.11E-04 ± 1.43E-05	8.51E-04 ± 3.34E-05	2.07 ± 0.02
Elman-RPROP	9.33E-04 ± 3.03E-04	1.19E-03 ± 3.33E-04	1.41 ± 0.09	8.27E-04 ± 3.49E-04	1.21E-03 ± 3.80E-04	1.82 ± 0.19	1.53E-03 ± 9.33E-04	1.80E-03 ± 7.71E-04	1.71 ± 0.19
Elman-PSO	5.24E-04 ± 2.64E-05	9.19E-04 ± 5.41E-05	1.76 ± 0.09	5.55E-04 ± 2.80E-05	6.26E-04 ± 3.21E-05	1.14 ± 0.19	5.54E-04 ± 3.46E-05	9.37E-04 ± 8.28E-05	1.68 ± 0.19
Elman-CQSO	4.34E-04 ± 3.72E-05	7.73E-04 ± 8.58E-05	1.76 ± 0.06	4.31E-04 ± 2.04E-05	6.28E-04 ± 4.78E-05	1.45 ± 0.08	4.69E-04 ± 3.77E-05	9.29E-04 ± 9.39E-05	1.96 ± 0.05
Jordan-RPROP	2.51E-03 ± 1.60E-03	2.85E-03 ± 1.63E-03	1.35 ± 0.11	1.47E-01 ± 1.64E-01	1.46E-01 ± 1.64E-01	1.32 ± 0.15	5.47E-02 ± 9.59E-02	5.74E-02 ± 1.00E-01	1.27 ± 0.12
Jordan-PSO	4.70E-04 ± 1.78E-05	8.06E-04 ± 5.23E-05	1.71 ± 0.06	5.03E-04 ± 1.85E-05	5.87E-04 ± 3.31E-05	1.17 ± 0.05	4.92E-04 ± 2.06E-05	8.65E-04 ± 5.52E-05	1.75 ± 0.06
Jordan-CQSO	4.54E-04 ± 2.43E-05	8.47E-04 ± 6.31E-05	1.85 ± 0.05	4.24E-04 ± 2.03E-05	6.06E-04 ± 4.82E-05	1.43 ± 0.08	4.93E-04 ± 3.64E-05	9.96E-04 ± 1.01E-04	1.99 ± 0.06
MRNN-RPROP	7.50E-03 ± 2.87E-03	7.72E-03 ± 2.76E-03	1.13 ± 0.06	7.69E-03 ± 2.66E-03	8.32E-03 ± 2.98E-03	1.13 ± 0.10	7.19E-03 ± 3.48E-03	8.38E-03 ± 3.70E-03	1.28 ± 0.16
MRNN-PSO	4.85E-04 ± 1.72E-05	8.57E-04 ± 4.77E-05	1.76 ± 0.06	5.25E-04 ± 2.39E-05	5.97E-04 ± 3.28E-05	1.14 ± 0.05	5.04E-04 ± 2.12E-05	8.66E-04 ± 4.99E-05	1.71 ± 0.05
MRNN-CQSO	4.82E-04 ± 3.11E-05	9.01E-04 ± 8.31E-05	1.85 ± 0.07	5.00E-04 ± 2.89E-05	6.99E-04 ± 6.89E-05	1.39 ± 0.09	5.10E-04 ± 4.12E-05	9.95E-04 ± 1.17E-04	1.92 ± 0.08
TDNN-RPROP	1.28E-03 ± 5.93E-04	1.73E-03 ± 6.29E-04	1.47 ± 0.07	3.51E-03 ± 1.66E-03	4.21E-03 ± 2.23E-03	1.15 ± 0.07	2.12E-03 ± 6.95E-04	2.68E-03 ± 7.22E-04	1.48 ± 0.10
TDNN-PSO	4.51E-04 ± 2.03E-05	8.39E-04 ± 5.08E-05	1.85 ± 0.06	5.32E-04 ± 2.16E-05	6.17E-04 ± 3.51E-05	1.16 ± 0.04	4.81E-04 ± 2.67E-05	8.87E-04 ± 7.45E-05	1.83 ± 0.06
TDNN-CQSO	4.79E-04 ± 5.22E-05	9.14E-04 ± 1.13E-04	1.89 ± 0.07	4.92E-04 ± 2.73E-05	5.70E-04 ± 6.15E-05	1.14 ± 0.08	4.56E-04 ± 4.23E-05	8.68E-04 ± 1.23E-04	1.85 ± 0.10

search. The G_E progression of the models shown in Figure 7.23b indicates that the FNN-CQSO had similar or better performance compared to the other models, adapting well to the changes until the last environmental change, where the FNN-CQSO produced slightly the worst error.

Table 7.55: Models ranking in forecasting the AWS time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1	1	1	5	1	1	1	2.33
Elman-RPROP	10	10	10	10	10	10	10	10
Elman-PSO	9	9	9	7	9	7	9	7.67
Elman-CQSO	2	2	3	8	3	6	2.67	5.33
Jordan-RPROP	12	12	13	13	13	13	12.67	12.67
Jordan-PSO	5	3	6	2	5	2	3.67	2.33
Jordan-CQSO	4	5	2	4	6	9	4	6
MRNN-RPROP	13	13	12	12	12	12	12.33	12.33
MRNN-PSO	8	6	7	3	7	3	7.33	4
MRNN-CQSO	7	7	5	9	8	8	6.67	8
TDNN-RPROP	11	11	11	11	11	11	11	11
TDNN-PSO	3	4	8	6	4	5	5	5
TDNN-CQSO	6	8	4	1	2	4	4	4.33

The performance ranking of the models presented in Table 7.55 indicates that the FNN-CQSO achieved the highest average rank over the three scenarios.

The overall ranking of the models for the nine scenarios, as given in Table 7.56, shows that the FNN-CQSO obtained the highest average rank, and thus emerged as the overall winner in predicting the AWS time series.

7.2.9 USD Time Series

Scenarios A1 to A3: The error values in Table 7.57 show that the Jordan-CQSO and the FNN-CQSO outperformed the other models for scenario A1. A comparison of the best performing models showed that the Jordan-CQSO had the best training performance, while the FNN-CQSO produced the best generalization performance. For scenario A2, the Jordan-CQSO model yielded the lowest errors, outperforming all the other models. The values in Table 7.58 show that the FNN-CQSO achieved the third and fifth ranking positions in terms of training and generalization performance, respectively.

Table 7.56: Overall models' ranking in forecasting the AWS time series, scenarios A to C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	2.33	2.33	1	1.33	1	2.33	1.44	2.00
Elman-RPROP	9.97	7.3	10.67	10.67	10	10	10.21	9.32
Elman-PSO	9.33	8.67	9	8	9	7.67	9.11	8.11
Elman-CQSO	4	4.67	3	4	2.67	5.33	3.22	4.67
Jordan-RPROP	12	12	13	13	12.67	12.67	12.56	12.56
Jordan-PSO	7.67	8	7.67	5.67	3.67	2.33	6.34	5.33
Jordan-CQSO	3.67	5	2.33	5	4	6	3.33	5.33
MRNN-RPROP	12.3	12.3	13.33	13.33	12.33	12.33	12.65	12.65
MRNN-PSO	6.33	6.33	7	6	7.33	4	6.89	5.44
MRNN-CQSO	3.33	3	3.67	2.67	6.67	8	4.56	4.56
TDNN-RPROP	11.67	11.67	11.67	11.67	11	11	11.45	11.45
TDNN-PSO	5	5.67	5.67	5.33	5	5	5.22	5.33
TDNN-CQSO	3.67	4	5.67	9.33	4	4.33	4.45	5.89

For scenario A3, the Jordan-CQSO produced the lowest training error and the FNN-CQSO produced the lowest generalization error. The ranking of the models shows that the FNN-CQSO achieved the first position in terms of generalization, and the fourth in terms of training for scenario C3.

All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were less than the threshold, except for the following: for scenario A1, FNN-CQSO vs Elman-RPROP, FNN-CQSO vs Jordan-RPROP in terms of T_E, and FNN-CQSO vs Jordan-CQSO in terms of G_E; for scenario A2, FNN-CQSO vs Elman-CQSO, FNN-CQSO vs Jordan-CQSO in terms of T_E; and FNN-CQSO vs Elman-CQSO for scenario A3.

The ρ values given in Table 7.57 show that all the models exhibited overfitting behaviour.

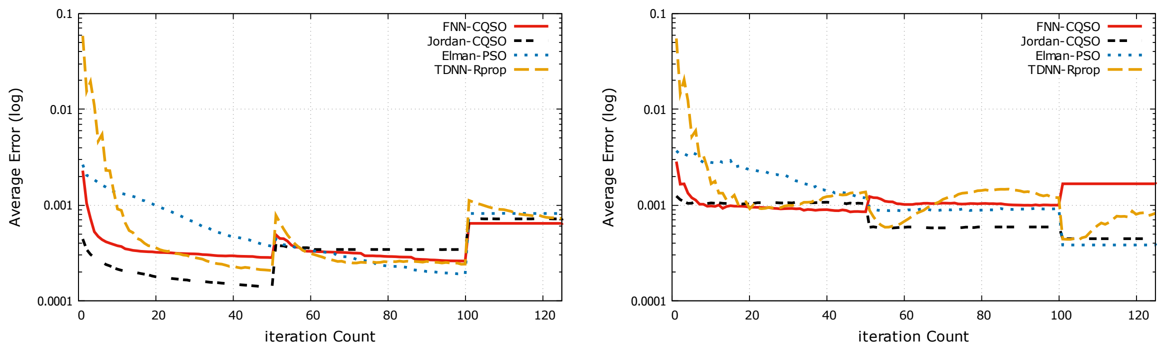
Figure 7.26 illustrates the performance progression over time for the four models that achieved the highest performance rank for scenario A2. As visualized in the figure, the T_E obtained by the models increased after every change. However, the G_E values

Table 7.57: Results of USD time series, scenarios A1 to A3

Scenario / **Model**	**A1** *(f:50, s:8)*			**A2** *(f:50, s:16)*			**A3** *(f:50, s:20)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	4.90E-04 ± 2.37E-05	1.01E-03 ± 5.77E-05	2.11 ± 0.16	4.12E-04 ± 3.64E-05	1.16E-03 ± 4.50E-05	2.93 ± 0.21	5.66E-04 ± 3.08E-05	1.10E-03 ± 4.18E-05	1.95 ± 0.07
Elman-RPROP	1.02E-03 ± 5.95E-04	2.06E-03 ± 6.74E-04	2.91 ± 0.42	1.64E-03 ± 4.07E-04	2.18E-03 ± 4.21E-04	1.53 ± 0.13	1.64E-03 ± 5.84E-04	2.32E-03 ± 6.15E-04	1.68 ± 0.21
Elman-PSO	6.74E-04 ± 3.94E-05	1.52E-03 ± 1.05E-04	2.29 ± 0.42	6.47E-04 ± 3.40E-05	1.30E-03 ± 8.90E-05	2.02 ± 0.13	8.70E-04 ± 5.58E-05	1.59E-03 ± 1.11E-04	1.84 ± 0.21
Elman-CQPSO	5.44E-04 ± 2.98E-05	1.30E-03 ± 1.25E-04	2.39 ± 0.17	3.82E-04 ± 1.85E-05	8.27E-04 ± 4.81E-05	2.19 ± 0.13	5.39E-04 ± 2.33E-05	1.12E-03 ± 4.60E-05	2.11 ± 0.11
Jordan-RPROP	2.75E-03 ± 1.39E-03	3.64E-03 ± 1.59E-03	2.03 ± 0.31	2.00E-03 ± 1.46E-03	2.47E-03 ± 1.41E-03	1.61 ± 0.11	1.22E-03 ± 6.17E-04	1.78E-03 ± 5.66E-04	1.94 ± 0.21
Jordan-PSO	7.48E-04 ± 3.78E-05	1.73E-03 ± 1.07E-04	2.33 ± 0.12	9.08E-04 ± 7.25E-05	1.58E-03 ± 1.21E-04	1.76 ± 0.09	1.03E-03 ± 5.54E-05	1.72E-03 ± 1.10E-04	1.68 ± 0.09
Jordan-CQPSO	4.45E-04 ± 1.81E-05	1.04E-03 ± 9.22E-05	2.34 ± 0.18	3.62E-04 ± 2.24E-05	7.46E-04 ± 2.93E-05	2.1 ± 0.11	4.17E-04 ± 1.58E-05	1.19E-03 ± 3.96E-05	2.89 ± 0.12
MRNN-RPROP	8.52E-03 ± 3.17E-03	9.43E-03 ± 3.22E-03	1.77 ± 0.39	7.66E-03 ± 4.01E-03	8.53E-03 ± 4.22E-03	1.47 ± 0.16	1.34E-02 ± 7.03E-03	1.45E-02 ± 7.39E-03	1.75 ± 0.27
MRNN-PSO	6.97E-04 ± 4.93E-05	1.75E-03 ± 1.45E-04	2.54 ± 0.17	6.67E-04 ± 3.60E-05	1.36E-03 ± 6.15E-05	2.06 ± 0.1	8.55E-04 ± 4.43E-05	1.55E-03 ± 9.20E-05	1.82 ± 0.07
MRNN-CQPSO	6.67E-04 ± 4.54E-05	1.72E-03 ± 1.29E-04	2.59 ± 0.13	4.61E-04 ± 2.69E-05	9.83E-04 ± 6.98E-05	2.14 ± 0.12	6.66E-04 ± 2.95E-05	1.23E-03 ± 6.93E-05	1.85 ± 0.11
TDNN-RPROP	8.46E-04 ± 1.42E-04	2.54E-03 ± 2.08E-04	3.34 ± 0.33	1.36E-03 ± 2.77E-04	1.93E-03 ± 2.95E-04	1.59 ± 0.15	1.04E-03 ± 2.84E-04	1.96E-03 ± 2.80E-04	2.33 ± 0.26
TDNN-PSO	7.46E-04 ± 3.18E-05	2.36E-03 ± 1.24E-04	3.2 ± 0.2	6.47E-04 ± 3.02E-05	1.42E-03 ± 8.30E-05	2.2 ± 0.11	8.53E-04 ± 4.02E-05	1.60E-03 ± 7.61E-05	1.89 ± 0.08
TDNN-CQPSO	6.76E-04 ± 3.25E-05	1.73E-03 ± 1.53E-04	2.54 ± 0.16	4.39E-04 ± 5.79E-05	8.30E-04 ± 3.59E-05	2 ± 0.14	4.94E-04 ± 2.41E-05	1.31E-03 ± 5.24E-05	2.68 ± 0.11

obtained by the models dropped after every change, except for the FNN-CQSO model which had its G_E values increased after the changes. This indicates that the other three models adapted to changes better than the FNN-CQSO model for scenario A2.

The performance ranking of the models given in Table 7.58 show that the Jordan-



(a) T_E (b) G_E

Figure 7.26: Training and generalization error results for USD time series, scenario A2

Table 7.58: Models ranking in forecasting the USD time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	2	1	3	5	4	1	3	2.33
Elman-RPROP	11	9	11	11	12	12	11.33	10.67
Elman-PSO	5	4	6	6	8	7	6.33	5.67
Elman-CQPSO	3	3	2	2	3	2	2.67	2.33
Jordan-RPROP	12	12	12	12	11	10	11.67	11.33
Jordan-PSO	9	7	9	9	9	9	9	8.33
Jordan-CQPSO	1	2	1	1	1	3	1	2
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	7	8	8	7	7	6	7.33	7
MRNN-CQPSO	4	5	5	4	5	4	4.67	4.33
TDNN-RPROP	10	11	10	10	10	11	10	10.67
TDNN-PSO	8	10	7	8	6	8	7	8.67
TDNN-CQPSO	6	6	4	3	2	5	4	4.67

CQSO obtained the highest average rank over the three scenarios. The FNN-CQSO obtained the second highest average rank.

Scenarios B1 to B3: The error values presented in Table 7.59 show that the FNN-CQSO obtained the lowest cumulative mean training and generalisation errors for sce-

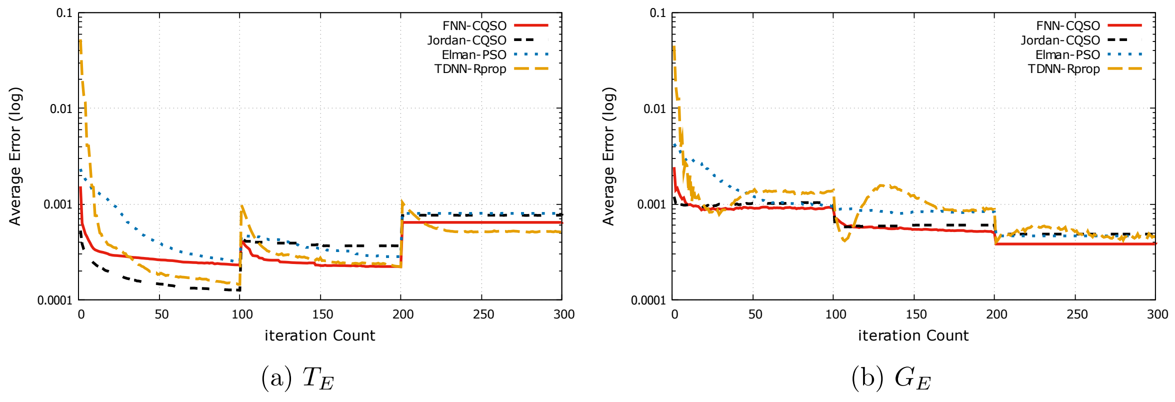
Table 7.59: Results of USD time series, scenarios B1 to B3

Scenario / Model	**B1** *(f:100, s:8)*			**B2** *(f:100, s:16)*			**B3** *(f:100, s:20)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	4.61E-04	9.05E-04	1.98	3.96E-04	6.28E-04	1.59	4.97E-04	1.05E-03	2.13
	± 1.92E-05	± 5.39E-05	± 0.12	± 1.77E-05	± 2.46E-05	± 0.04	± 2.47E-05	± 2.81E-05	± 0.09
Elman-RPROP	7.72E-04	1.73E-03	3.52	1.11E-03	1.55E-03	1.69	8.25E-04	1.45E-03	2.22
	± 2.92E-04	± 2.79E-04	± 0.64	± 5.78E-04	± 6.40E-04	± 0.15	± 2.41E-04	± 1.96E-04	± 0.26
Elman-PSO	6.13E-04	1.33E-03	2.19	5.95E-04	9.75E-04	1.65	6.23E-04	1.26E-03	2.03
	± 2.60E-05	± 8.53E-05	± 0.64	± 3.06E-05	± 5.28E-05	± 0.15	± 2.04E-05	± 4.29E-05	± 0.26
Elman-CQPSO	5.13E-04	1.37E-03	2.65	4.10E-04	7.15E-04	1.75	5.48E-04	1.17E-03	2.17
	± 2.48E-05	± 1.55E-04	± 0.23	± 1.94E-05	± 3.28E-05	± 0.07	± 4.03E-05	± 8.22E-05	± 0.13
Jordan-RPROP	1.71E-03	2.63E-03	2.52	1.74E-03	2.30E-03	1.81	3.35E-03	3.84E-03	1.47
	± 6.83E-04	± 6.86E-04	± 0.45	± 1.03E-03	± 1.12E-03	± 0.16	± 1.06E-03	± 1.09E-03	± 0.22
Jordan-PSO	6.49E-04	1.33E-03	2.10	6.99E-04	1.16E-03	1.67	8.17E-04	1.43E-03	1.77
	± 4.14E-05	± 8.23E-05	± 0.16	± 3.52E-05	± 6.90E-05	± 0.07	± 4.34E-05	± 8.15E-05	± 0.08
Jordan-CQPSO	4.74E-04	1.05E-03	2.21	4.38E-04	6.96E-04	1.61	3.85E-04	1.24E-03	3.22
	± 1.83E-05	± 7.85E-05	± 0.13	± 2.04E-05	± 2.29E-05	± 0.08	± 1.43E-05	± 5.39E-05	± 0.11
MRNN-RPROP	7.51E-03	8.77E-03	1.40	8.76E-03	9.49E-03	1.43	9.78E-03	1.06E-02	1.41
	± 1.80E-03	± 1.74E-03	± 0.22	± 3.92E-03	± 3.84E-03	± 0.18	± 3.71E-03	± 3.86E-03	± 0.30
MRNN-PSO	6.14E-04	1.44E-03	2.35	6.11E-04	1.01E-03	1.67	6.65E-04	1.38E-03	2.09
	± 2.76E-05	± 1.42E-04	± 0.20	± 4.36E-05	± 5.55E-05	± 0.07	± 3.18E-05	± 6.02E-05	± 0.10
MRNN-CQPSO	5.83E-04	1.52E-03	2.64	4.68E-04	7.95E-04	1.71	6.09E-04	1.22E-03	2.03
	± 4.17E-05	± 1.25E-04	± 0.18	± 1.62E-05	± 3.13E-05	± 0.07	± 2.86E-05	± 5.91E-05	± 0.12
TDNN-RPROP	5.61E-04	2.45E-03	4.49	7.39E-04	1.22E-03	1.88	6.63E-04	1.49E-03	2.69
	± 4.76E-05	± 1.59E-04	± 0.28	± 1.47E-04	± 1.10E-04	± 0.19	± 1.34E-04	± 1.30E-04	± 0.34
TDNN-PSO	6.84E-04	2.00E-03	2.91	6.20E-04	1.11E-03	1.80	6.60E-04	1.46E-03	2.22
	± 2.87E-05	± 1.57E-04	± 0.17	± 3.04E-05	± 4.11E-05	± 0.07	± 2.24E-05	± 3.90E-05	± 0.07
TDNN-CQPSO	6.84E-04	2.00E-03	2.91	4.91E-04	7.65E-04	1.58	4.55E-04	1.34E-03	2.98
	± 2.87E-05	± 1.57E-04	± 0.17	± 3.31E-05	± 3.18E-05	± 0.06	± 1.78E-05	± 4.09E-05	± 0.13

narios B1 and B2. For scenario B3, the Jordan-CQSO and the FNN-CQSO yielded the lowest training and generalization errors, respectively. All the p-values for the pair-wise comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold, except when compared to:

- Elman-RPROP and Jordan-CQSO (in terms of T_E) for scenario B1,
- Elman-CQSO for scenario B2, and
- Elman-RPROP, Elman-CQSO, TDNN-RPROP, TDNN-CQSO, Jordan-CQSO (in terms of G_E) for scenario B3.

Figure 7.27 shows the performance progression over time for the four best performing models. Observations made from the figure are similar to that of Figure 7.26. In addition to that, the FNN-CQSO model slightly outperformed the other models. This indicate that the performance of the FNN-CQSO improved compared to how the algorithm fared for scenario A2.



(a) T_E (b) G_E

Figure 7.27: Training and generalization error results for USD time series, scenario B2

The ρ values in Table 7.59 indicate that all the models overfitted, where the MRNN-RPROP model overfitted the least.

The average performance rank obtained by the models show that the FNN-CQSO model emerged as the winner in predicting the USD time series for scenarios B1, B2 and B3.

Scenarios C1 to C3: The error values given in Table 7.61 show that the FNN-CQSO outperformed the other models, except for scenario C3, where the Jordan-CQSO yielded the lowest training error. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were below the 0.0001 threshold, except in terms of the T_E

Table 7.60: Models ranking in forecasting the USD time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	3	1	1.67	1
Elman-RPROP	11	8	11	11	11	9	11	9.33
Elman-PSO	6	3	6	6	6	5	6	4.67
Elman-CQPSO	3	5	2	3	4	2	3	3.33
Jordan-RPROP	12	12	12	12	12	12	12	12
Jordan-PSO	8	4	9	9	10	8	9	7
Jordan-CQPSO	2	2	3	2	1	4	2	2.67
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	7	6	7	7	9	7	7.67	6.67
MRNN-CQPSO	5	7	4	5	5	3	4.67	5
TDNN-RPROP	4	11	10	10	8	11	7.33	10.67
TDNN-PSO	9.5	9.5	8	8	7	10	8.17	9.17
TDNN-CQPSO	9.5	9.5	5	4	2	6	5.5	6.5

of the following: FNN-CQSO vs Elman-RPROP, FNN-CQSO vs Jordan-RPROP, and FNN-CQSO vs TDNN-CQSO for scenario C1; FNN-CQSO vs Jordan-RPROP for scenario C2; FNN-CQSO vs Elman-RPROP, FNN-CQSO vs TDNN-RPROP, FNN-CQSO vs TDNN-CQSO for scenario C3. The only exception in terms of the G_E comparisons is FNN-CQSO vs Jordan-CQSO for scenarios C1 and C3.

All 13 models exhibited overfitting behaviour, as indicated by the ρ values given in Table 7.61. The MRNN-RPROP, however, was the least overfitted model.

Figure 7.28 illustrates the performance progression over time for the four models that achieved best performance in predicting the USD time series for scenario C2. Observations made from the figure are similar to that of Figure 7.27.

The average performance rank obtained by the models for the three scenarios, as shown in Table 7.62, indicates that the FNN-CQSO model emerged as the winner.

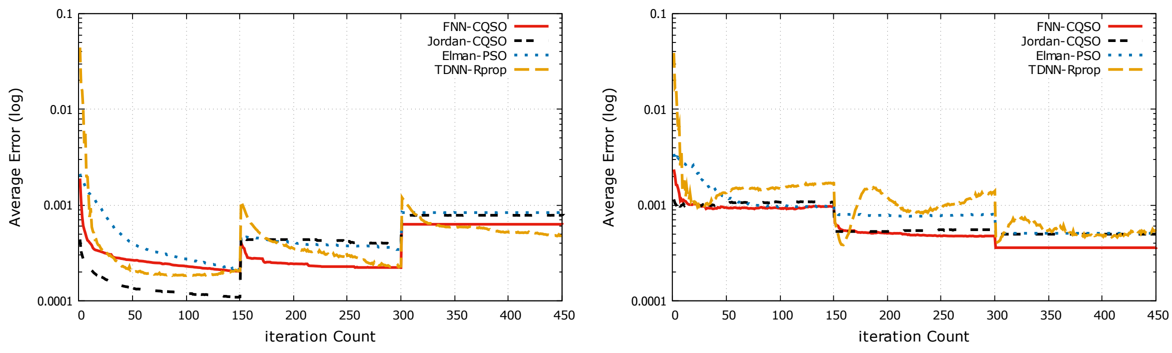
The overall ranking of the models in predicting the USD time series for all nine scenarios, presented in Table 7.63, shows that the Jordan-CQSO and the FNN-CQSO achieved the highest rank in terms of training and generalization performance, respectively.

Table 7.61: Results of USD time series, scenarios C1 to C3

Scenario / Model	**C1** *(f:150, s:8)*			**C2** *(f:150, s:16)*			**C3** *(f:150, s:20)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	4.42E-04 ± 1.27E-05	9.05E-04 ± 5.31E-05	2.05 ± 0.11	3.84E-04 ± 1.80E-05	6.15E-04 ± 2.40E-05	1.61 ± 0.03	4.53E-04 ± 2.22E-05	1.02E-03 ± 2.39E-05	2.28 ± 0.08
Elman-RPROP	8.08E-04 ± 3.00E-04	2.12E-03 ± 3.46E-04	4.28 ± 0.71	1.27E-03 ± 7.08E-04	1.87E-03 ± 8.04E-04	1.97 ± 0.21	1.65E-03 ± 1.18E-03	2.18E-03 ± 1.16E-03	2.28 ± 0.33
Elman-PSO	5.79E-04 ± 2.90E-05	1.41E-03 ± 1.26E-04	2.43 ± 0.71	5.75E-04 ± 2.76E-05	8.83E-04 ± 3.65E-05	1.55 ± 0.21	5.91E-04 ± 2.95E-05	1.33E-03 ± 6.18E-05	2.27 ± 0.33
Elman-CQPSO	5.37E-04 ± 3.50E-05	1.38E-03 ± 1.52E-04	2.55 ± 0.20	4.13E-04 ± 1.81E-05	7.06E-04 ± 3.92E-05	1.72 ± 0.10	4.91E-04 ± 2.23E-05	1.20E-03 ± 4.97E-05	2.46 ± 0.12
Jordan-RPROP	1.51E-03 ± 5.65E-04	2.66E-03 ± 7.13E-04	2.99 ± 0.63	1.97E-03 ± 8.07E-04	2.62E-03 ± 9.19E-04	1.91 ± 0.25	3.59E-03 ± 1.48E-03	4.29E-03 ± 1.62E-03	1.45 ± 0.21
Jordan-PSO	5.76E-04 ± 2.85E-05	1.20E-03 ± 8.70E-05	2.10 ± 0.13	5.69E-04 ± 2.76E-05	9.39E-04 ± 5.25E-05	1.66 ± 0.08	6.80E-04 ± 2.67E-05	1.31E-03 ± 5.98E-05	1.94 ± 0.08
Jordan-CQPSO	4.84E-04 ± 2.60E-05	1.08E-03 ± 8.96E-05	2.23 ± 0.13	4.49E-04 ± 1.72E-05	6.96E-04 ± 1.97E-05	1.56 ± 0.05	3.76E-04 ± 1.02E-05	1.28E-03 ± 3.65E-05	3.41 ± 0.10
MRNN-RPROP	6.71E-03 ± 2.19E-03	8.38E-03 ± 2.28E-03	1.64 ± 0.42	6.65E-03 ± 3.16E-03	7.12E-03 ± 2.95E-03	1.56 ± 0.25	9.84E-03 ± 4.17E-03	1.03E-02 ± 4.08E-03	1.15 ± 0.11
MRNN-PSO	6.16E-04 ± 3.84E-05	1.47E-03 ± 1.45E-04	2.39 ± 0.19	5.94E-04 ± 3.03E-05	9.33E-04 ± 3.58E-05	1.59 ± 0.08	5.94E-04 ± 2.33E-05	1.47E-03 ± 7.34E-05	2.48 ± 0.12
MRNN-CQPSO	6.25E-04 ± 4.83E-05	1.68E-03 ± 1.30E-04	2.72 ± 0.17	4.46E-04 ± 1.49E-05	8.28E-04 ± 4.07E-05	1.87 ± 0.09	5.97E-04 ± 3.00E-05	1.25E-03 ± 6.23E-05	2.12 ± 0.11
TDNN-RPROP	5.23E-04 ± 4.90E-05	2.57E-03 ± 2.06E-04	5.00 ± 0.25	6.22E-04 ± 9.26E-05	1.21E-03 ± 9.19E-05	2.12 ± 0.18	5.27E-04 ± 8.36E-05	1.47E-03 ± 1.57E-04	3.12 ± 0.38
TDNN-PSO	6.64E-04 ± 3.08E-05	1.80E-03 ± 1.36E-04	2.71 ± 0.16	5.86E-04 ± 2.56E-05	9.41E-04 ± 4.15E-05	1.61 ± 0.05	5.86E-04 ± 2.55E-05	1.42E-03 ± 6.24E-05	2.46 ± 0.15
TDNN-CQPSO	5.23E-04 ± 4.90E-05	2.57E-03 ± 2.06E-04	5.00 ± 0.25	4.72E-04 ± 1.91E-05	7.32E-04 ± 2.22E-05	1.56 ± 0.06	4.22E-04 ± 1.18E-05	1.34E-03 ± 4.54E-05	3.18 ± 0.10

7.2.10 LM Time Series

Scenarios A1 to A3: Table 7.64 shows that the FNN-CQSO outperformed the other models by producing the lowest errors. All the p-values for the pairwise comparisons between the T_E produced by FNN-CQSO and the remaining models were less than the



(a) T_E (b) G_E

Figure 7.28: Training and generalization error results for USD time series, scenario C2

Table 7.62: Models ranking in forecasting the USD time series, scenarios C1 to C3

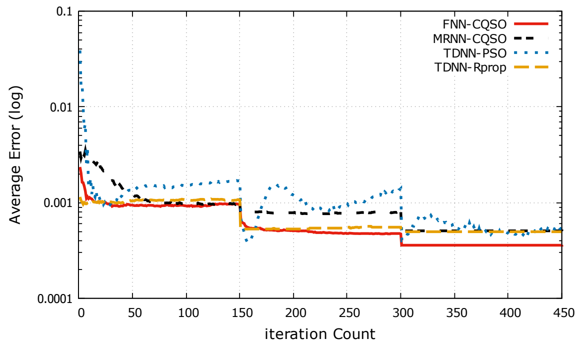
Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	3	1	1.67	1
Elman-RPROP	11	9	11	11	11	11	11	10.33
Elman-PSO	7	5	7	6	7	6	7	5.67
Elman-CQPSO	5	4	2	3	4	2	3.67	3
Jordan-RPROP	12	12	12	12	12	12	12	12
Jordan-PSO	6	3	6	8	10	5	7.33	5.33
Jordan-CQPSO	2	2	4	2	1	4	2.33	2.67
MRNN-RPROP	13	13	13	13	13	13	13	13
MRNN-PSO	8	6	9	7	8	9	8.33	7.33
MRNN-CQPSO	9	7	3	5	9	3	7	5
TDNN-RPROP	4	11	10	10	5	10	6.17	10.17
TDNN-PSO	10	8	8	9	6	8	8	8.33
TDNN-CQPSO	4	11	5	4	2	7	3.5	7.17

0.0001 threshold, except for the FNN-CQSO vs Jordan-CQSO comparison for scenario A3, where the error difference between the two models was statistically insignificant. The p-values for the G_E comparisons indicate that, for the pairs FNN-CQSO vs Elman-RPROP, FNN-CQSO vs Elman-CQSO, FNN-CQSO vs Jordan-PSO, FNN-CQSO vs Jordan-CQSO, FNN-CQSO vs MRNN-PSO, FNN-CQSO vs TDNN-RPROP, and FNN-

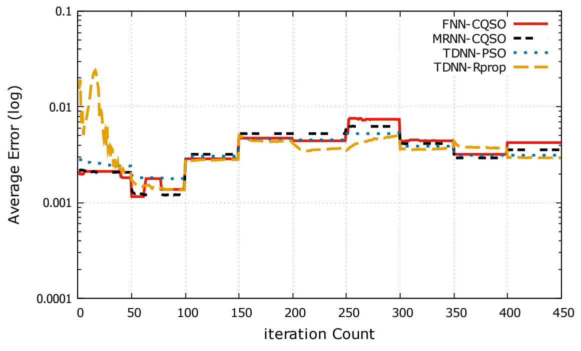
Table 7.63: Overall models ranking in forecasting the USD time series, scenarios A to C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	3	2.33	1.67	1	1.67	1	2.33	1.44
Elman-RPROP	11.33	10.67	11	9.33	11	10.33	11.11	10.11
Elman-PSO	6.33	5.67	6	4.67	7	5.67	6	5.34
Elman-CQPSO	2.67	2.33	3	3.33	3.67	3	2.67	2.89
Jordan-RPROP	11.67	11.33	12	12	12	12	11.67	11.78
Jordan-PSO	9	8.33	9	7	7.33	5.33	8.78	6.89
Jordan-CQPSO	1	2	2	2.67	2.33	2.67	1.67	2.45
MRNN-RPROP	13	13	13	13	13	13	13.00	13
MRNN-PSO	7.33	7	7.67	6.67	8.33	7.33	7.33	7
MRNN-CQPSO	4.67	4.33	4.67	5	7	5	4.56	4.78
TDNN-RPROP	10	10.67	7.33	10.67	6.17	10.17	9.33	10.50
TDNN-PSO	7	8.67	8.17	9.17	8	8.33	7.95	8.72
TDNN-CQPSO	4	4.67	5.5	6.5	3.5	7.17	4.72	6.11

CQSO vs TDNN-CQSO, the difference in performance was insignificant for scenario A1. For scenario A2, the difference in performance was insignificant for the pairs FNN-CQSO vs Jordan-PSO and FNN-CQSO vs Jordan-RPROP. For scenario A3, the p-values indicate that the FNN-CQSO model produced similar generalization results compared to the remaining CQSO trained models and the Jordan-PSO model.



(a) T_E



(b) G_E

Figure 7.29: Training and generalization error results for LM time series, scenario A1

Table 7.64: Results of LM time series, scenario A1 to A3

Scenario / Model	**A1** *(f:50, s:10)*			**A2** *(f:50, s:25)*			**A3** *(f:50, s:31)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	2.29E-03 ± 7.76E-05	3.47E-03 ± 7.04E-05	1.51 ± 0.06	2.38E-03 ± 1.70E-06	2.56E-03 ± 4.53E-06	1.07 ± 0	2.37E-03 ± 1.38E-05	2.24E-03 ± 3.65E-05	0.98 ± 0.02
Elman-RPROP	5.10E-03 ± 1.29E-03	6.54E-03 ± 1.19E-03	1.47 ± 0.15	4.95E-03 ± 1.04E-03	5.48E-03 ± 1.06E-03	1.13 ± 0.04	5.13E-03 ± 2.10E-03	5.23E-03 ± 2.09E-03	1.02 ± 0.03
Elman-PSO	3.27E-03 ± 2.21E-04	3.73E-03 ± 2.57E-04	1.17 ± 0.15	3.38E-03 ± 1.96E-04	2.77E-03 ± 1.12E-04	0.83 ± 0.04	3.71E-03 ± 4.25E-04	2.53E-03 ± 1.25E-04	0.73 ± 0.03
Elman-CQPSO	2.48E-03 ± 1.26E-04	3.78E-03 ± 8.31E-05	1.55 ± 0.07	2.54E-03 ± 5.18E-05	2.63E-03 ± 4.22E-05	1.04 ± 0.01	2.66E-03 ± 8.07E-05	2.30E-03 ± 5.83E-05	0.87 ± 0.03
Jordan-RPROP	4.96E-03 ± 1.62E-03	6.38E-03 ± 1.62E-03	1.4 ± 0.08	1.71E-02 ± 1.02E-02	1.68E-02 ± 9.22E-03	1.07 ± 0.03	9.80E-03 ± 3.21E-03	1.00E-02 ± 3.21E-03	1.03 ± 0.01
Jordan-PSO	2.62E-03 ± 1.28E-04	3.63E-03 ± 8.90E-05	1.41 ± 0.07	2.58E-03 ± 5.63E-05	2.62E-03 ± 5.12E-05	1.02 ± 0.02	2.56E-03 ± 7.67E-05	2.30E-03 ± 4.61E-05	0.9 ± 0.03
Jordan-CQPSO	2.54E-03 ± 1.51E-04	3.76E-03 ± 9.30E-05	1.51 ± 0.08	2.43E-03 ± 2.92E-05	2.60E-03 ± 3.78E-05	1.07 ± 0.01	2.40E-03 ± 2.27E-05	2.27E-03 ± 4.59E-05	0.95 ± 0.02
MRNN-RPROP	6.52E-03 ± 1.33E-03	7.65E-03 ± 1.29E-03	1.22 ± 0.05	7.49E-03 ± 2.48E-03	8.13E-03 ± 2.82E-03	1.09 ± 0.04	9.32E-03 ± 2.22E-03	9.67E-03 ± 2.36E-03	1.04 ± 0.02
MRNN-PSO	3.55E-03 ± 3.85E-04	4.06E-03 ± 4.17E-04	1.2 ± 0.1	3.70E-03 ± 2.85E-04	2.88E-03 ± 1.47E-04	0.8 ± 0.05	3.75E-03 ± 3.52E-04	2.57E-03 ± 1.50E-04	0.71 ± 0.05
MRNN-CQPSO	2.38E-03 ± 6.72E-05	3.77E-03 ± 6.46E-05	1.59 ± 0.05	2.55E-03 ± 6.27E-05	2.62E-03 ± 3.93E-05	1.03 ± 0.02	2.57E-03 ± 5.97E-05	2.31E-03 ± 4.19E-05	0.9 ± 0.02
TDNN-RPROP	2.41E-03 ± 2.47E-04	3.91E-03 ± 3.61E-04	1.63 ± 0.03	4.81E-03 ± 2.31E-03	5.96E-03 ± 2.48E-03	1.28 ± 0.05	3.46E-03 ± 6.08E-04	3.85E-03 ± 9.69E-04	1.06 ± 0.06
TDNN-PSO	2.57E-03 ± 1.64E-04	3.55E-03 ± 1.98E-04	1.4 ± 0.07	2.74E-03 ± 1.28E-04	2.93E-03 ± 1.73E-04	1.07 ± 0.04	3.15E-03 ± 3.30E-04	2.92E-03 ± 2.42E-04	0.94 ± 0.04
TDNN-CQPSO	2.77E-03 ± 1.08E-04	3.80E-03 ± 9.71E-05	1.38 ± 0.06	2.57E-03 ± 5.40E-05	2.77E-03 ± 5.21E-05	1.08 ± 0.04	2.64E-03 ± 3.41E-05	2.25E-03 ± 3.56E-05	0.85 ± 0.02

The ρ values in Table 7.64 show that all the models overfitted for the gradual scenario A1. However, for the more severely changing scenario A2, the models exhibited minor or no signs of overfitting, except for the Elman-RPROP and the TDNN-RPROP models which overfitted. The ρ values also indicated that the generalization behaviour

Table 7.65: Models ranking in forecasting the LM time series, scenarios A1 to A3

Model	A1		A2		A3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	1	1	1	1	1	1	1	1
Elman-RPROP	12	12	11	10	11	11	11.33	11
Elman-PSO	9	4	8	7	9	7	8.67	6
Elman-CQPSO	4	7	3	5	6	5	4.33	5.67
Jordan-RPROP	11	11	13	13	13	13	12.33	12.33
Jordan-PSO	7	3	6	4	3	4	5.33	3.67
Jordan-CQPSO	5	5	2	2	2	3	3	3.33
MRNN-RPROP	13	13	12	12	12	12	12.33	12.33
MRNN-PSO	10	10	9	8	10	8	9.67	8.67
MRNN-CQPSO	2	6	4	3	4	6	3.33	5
TDNN-RPROP	3	9	10	11	8	10	7	10
TDNN-PSO	6	2	7	9	7	9	6.67	6.67
TDNN-CQPSO	8	8	5	6	5	2	6	5.33

of the models improved for abrupt scenario A3 compared to scenario A2, where only the RPROP trained models slightly overfitted.

Figure 7.29 illustrates the performance progression over time for the four best performing models for scenario A1. The models selected include the FNN-CQSO, MRNN-CQSO, TDNN-RPROP and TDNN-PSO. As visualized in Figure 7.29a, all the models kept track of the moving lowest value during training, and adapted well to the changes. The MRNN-PSO, however, had the worst T_E progression compared to the other models, while the FNN-CQSO had the best T_E progression. Figure 7.29b clearly shows that the G_E values produced by the four models throughout the search were higher than the corresponding T_E values shown in Figure 7.29a. Therefore, all four models overfitted the data. Figure 7.29b also shows that, after the initial 50 epochs (i.e. when the first window slide occur), the models produced a similar G_E throughout the remaining epochs. This implies that, in predicting the LM time series for the A1 scenario, the FNN-CQSO produced at least an equal or better performance compared to the RNNs trained using either RPROP, PSO or CQSO.

The performance ranking of the models given in Table 7.65 shows that the FNN-CQSO achieved the highest average rank over the three scenarios.

Scenarios B1 to B3: The error values presented in Table 7.66 show that the TDNN-RPROP outperformed the remaining models for scenario B1. For scenario B2, the FNN-

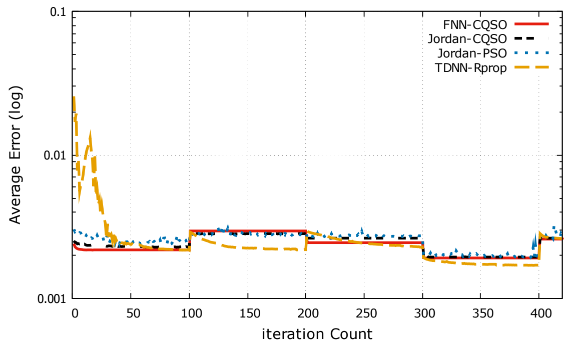
Table 7.66: Results of LM time series, scenario B1 to B3

Scenario / Model	**B1** *(f:100, s:10)*			**B2** *(f:100, s:25)*			**B3** *(f:100, s:31)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQPSO	2.39E-03 ± 8.84E-05	3.78E-03 ± 4.56E-05	1.6 ± 0.06	2.38E-03 ± 7.10E-07	2.55E-03 ± 3.54E-06	1.07 ± 0	2.41E-03 ± 1.14E-05	2.26E-03 ± 1.77E-05	0.94 ± 0.01
Elman-RPROP	6.22E-03 ± 2.29E-03	8.06E-03 ± 2.60E-03	1.41 ± 0.11	5.32E-03 ± 1.36E-03	5.73E-03 ± 1.32E-03	1.12 ± 0.05	6.28E-03 ± 2.24E-03	6.71E-03 ± 2.34E-03	1.08 ± 0.05
Elman-PSO	3.37E-03 ± 1.85E-04	3.79E-03 ± 2.07E-04	1.15 ± 0.11	3.78E-03 ± 2.50E-04	2.88E-03 ± 1.44E-04	0.78 ± 0.05	3.96E-03 ± 3.87E-04	2.52E-03 ± 1.68E-04	0.67 ± 0.05
Elman-CQPSO	2.42E-03 ± 1.05E-04	3.79E-03 ± 8.47E-05	1.58 ± 0.05	2.67E-03 ± 1.01E-04	2.69E-03 ± 7.21E-05	1.01 ± 0.02	2.71E-03 ± 1.00E-04	2.33E-03 ± 5.33E-05	0.87 ± 0.03
Jordan-RPROP	6.62E-02 ± 1.21E-01	6.57E-02 ± 1.17E-01	1.44 ± 0.07	7.65E-02 ± 1.35E-01	8.15E-02 ± 1.44E-01	1.09 ± 0.02	5.65E-03 ± 1.36E-03	5.81E-03 ± 1.41E-03	1.03 ± 0.02
Jordan-PSO	2.46E-03 ± 1.39E-04	3.68E-03 ± 7.05E-05	1.52 ± 0.08	2.54E-03 ± 6.58E-05	2.59E-03 ± 5.00E-05	1.02 ± 0.02	2.48E-03 ± 3.65E-05	2.33E-03 ± 3.97E-05	0.94 ± 0.02
Jordan-CQPSO	2.32E-03 ± 1.16E-04	3.99E-03 ± 9.43E-05	1.75 ± 0.09	2.44E-03 ± 2.45E-05	2.58E-03 ± 3.52E-05	1.06 ± 0.01	2.42E-03 ± 2.38E-05	2.28E-03 ± 3.41E-05	0.94 ± 0.02
MRNN-RPROP	4.40E-03 ± 8.68E-04	5.68E-03 ± 9.01E-04	1.35 ± 0.08	5.99E-03 ± 9.01E-04	6.38E-03 ± 9.13E-04	1.07 ± 0.02	6.99E-03 ± 2.29E-03	7.19E-03 ± 2.39E-03	1.03 ± 0.02
MRNN-PSO	3.44E-03 ± 2.80E-04	3.95E-03 ± 3.39E-04	1.18 ± 0.11	3.67E-03 ± 2.80E-04	2.80E-03 ± 1.49E-04	0.78 ± 0.05	3.84E-03 ± 4.47E-04	2.69E-03 ± 1.60E-04	0.74 ± 0.06
MRNN-CQPSO	2.59E-03 ± 1.46E-04	3.66E-03 ± 7.00E-05	1.44 ± 0.07	2.64E-03 ± 5.24E-05	2.71E-03 ± 4.89E-05	1.03 ± 0.01	2.65E-03 ± 7.23E-05	2.34E-03 ± 4.04E-05	0.89 ± 0.02
TDNN-RPROP	2.09E-03 ± 7.59E-05	3.51E-03 ± 1.40E-04	1.68 ± 0.05	2.68E-03 ± 2.41E-04	3.48E-03 ± 3.38E-04	1.3 ± 0.04	3.09E-03 ± 3.05E-04	3.36E-03 ± 4.29E-04	1.07 ± 0.04
TDNN-PSO	2.66E-03 ± 1.62E-04	3.89E-03 ± 2.71E-04	1.47 ± 0.06	2.95E-03 ± 2.63E-04	3.09E-03 ± 2.14E-04	1.06 ± 0.04	2.73E-03 ± 1.17E-04	2.67E-03 ± 1.60E-04	0.98 ± 0.04
TDNN-CQPSO	3.64E-03 ± 2.31E-04	4.08E-03 ± 1.21E-04	1.16 ± 0.11	2.60E-03 ± 4.85E-05	2.83E-03 ± 4.90E-05	1.09 ± 0.03	2.61E-03 ± 2.80E-05	2.24E-03 ± 3.09E-05	0.86 ± 0.01

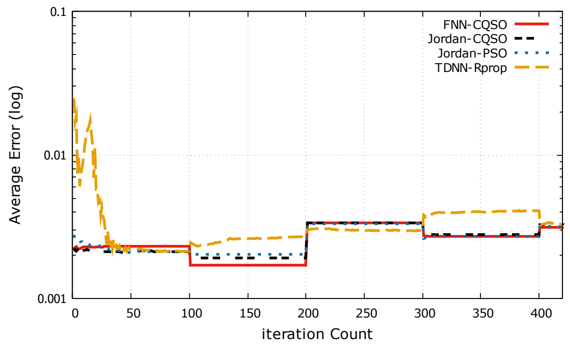
CQSO outperformed the other models by yielding the lowest errors. For scenario B3, the FNN-CQSO and the TDNN-CQSO respectively obtained the best T_E and G_E values. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models were less than the 0.0001 threshold, except for the following pairs: for scenario B1, FNN-CQSO vs Elman-CQSO in terms of T_E, and FNN-CQSO vs Elman-PSO, FNN-CQSO vs Elman-CQSO, FNN-CQSO vs MRNN-PSO, and FNN-CQSO vs TDNN-CQSO in terms of G_E. For scenario B2, the exceptions are FNN-CQSO vs Jordan-PSO and FNN-CQSO vs Jordan-CQSO, all in terms of G_E. The exceptions for scenario B3 are FNN-CQSO vs Jordan-CQSO in terms of both T_E and G_E, and FNN-CQSO vs TDNN-CQSO in terms of G_E.

The ρ values in Table 7.66 reveal that all the models overfitted for scenario B1. For scenario B2, the models exhibited either minor or no sign of overfitting. For scenario B3, the PSO and the CQSO trained models did not overfit, while the RPROP models slightly overfitted.

Figure 7.30 illustrates the performance progression over time for the FNN-CQSO and three other models in predicting the LM time series for scenario B2. Figure 7.30 shows that the RPROP based model took about 40 epochs to locate the lowest value, while the other three models, including the FNN-CQSO, took less than five epochs. The figure also reveals that the models kept close track of the moving lowest value. The FNN-CQSO progression indicates that it performed mostly better than the other models throughout the search.



(a) T_E



(b) G_E

Figure 7.30: Training and generalization error results for LM time series, scenario B2

Table 7.67: Models ranking in forecasting the LM time series, scenarios B1 to B3

Model	B1		B2		B3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQPSO	3	4	1	1	1	2	1.67	2.33
Elman-RPROP	12	12	11	11	12	12	11.67	11.67
Elman-PSO	8	6	10	8	10	7	9.33	7
Elman-CQSO	4	5	6	4	6	5	5.33	4.67
Jordan-RPROP	13	13	13	13	11	11	12.33	12.33
Jordan-PSO	5	3	3	3	3	4	3.67	3.33
Jordan-CQSO	2	9	2	2	2	3	2	4.67
MRNN-RPROP	11	11	12	12	13	13	12	12
MRNN-PSO	9	8	9	6	9	9	9	7.67
MRNN-CQSO	6	2	5	5	5	6	5.33	4.33
TDNN-RPROP	1	1	7	10	8	10	5.33	7
TDNN-PSO	7	7	8	9	7	8	7.33	8
TDNN-CQSO	10	10	4	7	4	1	6	6

The average performance ranking of the models given in Table 7.67 shows that the FNN-CQSO emerged as the winner.

Scenarios C1 to C3: Table 7.68 shows that the TDNN-RPROP obtained the lowest T_E and G_E values for scenario C1 compared to the remaining models. For scenarios C2 and C3, the performance of the FNN-CQSO improved by producing the best results. All the p-values for the pairwise comparisons between the FNN-CQSO and the other models are below the 0.0001 threshold, except when compared to:

- ELMAN-PSO in terms of G_E for scenario C1,
- ELMAN-CQSO for scenario C1,
- JORDAN-RPROP in terms of T_E for scenario C1,
- JORDAN-PSO in terms of G_E for scenarios C2 and C2,
- JORDAN-CQSO for scenarios C1, C2 and C3,

- MRNN-PSO in terms of G_E for scenario C1,
- MRNN-CQSO in terms of T_E and G_E for scenarios C1 and C3, respectively,
- TDNN-RPROP in terms of T_E for scenario C2,

Table 7.68: Results of LM time series, scenario C1 to C3

Scenario / Model	**C1** *(f:150, s:10)*			**C2** *(f:150, s:25)*			**C3** *(f:150, s:31)*		
	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
FF-CQSO	2.58E-03 ± 1.34E-04	3.74E-03 ± 4.64E-05	1.48 ± 0.07	2.38E-03 ± 5.57E-06	2.54E-03 ± 1.15E-05	1.07 ± 0.01	2.42E-03 ± 7.85E-06	2.26E-03 ± 1.67E-05	0.93 ± 0.01
Elman-RPROP	4.08E-03 ± 8.13E-04	5.31E-03 ± 6.67E-04	1.42 ± 0.09	5.72E-03 ± 1.95E-03	6.13E-03 ± 1.97E-03	1.12 ± 0.05	8.76E-03 ± 2.87E-03	9.10E-03 ± 3.25E-03	1.01 ± 0.04
Elman-PSO	2.99E-03 ± 1.18E-04	3.68E-03 ± 2.36E-05	1.25 ± 0.09	3.83E-03 ± 3.38E-04	2.85E-03 ± 1.04E-04	0.77 ± 0.05	3.51E-03 ± 2.41E-04	2.45E-03 ± 1.14E-04	0.72 ± 0.04
Elman-CQSO	2.53E-03 ± 1.41E-04	3.75E-03 ± 7.92E-05	1.51 ± 0.07	2.69E-03 ± 8.03E-05	2.73E-03 ± 4.93E-05	1.02 ± 0.02	2.86E-03 ± 1.31E-04	2.37E-03 ± 5.75E-05	0.84 ± 0.03
Jordan-RPROP	3.22E-03 ± 6.14E-04	4.65E-03 ± 6.43E-04	1.51 ± 0.07	7.13E-02 ± 1.22E-01	7.69E-02 ± 1.32E-01	1.14 ± 0.03	4.32E-03 ± 6.13E-04	4.39E-03 ± 5.91E-04	1.02 ± 0.02
Jordan-PSO	2.50E-03 ± 9.54E-05	3.62E-03 ± 6.41E-05	1.47 ± 0.06	2.49E-03 ± 4.13E-05	2.58E-03 ± 2.69E-05	1.04 ± 0.02	2.47E-03 ± 5.24E-05	2.30E-03 ± 3.56E-05	0.93 ± 0.02
Jordan-CQSO	2.46E-03 ± 1.44E-04	3.92E-03 ± 9.69E-05	1.64 ± 0.11	2.41E-03 ± 1.73E-05	2.57E-03 ± 2.33E-05	1.07 ± 0.01	2.42E-03 ± 1.81E-05	2.28E-03 ± 3.50E-05	0.94 ± 0.01
MRNN-RPROP	3.88E-03 ± 7.60E-04	4.92E-03 ± 7.34E-04	1.34 ± 0.08	4.92E-03 ± 4.17E-04	5.27E-03 ± 4.40E-04	1.08 ± 0.03	4.94E-03 ± 7.75E-04	5.04E-03 ± 7.92E-04	1.02 ± 0.02
MRNN-PSO	3.54E-03 ± 3.95E-04	3.88E-03 ± 2.86E-04	1.17 ± 0.11	3.87E-03 ± 2.87E-04	2.78E-03 ± 1.05E-04	0.74 ± 0.04	3.99E-03 ± 4.39E-04	2.64E-03 ± 2.50E-04	0.7 ± 0.07
MRNN-CQSO	2.61E-03 ± 1.30E-04	3.88E-03 ± 8.06E-05	1.5 ± 0.06	2.65E-03 ± 5.44E-05	2.67E-03 ± 4.53E-05	1.01 ± 0.02	2.67E-03 ± 8.17E-05	2.32E-03 ± 5.60E-05	0.87 ± 0.02
TDNN-RPROP	2.10E-03 ± 1.06E-04	3.54E-03 ± 1.78E-04	1.69 ± 0.05	2.56E-03 ± 1.74E-04	3.37E-03 ± 2.54E-04	1.32 ± 0.03	3.16E-03 ± 5.18E-04	3.40E-03 ± 6.99E-04	1.06 ± 0.04
TDNN-PSO	2.69E-03 ± 2.13E-04	3.64E-03 ± 1.78E-04	1.39 ± 0.07	2.74E-03 ± 2.47E-04	2.98E-03 ± 2.40E-04	1.1 ± 0.03	2.74E-03 ± 1.53E-04	2.61E-03 ± 1.76E-04	0.95 ± 0.03
TDNN-CQSO	5.22E-03 ± 2.42E-04	4.76E-03 ± 1.36E-04	0.92 ± 0.04	2.68E-03 ± 6.66E-05	2.80E-03 ± 4.58E-05	1.05 ± 0.04	2.69E-03 ± 2.91E-05	2.28E-03 ± 2.63E-05	0.85 ± 0.01

- TDNN-PSO in terms of T_E for scenario C1, and
- TDNN-CQSO in terms of G_E for scenario C3.

For these exceptions, all the models produced statistically similar errors.

The ρ values in Table 7.68 show that, while all the models overfitted for scenario C1, the models exhibited no or minor overfitting behaviour for scenarios C2 and C3, except for the Jordan-RPROP and the TDNN-RPROP that overfitted for scenario C2.

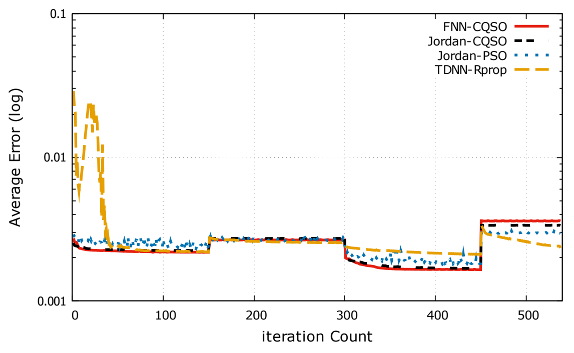
Figure 7.31 illustrates the performance progression over time achieved by the FNN-CQSO and three other models for the C3 scenario. The observations made from the figure are similar to that of Figure 7.30.

The average performance ranking over the three scenarios shown in Table 7.69 shows that the FNN-CQSO emerged as the winner.

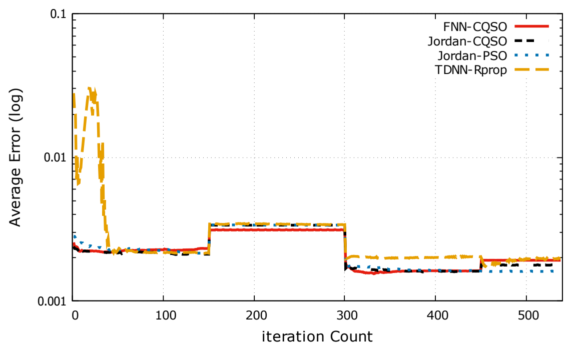
The overall average ranking of the models in predicting the LM time series for all nine scenarios is presented in Table 7.70. The table shows that the FNN-CQSO model emerged as the overall winner in predicting the LM time series. This shows that the FNN

Table 7.69: Models ranking in forecasting the LM time series, scenarios C1 to C3

Model	C1		C2		C3		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	5	5	1	1	1	1	2.33	2.33
Elman-RPROP	12	13	12	12	13	13	12.33	12.67
Elman-PSO	8	4	9	8	9	7	8.67	6.33
Elman-CQSO	4	6	7	5	7	6	6	5.67
Jordan-RPROP	9	10	13	13	11	11	11	11.33
Jordan-PSO	3	2	3	3	3	4	3	3
Jordan-CQSO	2	9	2	2	2	2	2	4.33
MRNN-RPROP	11	12	11	11	12	12	11.33	11.67
MRNN-PSO	10	7	10	6	10	9	10	7.33
MRNN-CQSO	6	8	5	4	4	5	5	5.67
TDNN-RPROP	1	1	4	10	8	10	4.33	7
TDNN-PSO	7	3	8	9	6	8	7	6.67
TDNN-CQSO	13	11	6	7	5	3	8	7



(a) T_E



(b) G_E

Figure 7.31: Training and generalization error results for LM time series, scenario C3

trained using CQSO performed better than the RNNs trained using either a RPROP, PSO or CQSO algorithm.

Table 7.70: Overall model's average ranking in forecasting the LM time series, scenarios A to C

Model	A		B		C		Average Rank	
	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
FF-CQSO	1	1	1.67	2.33	2.33	2.33	1.67	1.89
Elman-RPROP	11.33	11	11.67	11.67	12.33	12.67	11.78	11.78
Elman-PSO	8.67	6	9.33	7	8.67	6.33	8.89	6.44
Elman-CQSO	4.33	5.67	5.33	4.67	6	5.67	5.22	5.34
Jordan-RPROP	12.33	12.33	12.33	12.33	11	11.33	11.89	12
Jordan-PSO	5.33	3.67	3.67	3.33	3	3	4	3.33
Jordan-CQSO	3	3.33	2	4.67	2	4.33	2.33	4.11
MRNN-RPROP	12.33	12.33	12	12	11.33	11.67	11.89	12
MRNN-PSO	9.67	8.67	9	7.67	10	7.33	9.56	7.89
MRNN-CQSO	3.33	5	5.33	4.33	5	5.67	4.55	5
TDNN-RPROP	7	10	5.33	7	4.33	7	5.55	8
TDNN-PSO	6.67	6.67	7.33	8	7	6.67	7	7.11
TDNN-CQSO	6	5.33	6	6	8	7	6.67	6.11

7.3 Summary

The chapter aimed to investigate if using recurrent/delayed connections are unnecessary in NNs used for non-stationary time series forecasting when a PSO designed for dynamic environments is used as the training algorithm. A set of experiments using ten forecasting problems were carried out to test this hypothesis. In the experiments, FNNs were trained using CQSO to forecast each of the problems under nine different dynamic scenarios and the results were compared to the results obtained from four different RNNs (i.e. Elman NN, Jordan NN, MRNN and TDNN), each trained differently using RPROP, PSO and dynamic PSO algorithms. Mann Whitney U tests were used to check the statistical significance of the differences in performance between the results obtained from the FNN trained with CQSO and each of the remaining NN models.

An analysis of the results showed that for nine out of the ten problems, the FNN trained with CQSO achieved the highest performance ranking. For the tenth problem, a Jordan NN trained with the CQSO algorithm outperformed the FNN-CQSO model. However, the FNN-CQSO model had significantly superior performance compared to all the other models, including the Jordan NN trained with RPROP and PSO algorithms.

It was observed that, in general, training the RNNs with CQSO improved performance over training with RPROP or PSO.

The results supported the hypothesis that a FNN trained with a dynamic PSO algorithm is sufficient for time series forecasting in non-stationary environments and that the model is able to handle temporal relationships without necessarily introducing recurrent connections.

Chapter 8

Conclusions

"It's how you get to the conclusion that makes it interesting"

– Sylvester Stallone

This chapter provides a summary of the major findings and contributions of this thesis. The chapter also suggests related opportunities for future research. Section 8.1 presents the conclusions arrived at and Section 8.2 discusses the possible topics for future research.

8.1 Summary of Conclusions

This thesis had two primary objectives. The first objective was to investigate the applicability and efficiency of a dynamic PSO algorithm as training algorithm for FNN forecasters for non-stationary environments. The second objective was to test the hypothesis that recurrent/delayed connections are not necessary in NNs used for non-stationary time series forecasting if a dynamic PSO algorithm is used as the training algorithm. Thus, two empirical studies were conducted. In the first empirical analysis, a dynamic PSO algorithm (i.e. CQSO) was applied to train a FNN forecaster on a number of forecasting problems under a representative selection of dynamic scenarios to examine the applicability of the dynamic PSO as a training algorithm. The second empirical analysis trained a FNN and four different types of SRNNs using a dynamic PSO on a selection of forecasting problems under different dynamic scenarios, to test if the recurrent/delay connections in the SRNNs are necessary. For the two empirical studies, the results obtained for the dynamic PSO are compared against standard PSO and backpropagation (RPROP) algorithms. This chapter summarizes the major observations and contribution made in the course of this study.

The research work started with a review of time series analysis and forecasting, as well as discussions on the widely used time series forecasting methods. NNs used in modeling time series were discussed next, with performance issues in designing NN forecasters. The optimization methods applied to training NNs were also discussed.

The basic PSO algorithm and its control parameters were discussed, followed by a description of PSO's application to NN training. Even though PSO has been successfully applied to train NNs, it was usually assumed that the training data is static. It was suggested in this thesis that dynamic PSO algorithms be used to train NN forecasters under non-stationary environments as an alternative to standard PSO. Thus, dynamic optimization problems, and the different characteristics of dynamic environments were discussed. This was followed by discussion on the challenges faced by standard PSO when applied to dynamic environments, and methods of addressing these challenges. Modificaions of PSO to suit dynamic environments were discussed.

For the first empirical study, an experimental procedure was designed to compare the performance of the CQSO, PSO and RPROP on a representative selection of forecasting problems under a representative selection of dynamic environments. A sliding window approach was used to simulate dynamic environments of varying spatial and temporal severity, by adjusting the step size of the sliding window and the number of algorithm iterations between the sliding window shifts, respectively.

A set of ten, different well studied time series with varying complexities were used in the study. Seven of these are real world time series and the other three were artificially generated.

An empirical comparison, in terms of T_E and G_E, shows that the dynamic PSO (i.e. the CQSO) outperformed both the standard PSO and the RPROP algorithms. Thus, CQSO was shown to be very efficient in training FNN forecasters, and is a viable alternative to both the standard PSO and RPROP.

It was observed throughout the empirical analysis that the CQSO and the PSO were more successful under the severely and abruptly changing scenarios, while RPROP was more successful under the gradually changing scenarios. The CQSO also converged faster than PSO and RPROP. RPROP was, however, more sensitive to stale data in the sliding window.

The CQSO involved more parameter optimization than both PSO and RPROP, since, in addition to the standard PSO parameters, other parameters specific to the CQSO also require optimization. Hence, the CQSO required more fine-tuning than PSO and RPROP, but have a higher potential to outperform the other two algorithms.

RPROP overfitted whenever the temporal severity allowed RPROP to train for too long. The CQSO, however, exhibited minor or no overfitting for most of the scenarios considered.

For the second empirical study, the performance of FNNs trained using a dynamic PSO algorithm to forecast ten problems under different dynamic scenarios was compared to that of four different types of RNNs (i.e. Elman NN, Jordan NN, Multi-Recurrent NN and Time Delay NN), each trained separately using RPROP, standard PSO and the dynamic PSO algorithm.

An analysis of the results showed that the FNN trained with CQSO achieved superior performance compared to the other RNN models. It was observed that, in general, training the RNNs with CQSO improved performance over training with RPROP or PSO.

It is also observed that the Jordan NN trained with CQSO achieved the best result under most of the scenarios for all the problems.

The findings supported the hypothesis that a FNN trained with a dynamic PSO algorithm is sufficient for time series forecasting in non-stationary environments (for the time series considered in this study), and the model is able to handle temporal relationship without necessarily introducing recurrent connections.

8.2 Future Work

This section briefly summarizes possible future research topics suggested by the work presented in this thesis.

This study considered only one dynamic PSO algorithm as a representation of the existing dynamic PSO algorithms. Thus, evaluation of additional dynamic PSO algorithms can provide further insight into the applicability and behaviours of the dynamic PSOs in training NN forecasters for non-stationary environments. Other properties of training algorithms not explored in this work, such as recovery speed after a change can also be investigated.

The control parmeters of all the training algorithms used in this study, including the dynamic PSO, were optimised once for each problem. Afterwards, the algorithm control parameters remained static throughout the algorithm runs. Since existing adaptive parameter strategies for PSO were developed for static environments, they can not be blindly applied to dynamic problems. The efficiency of existing dynamic NN training algorithms can be improved by considering adaptive control parameter strategies, either by adapting the existing strategies accordingly, or by developing new self-adaptive control parameter strategies specific to dynamic environments.

The number of hidden units and the number of connections in the NN were optimised once for each problem, and did not change during the algorithm run. However, a real-life forecasting problem with non stationary data may require changes to the NN architecture in order to prevent underfitting and overfitting. Developing adaptive architecture selection strategies specific to dynamic environments is an important topic for future research.

PSO is not the only CI technique that has already been adapted to dynamic environments. Another broad CI field that has been successfully applied to dynamic problems is evolutionary computation (EC). EC algorithms have been applied to NN training before, and it would be interesting to determine the applicability and efficiency of dynamic EC algorithms to the training of NN forecasters for non-stationary environments.

Bibliography

[1] S A Abdulkarim. Time Series Prediction with Simple Recurrent Neural Networks. *Bayero Journal of Pure and Applied Sciences*, 9(1):19–24, 2016.

[2] S A Abdulkarim and A B Garko. Forecasting Maternal Mortality Rate using Particle Swarm Optimization Based Artificial Neural Network. *Dutse Journal of Pure and Applied Sciences*, 1(1):55–59, 2015.

[3] R Adhikari and R K Agrawal. Effectiveness of PSO Based Neural Network for Seasonal Time Series Forecasting. In *Proceedings of the Fitfh Indian Internatiinal Conference on Artificial Intelligence*, pages 231–244, 2011.

[4] R Adhikari and R K Agrawal. Forecasting Strong Seasonal Time Series with Artificial Neural Networks. *Journal of Scientific and Industrial Research*, (10):657, 2012.

[5] R Adhikari, R K Agrawal, and L Kant. PSO based Neural Networks vs. Traditional Statistical Models for Seasonal Time Series Forecasting. In *Proceedings of the 3rd IEEE Conference on Advance Computing*, pages 719–725, 2013.

[6] M Adya and F Collopy. How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation. *Journal of Forecasting*, 495, 1998.

[7] M H Ahmadi, S S G Aghaj, and A Nazeri. Prediction of Power in Solar Stirling Heat Engine by Using Neural Network Based on Hybrid Genetic Algorithm and Particle Swarm Optimization. *Neural Computing and Applications*, 22(6):1141–1150, 2013.

[8] W Al-Saedi, S W Lachowicz, D Habibi, and O Bass. Power Flow Control in Grid-connected Microgrid Operation using Particle Swarm Optimization under Variable Load Conditions. *International Journal of Electrical Power & Energy Systems*, 49:76–85, 2013.

[9] A Alexandridis, E Chondrodima, and H Sarimveis. Cooperative Learning for Radial Basis Function Networks using Particle Swarm Optimization. *Applied Soft Computing*, 49:485–497, 2016.

[10] P J Angeline. Tracking Extrema in Dynamic Environments. In *Proceedings of International Conference on Evolutionary Programming*, pages 335–345. Springer, 1997.

[11] A F Atiya, S M El-Shoura, S I Shaheen, and M S El-Sherif. A Comparison Between Neural-network Forecasting Techniques - Case Study: River Flow Forecasting. *IEEE Transactions on neural networks*, 10(2):402–409, 1999.

[12] M Balasaraswathi and B Kalpana. Metaheuristics for Mining Massive Datasets: A Comprehensive Study of PSO for Classification. *Advances in Natural and Applied Sciences*, 9(5):27–39, 2015.

[13] R Battiti. First-and Second-order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural computation*, 4(2):141–166, 1992.

[14] S Becker and Y L Cun. Improving the Convergence of Back-propagation Learning with Second Order Methods. In *Proceedings of the Connectionist Models Summer School of San Matteo, CA:Margan Kaufmann*, pages 29–37, 1988.

[15] T Blackwell and J Branke. Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

[16] T Blackwell, J Branke, and X Li. Particle Swarms for Dynamic Optimization Problems. *Swarm Intelligence. Springer Berlin Heidelberg*, pages 193–217, 2008.

[17] T M Blackwell and P J Bentley. Dynamic Search With Charged Swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002.

[18] T Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[19] B E Boser, I M Guyon, and V N Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

[20] G E P Box and G M Jenkins. Time Series Analysis Forecasting and Control. Technical report, Wisconsin University Madison Department of Statistics, 1970.

[21] G E P Box, G M Jenkins, G C Reinsel, and G M Ljung. *Time Series Analysis: Forecasting and Control.* John Wiley & Sons, 2015.

[22] J Branke, E Salihoğlu, and Ş Uyar. Towards an Analysis of Dynamic Environments. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1433–1440. ACM, 2005.

[23] D Brezak, T Bacek, D Majetic, J Kasac, and B Novakovic. A Comparison of Feedforward and Recurrent Neural Networks in Time Series Forecasting. In *Proceedings of IEEE Conference on Computational Intelligence for Financial Engineering and Economics*, pages 1–6, 2012.

[24] A G Brown. *Nerve Cells and Nervous Systems: An Introduction to Neuroscience.* Springer Science & Business media, 2012.

[25] L Cao and F E H Tay. Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, 2003.

[26] A Carlisle and G Dozier. Adapting Particle Swarm Optimization to Dynamic Environments. In *Proceedings of International Conference on Artificial Intelligence*, volume 1, pages 429–434, 2000.

[27] A Carlisle and G Dozier. An Off-the-shelf PSO. In *Proceedings of the Workshop on Particle Swarm Optimization*, volume 1, pages 1–6. Citeseer, 2001.

[28] A Carlisle and G Dozler. Tracking Changing Extrema with Adaptive Particle Swarm Optimizer. In *Proceedings of the 5th Biannual World Automation Congress*, volume 13, pages 265–270. IEEE, 2002.

[29] K Chang, R Chen, and T B Fomby. Prediction-based Adaptive Compositional Model for Seasonal Time Series Analysis. *Journal of Forecasting*, 36(7):842–853, 2017.

[30] C Chatfield. *The Analysis of Time Series: An Introduction.* CRC press, 2016.

[31] H Cheng, P Tan, J Gao, and J Scripps. Multistep-ahead Time Series Prediction. *Advances in knowledge discovery and data mining*, pages 765–774, 2006.

[32] M Clerc and J Kennedy. The Particle Swarm Explosion , Stability , and Convergence in a Multidimensional Complex Space. *IEEE Transaction on Evolutionary Computation*, 6(1):58–73, 2002.

[33] C Cortes and V Vapnik. Support-vector Networks. *Machine learning*, 20(3):273–297, 1995.

[34] T Cover and P Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[35] P S Crowther and R J Cox. A method for optimal division of data sets for use in neural networks. In *Proceedings of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 1–7. Springer, 2005.

[36] G Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[37] M Dadgar, S Jafari, and A Hamzeh. A PSO-based Multi-robot Cooperation Method for Target Searching in Unknown Environments. *Neurocomputing*, 177:62–74, 2016.

[38] K Deb, D Joshi, and A Anand. Real-coded Evolutionary Algorithms with Parent-centric Recombination. In *Proceedings of Congress on Evolutionary Computation*, volume 1, pages 61–66. IEEE, 2002.

[39] J Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, pages 1–30, 2006.

[40] G. Dorffner. Neural Networks for Time Series Processing. *Neural Network World*, 1996.

[41] H Drucker, C J C Burges, L Kaufman, A J Smola, and V Vapnik. Support Vector Regression Machines. In *Advances in Neural Information Processing Systems*, pages 155–161, 1997.

[42] J G Duhain. *Particle Swarm Optimization in Dynamically Changing Environment An Empirical Study*. MSc Thesis, University of Pretoria, 2011.

[43] J G Duhain and A P Engelbrecht. Towards a more Complete Classification System for Dynamically Changing Environments. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[44] R Eberhart and J Kennedy. A New Optimizer using Particle Swarm Theory. In *Proceedings of the IEEE 6th International Symposium on Micro Machine and Human Science*, volume 1, pages 39–43, 1995.

[45] R Eberhart, P Simpson, and R Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, Inc., 1996.

[46] R C Eberhart and Y Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 84–88. IEEE, 2000.

[47] R C Eberhart and Y Shi. Tracking and Optimizing Dynamic Systems with Particle Swarms. In *Proceedings of Congress on Evolutionary Computation*, volume 1, pages 94–100. IEEE, 2001.

[48] J L Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[49] A P Engelbrecht. *Fundamentals of Computational Swarm Intelligence.* 2nd Ed. Hoboken, US: John Wiley & Sons, Ltd, 2006.

[50] A P Engelbrecht. Fitness Function Evaluations: A Fair Stopping Condition? In *Proceedings of IEEE Symposium on Swarm Intelligence (SIS)*, pages 1–8, 2014.

[51] R F Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica: Journal of the Econometric Society*, pages 987–1007, 1982.

[52] E Fix and J L Hodges Jr. Discriminatory Analysis-nonparametric Discrimination: Consistency Properties. Technical report, California Univ Berkeley, 1951.

[53] R J Frank, N Davey, and S P Hunt. Time Series Prediction and Neural Networks. *Journal of intelligent and robotic systems*, 31(1-3):91–103, 2001.

[54] I M Galvan and P Isasi. Multi-step Learning Rule for Recurrent Neural Models : An Application to Time Series Forecasting. *Neural Processing Letters*, 13(2):115–133, 2001.

[55] Everette S Gardner J. Exponential Smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–28, 1985.

[56] J Ghorpade-Aher and V A Metre. PSO Based Multidimensional Data Clustering: A Survey. *International Journal of Computer Applications (0975–8887)*, 87(16), 2014.

[57] B Gordan, D J Armaghani, M Hajihassani, and M Monjezi. Prediction of Seismic Slope Stability Through Combination of Particle Swarm Optimization and Neural Network. *Engineering with Computers*, 32(1):85–97, 2016.

[58] E A Grimaldi, F Grimaccia, M Mussetta, and R E Zich. PSO as an Effective Learning Algorithm for Neural Network Applications. In *Proceedings of International Conference on Computational Electromagnetics and Its Applications*, pages 557–560. IEEE, 2004.

[59] S Grossberg. Recurrent Neural Networks. *Scholarpedia*, 8(2):1888, 2013.

[60] M Hajihassani, D J Armaghani, M Monjezi, E T Mohamad, and A Marto. Blast-induced Air and Ground Vibration Prediction: A Particle Swarm Optimization-based Artificial Neural Network Approach. *Environmental Earth Sciences*, 74(4):2799–2817, 2015.

[61] C Hamzacebi. Improving Artificial Neural Networks Performance in Seasonal Time Series Forecasting. *Information Sciences*, 178(23):4550–4559, 2008.

[62] K R Harrison, B M Ombuki-berman, and A P Engelbrecht. A Radius-Free Quantum Particle Swarm Optimization Technique for Dynamic Optimization Problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 578–585, 2016.

[63] S Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, 1994.

[64] K Hechenbichler and K Schliep. Weighted k-nearest-neighbor Techniques and Ordinal Classification. In *Discussion Paper 399, SFB 386.* Citeseer, 2006.

[65] A Heydari and F Keynia. Prediction of Wind Power Generation through Combining Particle Swarm Optimization and Elman Neural Network (El-PSO). *International Energy Journal*, 15(2), 2016.

[66] K W Hipel and A I McLeod. *Time Series Modelling of Water Resources and Environmental Systems*, volume 45 of *Developments in Water Science.* Elsevier, 1994.

[67] G C Homans. Social Behavior as Exchange. *American Journal of Sociology*, 1958.

[68] K Hornik, M Stinchcombe, and H White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366, 1989.

[69] X Hu and R C Eberhart. Tracking Dynamic Systems with PSO: where's the Cheese. In *Proceedings of the Workshop on Particle Swarm Optimization*, pages 80–83, 2001.

[70] X Hu and R C Eberhart. Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1666–1670, 2002.

[71] D Hunter, H Yu, M S Pukish III, J Kolbusz, and B M Wilamowski. Selection of Proper Neural Network Sizes and Architectures –A comparative Study. *IEEE Transactions on Industrial Informatics*, 8(2):228–240, 2012.

[72] D R Hush, B Horne, and J M Salas. Error Surfaces for Multilayer Perceptrons. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1152–1161, 1992.

[73] A M Ibrahim and N H El-Amary. Particle Swarm Optimization Trained Recurrent Neural Network for Voltage Instability Prediction. *Journal of Electrical Systems and Information Technology*, 2017.

[74] S Iqbal, X Zang, Y Zhu, and X Liu. Introducing Undergraduate Electrical Engineering Students to Chaotic Dynamics: Computer Simulations with Logistic Map and Buck Converter. In *Proceedings of IEEE Modelling Symposium*, pages 47–52, 2014.

[75] 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*, pages 36–40, 1999.

[76] G K Jha, P Thulasiraman, and R K Thulasiram. PSO Based Neural Network for Time Series Forecasting. In *Proceedings of IEEE International Joint Conference on Neural Networks*, pages 1422–1427, 2009.

[77] M Jordan. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In *Proceedings of the 8th Annual Conference of the Cognitive Science Society (Erlbaum, Hillsdale, NJ)*, 1986.

[78] J Kennedy. Small Worlds and Mega-minds: Effects of Neighborhood Topology on Particle Swarm Performance. *Proceedings of the IEEE Congress on Evolutionary Computation*, 3:1931–1938, 1999.

[79] J Kennedy and R Mendes. Population Structure and Particle Swarm Performance. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 2, pages 1671–1676, 2002.

[80] J Kennedy and R Mendes. Neighborhood Topologies in Fully Informed and Best-of-neighborhood Particle Swarms. *... on Systems Man and Cybernetics Part ...*, 2006.

[81] James Kennedy. The Particle Swarm: Social Adaptation of Knowledge. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 303–308, 1997.

[82] V Kurbalija. *Time Series Analysis and Prediction using Case Based Reasoning Technology*. PhD thesis, University of Novi Sad, 2009.

[83] A Laepes and R Farben. Nonlinear Signal Processing using Neural Networks: Prediction and System Modelling. In *Technical Report, Los Alamos National Laboratory, Los Alamos, NM*, 1987.

[84] I A Lawal, S A Abdulkarim, M K Hassan, and J M Sadiq. Improving HSDPA Traffic Forecasting Using Ensemble of Neural Networks. In *Proceedings of IEEE International Conference on Machine Learning and Applications*, pages 308–313, 2016.

[85] Y A LeCun, L Bottou, G B Orr, and K Müller. Efficient Backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer, 2012.

[86] C Lemke. *Combinations of Time Series Forecasts : When and Why Are They Beneficial?* PhD thesis, Bournemouth University, 2010.

[87] C Li, J C Sprott, and W Thio. Linearization of the Lorenz System. *Physics Letters A*, 379(10):888–893, 2015.

[88] X Li and K H Dam. Comparing Particle Swarms for Tracking Extrema in Dynamic Environments. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 3, pages 1772–1779, 2003.

[89] Q Lv and J Yu. Self-organizing Feature Map Neural Network based on Particle Swarm Optimizer and its Application. *Control and Decision*, 20(10):1115, 2005.

[90] S Magnussen and E Tomppo. The k-nearest Neighbor Technique with Local Linear Regression. *Scandinavian Journal of Forest Research*, 29(2):120–131, 2014.

[91] S Makridakis and M Hibon. ARMA Models and the Box–Jenkins Methodology. *Journal of Forecasting*, 16(3):147–163, 1997.

[92] K M Malan and A P Engelbrecht. Algorithm Comparisons and the Significance of Population Size. In *Proceedings of Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 914–920, 2008.

[93] H B Mann and D R Whitney. On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[94] M Masdari, F Salehi, M Jalali, and M Bidaki. A Survey of PSO-based Scheduling Algorithms in Cloud Computing. *Journal of Network and Systems Management*, 25(1):122–158, 2017.

[95] R Mendes, P Cortez, M Rocha, and J Neves. Particle Swarms for Feedforward Neural Network Training. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pages 1895–1899, 2002.

[96] A A Mohammad, Z Sohrab, L Ali, E Ali, and E Ioannis. Reservoir Permeability Prediction by Neural Networks Combined with Hybrid Genetic Algorithm and Particle Swarm Optimization. *Geophysical Prospecting*, 61(3):582–598, 2013.

[97] E Momeni, D J Armaghani, M Hajihassani, and M Amin. Prediction of Uniaxial Compressive Strength of Rock Samples using Hybrid Particle Swarm Optimization-based Artificial Neural Networks. *Measurement*, 60:50–63, 2015.

[98] R W Morrison. Performance Measurement in Dynamic Environments. In *Proceedings of the Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, number 5–8, 2003.

[99] K R Müller, A J Smola, G Rätsch, B Schölkopf, J Kohlmorgen, and V Vapnik. Predicting Time Series with Support Vector Machines. In *Proceedings of International Conference on Artificial Neural Networks. Springer*, pages 999–1004, 1997.

[100] K Nam and T Schaefer. Forecasting International Airline Passenger Traffic using Neural Networks. *Logistics and Transportation Review*, 1995.

[101] P S G D Neto, G G Petry, R L J Aranildo, and T A E Ferreira. Combining Artificial Neural Network and Particle Swarm System for Time Series Forecasting. In *Proceedings of IEEE International Joint Conference on Neural Networks*, pages 2230–2237, 2009.

[102] G Pampara and A P Engelbrecht. Towards a Generic Computational Intelligence Library: Preventing Insanity. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1460–1467, 2015.

[103] U Paquet and A P Engelbrecht. Training Support Vector Machines with Particle Swarms. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pages 1593–1598, 2003.

[104] V K Pathirana. *Nearest Neighbor Foreign Exchange Rate Forecasting with Mahalanobis Distance.* Phd thesis, University of South Florida, 2015.

[105] D Petković, M Gocic, S Shamshirband, S N Qasem, and S Trajkovic. Particle Swarm Optimization-based Radial Basis Function Network for Estimation of Reference Evapotranspiration. *Theoretical and Applied Climatology*, 125(3-4):555–563, 2016.

[106] R Poli. Analysis of the Publications on the Applications of Particle Swarm Optimisation. *Journal of Artificial Evolution and Applications*, 2008.

[107] M Qi and G P Zhang. An Investigation of Model Selection Criteria for Neural Network Time Series Forecasting. *European Journal of Operational Research*, 132(3):666–680, 2001.

[108] A Rakitianskaia. *Using Particle Swarm Optimisation to Train Feedforward Neural Networks in Dynamic Environments.* Msc thesis, University of Pretoria, 2011.

[109] A Rakitianskaia and A P Engelbrecht. Cooperative Charged Particle Swarm Optimiser. In *IEEE Congress on Evolutionary Computation*, pages 933–939, 2008.

[110] A Rakitianskaia and A P Engelbrecht. Training Neural Networks with PSO in Dynamic Environments. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 667–673, 2009.

[111] A Rakitianskaia and A P Engelbrecht. Saturation in PSO Neural Network Training: Good or Evil? In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 2, pages 125–132, 2015.

[112] A S Rakitianskaia and A P Engelbrecht. Training Feedforward Neural Networks with Dynamic Particle Swarm Optimisation. *Swarm Intelligence*, 6(3):233–270, 2012.

[113] M M Rashidi, M Ali, N Freidoonimehr, and F Nazari. Parametric Analysis and Optimization of Entropy Generation in Unsteady MHD Flow over a Stretching Rotating Disk using Artificial Neural Network and Particle Swarm. *Energy*, 55:497–510, 2013.

[114] M Riedmiller. Advanced Supervised Learning in Multi-layer Perceptrons-from Backpropagation to Adaptive Learning Algorithms. *Computer Standards and Interfaces*, 16(3):265–278, 1994.

[115] M Riedmiller. Rprop - Description and Implementation Details. Technical Report January, 1994.

[116] M Riedmiller and H Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proeedings of IEEE Internation Conference on Neural Networks*, pages 586–591, 1993.

[117] A Röbel. The Dynamic Pattern Selection Algorithm: Effective Training and Controlled Generalization of Backpropagation Neural Networks. In *Technical Report, Tech- nische Universitat Berlin*, 1994.

[118] F F Rodríguez, S S Rivero, and J Andrada-Félix. Technical Analysis in Foreign Exchange Markets: Linear versus Nonlinear Trading Rules. 2000.

[119] N I Sapankevych and R Sankar. Time Series Prediction using Support Vector Machines: A Survey. *IEEE Computational Intelligence Magazine*, 4(2), 2009.

[120] M L Shahreza, D Moazzami, B Moshiri, and M R Delavar. Anomaly Detection using a Self-organizing Map and Particle Swarm Optimization. *Scientia Iranica*, 18(6):1460–1468, 2011.

[121] R Sharda and R B Patil. Connectionist Approach to Time Series Prediction: An Empirical Test. *Journal of Intelligent Manufacturing*, 3(5):317–323, 1992.

[122] K G Sheela and S N Deepa. Review on Methods to Fix Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering*, 2013.

[123] Y Shi and R Eberhart. A Modified Particle Swarm Optimizer. In *Proceedings of the International Conference on Evolutionary Computation*, 1998.

[124] Y Shi and R C Eberhart. Parameter Selection in Particle Swarm Optimization. Technical report, 1998.

[125] Y Shi and R C Eberhart. Empirical Study of Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 1945–1950, 1999.

[126] Y Shi and R C Eberhart. Empirical Study of Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, NJ*, volume 3, pages 1948–1950, 1999.

[127] R Sitte and J Sitte. Analysis of the Predictive Ability of Time Delay Neural Networks Applied to the S&P 500 Time Series. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Review)*, 30(4):568–572, 2000.

[128] H A Soodi and A M Vural. STATCOM Estimation Using Back-Propagation, PSO, Shuffled Frog Leap Algorithm, and Genetic Algorithm Based Neural Networks. *Computational Intelligence and Neuroscience*, 2018.

[129] A Suresh, K V Harish, and N Radhika. Particle Swarm Optimization over Back-propagation Neural Network for Length of Stay Prediction. *Procedia Computer Science*, 46:268–275, 2015.

[130] Z Tang, C D Almeida, and P A Fishwick. Time Series Forecasting using Neural Networks vs. Box- Jenkins Methodology. *Simulation*, 57(5):303–310, 1991.

[131] Z Tang, C de Almeida, and Pl A Fishwick. Time Series Forecasting using Neural Networks vs. Box-Jenkins Methodology. *Simulation*, 57(5):303–310, 1991.

[132] Z Tang and P A Fishwick. Feedforward Neural Nets as Models for Time Series Forecasting. *ORSA Journal on Computing*, (4):374–385, 1993.

[133] R Taormina and K Chau. Neural Network River Forecasting with Multi-objective Fully Informed Particle Swarm Optimization. *Journal of Hydroinformatics*, 17(1):99–113, 2015.

[134] H. Tong. *Threshold Models in Non-Linear Time Series Analysis*, volume 21. Springer Science & Business Media, 2012.

[135] C Ulbricht. Handling Time-warped Sequences with Neural Networks. In *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, pages 180–189, 1996.

[136] C Ulbricht, G Dorffner, and S Canu. *Mechanisms for Handling Sequences with Neutral Networks.* Österreichisches Forschungsinstitut für Artificial Intelligence, 1992.

[137] N J Unger, B M Ombuki-Berman, and A P Engelbrecht. Cooperative Particle Swarm Optimization in Dynamic Environments. In *Symposium on Swarm Intelligence (SIS)*, pages 172–179. IEEE, 2013.

[138] F Van den Bergh. *An Analysis of Particle Swarm Optimizers.* PhD thesis, University of Pretoria, 2001.

[139] F Van den Bergh and A P Engelbrecht. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal*, 2000(26):84–90, 2000.

[140] F Van den Bergh and A P Engelbrecht. Effects of Swarm Size on Cooperative Particle Swarm Optimisers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 892–899, 2001.

[141] F Van den Bergh and A P Engelbrecht. Training Product Unit Networks using Cooperative Particle Swarm Optimisers. In *Proceedings of IEEE International Joint Conference on Neural Networks*, volume 1, pages 126–131, 2001.

[142] F Van den Bergh and A P Engelbrecht. A Cooperative Approach to Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

[143] A B Van Wyk. *An Analysis of Overfitting in Particle Swarm Optimised Neural Networks.* Msc thesis, University of Pretoria, 2015.

[144] A B van Wyk and A P Engelbrecht. Analysis of Activation Functions for Particle Swarm Optimised Feedforward Neural Networks. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 2, pages 423–430, 2016.

[145] A Waibel, T Hanazawa, G Hinton, K Shikano, and K J Lang. Phoneme Recognition using Time-delay Neural Networks. In *Readings in Speech Recognition*, pages 393–404. Elsevier, 1990.

[146] M P Wallace. Neural Networks and Their Application to Finance. *Business Intelligence Journal*, 1:pp. 67–77, 2008.

[147] K Weicker. An Analysis of Dynamic Severity and Population Size. In *Proceedings of International Conference on Parallel Problem Solving from Nature*, pages 159–168. Springer, 2000.

[148] L F A Wessels and E Barnard. Avoiding False Local Minima by Proper Initialization of Connections. *IEEE Transactions on Neural Networks*, 3(6):899–905, 1992.

[149] B M Wilamowski, S Iplikci, O Kaynak, and MÖ Efe. An algorithm for Fast Convergence in Training Neural Networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, 2001.

[150] G Wu, G Liu, and M Hao. The Analysis of Emotion Recognition from GSR Based on PSO. In *Proceedings of International Symposium on Intelligence Information Processing and Trusted Computing*, pages 360–363. IEEE, 2010.

[151] N Yalcin, G Tezel, and C Karakuzu. Epilepsy Diagnosis using Artificial Neural Network Learned by PSO. *Turkish Journal of Electrical Engineering & Computer Sciences*, 23(2):421–432, 2015.

[152] W Yeh, Y Yeh, P Chang, Y Ke, and V Chung. Forecasting Wind Power in the Mai Liao Wind Farm Based on the Multi-layer Perceptron Artificial Neural Network Model with Improved Simplified Swarm Optimization. *International Journal of Electrical Power & Energy Systems*, 55:741–748, 2014.

[153] G Zhang, B E Patuwo, and M Y Hu. Forecasting with Artificial Neural Networks:: The State of the Art. *International Journal of Forecasting*, 14(1):35–62, 1998.

[154] G P Zhang. A Neural Network Ensemble Method with Jittered Training Data for Time Series Forecasting. *Information Sciences*, 177(23):5329–5346, 2007.

Appendix A

Acronyms

A list of acronyms used throughout the text is given in this appendix.

AR	Auto Regressive
ARCH	Auto Regressive Conditional Heteroscedasticity
AWS	Australian Wine Sales Time Series
BP	Backpropagation
CI	Computational Intelligence
CIlib	Computational Intelligence Library
CMF	Collective Mean Fitness
CPSO	Charged Particle Swarm Optimization
CCPSO	Cooperative Charged Particle Swarm Optimization
CQSO	Cooperating Quantum Particle Swarm Optimization
DE	Dynamic Environment
DMT	Daily Minimum Temperature Time Series
DOP	Dynamic Optimization Problem
EC	Evolutionary Computation
FNN	Feedforward Neural Network
GA	Genetic Algorithm
GARCH	Generalized Auto Regressive Conditional Heteroscedasticity
HIT	Hourly Internet Traffic Time Series
IAP	International Airline Passengers Time Series

***k*-NN**	k-Nearest Neighbour
LM	Logistic Map Time Series
MG	Mackay Glass Time Series
MPSO	Multiswarm Particle Swarm Optimization
MRNN	Multi-Recurrent Neural Network
MSE	Mean Square Error
NAR	Non-linear Auto Regressive
NN	Artificial Neural Network
PSO	Particle Swarm Optimization
PCX	Parent Centric Crossover
QPSO	Quantum Particle Swarm Optimization
RBF	Radial Bases Function Network
RNN	Recurrent Neural Network
RPROP	Resilient Propagation
SAM	Sunspot Annual Measure Time Series
SSE	Sum Square Error
S&P	Standard and Poor Time Series
SRNN	Simple Recurrent Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
TAR	Threshold Autoregressive
TDNN	Time Delay Neural Network
USD	United State Death Time Series

Appendix B

Symbols

B.1 Chapter 2

This appndix lists the mathematical symbols, with their definations, used throughout this thesis. Each section lists the symbols used per chapter. Symbols in bold text indicate vectors.

t	Time step
Y	Time series
T	Trend
S	Seasonal variation
R	Random variation
d	Prediction horizon (lag)
ϵ	Residual error
$\mathcal{N}$	Normal distribution
σ^2	Variance
$\hat{y}$	Forecasted value
ϕ	Model parameter
θ	Model parameter
(p, q)	Order of ARIMA model
$\eth$	Differencing of degree
m	k-NN embedding dimension

τ	k-NN sampling frequency
k	Number of nearest neighbours
$\mho$	Distance function

B.2 Chapter 3

φ	Output layer activation function
λ	Hidden layer activation function
$\boldsymbol{v}$	Hidden to output layer weight vector
$\boldsymbol{w}$	Input to hidden layer weight vector
β	Bias
p	Size of the sliding window
f_L	Linear function
n_t	Special type of neurons with delayed patterns
η	Learning rate
Δ	Update value
$\mathbf{x}$	Position vector
$\mathbf{z}$	Neural network input layer
$\mathbf{s}$	Neural network state layer
$\mathcal{E}$	Neural network Error
$fanin$	Number of connections into a neuron

B.3 Chapter 4

$\mathbf{v}$	Velocity vector
ω	Inertia weight
c_1	PSO cognitive acceleration coefficient
c_2	PSO social acceleration coefficient
$\mathbf{r}$	Vectors of random values
$\otimes$	Component-wise multiplication operator
$\mathbf{y}$ or *pbest*	Best position so far visited by a particle
$\hat{\mathbf{y}}$ or *nbest*	Best position found within the neighborhood

$\vec{\varphi}$	Vector of time-dependent control parameters

B.4 Chapter 5

d	Dimension
dom	Domain
$\vec{x}^*$	Minimum found
$\mathbf{r}_{cloud}$	Vector of quantum radius
r_{excl}	Exclusion radius
$r_{conv.}$	Convergence radius

B.5 Chapter 6

$x(t), y(t), z(t)$	States of the Lorenz system at time t
r, σ, b	Constant parameters of the Lorenz system
K	Sampling time for Lorenz system
τ, a, b, c	Constant parameters of MG time series
G	Constant parameter of the Logistic map equation
N	Number of observations in dataset
w	Window size
s	Step value for sliding the window over the dataset
f	Frequency of change
T	Numbers of iterations to traverse the entire dataset
N_w	The total number of weights and biases
CMF	Cummulative mean fitness
$F(t)$	Measure of quality of the solution
ρ	Generalization factor
G_E	Generalization error
T_E	Training error
μ	Mean
H_0	Null hypothesis
H_1	Alternative hypothesis

Appendix C

Derived publications

This appendix lists all papers that are currently under review, that led to, or are derived from the work presented in this thesis.

Abdulkarim, S.A., and Engelbrecht, A.P.: Time Series Forecasting with Feedforward Neural Networks Trained using Particle Swarm Optimizers for Dynamic Environments. Under review.

Abdulkarim, S.A., and Engelbrecht, A.P.: Time Series Forecasting using Neural Networks: Are Recurrent Connections Necessary. Under review.