UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Using Particle Swarm Optimisation to Train Feedforward Neural Networks in Dynamic Environments

by

Anna Rakitianskaia

Submitted in partial fulfillment of the requirements for the degree

Master of Science (Computer Science)

in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria, Pretoria

December 2011

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Using Particle Swarm Optimisation to Train Feedforward Neural Networks in Dynamic Environments

by

Anna Rakitianskaia
E-mail: myearwen@gmail.com

Abstract

The feedforward neural network (NN) is a mathematical model capable of representing any non-linear relationship between input and output data. It has been succesfully applied to a wide variety of classification and function approximation problems. Various neural network training algorithms were developed, including the particle swarm optimiser (PSO), which was shown to outperform the standard back propagation training algorithm on a selection of problems. However, it was usually assumed that the environment in which a NN operates is static. Such an assumption is often not valid for real life problems, and the training algorithms have to be adapted accordingly. Various dynamic versions of the PSO have already been developed. This work investigates the applicability of dynamic PSO algorithms to NN training in dynamic environments, and compares the performance of dynamic PSO algorithms to the performance of back propagation. Three popular dynamic PSO variants are considered. The extent of adaptive properties of back propagation and dynamic PSO under different kinds of dynamic environments is determined. Dynamic PSO is shown to be a viable alternative to back propagation, especially under the environments exhibiting infrequent gradual changes.

Supervisor	**:** Prof. A. P. Engelbrecht
Department	**:** Department of Computer Science
Degree	**:** Master of Science

...*Ergo,* God exists.

Argumentum Ornithologicum, by Jorge Luis Borges (1952)

Acknowledgements

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

- Professor A. P. Engelbrecht, my supervisor, for his insight, guidance and support.
- My father, a scientist, for all the research-related advice and constant motivation.
- My mother, for her patience, understanding, love and cooking.
- My sister Anastassia, for the music.
- Julien Duhain and Olosegun Olorunda, for their friendship.
- My colleagues, for their useful criticism and all the involved academic discussions.
- My friends, both real and virtual ones, for providing me with a life outside of research and smiling at me.

This thesis was produced exclusively with the use of the following open source and freeware software tools. Special thanks to the authors of these superb software packages:

Typesetting using $\LaTeX\,2_\varepsilon$; Bibliographic references maintained using BibTeX; Graph production using gnuplot.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Even though our senses fool our consciousness into believing that we exist in a 3-dimensional world, it is the fourth dimension - time - that governs the Universe and makes it curve. The state of the world we live in changes every instant, and every instant we adapt to it - consciously or unconsciously, depending on the nature and extent of the change. We put on an extra layer of clothes when it is chilly outside, and take a train when our flight has been cancelled due to a recent volcano eruption.

Due to the nature of the Universe, it is never correct to assume that our environment is static. Even though such crude approximation may work for a short while, it will always fail in the long term. Both measurable and hidden parameters of a problem tend to change over time, causing the once found solutions to loose precision and deteriorate. Examples of dynamic environments are the stock exchange, road congestion due to traffic, different price markets, such as electricity or food markets, etc. From a mathematical perspective, a dynamic environment can be visualised as a function with floating optima. An optimum may change its position and value, existing optima may disappear and new optima may emerge. Alternatively, for classification problems it is decision boundaries separating different classes that change over time. In both cases, the task of the optimisation algorithm becomes more difficult, since the algorithm should not only find the optimal solution, but also detect environmental changes and promptly adapt to them, which might entail dismissing the old solution entirely. This is why current developments in computational intelligence (CI), or nature-inspired mechanisms that simulate intelligent behavior [32], try to encompass the temporal aspects of the problems they attempt to solve.

A successful example of a new approach to CI is particle swarm optimisation (PSO), a population-based optimisation technique that models social behaviour of a bird flock in order to traverse the search space and find an optimal solution [65, 32]. Due to the success of the PSO in static environments [65], numerous variants of PSO that cater for dynamic changes have been developed to date, including the simple restarting PSO [31], the charged PSO based on electrostatic principles [7], and the quantum PSO based on the model of an atom [9], amongst others.

However, one of the oldest fields of CI research, namely neural networks (NNs) [4, 28, 47, 91, 101, 142] – powerful mathematical models inspired by the human brain and capable of representing any non-linear relationship between input and output data – have remained conservative towards the emerging field of optimisation in dynamic environments. It has been assumed that the standard NN training algorithms based on gradient descent are implicitly dynamic [59], and if the NN fails to adapt to the changes, then restarting the training process would be the most efficient solution. In order to avoid re-training, redundancy in the form of ensemble classifiers has also been proposed [19, 111, 114, 115, 116]. The chances of obtaining at least one acceptable solution using ensemble classifiers are increased by training a number of separate NNs on the same problem over different time periods. However, the ensemble approach does not offer any training algorithm improvements to make each classifier aware of dynamic changes.

NNs are widely used in real life [25, 124, 141], and it is necessary to ensure that NNs can be effectively trained in dynamic environments. PSO has been successfully applied to NN training before [67, 34, 44, 83, 118], and in this work the applicability of dynamic versions of PSO to NN training in dynamic environments is studied. The main focus of this work is on classification problems with dynamic decision boundaries, further referred to as dynamic classification problems. The behaviour of both the standard gradient descent back propagation and various dynamic PSOs on different dynamic problems is analysed.

The rest of the chapter is outlined as follows. Section 1.1 lists the main objectives of this thesis. Section 1.2 summarises the original contributions of this work. Section 1.3 outlines the structure of the rest of this thesis.

1.1 Objectives

The primary objectives of this thesis are summarised as follows:

- To provide an overview of the CI techniques used in this work, namely neural networks and particle swarm optimisation.
- To provide an overview of dynamic environments and dynamic classification problems.
- To provide an overview of the relevant PSO approaches to dynamic optimisation.
- To apply dynamic PSO algorithms to NN training on dynamic classification problems.
- To analyse the behaviours and performance that back propagation and the dynamic PSO algorithms exhibit when applied to NN training on dynamic classification problems.
- To identify dynamic environment types for which every algorithm considered is best suited.

1.2 Contributions

The novel contributions of this thesis include the following:

- The first analysis of the applicability of dynamic PSO algorithms to NN training on dynamic classification problems.
- An empirical analysis of the adaptive properties of back propagation.
- An empirical comparison of back propagation and dynamic PSO algorithms applied to NN training on dynamic classification problems.
- The discovery that dynamic PSO training algorithms are more efficient under infrequent gradual changes, and that back propagation is more efficient under frequent abrupt changes.
- The conclusion that dynamic PSO training algorithms are preferrable over back propagation on dynamic classification problems with rugged error lanscape, and where multiple new decision boundaries may appear and disappear.
- The conclusion that dynamic PSO training algorithms are more sensitive than back propagation to specific properties of a dynamic environment, such as frequency and severity of change. However, if properly optimised for a specific dynamic environment, the dynamic PSO training algorithms are likely to perform better than back propagation.

1.3 Thesis Outline

The remainder of the thesis is organised as follows:

- **Chapter** 2 discusses PSO in detail.

- **Chapter 3** discusses supervised feedforward NNs. The back propagation training algorithm is described, and major performance issues are outlined.
- **Chapter 4** discusses dynamic environments. A few real-life examples of dynamic problems are described, and different attributes of dynamic classification problems are discussed. Existing PSO approaches to dynamic optimisation are also discussed.
- **Chapter 5** is dedicated to existing approaches to train NNs in dynamic environments. The drawbacks of back propagation are discussed, and alternative training algorithms are suggested. The problems of architecture selection, parameter optimisation, and overfitting in the context of dynamic environments are also discussed.
- **Chapter 6** presents an empirical study conducted for this thesis. The experimental procedure is discussed in detail, experimental results are presented and analysed, and conclusions are given.
- **Chapter 7** provides a summary of the thesis, and gives a list of possible future research directions.

The following appendices are included, containing a number of lists with relevant information for quick referencing purposes:

- **Appendix A** provides a list of the acronyms used and defined in this work, as well as their associated definitions.
- **Appendix B** lists and defines the mathematical symbols used in this work, categorised according to the relevant chapter in which they appear.
- **Appendix C** lists the publications derived from this work.

Chapter 2

Particle Swarm Optimisation

The intelligence of the creature known as a crowd is the square root of the number of people in it.

Terry Pratchett

PSO is a relatively new optimisation technique. Kennedy and Eberhart formally introduced PSO in 1995 [65], much later than population-based evolutionary algorithms such as the genetic algorithm (GA) (first proposed by A. S. Fraser [36] in 1957). PSO is a population-based iterative search algorithm that manipulates a pool of potential solutions (particles) in order to find an optimum. The search is conducted by imitating the social behaviour of a bird flock.

Observations of bird flocks, schools of fish, bee swarms, and animal herds have shown that the collective effort of a group is usually more rewarding than individual effort [51]. Suppose a certain task has to be achieved. Each individual within a group has a certain capability of reaching the goal. While working in a group, the behaviour of an individual is guided not only by the individuals' perception of how to achieve the goal, but also by the social dynamics. All individuals within a group share the experience of pursuing a common goal, and each individual learns not only from its own experience, but also from the experience of its neighbours. This speeds up the search process significantly.

This kind of social behaviour inspired the creation of the PSO algorithm, which is discussed in this chapter. The rest of the chapter is structured as follows: Section 2.1 provides a formal definition of the PSO, and Section 2.2 describes various swarm information sharing strategies. PSO parameters and their impact on algorithm efficiency are discussed in Section 2.3. Section 2.4 concludes the chapter.

2.1 Basic PSO algorithm

PSO operates on a group (referred to as a *swarm*) of individuals. Each individual, referred to as a *particle*, represents a candidate solution to the optimisation problem. For an n-dimensional optimisation problem, a particle is represented by an n-dimensional vector, $\vec{x}$, also referred to as particle *position*. Every particle has a *fitness* value, which indicates the quality of the candidate solution represented by the particle. The n-dimensional search space of the problem is the environment in which the swarm operates. In addition to a position within the search space, each particle possesses a velocity vector $\vec{v}$, which determines the step size and direction of the particle's movement. Social interaction is imitated by forming *neighbourhoods* within a swarm. Each particle stores its own best position found so far, and can also query neighbouring particles for the best position as discovered by the neighbourhood thus far. Various PSO neighbourhood topologies have been proposed in the literature and applied in practice (refer to Section 2.2 for further details). The structure and size of the neighbourhood determines the way in which information is shared between the particles.

PSO searches for an optimum by moving the particles through the search space. At each time step, t, the position $\vec{x}_y(t)$ of particle y is modified by adding the particle velocity $\vec{v}_y(t)$ to the previous position vector:

$$\vec{x}_y(t) = \vec{x}_y(t-1) + \vec{v}_y(t) \tag{2.1}$$

The velocity vector determines the step size and direction of the particle. The velocity update equation is given by

$$\vec{v}_y(t) = \omega\vec{v}_y(t-1) + c_1\vec{r_1}(\vec{x}_{pbest,y}(t) - \vec{x}_y(t)) + c_2\vec{r_2}(\vec{x}_{nbest,y}(t) - \vec{x}_y(t)) \tag{2.2}$$

where ω is the inertia weight [109], controlling the influence of previous velocity values on the new velocity; c_1 and c_2 are acceleration coefficients used to scale the influence of the *cognitive* (second term of Equation (2.2)) and *social* (third term of Equation (2.2)) components; $\vec{r_1}$ and $\vec{r_2}$ are vectors with each component sampled from a uniform distribution $U(0, 1)$; $\vec{x}_{pbest,y}(t)$ is the *personal best* of particle y, or, in other words, the best position encountered by this particle so far; similarly, $\vec{x}_{nbest,y}(t)$ is the *neighbourhood best* of particle y, or the best position found by any of the particles in the neighbourhood of particle y. Thus, each particle is attracted to both the best position encountered by itself so far, as well as the overall best position found by the neighbourhood so far. A maximum velocity $\vec{V}_{max}$ [108] is sometimes used to limit (or clamp) particle velocity in every dimension. Velocity clamping is done to prevent particles from traversing the search space too fast, since unreasonably big steps prevent particles from exploiting good regions. $\vec{V}_{max}$ is enforced by restricting $\vec{v}_y(t)$ per dimension:

$$v_{yl}(t) = \begin{cases} V_{max,l} & \text{if } v_{yl}(t) > V_{max,l} \\ -V_{max,l} & \text{if } v_{yl}(t) < -V_{max,l} \\ v_{yl}(t) & \text{otherwise} \end{cases} \tag{2.3}$$

For more information on various PSO parameters, refer to Section 2.3.

The PSO algorithm is usually stopped when the quality of the found solution is satisfactory, or when the maximum number of iterations or fitness function evaluations has been reached [32].

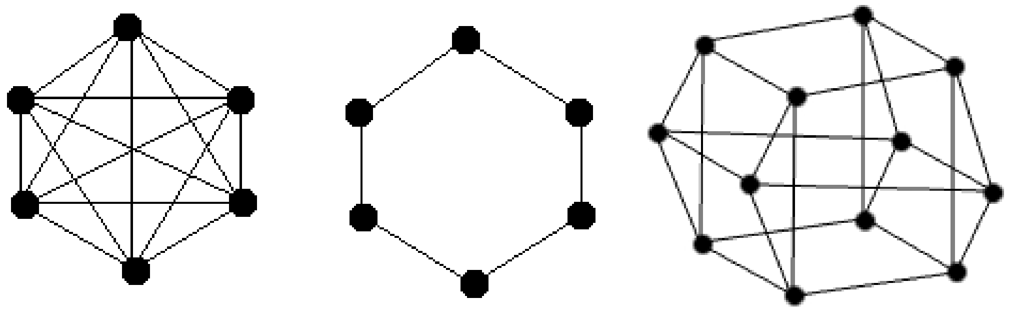
The PSO algorithm is outlined in Algorithm 2.1.

```
1. Set iteration count $t = 0$
2. Initialise the swarm $P(t)$ of $S_P$ particles, such that $\vec{x}_y(t) \sim U(X^n)$, where $X^n$ is the search space
of the problem.
3. Set $\vec{x}_{pbest,y}(t)$ and $\vec{x}_{nbest,y}(t)$ of each particle to its current position, $\vec{x}_y(t)$.
4. Set $\vec{v}_y(t)$ of each particle to $\vec{0}$
repeat:
    for each particle $y = 1, \ldots, S_P$ do:
        Evaluate fitness $f(\vec{x}_y)$ of particle $y$
        Set the personal best position:
        if $f(\vec{x}_y(t)) > f(\vec{x}_{pbest,y}(t))$ then
            $\vec{x}_{pbest,y}(t) = \vec{x}_y(t)$
        end if
        Set the neighbourhood best position:
        if$f(\vec{x}_{pbest,y}(t)) > f(\vec{x}_{nbest,y}(t))$then
            $\vec{x}_{nbest,y}(t) = \vec{x}_{pbest,y}(t)$
        end if
    end for
    for each particle $y = 1, \ldots, S_P$ do:
        Update particle velocity $\vec{v}_y(t)$ using equation (2.2)
        If applicable, clamp $\vec{v}_y(t)$ to $\vec{V}_{max}$ using equation (2.3)
        Update particle position $\vec{x}_y(t)$ using equation (2.1)
    end for
    $t = t + 1$
until stopping criteria are met.
```

Algorithm 2.1: Outline of the PSO algorithm with synchronous update

2.2 Neighborhood Topologies

As mentioned in the previous section, each particle in the swarm moves towards both the best position encountered by itself so far, as well as the overall best position found by its neighbourhood. A particle's neighbourhood is determined topologically rather than spatially, meaning that the distance between particles is determined by particles' indices and not the actual position in the search space [64]. The preference is given to topological structure, since, when a particle swarm converges, the spatial distances between particles tend to zero, thus disrupting spatial neighbourhoods.



(a) Star topology (b) Ring topology (c) Von Neumann topology

Figure 2.1: PSO Neighbourhood Topologies

This section discusses three conventional PSO neighbourhood topologies (sometimes also referred to as *information sharing strategies*), namely the star topology, the ring topology, and the Von Neumann topology. These topologies are also graphically depicted in Figure 2.1.

2.2.1 Star Topology

The star topology was first introduced by Kennedy and Eberhart [65]. For the star topology, all particles share information about the search space with every other particle in the swarm. The star topology is visualised as a fully interconnected network of nodes, as illustrated in Figure 2.1(a). The neighbourhood of each particle is the entire swarm. Consequently, $\vec{x}_{nbest,y}(t)$ used in equation (2.2) is the same for all particles (i.e. *global* best), since only one neighbourhood exists. Thus, all particles, apart from their personal best positions, imitate the same neighbourhood best particle, which limits exploration and facilitates fast convergence of the swarm.

2.2.2 Ring Topology

The ring topology was first introduced by Kennedy and Eberhart [65]. For the ring neighbourhood topology, the neighbourhood of a particle y consists of m particles in the immediate topological proximity of particle y. For example, if $m = 2$, then the neighbourhood of particle y consists of two other particles, $y - 1$ and $y + 1$. An example of a ring topology is shown in Figure 2.1(b). Fewer connections between particles result in a number of overlapping neighbourhoods. Every neighbourhood has its own neighbourhood best, thus particles from different neighbourhoods are attracted to different fruitful areas of the search space, which facilitates exploration. Since the neighbourhoods are overlapping, the swarm is still likely to converge, but at a slower speed: Better exploration has a negative effect on the speed of convergence [64, 66].

2.2.3 Von Neumann Topology

The Von Neumann topology was first introduced by Kennedy and Mendes [68]. This neighbourhood topology connects the particles in a grid-like structure such that every particle connects to its four immediate neighbours. The Von Neumann topology can be visualised as a square lattice, the extremities of which are connected. An example of the Von Neumann topology for a 12 particle swarm is depicted in Figure 2.1(c).

Peer *et al.* [93] showed that the Von Neumann neighbourhood topology maintains swarm diversity better due to the fact that the influence of a single particle propagates through the structure slowly, thus making it harder for a single particle to dominate the swarm. It was empirically shown in [79] that PSO utilising the Von Neumann topology (sometimes also referred to as the *fine-grained PSO*) performs well in dynamic environments. The fine-grained PSO managed to outperform PSO with other information sharing strategies on a selection of high-dimensional dynamic problems [79].

2.3 Impact of Parameters

Different properties of the swarm dynamics have to be reinforced for different optimisation problems. For example, it is necessary to reinforce exploration when a function with a rough surface is being minimised in order to avoid premature convergence on a local minimum, and a stronger exploitation ability will promote faster convergence on an optimal solution when a unimodal function with a non-flat surface is considered, or when a promising region that may contain an optimum has been found. The dynamics of the swarm, such as the ability to explore and to exploit, depend not only on the swarm topology, but also on the values chosen for the PSO parameters. In order to achieve optimal performance of the algorithm, these parameters have to be fine-tuned for each specific optimisation problem. This section discusses the standard PSO parameters and their impact on swarm dynamics.

2.3.1 Acceleration parameters

Coefficients c_1 and c_2 in equation (2.2) are referred to as acceleration coefficients. The search trajectory of each particle is influenced by these coefficients, in addition to the inertia weight (see Section 2.3.2). The first one, c_1, controls the impact of the cognitive component on the particle trajectory, and the second one, c_2, controls the impact of the social component.

If $c_2 << c_1$, each is more strongly attracted to its personal best than to the neighbourhood best. The bias towards personal best facilitates individual search by each particle, and reduces the social behaviour of the swarm (i.e. striving towards the same global best position). As a result, exploration is promoted, but the swarm converges slowly, or not at all.

On the other hand, if $c_1 << c_2$, particles are attracted more to the best position in the neighbourhood than to their personal best positions. This may result in premature convergence due to lack of exploration.

Since the main principle of PSO rests on a combination of both personal and social knowledge of the search space, the acceleration coefficients are usually chosen such that $c_1 \approx c_2$. If the values of c_1 and c_2 are too large, then particle velocities accelerate too fast, which may result in swarm divergence. If, on the other hand, the values of c_1 and c_2 are too small, then particles move too slowly, and the swarm may take too long to converge. An empirical study conducted by Kennedy [63] has shown that the values for c_1 and c_2 should be chosen such that $c_1 + c_2 \leq 4$, otherwise particle velocities and positions may explode towards infinity. The behaviour of particle velocities is also largely dependent on the inertia weight, which is discussed below.

2.3.2 Inertia Weight

Another important parameter of the PSO is the inertia weight, denoted as ω. The inertia weight controls the influence of the previous velocity values on the new velocity, as follows from equation (2.2).

If $\omega \geq 1$, particle velocities will grow over time. This facilitates exploration by increasing swarm diversity and the area of the search space covered by the swarm. However, if the velocity becomes too large, the particle's direction will be dominated by its previous velocity, preventing the particle from backtracking to previously found good regions, and possibly causing the particle to leave the boundaries of the search space. This will yield divergent behaviour, which is undesirable.

Studies have shown that $\omega < 1$ does not guarantee swarm convergence, either, since c_1 and c_2 also play a significant role in determining swarm dynamics [21]. In fact, neither the inertia weight nor the acceleration coefficients can be optimised in isolation. Van den Bergh [117] has empirically shown that the swarm exhibits convergent behaviour when c_1, c_2, and ω adhere to the following relation:

$$\omega > \frac{(c_1 + c_2)}{2} - 1 \quad \text{and} \quad \omega \leq 1.$$

2.3.3 Velocity Clamping

The aim of the swarm is to converge on an optimal solution. However, certain combinations of PSO parameters result in growing particle velocities, which yield divergent behaviour [21, 63, 117]. It was suggested by Kennedy and Eberhart [67] to keep the velocity within sensible bounds by putting a limit on the maximum velocity that the particles are allowed to attain. Velocity clamping is carried out by limiting particle velocities in every dimension according to equation (2.3). $\vec{V}_{max}$ is a problem-dependent parameter, chosen empirically [108]. Although $\vec{V}_{max}$ does not have to be set for the standard PSO to work, $\vec{V}_{max}$ is still a useful parameter which essentially limits the step size of particles to a certain maximum. Limiting particle step sizes is a convenient way to control the speed at which the swarm traverses the search space. A small $\vec{V}_{max}$ prevents the particles from traversing the search space fast, thus exploitation is facilitated. A large $\vec{V}_{max}$, on the other hand, allows the particles to develop large velocities and thus rapidly explore.

2.4 Summary

This chapter provided an overview of the PSO algorithm, a population based optimisation technique inspired by the swarming behaviour of bird flocks. The basic PSO algorithm was outlined and discussed, including all of its major characteristics, such as neighbourhood topologies and algorithm parameters. The provided overview of the PSO algorithm was restricted to the concepts used in this thesis. For a more extensive coverage of the PSO, refer to [20, 88, 90, 94].

The next chapter discusses artificial neural networks, and shows how the PSO can be used to train neural networks.

Chapter 3

Artificial Neural Networks

I not only use all the brains that I have, but all that I can borrow.

Woodrow Wilson (1856 - 1924)

NNs are very abstract and simplified mathematical models of the human brain, able to carry out tasks such as pattern recognition, classification, and function approximation [28, 47, 91, 101].

The human brain, as, in fact, any animal brain, is structured as a set of interconnected nerve cells, called neurons. The neurons communicate by sending electrochemical signals through the network of connections, or synapses. A neuron will send the signal through only if a certain internal chemical threshold is exceeded. The connections between actively communicating neurons strengthen, and the unused connections weaken. As a result of this process, the neural network learns the mapping between the inputs as sensed from the environment and sent to the brain for processing, and the desired responses to the environment, or outputs.

NNs were designed to mimic the learning mechanism described above. NNs have the ability to learn any non-linear mapping between given inputs and outputs, which can also be seen as discovering patterns in data. This CI technique has been successfully applied to problems such as face recognition [24], handwritten character recognition [77], and spam filtering [25], amongst many others.

The rest of this chapter provides an overview of the basic concepts related to NNs and the variations of NNs used in the experiments conducted for this study. Section 3.1 describes the functionality of a single neuron. The feedforward NN structure is discussed in Section 3.2. Section 3.3 outlines various algorithms used to train NNs. Performance issues related to NN training are discussed in Section 3.4. Section 3.5 concludes the chapter with some final remarks.

3.1 Artificial Neuron

The basic building blocks of NNs are artificial neurons. An artificial neuron represents a mathematical function that simplistically models a biological neuron. The neuron receives a set of inputs either from the environment or from other neurons, and responds by either firing (sending out a strong signal) or not firing, based on a certain threshold. The output signal is modulated by an activation function internal to the neuron, which allows the neuron to represent a non-linear mapping from $\mathbb{R}^I$ to $[0, 1]$ or $[-1, 1]$, depending on the particular activation function used. Thus, an artificial neuron can represent any linearly separable function by representing the boundary hyperplane that divides the search space into two mutually exclusive subspaces. Figure 3.1 illustrates this model.

A neuron receives n inputs, where each input z_i has an accompanying weight w_i to strengthen or weaken the signal. The neuron calculates the weighted sum of the inputs, and uses an activation function f to produce the output signal u:

$$\begin{aligned} net &= \sum_{i}^{n} z_i w_i \\ u &= f(net) - \theta \end{aligned}$$

The output signal u is also influenced by a threshold value θ, referred to as the *bias*. The

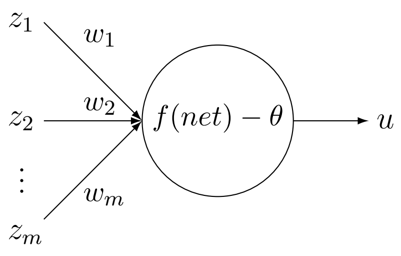


Figure 3.1: An Artificial Neuron

purpose of the bias is to offset the hyperplane or function that the neuron represents from the origin.

Different activation functions that can be used in a neuron are discussed below.

Activation Functions

It is the activation function which determines whether or not a neuron will fire given a specific set of input values, and how strong the output signal is going to be. Three example activation functions are depicted in Figure 3.2.

The simplest activation function is the linear function with a positive slope, where the produced output signal is directly proportional to the received net input signal. A linear activation function, however, does not introduce any non-linearity, thus limiting the function representation capabilities of a neuron.

A simple alternative to the linear function is the step function, which discretises the output. That is, a neuron outputs only one of the two possible values: *e.g.* zero if the weighted sum of inputs is below the threshold, and one otherwise. The step function is not differentiable, which is an important disadvantage, since gradient-based NN training algorithms such as back propagation make use of the activation function derivatives (see Section 3.3.2).

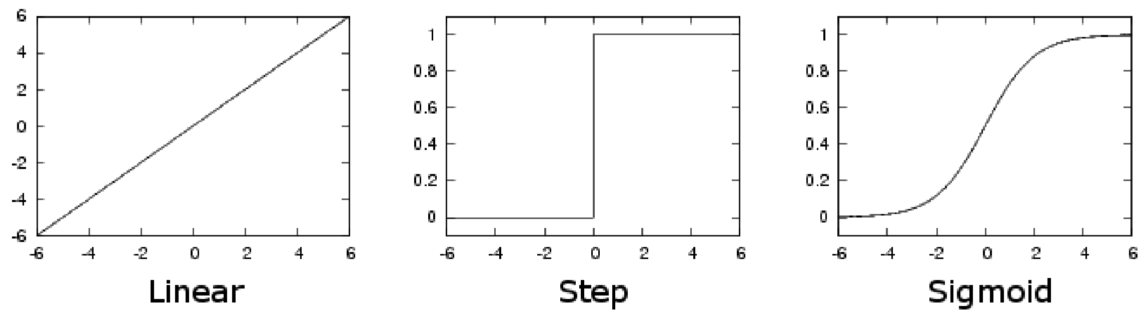


Figure 3.2: Activation Functions

The most commonly used activation function is the sigmoid function, given by

$$f(net - \theta) = \frac{1}{1 + e^{-\lambda(net-\theta)}} \tag{3.1}$$

where λ is a scalar that controls the steepness of the function, usually set to $\lambda = 1$. The sigmoid function is both bounded and continuous, with a range of $(0, 1)$. For all the experiments conducted for this research, the sigmoid function was used. For more information on different activation functions, refer to [1, 48].

3.2 Feedforward Neural Network Structure

A NN is essentially a collection of interconnected neurons aligned in layers. As mentioned in Section 3.1, a single neuron is capable of representing a linearly separable function. In a NN, where many neurons are combined, the information capacity drastically increases. A NN with a large enough number of neurons can represent any non-linear function [52, 62, 76], or, in case of classification, any number of decision boundaries of varying complexity. A number of approaches to structure a collection of neurons were developed [32, 101], but perhaps the most well-known and widely used NN structure in existence is the feedforward neural network (FFNN). An example of a feedforward structure is shown in Figure 3.3. A FFNN is made up of at least three layers: an input layer, a hidden layer, and an output layer.

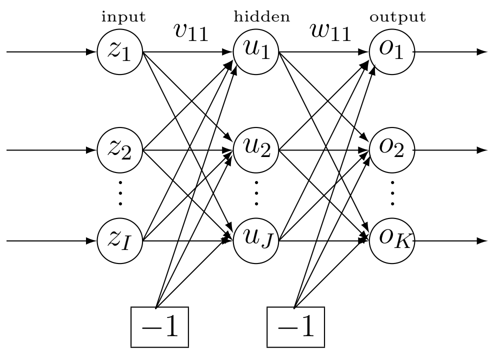


Figure 3.3: FFNN with a single hidden layer

The *input layer* is the entry point to the NN. As the name suggests, it is the neurons in the input layer that receive raw data to be processed by the NN. The function of the input layer is simple: the input layer receives data and sends it through to the hidden layer. Thus, the number of neurons in the input layer is usually equal to the number of attributes in the data being processed. A linear activation function is normally used across the input layer, so that all input values reach the hidden layer without distortion. Feedforward NNs are usually fully connected, meaning that each neuron in a layer is connected to every neuron in the consecutive layer. The connections between neurons are weighted, and the weight of a connection influences the strength of the transmitted signal.

The *hidden layer* recieves signals from the input layer. A non-linear activation function such as the sigmoid function is usually used across all the units in the hidden layer. A number of consecutive hidden layers can be employed in a FFNN, however, it has been mathematically proved that a FFNN with a single hidden layer is capable of representing any non-linear relationship between inputs and targets, provided there are enough neurons in the hidden layer [52, 62, 76]. The purpose of hidden units is to introduce non-linearity into the approximated relationship between inputs and targets, and the number of hidden units determines the level of non-linearity that the NN is capable of representing. Hence, if too few hidden units are used, it will be impossible to map the data precisely, because the information capacity of the NN will not correspond to the amount of information represented by the training data. On the other hand, if too many hidden units are used, the information capacity of the NN will exceed the necessary minimum, and the redundant hidden units will learn unnecessary information such as the order of patterns or data noise, reducing the ability of the NN to correctly predict the outputs of previously unseen patterns [4]. The ability of the NN to correctly predict the outputs of previously unseen patterns is also referred to as the ability to *generalise*. The deterioration of the generalisation ability of the NN due to learning unnecessary information is known as *overfitting*. Other causes of overfitting are training the NN for too long, and using a training set that is either not representative of the mapping to be learned, or contains noise [32]. For a more extensive discussion of overfitting, refer to [4, 10].

Thus, it is important to have a correct number of hidden units in the NN. A number of techniques to optimise the number of hidden units have been developed to date. These techniques are generally referred to as *architecture selection*. More details on architecture selection are given in Section 3.4.1.

In addition to the inputs from the preceding layer, hidden and output units also receive an input signal from a bias unit. Bias units represent the threshold values of units in the next layer. A bias is a neuron with no inputs, which outputs a constant value (usually -1, although it can be any other non-zero value) used by the hidden and output units as an additional input. During NN training, the weights connecting bias units with other units are adjusted together with all the other weights. The NN weights define the position of the hypersurface that the NN represents, and without the bias units this hypersurface would be constrained to pass through the origin, which would be a significant limitation to the approximation ability of the NN [81].

The number of dependent variables, or outputs, determines the number of neurons in the *output layer* of a NN. The output range of the output neurons depends on the activation function used. For example, if the sigmoid function is used, the output of each neuron will be in $(0, 1)$. This implies that the target values must be scaled to the activation function range before training, and scaled back afterwards. For more information on data pre-processing, refer to Section 3.4.2.

The NN output is calculated by a single forward pass of an input pattern p through the FFNN. The output of each neuron in the output layer, $o_k, k = 1, \ldots, K$, with I input neurons, J hidden neurons and K output neurons, is given by

$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} u_j\right) = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{u_j}\left(\sum_{i=1}^{I+1} v_{ji} z_i\right)\right) \tag{3.2}$$

for all $i = 1, \ldots, I, j = 1, \ldots, J$, v_{ji} is a weight connecting the j^{th} hidden neuron and the i^{th} input neuron, w_{kj} is a weight connecting the k^{th} output neuron and the j^{th} hidden neuron, the $(I+1)^{\text{th}}$ input neuron and the $(J+1)^{\text{th}}$ hidden neuron are the bias neurons, u_j and z_i are outputs of the hidden layer and the input layer, respectively, f_{o_k} and f_{u_j} are the activation functions in the output and the hidden layer, respectively.

Other variations of NNs include functional link NNs [40, 56], product unit NNs [29, 61], recurrent NNs [92, 81, 132], and time-delay NNs [72, 120], amongst others. The discussion of these and other NN variations falls outside the scope of this thesis. For more information, refer to [4, 32, 101, 112, 142].

Standard FFNNs were used in the experiments conducted for this study.

3.3 Training Algorithms

The NN itself is simply a structure capable of representing a mapping between the input space and the output space, requiring execution of a training algorithm in order to learn that mapping. Training can be *supervised*, when the goal of the training algorithm is to minimise the difference between the target outputs and the actual outputs. Training can also be *unsupervised*, when no target outputs are defined, and the goal of the training algorithm is to structure the unlabeled data patterns (*e.g.* cluster patterns and maximise the difference between the obtained clusters). Another NN training paradigm is *reinforced* training, when the NN acts as an agent that takes actions (updates NN weights) in an environment. The training is accomplished through the NN being either rewarded or punished by the environment based on the actions the NN agent takes [47]. This study deals with supervised NNs only.

This section provides a discussion of the supervised training algorithms used in this study. Section 3.3.1 outlines the stages of the learning process, Section 3.3.2 describes the back propagation training algorithm, and Section 3.3.3 discusses how the PSO algorithms can be applied as NN training algorithms.

3.3.1 The Learning Process

NN training involves finding a set of weights that will accurately approximate the mapping $f_{NN} : \mathbb{R}^I \rightarrow \mathbb{R}^K$, where I is the number of inputs and K is the number of outputs. The data patterns of the data set D are randomly divided into a training set D_T and a generalisation set D_G, such that $D_T \cap D_G = \emptyset$. D_T is used to train the NN, and D_G is used to evaluate the generalisation ability of the NN. During training, the weight vector W, which consists of all the NN weights, is iteratively adjusted in order to minimise the empirical error produced by the NN, given by

$$E_T(D_T; W) = \frac{1}{P_T} \sum_{p=1}^{P_T} (f_{NN}(\vec{z}_p, W) - \vec{t}_p)^2 \tag{3.3}$$

where P_T is the total number of training patterns, f_{NN} is the function that the NN currently represents, $\vec{z}_p$ and $\vec{t}_p$ are input and target vectors, respectively. Decreasing empirical error yields decreasing generalisation error, unless overfitting occurs. In the context of NN training, the empirical error in equation (3.3) is referred to as the *objective function* to be optimised by the training algorithm.

Two types of supervised learning algorithms can be distinguished based on when weights are updated: *stochastic*, or *on-line* learning, when the weight vector is adjusted every time a data pattern is presented, and *batch*, or *off-line* learning, when the changes are accumulated and applied to the weight vector only after the complete training set has been presented to the NN [32, 104].

Many different supervised training algorithms exist, including gradient descent [127], scaled conjugate gradient [18, 84], leapfrog [3, 50], simulated annealing [95, 107], evolutionary algorithms [11, 35, 57], and particle swarm optimisation [67, 83], amongst others. Training algorithms used in the experiments conducted for this study are described below.

3.3.2 Back Propagation with Gradient Descent

The most commonly used and popular algorithm to train a FFNN is back propagation, developed by Werbos [127], which uses gradient descent to adjust weight values such that E_T is minimised. Each iteration of this algorithm consists of two phases:

1. **A feedforward pass**, which propagates ("feeds") a pattern through the NN and calculates the output.

2. **Back propagation**, which compares the output obtained in the feedforward phase to the desired output, calculates the error, and propagates the error back from the output layer through the hidden layers to the input layer. The weights are adjusted during back propagation as a function of the error value.

Pseudocode of stochastic back propagation with gradient descent for a NN with 3 layers is given in Algorithm 3.1. The algorithm starts by randomising the weights such that the mean of the weights is approximately zero (for more information on weight initialisation, refer to Section 3.4.3) and setting the iteration counter to 0. A single iteration of back propagation is referred to as an *epoch*. The feedforward pass consists of calculating the NN output $o_{k,p}$ for each pattern p and each output unit using equation (3.2). In case of stochastic learning, a pattern p is randomly selected from the training set at each iteration. Choosing patterns randomly prevents the NN from learning the order in which the patterns are presented [32]. Weight adjustments, Δw_{kj} and Δv_{ji}, depend on the activation functions used in the output and the hidden layer and the objective function, since the weight adjustment formulae are derived from the gradients of these functions. Assume that the sigmoid function, given by equation (3.1), is used as the activation function in the hidden and output layers, and the sum squared error (SSE) is the objective function. Then, for each pattern, p, the error function E_p is given by

$$E_p = \frac{1}{2}\frac{\sum_{k=1}^{K}(t_{k,p} - o_{k,p})^2}{K} \tag{3.4}$$

where K is the number of output units, $t_{k,p}$ are the desired outputs of p, and $o_{k,p}$ are the actual outputs produced by the NN for pattern p. The overall error produced by the NN is minimised by calculating the gradient of E_p in the weight space, and moving the weight vector along the negative gradient. The formula to adjust the weights w_{kj} between hidden unit j and output unit k is then given by

$$\Delta w_{kj} = \eta(-\frac{\partial E_p}{\partial w_{kj}}) = \eta(t_{k,p} - o_{k,p})(1 - o_{k,p})o_{k,p}u_{j,p} \tag{3.5}$$

where $u_{j,p}$ is the output of hidden unit j, and η is the *learning rate*, discussed in more detail in Section 3.3.2. The error signal on the output layer is then propagated back to change the weights v_{ji} between each hidden unit j and each input unit i:

$$\Delta v_{ji} = \eta(-\frac{\partial E_p}{\partial v_{ji}}) = z_{i,p}(1 - u_{j,p})\sum_{k=1}^{K} w_{kj}\Delta w_{kj} \tag{3.6}$$

where $z_{i,p}$ is the i^{th} input value. The momentum term α is a constant value used to control the influence of past weight values on the current weight values. Momentum is discussed in more detail in Section 3.3.2.

```
1. Set the current training iteration $t = 0$
2. Initialise $v_{ji}, w_{kj} \sim U(-m, m), m \in \mathbb{R}$
**repeat:**
    **for** each pattern **do:**
        Calculate NN outputs using equation (3.2).
        Calculate error $E_p$ using equation (3.4).
        Calculate $\Delta w_{kj}$ using equation (3.5).
        Adjust the weights $w_{kj}$ using:
        $w_{kj} = w_{kj} + \Delta w_{kj}(t) + \alpha \Delta w_{kj}(t-1)$
        Calculate $\Delta v_{ji}$ using equation (3.6).
        Adjust the weights $v_{ji}$ using:
        $v_{ji} = v_{ji} + \Delta v_{ji}(t) + \alpha \Delta w_{ji}(t-1)$
    **end for**
    $t = t + 1$
**until** stopping criteria are met
```

Algorithm 3.1: Gradient Descent Back Propagation

The stopping criteria is usually defined by a maximum number of epochs, by classification error, or by setting a threshold on the mean squared error (MSE) produced by the generalisation set, given by

$$E_T = \frac{\sum_{p=1}^{P} \sum_{k=1}^{K} (t_{k,p} - o_{k,p})^2}{P_G K} \tag{3.7}$$

where P_G is the total number of patterns in the generalisation set.

The performance of stochastic back propagation is strongly dependent on the values chosen for α and η. The impact of these two parameters is discused below.

Learning Rate

The learning rate, denoted by η, controls the step sizes of the weight adjustments. A small η makes weight adjustments correspondingly small, which means that the algorithm takes small steps towards the optimum. It slows down convergence, but ensures that the optimisation process follows the gradient path smoothly. An obvious pitfall of this approach is susceptibility to being trapped in a local optimum.

Large η promotes exploration by making large weight updates, resulting in large jumps across the search space. Although a large η can help to avoid local minima, it may also prevent the algorithm from finding a good optimum by jumping over potentially good solutions.

Thus, it is important to consider the exploration-exploitation trade-off in the context of a problem to be solved, and choose η accordingly. Apart from the option of finding an optimal η value empirically, a number of adaptive strategies have been suggested that adjust the value of η during training [60, 80, 139]. However, most of these algorithmic adaptations have been developed with static environments in mind, with the main objective of speeding up convergence. This study, however, deals with NN training in dynamic environments, where fast convergence is not necessarily desirable, and increased exploration capacity is needed (for a discussion of dynamic environments, refer to Chapter 4). Transferring adaptive learning rate techniques from static environments to dynamic environments is out of the scope of this study, and optimal static η values were chosen empirically.

Momentum

Momentum, denoted by α, controls the influence of previous weight values on the current weights. Momentum was introduced as a countermeasure to fluctuating changes caused by the stochastic approach to NN training. When the NN weights are updated after each pattern presentation, the error gradient may change from one pattern to the other, causing the cumulative weight update to be small due to consequent weight updates cancelling out one another [32]. By taking older weights into account, the momentum term averages out the weight changes and ensures that the search path leads in an average downhill direction. The optimal α is problem dependent, and a static value is often used, derived empirically via trial and error. However, adaptive versions have also been suggested [89, 138]. Adaptive momentum strategies were originally developed for static environments, and it cannot be assumed that the suggested strategies would work in dynamic environments without further changes. Since adaptive momentum falls outside the scope of this thesis, a static value was empirically chosen.

Back propagation is based on error gradients, meaning that it is essentially a hill-climbing algorithm. As with any hill-climbing approach, its major disadvantage is susceptibility to premature convergence on local minima. Another disadvantage of hill-climbing approaches is the dependence on the starting point of the search, which would be the initial weights in case on NNs. Alternative training algorithms addressing these issues were suggested, such as evolutionary algorithms [11, 35, 57] and particle swarm optimisation [67, 83].

3.3.3 Population Based Algorithms: Particle Swarm Optimisation

Evolutionary algorithms (EA) were the first population-based algorithms that were applied to NN training by evolving a population of NNs until one NN is found that minimises the MSE to a small enough value [35]. Fogel *et al* [35] were the first to suggest evolving NNs, initiating a number of studies on the applicability of EAs to NN training [11, 95, 136, 137].

PSO is another population-based algorithm (refer to Chapter 2) that has been successfully applied to train NNs [44, 67, 83, 118]. In order to train a NN using PSO, the following needs to be done:

- A fitness function has to be defined, which is usually simply the MSE, calculated using equation (3.7).

- An appropriate representation of candidate solutions has to be determined. Each particle is used to represent a candidate solution, which is a vector of all of the weights and biases of a NN. Every element of a particle represents a single weight or bias, using floating-point numbers. Therefore, each particle has a dimension equal to the total number of weights in the NN [83].

The PSO is then used, as discussed in Chapter 2, to adjust the weight and bias values (using the particle velocity and position updates) such that the given fitness function is minimised.

Recent research has shown PSO to be a very effective NN training algorithm [34, 44, 83, 118]. PSO outperformed standard back propagation on a selection of classification, function approximation, and prediction problems. Additionally, PSO has also been applied to train product unit NNs [34, 59], recurrent NNs [106], RBFNNs [97], and SOMs [96, 133].

The advantages that PSO offers in comparison with back propagation are:

- weaker dependence on the initial weight values, since multiple starting points (*i.e.*, particles) are used in the search process,

- derivative information of the activation functions and the error function is not used, thus the activation functions and the error function do not have to be differentiable,

- computationally more efficient, and

- more robust on rugged surfaces, since population-based search is less prone to premature convergence on local minima than back propagation [83].

The major disadvantages of PSO, as compared to back propagation, are slower speed of convergence and more algorithm parameters ($\omega, c_1, c_2, \vec{V}_{max}$, if applicable, and swarm size) to optimise before optimal performance can be expected. For a discussion of PSO algorithm parameters, refer to Section 2.

Apart from performance issues associated with each specific training algorithm, there are other factors that influence the performance of a NN. These factors are discussed in the next section.

3.4 Performance Issues

Training a NN, regardless of the training method chosen, is not an easy task, since the performance of a NN is dependent on factors such as the NN architecture and data format. This section addresses aspects that have an influence on the performance of a NN. Section 3.4.1 discusses architecture selection, Section 3.4.2 discusses the data pre-processing that is required, and Section 3.4.3 discusses weight initialisation.

3.4.1 Architecture Selection

As discussed in Section 3.2, using the correct number of hidden units is crucial to obtain good NN performance, since too few hidden units results in poor information capacity and underfitting, and too many hidden units results in overfitting and poor generalisation ability [32]. A simple approach to architecture selection would be to create a number of NNs with different architectures, compare their training and generalisation performance on a given problem, and select the architecture that produces the lowest generalisation error. More complex architecture selection techniques include network construction (growing) [37, 49, 71], pruning [32, 110], and regularization [41, 131]. For a more detailed overview of architecture selection techniques, refer to [32].

This study deals with NN training in dynamic environments, which further complicates the problem of finding an optimal architecture. For a discussion of architecture selection in the context of dynamic environments, refer to Chapter 5.

3.4.2 Data Preparation

In order for NN training to be effective, the data must be converted to the format acceptable for NN training. The output of a NN for a data pattern p is calculated mathematically. This restricts the input data to floating-point values. Nominal attributes have to be transformed to floating-point values. This is either done by binary encoding, or by mapping various nominal attributes to different numerical values [32].

The numerical data patterns should also be scaled to the active range and domain of the activation functions employed. NN training requires comparison of target values to calculated outputs in order to minimise the difference between them. The calculated outputs of the NN are produced by the activation functions, and thus always fall within the activation function range. The target values must be scaled to the range of the activation function in order to be comparable to the calculated outputs. Targets that do not fall within the activation function range will be unreachable to the NN, and thus the produced NN error will remain high and unrepresentative. The active range of the sigmoid function is $(0, 1)$. However, this function is asymptotic, and therefore the actual outputs are always bigger than 0 and smaller than 1. For this reason, the targets are usually scaled to $[0.1, 0.9]$ in case of continuous-valued targets, and the set $\{0.1, 0.9\}$ for binary-valued targets (used for classification).

Scaling inputs is optional. However, performance can be improved by scaling the inputs to the active domain of the activation function, which can be defined as the interval on which $f'(x)$ changes significantly for different values of x. For example, the active domain of the sigmoid function is given as $[-\sqrt{3}, \sqrt{3}]$ [46]. Scaling to the active domain is done in order to avoid the asymptotic ends of the activation function, since the closer inputs come to the asymptotic ends, the less significant will be the difference between the outputs they produce, thus making it difficult for the NN to differentiate between different patterns. In case of back propagation, unscaled input values also have a negative effect on the training algorithm performance, since the derivatives of the activation function are very small near the asymptotic ends, and small derivative values yield small weight updates, resulting in slow convergence. The PSO does not use the activation function derivatives, and therefore is less sensitive to the scaling of inputs. However, scaled inputs in case of PSO training still allow the calculated outputs to better reflect the difference between patterns.

3.4.3 Weight Initialisation

The initial weights can have a strong effect on NN performance. In case of gradient-based training algorithms, such as back propagation, a single set of initial weights needs to be generated. This set of weights, further referred to as the weight vector, becomes the starting point of the hill-climbing search for an optimum, and if the starting point happens to be far from the optimum, slow convergence is usually observed [55]. The weight vector is usually generated randomly in a small region around zero, which helps keeping neuron inputs in the active domain of the activation function. If the activation function goes through the origin, then small random weights centered around zero prevent bias toward any solution regardless of the input values [32]. Wessels and Barnard [128] derived a rule for weight initialisation interval calculation from the number of connections leading to a neuron:

$$\left[\frac{-1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}\right]$$

where $fanin$ is the number of connections leading to a unit. Due to the strong dependence of back propagation performance on the initial weights, the interval from which the weights were sampled was chosen empirically for each problem in the experiments conducted for this study.

In case of population-based methods such as PSO, the number of generated weight vectors is equal to the number of particles. This implies that the search for an optimum has multiple starting points, and thus the success of the search is less influenced by the initial weight values. This is a natural advantage of population-based methods over back propagation. The initial population needs to be sampled from a uniform distribution to induce no bias and cover as much of the search space as possible.

3.5 Summary

This chapter provided an overview of feedforward neural networks and training algorithms used in this study, namely, back propagation and PSO. Different factors influencing NN performance, such as data format and weight initialisation, have also been discussed. The next chapter provides an overview of dynamic environments, or environments with temporal characteristics.

Chapter 4

Dynamic Environments

Nothing will die;

All things will change

Thro' eternity...

Alfred Tennyson (1830)

Most optimisation algorithms from the CI field assume that the search landscape is static. However, this assumption is not valid for many real-world problems. Dynamic environments can often be observed in real life, for example, the stock exchange, or traffic conditions on roads. Given a problem in such an environment (*e.g.* constructing an optimal traffic lights schedule), it is clear that a solution once found may become suboptimal because certain properties of the environment will change over time (*e.g.* increased road congestion during the rush hour will yield a traffic light schedule optimised for midday road congestion suboptimal). Temporal properties introduce extra complexity into any problem, since a solution once found will have to be adapted every time a temporal property changes.

To adapt existing CI methods to dynamic environments, the nature of dynamic environments has to be studied. Different characteristics of dynamic environments and optimisation problems with temporal properties, or dynamic optimisation problems, are discussed in this chapter, as well as some of the existing CI algorithms applied to dynamic optimisation problems. Section 4.1 presents a few real-life examples of dynamic optimisation problems, Section 4.2 discusses characteristics of dynamic environments and introduces the notion of concept drift, Section 4.3 discusses CI algorithms developed for dynamic optimisation problems that have been applied in this study, and Section 4.4 concludes the chapter.

4.1 Real-Life Examples of Dynamic Optimisation Problems

Real-life examples of dynamic problems are provided in this section in order to illustrate the fact that dynamic problems to which NNs are applicable are common place, and that research in the development of NN training algorithms for dynamic environments is in fact necessary.

Spam filtering

Anyone who ever dealt with electronic mail knows what spam is and how annoying it can be. Getting rid of spam is not only time-consuming, but quite frustrating as well; e-mails with similar subjects and content arrive repetitively. In case of large companies, the issue is even more serious: spam causes financial losses due to wasted bandwidth [25].

The modern approach to avoid spam is to use spam filtering algorithms. The basic idea behind spam filtering is the ability of a filter to classify an e-mail as being spam or not. In order to do this, specific features that identify spam have to be found. Filters normally look for certain keywords and text patterns typical for spam, and based on the information obtained after analysing message content, decide whether the e-mail is legitimate or not [25].

Spam filtering is a classification problem, and CI techniques can be used to solve it. For example, a supervised neural network (NN) can be trained to differentiate between spam and legitimate e-mails. However, it is not clear for how long a trained NN will classify e-mails correctly. In our ever-changing world, nothing is static. Neither is spam, especially taking into account the fact that it is written by spammers – human beings who realise that they have to by-pass complex filters. The content of spam messages changes over time, partly because spammers try to fool filters by introducing hidden context that would not alter the text as viewed by the user, but will cause filters to incorrectly classify spam as legitimate mail. Also, every now and again new forms of spam emerge, and if these forms do not follow the existing spam patterns, filters such as NNs fail, since the features (*i.e.* NN inputs) used to characterise spam become unrepresentative and have to be updated.

Thus, a once trained spam filtering NN will gradually lose precision. NNs have already been successfully applied to spam filtering under the assumption that the problem remains static, or that the existing NN training algorithms can be applied to dynamic problems without further modifications [58, 103, 135]. However, it has been observed that NNs trained using back propagation struggle to learn from new e-mail examples if these e-mails represent temporally different characteristics [135]. Therefore, new training algorithms have to be developed that will allow NNs to adapt to temporal characteristics of the learning problem.

Antibiotic Resistance

Antibiotics, or the antimicrobial drugs used to fight against bacteria-caused infections, were discovered in the 20th century and became widespread in the 1940's [115]. The discovery of antibiotics caused a revolution in the medical science, significantly decreasing the death rate from various diseases, and many illnesses that used to be incurable can now be effectively combated with the use of antibiotics.

Bacteria, however, is gradually adapting to antibiotics via evolution. Some antibiotics have lost most of their medical value due to the fact that harmful bacteria has evolved, and is no longer affected by the drugs. The evolution of infectious micro-organisms is faster than the process of creating new antibiotics [115]. The phenomenon of micro-organisms' adaptation to antibiotics is known as *antibiotic resistance*. Mutation in antibiotic-interacting micro-organisms' proteins is one of the primary causes for antibiotic resistance. Prediction of resistance mutations in these proteins is valuable for the molecular dissection of antibiotic resistance mechanisms, as well as for predicting features used for the development of new drugs to counter resistant strains [14]. NNs have been successfully applied to the prediction of resistance mutations before [14, 73, 115, 122]. While the results obtained with the existing NN training algorithms were very promising, the authors agreed that further performance improvement is desirable [14, 73]. Alternative training algorithms designed specifically for dynamic environments have a high potential of improving NN's performance on the antibiotic resistance problem.

4.2 Dynamic Optimisation Problems

This section discusses various types of dynamic environments and dynamic optimisation problems. Section 4.2.1 discusses the notion of dynamic optimisation problems in the context of NN training. Section 4.2.2 discusses the terminology used in this study. Section 4.2.3 discusses attributes used to describe dynamic environments and dynamic optimisation problems. Section 4.2.4 focuses on terminology specific to dynamic classification problems, and introduces the notion of concept drift.

4.2.1 Primary Concepts

Optimisation problems constitute a broad application area of CI techniques. In computer science, the term *"optimisation problem"* refers to the problem of finding an acceptable solution from the set of all feasible solutions (feasible space of the search space). Mathematically, an optimisation problem is the problem of either minimising or maximising an objective function within the search space, where the search space is the set of all feasible solutions (feasible space). Classification problems can be seen as a special case of function optimisation problems, where the objective is to minimise classification error. Training a NN is also an optimisation problem, where the error function of the NN is the objective function, and the goal of a training algorithm is to find a set of NN weights such that the error produced by the NN is minimised.

Dynamic optimisation problems are optimisation problems where one or more of the following may change over time:

1. the objective function,
2. function parameters,
3. constraints and boundary constraints.

This study deals with unconstrained dynamic optimisation problems only. Considering the objective function landscape as a hypersurface, it can be argued that environment changes are basically perturbations of this hypersurface. The changes in the environment result in *floating optima* of the objective function: An optimum may change its position and/or magnitude, existing optima may disappear while new optima may appear somewhere else on the hypersurface. For a static optimisation problem, the objective of an optimisation algorithm is to find an optimum of the objective function. For a dynamic optimisation problem, the objective changes from simply locating the optima to tracking the optima as the optima changes, as well as locating new emerging optima and detecting the disappearing optima. Assuming minimisation, a dynamic optimisation problem is formally defined as [33]:

$$\begin{aligned}&\text{minimise } f(\vec{x}, \vec{\varpi}(t))\\&\text{subject to } x_l \in dom(x_l)\end{aligned}$$

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

4.2.2 Terminology

Before the discussion of optimisation in dynamic environments can continue, terminology used in this study should be clarified. As formally defined in Section 4.2.1, a dynamic optimisation problem is an optimisation problem with time-dependent control parameters. For the remainder of this text, dynamic optimisation problems are referred to simply as *dynamic problems*.

The term *"dynamic environments"* is often used interchangeably with "dynamic problems" [12, 31, 54, 79, 113], since the optimisation problem constitutes the environment in which the optimisation algorithm operates. However, such interchangeable terminology can be a source of confusion, since the optimisation problem itself exists in a certain environment, or context. Changes in the context may cause changes in the objective function of the optimisation problem. For example, in the case of supervised NN training, a mapping between inputs and targets is found such that the error function is minimised. Any changes in this mapping, *e.g.* changes in the underlying distribution of inputs, changes in the number of inputs and targets, or changes in decision boundaries, cause changes in the search landscape as defined by the error function. Hence, the mapping between inputs and targets is the dynamic environment of the error function. A change in the environment may yield an increase in NN error, since the current weight vector may no longer accurately represent the mapping between inputs and targets. In this study, the term *dynamic environment* is used to refer to the changeable context of a dynamic problem, such that any changes in the context yield changes in the landscape of the objective function. Any optimisation problem that has such a dynamic environment becomes a dynamic problem.

The next section describes characteristics used in this study to discern between different types of dynamic environments and dynamic problems. Due to the strong bond between dynamic problems and corresponding dynamic environments, the characteristics listed in the following section apply to both dynamic environments and dynamic problems.

4.2.3 Characteristics of Dynamic Environments

Depending on the dynamic environment, the objective function landscape of the dynamic problem can change continuously, at regular time intervals, or unpredictably. This property of dynamic problems is referred to as *temporal severity* or *frequency of change* in the literature [33].

The *severity of change* may vary from small changes, when optima move from their current positions in small steps with every environment change (*gradual change*), to a complete change of the objective function landscape (*abrupt change*). The severity of change in optima locations and magnitude is referred to as *spatial severity* [33].

Different combinations of temporal severity and spatial severity result in different types of dynamic environments, ranging from dynamic environments exhibiting infrequent gradual changes, to dynamic environments exhibiting frequent abrupt changes. Frequent abrupt changes are the hardest to track and adapt to, since optimisation algorithms are required to make significant adjustments in a short period of time.

Temporal severity and spatial severity are general characteristics that apply to any dynamic optimisation problem, including NN training in a dynamic environment. Since this study focuses on dynamic classification problems, the next section describes how the general characteristics of dynamic problems apply to dynamic classification problems.

4.2.4 Concept Drift

This thesis focuses on classification problems, which make up a large subset of the optimisation problems that NNs can be applied to. The goal of a NN used to solve a classification problem is to learn the classification *concepts* from the given data. Assuming supervised learning, the term *concept* refers to a pattern that can be discovered in the training data. The mapping between inputs and targets, i.e. the predictive model that is represented by the NN, can be accurately described by a collection of such concepts. In case of classification, concept learning implies learning to distinguish between different classes by approximating decision boundaries between the classes.

Real-world problems are often dynamic, and classification problems are no exception. The underlying data distribution may change over time, causing changes in the decision boundaries. Changes in the decision boundaries will yield changes in the target concepts. This phenomena is referred to as *concept drift* [105]. The term "concept drift" belongs to the field of data mining, and is usually used to refer to drifting concepts as observed in large data sets and continuous data sets over time. In case of classification, drifting concepts imply changes in the decision boundaries that separate classes. The boundaries between classes may shift, new boundaries may appear, and old boundaries may become obsolete.

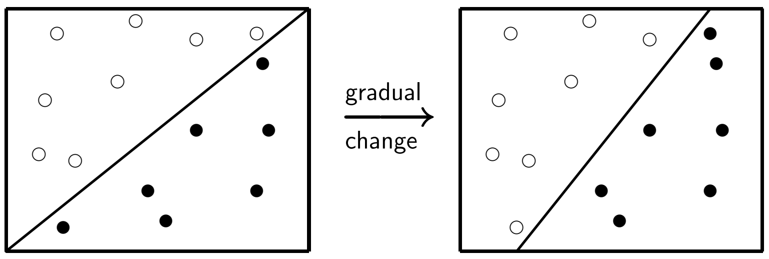
Concept drift complicates the process of concept learning by the NN, because the learned concepts become obsolete as the actual concepts drift, requiring the learned model to be revised. If decision boundaries change over time, the NN will have to detect and track such changes in order to update the learnt model accordingly.

Types of Concept Drift

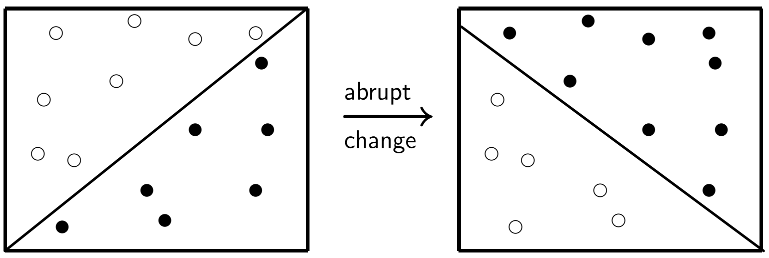
Literature distinguishes two types of concept drift: *actual*, or *real concept drift*, and *virtual concept drift* [129]. Actual concept drift is observed when the target concepts change due to changes in the underlying *hidden context*, i.e., attributes of the environment not represented explicitly by the attributes of the optimisation problem [105]. Changes in the hidden context, however, may also cause changes in the underlying data distribution, which may require the learnt model to be revised, since the model's error may no longer be acceptable with the new data distribution [116]. This kind of concept drift is referred to as virtual concept drift [129]. Tsymbal [114] argues that, from the practical point of view, it is not important what kind of concept drift occurs, since, irrespective of the type of concept drift, the learnt model will have to be revised.

Classification problems with concept drift, also referred to further in this text as dynamic classification problems, make up a subset of dynamic optimisation problems, where the objective is to minimise classification error, and track the decision boundaries as they shift, as well as detect disappearing decision boundaries and locate new decision boundaries. Therefore, characteristics of dynamic problems described in the previous section, such as spatial and temporal severity, are applicable to problems with concept drift. Assuming classification problems, the frequency of concept changes impacts on the frequency with which the decision boundaries change, and the severity of concept changes impacts on the extent by which the decision boundaries shift, appear or disappear. The most severe change would be a complete disappearance of a decision boundary or appearance of a new decision boundary. Figure 4.1 illustrates spatial severity of changes as applied to concept drift.

It is interesting to note that the body of research on concept drift has never been merged with the body of research on continuous dynamic optimisation problems, and the methods for continuous dynamic optimisation were never applied to concept drift problems. In this thesis, continuous dynamic optimisation algorithms are applied to NN



(a) Gradual decision boundary change



(b) Abrupt decision boundary change

Figure 4.1: Spatial severity applied to concept drift

training in the presence of concept drift.

4.3 Existing Dynamic Optimisation Methods

This section discusses some of the existing CI methods developed for dynamic optimisation problems and classification problems with concept drift, with the focus on methods used in this study. Section 4.3.1 introduces the dynamic optimisation research field, Section 4.3.2 discusses several ways in which particle swarm optimisation (PSO) may be adapted to dynamic environments, and Section 4.3.3 outlines existing approaches to solve classification problems in the presence of concept drift.

4.3.1 Optimisation in Dynamic Environments

The field of CI research in dynamic environments is still very young, and many possible approaches to dynamic problems have not been exhaustively experimented with yet.

However, the literature survey conducted for the purpose of this study showed that various population-based algorithms, such as PSO or genetic algorithms, were applied to dynamic environments with a high degree of success [8, 9, 7, 13, 22, 42, 79]. Both evolutionary algorithms and swarm algorithms are adaptable by nature, and thus require relatively minor changes to their standard algorithm structures to work in dynamic environments. Evolutionary algorithms, however, are out of scope of this study, and therefore only dynamic approaches to PSO used in this work are discussed. For a more comprehensive review of evolutionary optimisation in dynamic environments, refer to [12, 86, 134]. For a more comprehensive review of PSO in dynamic environments, refer to [6, 9].

4.3.2 Dynamic Particle Swarm Optimisation

Generally speaking, all dynamic optimisation algorithms have to go through two phases:

1. **Change detection:** Some kind of an environment change sensor has to be implemented to make the algorithm aware of the changes that occur.

2. **Response to the change:** The existing solution has to be adjusted, if necessary, whenever the context changes.

Both phases are discussed below in the context of particle swarm optimisation.

Change Detection

In order to respond to a change in the environment, the change has to be detected by the PSO. Change detection is usually accomplished by making use of one or more *sentries*. A sentry is either a dedicated particle or a fixed point in the search space [15, 17]. The only difference between a normal particle and a sentry particle is that sentry particles keep a record of their previous fitness values: at the start of each iteration, sentry particles are re-evaluated, and if the difference between the previous fitness and the new fitness exceeds a certain threshold, it can be assumed that a change has occurred. It was suggested by Hu and Eberhart [53] to use the global best particle as a sentry. Using the global best particle ensures that changes in the current optimum are always detected, which is beneficial in gradually changing environments, where the optimum can be tracked as it changes. However, this approach prevents the swarm from detecting environmental changes that occur away from the current optimum, which implies that new emerging optima have a high chance of being overlooked by the algorithm. Carlisle *et al* [16] suggested that one or more randomly chosen particles be used. Random sentries cover a wider area than global best sentries; however, only until the swarm converges. Static point sentries, on the other hand, are independent of the swarm diversity. Hence, static points uniformly distributed throughout the search space can be more effective sentries than the swarm particles [15]. The number of sentries to use in order to efficiently detect changes is problem dependent. Using more sentries increases the probability of detecting the change, but also increases overall computational complexity.

Response to the Change

Standard PSO faces the following problems when optimisation in dynamic environments is required:

1. **Outdated memory:** Once the environment changes, previous values stored in PSO memory (personal best and global best positions) are no longer representative of the new environment [33], and thus provide the swarm with misleading information instead of leading the search towards an optimum.

2. **Loss of swarm diversity:** It was formally proved [21, 117] that with a standard PSO, the swarm will gradually loose diversity from iteration to iteration, until all particles converge on a weighted average of the personal best and global best positions. Once converged, PSO will not explore any longer, because particle

velocities, according to equation (2.2), will tend to zero as the distance between the current and the global best position, as well as the distance between the current and the personal best position decrease. A converged PSO has no exploration capabilities and will not be able to adapt to an environment change [33].

A number of PSO variations have been developed, differing in the way that the above issues are addressed. A review of popular dynamic PSO algorithms used in this study is given below.

Reinitialising PSO

The reinitialising PSO approaches the aforementioned problems in a simple, naive manner. The outdated memory issue is addressed by re-evaluating particle positions, as well as the stored global and personal best positions. Diversity of the swarm is boosted by means of reinitialising the positions, velocities and personal best positions of a percentage of particles. The particles to be reinitialised are randomly selected. The disadvantage of this approach is partial loss of knowledge about the search space due to particle reinitialisation [54].

The ratio of particles to be reinitialised is problem dependent and should be chosen empirically. For example, extensive abrupt changes may require most of the swarm to be reinitialised, and minor gradual changes may be addressed by reinitialising only a small percentage of the swarm.

Charged PSO

The charged PSO [7] is based on electrostatic principles. All particles in a charged PSO store a charge, represented by a positive scalar value. A charge magnitude equal to zero means that a particle is neutral (i.e. does not bear a charge), and a value greater than zero indicates a charged particle. Charge magnitude can not be negative, and does not change during algorithm execution.

Charged particles repel from one another if the distance between them is small enough. This prevents charged particles from converging to a single point, thus facilitating exploration and addressing the diversity loss problem. Repulsive forces are introduced by adding an acceleration term, $\vec{a}_y$, to the standard velocity equation:

$$\vec{v}_y(t) = \omega\vec{v}_y(t-1) + c_1\vec{r}_1(\vec{x}_{pbest} - \vec{x}_y(t)) + c_2\vec{r}_2(\vec{x}_{gbest} - \vec{x}_y(t)) + \vec{a}_y(t)$$

Repulsion between two particles y and g at time step t is defined as [7]

$$\vec{a}_{yg}(t) = \begin{cases} \left(\frac{Q_y Q_g}{||\vec{d}_{yg}(t)||^3}\right)(\vec{d}_{yg}(t)) & \text{if } R_c \leq ||\vec{d}_{yg}(t)|| \leq R_p \\ 0 & \text{otherwise} \end{cases}$$

where $\vec{d}_{yg}(t) = \vec{x}_y(t) - \vec{x}_g(t)$, Q_y is the charge magnitude of particle y, R_c is the core radius and R_p is the perception limit of a particle. These two limits define the distance range $[R_c, R_p]$ at which charged particles will repel one another. Neutral particles are assigned $Q_y = 0$, and thus do not contribute to the calculation of acceleration. Charged particles are assigned $Q_y > 0$. The value of Q_y assigned to charged particles controls the extent of acceleration and should be chosen empirically, since optimal acceleration is problem dependent [7]. R_c and R_p are the radii of repulsion, and can not be assigned negative values. The lower radius R_c of the interval $[R_c, R_p]$ is a safeguard against singularity of the inverse square law [7], and the upper radius R_p is a tunable parameter that controls the domain of influence of the repulsion. If $R_c \approx R_p$, almost no repulsion will be observed. Therefore, R_p is usually chosen such that $R_c \ll R_p$. As can be concluded from equation (4.1), acceleration is inversely proportional to the distance between the charged particles, and the further two charged particles are from each other, the weaker they will repel. Thus, repelling forces maintain swarm diversity without yielding divergent behaviour.

For the problems considered in [7], the most efficient charged PSO architecture had 50% of the swarm charged, and the rest of the particles were neutral. Neutral particles are normal PSO particles that obey standard PSO position and velocity update rules.

Thus, one half of the population acted as a standard PSO swarm and refined found solutions, thereby facilitating exploitation. Hence, the charged PSO achieved a balance between exploration and exploitation.

The problem of outdated swarm memory is addressed by re-evaluating the fitness of each particle in the swarm, the personal best of each particle, and the global best of each neighbourhood, whenever a change occurs.

Quantum PSO

The quantum PSO [9] is vaguely based on the model of an atom. The orbiting electrons of an atom are replaced by a quantum cloud, where the position of each electron is determined not by its previous position and trajectory dynamics, but by a probability distribution instead. A percentage of particles in the swarm are treated as the "quantum cloud", and at each iteration the cloud is randomised in the spherical area with radius r_{cloud} centred around the global best particle of the swarm. The particles that do not constitute the cloud behave according to the standard PSO velocity and position update equations. Since the quantum cloud is completely randomised at each iteration, the swarm does not completely converge on a small area; hence swarm diversity is preserved. The non-quantum particles refine the current solution while the quantum cloud searches for new and better solutions. In this manner, a good balance between exploration and exploitation is achieved.

The problem of outdated memory is again addressed by complete re-evaluation of the swarm memory.

Fine-Grained PSO

The term *fine-grained PSO* was introduced by Li and Dam in [79]. It is in fact nothing more than a reinitialising PSO that makes use of the Von Neumann topology. Such choice is based on the fact that the grid-like Von Neumann structure does not promote information through the swarm as fast as other topologies do, which improves diversity maintenance [79], addressing the diversity loss problem. The fine-grained PSO does not offer any specific way of dealing with the outdated memory problem, and re-evaluation of each particle's current position, local best position, and global best position is usually used. Since the Von Neumann topology was shown to be beneficial in dynamic optimisation [79], it was used in all PSO experiments conducted for this study.

4.3.3 Optimisation in the Presence of Concept Drift

The term "concept drift" belongs to the data mining field, and refers to the phenomena of drifting concepts as observed in data sets over time. Most existing approaches to handle concept drift do not deal with the process of concept learning, but rather with the way in which data is presented to the learner [69, 74, 123, 130]. Three major branches of concept drift handling techniques can be distinguished, namely instance selection, instance weighting, and ensemble learning [116].

In a dynamic environment, the learner can never assume the current predictive model to be final. This implies that the learner must always try to learn new, up-to-date information from the environment. In case of data mining in presence of concept drift, the up-to-date information can only be learned from up-to-date data, and up-to-date data is obtained by instance selection [74, 130]. The goal of the learner comes down to discovering concepts in data, and if data accurately represents the existing concepts, the learner's success will depend entirely on the learner's ability to train and generalise. The simplest form of instance selection is windowing, implemented by fixing the number of instances in the training set and dismissing the oldest instances as new instances arrive [130]. More complex forms of instance selection that delete noisy, irrelevant and redundant instances have also been developed [26, 74]. The windowing approach to instance selection was applied in this study, described in more detail in Chapter 6.

Instance weighting is sometimes used instead of instance selection with the learning algorithms that have the ability to process weighted instances [69]. An example of such algorithms are support vector machines (SVMs) [23]. Instances are weighted according to their age and relevance, and most recent and most relevant instances contribute the most to the learning process. This study, however, deals with NN training, to which instance weighting is not applicable.

A more advanced approach that deals with the classifiers rather than the training data is ensemble learning [102, 114]. With ensemble learning, a selection of classifiers is combined. Each classifier maintains a separate concept description. The quality of concept descriptions provided by each classifier is measured, and the best-performing classifier has the most influence on the final classification [114]. Worst performing classifiers can be dynamically replaced by new classifiers, which start learning the concept from scratch. The chances of obtaining at least one acceptable concept description are increased by training a number of separate classifiers on the same problem over different time periods. Much research on handling concept drift using classifier ensembles was done [19, 102, 111, 114, 115, 116]. However, the ensemble approach treats individual classifiers as black boxes, and does not look into the learning process of a classifier. Ensemble classifiers offer no training algorithm improvements to make each classifier more adaptive and flexible in dynamic environments, assuming redundancy to be the only effective solution. Redundancy can indeed be effective, however, the performance of ensemble classifiers can be further improved by improving the performance of each single classifier (a NN, in case of this study) by adapting the learning process, or, in other words, the training algorithm, to the drifting concepts.

As opposed to the aforementioned concept drift handling techniques, this study deals with the learning process of a single classifier, and the chosen classifier is a NN. Dynamic NN training algorithms are suggested, and the dynamic properties of back propagation are studied.

4.4 Summary

This chapter provided an overview of various characteristics of dynamic environments. Both dynamic optimisation problems and concept drift have been defined and discussed. An overview of dynamic PSO algorithms applied in this study was given, and the main existing approaches to handle concept drift have also been described.

Population based algorithms such as PSO can be adapted to work well in dynamic environments. However, these algorithms have never been applied to concept drift problems before. The next chapter discusses how dynamic PSO can be applied to training NNs in the presence of concept drift, and why such application can be beneficial.

Chapter 5

Neural Networks in Dynamic Environments

When its wings stop moving and the butterfly is at rest, it changes for the last time... into dust.

Anonymous

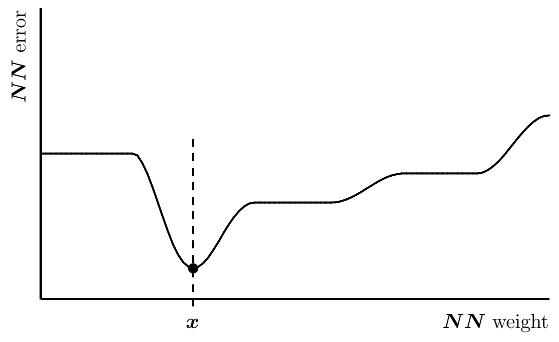
Neural networks (NNs) are widely applied to classification and function approximation problems due to their excellent pattern recognition ability. NNs were applied to problems with drifting concepts before, either as stand-alone classifiers, or in classifier ensembles [123, 126]. However, to the author's knowledge, no training methods specific to dynamic environments were developed. Chapter 4 discussed existing approaches to NN training in the precence of concept drift, and in all of these approaches NNs were treated as black boxes. It was implicitly assumed that the existing training algorithms should be able to adapt to dynamic environments, and in case that fails, completely re-initialising a NN would be the best thing to do [19, 102, 111, 114, 115, 116]. This assumption is not necessarily true. Existing NN training algorithms can be adjusted, enabling a NN to dynamically adapt to the changing environment.

The aim of this chapter is to provide a discussion on NN training algorithms in dynamic environments. The rest of the chapter is structured as follows: Section 5.1 discusses the behaviour of back propagation in dynamic environments, Section 5.2 suggests an alternative approach to NN training in dynamic environments by means of PSO, Section 5.3 highlights the problems that arise when architecture selection and parameter optimisation are considered in the context of dynamic environments, and Section 5.4 describes the problem of overfitting in dynamic environments. Section 5.5 concludes the chapter.

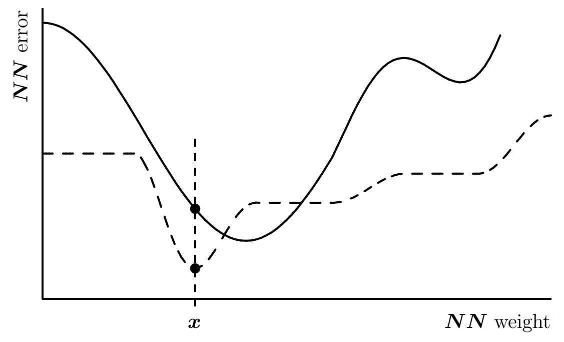
5.1 Dynamism of Back Propagation

Back propagation (discussed in Chapter 3) is a gradient descent method, which implies a hill-climbing approach to training. The algorithm minimises the objective function by following its steepest slope. In order to visualise the behaviour of back propagation in a dynamic environment, it is important to remember that the objective function being optimised is the error function of the NN, and not the mapping between inputs and targets as represented by the data set. This mapping is thus the context in which the error function exists, and changes in the mapping, or, in other words, in the environment, yield changes in the error function. A change in the environment may yield an increase in NN error, since the current weight vector would no longer accurately represent the environment. An increase in NN error implies that the current position is not necessarily an optimum anymore, thus the gradient descent may start climbing downhill again. This automatic response to environment changes makes back propagation an *implicitly dynamic* training algorithm [59].

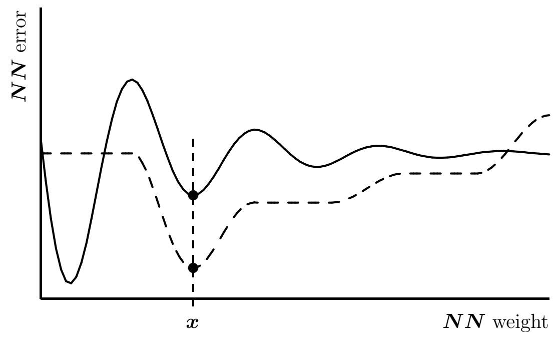
However, the success of such implicit dynamism is largely dependent on the landscape of the error function. The error function landscape may have flat regions where gradient



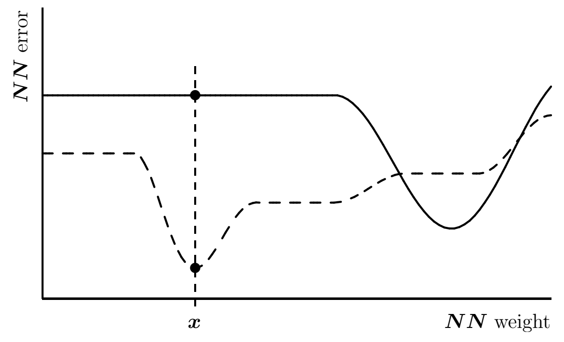
(a) Error function before environment change, x is the current position



(b) Current position becomes a position on the slope after environment change



(c) Current position becomes a local minimum after environment change



(d) Current position becomes a position on a plateau after environment change

Figure 5.1: Back Propagation in dynamic environments

descent is inefficient, and may have local minima which may trap gradient descent [35, 44, 95]. Dedicated studies of NN error surfaces [38, 55] have shown that the main geometrical features of the error surfaces are large plateaus, sometimes asymptotically tending towards infinity, as well as step-like transitions, narrow valleys, and steep ridges. When the error surface changes due to a change in the environment, the current position of the weight vector may happen to map to a region of the changed error surface that is hard to optimise, yielding poor adaptation to the change. A few examples of possible scenarios are given below.

Figure 5.1(a) illustrates the error function before a change in the environment. For the sake of clarity, 2-dimensional space is used, where x refers to the current position of the NN in the weight space. Figure 5.1(b) illustrates a scenario where a change in the environment causes the current position to be on the slope of the error function. Under such scenario, back propagation exhibits implicit dynamism and will find the new optimum by following the error function slope in the downhill direction.

Figure 5.1(c) illustrates a scenario where the changed error function causes the current position to become a local minimum. As a hill-climber, back propagation will not be able to escape the local minimum, and will therefore fail to locate the global minimum.

Figure 5.1(d) illustrates a scenario where the current position becomes a position on a flat region after the environment change, once again preventing gradient descent from discovering the new global optimum.

Thus, even though implicit dynamism of back propagation can not be altogether denied, back propagation should not be relied upon as an ultimate dynamic training algorithm, since a number of dynamic scenarios exist under which back propagation will fail to either detect or track the changes.

The success of back propagation is also highly dependent on the initial set of weights [35, 44, 95], and choosing a good starting point is crucial for algorithm convergence. In the context of dynamic environments, the surface of the error function changes, and the success of adaptation after each change also depends on the current weight vector. Thus, the algorithm's success becomes dependent not only on the initial weights, but also on the current weights.

This study evaluates the implicit dynamism of back propagation. Two variants of back propagation were applied on a selection of dynamic classification problems: standard back propagation (BP), as described in Chapter 3, and reinitialising back propagation (RBP). Reinitialising back propagation applied the standard back propagation algorithm between environment changes, and completely reinitialised NN weights and biases whenever an environment change occured. Reinitialising back propagation was used to test the supposition, made in [19, 102, 111, 114], that restarting back propagation after environment changes is an efficient approach to NN training in the presence of concept drift.

Due to the pitfalls described above, back propagation can not be relied upon as the best possible NN training algorithm for dynamic classification problems. Therefore, the authors of this thesis suggested applying PSO to train NNs in the presence of concept drift.

5.2 Population-Based Dynamic Training

As discussed in Chapter 3, population-based algorithms such as PSO have shown to be efficient alternatives to training NNs in static environments [44, 67, 83, 118]. A detailed discussion of PSO was given in Chapter 2, and the application of PSO to NN training was discussed in Chapter 3. Dynamic versions of PSO have also been developed (refer to Chapter 4), and were successfully applied on a selection of dynamic function optimisation problems [9, 7, 98].

Due to the success of these dynamic PSO algorithms and the efficiency of PSO as a NN training algorithm, this study hypothesises that dynamic PSO algorithms can be applied to efficiently train NNs in the presence of concept drift. To test this hypothesis, a number of dynamic PSO algorithms were applied to train NNs on a selection of dynamic classification problems. To the author's knowledge, this is the first study that analyses dynamic PSO training algorithms.

PSO is a population-based algorithm, thus it uses multiple particles, or weight vectors, to conduct the search for the optimal set of NN weights. If the weight vectors are sufficiently scattered across the weight space, the PSO will have a high probability of escaping search stagnation, such as illustrated in Figures 5.1(c) and 5.1(d), since the particles that end up in local minima, saddle points, or flat regions of the error function, will communicate with the rest of the swarm, and may be guided towards more fruitful regions of the search space by the particles that are located on the error function slope. Dynamic PSO algorithms increase the probability of escaping stagnation by preserving swarm diversity, thus increasing the probability of having particles located on the slope of the error function after a change in the environment.

The search space of the PSO is unbounded, and, although the NN weight space is not bounded either, the studies of NN error surfaces suggest that error surfaces tend to have large flat regions asymptotically approaching infinity [38, 55]. Unbounded exploration exhibited by the swarm is thus likely to lead to unfruitful search space regions. In order to get any guidelines to the possible bounds of the active search space, a more substantial investigation of NN error surfaces is needed. Such investigation is out of the scope of this thesis, and is proposed as a topic for future research.

Dynamic PSO algorithms used in this study were discussed in detail in Chapter 4. The following dynamic PSO algorithms were applied to NN training in this work:

- Reinitialising PSO (RPSO) (refer to Section 4.3.2)
- Charged PSO (CPSO) (refer to Section 4.3.2)
- Quantum PSO (QPSO) (refer to Section 4.3.2)

The research field of dynamic PSO is still relatively young, and no standard or optimal approach to optimisation in dynamic environments with PSO has been identified yet. The three algorithms listed above were chosen based on their relative popularity. The reinitialising PSO was chosen as the most natural, naive way of adapting the standard PSO to dynamic environments. The charged PSO and the quantum PSO were chosen due to the relatively solid theoretical and empirical research behind them that showed these algorithms to be effective on a selection of dynamic optimisation problems (refer to [9, 7, 5, 78, 98, 99]). All three dynamic PSO algorithms are based on the principle of preserving swarm diversity, and each of the three algorithms provides a unique approach to the task of preserving swarm diversity while retaining the exploitation ability of the swarm.

5.3 Architecture Selection and Parameter Optimisation in Dynamic Environments

Section 3.4 in Chapter 3 discussed the influence of various factors on the performance of NN training algorithms. It was argued that choosing an appropriate NN architecture is crucial to avoid such phenomena as underfitting and overfitting, caused by using too few or too many hidden units. Training a NN in a dynamic environment introduces an extra complication to architecture selection: for a dynamic classification problem, new decision boundaries may appear as the time progresses, and previously learned boundaries may disappear. Since the optimal number of hidden units depends on the number of decision boundaries, as well as the shape of these boundaries, it is important that the NN architecture also be adapted with environment changes to prevent underfitting or overfitting. However, this topic is outside the scope of this work, and is only mentioned here as an important direction for future research. Most of the problems used in the empirical study conducted for this thesis were artificially generated, and the total number of decision boundaries fluctuated in preset ranges. The complexity of decision boundaries (i.e. boundary shape, such as linear, curved, etc.) was also known beforehand, and did not significantly change throughout the algorithm run. The number of hidden units was chosen empirically for each problem (refer to Chapter 6) and remained static throughout the algorithm run.

Data pre-processing and weight initialisation, discussed in Chapter 3, are carried out only once, at the beginning of the search process, therefore the presence or absence of concept drift in the training data has no influence on how these two steps are handled.

Tunable parameters of a training algorithm also have an influence on the NN performance. The algorithm parameters of back propagation were discussed in Chapter 3, the algorithm parameters shared by all variations of the PSO algorithms used in this study were discussed in Chapter 2, and the algorithm parameters specific to each dynamic PSO algorithm were discussed in Chapter 4. In static environments, the algorithm parameters are usually either set to static values chosen empirically, or adapted throughout the algorithm run [60, 80, 89, 138, 139]. It is not clear what approach to parameter optimisation would be the most appropriate in the context of dynamic environments. Static optimal values may become suboptimal as the time progresses, and the strategies for adaptive parameters were developed with static environments in mind, generally aiming at speeding up convergence [60, 80, 89, 138, 139], which is not necessarily desirable when dynamic environments are concerned. For the purpose of this study, static values for training algorithm parameters were chosen such that the average error over all algorithm iterations is minimised. Developing adaptive algorithm parameters for dynamic environments is out of the scope of this study and should be considered as possible future work.

5.4 Overfitting in Dynamic Environments

As discussed in Chapter 3, overfitting is another name for the inability of a NN to generalise, or, in other words, to correctly classify previously unseen patterns. A NN that can not generalise has little practical use, since the whole purpose of NN training is to create a predictive model that can approximate the output of a previously unseen pattern without human assistance.

Overfitting occurs when the NN architecture is too large, when the NN is trained for too long, and when there is noise in the training data [32]. The problem of architecture selection in the context of dynamic environments was discussed in Section 5.3. The problem of training for too long becomes more severe in the presence of dynamic changes, since NN training is never stopped when a dynamic problem is optimised. Continuous training is carried out so that the training algorithm could detect the changes and adapt the NN weights when a change occurs. The drawback of continuous training is the increased probability of overfitting due to memorisation of patterns by the NN. The risk may not be as severe in environments where changes occur frequently, i.e. where the training algorithm is not given enough time to overfit. However, in most real-life problems, concept drift is rather scarce [114], thus continuous training may prove an ineffective approach to real-life dynamic classification problems. A possible alternative to continuous training would be to have a separate change-detecting agent that would trigger the training algorithm when a change occurs and suspend training when signs of overfitting are observed. Developing such agents is outside the scope of this thesis, and is proposed as a topic for future research.

As discussed in Section 4.3.3, instance selection is an important aspect of training in the presence of concept drift, since outdated training data may confuse the learner and slow down the process of adapting to changes in the environment. Outdated data may also be a cause of overfitting, since training data that no longer represents the mapping between inputs and targets can be considered as data noise. In this study, a windowing approach to instance selection was applied, described in detail in Chapter 6.

A study of overfitting in the context of dynamic environments is outside the scope of this thesis, however, it is a very important topic for future research.

5.5 Summary

This chapter discussed the ability of back propagation to train NNs in dynamic environments. Problems with back propagation in dynamic environments were discussed and illustrated. A motivation was given for the application of dynamic PSO algorithms to train NNs in dynamic environments. It was formally hypothesised that dynamic PSO algorithms can be applied to efficiently train NNs in the presence of concept drift. Architecture selection, parameter optimisation, and the problem of overfitting in dynamic environments were also discussed. The next chapter will present the experimental study conducted for the thesis.

Chapter 6

Empirical Analysis

Absence of evidence is not evidence of absence.

Martin Rees

This chapter provides a detailed description of the experimental procedure followed and experimental results obtained for this study. Five classification problems were considered under a number of different dynamic scenarios. The main goal of the experimental work was to test the efficiency of dynamic PSO algorithms applied to NN training in the presence of concept drift. The performance of dynamic PSO algorithms is compared to the performance of back propagation and back propagation with reinitialisation, and the behaviour of these algorithms in the presence of concept drift of varying spatial and temporal severity is investigated.

The rest of the chapter is structured as follows: Section 6.1 describes the experimental procedure followed. Experimental results are presented in Section 6.2. Section 6.3 summarises the experimental conclusions arrived at, and ends the chapter with some final remarks.

6.1 Experimental Procedure

This section describes the experimental procedure followed. Section 6.1.1 describes the performance measures used in the experiments. Section 6.1.2 describes how problems with concept drift were simulated. Section 6.1.3 provides a description of the parameter optimisation process.

6.1.1 Measuring NN Performance in Dynamic Environments

The goal of the standard algorithm performance measures developed for static environments is to indicate the quality of the algorithm, i.e. reliability, efficiency, and robustness, as well as the quality of the found solution (accuracy) at the current algorithm state [33]. Usually, the performance measurements obtained at the last algorithm iteration are used to evaluate the algorithm performance. In dynamic environments, however, the algorithms must not only find an optimum, but also detect changes in the optimum, track the optimum, and locate new better optima as they appear. Clearly, standard performance measures that reflect the current algorithm state only do not provide any information regarding the change detection and response to the change exhibited by the algorithm, and thus cannot be used in dynamic environments.

A number of performance measures for dynamic environments have been proposed in the literature, such as the difference between the global optimum and the current solution just before a change [113], the Euclidean distance from the current solution to the global optimum at each algorithm iteration [125], and the average best solution at each iteration over many runs of the same algorithm on a dynamic problem [2, 39, 43], amongst others. However, most of these measures assume either knowledge of the global optimum position, or knowledge of when exactly the change is going to occur. Where NN training is concerned, knowledge of the position of the global optimum beforehand is impossible due to the “black box” nature of NNs; thus, the measures which assume such knowledge are inapplicable to NN training. Timely prediction of the occurrence of dynamic changes is also infeasible when a real-life scenario is considered. Above all, the listed measures require consideration of the progression of the measurement through the algorithm run instead of providing a single measurement value. Even though a study of the measurement progression gives an idea of overall algorithm performance, it is not clear how two different algorithms can be statistically compared when such a performance measure is used. This is why Morrison [85] suggested that a representative performance measure in a dynamic environment should reflect algorithm performance "across the entire range of landscape dynamics". Morrison [85] proposed that the *collective mean fitness*, or the average over all previous fitness values, be used as given by:

$$F_{mean}(T) = \frac{\sum_{t=1}^{T} F(t)}{T} \tag{6.1}$$

where T is the total number of iterations, and $F(t)$ is algorithm fitness after iteration t. The term "fitness" is borrowed from the evolutionary computation (EC) field, and refers to an applicable measure that reflects the quality of the current solution. In case of NN training, the mean squared error (MSE) is usually used as a fitness measure.

Collective mean fitness represents the entire algorithm performance history, hence it gives an indication of the adaptive properties of the algorithm. This measure allows for convenient statistical comparison between algorithms, and does not depend on any additional knowledge about the search space such as the location of the global optimum. The collective mean fitness was used as a performance measure in all the experiments conducted in this study.

When referring to fitness in the context of NNs, it should be clarified what exactly is meant by this term. In the current work, the MSE calculated over the data set during each epoch was used to measure algorithm fitness at each iteration. This measure reflects the quality of a NN, i.e. the NN's ability to recognise the training patterns for the training MSE (E_T), and the NN's ability to generalise for the generalisation MSE (E_G). It was theoretically shown by Wan [121] that minimisation of the MSE consequently minimises the probability of misclassification. Thus, MSE does not loose its meaning when classification problems are concerned.

Since a study of overfitting is outside the scope of this work, no measures were taken to prevent overfitting of training data. Counter-overfitting techniques developed to date were designed for static problems [75, 131, 140], and, to the author's knowledge, no studies of overfitting in the context of dynamic environments were published to date. Existing techniques may require modifications in order to become applicable to dynamic problems. For a discussion of overfitting in the context of dynamic environments, refer to Chapter 5. The generalisation error was nonetheless recorded throughout the experiments and reported in the analysis that follows. The generalisation factor, ρ_F, developed by Röbel [100], was used as a measure of overfitting. The generalisation factor is calculated by taking the ratio between the generalisation error, E_G, and the training error, E_T:

$$\rho_F = \frac{E_G}{E_T}$$

The generalisation factor is not an absolute measure of overfitting, and both E_T and E_G should still be considered as more precise measures of training algorithm performance. However, ρ_F gives an indication of the overfitting behaviour of an algorithm. A $\rho_F < 1$ is desirable, because $E_G < E_T$ is an indication of good generalisation performance of the NN. A $\rho_F > 1$ implies that $E_G > E_T$, which is an indication of overfitting. Since this study considers dynamic problems only, ρ_F, like the other performance measures used, was calculated according to equation (6.1). Thus, the reported values of ρ_F reflected the generalisation factor across the entire algorithm run.

For the dynamic PSO training algorithms considered, swarm diversity was also measured. The diversity measurement used is the average distance around the swarm center [70], given by:

$$d = \frac{1}{S_P}\sum_{y=1}^{S_P}\sqrt{\sum_{l=1}^{n}(x_{yl} - \bar{x}_l)^2}$$

where S_P is the swarm size, n is the dimensionality of the problem space, $\mathbf{x}_y$ and $\bar{\mathbf{x}}$ are particle position y and the swarm center, respectively. The choice of diversity measure is based on [87], where this measure was shown to be a valid diversity measure. As with the other measures used in this study, average swarm diversity was reported, calculated using equation (6.1).

Whenever experimental results were compared, the two-tailed non-parametric Mann-Whitney U test [82] was used to determine whether the difference in algorithm performance was statistically significant. The choice of the significance test is based on [27], where the authors showed that the Mann-Whitney U test is safer than the 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 corresponded to rejection of the null hypothesis that there is no statistically significant difference between the sample means. For the sake of convenience, all p-values were bounded below by 0.0001.

All reported results are averages over 30 independent simulations. The next section describes how concept drift of varying spatial and temporal severity was simulated.

6.1.2 Simulating Concept Drift

In this study, one real-life data set was used, and another four data sets were artificially generated to simulate dynamic classification problems of varying dimensionality and decision boundary shape. The process of generating a data set with concept drift applied in this study is described below.

M points were randomly chosen from the specified domain. M data patterns were then obtained by assigning a target classification to every input vector according to current problem-specific decision boundaries. The boundaries were updated N times, thus simulating N environment changes. After every such change, the target classification of every pattern, $p = 1, \ldots, p_M$, was updated accordingly, and the updated M patterns were appended to the previous M patterns. Thus, the total number of data patterns in a complete data set is calculated as follows:

$$P = M + M * N$$

Classification problems with concept drift were simulated by sliding a *window* over such a data set. Windowing is a simple pattern selection technique widely applied in data mining of continuous data streams [130]. For more information on windowing and pattern selection, refer to Chapter 4. In this study, the size of the window was set to M, thus at every iteration the NN was presented with data patterns which represented a complete set of M points. The data patterns inside the window were split into two subsets for training and generalisation purposes: 80% of the patterns were used as a training set, D_T, and the other 20% were used as a generalisation set, D_G. Since the aim was to simulate decision boundaries that change over time, the original data set was not shuffled to preserve the pattern order. The patterns were only shuffled inside the window before being split into D_T and D_G to prevent NNs from learning the pattern order instead of the classification boundaries.

Shifting the window by S patterns implies discarding the first S patterns from the window, and appending the next S patterns from the data set to the window. The window step size, S, controls the spatial severity of changes: changes become more drastic for larger values of S, since a lot of new information is added while a lot of previously valid information is discarded. If the decision boundaries change every M patterns in the data set and the window size is equal to M, a shift by $S < M$ patterns may introduce *new decision boundaries* into the window while still keeping the data patterns classified according to the previous decision boundaries inside the window. An example to illustrate this process is shown in Figure 6.1. Here, $M = 6$ and $S = 4$. When the window shifts, four patterns, $\{p_1, p_2, p_3, p_4\}$, are discarded, and new patterns $\{p'_1, p'_2, p'_3, p'_4\}$ are

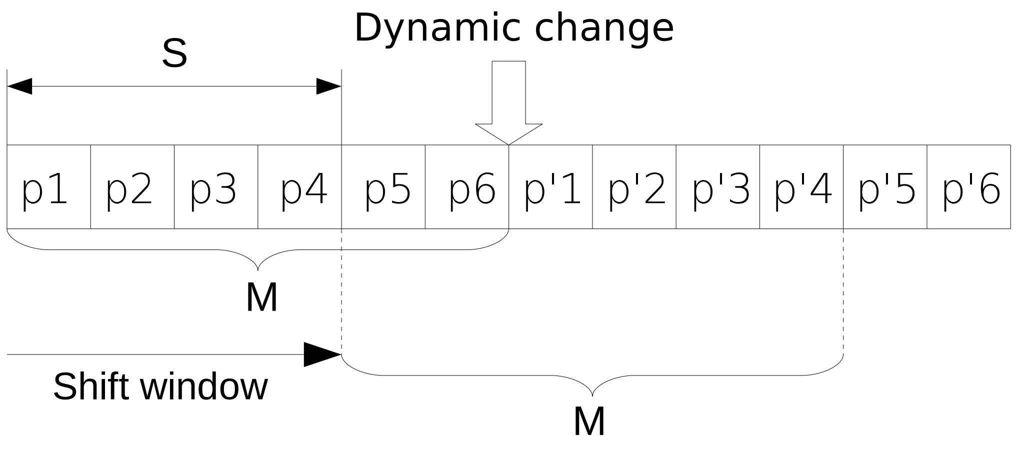


Figure 6.1: Introducing new decision boundaries by window shifts

added, while p_5 and p_6 remain in the window. Thus the shifted window contains patterns representing the environment both before and after the dynamic change, or, in other words, the window contains patterns representing both the old decision boundaries and the updated decision boundaries. As the window slides along the data set, more patterns classified according to the previous decision boundaries are replaced by the patterns classified according to the current decision boundaries, until the previous boundaries are completely discarded. This implies that the training algorithm will often have to deal with more than one decision boundary within the data set, with a possibility of conflicting boundaries that contradict each other, as shown in Figure 6.2. Conflicting boundaries make dynamic adaptation more challenging for the training algorithms.

Two major characteristics of a dynamic problem are: (a) spatial severity of changes, and (b) temporal severity of changes. The exact meaning of these terms applied to problems with concept drift is given in Chapter 4. In order to provide a representative coverage of the existing types of concept drift, different combinations of spatial severity and temporal severity were simulated, resulting in a number of different dynamic scenarios. Every dynamic scenario is characterised by two variables:

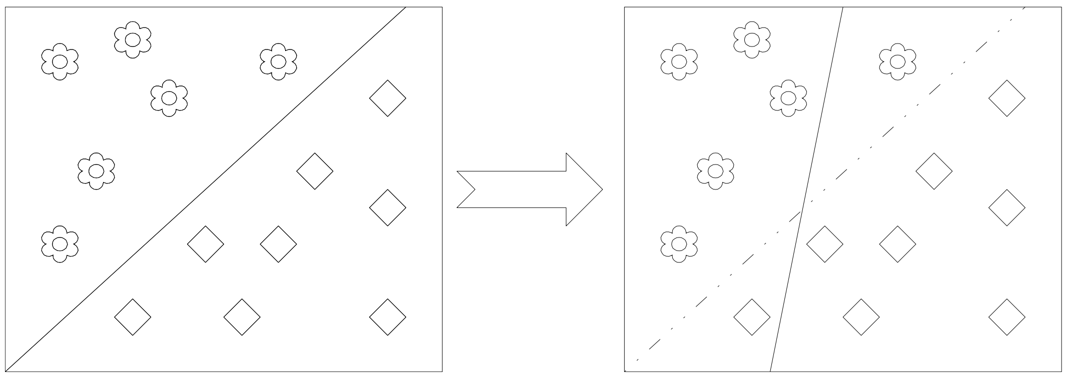


Figure 6.2: Conflicting boundaries

- The step size, S, which refers to the number of patterns by which the window shifts in order to simulate an environment change. This attribute determines the abruptness of the environment change, or, in other words, the level of spatial severity.
- The number of algorithm iterations, F, that the current training algorithm is allowed to run before the window shifts. This attribute controls the frequency of changes, or, in other words, the level of temporal severity.

Evaluation of the considered training algorithms under all possible combinations of values of the above two variables will provide a perfect coverage of possible types of concept drift. However, since such an exhaustive evaluation is practically infeasible, discrete ranges of values were considered for both variables on every problem. All possible combinations of values in these discrete ranges were considered. The discrete ranges were problem-specific, and are reported later in this chapter.

The next section outlines the parameter optimisation method applied in this work.

6.1.3 Parameter Optimisation

Since the aim of this research was to test whether dynamic PSO algorithms can be applied to efficiently train NNs in the presence of concept drift, measures had to be taken to ensure that the dynamic PSO algorithms considered exhibited efficient performance. Performance of the dynamic PSO algorithms is strongly dependent on the algorithm parameters, and the relevant parameters were optimised according to the procedure described below. In order to ensure a fair comparison between the performance of back propagation and the performance of the dynamic PSO algorithms, the algorithm parameters of back propagation and back propagation with reinitialisation were also optimised.

An iterative approach to parameter optimisation was used. Parameters were optimised one at a time. For each parameter, the algorithm was tested under a selected range of possible values for this parameter, while the other parameters remained fixed. In order to keep the optimisation process statistically sound, 30 runs were conducted for every value in the chosen discrete range. The parameter value yielding the lowest average training and generalisation errors was subsequently chosen as optimal, and optimisation proceeded to the next parameter. For optimisation of the remaining parameters, all the parameters already optimised were fixed to their best values.

It is important to note at this point that parameter optimisation, alike to simulating different types of concept drift described in Section 6.1.2, should be treated as limited and thus approximate. As already discussed in Chapter 5, static parameter values may yield suboptimal performance after an environment change, and, although static values were chosen such that the average error over all iterations was minimised, parameters that adapt themselves to environment changes could yield better algorithm performance than static parameters. Unfortunately, no adaptive parameter strategies specific to dynamic environments have been developed yet, and the development of such adaptive parameter strategies is proposed as possible future work.

Another limitation of the parameter optimisation process was computational time constraints. Just as it is impossible to exhaustively test the chosen training algorithms under all possible dynamic scenarios, it is computationally infeasible to optimise all possible parameters under every scenario considered. Discrete ranges for parameter values were chosen to limit the total number of simulations required, and the parameters of each algorithm were optimised once for every problem, instead of once for every scenario. Thus, problem-specific error surfaces were taken into account by the optimisation process, but not the specific environment changes simulated in each scenario. The resulting comparison of algorithms should thus be treated as relative instead of absolute. The authors believe that the selected instances of algorithms, problems, and dynamic environments provide a good general idea of the entire range of existing configurations of both the algorithms and the dynamic scenarios.

Specific parameters optimised for each training algorithm are discussed below.

Back Propagation

The parameters that influence the performance of back propagation are:

1. The interval in which the **initial NN weights** are randomised. Since back propagation is known to be sensitive to the initial weight values [55], extra care has to be taken in choosing them. The intervals considered are listed in Table 6.1. Since the sigmoid function was used as an activation function in all the experiments conducted, any interval larger than $[-5, 5]$ would have caused derivative values to be approximately equal to 0, thus diminishing weight adjustments and preventing efficient learning.

2. **Learning rate:** This parameter determines the step size taken by the algorithm in the direction of the negative gradient of the error surface. The values tested are listed in Table 6.1.

3. **Momentum:** This parameter controls the extent to which the memory of previous weight changes influences weight changes in the current epoch. The values considered are listed in Table 6.1.

Table 6.1: Parameter Ranges for Back Propagation

Parameter	**Range**
NN weight initialisation range	$\{[-1,1],[-2,2],[-3,3],[-4,4],[-5,5]\}$
Learning Rate	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$
Momentum	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$

All NNs used a single hidden layer. The number of hidden units was fixed for each problem throughout the experiments. Dynamic problems and the corresponding number of hidden units are listed in Table 6.2. These numbers were determined iteratively, using the approach described in Section 6.1.3. More sophisticated architecture selection techniques such as network growing [37, 49, 71], pruning [32, 110], and regularization [41, 131] were developed for static environments, thus they were not applied in the experiments.

Particle Swarm Optimisation

The following PSO parameters were optimised:

1. **Maximum velocity ($\vec{V}_{max}$):** This parameter limits the step size of a particle and thus prevents particles from moving too fast. Enforcing a speed limit promotes exploitation. Parameter values considered are listed in Table 6.3. The same value was used across all dimensions of $\vec{V}_{max}$, hence it is referred to as V_{max} further in the text. Positive infinity is used to refer to unconstrained particle velocity.

2. **Swarm size:** Larger swarms take longer to converge, but explore more if the particles are initialised uniformly. The swarm sizes tested are listed in Table 6.3.

The NN weight initialisation range was also optimised iteratively. In the case of PSO training, the initial NN weights correspond to the initial particle positions. Since

Table 6.2: Number of Hidden Units

Problem	**# Hidden Units**
Moving Hyperplane	6
Dynamic Sphere	4
Sliding Thresholds	3
Electricity Pricing	6
SEA Concepts	4

the sigmoid function was used as the activation function, large initial weights may have led the swarm into unfruitful regions of the error surface (such as plateaus) due to the asymptotic nature of the sigmoid function. The weight initialisation ranges considered are listed in Table 6.3.

As already discussed in Chapter 4, the *von Neumann* neighbourhood topology was shown to facilitate swarm diversity, and thus facilitate exploration. The von Neumann topology was shown to perform better than other neighbourhood topologies in dynamic environments [79], and therefore this topology was used in all PSO experiments conducted for this study.

For all the experiments, the inertia weight was set to 0.729844, while the values of the acceleration coefficients were set to 1.496180. This choice is based on [30], where it was shown that such parameter settings facilitate convergent behaviour. Although preservation of diversity is vital in the context of dynamic environments, convergent behaviour is still necessary, since a solution must be found in between environment changes.

Initial particle velocities were set to 0.

Each dynamic PSO algorithm required extra parameters to be optimised. Parameter optimisation specific to each dynamic PSO algorithm is described below. Due to the

Table 6.3: Parameter Ranges: PSO

Parameter	**Range**
V_{max}	$\{0.1, 0.5, 1, 2, 5, 10, 20, +\infty\}$
Swarm Size	$\{15, 20, 30, 50\}$
NN weight initialisation range	$\{[-1, 1], [-2, 2], [-3, 3], [-4, 4], [-5, 5]\}$
Reinitialising PSO: Reinitalisation Ratio	$\{0.25, 0.5, 0.75, 1.0\}$
Charged PSO: Charge Magnitude	$\{0.1, 0.3, 1, 5, 10, 20\}$
Quantum PSO: Radius	$\{1, 1.5, 2, 3, 5, 10\}$

infeasibility of empirically testing all possible combinations of parameters, some parameters were assigned values based on theoretical analysis and previously published research instead of empirical evidence.

For the **reinitialising PSO**, the reinitialisation ratio had to be fine-tuned. The value range for this parameter is $[0, 1]$, where 0 results in no reinitialisation, thus turning off the dynamic properties of the algorithm, and 1 results in reinitialisation of the entire swarm. Values considered are listed in Table 6.3, and were chosen to represent progression from reinitialising a quarter of the swarm to a complete reinitialisation of the swarm.

For the **charged PSO**, the charge magnitude of charged particles was optimised. The range of values tested is given in Table 6.3. Radii of repulsion R_c and R_p were set to 1 and 30, respectively. Such a wide range was chosen in order to constrain acceleration the least and thus promote exploration. Following from equation (4.1) given in Chapter 4, the repelling force is inversely proportional to the distance between the charged particles. Thus, the further any two charged particles are from each other, the less they repel. The repelling force is also directly proportional to the charge magnitude. Therefore, the strength of repulsion can be controlled by choosing an appropriate value for the charge magnitude. For the experiments conducted in this study, 50% of each swarm was charged with another 50% kept neutral. Such choice is based on studies published in [7], where it was shown that a good balance between exploration and exploitation is then obtained.

For the **quantum PSO**, the radius of the quantum cloud was optimised. The values considered for this parameter are listed in Table 6.3. For the experiments conducted in this study, 50% of the swarm constituted the cloud. This choice is based on the empirical studies presented in [9], where it was shown that this parameter setting generally promotes a good balance between exploration and exploitation. As described in Chapter 4, the quantum cloud is randomised at every iteration. Thus, a swarm made of quantum particles only will be incapable of exploitation. Since neutral particles are the only exploitation force of the quantum PSO, it is necessary to keep at least 50% of the particles neutral.

6.1.4 Naming Conventions

The following abbreviations are used in this chapter where appropriate for the sake of convenience:

- **BP**: back propagation;
- **RBP**: reinitialising back propagation;
- **RPSO**: reinitialising PSO;
- **CPSO**: charged PSO;
- **QPSO**: quantum PSO;

The rest of the chapter is dedicated to the analysis of experimental results.

6.2 Classification problems

Four synthetically generated classification problems and one real-life classification problem were used in the experiments. The considered problems are discussed in this section, and the empirical analysis of obtained results is presented.

6.2.1 SEA Concepts

This artificially generated problem divided a three-dimensional feature space into two classes.

Problem Definition

This problem was adopted from [111, 116]. The data set consisted of 10 000 patterns, obtained by randomly generating 10 000 3-dimensional points, $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{10000}\}, \vec{x}_l \in [0; 10)^3, l = 1, \ldots, 10\ 000$. The generated points were divided into four equal concept blocks, 2500 points each, as illustrated in Figure 6.3. In each block, the class label of each data point $\vec{x}$ was determined as follows:

$$Classification(\vec{x}) = \begin{cases} Class\ A & \text{if } x_1 + x_2 \leq \theta \\ Class\ B & \text{otherwise,} \end{cases} \tag{6.2}$$

where x_1 and x_2 are the values of the first two dimensions, and θ is the threshold value. Thus, only the first two dimensions determined the class label of a point. Threshold values of 8, 9, 7, and 9.5 were used in the four blocks, and 10% class noise was inserted into each block by changing the class label of randomly chosen data patterns in that block. The patterns were then recorded into a single data set in sequential order, block by block, resulting in a data set of 10 000 patterns. Just as with the previous problems, a window was slided over the data set to simulate a dynamic environment. The window size was fixed to 2500 patterns, equal to the size of a concept block.

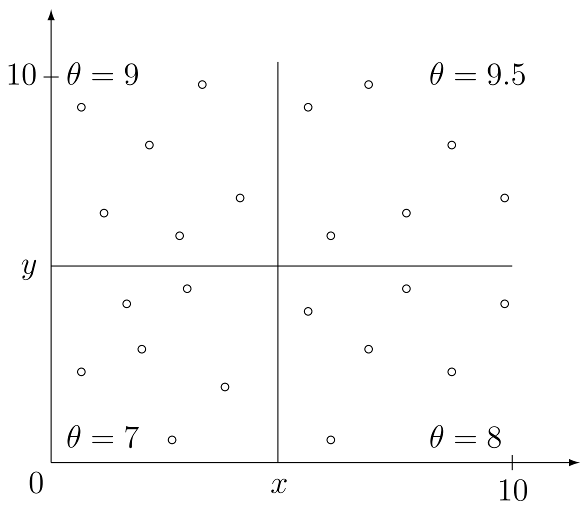


Figure 6.3: SEA Concepts

It should be noted at this point that, in case of NN training, the dimensionality of the optimisation problem is determined by the total number of weights and biases, and not by the dimensionality of the input patterns. As stated in Chapter 3, a NN comprises of three layers - an input layer, a hidden layer, and an output layer. In all the experiments conducted for this study, fully connected NNs were used, thus the total number of NN weights for each problem, taking bias units into account, is calculated as follows:

$$n_w = (I + 1)J + (J + 1)K \tag{6.3}$$

where n_w is the total number of NN weights, I is the number of inputs, J is the number of hidden units, and K is the number of outputs. A NN with 3 input units, 4 hidden units and 1 output unit was trained on the SEA concepts problem. According to equation (6.3), the total number of weights and biases, corresponding to the dimensionality of the problem, was equal to 21.

Twenty different dynamic scenarios outlined in Table 6.4 were applied to the SEA concepts problem. The chosen scenarios simulated different combinations of spatial and temporal severity, providing representative coverage of different dynamic environment types. The technique of simulating dynamic scenarios applied in this study was described

Table 6.4: Dynamic Scenarios for the SEA Concepts Problem

F \ S	50	100	500	1000	2500
10	A1	A2	A3	A4	A5
50	B1	B2	B3	B4	B5
100	C1	C2	C3	C4	C5
250	D1	D2	D3	D4	D5

in Section 6.1.2. As shown in Table 6.4, the values for both the frequency of change and the step size increase non-linearly, because the influence of parameter values was expected to be stronger for small values. Change frequencies of 10, 50, 100, and 250 iterations was considered. Different levels of spatial severity were simulated by shifting the sliding window by 50, 100, 500, 1000, and 2500 patterns per step.

Parameter Optimisation

All algorithm parameters were optimised according to the procedure described in Section 6.1.3. Corresponding optimal parameters discovered are shown in Table 6.5.

Analysis of Empirical Data

For every scenario considered, every training algorithm traversed the entire data set of 10 000 patterns. Since both the step size and the frequency of changes varied from scenario to scenario, every scenario required a different total number of iterations to traverse the entire data set. The number of iterations is calculated as

$$T = F * \frac{P - P_w}{S} + F \tag{6.4}$$

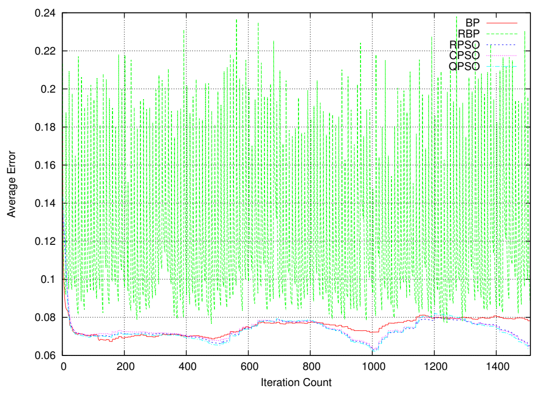
where F is the number of iterations on a window between the window shifts (i.e. change frequency), P is the total number of patterns in the data set, P_w is the window size, and

Table 6.5: Optimal Parameters for the SEA Concepts Problem

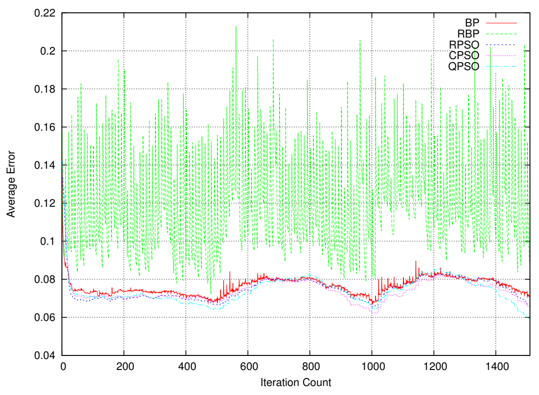
Algorithm	**Parameters**			
Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-5;5]$	0.1	0.7	
Reinitialising Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-5;5]$	0.1	0.7	
Reinitialising PSO	Weight Interval	V_{max}	Swarm Size	Reinit. Ratio
	$[-3;3]$	0.5	50	0.25
Charged PSO	Weight Interval	V_{max}	Swarm Size	Charge Magnitude
	$[-3;3]$	0.5	30	0.3
Quantum PSO	Weight Interval	V_{max}	Swarm Size	Cloud Radius
	$[-1;1]$	0.5	50	1.5

S is the step size. The average error results and the corresponding standard deviation values reported in this work were obtained after the number of iterations as given by equation (6.4). The window size was fixed to 2500 patterns, and both the step size and the frequency of changes were determined by the scenario in use. The rest of this section provides an analysis of the empirical results obtained for the SEA concepts problem.

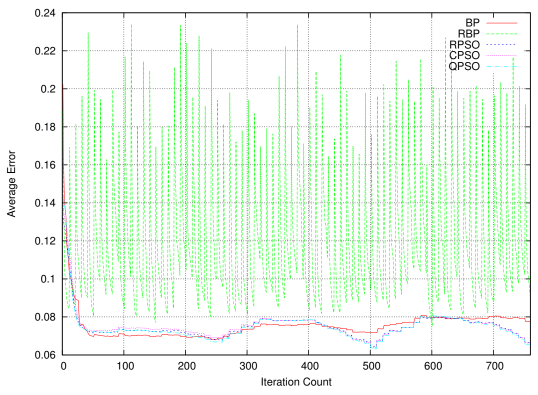
Scenarios A1 to A5: Figures 6.4 and 6.5 illustrate the progression of E_T and E_G over time as observed under scenarios A1 to A5. Error profiles in figures 6.4 and 6.5 show that BP and the three dynamic PSOs perfromed similarly to one another under scenarios A1 to A4, while RBP fluctuated a lot. Scenarios A1 to A5 shared the same frequency of change, equal to 10 iterations. Thus, RBP reinitialised the NN weights every 10 iterations, and had only 10 iterations to converge on a solution, which proved to be insufficient.



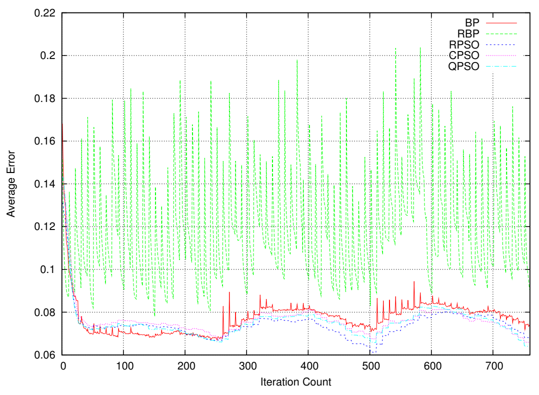
(a) E_T for A1



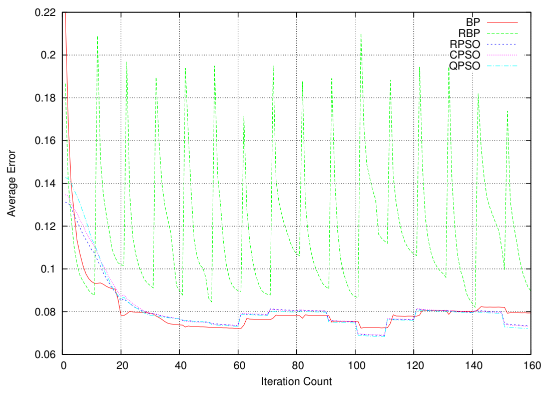
(b) E_G for A1



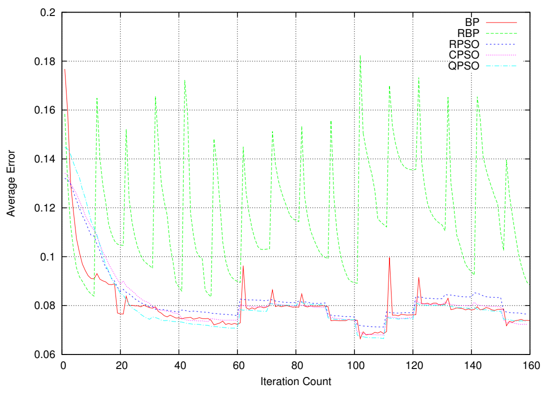
(c) E_T for A2



(d) E_G for A2

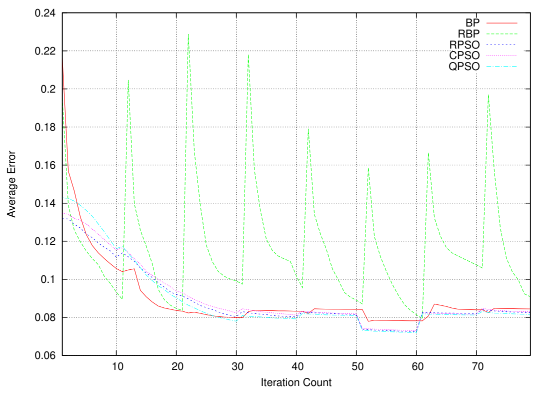


(e) E_T for A3

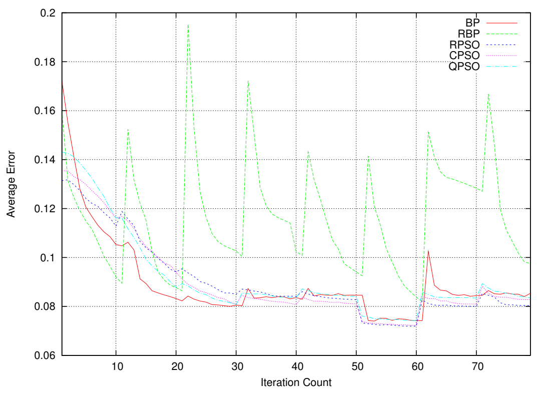


(f) E_G for A3

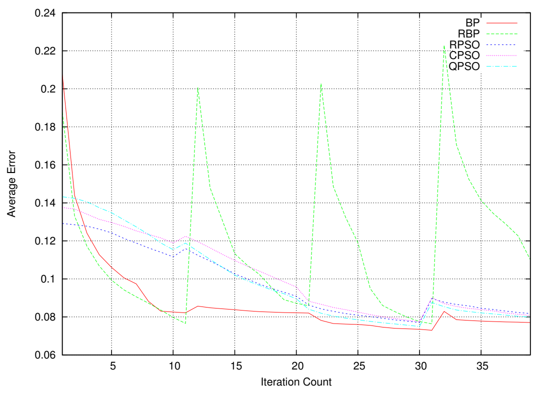
Figure 6.4: Training and Generalisation Error results for SEA concepts, scenarios A1 to A3



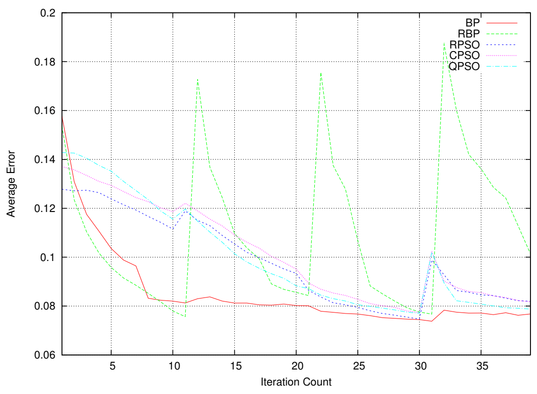
(a) E_T for A4



(b) E_G for A4



(c) E_T for A5



(d) E_G for A5

Figure 6.5: Training and Generalisation Error results for SEA concepts, scenarios A4 to A5

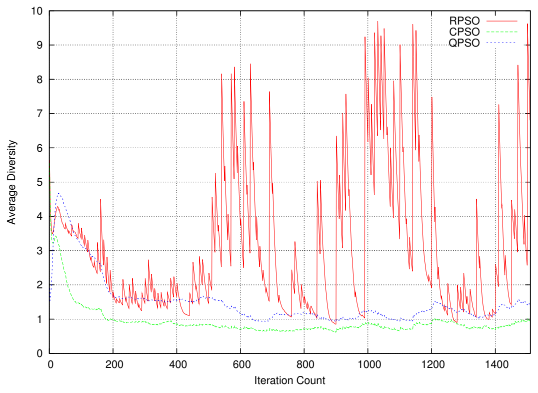
Figures 6.4 and 6.5 illustrate that the increasing spatial severity of changes from A1 to A5 made adaptation more difficult for the training algorithms: generalisation error peaks became higher after every environment change as the spatial severity increased from A1 to A5. Abrupt scenarios discarded a lot of old data and added a lot of new data to the training and generalisation sets at every environment change, and the training algorithms were required to make more significant changes to the current solution under abrupt scenarios.

However, as illustrated in figure 6.5, the generalisation error peaks produced by BP decreased under the most abrupt scenario A5. Under scenario A5, all contents of the sliding window were replaced by new data patterns. Thus, only one concept, and consequently only one decision boundary, was represented in the sliding window at any one time. This made scenario A5 easier to adapt to than scenarios A1 to A4, where the training set often contained patterns from neighbouring concept blocks.

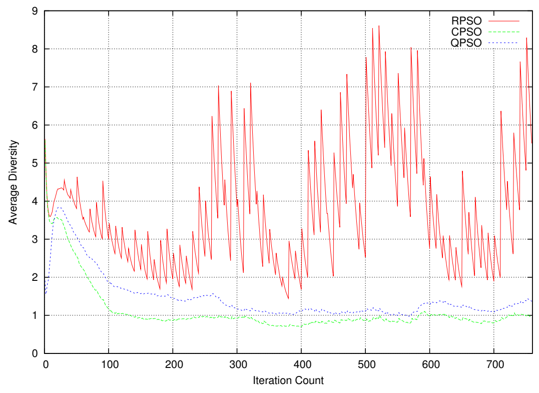
Figure 6.4 illustrates that the dynamic PSOs exploited better than BP under gradual changes: as illustrated in figures 6.4(a) and 6.4(c), the dynamic PSOs reached a lower minimum E_T between environment changes. Such behaviour can be due to the susceptibility of BP to get trapped in local minima.

Figures 6.4 and 6.5 also illustrate that the dynamic PSOs took longer to converge than both BP and RBP in the beginning of the algorithm run. The difference in convergence speed is especially visible in figures 6.5(c) and 6.5(d): the three dynamic PSOs took 25 iterations to reach the minimum attained by BP and RBP in 10 iterations. However, further examination of figures 6.4 and 6.5 shows that, once the dynamic PSOs reached a minimum, the training algorithms maintained the obtained minimum better than BP or RBP: under scenarios A1 to A4, both BP and RBP produced higher E_T and E_G than the dynamic PSOs after every environment change. The only difference is in scenario A5 which simulated the most abrupt changes: as illustrated in figures 6.5(c) and 6.5(d), the dynamic PSOs did not have enough time to converge on a good minimum. Swarm behaviour of the dynamic PSOs can be further analysed by studying the progression of swarm diversity over time, as illustrated in figure 6.6. Under gradual scenarios, the QPSO and the CPSO took between 100 and 200 iterations to converge around a solution, and maintained the obtained swarm diversity throughout the algorithm run. Under abrupt scenarios, especially under the most abrupt scenario A5 (see figure 6.6(e)), the dynamic PSOs were not allowed enough iterations to converge around a solution.

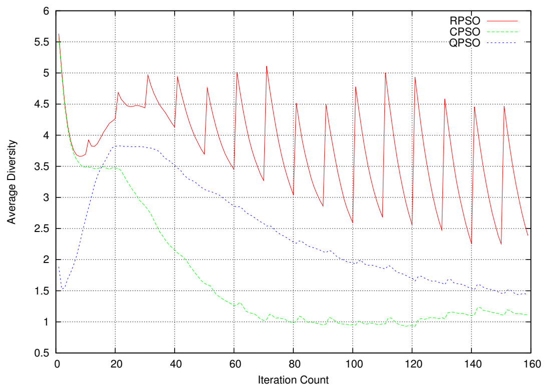
It is interesting to compare the swarm behaviour of the three dynamic PSOs considered to better understand the advantages and disadvantages of each. Figure 6.6 shows that the diversity of the RPSO fluctuated a lot more than the diversity of both the CPSO



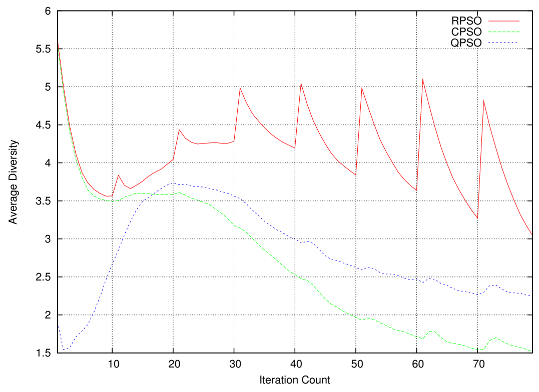
(a) Swarm diversity results for A1



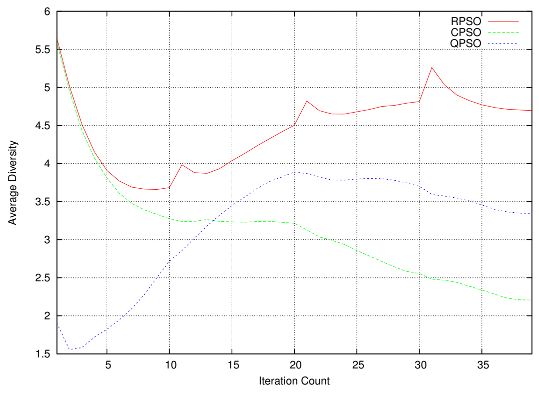
(b) Swarm diversity results for A2



(c) Swarm diversity results for A3



(d) Swarm diversity results for A4



(e) Swarm diversity results for A5

Figure 6.6: Swarm diversity results for SEA concepts, scenarios A1 to A5

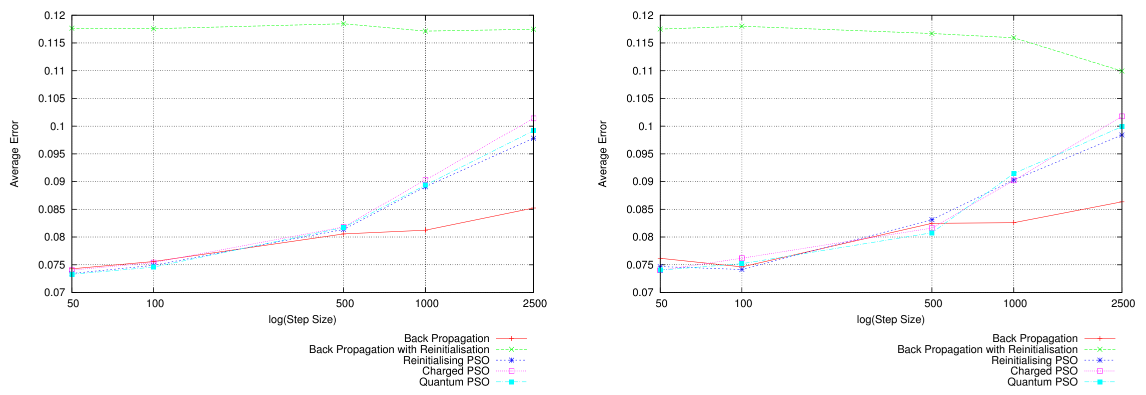
and the QPSO under scenarios A1 to A5. As discussed in Chapter 4, the RPSO simply reinitialises a percentage of randomly selected particles in response to an environment change. While this approach maintains diversity of the swarm, it does not prevent the loss of important information, as opposed to the CPSO and the QPSO, where the global best particle is re-evaluated, but never reinitialised. Weaker dependence upon the swarm memory exhibited by the RPSO promotes exploration. However, the dominance of exploration over exploitation may lead to divergent behaviour. Figures 6.6(a) and 6.6(b) show that the diversity of the RPSO not only fluctuated violently, but also increased over time, which may be interpreted as a sign of divergent behaviour. It was also shown in [119] that bounded activation functions may lead to explosion in velocity values [21]. Although particle velocity was in this case bounded by a rather small value of V_{max}, the sigmoid activation function used in all hidden and output neurons could have been a contributing factor to the divergent behaviour observed. This will be further investigated by van Wyk and Engelbrecht, who have already published results on the significance of the activation function in NN training with PSO [119].

Figure 6.6 illustrates that the QPSO started from a lower initial diversity than the CPSO, but then took longer to converge on a stable diversity value. The swarm diversity behaviour of the CPSO and the QPSO was determined by the optimal algorithm parameters used for each algorithm, as reported in Table 6.5. The QPSO and the CPSO exhibited similar behaviour, taking between 100 and 200 iterations to converge around a minimum solution, and maintaining stable swarm diversity once a minimum was found.

In order to carry out a statistically sound comparison between the different training algorithms, collective mean fitness results were calculated. Table 6.6 summarises the collective mean fitness results, generalisation factor values, and average swarm diversity (where applicable) obtained for scenarios A1 to A5. P-values corresponding to training error comparisons between the algorithms are reported in Table 6.7. P-values corresponding to generalisation error comparisons between the algorithms are reported

Table 6.6: SEA Concepts Results for Scenarios A1 to A5

Scenario / Algorithm	**A1** *(frequency: 10, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.074272 ± 0.00206	0.076167 ± 0.004317	1.05294 ± 0.105819	n/a
Reinit. Back Propagation	0.11765 ± 0.004987	0.117498 ± 0.007621	0.974601 ± 0.037155	n/a
Reinitialising PSO	0.073435 ± 0.000793	0.074669 ± 0.004909	0.998969 ± 0.084563	3.09238 ± 0.891353
Charged PSO	0.074025 ± 0.000728	0.074008 ± 0.004829	0.996476 ± 0.085886	0.932965 ± 0.193641
Quantum PSO	0.073285 ± 0.000881	0.074031 ± 0.004996	1.00672 ± 0.080596	1.51885 ± 0.633338
	A2 *(frequency: 10, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.075578 ± 0.002115	0.074614 ± 0.004726	0.992619 ± 0.102015	n/a
Reinit. Back Propagation	0.117573 ± 0.006597	0.118017 ± 0.006797	0.98338 ± 0.047332	n/a
Reinitialising PSO	0.074902 ± 0.000935	0.074139 ± 0.006028	0.995458 ± 0.095782	3.74929 ± 0.783939
Charged PSO	0.075401 ± 0.000924	0.076178 ± 0.005241	1.01196 ± 0.088621	1.11805 ± 0.184048
Quantum PSO	0.074578 ± 0.000829	0.075184 ± 0.00473	1.00492 ± 0.068956	1.46429 ± 0.395646
	A3 *(frequency: 10, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.080557 ± 0.012953	0.082473 ± 0.012549	1.04732 ± 0.110472	n/a
Reinit. Back Propagation	0.118469 ± 0.016798	0.116704 ± 0.017312	0.952783 ± 0.040457	n/a
Reinitialising PSO	0.081363 ± 0.001426	0.083129 ± 0.00422	1.0132 ± 0.052644	3.78941 ± 1.15251
Charged PSO	0.081821 ± 0.002034	0.08162 ± 0.003807	1.00177 ± 0.041716	1.69461 ± 0.377857
Quantum PSO	0.081712 ± 0.001563	0.080766 ± 0.003235	0.989244 ± 0.03871	2.41796 ± 1.41779
	A4 *(frequency: 10, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.081234 ± 0.012918	0.082579 ± 0.011709	1.04571 ± 0.084766	n/a
Reinit. Back Propagation	0.117146 ± 0.023963	0.115934 ± 0.023393	0.957676 ± 0.049642	n/a
Reinitialising PSO	0.089078 ± 0.003179	0.090288 ± 0.00601	1.01818 ± 0.063056	4.12736 ± 1.0331
Charged PSO	0.090294 ± 0.003153	0.090222 ± 0.004523	1.00199 ± 0.036437	2.66233 ± 1.1681
Quantum PSO	0.089353 ± 0.002067	0.091455 ± 0.005	1.00933 ± 0.039309	2.79456 ± 1.41328
	A5 *(frequency: 10, step size: 2500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.085228 ± 0.036716	0.086351 ± 0.034788	1.02317 ± 0.069827	n/a
Reinit. Back Propagation	0.117479 ± 0.023246	0.109948 ± 0.019195	0.928082 ± 0.049888	n/a
Reinitialising PSO	0.097821 ± 0.005196	0.098403 ± 0.006403	1.00362 ± 0.033017	4.46154 ± 0.783499
Charged PSO	0.101398 ± 0.005385	0.101769 ± 0.006693	0.997178 ± 0.029466	3.11829 ± 0.699315
Quantum PSO	0.099188 ± 0.005198	0.099901 ± 0.00484	1.00341 ± 0.029998	3.17143 ± 0.954826



(a) Average Training Error Results (b) Average Generalisation Error Results

Figure 6.7: Average Error results for SEA concepts, scenarios A1 to A5

in Table 6.8.

Table 6.6 shows that RBP performed consistently and significantly worse that the other four algorithms under scenarios A1 to A5. As it has already been explained, a complete restart of the training algorithm did not prove efficient in such frequently changing environments as simulated by scenarios A1 to A5. Table 6.6 shows that BP performed significantly worse than the dynamic PSOs under the most gradual scenario (A1), showed similar (insignificantly different) performance to dynamic PSOs under slightly more abrupt scenario A2, and significantly outperformed the dynamic PSOs under scenarios of increasing abruptness (A3 to A5) (refer to Tables 6.7 and 6.8 for supporting p-values). As was already determined, the advantage of BP under scenarios of high temporal severity is due to the slower convergence speed of the dynamic PSOs.

Table 6.6 also shows that the three dynamic PSOs produced similar training and generalisation results, bearing no statistically significant difference between one another under most scenarios (refer to Tables 6.7 and 6.8 for supporting p-values).

Mean E_T and E_G values obtained under scenarios A1 to A5 were also visualised in figure 6.7. Figure 6.7 illustrates that the performance of all algorithms except RBP deteriorated as the spatial severity increased. The dynamic PSOs performed visibly

Table 6.7: Mann-Whitney U p-values obtained for the average training error comparisons on the SEA concepts problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

	BP vs RBP					BP vs RPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.27936	**0.0001**	**0.00120**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.01618**	**0.00069**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	0.292726	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.66527	**0.0001**
D	0.476152	0.230029	0.260089	0.423131	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.065414	**0.01097**
	BP vs CPSO					BP vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.00247**	0.959278	**0.0001**	**0.00051**	**0.0001**	**0.0001**	0.06113	**0.0001**	**0.00122**	**0.0001**
B	**0.0001**	**0.0001**	0.057077	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.02073**	**0.00033**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	0.935996	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.513407	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	0.752762	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.04622**	**0.00764**
	RBP vs RPSO					RBP vs CPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.00212**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	0.10909	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.260089	**0.02247**
D	**0.0001**	**0.0001**	**0.0001**	0.843422	**0.0001**	**0.0001**	**0.0001**	**0.00274**	0.112435	**0.0001**
	RBP vs QPSO					RPSO vs CPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.00019**	**0.00801**	**0.03318**	0.272832	0.146226	**0.01833**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.01424**	**0.02738**	0.423131	0.087794	0.177408
C	**0.0001**	**0.0001**	**0.0001**	0.099517	**0.0001**	0.253866	0.085041	0.072293	0.644024	0.077194
D	**0.0001**	**0.0001**	**0.0001**	0.81482	**0.0001**	**0.02633**	0.146226	**0.00019**	**0.01618**	0.197283
	RPSO vs QPSO					CPSO vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	0.440413	0.272832	0.197283	0.513407	0.335329	**0.00045**	**0.00016**	0.786452	0.299565	0.069938
B	0.970925	0.982572	0.866438	0.532582	0.571977	**0.03318**	0.069938	0.809129	0.213209	0.067645
C	0.935995	0.06878	0.854916	0.628269	0.197283	0.335329	0.970925	**0.0479**	0.449202	**0.00838**
D	**0.01871**	**0.02339**	**0.03713**	0.994216	0.730546	0.741625	0.503955	**0.02633**	**0.01833**	0.126618

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

	BP vs RBP					BP vs RPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.12295	0.051413	**0.00033**	**0.00222**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.03713**	0.449202	0.602392	0.112435	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.172677	0.093511	0.633505	0.172677	1.00578	0.414639
D	**0.0001**	**0.0001**	**0.0001**	0.090617	**0.01365**	**0.03075**	**0.03998**	0.571977	0.069938	0.119364
	BP vs CPSO					BP vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	0.067645	0.602392	**0.00319**	**0.00061**	**0.0001**	0.051413	0.213209	**0.01198**	**0.00015**	**0.0001**
B	**0.00336**	0.513407	0.299565	0.15468	**0.00015**	**0.02247**	0.644024	0.342795	0.675994	**0.0001**
C	0.197283	0.982572	0.241747	0.763945	0.119364	0.866438	0.602392	0.247758	0.485326	0.994216
D	**0.00838**	0.102632	0.335329	0.093511	**0.01551**	0.10909	**0.03195**	**0.02247**	0.235837	**0.00801**
	RBP vs RPSO					RBP vs CPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0096**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.102632
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.00013**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.582035	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.571977
D	**0.0001**	**0.0001**	**0.0001**	**0.00023**	0.373698	**0.0001**	**0.0001**	**0.0001**	**0.00029**	0.831957
	RBP vs QPSO					RPSO vs CPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.02247**	0.66527	0.207802	0.187153	0.644024	0.06113
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.313557	0.935996	0.172677	0.877985	0.947634
C	**0.0001**	**0.0001**	**0.0001**	**0.00014**	0.358039	0.866438	0.741628	0.752762	0.708548	0.342795
D	**0.0001**	**0.0001**	**0.0001**	**0.00634**	0.809129	0.866438	0.719519	0.119364	0.924366	0.562001
	RPSO vs QPSO					CPSO vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	0.602392	0.406247	**0.0479**	0.582035	0.389768	0.959278	0.414639	0.503955	0.230029	0.140112
B	0.843422	0.797771	0.592174	0.27936	0.831957	0.358039	0.820526	0.085041	0.52295	0.654613
C	0.15468	0.959278	0.854916	0.406247	0.612688	0.168039	0.730546	0.901142	0.592174	0.15468
D	0.654613	0.877985	**0.00201**	0.542302	0.602392	0.449202	0.654613	0.197283	0.562001	1.00578

Table 6.9: SEA Concepts Algorithm Ranking for Scenarios A1 to A5

Algorithm	A1		A2		A3		A4		A5		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	2.5	3.5	2.5	1	3	1	1	1	1	2.1	2
RBP	5	5	5	5	5	5	5	5	5	3.5	5	4.7
RPSO	1.5	2.5	1.5	2.5	3	4	3	3	3	3.5	2.4	3.1
CPSO	3	2.5	3.5	2.5	3	1.5	3	3	3	3.5	3.1	2.6
QPSO	1.5	2.5	1.5	2.5	3	1.5	3	3	3	3.5	2.4	2.6

worse than BP under more abrupt scenarios A4 and A5, due to the slower convergence speed of the dynamic PSOs.

As ρ_F reported in Table 6.6 indicates, RBP was least susceptible to overfitting under the considered scenarios ($\rho_F < 1$). RBP was the only algorithm which made no use of previously learned information when a change occurred, and started the search for decision boundaries anew every time the environment changed. Hence, RBP had only 10 iterations under scenarios A1 to A5 to converge on a solution before being completely reinitialised, and 10 iterations was bearly enough to fit the data, thus diminishing the danger of overfitting.

BP obtained the highest ρ_F under most scenarios, as Table 6.6 shows. As opposed to the dynamic PSOs, BP uses a single algorithm starting point (a single set of weights), which makes BP more sensitive to the initial weights after every environment change. The dependence upon initial weights made BP more susceptible to overfitting than the dynamic PSOs, because every time the environment changed, the old weights mapped the previous environment state too closely, and BP had to spend time "unlearning" the old, outdated weight values. The dynamic PSOs, on the other hand, re-evaluated all particles in the swarm after an environment change, promptly discovering a new global best particle which reflected the current environment state more closely.

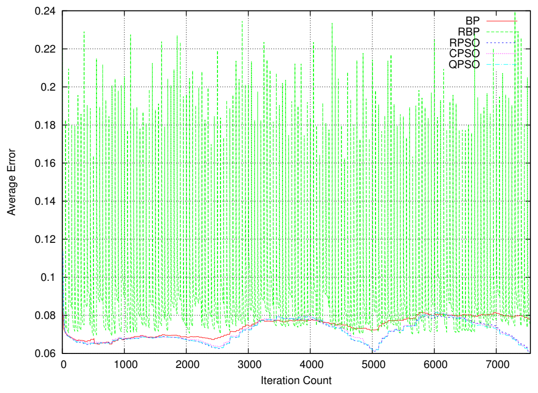
An examination of the standard deviation values reported in Table 6.6 shows that the value of ρ_F produced by BP fluctuated more than that obtained by the dynamic PSOs and RBP. This once again indicates that BP was more sensitive to initial weights than the other algorithms, and showed a less robust performance than the other algorithms considered.

Taking p-values into account, the training algorithms considered were ranked in terms of both average E_T and average E_G. Table 6.9 lists the obtained ranks, together with the average ranks. Average ranks reported in Table 6.9 show that BP produced the lowest average rank in terms of both E_T and E_G, and RBP produced the highest average rank in terms of both E_T and E_G over scenarios A1 to A5. Thus, the overall winner under temporally severe scenarios A1 to A5 is BP.

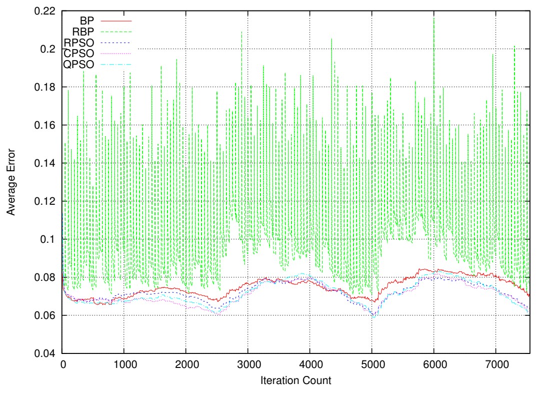
Scenarios B1 to B5: Figures 6.8 and 6.9 illustrate the progression of E_T and E_G over time as obtained by the training algorithms considered under scenarios B1 to B5. Scenarios B1 to B5 simulated less frequent environment changes than scenarios A1 to A5: the sliding window shifted every 50 iterations instead of 10. Figure 6.9 illustrates that the decrease in change frequency had a positive effect on the dynamic PSOs: even under abrupt scenarios B4 and B5, the dynamic PSOs managed to track and adjust the found solution as the environment changed. The performance improvement of the dynamic PSOs is due to the fact that lower frequency allowed the dynamic PSOs to converge around a solution even under abrupt changes.

It becomes evident from figures 6.8 and 6.9 that RBP failed to exploit fruitful areas of the search space as effectively as the other training algorithms. In case of the SEA concepts problem, the windowing approach did not introduce conflicting boundaries, since the concept blocks were mutually exclusive. Thus, previously learned information remained useful even after an environment change, and the algorithms which made use of previously learned information had an advantage over RBP.

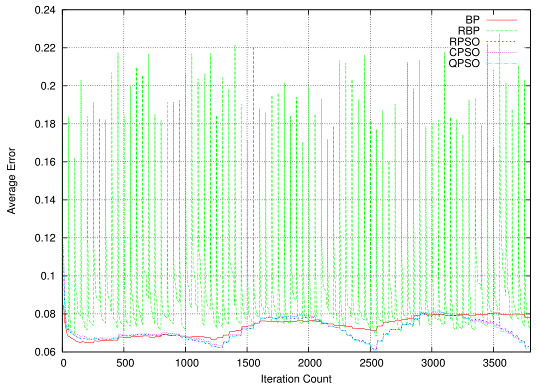
Figure 6.8 also shows that the dynamic PSOs once again exploited better than BP under gradual and semi-gradual scenarios (A1 to A3): The E_T and E_G values produced



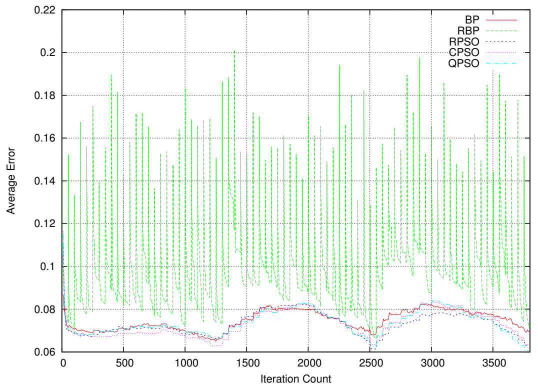
(a) E_T for B1



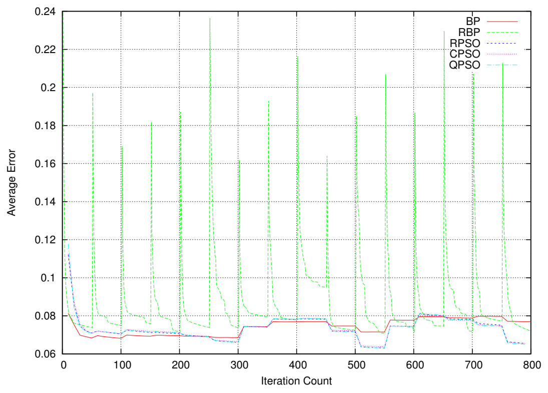
(b) E_G for B1



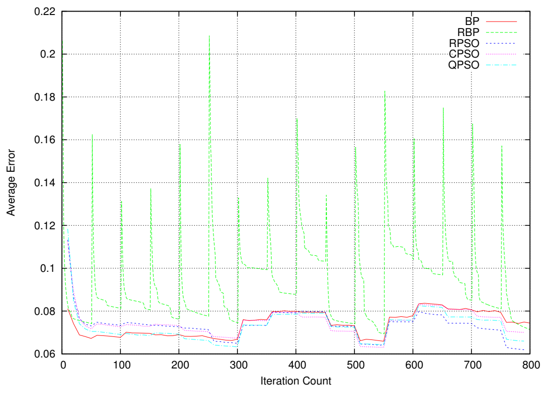
(c) E_T for B2



(d) E_G for B2

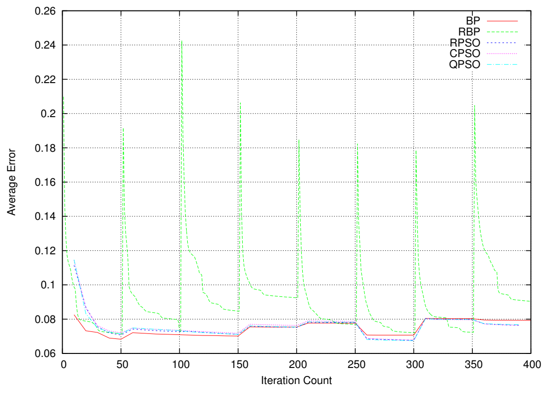


(e) E_T for B3

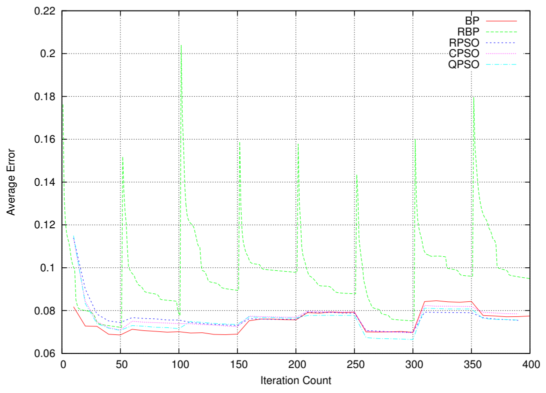


(f) E_G for B3

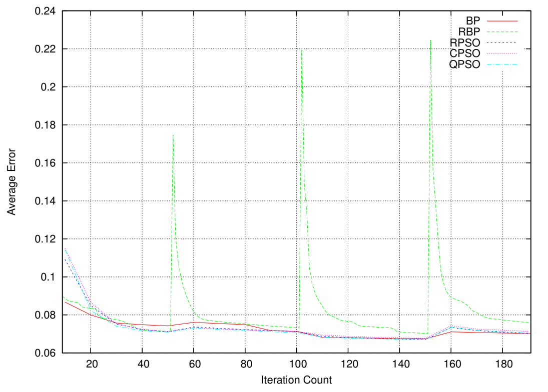
Figure 6.8: Training and Generalisation Error results for SEA concepts, scenarios B1 to B3



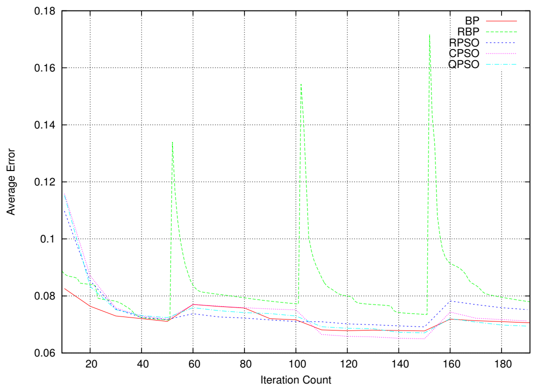
(a) E_T for B4



(b) E_G for B4



(c) E_T for B5

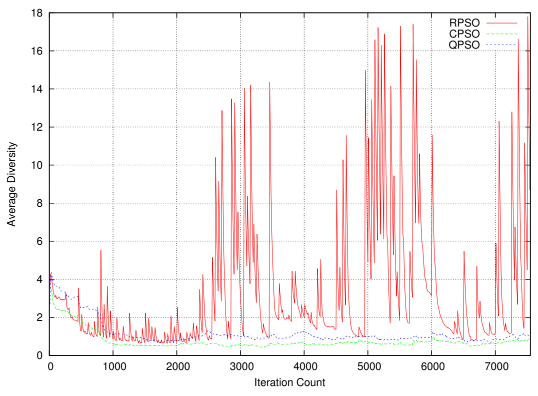


(d) E_G for B5

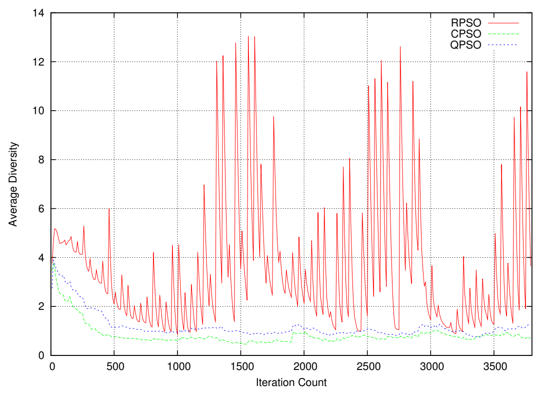
Figure 6.9: Training and Generalisation Error results for SEA concepts, scenarios B4 to B5

by the dynamic PSOs reached lower minimum values than the same errors produced by BP. This indicates that the dynamic PSOs are more apt to track gradually changing decision boundaries than BP. Population-based dynamic PSOs explore a wider area around the current solution than the hill-climbing approach of BP, which helps the dynamic PSOs follow the changing boundaries more closely.

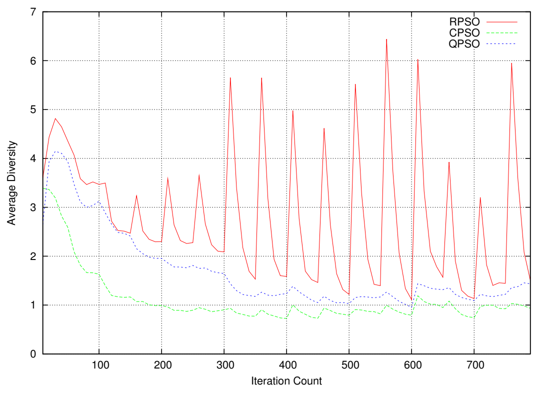
The progression of swarm diversity over time under scenarios B1 to B5 is depicted in figure 6.10. Figure 6.10 illustrates that the CPSO and the QPSO converged to a certain diversity level and maintained that diversity level throughout the algorithm run under scenarios B1 to B4. Under the most abrupt scenario B5 (complete replacement



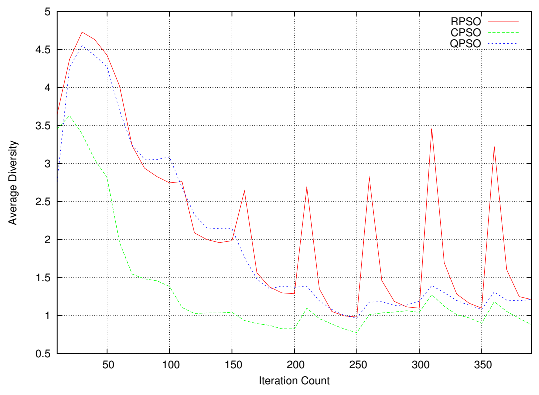
(a) Swarm diversity results for B1



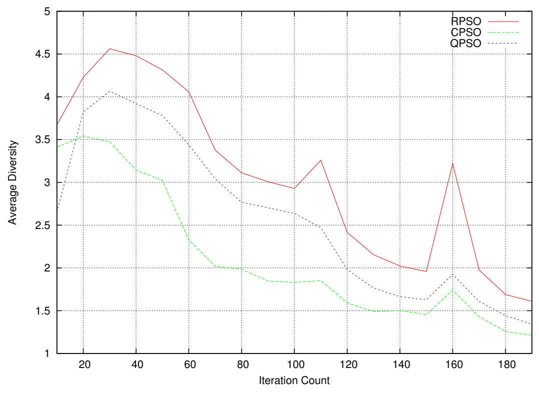
(b) Swarm diversity results for B2



(c) Swarm diversity results for B3



(d) Swarm diversity results for B4



(e) Swarm diversity results for B5

Figure 6.10: Swarm diversity results for SEA concepts, scenarios B1 to B5

of concept) all three dynamic PSOs failed to converge to a stable diversity level due to the slow convergence speed of these algorithms. The diversity of the RPSO was rather unstable under gradual scenarios B1 and B2, thus showing that a reinitialisation of 25% of particles under gradual scenarios discarded too much of useful previously learned information.

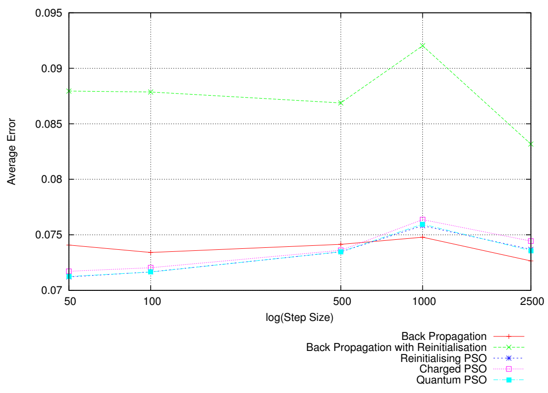
Table 6.10 summarises the mean E_T and E_G, ρ_F, and average swarm diversity values obtained by the algorithms for scenarios B1 to B5. Generalisation factor values reported in Table 6.10 show that all algorithms exhibited minor to no overfitting under scenarios B1 to B5, except for RBP, which produced the highest ρ_F under gradual to semi-gradual scenarios B1 to B3. As it was already discussed, the lack of memory in RBP made this algorithm inferior to the memory-preserving algorithms. Poor generalisation performance of RBP under gradual scenarios can be due to bias towards one of the two decision boundaries represented inside the sliding window, since, due to gradual insertion of a new concept into the window, one decision boundary was often represented by a larger number of training patterns than the other boundary.

Average swarm diversity values listed in Table 6.10 show that the average diversity of the CPSO and the QPSO increased with an increase of spatial severity, because abrupt changes required the dynamic PSOs to explore wider. The CPSO consistently obtained the smallest average diversity: the CPSO was the only dynamic PSO algorithm out of the three dynamic PSOs considered which did not reinitialise a percentage of particles, thus the CPSO preserved most memory and converged to a smaller area around a solution. The RPSO maintained the largest average diversity throughout scenarios B1 to B5 due to its weak memory of previously found solutions.

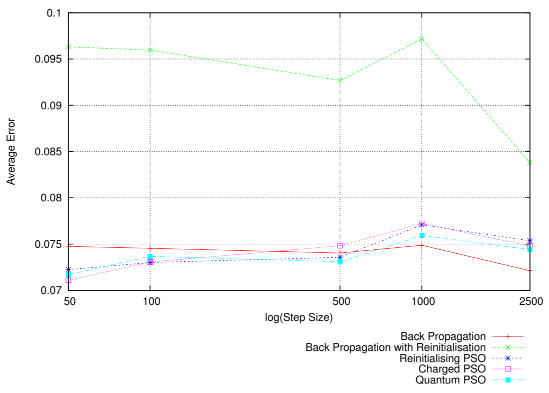
Figure 6.11 shows average training and generalisation error results obtained by the five algorithms under scenarios B1 to B5. Figure 6.11 shows that algorithm fitness decreased as the abruptness of change increased from B1 to B4, indicating that abrupt changes were harder to adapt to than gradual changes, since more drastic changes had to

Table 6.10: SEA Concepts Results for Scenarios B1 to B5

Scenario / Algorithm	**B1** *(frequency: 50, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.074078 ± 0.001688	0.074755 ± 0.005007	1.02208 ± 0.081964	n/a
Reinit. Back Propagation	0.087951 ± 0.00329	0.096313 ± 0.00589	1.05347 ± 0.071199	n/a
Reinitialising PSO	0.071208 ± 0.000677	0.072236 ± 0.005498	1.02964 ± 0.101128	3.27537 ± 1.87885
Charged PSO	0.071717 ± 0.000875	0.071091 ± 0.004605	0.999089 ± 0.078706	0.770277 ± 0.258123
Quantum PSO	0.071246 ± 0.000872	0.071703 ± 0.004318	1.00751 ± 0.09248	1.16778 ± 0.523124
	B2 *(frequency: 50, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.073412 ± 0.001436	0.074548 ± 0.005521	1.0423 ± 0.105881	n/a
Reinit. Back Propagation	0.087863 ± 0.003877	0.095976 ± 0.005494	1.06418 ± 0.064687	n/a
Reinitialising PSO	0.071659 ± 0.00059	0.073009 ± 0.005974	1.02815 ± 0.099551	3.53494 ± 1.55104
Charged PSO	0.07204 ± 0.000625	0.073066 ± 0.00437	1.00374 ± 0.086135	0.844318 ± 0.218703
Quantum PSO	0.071649 ± 0.000882	0.073669 ± 0.005507	1.03363 ± 0.104276	1.18129 ± 0.398826
	B3 *(frequency: 50, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.07414 ± 0.00119	0.074036 ± 0.003998	0.996259 ± 0.062547	n/a
Reinit. Back Propagation	0.086878 ± 0.007596	0.092688 ± 0.0074	1.03577 ± 0.072633	n/a
Reinitialising PSO	0.07347 ± 0.000578	0.073569 ± 0.003802	1.0193 ± 0.072363	2.77745 ± 1.22533
Charged PSO	0.073609 ± 0.000684	0.074822 ± 0.003874	1.01493 ± 0.070342	1.11491 ± 0.262838
Quantum PSO	0.073475 ± 0.000735	0.073081 ± 0.00474	0.986286 ± 0.057898	1.70271 ± 0.684032
	B4 *(frequency: 50, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.074784 ± 0.001933	0.074867 ± 0.005506	0.997821 ± 0.102796	n/a
Reinit. Back Propagation	0.09203 ± 0.015599	0.097183 ± 0.015191	1.02118 ± 0.063948	n/a
Reinitialising PSO	0.075825 ± 0.000901	0.077073 ± 0.005139	1.02841 ± 0.08129	2.23834 ± 0.637953
Charged PSO	0.076371 ± 0.001175	0.077229 ± 0.005933	1.00032 ± 0.096202	1.35719 ± 0.296521
Quantum PSO	0.075958 ± 0.000959	0.075926 ± 0.005911	0.999026 ± 0.079358	1.99361 ± 0.780317
	B5 *(frequency: 50, step size: 2500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.072648 ± 0.016249	0.072121 ± 0.017849	0.974087 ± 0.058723	n/a
Reinit. Back Propagation	0.083178 ± 0.008553	0.083807 ± 0.009732	0.991642 ± 0.073368	n/a
Reinitialising PSO	0.073692 ± 0.001728	0.075357 ± 0.005735	1.00928 ± 0.073479	3.05492 ± 1.24666
Charged PSO	0.074436 ± 0.001995	0.074777 ± 0.005022	1.0121 ± 0.084311	2.11189 ± 1.26007
Quantum PSO	0.073581 ± 0.001283	0.074403 ± 0.004463	1.01723 ± 0.064235	2.56099 ± 1.27871



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.11: Average Error results for SEA concepts, scenarios B1 to B5

Table 6.11: SEA Concepts Algorithm Ranking for Scenarios B1 to B5

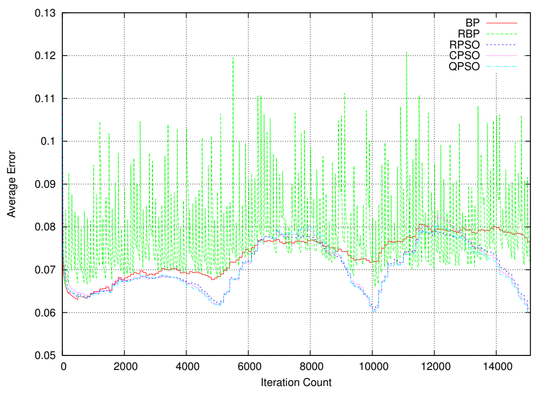
Algorithm	B1		B2		B3		B4		B5		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	4	4	2.5	2.5	2.5	1	2.5	1	1	2.5	2.5
RBP	5	5	5	5	5	5	5	5	5	5	5	5
RPSO	1.5	2	1.5	2.5	2.5	2.5	3	2.5	3	3	2.3	2.5
CPSO	3	2	3	2.5	2.5	2.5	3	2.5	3	3	2.9	2.5
QPSO	1.5	2	1.5	2.5	2.5	2.5	3	2.5	3	3	2.3	2.5

be made to the current solution. However, the performance of all algorithms considered improved under the most abrupt scenario B5. Performance improvement under scenario B5 was due to the fact that, under this scenario, an environment change resulted in a complete replacement of concept inside the sliding window. Thus, the training algorithms were always presented with a single decision boundary only, as opposed to gradual and semi-gradual scenarios, where the sliding window most of the time contained patterns from two bordering concept blocks.

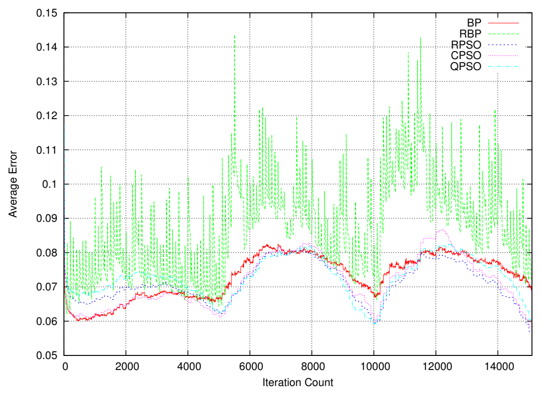
The algorithms were ranked in terms of average E_T and E_G, taking the p-values reported in tables 6.7 and 6.8 into account. Algorithm ranks together with average ranks are listed in Table 6.11.

As becomes evident from Table 6.11, RBP showed the worst performance under scenarios B1 to B5. Such poor performance is due to the fact that RBP keeps no memory of the previously learned information when the environment changes, and previously learned information was useful for the SEA concepts problem. BP was significantly outperformed by the dynamic PSOs under the gradual scenarios B1 and B2, and BP significantly outperformed all the other algorithms under the abrupt scenarios B4 and B5. No statistically significant difference between BP and the dynamic PSOs was observed under scenario B3. The superior performance of the dynamic PSOs under gradual changes, and the superior performance of BP under abrupt changes indicate that the dynamic PSOs were more apt at tracking gradual changes than BP on the SEA concepts problem, due to better exploration ability of the population-based approach. BP, on the other hand, adjusted to abrupt changes faster due to faster convergence of the hill-climbing approach. No statistically significant difference was observed between the three dynamic PSOs under scenarios B1 to B5, except that the CPSO was outperformed by the RPSO and the QPSO in terms of E_T under gradual scenarios B1 and B2. The CPSO, as confirmed by the average diversity values in Table 6.10, explored less than the other two dynamic PSOs; thus, the CPSO obtained higher training errors. The average ranks in Table 6.11 confirm the inferiority of RBP under scenarios B1 to B5, and show that no overall winner among the other algorithms can be pointed out, since the average rank in terms of E_G is equal for BP, RPSO, CPSO, and QPSO. Considering E_T only, the QPSO and the RPSO performed superior to the other algorithms considered.

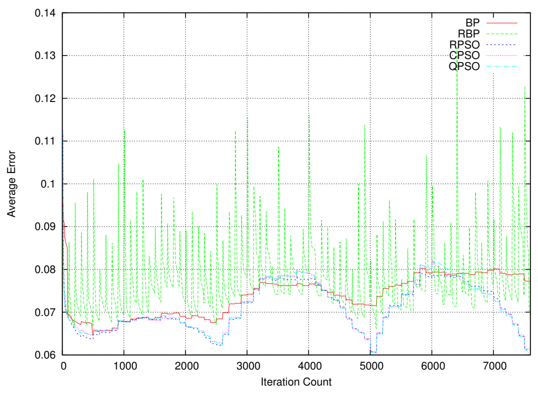
Scenarios C1 to C5: Figures 6.12 and 6.13 illustrate the progression of E_T and E_G over time as obtained by the various algorithms considered. Similar traits as were observed under scenarios B1 to B5 are observed under scenarios C1 to C5: the dynamic PSOs take longer than BP and RBP to converge around a solution in the beginning of the algorithm run, and track gradual changes better than both BP and RBP.



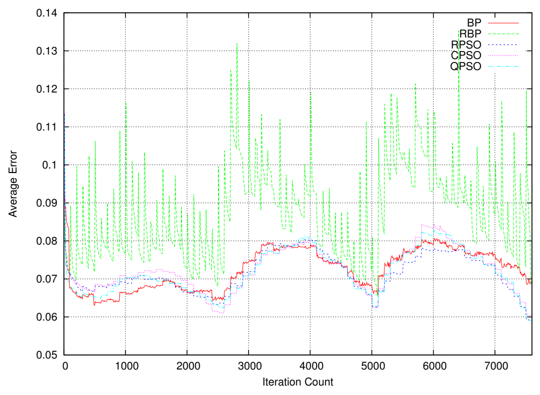
(a) E_T for C1



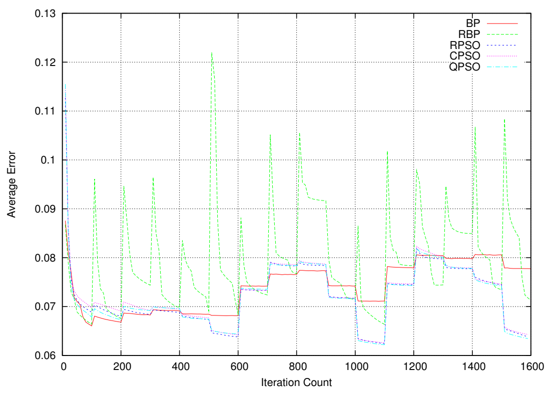
(b) E_G for C1



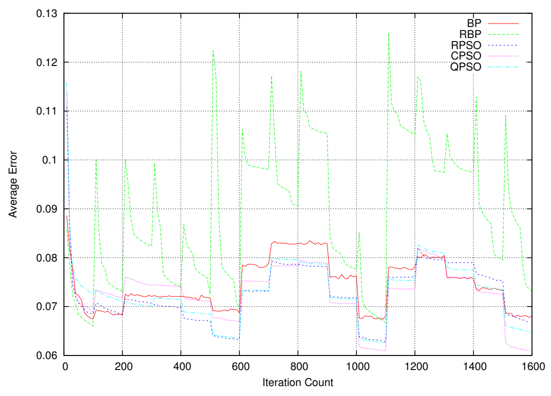
(c) E_T for C2



(d) E_G for C2

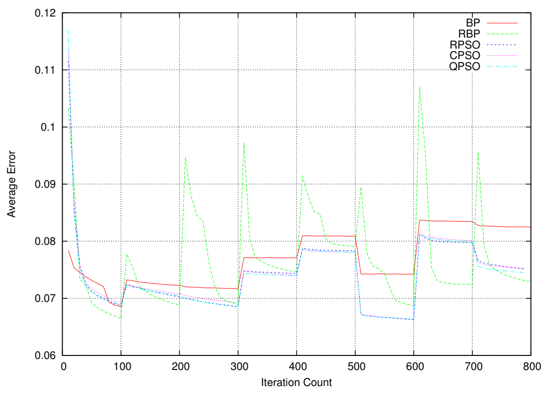


(e) E_T for C3

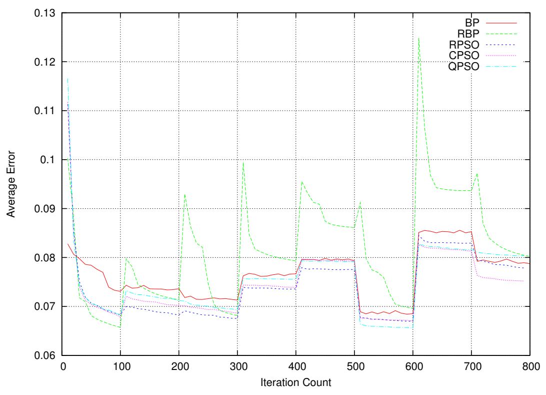


(f) E_G for C3

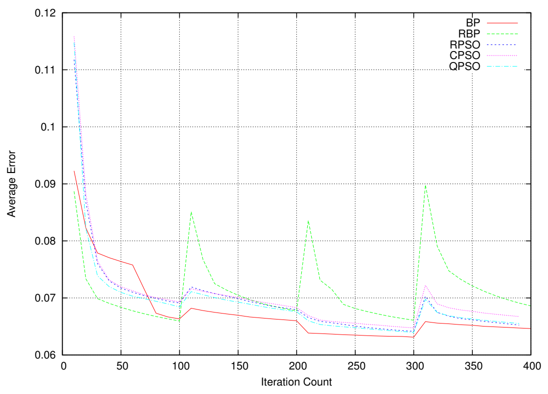
Figure 6.12: Training and Generalisation Error results for SEA concepts, scenarios C1 to C3



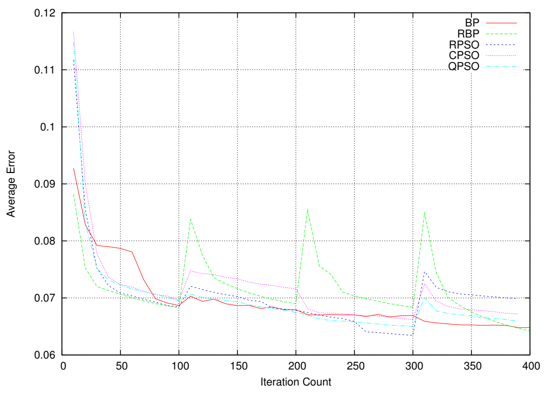
(a) E_T for C4



(b) E_G for C4



(c) E_T for C5



(d) E_G for C5

Figure 6.13: Training and Generalisation Error results for SEA concepts, scenarios C4 to C5

Figures 6.12 and 6.13 illustrate that RBP did not generalise well under scenarios C1 to C4. However, the generalisation performance of RBP improved under the most abrupt scenario C5. Scenario C5 was the only scenario that completely replaced the concept inside the sliding window whenever the environment changed. Scenarios C1 to C4, on the other hand, gradually added a new concept to the old concept, thus most of the time the sliding window contained two neighbouring concepts, or, in other words, two decision boundaries instead of one. Poor generalisation performance of RBP under scenarios C1 to C4 is due to RBP approximating one decision boundary better than the second one, since, due to the gradual insertion of a new concept into the window, one decision boundary was often represented by a larger number of patterns than the other boundary.

Figure 6.14 illustrates the progression of swarm diversity over time. While the CPSO and the QPSO converged to a certain diversity level and maintained it throughout the algorithm run, the RPSO once again exhibited divergent behaviour under gradual scenarios C1 and C2. As it was already discussed, weak reliance on previously learned information makes the RPSO susceptible to divergent behaviour under scenarios where little new information is added at every environment change.

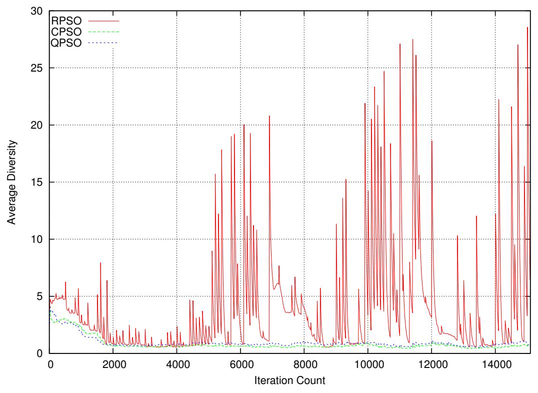
Table 6.12 presents average E_T and E_G values, as well as average ρ_F obtained by the training algorithms under scenarios C1 to C5. Average swarm diversity is also reported where applicable. Generalisation factor values reported in Table 6.12 confirm that RBP was most prone to overfitting under scenarios C1 to C3. The other algorithms considered showed minor to no overfitting. Table 6.12 also shows that out the three dynamic PSOs, the CPSO produced the lowest average diversity. The CPSO depends on swarm memory stronger than both the RPSO and the QPSO, since the CPSO never reinitialises particles. Lack of reinitialisation allows the CPSO to converge to a smaller area around a solution.

Figure 6.15 illustrates average E_T and E_G values obtained by the algorithms under scenarios C1 to C5. Figure 6.15 shows that the fitness of BP and the dynamic PSOs decreased as the abruptness of change increased from C1 to C4, since abrupt changes were harder to adapt to than gradual changes. The performance of all considered algorithms improved under the most abrupt scenarios C5, because, as already explained earlier in this section, C5 was the only scenario that presented the algorithms with a single decision boundary at any one time. The performance of RBP started to improve under abrupt scenario C4, since complete reinitialisation of weights is most effective under abrupt scenarios where the preservation of previously learned information is less important.

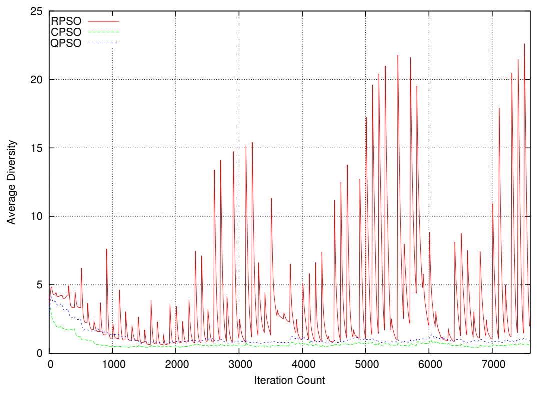
The algorithms were ranked based on the average E_T and E_G values as reported in Table 6.12, taking p-values listed in Tables 6.7 and 6.8 into account. Algorithm ranks,

Table 6.12: SEA Concepts Results for Scenarios C1 to C5

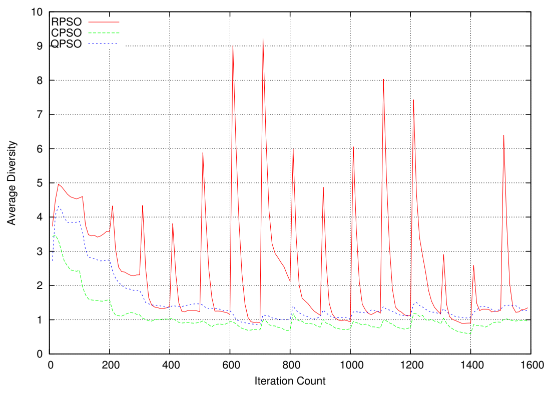
Scenario / Algorithm	**C1** *(frequency: 100, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.073506 ± 0.001122	0.072902 ± 0.003983	0.982364 ± 0.07719	n/a
Reinit. Back Propagation	0.077631 ± 0.002366	0.086025 ± 0.005988	1.04864 ± 0.090041	n/a
Reinitialising PSO	0.070671 ± 0.000646	0.071146 ± 0.004587	1.02146 ± 0.096665	3.81076 ± 2.44116
Charged PSO	0.07088 ± 0.000762	0.071306 ± 0.004753	0.987295 ± 0.100106	0.815575 ± 0.412588
Quantum PSO	0.070579 ± 0.00085	0.072789 ± 0.005814	1.04779 ± 0.101841	0.947459 ± 0.363938
	C2 *(frequency: 100, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.073749 ± 0.001783	0.072401 ± 0.005544	0.984114 ± 0.107786	n/a
Reinit. Back Propagation	0.077723 ± 0.002391	0.086938 ± 0.005286	1.09037 ± 0.10604	n/a
Reinitialising PSO	0.070901 ± 0.000572	0.071495 ± 0.005306	1.02443 ± 0.107218	3.64572 ± 1.52371
Charged PSO	0.071195 ± 0.000732	0.072334 ± 0.004217	1.02423 ± 0.083891	0.677161 ± 0.188148
Quantum PSO	0.071195 ± 0.000575	0.071782 ± 0.004171	1.00872 ± 0.069144	1.1062 ± 0.566274
	C3 *(frequency: 100, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.074033 ± 0.001444	0.074472 ± 0.004278	1.0306 ± 0.082937	n/a
Reinit. Back Propagation	0.079415 ± 0.006442	0.088763 ± 0.010017	1.0812 ± 0.096053	n/a
Reinitialising PSO	0.072101 ± 0.000826	0.072631 ± 0.005729	1.00469 ± 0.112067	2.43703 ± 0.720238
Charged PSO	0.072522 ± 0.00072	0.072996 ± 0.004072	1.02267 ± 0.074461	1.064 ± 0.255574
Quantum PSO	0.072109 ± 0.000685	0.072983 ± 0.004936	1.02516 ± 0.06781	1.54548 ± 0.691352
	C4 *(frequency: 100, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.076934 ± 0.014302	0.0765 ± 0.01565	1.02344 ± 0.08496	n/a
Reinit. Back Propagation	0.07673 ± 0.005086	0.081602 ± 0.007463	1.01743 ± 0.092138	n/a
Reinitialising PSO	0.074013 ± 0.000879	0.074156 ± 0.00472	0.99061 ± 0.08843	2.54484 ± 1.73896
Charged PSO	0.074196 ± 0.001033	0.074275 ± 0.004435	0.993286 ± 0.078749	1.21395 ± 0.287803
Quantum PSO	0.073903 ± 0.000713	0.07515 ± 0.004517	1.00616 ± 0.080637	1.98942 ± 1.02498
	C5 *(frequency: 100, step size: 2500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.067721 ± 0.004688	0.069545 ± 0.004448	1.03044 ± 0.089939	n/a
Reinit. Back Propagation	0.071498 ± 0.001767	0.071627 ± 0.005087	1.01818 ± 0.097463	n/a
Reinitialising PSO	0.069651 ± 0.001247	0.070552 ± 0.004322	0.998287 ± 0.070525	2.74491 ± 1.10069
Charged PSO	0.070355 ± 0.001678	0.071822 ± 0.00472	1.02021 ± 0.083689	1.60652 ± 0.949506
Quantum PSO	0.069229 ± 0.001408	0.06982 ± 0.004534	1.00949 ± 0.07159	2.5932 ± 1.89665



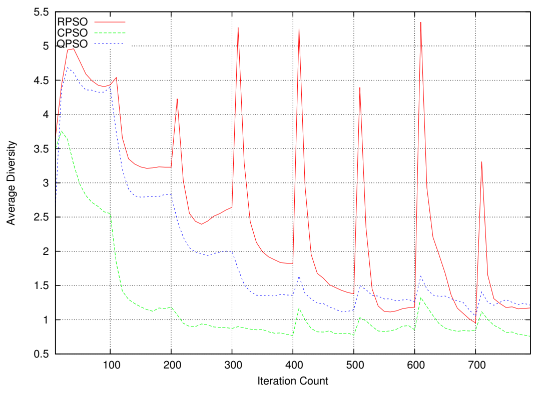
(a) Swarm diversity results for C1



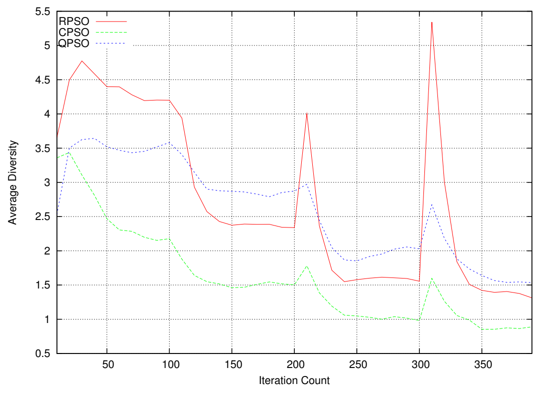
(b) Swarm diversity results for C2



(c) Swarm diversity results for C3

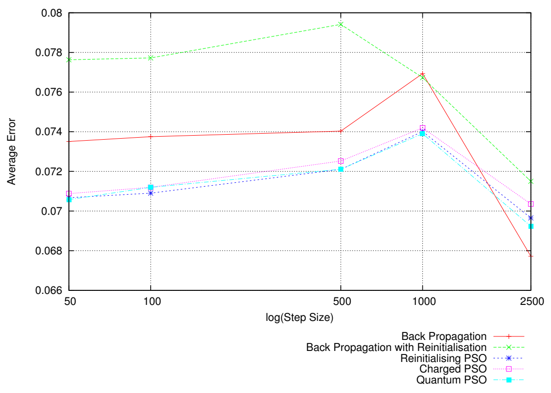


(d) Swarm diversity results for C4

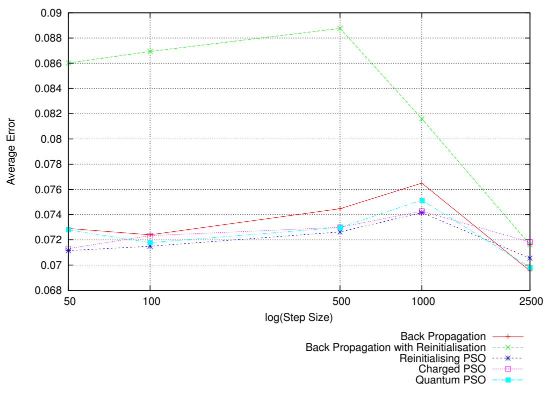


(e) Swarm diversity results for C5

Figure 6.14: Swarm diversity results for SEA concepts, scenarios C1 to C5



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.15: Average Error results for SEA concepts, scenarios C1 to C5

Table 6.13: SEA Concepts Algorithm Ranking for Scenarios C1 to C5

Algorithm	C1		C2		C3		C4		C5		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	2.5	4	2.5	4	2.5	3	2.5	1	3	3.2	2.6
RBP	5	5	5	5	5	5	3	5	5	3	4.6	4.6
RPSO	2	2.5	2	2.5	1.5	2.5	3	2.5	3	3	2.3	2.6
CPSO	2	2.5	2	2.5	3	2.5	3	2.5	3	3	2.6	2.6
QPSO	2	2.5	2	2.5	1.5	2.5	3	2.5	3	3	2.3	2.6

including average rank, are reported in Table 6.13. Table 6.13 shows that the dynamic PSO algorithms performed insignificantly different from one another in terms of E_T under all scenarios except C3, where the CPSO was significantly outperformed by the QPSO and the RPSO. As already pointed out, the CPSO, due to its dependence on swarm memory, explores less than the QPSO and the CPSO, which proved inefficient under semi-abrupt scenario C3. However, no statistically significant difference was observed between the different dynamic PSOs in terms of E_G. Thus, the CPSO compensated inferior training performance by good generalisation performance. RBP was significantly outperformed by all other algorithms under scenarios C1 to C3, proving that weight reinitialisation is not an effective approach under gradual scenarios. However, RBP slightly recovered under more abrupt scenarios C4 and C5, competing with the dynamic PSOs and BP in terms of either E_T or E_G. BP was significantly outperformed by the dynamic PSOs in terms of E_T under scenarios C1 to C3, and significantly outperformed the dynamic PSOs under the abrupt scenario C5. However, no statistically significant difference in terms of E_G was observed between BP and the dynamic PSOs throughout C1 to C5. Thus, no top performer can be pointed out.

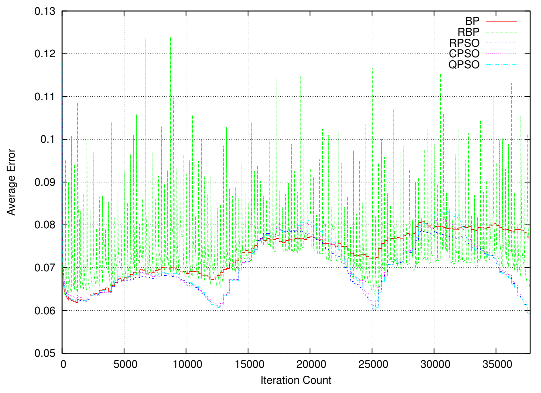
Scenarios D1 to D5: Figures 6.16 and 6.17 show the progression of E_T and E_G produced by the different algorithms over time under scenarios D1 to D5. Scenarios D1 to D5 simulated environments of low temporal severity: an environment change happened every 250 iterations. Thus, the dynamic PSOs and RBP were given enough time to find a solution before an environment change even in abruptly changing environments, and any fault to do so can be attributed to the training algorithms instead of the temporal severity property of the dynamic scenarios considered. The progression of diversity over time, as illustrated in figure 6.18, shows that the three dynamic PSOs have indeed converged to a stable diversity level under scenarios D1 to D5, with the exception of the RPSO, which once again showed divergent behaviour under the gradual scenarios D1 and D2. Figures 6.16 and 6.17 show that the algorithms exhibited similar trends as observed under scenarios C1 to C5.

Table 6.14 reports average error values used for statistical comparison of the algorithms, and the corresponding generalisation factor values. The obtained values of ρ_F show that RBP was most prone to overfitting compared to the other algorithms considered under gradual to semi-gradual scenarios (D1 to D3). As already explained, such behaviour of RBP is due to RBP producing solutions biased towards the decision boundary represented by a larger number of training patterns inside the sliding window.

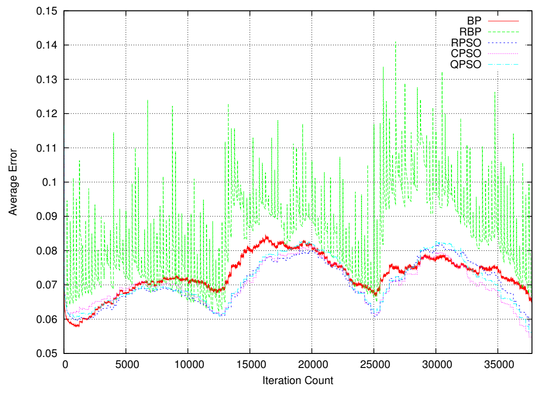
The small average diversity values reported in Table 6.14 confirm that the dynamic PSOs had enough time to exploit the found solutions.

Table 6.14: SEA Concepts Results for Scenarios D1 to D5

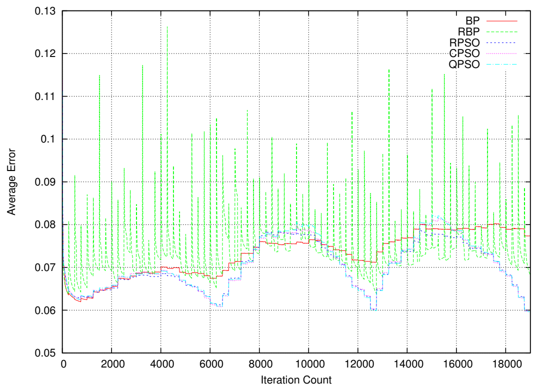
Scenario / Algorithm	**D1** *(frequency: 250, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.073396 ± 0.001278	0.072911 ± 0.00406	1.00329 ± 0.08632	n/a
Reinit. Back Propagation	0.073543 ± 0.001375	0.083022 ± 0.004974	1.07181 ± 0.072254	n/a
Reinitialising PSO	0.070056 ± 0.000934	0.070433 ± 0.004267	0.994687 ± 0.082058	5.23668 ± 3.47069
Charged PSO	0.070774 ± 0.001174	0.070202 ± 0.003386	1.00667 ± 0.090573	0.572216 ± 0.298635
Quantum PSO	0.070851 ± 0.001382	0.07095 ± 0.005426	1.00108 ± 0.099945	0.946202 ± 0.549186
	D2 *(frequency: 250, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.07305 ± 0.001589	0.073786 ± 0.004696	1.01535 ± 0.082793	n/a
Reinit. Back Propagation	0.073585 ± 0.00157	0.082509 ± 0.006086	1.08168 ± 0.115276	n/a
Reinitialising PSO	0.070082 ± 0.000753	0.071375 ± 0.004665	1.00878 ± 0.085361	3.97962 ± 2.72183
Charged PSO	0.070429 ± 0.000892	0.07152 ± 0.004626	1.01599 ± 0.110171	0.579066 ± 0.26082
Quantum PSO	0.07056 ± 0.000799	0.07096 ± 0.00482	1.00341 ± 0.086559	1.02523 ± 0.532655
	D3 *(frequency: 250, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.073626 ± 0.001459	0.073364 ± 0.005627	0.993279 ± 0.072443	n/a
Reinit. Back Propagation	0.074375 ± 0.005637	0.08254 ± 0.007563	1.07956 ± 0.085989	n/a
Reinitialising PSO	0.070449 ± 0.000807	0.073815 ± 0.005382	1.04313 ± 0.092578	2.49692 ± 1.03507
Charged PSO	0.071185 ± 0.000702	0.071683 ± 0.004437	1.01557 ± 0.078838	0.91497 ± 0.531918
Quantum PSO	0.070834 ± 0.000589	0.070073 ± 0.003652	0.972879 ± 0.083707	1.33522 ± 0.584105
	D4 *(frequency: 250, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.072973 ± 0.001673	0.075229 ± 0.005077	1.05637 ± 0.101737	n/a
Reinit. Back Propagation	0.073241 ± 0.002844	0.077941 ± 0.005381	1.03606 ± 0.069982	n/a
Reinitialising PSO	0.072299 ± 0.000698	0.072563 ± 0.004719	1.01203 ± 0.095537	2.25533 ± 0.933986
Charged PSO	0.072747 ± 0.000734	0.07269 ± 0.004979	1.00733 ± 0.104694	0.95723 ± 0.409318
Quantum PSO	0.072287 ± 0.000805	0.073863 ± 0.005888	1.01618 ± 0.094414	1.60519 ± 0.900659
	D5 *(frequency: 250, step size: 2500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.065106 ± 0.004135	0.064206 ± 0.005316	0.988269 ± 0.08392	n/a
Reinit. Back Propagation	0.071554 ± 0.010663	0.06982 ± 0.013026	0.955725 ± 0.097808	n/a
Reinitialising PSO	0.065198 ± 0.001192	0.066019 ± 0.006596	1.01572 ± 0.129133	3.38417 ± 2.38783
Charged PSO	0.065742 ± 0.001147	0.067239 ± 0.005103	1.02713 ± 0.095238	1.43895 ± 0.758732
Quantum PSO	0.065092 ± 0.000992	0.066777 ± 0.004757	1.02698 ± 0.106289	2.76199 ± 2.56782



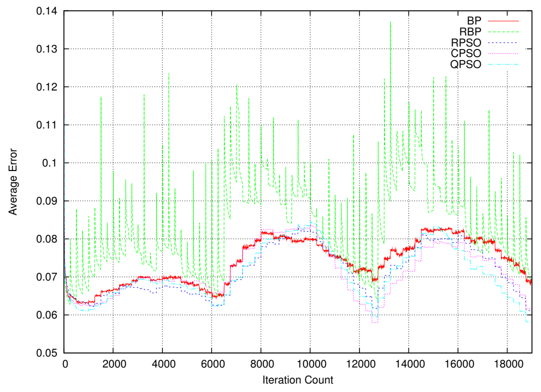
(a) E_T for D1



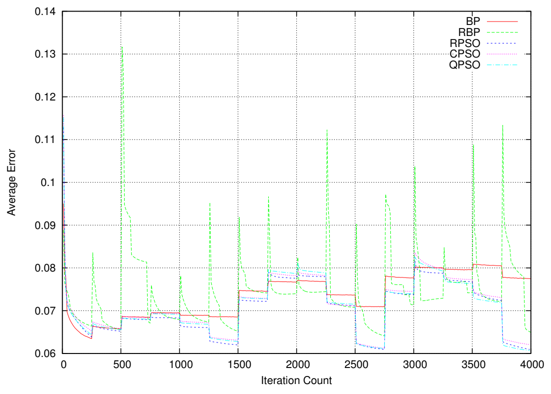
(b) E_G for D1



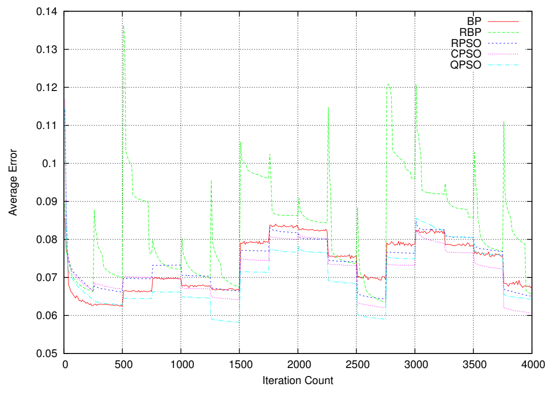
(c) E_T for D2



(d) E_G for D2

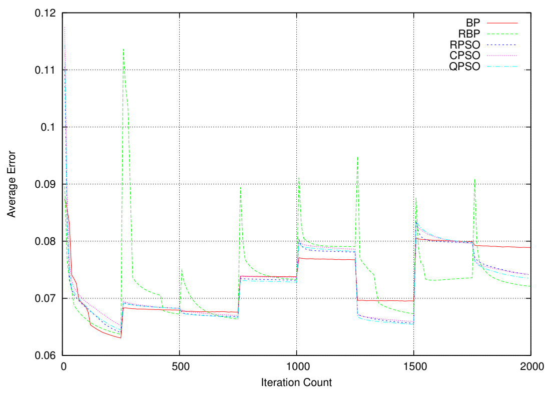


(e) E_T for D3

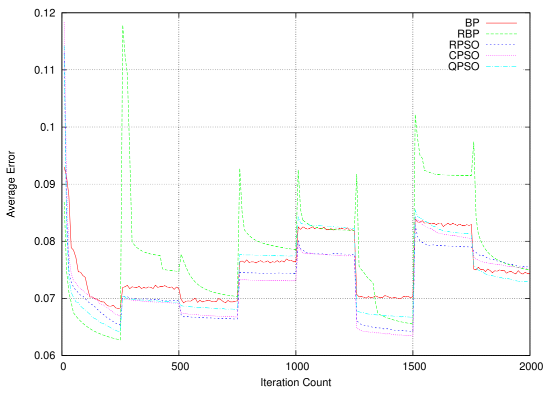


(f) E_G for D3

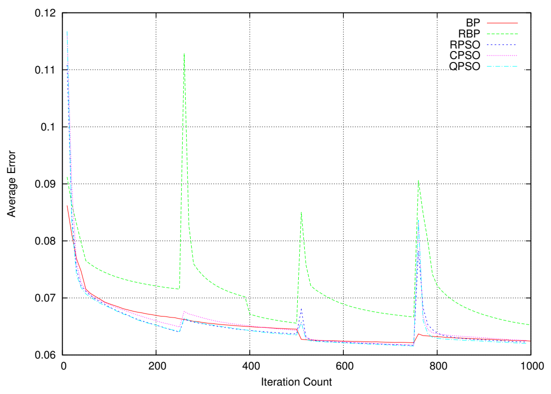
Figure 6.16: Training and Generalisation Error results for SEA concepts, scenarios D1 to D3



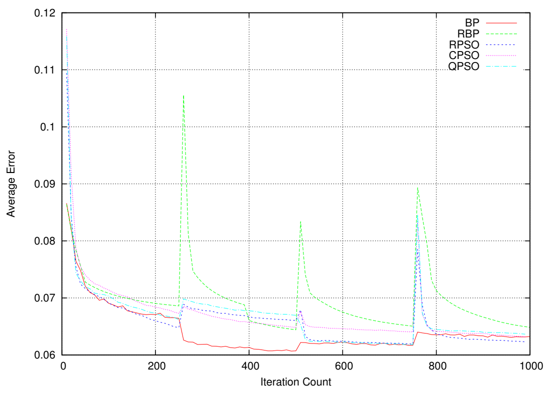
(a) E_T for D4



(b) E_G for D4



(c) E_T for D5



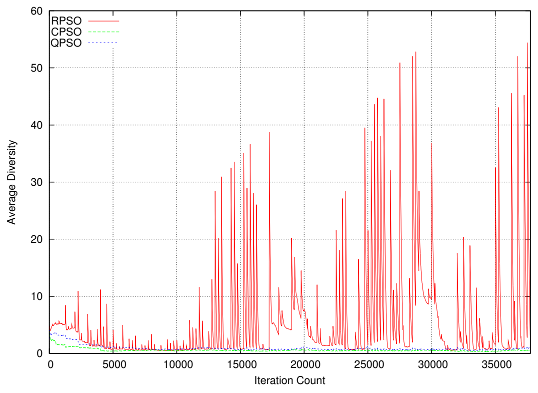
(d) E_G for D5

Figure 6.17: Training and Generalisation Error results for SEA concepts, scenarios D4 to D5

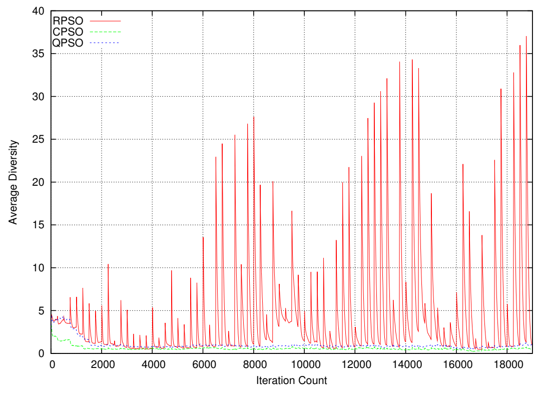
Table 6.15: SEA Concepts Algorithm Ranking for Scenarios D1 to D5

Algorithm	D1		D2		D3		D4		D5		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4.5	4	4.5	4	4.5	2.5	4	3	2	3	3.9	3.3
RBP	4.5	5	4.5	5	4.5	5	4	3	5	3	4.5	4.2
RPSO	1	2	2	2	1	2.5	1.5	3	3.5	3	1.8	2.5
CPSO	2.5	2	2	2	3	2.5	4	3	3.5	3	3	2.5
QPSO	2.5	2	2	2	2	2.5	1.5	3	1	3	1.8	2.5

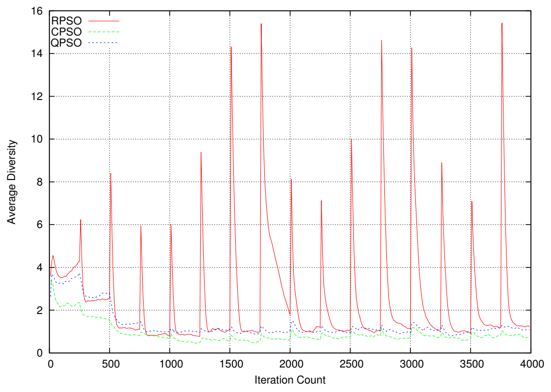
The algorithms were ranked in terms of E_T and E_G, taking the p-values reported in Tables 6.7 and 6.8 into account. The obtained ranks, together with average ranks, are



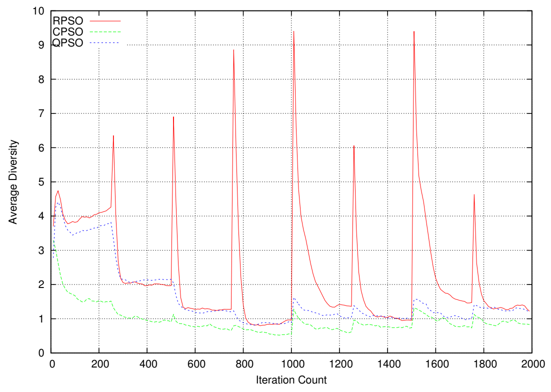
(a) Swarm diversity results for D1



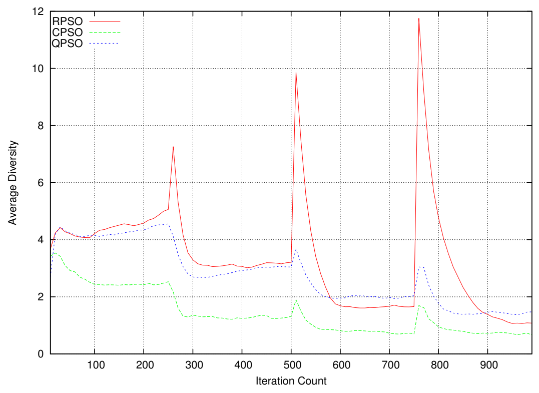
(b) Swarm diversity results for D2



(c) Swarm diversity results for D3



(d) Swarm diversity results for D4



(e) Swarm diversity results for D5

Figure 6.18: Swarm diversity results for SEA concepts, scenarios D1 to D5

listed in Table 6.15. Table 6.15 shows that all three dynamic PSOs, given enough time to converge, managed to significantly outperform both BP and RBP under gradual scenarios D1 and D2. The dynamic PSOs also outperformed both BP and RBP in terms of E_T under scenarios D3 and D4. However, no statistical difference was observed between the corresponding values of E_G. Thus, BP caught up with the dynamic PSOs under semi-abrupt scenarios in terms of generalisation. Table 6.15 also shows that the CPSO was outperformed by the other two dynamic PSOs in terms of training, but generalised equally well.

The average ranks reported in Table 6.7 show that the three dynamic PSOs outperformed both BP and RBP under scenarios D1 to D5. Taking E_T into account, the QPSO and the RPSO can be considered as the top performers.

All scenarios: The average training errors produced by the five algorithms under the 20 different dynamic scenarios are illustrated in Figure 6.19, and the average generalisation errors are illustrated in Figure 6.20.

Figure 6.19 illustrates that RBP performed inferior to all other algorithms in terms of E_T under most scenarios considered. The least temporally severe scenarios D1 to D4 were the only exception: the training performance of RBP did not differ from the training performance of BP under D1 to D4, since RBP had enough time to converge on a similarly good solution as BP. However, Figure 6.20 shows that the low E_T values obtained by RBP under D1 to D4 were not supported by correspondingly low E_G values, thus RBP was prone to overfitting. Figures 6.19 and 6.20 illustrate that the dynamic PSOs outperformed BP and RBP under gradual scenarios of low temporal severity in terms of both E_T and E_G, and were outperformed by BP under abrupt scenarios of high temporal severity. The dynamic PSOs tracked gradual changes better than BP due to the population-based principle which facilitates exploration. However, the same population-based principle made the dynamic PSOs converge slower than BP, which

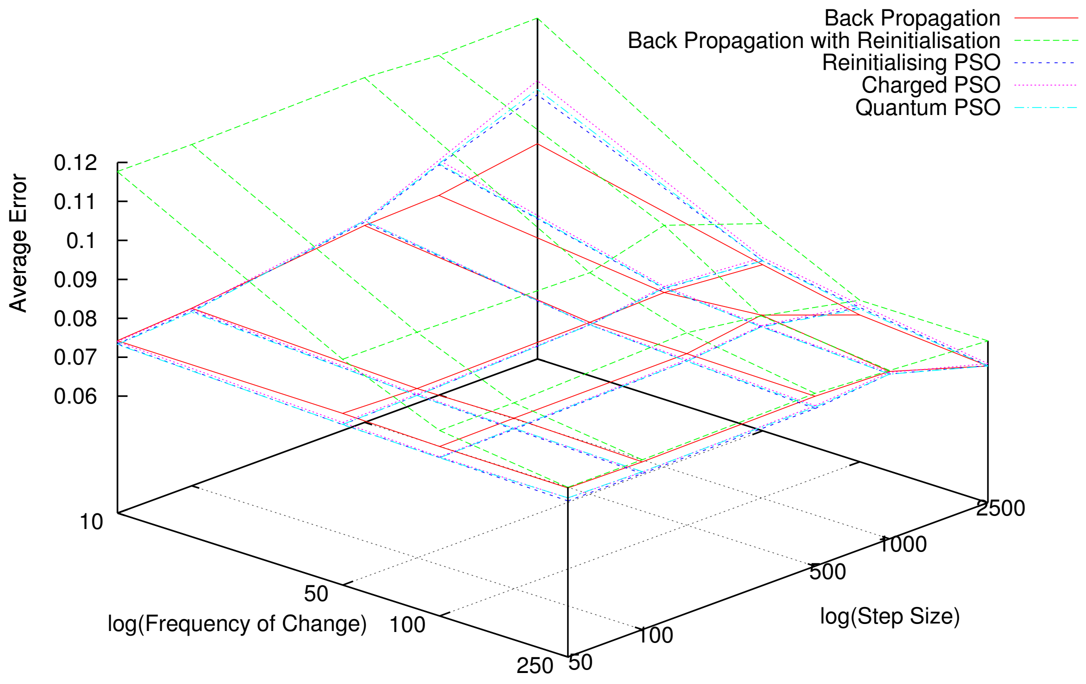


Figure 6.19: Average Training Error Results for SEA Concepts

made BP more effective than the dynamic PSOs under abrupt changes and changes of high temporal severity. The influence of temporal severity on algorithm performance increased as the spatial severity increased. This can be explained by the fact that, under gradual changes, all algorithms successfully adapted to the changes in a small number of iterations. Abrupt changes, on the other hand, required more updates to be made to the current model, which required more algorithm iterations. Thus, all considered algorithms struggled under scenarios of frequent to moderately frequent abrupt changes.

Table 6.16 summarises the average ranks obtained under scenarios A1 to A5, B1 to B5, C1 to C5, and D1 to D5, and provides the final average ranks. Table 6.16 shows that the overall top performers in terms of E_T are the RPSO and the QPSO, and the overall top performers in terms of E_G are the CPSO and the QPSO. Thus, the QPSO

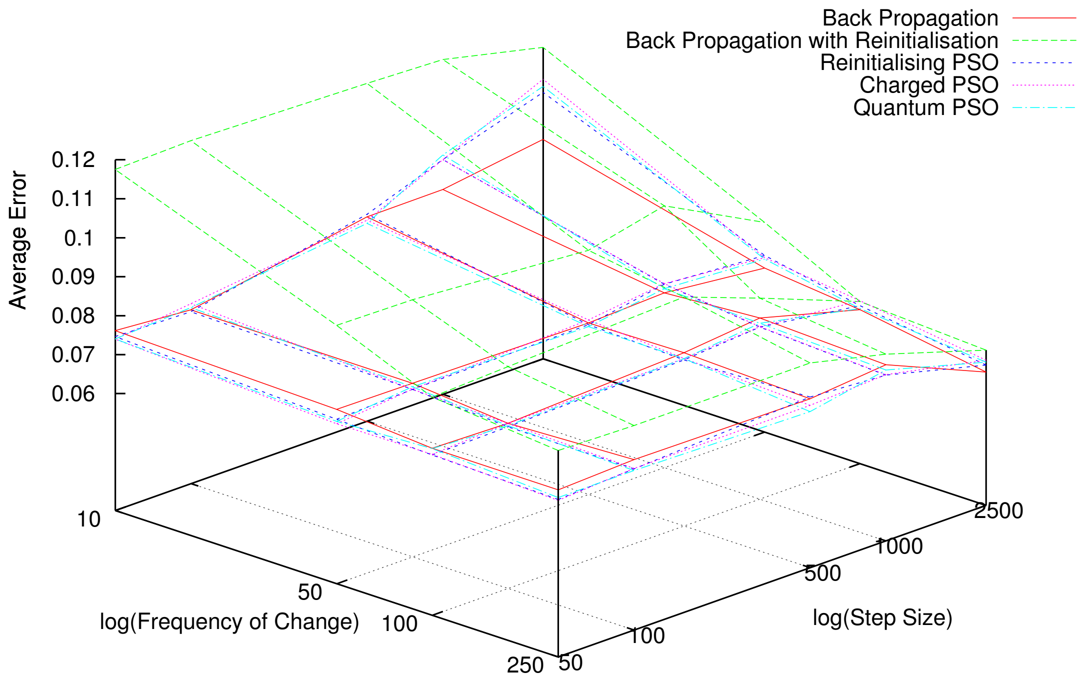


Figure 6.20: Average Generalisation Error Results for SEA Concepts

Table 6.16: SEA Concepts Algorithm Ranking for Scenarios A to D

Algorithm	Average R(A)		Average R(B)		Average R(C)		Average R(D)		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	2.1	2	2.5	2.5	3.2	2.6	3.9	3.3	2.925	2.6
RBP	5	4.7	5	5	4.6	4.6	4.5	4.2	4.775	4.625
RPSO	2.4	3.1	2.3	2.5	2.3	2.6	1.8	2.5	2.2	2.675
CPSO	3.1	2.6	2.9	2.5	2.6	2.6	3	2.5	2.9	2.55
QPSO	2.4	2.6	2.3	2.5	2.3	2.6	1.8	2.5	2.2	2.55

outperformed both the RPSO and the CPSO in terms of either E_T or E_G. Therefore, the QPSO can be considered as the overall top performer for the SEA concepts problem. The QPSO combines the memory capacity (exploitation) of the CPSO with the partial reinitialisation (exploration) of the RPSO, which makes the QPSO the most versatile of the three dynamic PSOs considered in this study. Table 6.16 shows that the worst performance was exhibited by RBP, indicating that a complete restart of the algorithm is not an efficient approach on a simple dynamic problem such as the SEA concepts, and algorithms that adapt to environment changes without a complete restart should be preferred.

6.2.2 Moving Hyperplane

This artificially generated classification problem used a 10-dimensional hyperplane to separate two classes.

Problem Definition

The hyperplane is given by

$$\sum_{l=1}^{n} a_l x_l + c = a_0,$$

where n is the number of dimensions over which the hyperplane is defined, a_l for $l = 1, 2, \ldots, n$ are linear coefficients, and c is a constant. All points satisfying $\sum_{l=1}^{n} a_l x_l + c > a_0$ are labelled as class A, and all points satisfying $\sum_{l=1}^{n} a_l x_l + c \leq a_0$ as class B. For the purpose of this study, n was set to 10, yielding a 10-dimensional hyperplane. The linear coefficients and c are real numbers chosen from the interval $[0, 1]$. A data set was generated according to the procedure described in Section 6.1.2, where the number of data points, M, was set to 1000, and the number of environment changes, N, was set to 10. A set of M 10-dimensional points, $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{1000}\}$, was randomly generated such that $\vec{x}_l \in [0, 1]^{10}$, $l = 1, \ldots, 1000$. The hyperplane was generated N times by uniformly randomising its coefficients, $\{a_1, a_2, \ldots, a_N\} \in [0, 1]$, the constant $c \in [0, 1]$ and the threshold value, $a_0 \in [0, 1]$. Target classification of M patterns was updated N times, and every time the updated patterns were appended to the data set, yielding a data set of 11 000 patterns. The size of the sliding window was set to 1000, equal to M.

Table 6.17: Dynamic Scenarios considered for the Moving Hyperplane

F \ S	**50**	**100**	**500**	**1000**
10	A1	A2	A3	A4
50	B1	B2	B3	B4
100	C1	C2	C3	C4
250	D1	D2	D3	D4

A NN with 10 input units, 6 hidden units (as listed in Table 6.2), and a single output unit was used for the moving hyperplane problem. According to equation (6.3), the total number of weights for this problem was equal to 73. Thus, the dimensionality of the moving hyperplane problem was 73.

The moving hyperplane problem was considered under sixteen different dynamic scenarios. The chosen scenarios simulated different combinations of spatial and temporal severity, providing representative coverage of different dynamic environment types. The technique of simulating dynamic scenarios applied in this study was described in Section 6.1.2. Parameter settings corresponding to each dynamic scenario are listed in Table 6.17. The four different values for the frequency of change F were $\{10, 50, 100, 250\}$, where a value of 10 results in a scenario of high temporal severity (i.e. frequent changes), and a value of 250 results in a scenario of low temporal severity (i.e. infrequent changes). The four different values for the step size S were $\{50, 100, 500, 1000\}$, where a step size of 50 results in a scenario of low spatial severity (i.e. gradual changes), and a value of 1000 results in a scenario of high spatial severity (i.e. abrupt changes).

Table 6.18: Optimal Parameters for the Moving Hyperplane problem

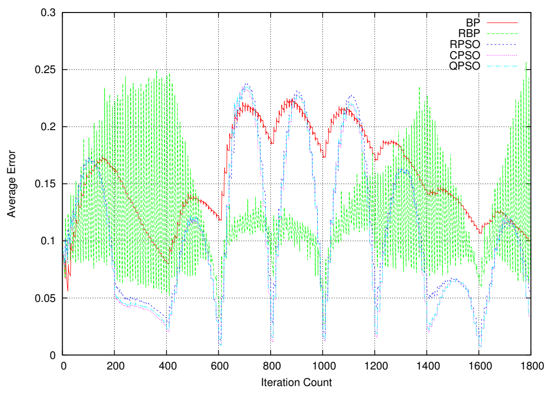
Algorithm	**Parameters**			
Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-2, 2]$	0.1	0.7	
Reinitialising Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-2, 2]$	0.1	0.7	
Reinitialising PSO	Weight Interval	V_{max}	Swarm Size	Reinitialisation Ratio
	$[-1, 1]$	0.5	30	0.5
Charged PSO	Weight Interval	V_{max}	Swarm Size	Charge Magnitude
	$[-2, 2]$	0.5	50	20
Quantum PSO	Weight Interval	V_{max}	Swarm Size	Cloud Radius
	$[-1, 1]$	0.1	30	1

Parameter Optimisation

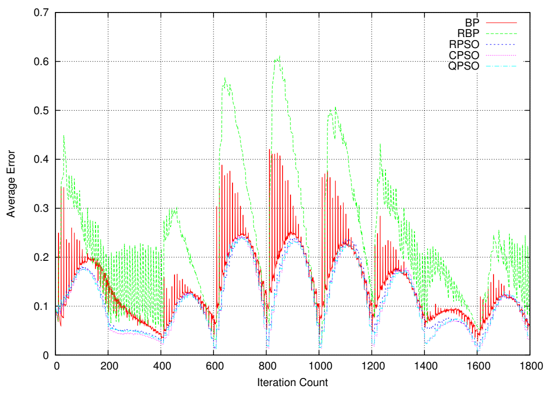
All algorithm parameters were optimised according to the procedure described in Section 6.1.3. Corresponding optimal parameters discovered are listed in Table 6.18.

Analysis of Empirical Data

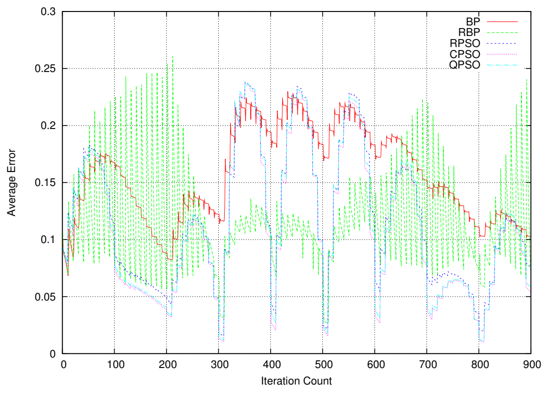
The number of iterations required to traverse the entire data set under every dynamic scenario considered was calculated using equation (6.4). As stated in Section 6.1.1, the two-tailed Mann-Whitney U test was used to determine whether the difference between algorithm performance was of any statistical significance. P-values corresponding to training error comparisons between the algorithms are reported in Table 6.20. P-values corresponding to generalisation error comparisons between the algorithms are reported in Table 6.21. The rest of this section is dedicated to the analysis of results obtained for the moving hyperplane problem.



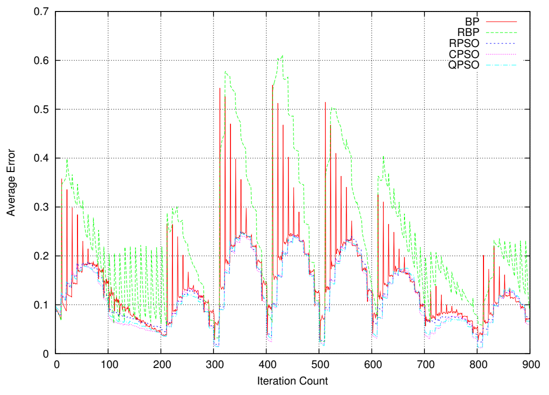
(a) E_T for A1



(b) E_G for A1



(c) E_T for A2



(d) E_G for A2

Figure 6.21: Training and Generalisation Error results for Moving Hyperplane, scenarios A1 to A2

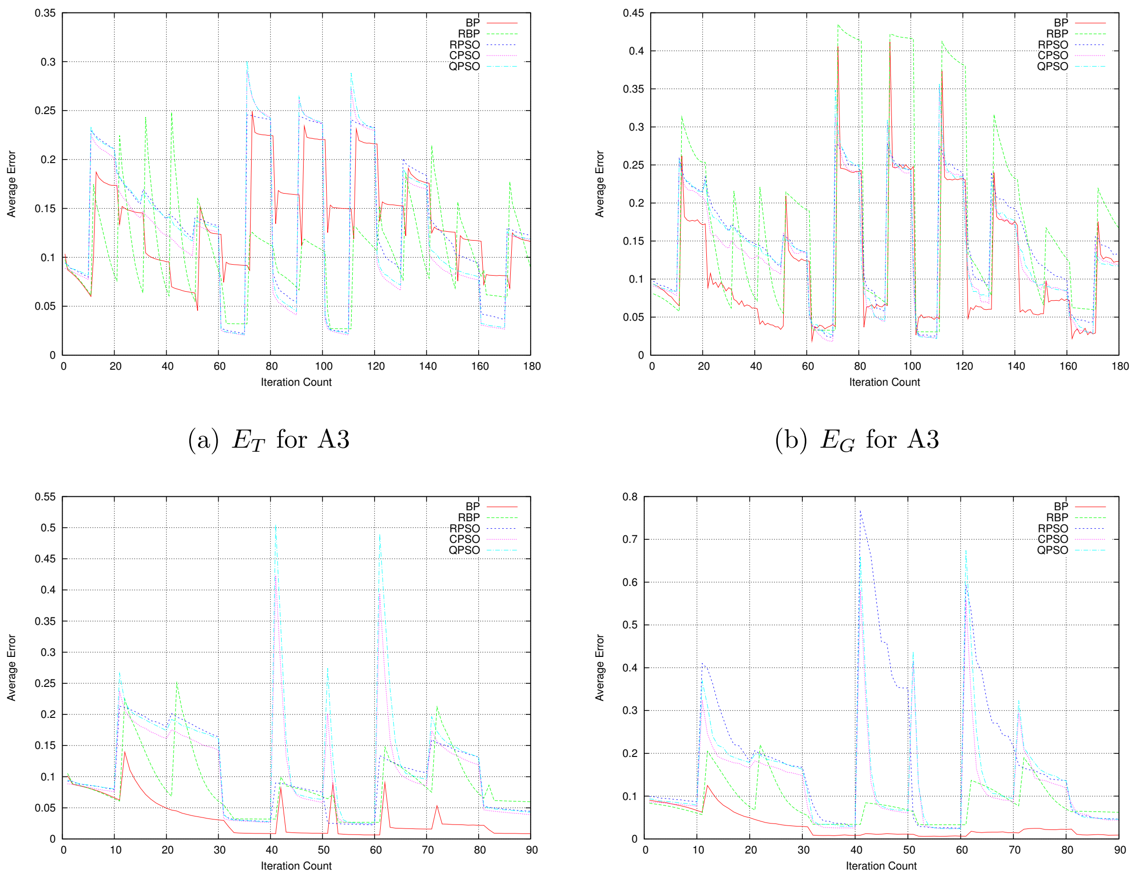
Scenarios A1 to A4: Scenarios A1 to A4 simulated dynamic environments of high temporal severity, $F = 10$. Figures 6.21 and 6.22 illustrate the progression of E_T and E_G over time as obtained by the five algorithms considered. Figure 6.21 illustrates that under gradual scenarios A1 and A2, the E_G error profile of BP peaked after every environment change, as opposed to the three dynamic PSOs, which produced smooth

E_G error profiles. Changes simulated under A1 and A2 were gradual, implying that little new data was added to the sliding window after every change, and the new optimum was expected to be in the near proximity of the old optimum. The population-based principle allowed the dynamic PSOs to promptly evaluate the area covered by the swarm, and immediately find a more up-to-date solution amongst the particles. Hill-climbing BP, on the other hand, required some iterations to move down the slope of the objective function and thus update the solution.

Figure 6.21 also illustrates that complete reinitialisation of weights carried out by RBP did not prove effective under gradual scenarios: E_T produced by RBP fluctuated severely, and RBP's E_G was often much worse than the corresponding E_T. As shown in [38, 55], the main geometrical features of the NN error surfaces are large plateaus, sometimes asymptotically tending towards infinity, as well as step-like transitions, narrow valleys, and steep ridges. Complete reinitialisation of weights might have placed the algorithm onto one of the unfruitful regions of the search space, thus causing RBP to perform poorly. In addition to the danger of being placed in an unfruitful search space region after reinitialisation, RBP had to deal with the possibility of encountering conflicting boundaries in the sliding window. The lack of memory of previously learned information made RBP less apt at discerning between new and stale data than the other algorithms considered.

The supposition that RBP exhibited sensitivity to stale data is confirmed by the fact that RBP's performance suddenly improved under the most abrupt scenario A4, as figures 6.22(c) and 6.22(d) illustrate. Scenario A4 replaced the entire contents of the sliding window after every environment change, thus no stale data was present in either the training or the generalisation set used by the algorithms.

Figure 6.22 illustrates that both BP and RBP showed better performance under abrupt scenarios. Under the most abrupt scenario A4, both BP and RBP adapted to drastic changes better than the dynamic PSOs. Dynamic PSOs struggled to adapt to



(a) E_T for A3

(b) E_G for A3

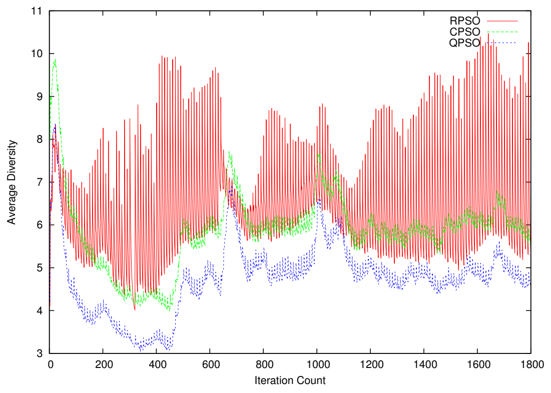
(c) E_T for A4

(d) E_G for A4

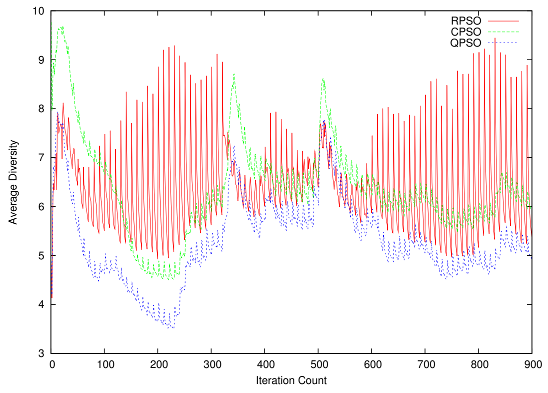
Figure 6.22: Training and Generalisation Error results for Moving Hyperplane, scenarios A3 to A4

abrupt changes under high temporal frequency due to the slow convergence speed of these algorithms. Swarm diversity profiles depicted in Figure 6.23 illustrate that swarm diversity of the three dynamic PSOs considered increased from A1 to A4, indicating that the swarms did not have enough time between environment changes to converge to a stable diversity level.

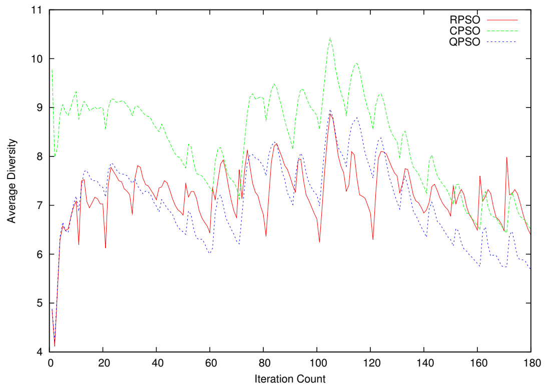
Table 6.19 summarises the collective mean fitness results, generalisation factor values, and average swarm diversity (where applicable) obtained for scenarios A1 to A4. The values of ρ_F in Table 6.19 indicate that BP showed no signs of overfitting across A1 to



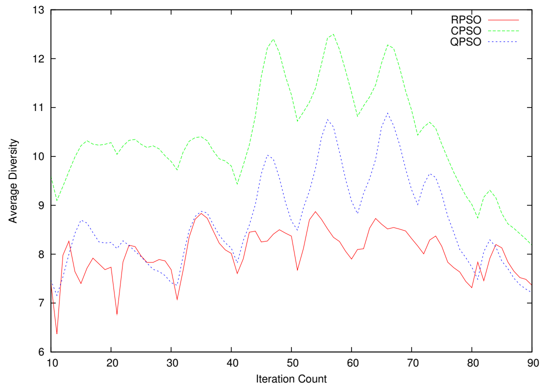
(a) Average Diversity Results for Scenario A1



(b) Average Diversity Results for Scenario A2



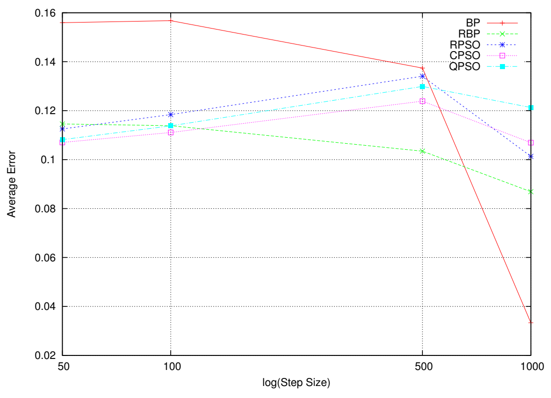
(c) Average Diversity Results for Scenario A3



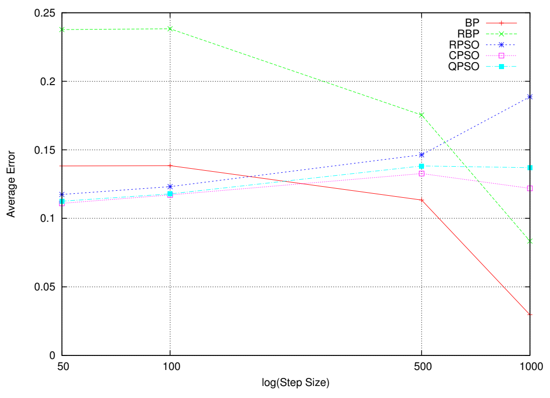
(d) Average Diversity Results for Scenario A4

Figure 6.23: Average Diversity results for Moving Hyperplane, scenarios A1 to A4

A4 ($\rho_F < 1$), and RBP overfitted under all scenarios except A4. Since A4 is the only scenario which introduced no conflicting boundaries, it can be concluded that RBP failed to generalise well for scenarios with conflicting boundaries. As ρ_F in Table 6.19 shows, the dynamic PSOs exhibited overfitting under all scenarios considered ($\rho_F > 1$). It was shown in a recent study by van Wyk and Engelbrecht [119] that the use of bounded activation functions such as the sigmoid function within the neurons of a NN can be a cause of divergent swarm behaviour due to uncontrollable increase of particle velocity, and, subsequently, overfitting. Although V_{max} was used to clamp particle velocities in case of the moving hyperplane problem, it was illustrated in Figure 6.23 that the



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.24: Average Error results for Moving Hyperplane, scenarios A1 to A4

swarms did not converge to a stable diversity level under scenarios A1 to A4, thus non-convergent behaviour can not be excluded from the possible reasons for the overfitting observed. Average diversity values reported in Table 6.19 confirm that swarm diversity increased with the increase of abruptness.

Figure 6.24 summarises the collective mean fitness results obtained by the algorithms over scenarios A1 to A4. Figure 6.24(a) illustrates that the dynamic PSOs significantly outperformed BP in terms of E_T under scenarios A1 to A3 (refer to Table 6.20 for supporting p-values). However, as Figure 6.24(b) illustrates, BP was outperformed by the dynamic PSOs in terms of E_G only under gradual scenarios A1 and A2 (refer to Table 6.21 for supporting p-values). RBP obtained good E_T results, significantly outperforming BP on all scenarios except A4, and significantly outperforming the dynamic PSOs under abrupt scenarios A3 and A4 (see Table 6.20). However, RBP failed to support good E_T results by equally good E_G results on all scenarios except A4, as Figure 6.24(b) illustrates.

Figure 6.24 illustrates that the performance of dynamic PSOs deteriorated from A1 to A3, and improved under scenario A4. Deteriorating performance of the dynamic PSOs from A1 to A3 is attributed to high temporal severity of scenarios A1 to A4,

Table 6.19: Moving Hyperplane Results for Scenarios A1 to A4

Scenario / Algorithm	**A1** *(frequency: 10, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.155488 ± 0.002328	0.132911 ± 0.004695	0.916668 ± 0.042586	n/a
Reinit. Back Propagation	0.114547 ± 0.001191	0.237748 ± 0.004665	2.06725 ± 0.068096	n/a
Reinitialising PSO	0.112531 ± 0.000872	0.117451 ± 0.003418	1.03508 ± 0.048711	6.40206 ± 0.265697
Charged PSO	0.107024 ± 0.001584	0.110943 ± 0.004261	1.02696 ± 0.043137	5.8464 ± 0.360262
Quantum PSO	0.108173 ± 0.00201	0.112485 ± 0.003668	1.03851 ± 0.036693	4.73568 ± 0.268908
	A2 *(frequency: 10, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.156778 ± 0.002442	0.138453 ± 0.005059	0.945204 ± 0.045852	n/a
Reinit. Back Propagation	0.113846 ± 0.001363	0.238293 ± 0.006099	2.05242 ± 0.05134	n/a
Reinitialising PSO	0.11839 ± 0.001208	0.123117 ± 0.004598	1.03222 ± 0.051763	6.38455 ± 0.256964
Charged PSO	0.111091 ± 0.001421	0.117111 ± 0.004598	1.04786 ± 0.04324	6.37904 ± 0.354355
Quantum PSO	0.113791 ± 0.001672	0.117842 ± 0.004111	1.02632 ± 0.049379	5.29956 ± 0.295893
	A3 *(frequency: 10, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.13741 ± 0.001797	0.113383 ± 0.002645	0.888158 ± 0.05447	n/a
Reinit. Back Propagation	0.103456 ± 0.002951	0.175411 ± 0.006053	1.53968 ± 0.063557	n/a
Reinitialising PSO	0.134078 ± 0.003818	0.146382 ± 0.009233	1.07155 ± 0.067244	7.2297 ± 0.237584
Charged PSO	0.12389 ± 0.0039	0.132682 ± 0.005736	1.05531 ± 0.063246	8.32977 ± 0.602811
Quantum PSO	0.129902 ± 0.005268	0.138235 ± 0.007295	1.05146 ± 0.047754	7.07132 ± 0.507922
	A4 *(frequency: 10, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.033327 ± 0.003651	0.029691 ± 0.004585	0.934991 ± 0.132065	n/a
Reinit. Back Propagation	0.08689 ± 0.005457	0.083383 ± 0.00689	0.936677 ± 0.076462	n/a
Reinitialising PSO	0.101326 ± 0.005489	0.188668 ± 0.053221	1.59503 ± 0.31717	7.83715 ± 0.477431
Charged PSO	0.106926 ± 0.008617	0.121909 ± 0.010094	1.09468 ± 0.107971	10.2516 ± 0.647508
Quantum PSO	0.121284 ± 0.01013	0.137007 ± 0.012799	1.10247 ± 0.118956	8.42618 ± 0.666673

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

	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.000199**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.000774**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	0.970925	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.003358**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.552109	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.002107**	**0.0001**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.00082**	**0.0001**	**0.027919**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	0.082354	**0.0001**	**0.0001**	0.299565	0.063239	**0.0001**	**0.000213**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.003034**	**0.001794**	**0.0001**	0.342795
D	**0.0001**	**0.0001**	0.52295	**0.0001**	**0.004311**	0.393843	0.063242	**0.038533**

Table 6.21: Mann-Whitney U p-values obtained for the average generalisation error comparisons on the Moving Hyperplane problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.708548	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.006047**	**0.0001**	**0.0001**	**0.0001**
D	0.090617	0.644024	**0.0001**	**0.0001**	**0.003358**	0.889553	**0.0001**	**0.0001**
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	0.959278	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	0.820526	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	0.146226	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.775176	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.218715	**0.0001**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.000332**	**0.0001**	0.168039	0.503955	**0.002468**	**0.0001**
B	**0.0001**	0.889553	**0.0001**	**0.000175**	0.633505	0.62306	**0.044588**	**0.001092**
C	**0.0001**	**0.0001**	**0.007297**	**0.0001**	**0.044588**	**0.002599**	**0.001222**	0.675994
D	**0.0001**	**0.0001**	0.797771	**0.0001**	0.197283	0.633505	0.053247	0.12295

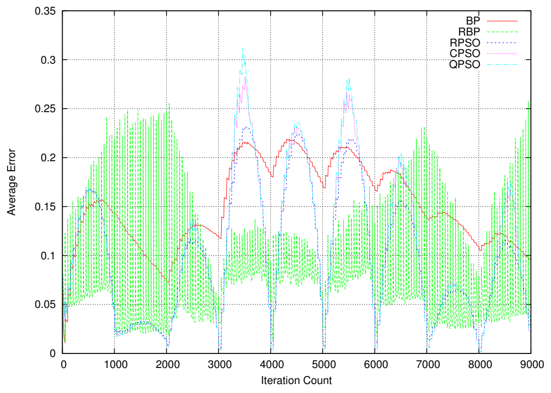
Table 6.22: Moving Hyperplane Algorithm Ranking for Scenarios A1 to A4

Algorithm	A1		A2		A3		A4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	5	1	1	1	4	2.5
RBP	4	5	2.5	5	1	5	2	2	2.375	4.25
RPSO	3	3	4	3	4	4	3	5	3.5	3.75
CPSO	1	1.5	1	1.5	2	2	4	3	2	2
QPSO	2	1.5	2.5	1.5	3	3	5	4	3.125	2.5

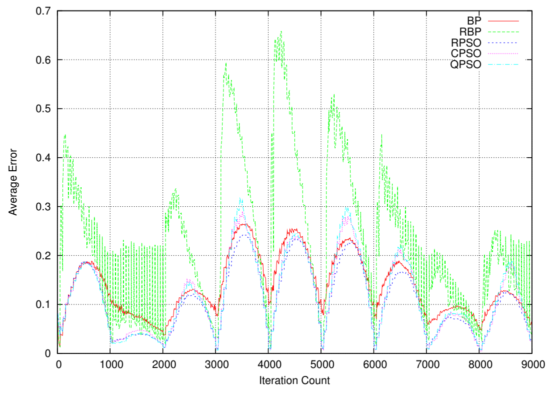
which did not allow the dynamic PSOs enough iterations to exploit found solutions. The performances of BP and RBP improved from A1 to A4, and improvement from A3 to A4 was visibly drastic. This performance improvement observed for all algorithms considered under scenario A4 can be explained by the fact that the step size of 1000 was equal to the window size, thus every time the environment changed, all patterns inside the window were replaced. A complete replacement of patterns in the sliding window does not allow for conflicting decision boundaries, and the algorithms are presented with up-to-date data only. Hence, the learning task is simplified.

The algorithms were ranked based on the E_T and E_G values reported in Table 6.19, taking the p-values in Tables 6.20 and 6.21 into account. The obtained ranks, together with average ranks, are reported in Table 6.22. Table 6.22 shows that, out of the three dynamic PSOs, the CPSO consistently obtained the highest rank. The CPSO also obtained the highest average rank over scenarios A1 to A4, compared to the other algorithms considered. The CPSO, as opposed to the RPSO and the QPSO, never reinitialises particles, thus the swarm memory is preserved after environment changes. Larger memory capacity made the CPSO superior to the other dynamic PSOs under frequent environment changes exhibited by scenarios A1 to A4.

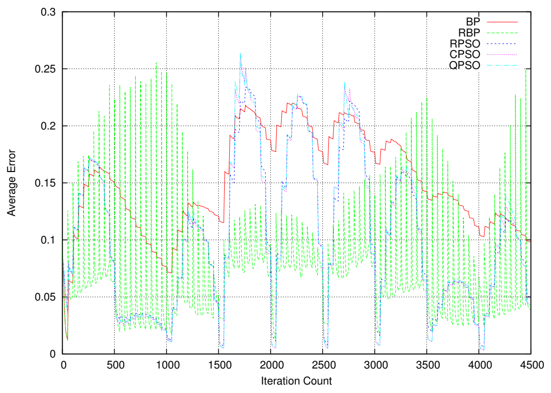
Scenarios B1 to B4: Figures 6.25 and 6.26 illustrate the progression of E_T and E_G over time as obtained by the algorithms under scenarios B1 to B4. Similar trends as were



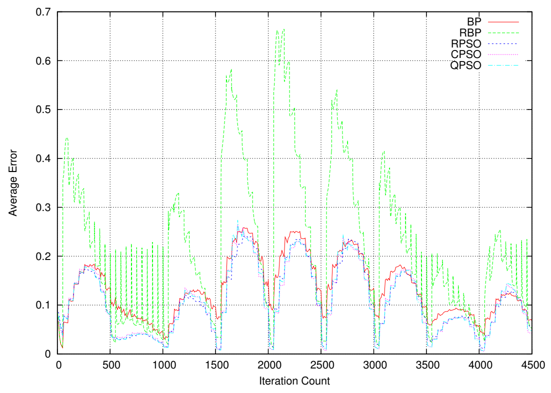
(a) E_T for B1



(b) E_G for B1



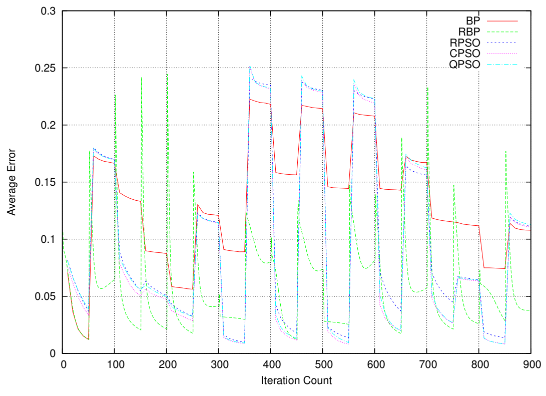
(c) E_T for B2



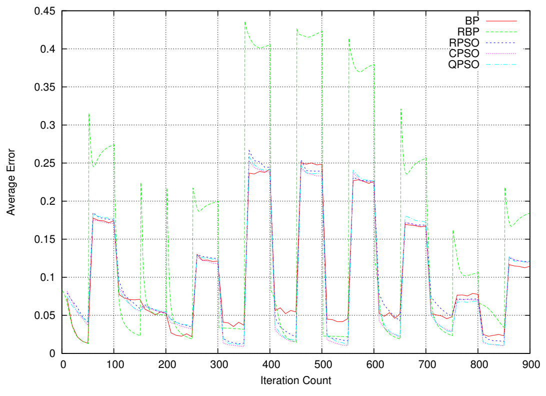
(d) E_G for B2

Figure 6.25: Training and Generalisation Error results for Moving Hyperplane, scenarios B1 to B2

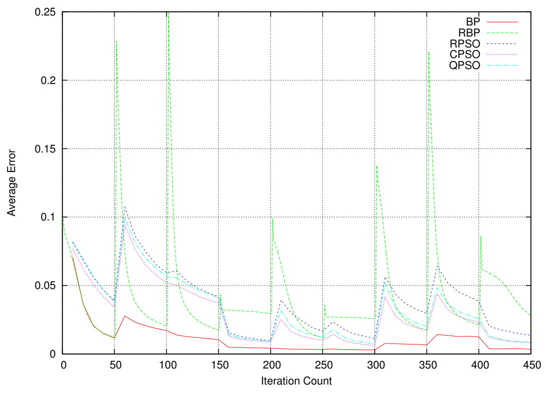
observed for A1 to A4 can be observed here: RBP once again severely fluctuated in terms of E_T under gradual scenarios B1 and B2, and did not generalise well under B1 to B3. In fact, a closer examination of Figure 6.26(b) reveals that between environment changes, RBP's E_G often increased while the E_G produced by the other algorithms decreased. A visual comparison of RBP's E_T profile to the RBP's E_G profile illustrated in figures 6.26(a) and 6.26(b), respectively, reveals that RBP's E_G sometimes increased while the RBP's E_T decreased. An increase in E_G with the simultaneous decrease in E_T is a typical sign of overfitting [32]. Scenarios B1 to B3 did not replace the entire contents of



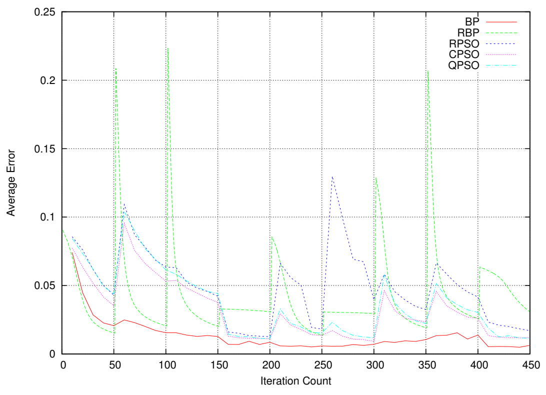
(a) E_T for B3



(b) E_G for B3



(c) E_T for B4

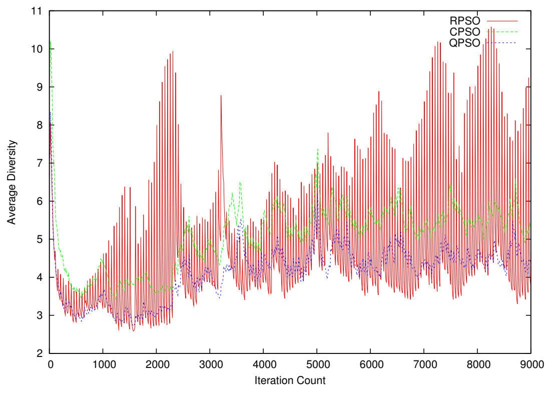


(d) E_G for B4

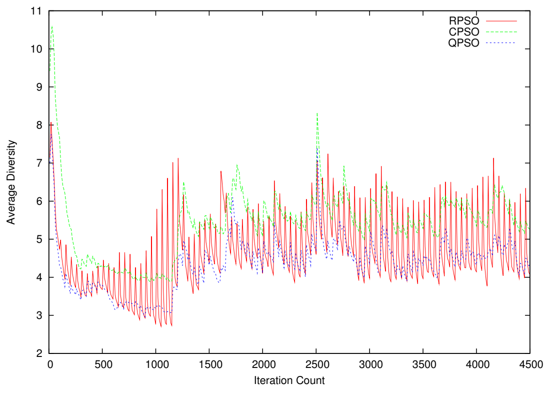
Figure 6.26: Training and Generalisation Error results for Moving Hyperplane, scenarios B3 to B4

the sliding window, since the step sizes of these scenarios were not equal to the sliding window size. Therefore, the sliding window contained both up-to-date and obsolete data most of the time, and the obsolete data confused RBP and caused overfitting.

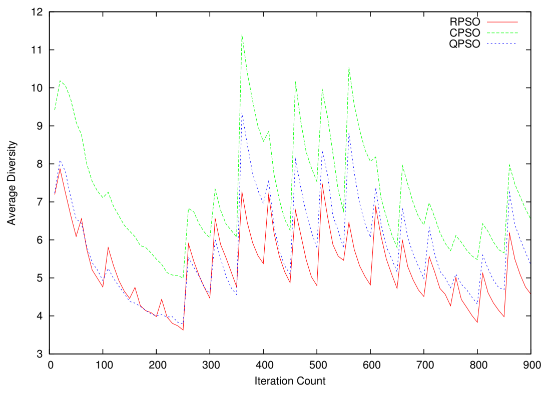
The dynamic PSOs performed better under B1 to B4 than under A1 to A4, as figures 6.25 and 6.26 illustrate. Given more time to converge around a solution ($F = 50$), the dynamic PSOs managed to exploit better than BP and generalise better than both BP and RBP under scenarios B1 to B3. The swarm diversity graphs in Figure 6.27 confirm that the dynamic PSOs converged to smaller areas between changes under scenarios B1



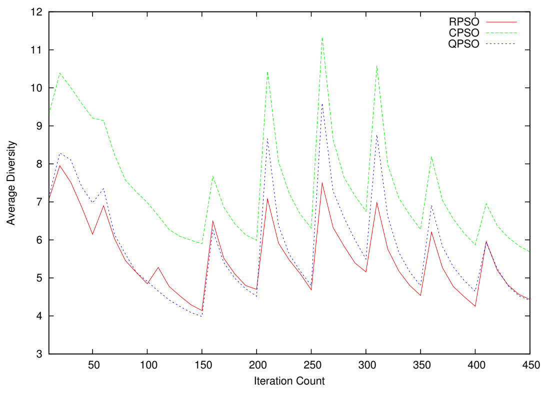
(a) Average Diversity Results for Scenario B1



(b) Average Diversity Results for Scenario B2



(c) Average Diversity Results for Scenario B3



(d) Average Diversity Results for Scenario B4

Figure 6.27: Average Diversity results for Moving Hyperplane, scenarios B1 to B4

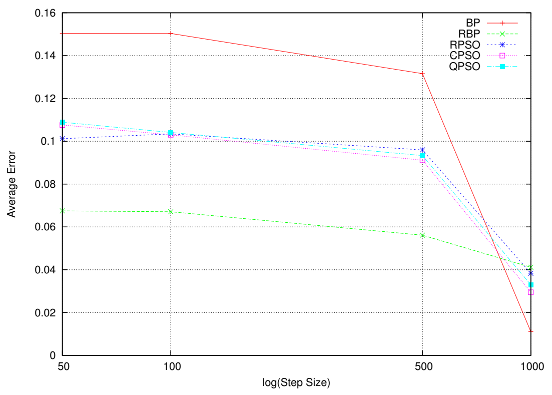
to B4 than under A1 to A4.

The ability of the dynamic PSOs to perform well for environments where conflicting boundaries may be present indicates that the dynamic PSOs are more tolerant to obsolete data than both BP and RBP. Hence, the dynamic PSOs may show better performance than BP and RBP on more complex NN error surfaces.

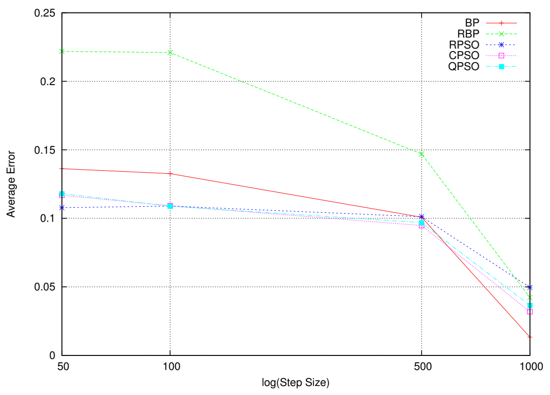
For the sake of a sound statistical comparison, collective mean E_T and E_G values for scenarios B1 to B4 were calculated, and are reported in Table 6.23, together with average ρ_F and average swarm diversity (where applicable). The ρ_F values confirm that RBP exhibited overfitting under scenarios B1 to B4. BP was not susceptible to overfitting

Table 6.23: Moving Hyperplane Results for Scenarios B1 to B4

Scenario / **Algorithm**	**B1** *(frequency: 50, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.1504 ± 0.00209	0.136252 ± 0.00552	0.971071 ± 0.06629	n/a
Reinit. Back Propagation	0.067478 ± 0.000596	0.221859 ± 0.005365	3.38495 ± 0.162336	n/a
Reinitialising PSO	0.101187 ± 0.00147	0.10774 ± 0.003387	1.07572 ± 0.059796	4.61891 ± 0.548936
Charged PSO	0.107609 ± 0.004842	0.116852 ± 0.006497	1.11021 ± 0.062296	5.09738 ± 0.397126
Quantum PSO	0.108914 ± 0.004429	0.11807 ± 0.006523	1.0803 ± 0.049455	4.13723 ± 0.27412
	B2 *(frequency: 50, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.150329 ± 0.001864	0.132623 ± 0.005332	0.929454 ± 0.054669	n/a
Reinit. Back Propagation	0.067098 ± 0.000852	0.220962 ± 0.006585	3.33462 ± 0.16363	n/a
Reinitialising PSO	0.103442 ± 0.001701	0.108933 ± 0.003523	1.0372 ± 0.0554	4.60983 ± 0.30018
Charged PSO	0.103015 ± 0.002516	0.109161 ± 0.004026	1.06099 ± 0.046747	5.45385 ± 0.35963
Quantum PSO	0.104123 ± 0.002512	0.108856 ± 0.003687	1.04081 ± 0.058347	4.35616 ± 0.350523
	B3 *(frequency: 50, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.131567 ± 0.001894	0.100828 ± 0.003609	0.814363 ± 0.038969	n/a
Reinit. Back Propagation	0.056117 ± 0.001308	0.14698 ± 0.005182	2.31657 ± 0.111655	n/a
Reinitialising PSO	0.095947 ± 0.001563	0.101191 ± 0.0039	1.04492 ± 0.056886	5.28544 ± 0.17348
Charged PSO	0.091132 ± 0.001667	0.094755 ± 0.003897	1.04192 ± 0.054619	7.23212 ± 0.701655
Quantum PSO	0.093299 ± 0.001708	0.09686 ± 0.003581	1.02721 ± 0.058822	5.77949 ± 0.302558
	B4 *(frequency: 50, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.011053 ± 0.000871	0.013329 ± 0.001963	1.17842 ± 0.16693	n/a
Reinit. Back Propagation	0.04116 ± 0.001592	0.042341 ± 0.003982	1.0141 ± 0.084281	n/a
Reinitialising PSO	0.038352 ± 0.003989	0.049599 ± 0.022523	1.18155 ± 0.337356	5.5384 ± 0.456901
Charged PSO	0.029461 ± 0.003598	0.031942 ± 0.004398	1.04907 ± 0.112993	7.44741 ± 0.943585
Quantum PSO	0.032953 ± 0.004226	0.03649 ± 0.005777	1.08903 ± 0.154251	5.85468 ± 0.384186



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.28: Average Error results for Moving Hyperplane, scenarios B1 to B4

under B1 to B3 ($\rho_F < 1$), but produced $\rho_F > 1$ under B4. Since B4 was the easiest dynamic scenario for BP to optimise, BP required less than 50 iterations to find a good solution. Training for 50 iterations caused BP to train for too long and thus overfit the data. All dynamic PSOs exhibited minor overfitting under B1 to B4.

Average swarm diversity values in Table 6.23 show that swarm diversity increased as the spatial severity increased from B1 to B4, but not as severely as under A1 to A4. As Figure 6.27 illustrates, the dynamic PSOs managed to converge to smaller areas around a solution under scenarios B1 to B4, thus the swarm diversity fluctuated less severely between environment changes.

Figure 6.28 illustrates the collective mean E_T and E_G obtained by the algorithms under B1 to B4. Figure 6.28 illustrates that, under a change frequency of 50 iterations, performance of all algorithms improved in terms of both E_T and E_G as the spatial severity increased. This can be explained by the fact that with the increasing abruptness of changes, the problem of conflicting boundaries became less severe, since the data inside the sliding window was refreshed faster (i.e. larger blocks of obsolete data were removed whenever a change occurred).

Figure 6.28 once again illustrates that under scenarios where conflicting boundaries may have been present, RBP significantly outperformed all algorithms in terms of E_T and was significantly outperformed by all algorithms in terms of E_G (see Tables 6.20 and 6.21 for supporting p-values), proving a complete reinitialisation of NN weights to be an inefficient approach for the moving hyperplane problem.

As Figure 6.28 illustrates, BP was significantly outperformed by the dynamic PSOs in terms of E_T and E_G under scenarios B1 to B3 (see Tables 6.20 and 6.21 for supporting p-values). Under scenario B3, the gap between E_G produced by BP and the dynamic PSOs became narrower, and, although BP was still significantly outperformed by the CPSO and the QPSO, no statistically significant difference in terms of E_G was observed between BP and the RPSO. Under the most abrupt scenario B4, BP significantly outperformed all the other algorithms in terms of both E_T and E_G. Thus, the dynamic PSOs showed better performance under gradual scenarios and scenarios where conflicting decision boundaries may have been present, and BP was the top performer under abrupt changes with no conflicting boundaries.

Figure 6.28 also illustrates that the RPSO significantly outperformed the other two dynamic PSOs under scenario B1, and was significantly outperformed by the CPSO and the QPSO under abrupt scenarios B3 and B4 (see Tables 6.20 and 6.21 for supporting p-values). As Figure 6.27 illustrated, the RPSO explored better than both the CPSO and the QPSO under B1, and, as the abruptness of the changes increased from B1 to B4 and the problem of conflicting boundaries became less severe, the RPSO converged to smaller areas around a solution, eventually exploring less than the other two dynamic PSOs under abrupt scenarios B3 and B4. Decreases in swarm diversity and consequently deteriorating exploration ability caused the RPSO to perform significantly worse than the CPSO and the QPSO under B3 and B4. The CPSO, on the other hand, as Figure 6.27 illustrates, maintained the highest diversity level out of the three dynamic PSOs considered over scenarios B2 to B4. Therefore, the CPSO managed to outperform both the RPSO and the QPSO under abrupt scenarios B3 and B4, where exploration was

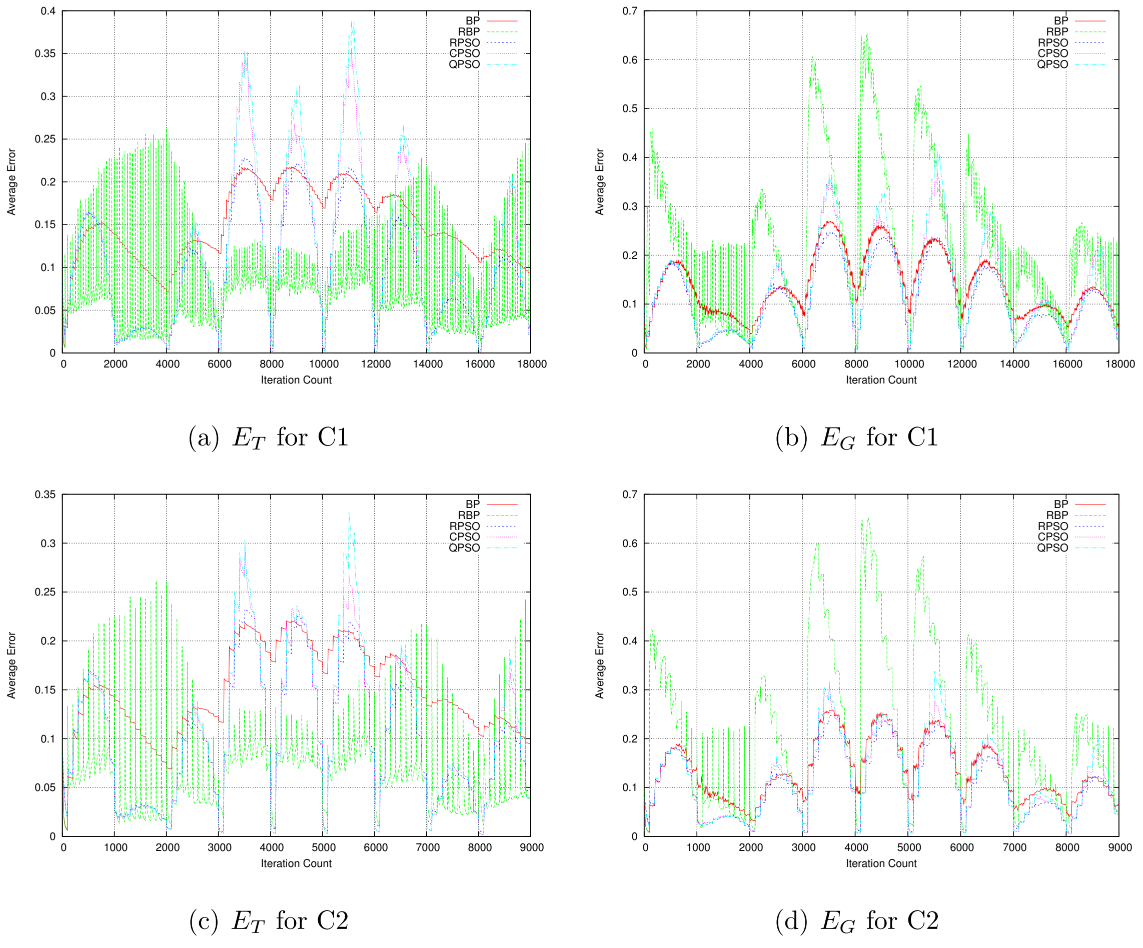
Table 6.24: Moving Hyperplane Algorithm Ranking for Scenarios B1 to B4

Algorithm	B1		B2		B3		B4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	5	3.5	1	1	4	3.125
RBP	1	5	1	5	1	5	5	4.5	2	4.875
RPSO	2	1	3	2	4	3.5	4	4.5	3.25	2.75
CPSO	3.5	2.5	3	2	2	1	2	2	2.625	1.875
QPSO	3.5	2.5	3	2	3	2	3	3	3.125	2.375

most crucial.

The algorithms were ranked in terms of E_T and E_G (taking the p-values in Tables 6.20 and 6.21 into account), and the obtained ranks together with the average ranks are reported in Table 6.24. The average rank confirms that RBP came first in terms of training, but obtained the lowest E_G rank at the same time. The CPSO obtained the second-highest average rank after RBP, and the highest average rank in terms of E_G. Thus, the CPSO can be considered as the top performer among the algorithms considered for scenarios B1 to B4.

Scenarios C1 to C4: Figures 6.29 and 6.30 illustrate the E_T and E_G profiles over time as obtained by the five training algorithms considered under scenarios C1 to C4. Scenarios C1 to C4 simulated changes of a moderately low temporal severity ($F = 100$), thus the algorithms were given more time to refine a solution between environment changes. Figure 6.29 illustrates that the performance of the CPSO and the QPSO deteriorated under gradual scenarios C1 and C2 compared to B1 and B2 (illustrated in Figure 6.25): both E_T and E_G under C1 and C2 peaked higher after environment changes than under B1 and B2. Performance deterioration of the CPSO and the QPSO under gradual scenarios with a decrease in temporal severity can be attributed to overfitting behaviour: since the gradual changes were easy to adjust to for these dynamic PSOs, the extra iterations between environment changes were used to refine an already good solution,

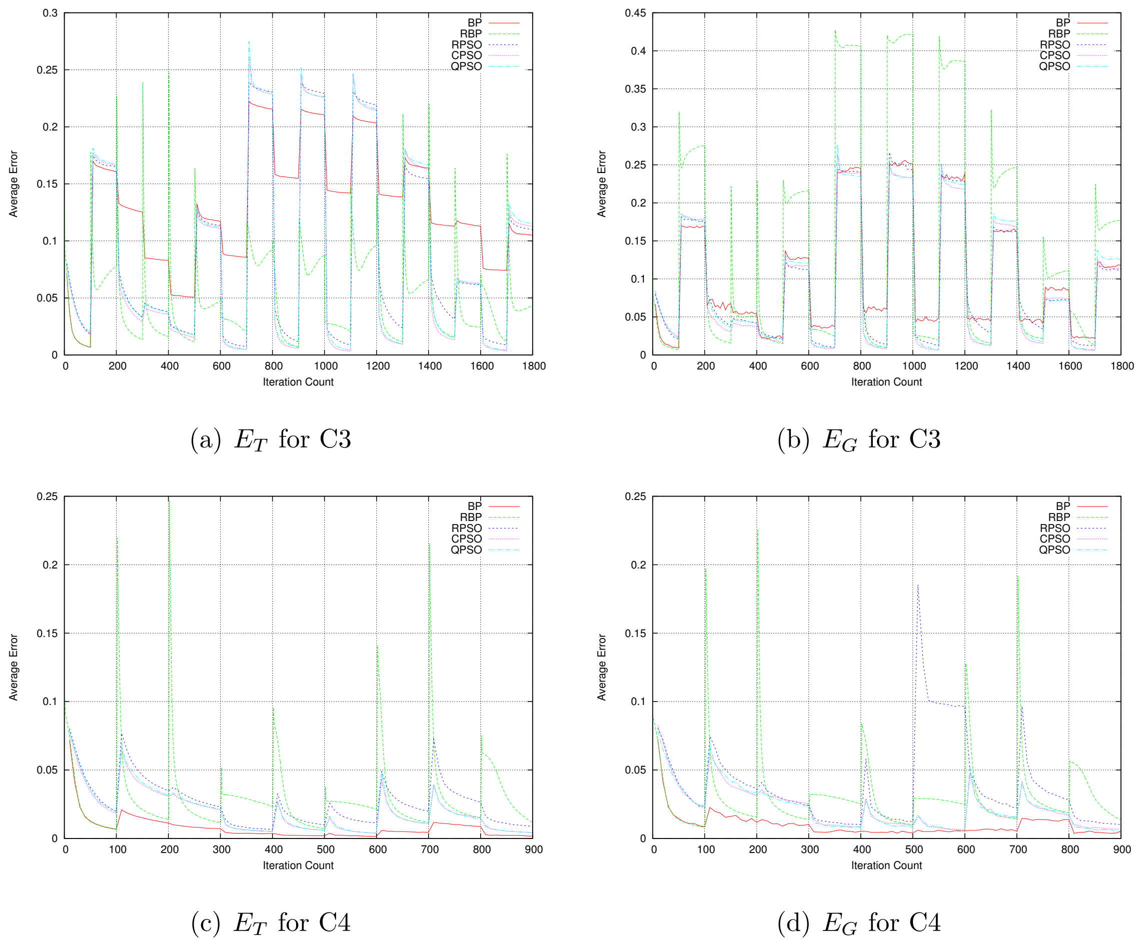


(a) E_T for C1 (b) E_G for C1

(c) E_T for C2 (d) E_G for C2

Figure 6.29: Training and Generalisation Error results for Moving Hyperplane, scenarios C1 to C2

thus causing the dynamic PSOs to train for too long and overfit the training data. The ρ_F values in Table 6.25 confirm that the dynamic PSOs exhibited overfitting under C1 and C2. Figure 6.29 illustrates that the RPSO was an exception to this rule and did not significantly deteriorate under gradual scenarios C1 and C2: the RPSO is the least memory-dependent of the three dynamic PSOs considered, because RPSO reinitialises a percentage of randomly selected particles, not sparing the neighbourhood best particles if these particles happened to be chosen for reinitialisation. Weaker dependence on memory helped the RPSO to promptly "unlearn" the obsolete information after every



(a) E_T for C3 (b) E_G for C3

(c) E_T for C4 (d) E_G for C4

Figure 6.30: Training and Generalisation Error results for Moving Hyperplane, scenarios C3 to C4

environment change.

However, under abrupt scenarios C3 and C4, as Figure 6.30 illustrates, the dynamic PSOs performed better than under B3 and B4 (refer to Figure 6.26): a lower minimum E_T and E_G were reached by the dynamic PSOs. The abrupt scenarios required the dynamic PSOs to move further away from the previous solution than in case of gradual scenarios, thus the extra iterations between environment changes offered by $F = 100$ allowed the dynamic PSOs to move closer to the new optimum than for $F = 50$.

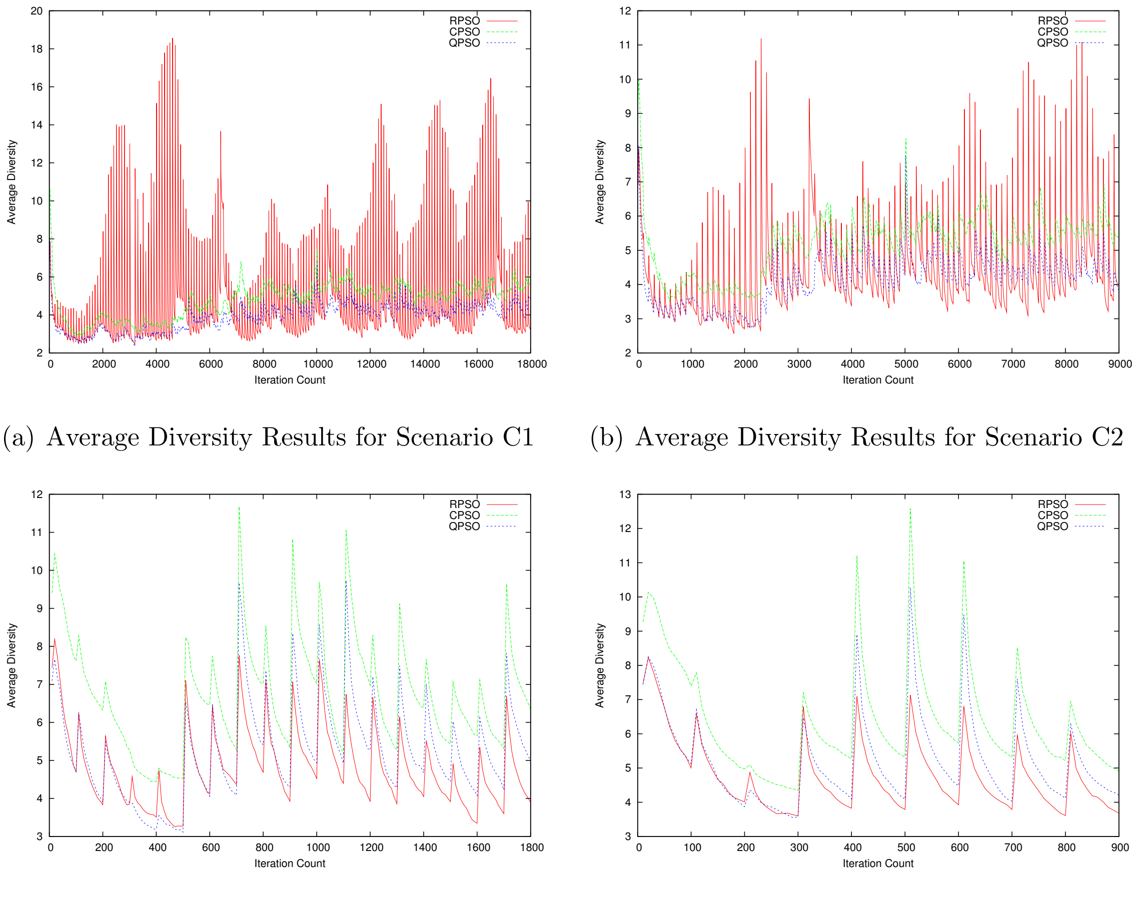
Table 6.25 reports the collective mean E_T and E_G values obtained by the algorithms for C1 to C4. E_T and E_G values produced by BP and RBP indicate that the performance of these algorithms has slightly improved compared to results obtained for B1 to B4 (reported in Table 6.23). Table 6.25 also lists the average ρ_F obtained by the algorithms for C1 to C4, as well as the average swarm diversity (where applicable). The ρ_F values obtained by the dynamic PSOs for C1 to C4 increased compared to ρ_F values for B1 to B4. This again indicates that the dynamic PSOs were susceptible to overfitting, and the susceptibility to overfitting increased together with the decrease in change frequency. Hence, training the NN with dynamic PSO for too long is a likely cause of dynamic PSO overfitting. A unique characteristic of the dynamic PSOs compared to a standard PSO is that the dynamic PSOs maintain diversity. Therefore, the dynamic PSOs never cease to search for better and more up-to-date solutions than the current solution. However, when the changes occur infrequently, the constant search for a better solution causes the dynamic PSOs to fit the training data too closely, having a negative effect on the generalisation ability of the trained NN.

The ρ_F values in Table 6.25 show that RBP overfitted severely under all scenarios where conflicting decision boundaries may have been present, indicating that RBP can not efficiently handle conflicting decision boundaries. Overfitting of RBP under C4 was minor. BP, on the other hand, was not susceptible to overfitting under C1 to C3 ($\rho_F < 1$), but overfitted under C4. No conflicting decision boundaries were introduced under scenario C4, and the sliding window contained only one decision boundary (the hyperplane) inside the sliding window at any one time. Thus, the learning task was rather simple, and overfitting exhibited by BP can be attributed to low temporal severity and training for too long.

The average swarm diversity values in Table 6.25 show that the swarm diversity of the three dynamic PSOs increased with an increase of change abruptness, but not as much as under B1 to B4. Swarm diversity profiles over time, as obtained by the dynamic PSOs under C1 to C4, are illustrated in Figure 6.31. Figure 6.31 illustrates that the

Table 6.25: Moving Hyperplane Results for Scenarios C1 to C4

Algorithm \ Scenario	**C1** *(frequency: 100, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.149038 ± 0.002822	0.138761 ± 0.007095	0.99098 ± 0.064281	n/a
Reinit. Back Propagation	0.05963 ± 0.000607	0.221091 ± 0.006619	3.80307 ± 0.204968	n/a
Reinitialising PSO	0.098834 ± 0.001292	0.11088 ± 0.004162	1.12948 ± 0.067907	4.61096 ± 0.658315
Charged PSO	0.115017 ± 0.005976	0.129655 ± 0.008136	1.1404 ± 0.043485	4.79946 ± 0.376872
Quantum PSO	0.119668 ± 0.006391	0.133049 ± 0.007534	1.14016 ± 0.058306	3.93866 ± 0.382199
	C2 *(frequency: 100, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.149106 ± 0.001946	0.132797 ± 0.005508	0.951655 ± 0.058313	n/a
Reinit. Back Propagation	0.059685 ± 0.001031	0.221858 ± 0.005652	3.70518 ± 0.175813	n/a
Reinitialising PSO	0.100531 ± 0.001239	0.107885 ± 0.003989	1.08224 ± 0.061265	4.36784 ± 0.468695
Charged PSO	0.105563 ± 0.004117	0.114033 ± 0.004939	1.09295 ± 0.057737	5.19947 ± 0.389818
Quantum PSO	0.109364 ± 0.005167	0.118749 ± 0.00727	1.09151 ± 0.071543	4.1837 ± 0.292221
	C3 *(frequency: 100, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.128096 ± 0.001797	0.10101 ± 0.003321	0.833071 ± 0.051034	n/a
Reinit. Back Propagation	0.046808 ± 0.001316	0.141273 ± 0.004968	2.68398 ± 0.130417	n/a
Reinitialising PSO	0.088614 ± 0.001428	0.093955 ± 0.004054	1.06138 ± 0.067674	4.81002 ± 0.268938
Charged PSO	0.084396 ± 0.001449	0.08831 ± 0.00282	1.05682 ± 0.060507	6.75772 ± 0.487067
Quantum PSO	0.086369 ± 0.001536	0.091123 ± 0.003255	1.06002 ± 0.049214	5.26014 ± 0.361809
	C4 *(frequency: 100, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.007695 ± 0.000638	0.009819 ± 0.001473	1.16462 ± 0.144197	n/a
Reinit. Back Propagation	0.0294 ± 0.001623	0.030122 ± 0.00328	1.01307 ± 0.133192	n/a
Reinitialising PSO	0.026101 ± 0.003031	0.038808 ± 0.028168	1.24758 ± 0.458551	4.78532 ± 0.321161
Charged PSO	0.019177 ± 0.003232	0.022115 ± 0.002804	1.12815 ± 0.139965	6.40992 ± 0.320218
Quantum PSO	0.019719 ± 0.002443	0.021954 ± 0.003473	1.07126 ± 0.155068	5.23114 ± 0.358585



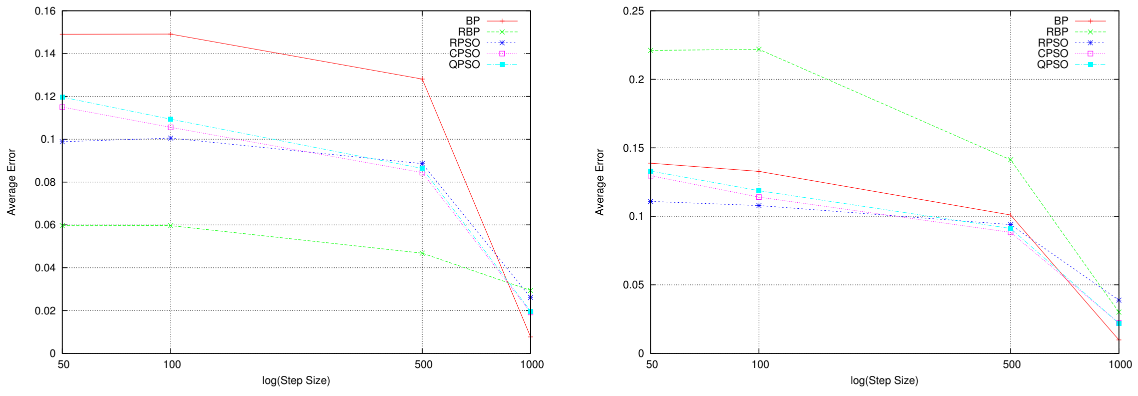
(a) Average Diversity Results for Scenario C1 (b) Average Diversity Results for Scenario C2

(c) Average Diversity Results for Scenario C3 (d) Average Diversity Results for Scenario C4

Figure 6.31: Average Diversity results for Moving Hyperplane, scenarios C1 to C4

swarms reached a lower minimum swarm diversity given more time to converge between environment changes.

Figure 6.32 illustrates the progression of collective mean E_T and E_G values over scenarios C1 to C4 as obtained by the five algorithms considered. RBP once again significantly outperformed all other algorithms in terms of E_T for C1 to C3, but at the same time performed significantly worse than all other algorithms in terms of E_G under the same scenarios (refer to Tables 6.20 and 6.21 for supporting p-values). The dynamic PSOs exhibited similar trends as were observed under scenarios B1 to B4: the RPSO performed significantly better than the other two dynamic PSOs under gradual scenarios



(a) Average Training Error Results (b) Average Generalisation Error Results

Figure 6.32: Average Error results for Moving Hyperplane, scenarios C1 to C4

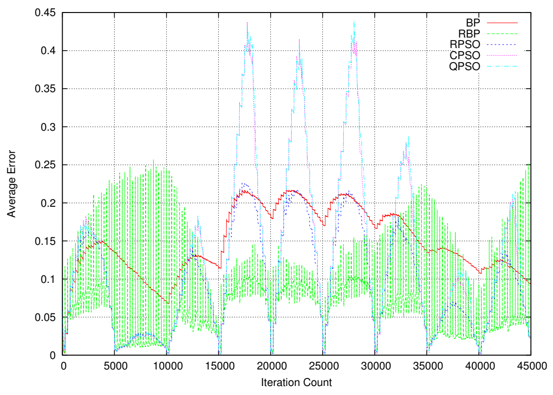
C1 and C2, but significantly worse under abrupt scenarios C3 and C4 (refer to Tables 6.20 and 6.21 for supporting p-values). The dynamic PSOs, despite some deterioration in performance under gradual scenarios due to training for too long, still managed to significantly outperform BP under scenarios C1 to C3. BP produced the lowest E_T and E_G values under scenario C4. However, as the ρ_F value in Table 6.25 indicates, BP was susceptible to overfitting under this scenario.

Algorithm ranks are reported in Table 6.26. As shown by the average ranks, RBP again came first in terms of E_T and last in terms of E_G, and the CPSO came second in terms of E_T and first in terms of E_G. Thus, once again the CPSO is considered as the top performer.

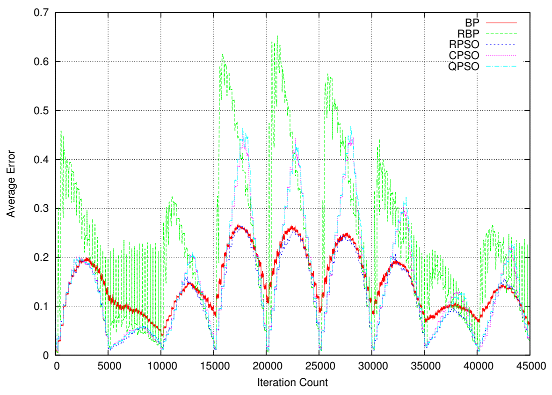
Scenarios D1 to D4: Figures 6.33 and 6.34 illustrate the progression of E_T and E_G over time as obtained by the algorithms under scenarios D1 to D4. Figure 6.33 illustrates that the CPSO and the QPSO deteriorated in performance under gradual scenarios with a decrease in change frequency from $F = 100$ to $F = 250$: maximum E_T and E_G values produced by the CPSO and the QPSO under D1 and D2 were higher than the corresponding maximum values under C1 and C2. Thus, the CPSO and the

Table 6.26: Moving Hyperplane Algorithm Ranking for Scenarios C1 to C4

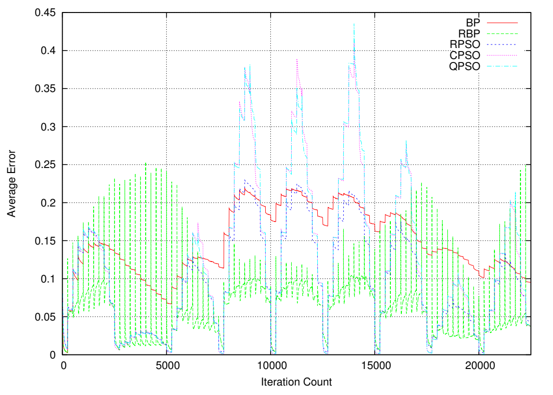
Algorithm	C1		C2		C3		C4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	5	4	1	1	4	3.25
RBP	1	5	1	5	1	5	5	4.5	2	4.875
RPSO	2	1	2	1	4	3	4	4.5	3	2.375
CPSO	3	2	3	2	2	1	2.5	2.5	2.625	1.875
QPSO	4	3	4	3	3	2	2.5	2.5	3.375	2.625



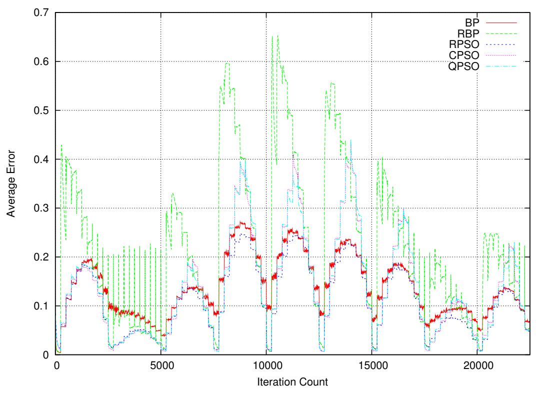
(a) E_T for D1



(b) E_G for D1

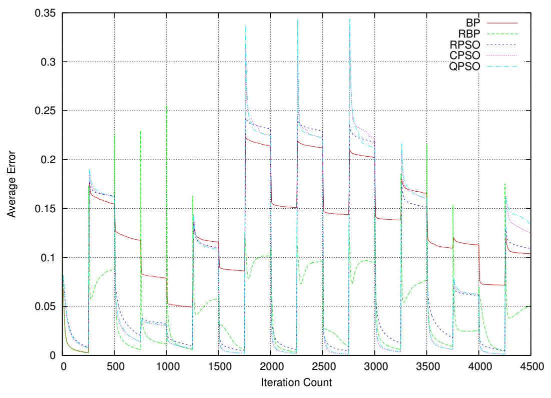


(c) E_T for D2

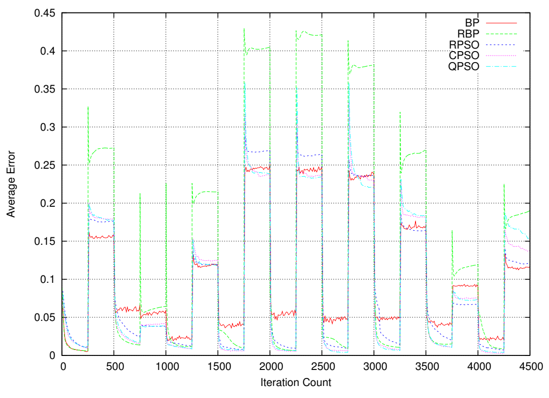


(d) E_G for D2

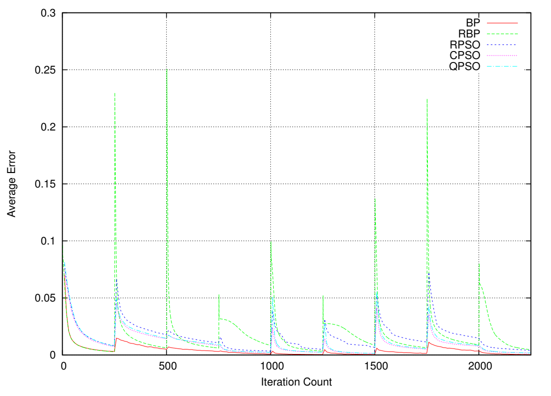
Figure 6.33: Training and Generalisation Error results for Moving Hyperplane, scenarios D1 to D2



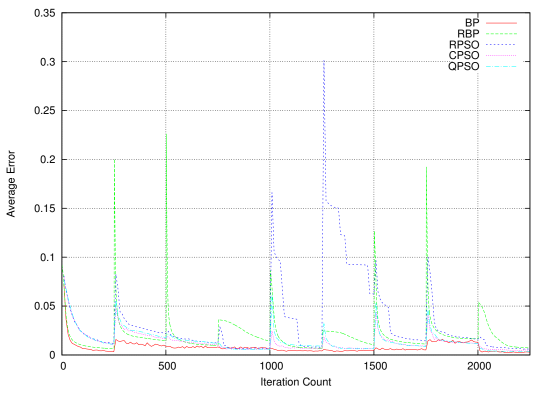
(a) E_T for D3



(b) E_G for D3



(c) E_T for D4

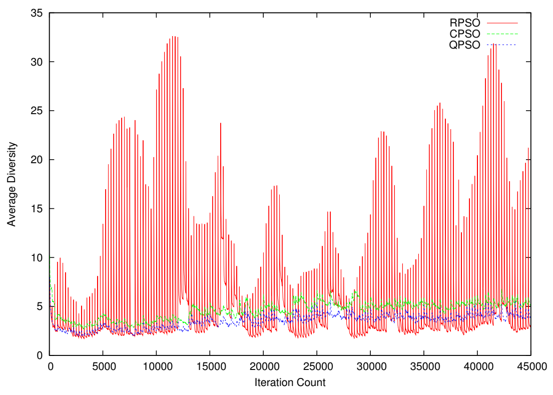


(d) E_G for D4

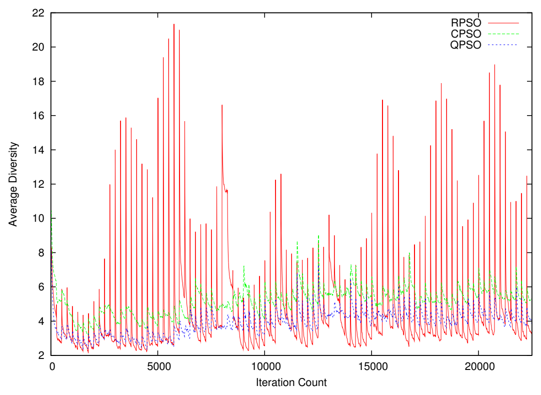
Figure 6.34: Training and Generalisation Error results for Moving Hyperplane, scenarios D3 to D4

QPSO have again exploited too much between environment changes. The RPSO did not deteriorate under D1 and D2 due to weaker dependence on the swarm memory: Figure 6.35 illustrates that the RPSO's diversity significantly increased after every environment change, and the RPSO had enough time to converge to a small area before the next environment change.

Figures 6.33 and 6.34 illustrate that BP and RBP did not visibly deteriorate in performance under D1 to D4, compared to C1 to C4. Figures 6.34(a) and 6.34(b) once again illustrate the inability of RBP to effectively train and generalise under scenarios



(a) Average Diversity Results for Scenario D1



(b) Average Diversity Results for Scenario D2

Figure 6.35: Average Diversity results for Moving Hyperplane, scenarios D1 to D2

with conflicting decision boundaries: RBP's E_T and E_G increased when the sliding window shifted by 500 patterns and replaced 50% of the sliding window contents. The RBP's error kept increasing until the window shifted again, replacing the other 50% of the window and thus discarding the remains of the stale data. Once the stale data was completely discarded, RBP's E_T and E_G began to decrease again.

Table 6.27 summarises the collective mean fitness results obtained for scenarios D1 to D4, as well as the average ρ_F for all algorithms considered, and average swarm diversity for the dynamic PSOs. Table 6.27 shows that under the most gradual scenario D1, all algorithms obtained $\rho_F > 1$. Thus, all algorithms exhibited overfitting. The exhibited overfitting behaviour is attributed to the low frequency of change of 250 iterations: since the gradual changes were easy to adjust to, the 250 iterations spent on NN training caused the algorithms to overfit the training data. The same applies to the abrupt scenario D4, under which all training algorithms again obtained $\rho_F > 1$: although changes were abrupt in case of D4, no conflicting boundaries were present, and only one decision boundary had to be approximated. Thus, the learning task was too simple for the given frequency of change.

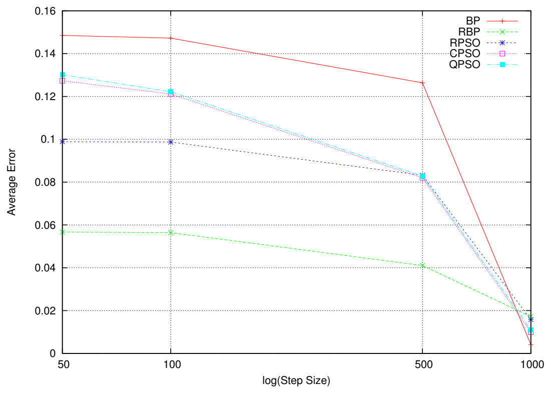
The algorithms were ranked based on the collective mean E_T and E_G values, taking

Table 6.27: Moving Hyperplane Results for Scenarios D1 to D4

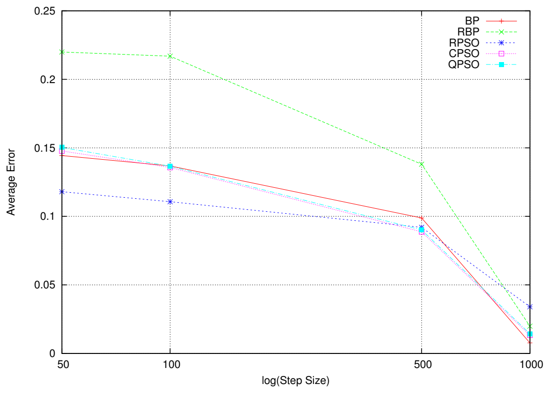
Scenario / **Algorithm**	**D1** *(frequency: 250, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.148513 ± 0.00199	0.144373 ± 0.007129	1.0558 ± 0.089037	n/a
Reinit. Back Propagation	0.056655 ± 0.00071	0.219934 ± 0.005605	3.90567 ± 0.184368	n/a
Reinitialising PSO	0.098873 ± 0.001553	0.118008 ± 0.004376	1.21006 ± 0.073892	4.43303 ± 0.799926
Charged PSO	0.127313 ± 0.002895	0.14769 ± 0.007511	1.18746 ± 0.08604	4.61526 ± 0.443067
Quantum PSO	0.130179 ± 0.003944	0.150337 ± 0.007289	1.18853 ± 0.069774	3.52732 ± 0.501969
	D2 *(frequency: 250, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.14725 ± 0.002158	0.136828 ± 0.006315	0.997445 ± 0.055028	n/a
Reinit. Back Propagation	0.056406 ± 0.00071	0.216878 ± 0.006479	3.83339 ± 0.212319	n/a
Reinitialising PSO	0.098709 ± 0.001347	0.110697 ± 0.004437	1.13458 ± 0.05672	4.15562 ± 0.481489
Charged PSO	0.121175 ± 0.004601	0.135658 ± 0.006594	1.13211 ± 0.059404	5.17929 ± 0.293442
Quantum PSO	0.122189 ± 0.004256	0.136576 ± 0.007123	1.14017 ± 0.055511	3.91117 ± 0.354609
	D3 *(frequency: 250, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.126362 ± 0.002513	0.098811 ± 0.003342	0.816696 ± 0.053073	n/a
Reinit. Back Propagation	0.041062 ± 0.000895	0.138068 ± 0.004618	3.0329 ± 0.164791	n/a
Reinitialising PSO	0.083174 ± 0.001437	0.091993 ± 0.009105	1.0946 ± 0.088386	4.18299 ± 0.329163
Charged PSO	0.082042 ± 0.004411	0.088738 ± 0.00514	1.07671 ± 0.053413	6.3934 ± 0.839511
Quantum PSO	0.082775 ± 0.002679	0.090142 ± 0.004242	1.07708 ± 0.076085	4.59905 ± 0.300309
	D4 *(frequency: 250, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.004087 ± 0.000332	0.007678 ± 0.001513	1.46905 ± 0.291061	n/a
Reinit. Back Propagation	0.017219 ± 0.001044	0.019877 ± 0.003026	1.18268 ± 0.154054	n/a
Reinitialising PSO	0.015818 ± 0.002082	0.034009 ± 0.028019	1.62148 ± 0.8127	4.04389 ± 0.354539
Charged PSO	0.00995 ± 0.001949	0.013311 ± 0.002892	1.23384 ± 0.202482	5.98838 ± 0.531167
Quantum PSO	0.011061 ± 0.001863	0.014333 ± 0.002704	1.20976 ± 0.184695	4.44401 ± 0.488091

Table 6.28: Moving Hyperplane Algorithm Ranking for Scenarios D1 to D4

Algorithm	D1		D2		D3		D4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	3	5	3	5	4	1	1	4	2.75
RBP	1	5	1	5	1	5	5	4.5	2	4.875
RPSO	2	1	3	1	3	2	4	4.5	3	2.125
CPSO	3	3	3	3	3	2	2	2.5	2.75	2.625
QPSO	4	3	3	3	3	2	3	2.5	3.25	2.625



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.36: Average Error results for Moving Hyperplane, scenarios D1 to D4

the p-values in Tables 6.20 and 6.21 into account. The obtained ranks are reported in Table 6.28. Collective mean E_T and E_G values obtained by the algorithms under D1 to D4 are illustrated in Figure 6.36. The same trends as observed for C1 to C4 are seen here: RBP performed significantly better than all other algorithms in terms of E_T under D1 to D3, and significantly worse than all other algorithms in terms of E_G under the same scenarios (refer to Tables 6.20 and 6.21 for supporting p-values). BP performed significantly worse than all other algorithms in terms of E_T under D1 to D3. No statistically significant difference was observed between E_G values produced by the CPSO, the QPSO, and BP under D1 and D2, but all three dynamic PSOs produced significantly lower E_G under D3. The RPSO, with its weak dependence on memory,

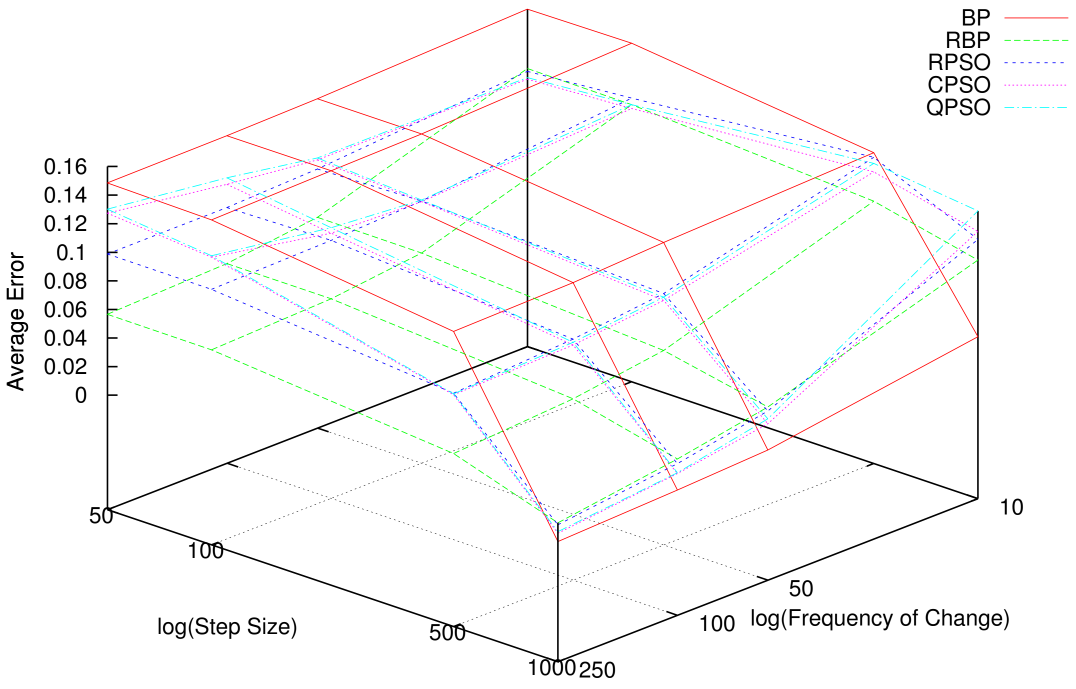


Figure 6.37: Average Training Error Results for Moving Hyperplane

significantly outperformed the other dynamic PSOs and BP under gradual scenarios D1 and D2 in terms of both E_T and E_G. No statistically significant difference was observed between the CPSO and the QPSO under most scenarios. Under the abrupt scenario D4, BP outperformed all other algorithms in terms of both training and generalisation. The average ranks in Table 6.28 indicate that the RPSO obtained the highest average E_G rank. Thus, the RPSO can be considered as the best performer for scenarios D1 to D4.

All scenarios: The average E_T and E_G values produced by the five algorithms under the 16 different dynamic scenarios are illustrated in Figures 6.37 and 6.38, respectively. Figure 6.37 illustrates that BP produced the highest E_T under all scenarios of low ($S = 50$) to moderately high ($S = 500$) spatial severity. Under scenarios of high spatial

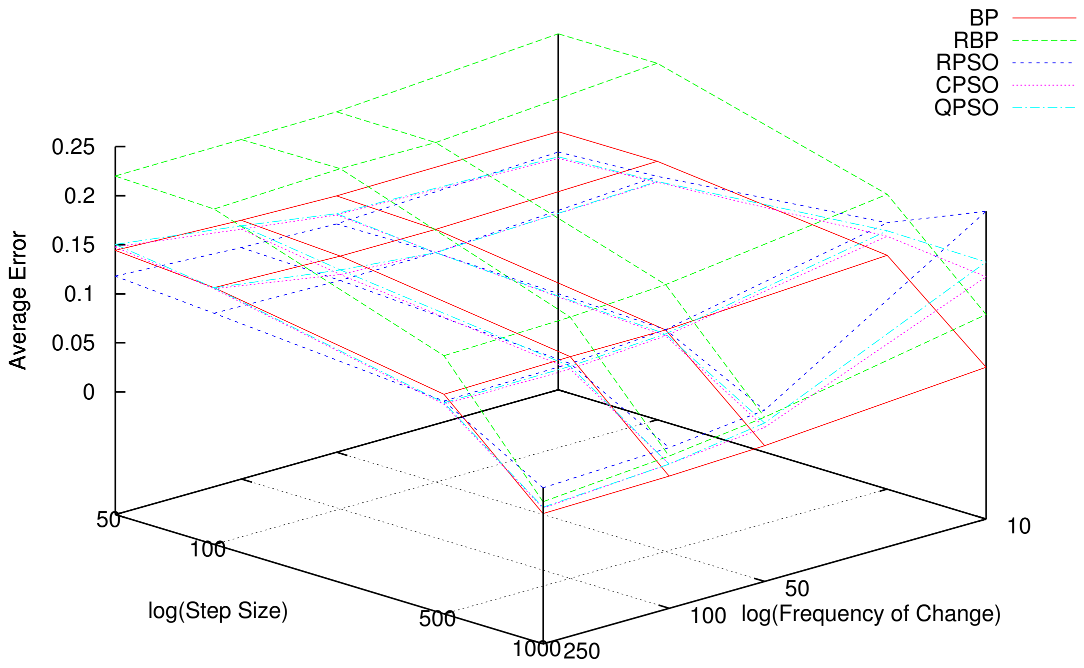


Figure 6.38: Average Generalisation Error Results for Moving Hyperplane

severity, BP consistently produced the lowest E_T. Since scenarios of high spatial severity ($S = 1000$) were the only scenarios where no conflicting boundaries were present, it can be concluded that BP exhibited sensitivity to the presence of conflicting boundaries within the training data. Figures 6.37 and 6.38 illustrate that changes in temporal severity (i.e. frequency of changes) had little effect on BP: BP is a fast-converging hill-climber, thus BP converged on a solution in a small number of epochs, and only slightly refined that solution as the frequency of changes decreased.

RBP, on the other hand, was restarted after every environment change, thus RBP required more epochs to converge on an optimal solution than BP. Figure 6.37 illustrates that RBP's average E_T decreased with the decrease of temporal severity. However, Figure 6.38 illustrates that the low E_T obtained by RBP was to no avail for most scenarios,

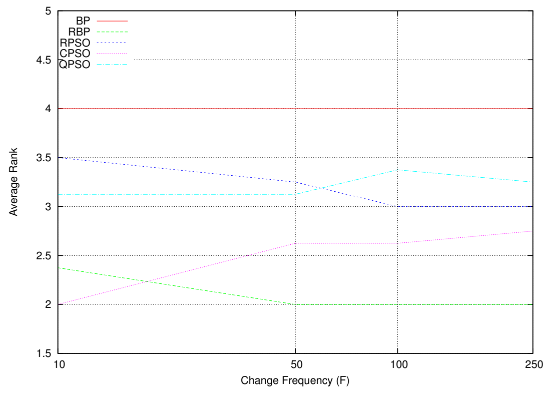
Table 6.29: Moving Hyperplane Average Algorithm Ranking for Scenarios A to D

Algorithm	Average R(A)		Average R(B)		Average R(C)		Average R(D)		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	2.5	4	3.125	4	3.25	4	2.75	4	2.90625
RBP	2.375	4.25	2	4.875	2	4.875	2	4.875	2.09375	4.71875
RPSO	3.5	3.75	3.25	2.75	3	2.375	3	2.125	3.1875	2.75
CPSO	2	2	2.625	1.875	2.625	1.875	2.75	2.625	2.5	2.09375
QPSO	3.125	2.5	3.125	2.375	3.375	2.625	3.25	2.625	3.21875	2.53125

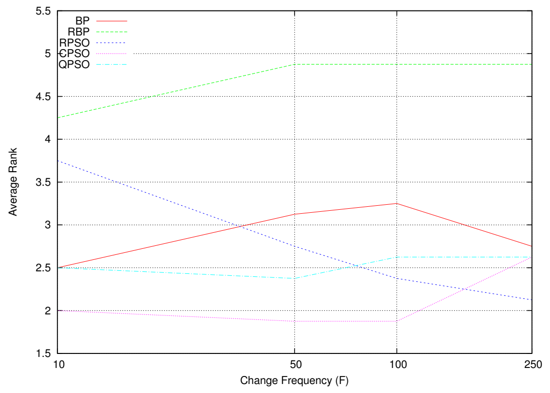
since the corresponding E_G was significantly higher than the E_G produced by all other algorithms under scenarios of low to moderately high spatial severity. RBP recovered only under abrupt scenarios with no conflicting boundaries, where RBP managed to outperform some of the dynamic PSOs in terms of E_G, especially under the most temporally severe scenario, A4 ($F = 10$). Thus, RBP also proved unsuccessful under scenarios where conflicting boundaries may have been present.

The dynamic PSOs showed the strongest resistance to conflicting boundaries, outperforming BP in terms of E_T under scenarios of low to moderately high spatial severity, and outperforming both BP and RBP in terms of E_G under the same scenarios. However, as Figures 6.37 and 6.38 illustrate, the dynamic PSOs were sensitive to temporal severity: the dynamic PSOs with their slow convergence speed failed to adjust to abrupt changes under high temporal severity (scenario A4). The CPSO and the QPSO also deteriorated in performance under gradual scenarios of low temporal severity, since training for too long under gradual changes caused these algorithms to over-exploit. The RPSO, however, proved more resistant to over-exploitation than the other two dynamic PSOs due to RPSO's weak dependence on swarm memory: the RPSO outperformed both the CPSO and the QPSO under gradual to moderately gradual scenarios of low to moderately low temporal severity.

Table 6.29 lists average algorithm ranks calculated over scenarios A to D, and Figure 6.39 illustrates how the average ranks changed over different frequencies of changes.



(a) Average Rank in terms of E_T



(b) Average Rank in terms of E_G

Figure 6.39: Average Rank results for Moving Hyperplane

Figure 6.39 confirms the already made observation that BP was little affected by changes in temporal severity and came last in terms of training. Figure 6.39 also illustrates that RBP's training rank improved as the temporal severity decreased, but at the same time RBP obtained the lowest E_G rank over all frequencies considered. Figure 6.39 illustrates that RPSO's average E_T and E_G rank increased as the temporal severity decreased from 10 to 250, while the CPSO's and the QPSO's average rank decreased as the change frequency became lower. Table 6.29 shows that on average, the CPSO obtained the second-best E_T rank after RBP, and the highest E_G rank. Thus, the CPSO can be considered as the most successful algorithm on the moving hyperplane problem. However, as Figure 6.39 illustrates, the choice of the best training algorithm remains specific to the characteristics of the given dynamic environment, such as the frequency and abruptness of changes, as well as the possibility of encountering conflicting decision boundaries (i.e. obsolete patterns) in the training data.

The next section presents an empirical analysis of the five algorithms applied to the dynamic sphere problem.

6.2.3 Dynamic Sphere

This dynamic classification problem was obtained by generating a hypersphere in $\mathbb{R}^3$ and using it to divide the space into two mutually exclusive classes.

Problem Definition

The hypersphere is given by

$$\sum_{l=1}^{n}(x_l + b_l) = R^2, \tag{6.5}$$

where R is the radius of the sphere, and $\vec{b}$ is the centre of the sphere. For the purpose of this study, n was set to 3. A 3-dimensional point outside the hypesphere is labelled as class A, and a 3-dimensional point inside the hypersphere is labelled as class B. A data set was generated according to the procedure described in Section 6.1.2, where the number of data points, M, was set to 1000, and the number of environment changes, N, was set to 10. A set of M 3-dimensional points, $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{1000}\}$ was randomly generated such that $\vec{x}_l \in [0, 1]^3,\ l = 1, \ldots, 1000$. The sphere was generated N times by randomising its centre point, $\vec{b} \in [0, 1]^3$, and radius, $R \in [0, 1]$. Target classification of M patterns was updated N times, and every time the updated patterns were appended to the data set, yielding a data set of 11 000 patterns in total. The size of the sliding window was set to 1000.

A NN with 3 input units, 4 hidden units and 1 output unit was trained on the Dynamic Sphere problem. The total number of weights and biases, corresponding to the dimensionality of the problem, was equal to 21, according to Equation (6.3).

Sixteen different dynamic scenarios as described in Section 6.2.2 were applied to the dynamic sphere problem. The chosen scenarios simulated different combinations of spatial and temporal severity, providing representative coverage of different dynamic environment types. The technique of simulating dynamic scenarios applied in this study is described in Section 6.1.2. Parameter settings corresponding to each dynamic scenario

Table 6.30: Optimal Parameters for the Dynamic Sphere problem

Algorithm	**Parameters**			
Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-3, 3]$	0.1	0.3	
Reinitialising Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-3, 3]$	0.1	0.3	
Reinitialising PSO	Weight Interval	V_{max}	Swarm Size	Reinitialisation Ratio
	$[-1, 1]$	1	50	0.5
Charged PSO	Weight Interval	V_{max}	Swarm Size	Charge Magnitude
	$[-5, 5]$	0.1	50	5
Quantum PSO	Weight Interval	V_{max}	Swarm Size	Cloud Radius
	$[-1, 1]$	0.1	20	1.5

are listed in Table 6.17.

Parameter Optimisation

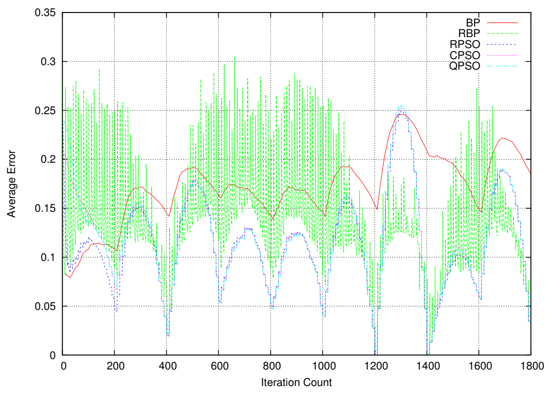
All algorithm parameters were optimised according to the procedure described in Section 6.1.3. Corresponding optimal parameters discovered are summarised in Table 6.30.

Analysis of Empirical Data

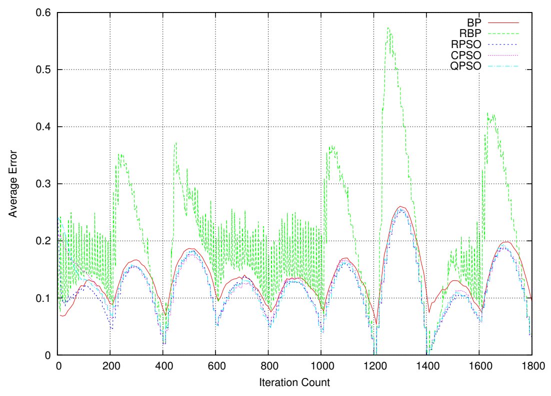
The number of iterations required to traverse the entire data set under every dynamic scenario considered was calculated using equation (6.4).

The two-tailed Mann-Whitney U statistical test was used to determine whether the difference between algorithm performance was of any statistical significance. P-values corresponding to training error comparisons between the algorithms are reported in Table

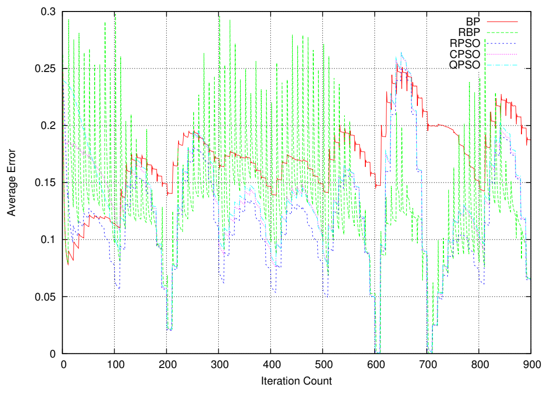
6.32. P-values corresponding to generalisation error comparisons between the algorithms are reported in Table 6.33.



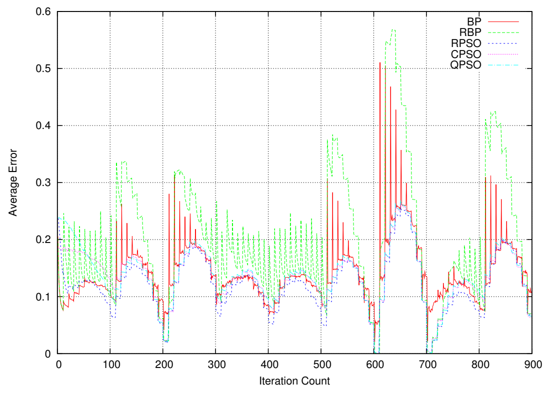
(a) E_T for A1



(b) E_G for A1



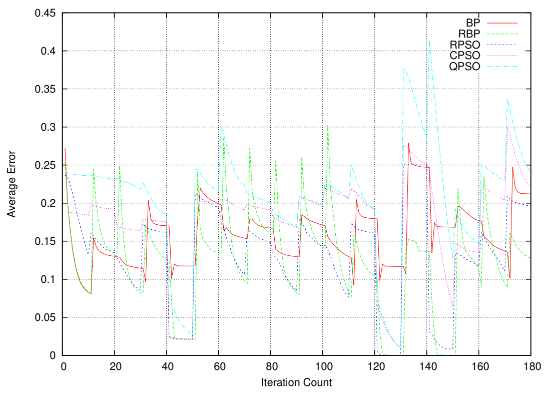
(c) E_T for A2



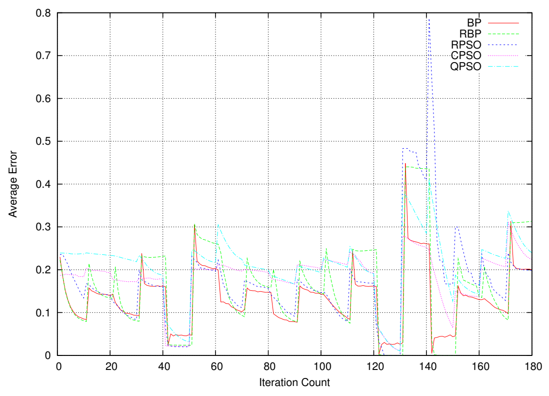
(d) E_G for A2

Figure 6.40: Training and Generalisation Error results for Dynamic Sphere, scenarios A1 to A2

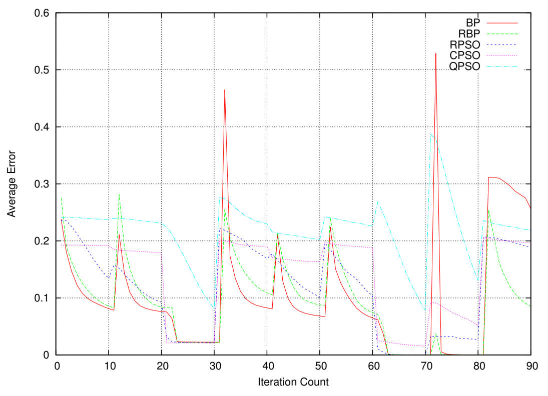
Scenarios A1 to A4: Figures 6.40 and 6.41 illustrate the progression of E_T and E_G over time as obtained by the algorithms under scenarios A1 to A4. Figure 6.40 indicates that the dynamic PSOs took longer than BP and RBP to find a solution in the beginning of the algorithm run under gradual scenarios A1 and A2, but tracked the changes better and exploited more. Figures 6.40 and 6.41 illustrate that RBP did not always manage



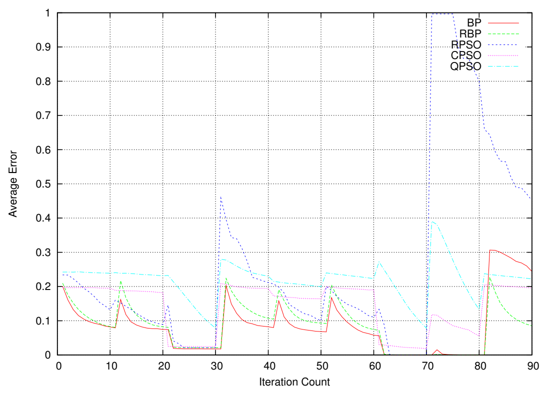
(a) E_T for A3



(b) E_G for A3



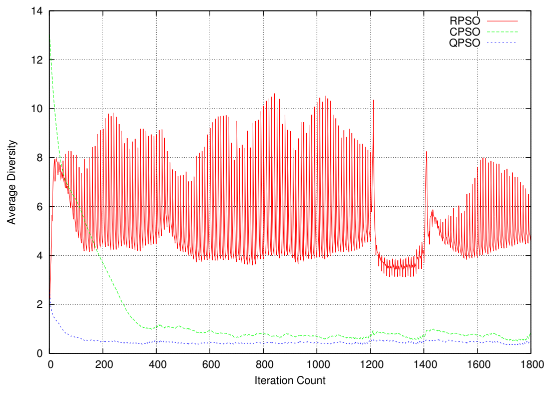
(c) E_T for A4



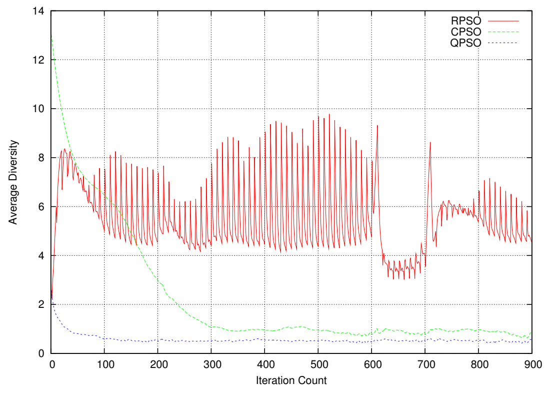
(d) E_G for A4

Figure 6.41: Training and Generalisation Error results for Dynamic Sphere, scenarios A3 to A4

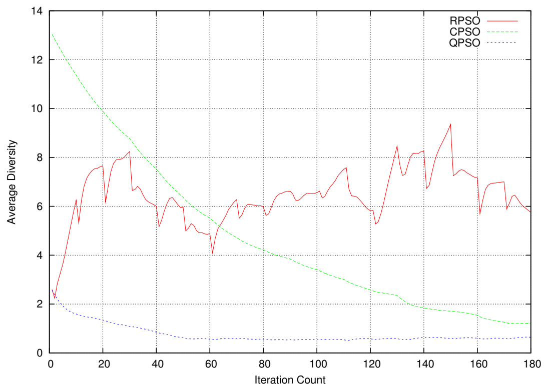
to reach the same minimum E_T as the dynamic PSOs did, but reached lower minimum E_T than BP under scenarios A1 to A3. Scenarios A1 to A4 simulated frequent changes (every 10 iterations), and the fact that an algorithm which was completely reinitialised after every change still managed to obtain a low training error indicates that the dynamic sphere was a rather trivial classification problem. Figures 6.40 and 6.41 also show that RBP failed to support a low E_T by an equally low E_G under scenarios A1 to A3, but generalised well under A4. A4 was the only scenario where no conflicting boundaries were present. Hence, RBP was sensitive to the presence of conflicting boundaries, and



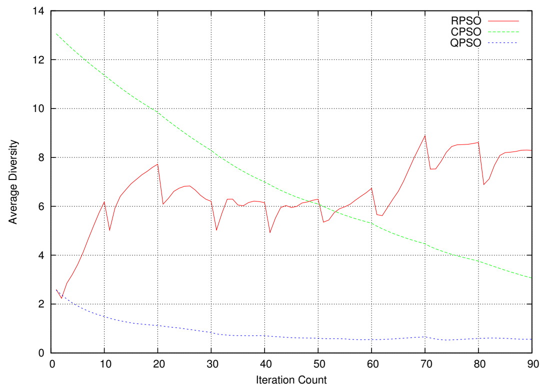
(a) Average Diversity Results for Scenario A1



(b) Average Diversity Results for Scenario A2



(c) Average Diversity Results for Scenario A3



(d) Average Diversity Results for Scenario A4

Figure 6.42: Average Diversity results for Dynamic Sphere, scenarios A1 to A4

failed to discern between stale and new data when both were present inside the sliding window.

Figure 6.41 illustrates that the CPSO and the QPSO did not manage to obtain a good minimum E_T and E_G under the abrupt scenarios A3 and A4: as shown in Table 6.30, both these algorithms were applied with a rather small V_{max} value, which prevented the particles from developing large velocities and moving through the search space fast. Limited velocity prevented the CPSO and the QPSO from locating a good solution in as little as 10 iterations. Such a small V_{max} value was chosen because it proved beneficial under most scenarios, as discussed later in this section.

Even under scenario A4, the benefits of smaller velocity can already be observed: figures 6.41(c) and 6.41(d) illustrate that, after iteration 70, the RPSO's E_G peaked much higher than the corresponding CPSO's and the QPSO's E_G. The RPSO used a larger V_{max}, thus the RPSO explored more than the other dynamic PSOs, and obtained an E_T and E_G close to 0 before iteration 70 under scenario A4. However, rapid exploration on an easy dynamic classification problem led the RPSO to an area on the NN error landscape where the RPSO became stuck after an environment change, and failed to recover despite the fact that the RPSO maintained a higher diversity than both the CPSO and the QPSO under scenarios A1 to A4, as Figure 6.42 illustrates.

Figure 6.42 also illustrates that the CPSO and the QPSO maintained a rather low swarm diversity under gradual scenarios. The success of these algorithms under A1 and A2 confirms that the dynamic sphere was not hard to optimise despite the small swarm diversity.

Table 6.31 lists the collective mean E_T and E_G values obtained by the algorithms under A1 to A4, along with collective mean swarm diversity where applicable, and collective mean ρ_F values. The ρ_F values in Table 6.31 indicate that the dynamic PSOs were susceptible to minor overfitting under all scenarios considered. RBP was also subject to overfitting and obtained the largest ρ_F under scenarios A1 to A3, but did not overfit under A4, thus confirming that RBP is sensitive to the presence of stale data. BP produced a $\rho_F < 1$ under scenarios A1 to A4.

The algorithms were ranked in terms of collective mean E_T and E_G values obtained under A1 to A4, taking the p-values reported in Tables 6.32 and 6.33 into account. The resulting ranks, along with the average ranks, are listed in Table 6.34. Collective mean E_T and E_G values obtained by the five algorithms under A1 to A4 are also illustrated in Figure 6.43.

Table 6.34 shows that the RPSO was the top performer in terms of both E_T and E_G under the gradual scenarios A1 and A2: all dynamic PSOs exploited better than

Table 6.31: Dynamic Sphere Results for Scenarios A1 to A4

Algorithm \ **Scenario**	**A1** *(frequency: 10, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.171713 ± 0.00311	0.135771 ± 0.004825	0.8671 ± 0.04248	n/a
Reinit. Back Propagation	0.131876 ± 0.002916	0.199343 ± 0.005105	1.30891 ± 0.051824	n/a
Reinitialising PSO	0.113942 ± 0.001718	0.115899 ± 0.003382	1.02229 ± 0.048443	5.31603 ± 0.214277
Charged PSO	0.119896 ± 0.002418	0.121289 ± 0.00364	1.00997 ± 0.041236	1.53772 ± 0.447136
Quantum PSO	0.120057 ± 0.00155	0.120964 ± 0.003453	1.00464 ± 0.041862	0.489555 ± 0.040246
	A2 *(frequency: 10, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.172275 ± 0.002906	0.140283 ± 0.004329	0.890718 ± 0.048617	n/a
Reinit. Back Propagation	0.130808 ± 0.003701	0.198831 ± 0.005248	1.29163 ± 0.053045	n/a
Reinitialising PSO	0.11798 ± 0.001374	0.120919 ± 0.003699	1.02911 ± 0.058348	5.46459 ± 0.322592
Charged PSO	0.133813 ± 0.005329	0.13525 ± 0.004943	1.01271 ± 0.041638	2.27272 ± 0.344643
Quantum PSO	0.136196 ± 0.004216	0.137025 ± 0.006435	1.00044 ± 0.034118	0.570012 ± 0.062034
	A3 *(frequency: 10, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.159435 ± 0.004758	0.132337 ± 0.007327	0.887263 ± 0.049432	n/a
Reinit. Back Propagation	0.122616 ± 0.008421	0.161899 ± 0.00782	1.13254 ± 0.042452	n/a
Reinitialising PSO	0.13249 ± 0.004636	0.172753 ± 0.0356	1.09221 ± 0.05367	6.45868 ± 0.391706
Charged PSO	0.176709 ± 0.004776	0.179689 ± 0.004688	1.01718 ± 0.048702	4.77798 ± 0.384953
Quantum PSO	0.20885 ± 0.006293	0.208684 ± 0.008191	1.00468 ± 0.017026	0.767477 ± 0.071615
	A4 *(frequency: 10, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.076418 ± 0.009425	0.076587 ± 0.010715	0.980779 ± 0.106462	n/a
Reinit. Back Propagation	0.093061 ± 0.010648	0.08518 ± 0.01001	0.904017 ± 0.051538	n/a
Reinitialising PSO	0.115917 ± 0.006892	0.270092 ± 0.041104	1.31458 ± 0.109629	6.49894 ± 0.385715
Charged PSO	0.137869 ± 0.009361	0.142018 ± 0.010995	1.02441 ± 0.049801	7.04183 ± 0.216855
Quantum PSO	0.21962 ± 0.004297	0.220084 ± 0.005145	1.00202 ± 0.010966	0.868152 ± 0.093542

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

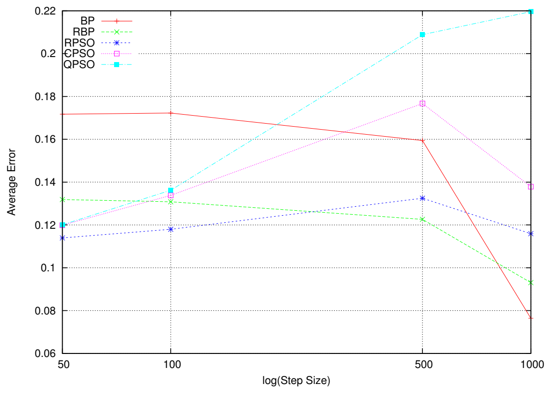
	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.001092**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.000511**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.901142
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.130369	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.013354**	**0.0001**	**0.0001**
B	0.099517	**0.000258**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.786452	**0.04622**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.07719	0.809129	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.901142	0.809129	**0.0001**	**0.028468**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.912745	0.612688	**0.014243**	0.090617

Table 6.33: Mann-Whitney U p-values obtained for the average generalisation error comparisons on the Dynamic Sphere problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

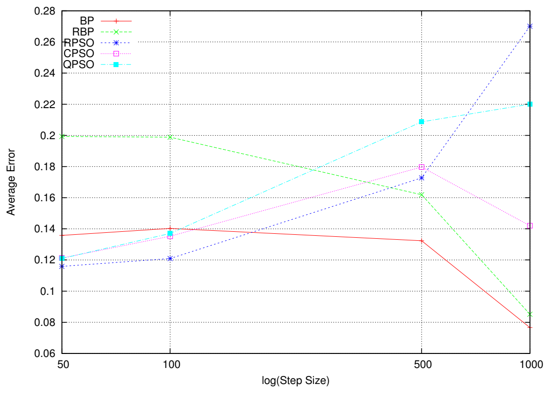
	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.001032**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.000774**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.012515**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	0.571977	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.292726	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.000153**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.030748**	**0.0001**	**0.001998**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.513407	0.423131	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.177408	**0.035773**	**0.0001**	**0.0001**
C	0.119364	**0.0001**	**0.0001**	**0.000134**	0.096477	0.831957	**0.000543**	**0.018329**
D	**0.0001**	**0.0001**	**0.0001**	**0.000774**	0.542302	0.970925	0.099517	0.467073

Table 6.34: Dynamic Sphere Algorithm Ranking for Scenarios A1 to A4

Algorithm	A1		A2		A3		A4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	3	1	1	1	3.5	2.5
RBP	4	5	2	5	1	2.5	2	2	2.25	3.625
RPSO	1	1	1	1	2	2.5	3	5	1.75	2.375
CPSO	2.5	2.5	3	2.5	4	4	4	3	3.375	3
QPSO	2.5	2.5	4	2.5	5	5	5	4	4.125	3.5



(a) Average Training Error Results

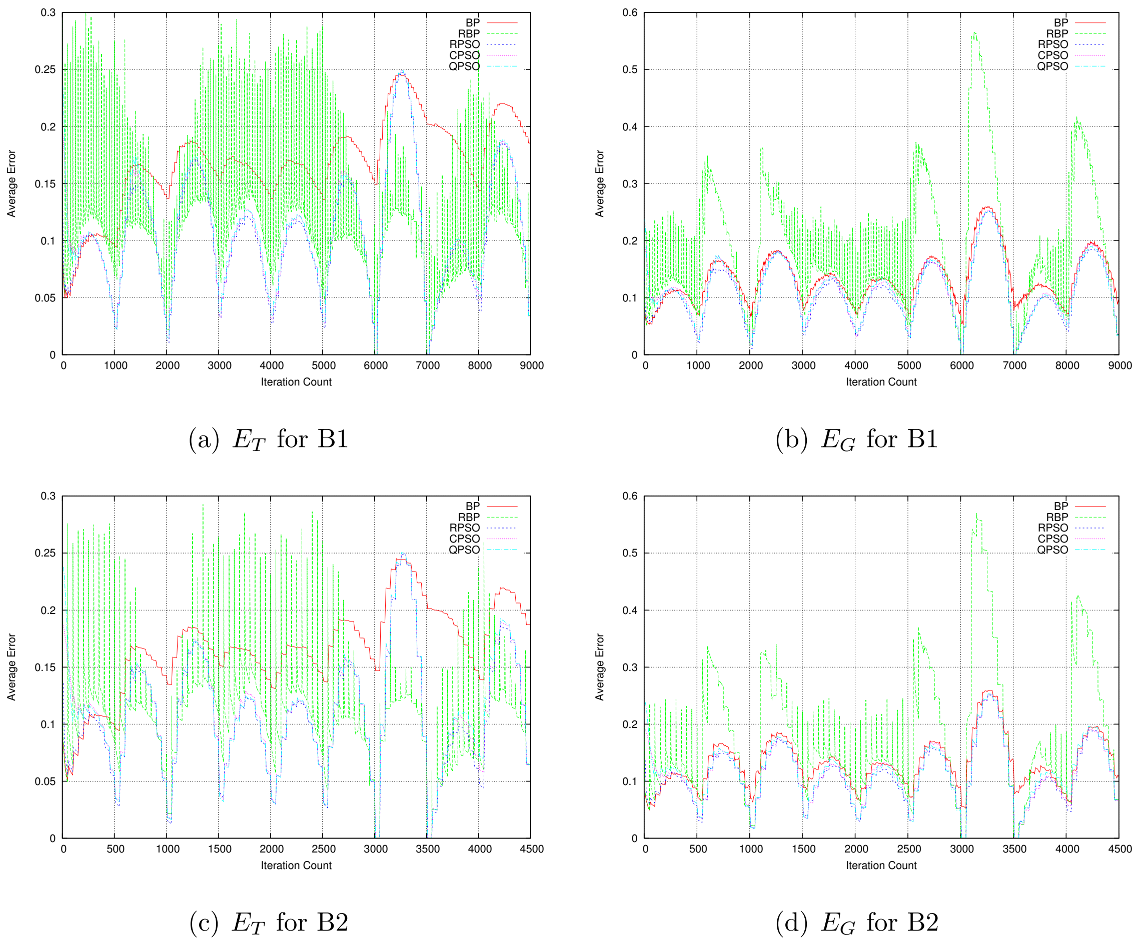


(b) Average Generalisation Error Results

Figure 6.43: Average Error results for Dynamic Sphere, scenarios A1 to A4

hill-climbing BP and RBP under A1 and A2, and the RPSO used a larger V_{max}, which enabled the RPSO to find a good solution faster than the other two dynamic PSOs. The CPSO and the QPSO showed second-best performance in terms of E_G under A1 and A2, bearing no statistically significant difference between one another. However, both these algorithms deteriorated with an increase in spatial severity, and were significantly outperformed by BP and RBP under A3 and A4. The RPSO competed with the hill-climbers under A3, coming second in terms of both E_T and E_G rank, but significantly deteriorated under A4, obtaining the lowest E_G rank. As both Table 6.34 and Figure 6.43 illustrate, RBP exhibited similar trends as for the moving hyperplane problem: RBP trained better than BP, but generalised worse, due to strong sensitivity to stale data.

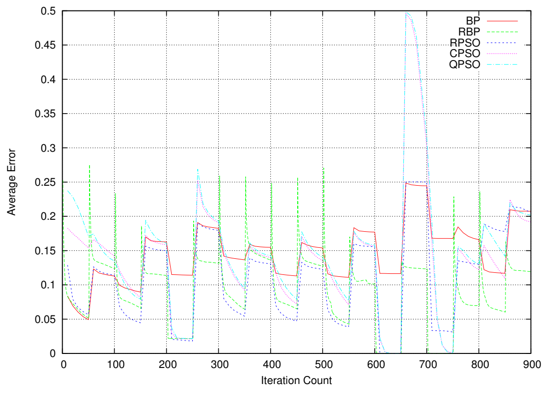
Thus, the dynamic PSOs significantly outperformed the hill climbers under gradual scenarios, but were outperformed under abrupt scenarios. The average ranks in Table 6.34 show that the RPSO obtained the highest average rank, resulting as the best performer under scenarios A1 to A4.



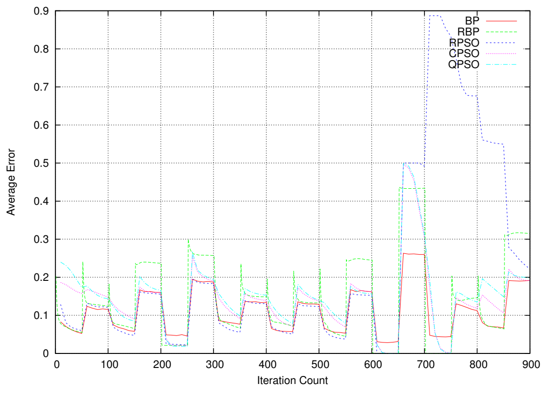
(a) E_T for B1 (b) E_G for B1 (c) E_T for B2 (d) E_G for B2

Figure 6.44: Training and Generalisation Error results for Dynamic Sphere, scenarios B1 to B2

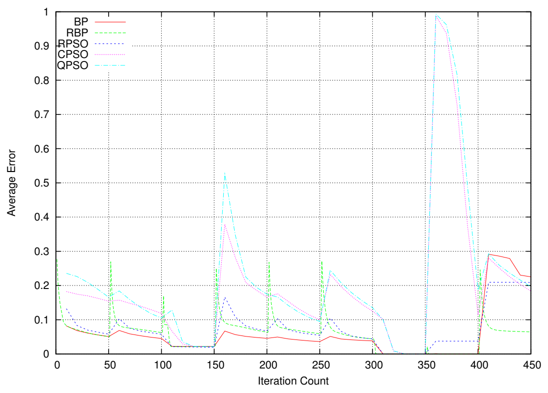
Scenarios B1 to B4: Figures 6.44 and 6.45 illustrate the E_T and E_G profiles obtained by the algorithms under scenarios B1 to B4. Figure 6.44 illustrates that the algorithms exhibited similar trends under B1 to B4 as under A1 to A4: the dynamic PSOs exploited



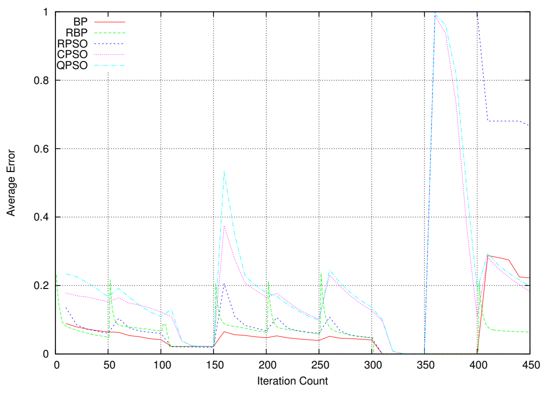
(a) E_T for B3



(b) E_G for B3



(c) E_T for B4

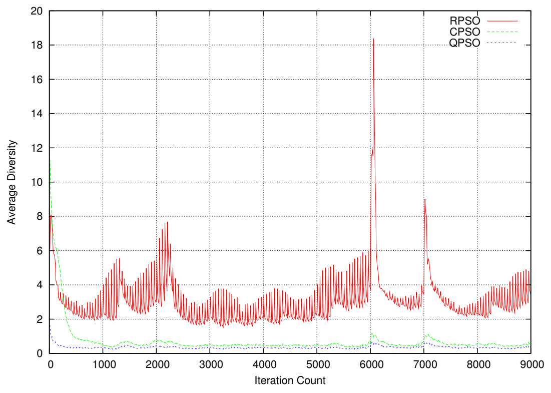


(d) E_G for B4

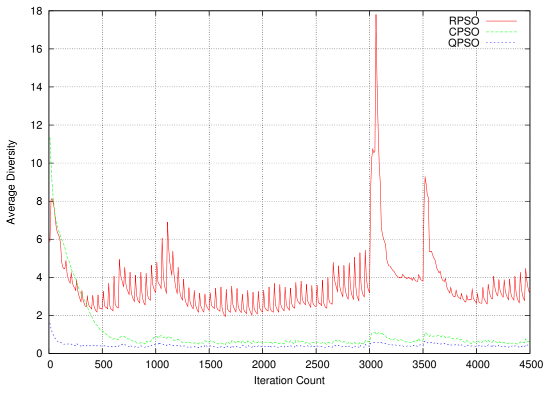
Figure 6.45: Training and Generalisation Error results for Dynamic Sphere, scenarios B3 to B4

better than both BP and RBP, and RBP trained well and generalised poorly. Figure 6.44 illustrates that RBP managed to reach the same or lower minimum E_T as that of the dynamic PSOs after most environment changes. Hence, 50 iterations (change frequency simulated under B1 to B4) was enough for RBP to converge on a minimum for the dynamic sphere problem.

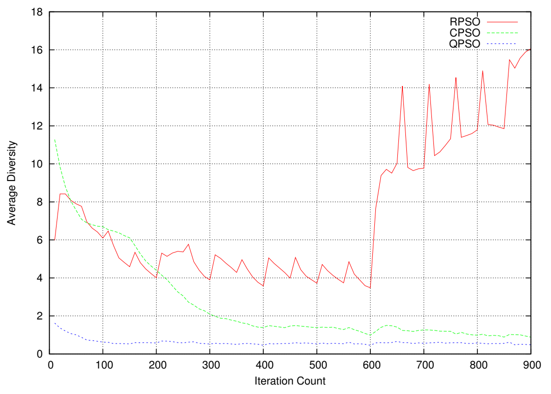
Figure 6.45 illustrates that the dynamic PSOs have improved in performance with a decrease in temporal severity: both E_T and E_G obtained by the dynamic PSOs decreased between environment changes. However, given more time to converge, all dynamic PSOs



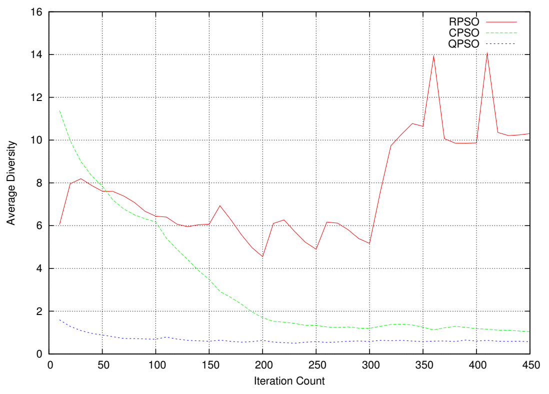
(a) Average Diversity Results for Scenario B1



(b) Average Diversity Results for Scenario B2



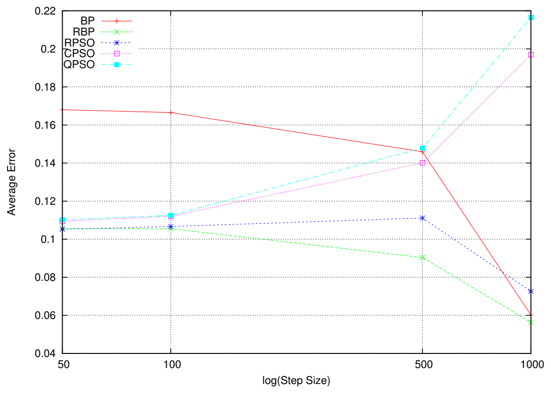
(c) Average Diversity Results for Scenario B3



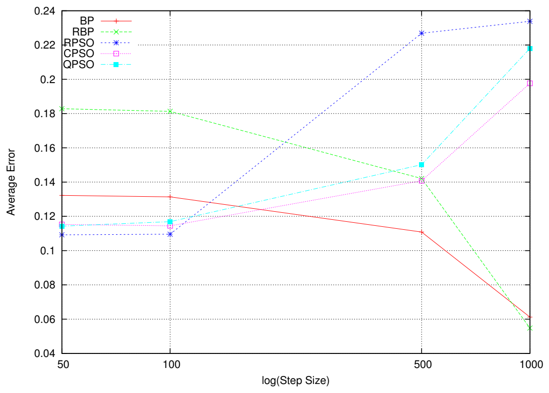
(d) Average Diversity Results for Scenario B4

Figure 6.46: Average Diversity results for Dynamic Sphere, scenarios B1 to B4

managed to obtain an $E_T \approx 0$ after a certain environment change (twelfth in case of B3, sixth in case of B4), producing a very high error after the next environment change. Figures 6.45(b) and 6.45(d) illustrate that the CPSO and the QPSO managed to recover in terms of E_G after around 50 iterations, but the RPSO stagnated and failed to generalise for the rest of the algorithm run. Swarm diversity profiles in Figure 6.46 illustrate that the RPSO indeed failed to converge to a small area after the high error peak at iteration 650 under B3 and iteration 350 under B4. RPSO's diversity peaked under gradual changes as well, but the RPSO managed to converge back to its stable diversity level under B1 and B2. Hence, over-exploitation (resulting in $E_T \approx 0$) exhibited



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.47: Average Error results for Dynamic Sphere, scenarios B1 to B4

by the dynamic PSOs under B3 and B4 may have led the swarms to an unfruitful search space region, such as a plateau or a local minimum, from which the swarms struggled to escape after the change in the error landscape.

Table 6.35 summarises the collective mean E_T, E_G, ρ_F, and swarm diversity values obtained by the algorithms under B1 to B4. The ρ_F values in Table 6.35 indicate that the dynamic PSOs have again exhibited overfitting under all scenarios considered. BP obtained a $\rho_F < 1$ under B1 to B3, but overfitted under B4, thus indicating that training BP for 50 epochs on the dynamic sphere problem was already training for too long. RBP again overfitted under B1 to B3, and produced a $\rho_F < 1$ under B4. The algorithms were ranked based on their collective mean E_T and E_G. RBP's rank in Table 6.36 indicates that RBP was indeed the top performer under the most abrupt scenario B4. Such success of a complete reinitialisation of NN weights points out the triviality of the problem, and indicates that trivial approaches (i.e. RBP) can indeed be more efficient than more sophisticated approaches (i.e. dynamic PSOs) when the problem to be optimised is simple enough. Application of sophisticated algorithms to trivial problems can indeed result in a waste of computational effort.

Table 6.36 lists algorithm ranks in terms of collective mean E_T and E_G values ob-

Table 6.35: Dynamic Sphere Results for Scenarios B1 to B4

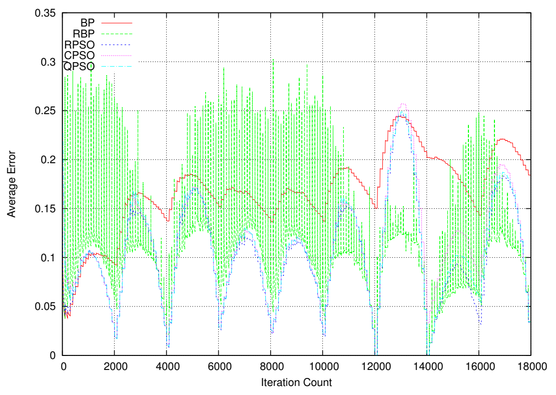
Algorithm \ **Scenario**	**B1** *(frequency: 50, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.168007 ± 0.002584	0.132217 ± 0.004245	0.876624 ± 0.052278	n/a
Reinit. Back Propagation	0.105521 ± 0.000871	0.182805 ± 0.00569	1.45788 ± 0.076276	n/a
Reinitialising PSO	0.105274 ± 0.001257	0.109222 ± 0.003762	1.0461 ± 0.062404	3.04782 ± 0.164552
Charged PSO	0.109405 ± 0.001918	0.115289 ± 0.00367	1.06513 ± 0.049522	0.728078 ± 0.133793
Quantum PSO	0.110216 ± 0.001549	0.114269 ± 0.003857	1.05057 ± 0.060397	0.352193 ± 0.015383
	B2 *(frequency: 50, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.166557 ± 0.002542	0.131353 ± 0.004144	0.869911 ± 0.054067	n/a
Reinit. Back Propagation	0.105463 ± 0.001317	0.181257 ± 0.005014	1.43268 ± 0.057606	n/a
Reinitialising PSO	0.106646 ± 0.001065	0.109564 ± 0.003063	1.02957 ± 0.053056	3.56361 ± 0.252406
Charged PSO	0.11202 ± 0.001841	0.114478 ± 0.004462	1.01971 ± 0.052804	1.06084 ± 0.234102
Quantum PSO	0.112497 ± 0.003126	0.116895 ± 0.004536	1.043 ± 0.054262	0.413402 ± 0.014286
	B3 *(frequency: 50, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.145928 ± 0.003437	0.11083 ± 0.005273	0.839866 ± 0.058265	n/a
Reinit. Back Propagation	0.090385 ± 0.002677	0.142071 ± 0.004415	1.27469 ± 0.069465	n/a
Reinitialising PSO	0.111102 ± 0.005848	0.226889 ± 0.04717	1.23073 ± 0.077915	7.32857 ± 1.11706
Charged PSO	0.140143 ± 0.00421	0.140779 ± 0.005707	1.01619 ± 0.042077	2.71457 ± 0.752959
Quantum PSO	0.147884 ± 0.006497	0.150163 ± 0.008159	1.02077 ± 0.035745	0.610085 ± 0.051794
	B4 *(frequency: 50, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.060173 ± 0.01075	0.061217 ± 0.012189	1.06281 ± 0.133312	n/a
Reinit. Back Propagation	0.056284 ± 0.004055	0.054775 ± 0.004384	0.967692 ± 0.077902	n/a
Reinitialising PSO	0.07264 ± 0.005138	0.233869 ± 0.009101	1.43268 ± 0.112701	7.69642 ± 0.961675
Charged PSO	0.196929 ± 0.013914	0.197626 ± 0.014396	1.0075 ± 0.039589	3.2151 ± 0.387966
Quantum PSO	0.216473 ± 0.013488	0.218016 ± 0.016347	1.01148 ± 0.036667	0.677642 ± 0.125423

Table 6.36: Dynamic Sphere Algorithm Ranking for Scenarios B1 to B4

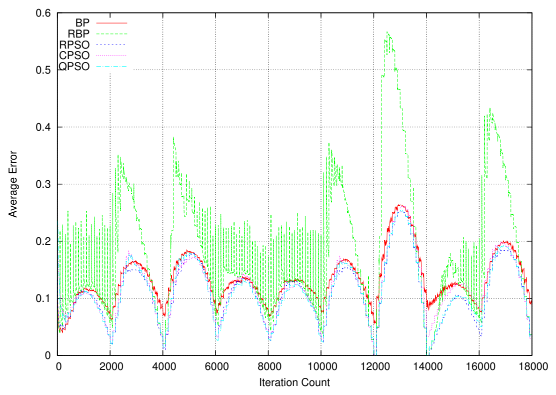
Algorithm	B1		B2		B3		B4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	4.5	1	2	2	4.125	2.75
RBP	1.5	5	1	5	1	2.5	1	1	1.125	3.375
RPSO	1.5	1	2	1	2	5	3	5	2.125	3
CPSO	3.5	2.5	3.5	2	3	2.5	4	3	3.5	2.5
QPSO	3.5	2.5	3.5	3	4.5	4	5	4	4.125	3.375

tained under B1 to B4. Collective mean E_T and E_G values for B1 to B4 are also visualised in Figure 6.47. As follows from Table 6.36 and Figure 6.47, the RPSO again showed top performance under gradual scenarios, and deteriorated in terms of E_G under the abrupt changes. The three dynamic PSOs outperformed BP and RBP under the gradual scenarios B1 and B2, and deteriorated under abrupt scenarios. Table 6.36 shows that the CPSO significantly outperformed the QPSO under abrupt changes: average diversity values in Table 6.35 show that the QPSO converged to a smaller area and thus explored less that the CPSO, resulting in poorer performance under abrupt changes, where exploration was important. BP outperformed all other algorithms under the abrupt scenario B3, and was outperformed by either the dynamic PSOs (B1 and B2) or RBP (B4) elsewhere. Average ranks in Table 6.36 indicate that RBP obtained the lowest average E_T rank, and the CPSO was the top performer in terms of E_G under B1 to B4.

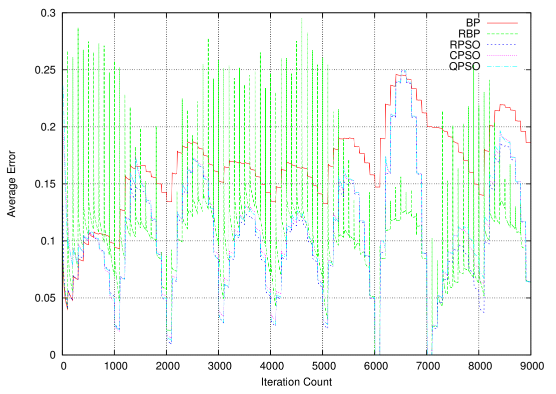
Scenarios C1 to C4: Figures 6.48 and 6.49 illustrate the progression of E_T and E_G over time as obtained by the algorithms under scenarios C1 to C4. Figure 6.48 illustrates that the decrease of change frequency from 50 iterations to 100 iterations did not have a drastic effect on the algorithm performance under gradual scenarios, and the algorithms exhibited the same trends under C1 to C2 as under B1 to B2. RBP obtained a lower minimum E_T than the other algorithms, indicating the triviality of the problem, and did not always generalise well, indicating that RBP is sensitive to stale data.



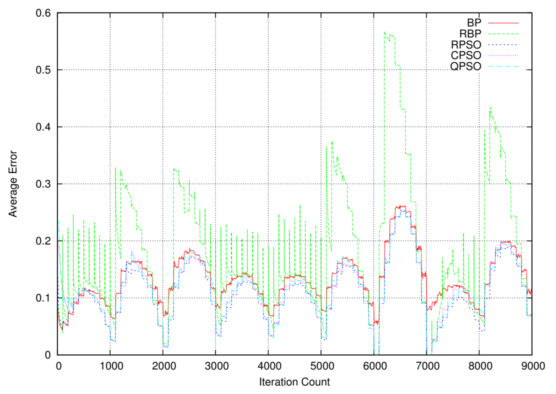
(a) E_T for C1



(b) E_G for C1



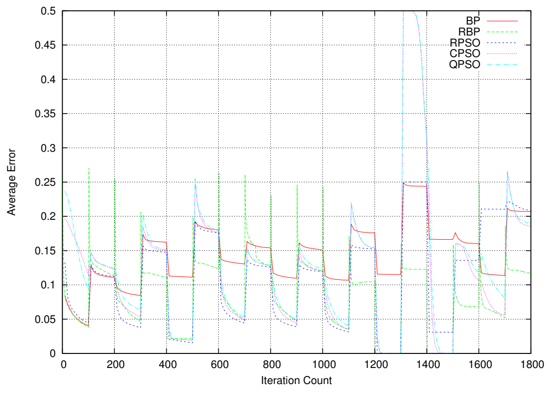
(c) E_T for C2



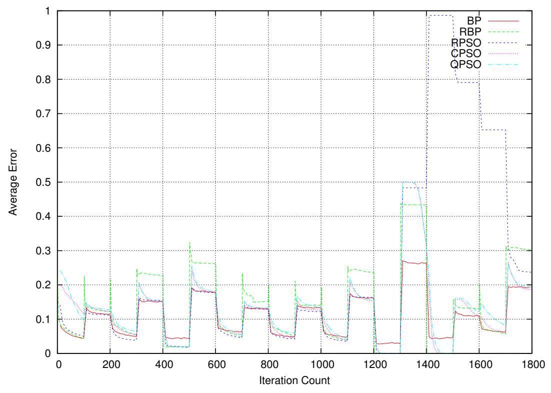
(d) E_G for C2

Figure 6.48: Training and Generalisation Error results for Dynamic Sphere, scenarios C1 to C2

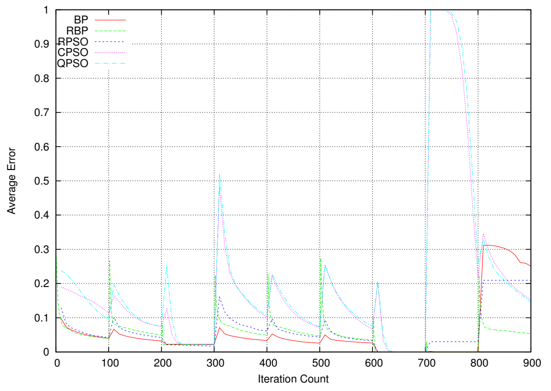
Figure 6.49 illustrates that the algorithms also exhibited similar trends under abrupt scenarios C3 and C4 as under B3 and B4: errors produced by the dynamic PSOs peaked when an environment change occurred after the dynamic PSOs obtained $E_T \approx 0$ and $E_G \approx 0$. Once again, the CPSO and the QPSO managed to recover after the sudden error increase, and the RPSO stagnated: the swarm diversity profiles in Figure 6.50 illustrate that the RPSO failed to converge to a small area after the error peaked under abrupt scenarios C3 and C4. Figures 6.49(c) and 6.49(c) illustrate that BP's errors also peaked closer to the end of the algorithm run, and the CPSO along with the QPSO



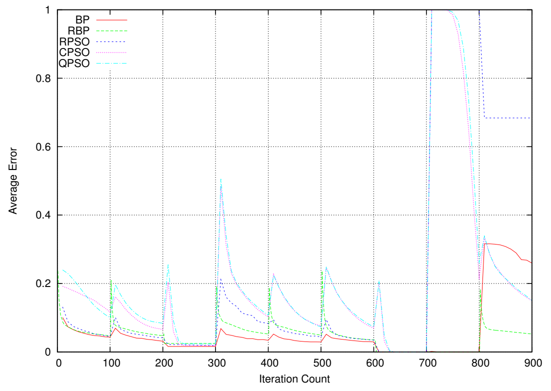
(a) E_T for C3



(b) E_G for C3



(c) E_T for C4

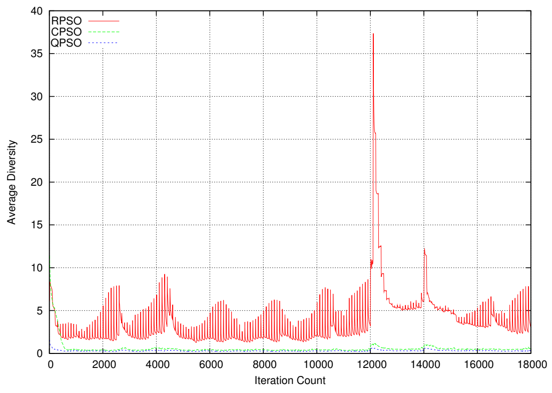


(d) E_G for C4

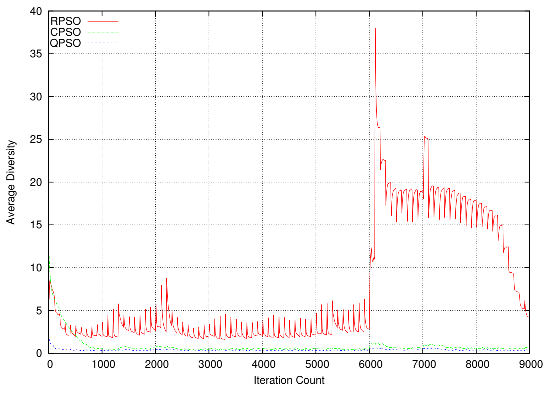
Figure 6.49: Training and Generalisation Error results for Dynamic Sphere, scenarios C3 to C4

visibly outperformed BP after the last environment change. RBP produced no error peaks under C4 due to complete independence from previously learned information.

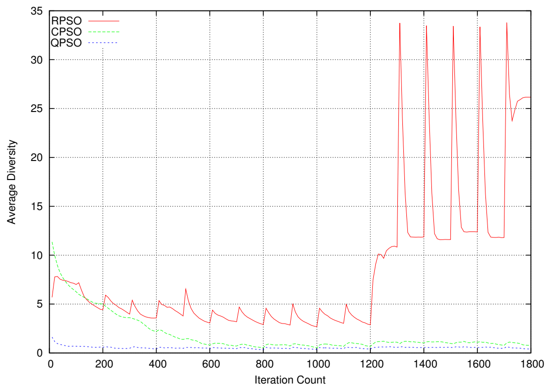
Table 6.37 lists the collective mean E_T, E_G, ρ_F, and swarm diversity values obtained under scenarios C1 to C4. The ρ_F values in Table 6.37 indicate that BP obtained a $\rho_F < 1$ under C1 to C3, but exhibited overfitting under C4. All other algorithms exhibited overfitting under C1 to C4, and even RBP overfitted under C4. Hence, training a completely reinitialised algorithm on the dynamic sphere problem for 100 iterations was already training for too long. If a completely reinitialised algorithm was subject to



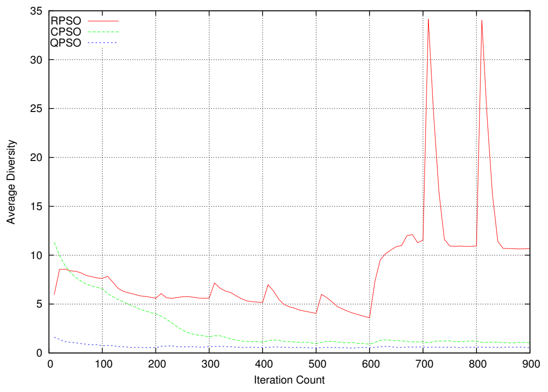
(a) Average Diversity Results for Scenario C1



(b) Average Diversity Results for Scenario C2



(c) Average Diversity Results for Scenario C3



(d) Average Diversity Results for Scenario C4

Figure 6.50: Average Diversity results for Dynamic Sphere, scenarios C1 to C4

overfitting after 100 iterations, it is quite natural that the dynamic PSOs with memory of previous solutions also over-exploit when allowed to train for 100 iterations between environment changes.

Average diversity values in Table 6.37 show that RPSO's average diversity increased under C1 to C4 compared to B1 to B4. Increases in swarm diversity as temporal severity decreases indicate divergent behaviour. The CPSO and the QPSO, on the other hand, obtained lower average diversity over C1 to C4 than under B1 to B4. Therefore, these algorithms did not diverge.

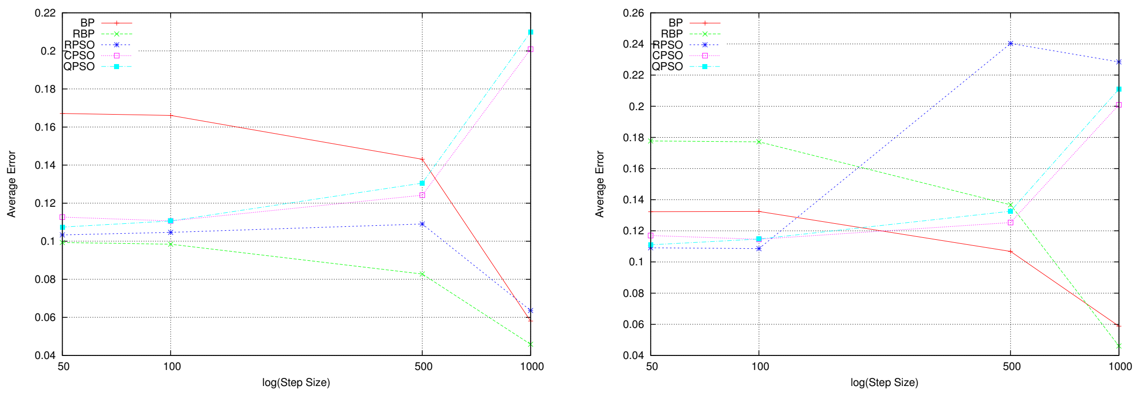
The algorithms were ranked based on their collective mean E_T and E_G values obtained

Table 6.37: Dynamic Sphere Results for Scenarios C1 to C4

Algorithm \ **Scenario**	**C1** *(frequency: 100, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.16711 ± 0.002519	0.132319 ± 0.004788	0.880798 ± 0.064164	n/a
Reinit. Back Propagation	0.099321 ± 0.000977	0.177713 ± 0.005864	1.48973 ± 0.075565	n/a
Reinitialising PSO	0.103326 ± 0.000945	0.109139 ± 0.003655	1.06411 ± 0.061053	3.50103 ± 0.376245
Charged PSO	0.11267 ± 0.026986	0.116969 ± 0.024469	1.05182 ± 0.054033	0.585116 ± 0.071748
Quantum PSO	0.107417 ± 0.00243	0.111008 ± 0.003637	1.04213 ± 0.058903	0.303734 ± 0.014627
	C2 *(frequency: 100, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.166076 ± 0.002259	0.132456 ± 0.004116	0.881805 ± 0.061776	n/a
Reinit. Back Propagation	0.098423 ± 0.000873	0.177158 ± 0.005948	1.49041 ± 0.080731	n/a
Reinitialising PSO	0.104653 ± 0.001422	0.108582 ± 0.003781	1.04456 ± 0.067094	7.34585 ± 0.883122
Charged PSO	0.110679 ± 0.003285	0.11461 ± 0.003748	1.03425 ± 0.048836	0.830069 ± 0.208063
Quantum PSO	0.110652 ± 0.003651	0.11474 ± 0.005502	1.04344 ± 0.058526	0.365559 ± 0.012645
	C3 *(frequency: 100, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.143071 ± 0.002946	0.106784 ± 0.005094	0.82033 ± 0.044328	n/a
Reinit. Back Propagation	0.082777 ± 0.002989	0.136712 ± 0.005531	1.35806 ± 0.0667	n/a
Reinitialising PSO	0.109036 ± 0.002281	0.240324 ± 0.013597	1.26354 ± 0.080892	8.38114 ± 1.05271
Charged PSO	0.1242 ± 0.003351	0.125316 ± 0.005589	1.00647 ± 0.047697	1.91204 ± 0.518048
Quantum PSO	0.130536 ± 0.005925	0.132597 ± 0.008417	1.01678 ± 0.052753	0.546288 ± 0.058308
	C4 *(frequency: 100, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.058047 ± 0.007239	0.058775 ± 0.00914	1.00516 ± 0.103986	n/a
Reinit. Back Propagation	0.045874 ± 0.002774	0.046078 ± 0.004682	1.01759 ± 0.117406	n/a
Reinitialising PSO	0.063607 ± 0.004789	0.228473 ± 0.013786	1.53888 ± 0.169998	8.45948 ± 2.03968
Charged PSO	0.200857 ± 0.015219	0.200924 ± 0.014838	1.00995 ± 0.043487	2.49748 ± 0.395299
Quantum PSO	0.209843 ± 0.014711	0.210977 ± 0.017679	1.01151 ± 0.043056	0.645333 ± 0.119236

Table 6.38: Dynamic Sphere Algorithm Ranking for Scenarios C1 to C4

Algorithm	C1		C2		C3		C4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	5	1	2	2	4.25	2.75
RBP	1	5	1	5	1	4	1	1	1	3.75
RPSO	2	2	2	1	2	5	3	5	2.25	3.25
CPSO	3.5	2	3.5	2.5	3	2	4	3	3.5	2.375
QPSO	3.5	2	3.5	2.5	4	3	5	4	4	2.875



(a) Average Training Error Results (b) Average Generalisation Error Results

Figure 6.51: Average Error results for Dynamic Sphere, scenarios C1 to C4

under C1 to C4, taking the p-values reported in Tables 6.32 and 6.33 into account. The obtained ranks, as well as the average ranks, are listed in Table 6.38. The collective mean E_T and E_G values as obtained for C1 to C4 are illustrated in Figure 6.51. Figure 6.51 illustrates that RBP obtained the lowest E_T values under C1 to C4. However, RBP's low E_T was supported by a correspondingly low E_G only under C4, where RBP showed to be the best performer (refer to Table 6.38). BP was again outperformed by the dynamic PSOs under gradual scenarios C1 to C2, and by RBP under C4. Only under C3 did BP manage to obtain the highest E_G rank. Out of the three dynamic PSOs, the RPSO consistently obtained the lowest E_T values under C1 to C4, but managed to support low E_T by a correspondingly low E_G only under gradual scenarios. The RPSO obtained the lowest E_G rank under C3 and C4, and, as the ρ_F values in Table 6.37 confirm, exhibited severe overfitting under C3 and C4. As was already mentioned, the RPSO stagnated after a sudden abrupt environment change under C3 and C4. Table 6.38 indicates that no statistically significant difference was observed between the CPSO and the QPSO under gradual scenarios, and the CPSO obtained a higher rank than the QPSO under the abrupt scenarios C3 and C4: average swarm diversity values in Table 6.37 indicate that the CPSO converged to a larger area than the QPSO. Hence, the CPSO explored more.

The average ranks in Table 6.38 indicate that RBP performed best in terms of E_T, and the CPSO obtained the highest E_G rank. Hence, despite poor performance of the dynamic PSOs under abrupt scenarios, at least a single dynamic PSO obtained a higher average rank than either BP or RBP on the dynamic sphere problem.

Scenarios D1 to D4: Figures 6.52 and 6.53 illustrate the progression of E_T and E_G obtained by the algorithms under scenarios D1 to D4. Figure 6.52 illustrates that the behaviour of BP, RBP, and RPSO under the gradual scenarios D1 and D2 closely resembled the behaviour of these algorithms under scenarios C1 and C2. The CPSO and the QPSO, however, produced very high error peaks closer to the end of the algorithm run. Since the only difference between C1 to C4 and D1 to D4 was the frequency of change, which decreased to 250 iterations in case of D1 to D4, it can be concluded that training for 250 iterations between environment changes caused the CPSO and the QPSO to over-exploit severely, and converge to a too small region around a solution. The swarm diversity profiles illustrated in Figure 6.54 confirm that the CPSO's and the QPSO's diversity remained close to zero under D1 to D4. One reason for the small diversity obtained is the small value of V_{max}, which prevented the particles from moving fast, and in case of D1 and D2 prevented the swarm from leaving a bad region in which the swarm happened to converge due to low temporal severity. The RPSO used a larger V_{max}

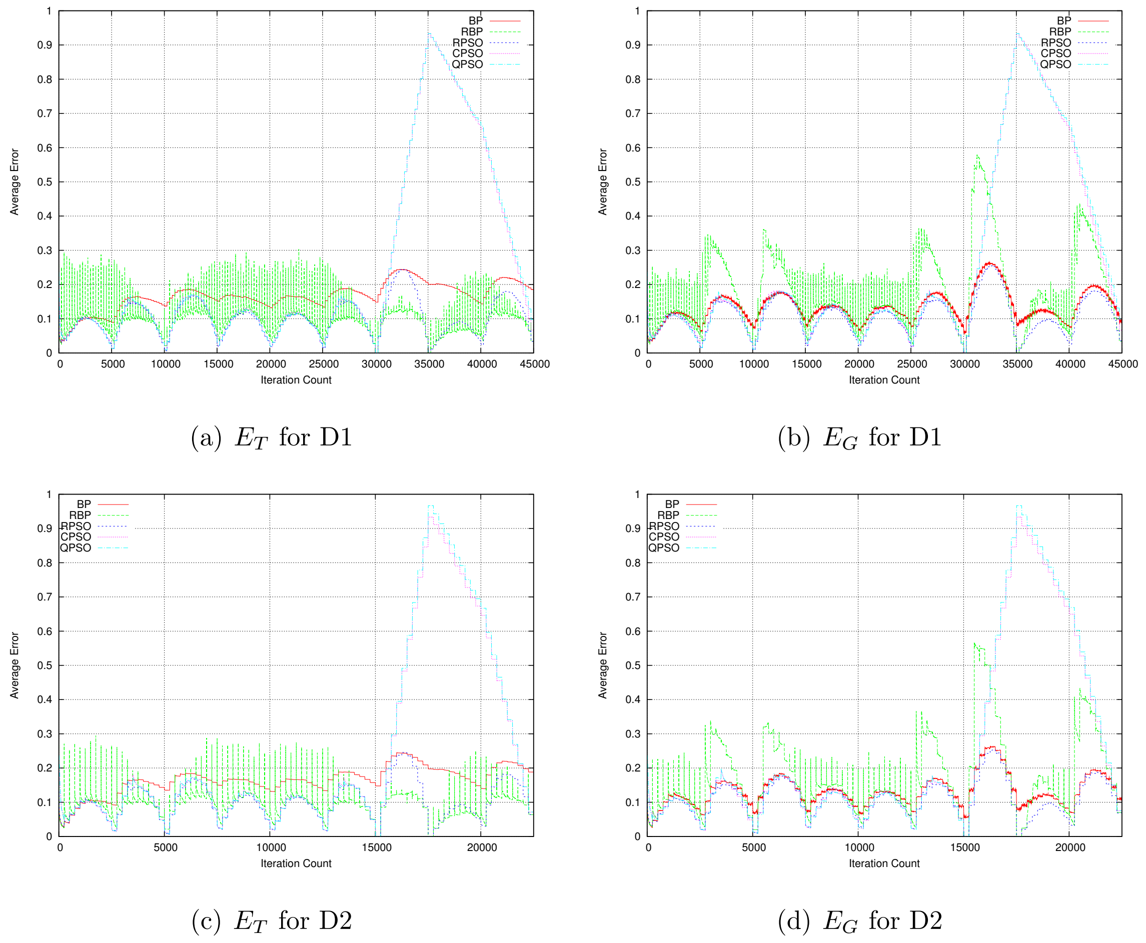


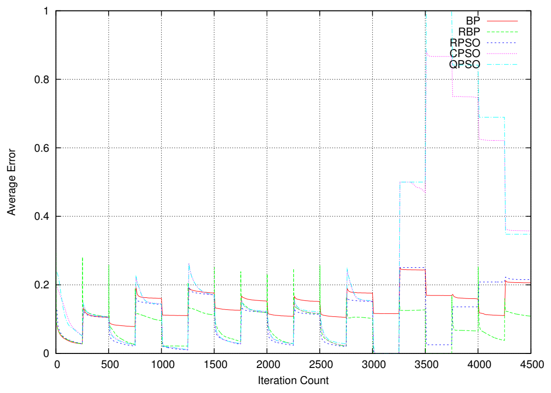
Figure 6.52: Training and Generalisation Error results for Dynamic Sphere, scenarios D1 to D2

value, and was less dependent on swarm memory, hence the RPSO did not deteriorate under gradual scenarios due to low temporal severity.

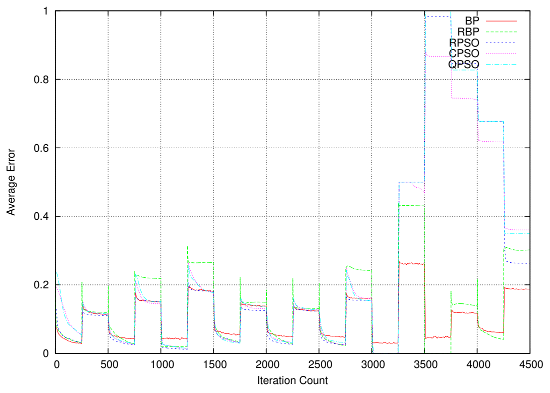
Figure 6.53 illustrates that BP, RBP, and RPSO exhibited similar behaviour under D3 and D4 as under C3 and C4. The CPSO and the QPSO, however, deteriorated in performance compared to C3 and C4: the CPSO and the QPSO no longer managed to recover from the error peak from which these algorithms did recover under all previous abrupt scenarios considered. Hence, the CPSO and the QPSO over-exploited due to low temporal severity under both gradual and abrupt scenarios. The fact that both

Table 6.39: Dynamic Sphere Results for Scenarios D1 to D4

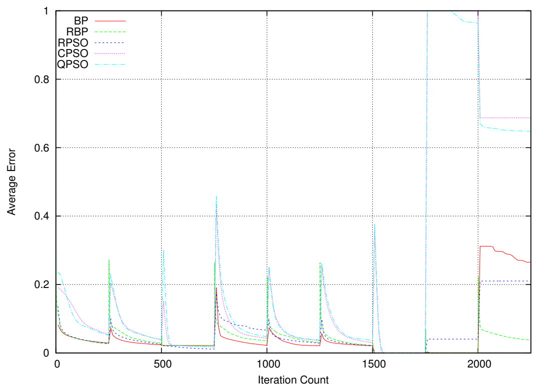
Scenario / **Algorithm**	**D1** *(frequency: 250, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.165768 ± 0.002174	0.132933 ± 0.003737	0.895851 ± 0.055219	n/a
Reinit. Back Propagation	0.091211 ± 0.000851	0.173511 ± 0.006563	1.59901 ± 0.085862	n/a
Reinitialising PSO	0.101189 ± 0.001321	0.10672 ± 0.004144	1.07123 ± 0.075417	4.7298 ± 0.914033
Charged PSO	0.244169 ± 0.029981	0.248832 ± 0.031562	1.06706 ± 0.065279	0.586799 ± 0.11553
Quantum PSO	0.246038 ± 0.020945	0.251531 ± 0.02237	1.06288 ± 0.062687	0.317741 ± 0.049831
	D2 *(frequency: 250, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.164773 ± 0.002493	0.13038 ± 0.005265	0.888469 ± 0.072247	n/a
Reinit. Back Propagation	0.090866 ± 0.001116	0.172943 ± 0.005622	1.59321 ± 0.081327	n/a
Reinitialising PSO	0.101856 ± 0.001426	0.107513 ± 0.004631	1.06235 ± 0.080856	11.6333 ± 1.3867
Charged PSO	0.242995 ± 0.036794	0.246834 ± 0.037879	1.04924 ± 0.061249	0.640546 ± 0.069317
Quantum PSO	0.247662 ± 0.026591	0.251565 ± 0.027257	1.04484 ± 0.049591	0.344882 ± 0.036625
	D3 *(frequency: 250, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.140808 ± 0.002707	0.105287 ± 0.004103	0.82433 ± 0.056355	n/a
Reinit. Back Propagation	0.073325 ± 0.001552	0.129212 ± 0.004556	1.43011 ± 0.085763	n/a
Reinitialising PSO	0.102763 ± 0.001444	0.240959 ± 0.008675	1.30686 ± 0.0761	10.142 ± 1.15623
Charged PSO	0.236357 ± 0.041818	0.238866 ± 0.04002	1.02684 ± 0.061654	1.51047 ± 0.495413
Quantum PSO	0.25326 ± 0.001802	0.254952 ± 0.006693	1.03908 ± 0.050078	0.454367 ± 0.032219
	D4 *(frequency: 250, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.054159 ± 0.007395	0.056111 ± 0.008284	1.07694 ± 0.137488	n/a
Reinit. Back Propagation	0.034517 ± 0.00331	0.035871 ± 0.004875	1.05263 ± 0.128093	n/a
Reinitialising PSO	0.056596 ± 0.008	0.226403 ± 0.023402	1.72321 ± 0.182445	8.34344 ± 0.897918
Charged PSO	0.240363 ± 0.006358	0.241697 ± 0.007598	0.999884 ± 0.068479	2.0639 ± 0.560476
Quantum PSO	0.238631 ± 0.019632	0.240392 ± 0.019668	1.02124 ± 0.06226	0.530062 ± 0.068598



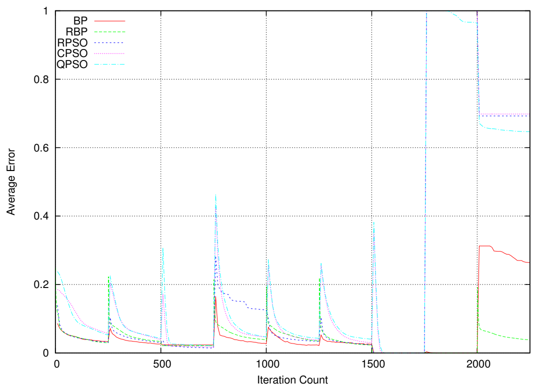
(a) E_T for D3



(b) E_G for D3



(c) E_T for D4



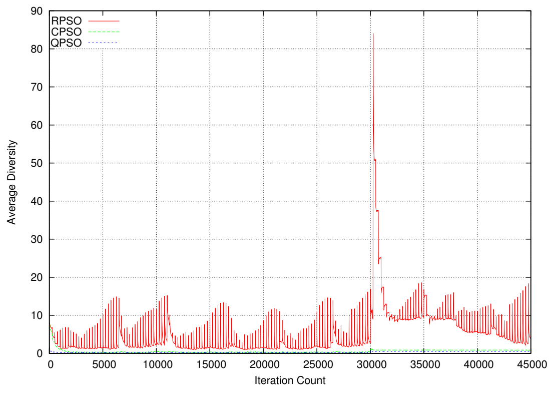
(d) E_G for D4

Figure 6.53: Training and Generalisation Error results for Dynamic Sphere, scenarios D3 to D4

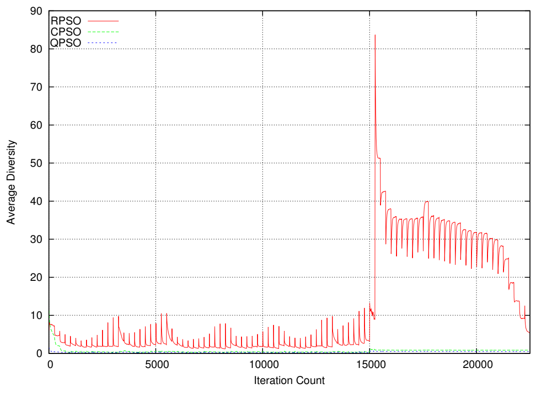
BP and RBP were not visibly affected by the extent of temporal severity indicates that the dynamic PSOs are more sensitive to the temporal severity property of a dynamic problem, and more susceptible to over-exploitation when allowed to train for too long.

Figure 6.54 illustrates the swarm diversity profile over time obtained by the dynamic PSOs under D1 to D4. Figure 6.54 shows that the QPSO and the CPSO maintained very low diversity, and the RPSO diverged after encountering an abrupt change under scenarios D3 and D4.

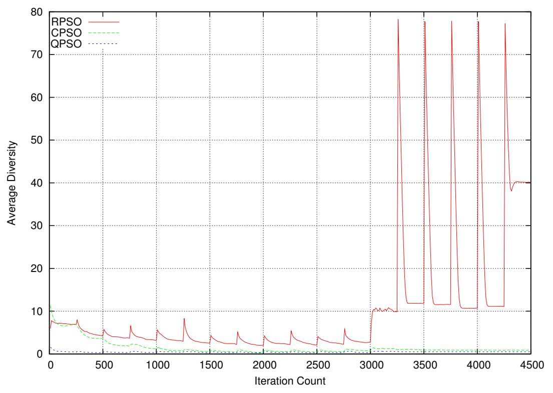
Table 6.39 lists collective mean errors, ρ_F values, as well as average swarm diversity



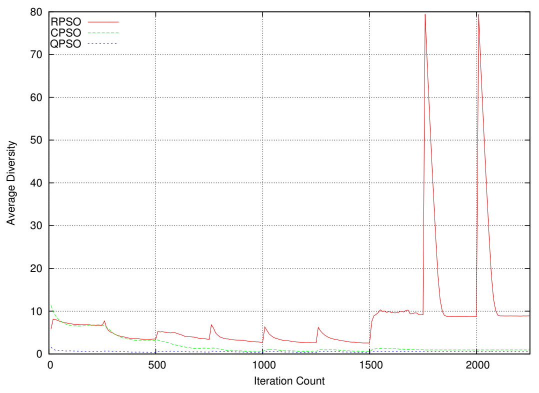
(a) Average Diversity Results for Scenario D1



(b) Average Diversity Results for Scenario D2



(c) Average Diversity Results for Scenario D3

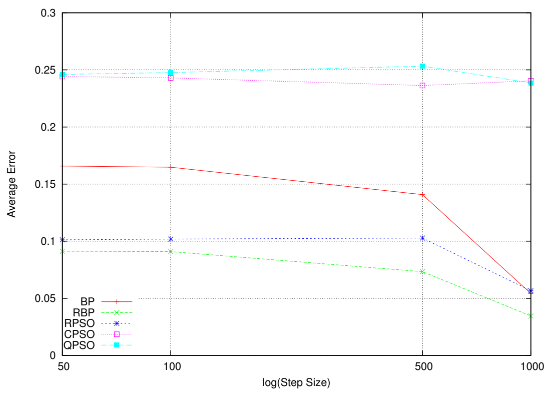


(d) Average Diversity Results for Scenario D4

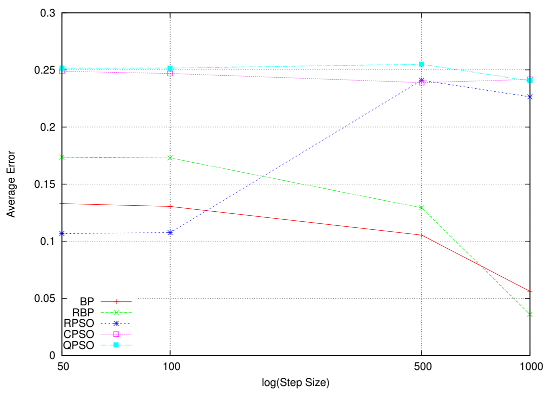
Figure 6.54: Average Diversity results for Dynamic Sphere, scenarios D1 to D4

(where applicable) obtained by the algorithms under scenarios D1 to D4. The ρ_F values show that the algorithms exhibited the same overfitting trends as under C1 to C4. Average diversity values show that the RPSO's average diversity increased compared to C1 to C4, and the CPSO's and the QPSO's average diversity decreased.

The collective mean E_T and E_G values reported in Table 6.39 are illustrated in Figure 6.55. For the sake of a convenient statistical comparison, the algorithms are ranked based on their collective mean E_T and E_G values from Table 6.39, and the p-values reported in Tables 6.32 and 6.33. The resulting ranks, along with the average ranks, are listed in Table 6.40. Figure 6.55 and Table 6.40 show that no statistically significant difference



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.55: Average Error results for Dynamic Sphere, scenarios D1 to D4

Table 6.40: Dynamic Sphere Algorithm Ranking for Scenarios D1 to D4

Algorithm	D1		D2		D3		D4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	3	2	3	2	3	1	2.5	2	2.875	1.75
RBP	1	3	1	3	1	2	1	1	1	2.25
RPSO	2	1	2	1	2	4	2.5	3	2.125	2.25
CPSO	4.5	4.5	4.5	4.5	4	3	4.5	4.5	4.375	4.125
QPSO	4.5	4.5	4.5	4.5	5	5	4.5	4.5	4.625	4.625

was observed between the CPSO and the QPSO under most scenarios, and these two algorithms produced the worst average ranks under scenarios D1 to D4. The RPSO performed more competitively, significantly outperforming all other algorithms in terms of E_G under the gradual scenarios D1 and D2, but also deteriorating in performance under D3 and D4. BP and RBP exhibited similar behaviour as was observed under C1 to C4, and RBP once again outperformed all other algorithms under the most abrupt scenario D4. The average ranks in Table 6.40 show that BP obtained the lowest average E_G rank under D1 to D4, thus BP can be considered the best performer under the least temporally severe scenarios.

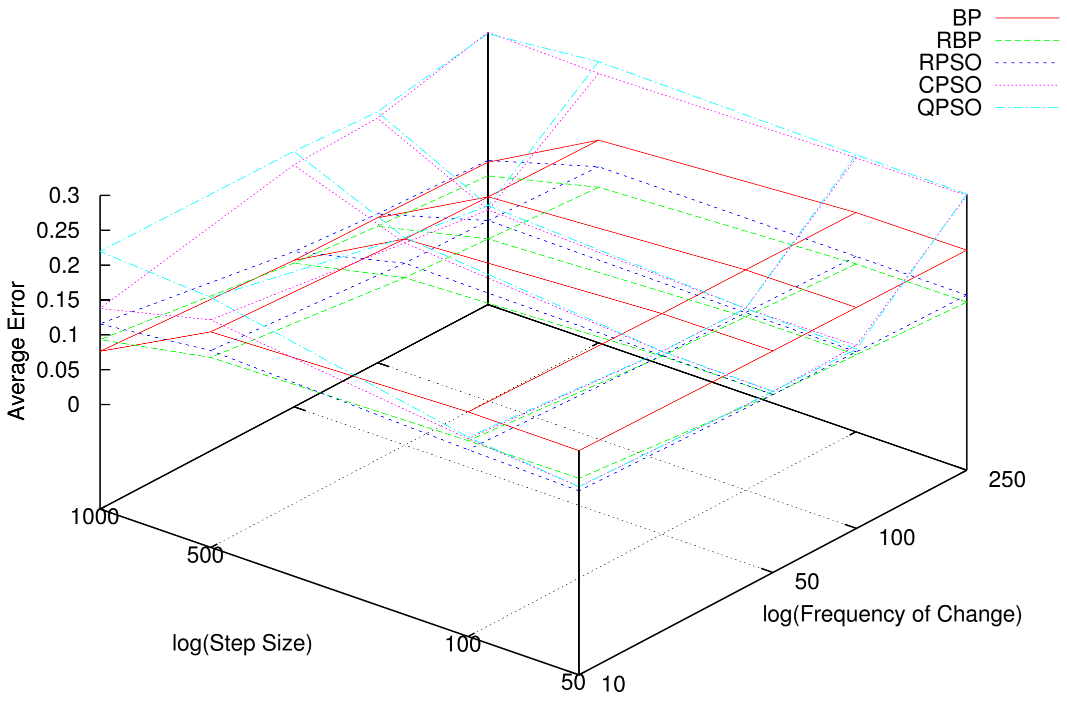


Figure 6.56: Average Training Error Results for Dynamic Sphere

All scenarios: In order to aid drawing more general patterns from the empirical data, the collective mean E_T values produced by the five algorithms under the 16 different dynamic scenarios are visualised in Figure 6.56, and the collective mean E_G values are visualised in Figure 6.57.

Figure 6.56 illustrates that the CPSO and the QPSO have performed their best under gradual scenarios of high temporal severity, and deteriorated as either abruptness of changes increased, or the frequency of changes decreased. The RPSO did not deteriorate in terms of E_T under either infrequent or abrupt changes. However, Figure 6.57 illustrates that the RPSO deteriorated even faster than the CPSO and the QPSO in terms of E_G as the spatial severity increased. Hence, all algorithms deteriorated in performance with an increase in spatial severity. The RPSO, as Figure 6.57 illustrates, was less affected

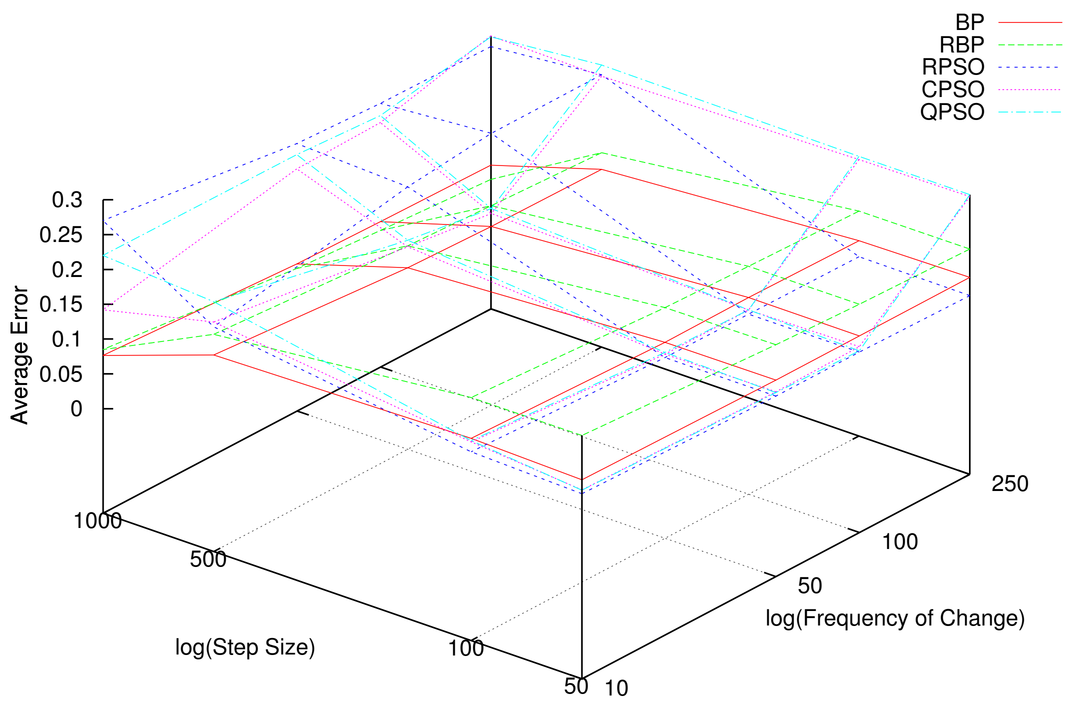


Figure 6.57: Average Generalisation Error Results for Dynamic Sphere

by changes in temporal severity than the CPSO and the QPSO: the RPSO was less memory-dependent, hence the RPSO was less prone to over-exploit when left training for too long. The larger V_{max} value used for the RPSO also promoted exploration.

The success of slowly-converging dynamic PSOs under scenarios of high temporal severity is an indication that the dynamic sphere was a trivial problem to optimise. This supposition is further confirmed by the fact that RBP outperformed all other algorithms under the most abrupt scenarios in terms of both E_T and E_G, as Figures 6.56 and 6.57 illustrate.

BP was outperformed by the dynamic PSOs under gradual scenarios, and by RBP under the most abrupt scenarios. However, BP showed superior performance under scenarios A3 to D3 (step size of 500 patterns), when the changes were too abrupt for

Table 6.41: Dynamic Sphere Average Algorithm Ranking for Scenarios A to D

Algorithm	Average R(A)		Average R(B)		Average R(C)		Average R(D)		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	3.5	2.5	4.125	2.75	4.25	2.75	2.875	1.75	3.6875	2.4375
RBP	2.25	3.625	1.125	3.375	1	3.75	1	2.25	1.34375	3.25
RPSO	1.75	2.375	2.125	3	2.25	3.25	2.125	2.25	2.0625	2.71875
CPSO	3.375	3	3.5	2.5	3.5	2.375	4.375	4.125	3.6875	3
QPSO	4.125	3.5	4.125	3.375	4	2.875	4.625	4.625	4.21875	3.59375

the dynamic PSOs, and RBP failed due to the presence of stale data inside the sliding window.

Average ranks calculated for scenarios A1 to D4, along with the overall average ranks, are summarised in Table 6.41. Overall average ranks in Table 6.41 indicate that RBP obtained the highest average rank in terms of training, and BP obtained the highest average rank in terms of generalisation. Since the ability to generalise is crucial for a trained NN to be of any use, the generalisation rank is considered more important than the training rank. Hence, BP can be labelled as the overall top performer on the dynamic sphere problem. Although the dynamic PSOs exhibited significantly better performance than both BP and RBP under gradual scenarios, the dynamic sphere problem was too trivial to justify the use of the population-based algorithms such as the dynamic PSOs.

6.2.4 Sliding Thresholds

For this problem, the Cartesian space was subdivided into three classes by means of two parallel linear thresholds.

Problem Definition

The two parallel linear thresholds, $f_1(\vec{x})$ and $f_2(\vec{x})$, are given by

$$\begin{aligned} f_1(\vec{x}) &= t_1 \\ f_2(\vec{x}) &= t_2 \\ t_1 &< t_2, \end{aligned}$$

where t_1 and t_2 are constant values, therefore $f_1(\vec{x})$ and $f_2(\vec{x})$ are parallel vertical lines. Thus, a 2D point can be classified based on its x-axis component only (x_1). Classification was done as follows:

$$Classification(\vec{x}) = \begin{cases} Class\ A & \text{if } x_1 \leq t_1 \\ Class\ B & \text{if } x_1 \geq t_2 \\ Class\ C & \text{otherwise} \end{cases} \tag{6.6}$$

A data set was generated according to the procedure described in Section 6.1.2, where the number of points, M, was set to 1000, and the total number of environment changes, N, was set to 10. A set of M 2-dimensional points, $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{1000}\}$ was randomly generated such that $\vec{x}_l \in [0,1]^2$, $l = 1, \ldots, 1000$, and each point $\vec{x}_l$ was assigned a class value according to equation (6.6). Thresholds were generated N times by setting t_1 and t_2 to random numbers from the interval $[0,1]$ such that $t_1 < t_2$. Target classification of M patterns was updated N times, and every time the updated patterns were appended to the data set, yielding a data set of 11 000 patterns in total. The size of the sliding window was set to 1000.

A NN with 2 input units, 3 hidden units, and 3 output units was trained on the Sliding Thresholds problem. According to equation (6.3), the total number of weights and biases, corresponding to the dimensionality of the problem, was equal to 24.

Sixteen different dynamic scenarios as described in Section 6.2.2 were considered for the sliding thresholds problem. The chosen scenarios simulated different combinations

Table 6.42: Optimal Parameters for the Sliding Thresholds problem

Algorithm	**Parameters**			
Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-1, 1]$	0.1	0.1	
Reinitialising Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-1, 1]$	0.1	0.2	
Reinitialising PSO	Weight Interval	V_{max}	Swarm Size	Reinitialisation Ratio
	$[-1, 1]$	1	50	0.75
Charged PSO	Weight Interval	V_{max}	Swarm Size	Charge Magnitude
	$[-1, 1]$	2	50	20
Quantum PSO	Weight Interval	V_{max}	Swarm Size	Cloud Radius
	$[-1, 1]$	2	50	2

of spatial and temporal severity, providing representative coverage of different dynamic environment types. The technique of simulating dynamic scenarios applied in this study is described in Section 6.1.2. Parameter settings corresponding to each dynamic scenario are listed in Table 6.17.

Parameter Optimisation

All algorithm parameters were optimised according to the procedure described in Section 6.1.3. Corresponding optimal parameters discovered are listed in Table 6.42.

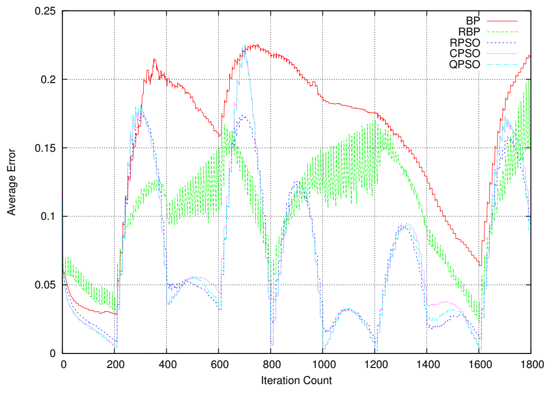
Analysis of Empirical Data

The number of iterations required to traverse the entire data set under every dynamic scenario considered was calculated using equation (6.4).

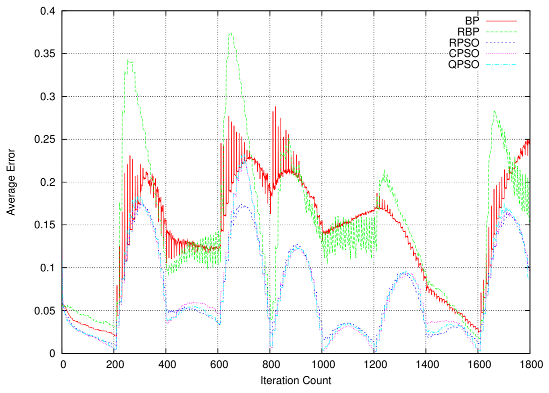
Sliding thresholds is a 2D problem that has only linear decision boundaries. Although linear decision boundaries are trivial to learn, this problem was rather difficult to optimise due to the sliding window approach used to simulate dynamic environments. As discussed in Section 6.1.2, a dynamic environment is simulated by sliding a window over a large data set. If the data set is traversed in order, the classification of the data patterns will change every $M = 1000$ patterns. The previous problems discussed dealt with one decision boundary per 1000 patterns. Thus, for the previous problems considered, a sliding window of N patterns, $N \leq M$, contained at least one decision boundary and at most two. For the sliding thresholds problem, there were two decision boundaries per every 1000 data patterns. After every 1000 patterns traversed, two new boundaries were introduced. Therefore, for the sliding thresholds problem, the sliding window contained at least two decision boundaries and at most four. Although linear boundaries are trivial, it can be problematic for the training algorithm to simultaneously detect two new boundaries. Furthermore, old boundaries and new boundaries may be mutually exclusive, i.e. classify the same pattern into different classes, since new boundaries were generated randomly.

The rest of this section is dedicated to the analysis of the empirical results obtained for the sliding thresholds problem.

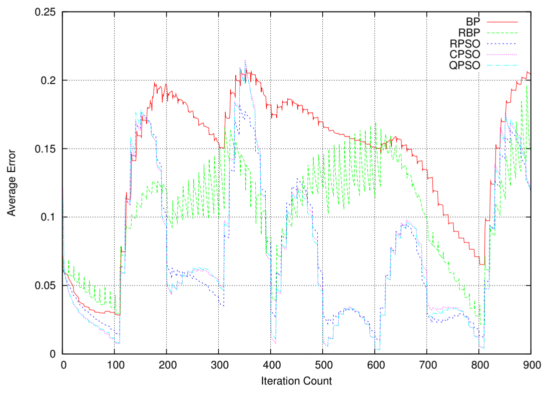
Scenarios A1 to A4: Figures 6.58 and 6.59 illustrate the E_T and E_G profiles obtained by the five algorithms considered under scenarios A1 to A4. Figure 6.58 illustrates that under gradual changes, both BP and RBP exploited the found solutions much worse than the dynamic PSOs. As already explained, the total number of decision boundaries for the sliding thresholds problem was more than for the other problems considered. Hence, the error function landscape became more difficult to traverse for the hill-climbers such as BP and RBP, and the population-based dynamic PSOs explored and exploited the weight space better.



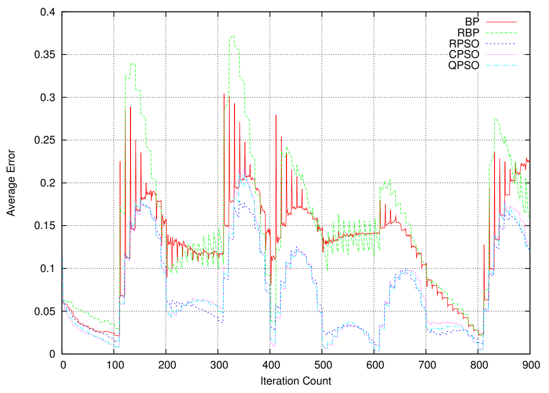
(a) E_T for A1



(b) E_G for A1



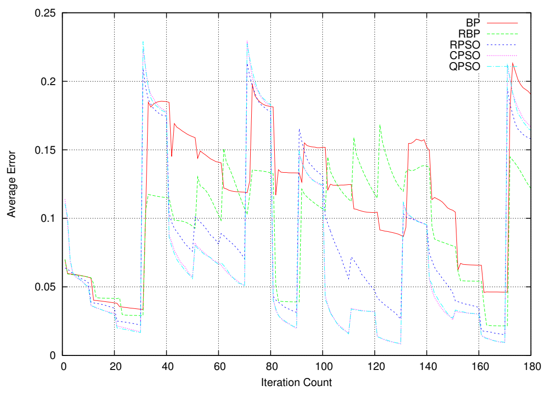
(c) E_T for A2



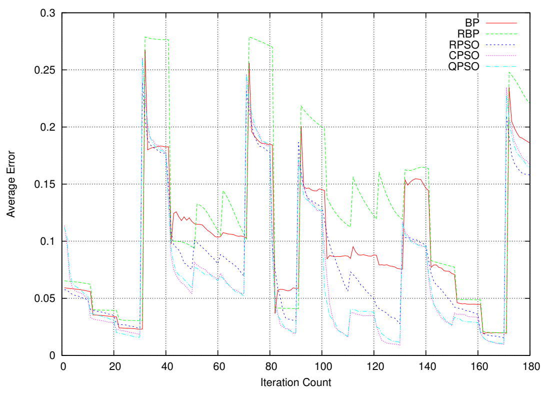
(d) E_G for A2

Figure 6.58: Training and Generalisation Error results for Sliding Thresholds, scenarios A1 to A2

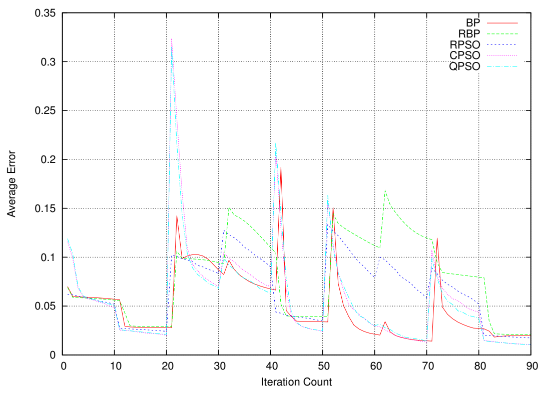
Figure 6.59 shows that under A3, when conflicting boundaries still may have been present, BP and RBP were outperformed by the dynamic PSOs. However, both the training and generalisation performance of BP and RBP significantly improved under abrupt scenario A4 where no conflicting boundaries were present. Figures 6.59(a) and 6.59(b) illustrate that RBP's E_T and E_G were decreasing between environment changes, as opposed to BP, whose E_T and E_G appeared rather stable between changes, thus indicating stagnation. Scenarios A1 to A4 simulated changes of high temporal severity, and RBP did not have enough time to converge on a good solution between environment



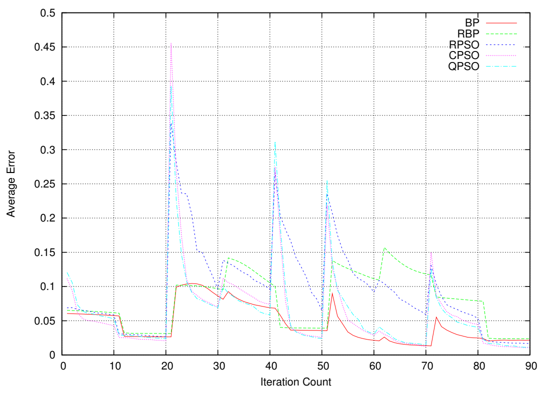
(a) E_T for A3



(b) E_G for A3



(c) E_T for A4



(d) E_G for A4

Figure 6.59: Training and Generalisation Error results for Sliding Thresholds, scenarios A3 to A4

changes. Figure 6.59 illustrates that the dynamic PSOs suffered from the same thing under abrupt changes: although the produced E_T and E_G decreased between changes, the frequency of 10 iterations did not allow the slow-converging dynamic PSOs to properly exploit the fruitful regions found. Therefore, both RBP and the dynamic PSOs are expected to outperform BP under abrupt environment changes of lower temporal severity.

Table 6.43 summarises the mean fitness results obtained for scenarios A1 to A4. Average ρ_F and average swarm diversity (where applicable) are also reported. The values of ρ_F in Table 6.43 show that all algorithms except BP were susceptible to overfitting

Table 6.43: Sliding Thresholds Results for Scenarios A1 to A4

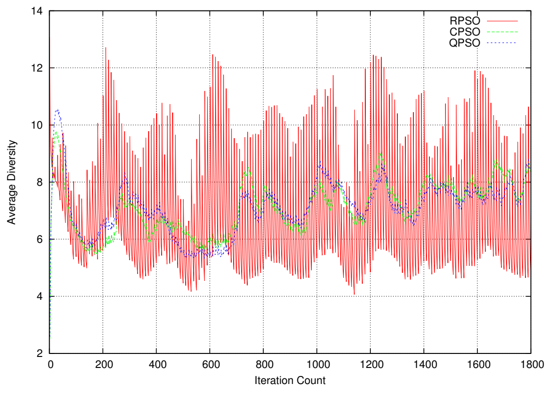
Scenario / **Algorithm**	**A1** *(frequency: 10, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.153493 ± 0.015032	0.136737 ± 0.014298	0.909784 ± 0.059484	n/a
Reinit. Back Propagation	0.103516 ± 0.000502	0.147665 ± 0.003263	1.46147 ± 0.068171	n/a
Reinitialising PSO	0.070907 ± 0.000673	0.071907 ± 0.002433	0.998407 ± 0.072136	6.82679 ± 0.196598
Charged PSO	0.072955 ± 0.004674	0.073412 ± 0.005112	1.01682 ± 0.0509	7.11741 ± 0.593174
Quantum PSO	0.071743 ± 0.003221	0.073101 ± 0.00487	1.02934 ± 0.083099	7.10982 ± 0.600528
	A2 *(frequency: 10, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.142729 ± 0.009063	0.125513 ± 0.011567	0.906443 ± 0.067477	n/a
Reinit. Back Propagation	0.102635 ± 0.00077	0.147616 ± 0.004737	1.48826 ± 0.095198	n/a
Reinitialising PSO	0.074074 ± 0.000665	0.074277 ± 0.003028	1.00798 ± 0.077885	7.00757 ± 0.217315
Charged PSO	0.074585 ± 0.003901	0.075893 ± 0.004829	1.00052 ± 0.050479	7.91395 ± 0.50089
Quantum PSO	0.074158 ± 0.003075	0.074692 ± 0.00366	1.0082 ± 0.054341	7.78085 ± 0.72978
	A3 *(frequency: 10, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.114955 ± 0.009776	0.098212 ± 0.012554	0.886697 ± 0.082879	n/a
Reinit. Back Propagation	0.09379 ± 0.001615	0.12228 ± 0.004462	1.31493 ± 0.132004	n/a
Reinitialising PSO	0.082315 ± 0.002789	0.084629 ± 0.005339	1.01245 ± 0.091781	7.59205 ± 0.356288
Charged PSO	0.072066 ± 0.002915	0.075276 ± 0.00405	1.0284 ± 0.085202	8.93874 ± 0.640144
Quantum PSO	0.071948 ± 0.00227	0.075056 ± 0.003118	1.04383 ± 0.072746	8.94221 ± 0.945251
	A4 *(frequency: 10, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.049715 ± 0.004035	0.045486 ± 0.005378	0.945583 ± 0.109429	n/a
Reinit. Back Propagation	0.080386 ± 0.002757	0.080073 ± 0.005149	1.02705 ± 0.163965	n/a
Reinitialising PSO	0.066253 ± 0.004782	0.096665 ± 0.024581	1.45872 ± 0.361835	7.84682 ± 0.271817
Charged PSO	0.058537 ± 0.005081	0.063731 ± 0.007158	1.04655 ± 0.094568	9.90675 ± 1.07051
Quantum PSO	0.057151 ± 0.005216	0.063549 ± 0.007882	1.0868 ± 0.091541	9.74897 ± 0.763392

Table 6.44: Mann-Whitney *U* p-values obtained for the average training error comparisons on the Sliding Thresholds problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

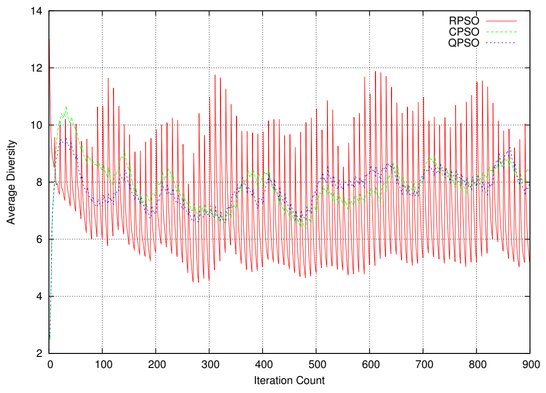
	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.000164**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.093511	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.074711	0.697635	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.686783	**0.002599**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.000175**	**0.023386**
D	**0.010973**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.002599**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	0.494594	0.854916	**0.0001**	**0.0001**	0.266409	0.708548	0.775176	0.202493
B	**0.0001**	**0.0001**	0.719519	**0.02959**	0.072293	0.241747	0.592174	0.423131
C	**0.0001**	**0.0001**	**0.007643**	0.241747	0.552109	0.423131	0.820526	0.458089
D	**0.0001**	**0.0001**	**0.0001**	0.130369	0.365817	0.159042	0.373698	0.27936

Table 6.45: Mann-Whitney *U* p-values obtained for the average generalisation error comparisons on the Sliding Thresholds problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

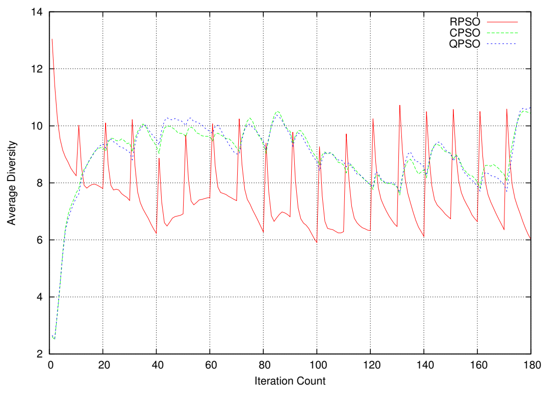
	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.030748**	**0.000102**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.003192**	**0.000117**	**0.020724**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.002599**	**0.001794**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.004311**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.168039	0.159042	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.763945	**0.002107**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.014864**	0.096477
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.001155**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	0.172671	0.786452	**0.0001**	**0.0001**	0.809127	0.260089	0.947634	0.866438
B	**0.0001**	**0.001998**	0.077194	0.087794	0.197283	0.207802	0.406247	0.592174
C	**0.0001**	**0.0001**	**0.009599**	0.292726	0.458089	0.654613	0.602392	0.414639
D	**0.0001**	**0.0001**	**0.0001**	0.177408	0.059075	0.172677	0.272832	0.159042



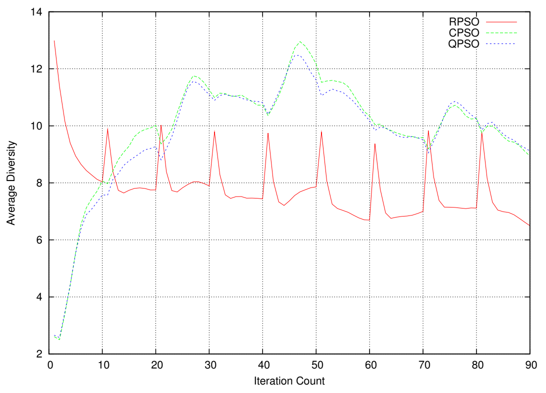
(a) Average Diversity Results for Scenario A1



(b) Average Diversity Results for Scenario A2



(c) Average Diversity Results for Scenario A3

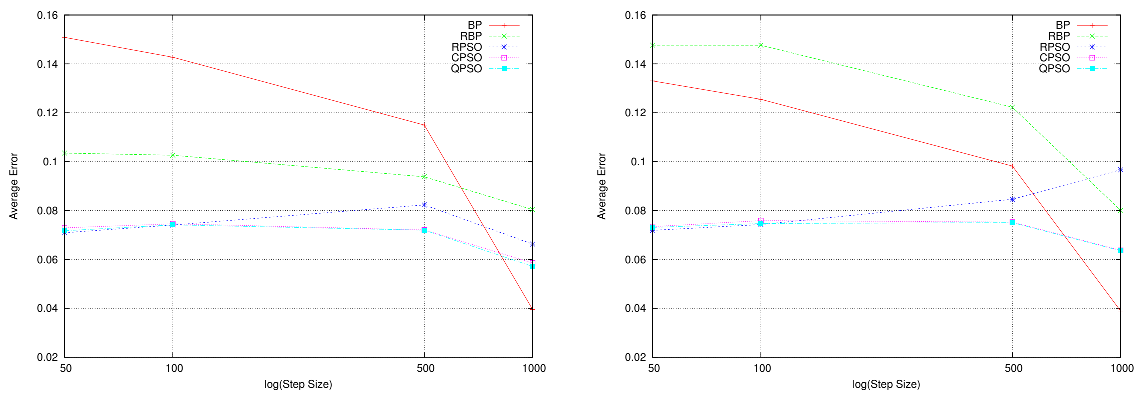


(d) Average Diversity Results for Scenario A4

Figure 6.60: Average Diversity results for Sliding Thresholds, scenarios A1 to A4

Table 6.46: Sliding Thresholds Algorithm Ranking for Scenarios A1 to A4

Algorithm	A1		A2		A3		A4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	5	4	1	1	4	3.25
RBP	4	5	4	5	4	5	5	4	4.25	4.75
RPSO	2	2	2	2	3	3	4	5	2.75	3
CPSO	2	2	2	2	1.5	1.5	2.5	2.5	2	2
QPSO	2	2	2	2	1.5	1.5	2.5	2.5	2	2



(a) Average Training Error Results (b) Average Generalisation Error Results

Figure 6.61: Average Error results for Sliding Thresholds, scenarios A1 to A4

under A1 to A4. Out of the three dynamic PSOs, the RPSO obtained the lowest ρ_F value under gradual scenario A1, and the highest ρ_F value under abrupt scenario A4. An examination of the swarm diversity profiles provides an explanation for the overfitting behaviour of the RPSO: Figure 6.60 illustrates that the RPSO converged to a smaller diversity under all scenarios, since the RPSO, as opposed to the CPSO and the QPSO, did not implement any diversity-preserving mechanism between environment changes. Thus, the RPSO had a higher chance of over-exploiting due to the decreased exploration ability.

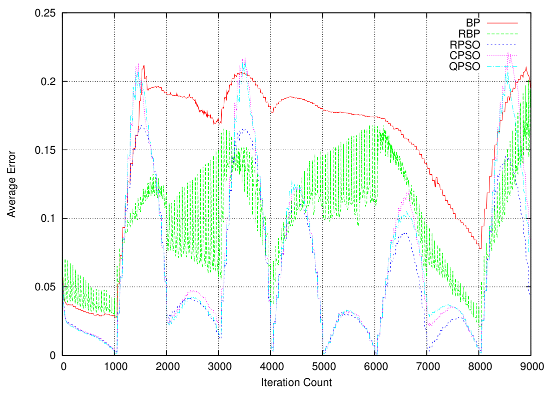
The algorithms were ranked based on their collective mean E_T and E_G values, taking the p-values reported in Tables 6.44 and 6.45 into account. The obtained ranks are listed in Table 6.46. Collective mean E_T and E_G values obtained for A1 to A4 are illustrated in Figure 6.61. As Figure 6.61 illustrates, both BP and RBP improved in performance as the abruptness of changes increased. Increasing abruptness reduced the total number of environment changes encountered during the algorithm run. Spatially severe changes also sped up the process of discarding obsolete data, since a larger amount of old data patterns was replaced by new data at every change. The dynamic PSOs, as Figure 6.61 illustrates, were less sensitive to the number of environment changes and the spatial severity of changes. Table 6.46 confirms that both BP and RBP were significantly outperformed by the dynamic PSOs under scenarios A1 to A3. However, BP was the top performer under A4 (see Table 6.46), confirming the observations made earlier based on the algorithms' error profiles.

The RBP, similar to the problems considered previously, trained significantly better than BP and generalised significantly worse for the scenarios where conflicting boundaries were present. Hence, a complete reinitialisation of weights did not prove efficient under temporally severe changes, where RBP was not given enough time to refine the found solution. The same applies to the RPSO, which deteriorated as the abruptness of changes increased.

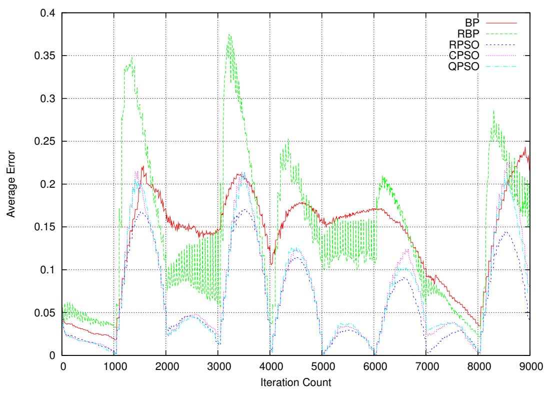
Table 6.46 shows that no statistically significant difference was observed between the CPSO and the QPSO. Average ranks in Table 6.46 also show that both these algorithms obtained the highest average rank, therefore they can both be considered as top performers for scenarios A1 to A4.

Scenarios B1 to B4: Figures 6.62 and 6.63 illustrate the progression of E_T and E_G over time as obtained by the five algorithms for scenarios B1 to B4. Figure 6.62 illustrates that, for the gradual scenarios B1 and B2, the algorithms exhibited similar trends as observed for A1 and A2: the dynamic PSOs exploited better than both BP and RBP, and RBP exploited better than BP.

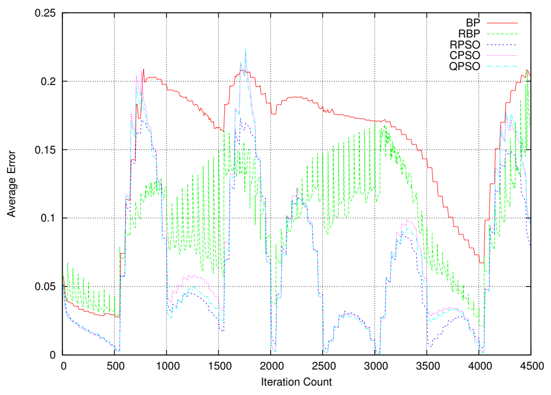
Figure 6.63 illustrates that RBP, given enough time to converge between changes (50 iterations instead of 10), managed to outperform BP under B3: RBP managed to obtain lower minimum E_T error values than BP between most changes. However, Figure 6.63(b) illustrates that RBP did not always support low E_T values by equally low E_G values. The tendency of RBP to overfit can be due to the sensitivity of this training algorithm to the presence of conflicting boundaries in the training data. This hypothesis is proved accurate by Figure 6.63(d), which illustrates that RBP generalised well under



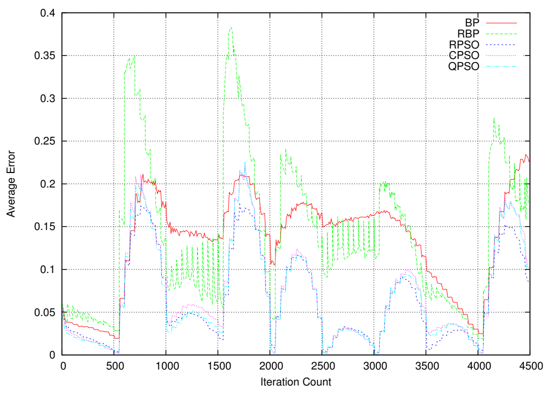
(a) E_T for B1



(b) E_G for B1



(c) E_T for B2

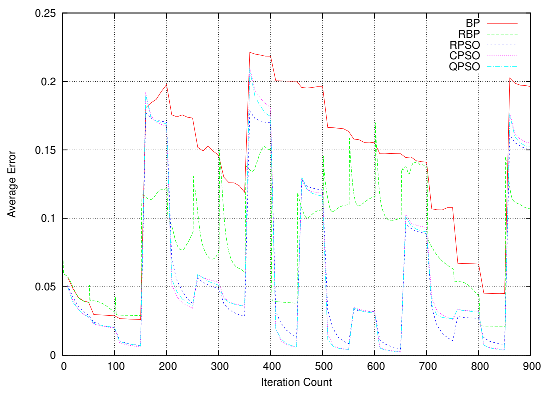


(d) E_G for B2

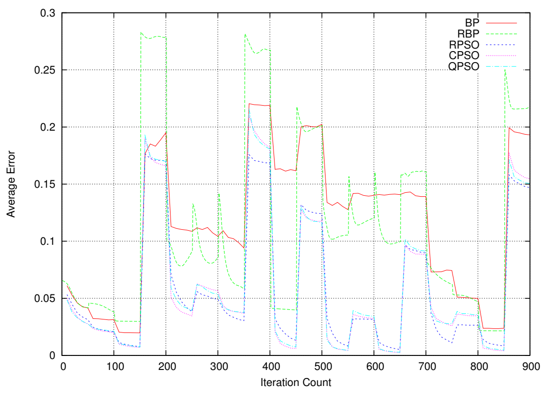
Figure 6.62: Training and Generalisation Error results for Sliding Thresholds, scenarios B1 to B2

the abrupt scenario B4 where no conflicting boundaries were present.

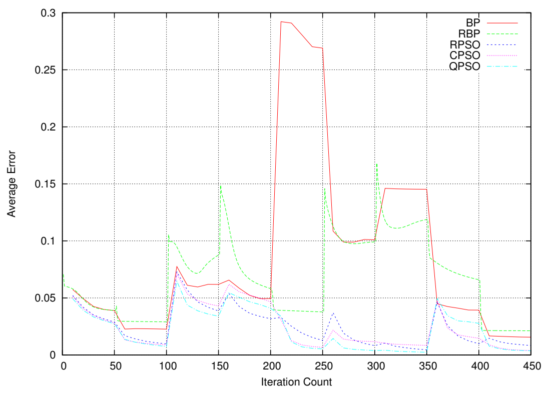
Figure 6.63 also illustrates that the dynamic PSOs benefited from decreased temporal severity by better exploiting the fruitful regions found, and reaching lower E_T and E_G values between environment changes. The same can not be said about BP: Figures 6.63(c) and 6.63(d) illustrate that BP failed to optimise the sliding thresholds decision boundaries under the B4 scenario. No conflicting decision boundaries where present under B4, thus stale data can not be blaimed for BP's failure. As discussed in Chapter 5, BP will fail to adapt to an environment change if the change of the error function



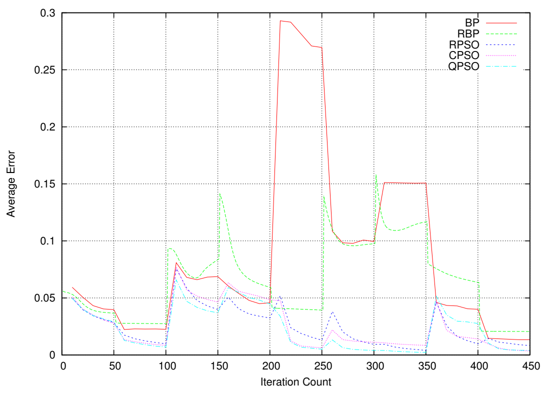
(a) E_T for B3



(b) E_G for B3



(c) E_T for B4

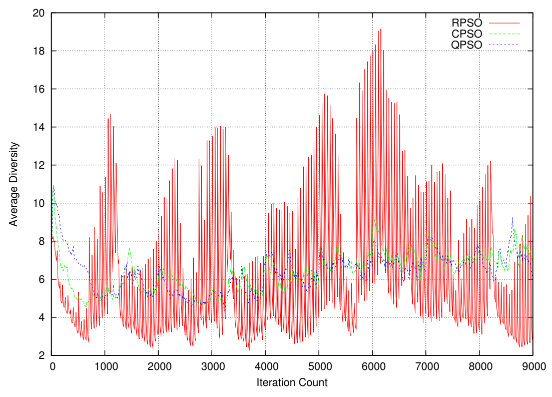


(d) E_G for B4

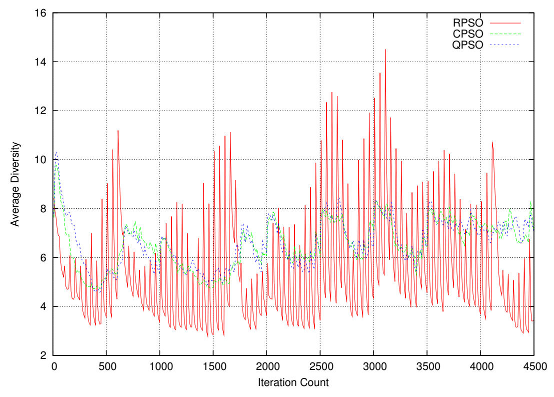
Figure 6.63: Training and Generalisation Error results for Sliding Thresholds, scenarios B3 to B4

landscape causes the current position of the NN weight vector to be in an unfruitful region such as a plateau of a local minima. The failure of BP to adapt to some changes under B4 indicates that these changes trapped BP in a region from which BP could not escape. The success of BP on the same problem under more temporally severe scenario A4 indicates that the decrease in temporal severity allowed BP to better exploit, thus leading BP deeper into areas where BP became stuck after environment changes.

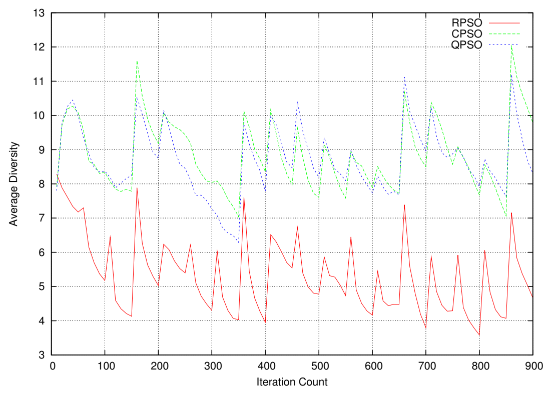
RBP was reinitialised after every change, thus RBP did not get trapped in unfruitful regions, such as plateaus or local minima, as easily as BP. However, RBP performed



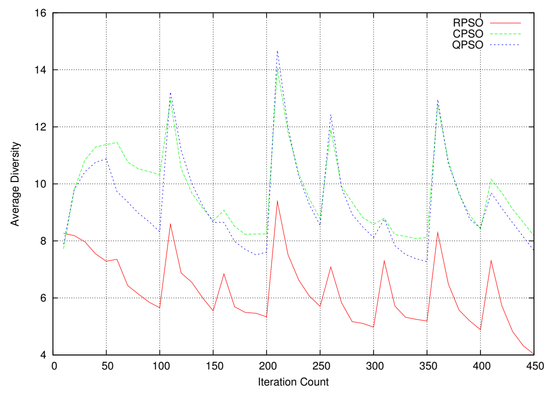
(a) Average Diversity Results for Scenario B1



(b) Average Diversity Results for Scenario B2



(c) Average Diversity Results for Scenario B3



(d) Average Diversity Results for Scenario B4

Figure 6.64: Average Diversity results for Sliding Thresholds, scenarios B1 to B4

significantly worse than the dynamic PSOs under B4: Figures 6.63(c) and 6.63(d) illustrate that RBP stagnated after some environment changes. Stagnation of RBP is attributed to the strong dependency of any hill-climbing algorithm on the starting point of the search: the RBP failed to find the optimum when the weight vector was put in an unfruitful region of the search space by reinitialisation.

Figure 6.64 illustrates the progression of swarm diversity over time for scenarios B1 to B4. Figures 6.64(c) and 6.64(d) illustrate that the dynamic PSOs did not converge to a stable diversity level between changes. Hence, the dynamic PSOs are expected to exploit more with further decreases in temporal severity.

Table 6.47: Sliding Thresholds Results for Scenarios B1 to B4

Scenario / **Algorithm**	**B1** *(frequency: 50, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.151092 ± 0.020754	0.13528 ± 0.017375	0.91583 ± 0.038688	n/a
Reinit. Back Propagation	0.094176 ± 0.00101	0.139925 ± 0.003195	1.55321 ± 0.078108	n/a
Reinitialising PSO	0.061455 ± 0.001925	0.062192 ± 0.002846	1.01317 ± 0.05285	5.98567 ± 0.570052
Charged PSO	0.072534 ± 0.005605	0.073516 ± 0.006696	1.00714 ± 0.062739	6.3813 ± 0.585326
Quantum PSO	0.070514 ± 0.004996	0.071295 ± 0.005128	1.00987 ± 0.071053	6.39708 ± 0.736462
	B2 *(frequency: 50, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.149287 ± 0.017802	0.132267 ± 0.014432	0.913431 ± 0.063787	n/a
Reinit. Back Propagation	0.093804 ± 0.001313	0.139525 ± 0.004149	1.55554 ± 0.104874	n/a
Reinitialising PSO	0.06368 ± 0.001258	0.065144 ± 0.002031	1.0333 ± 0.079781	5.30513 ± 0.623714
Charged PSO	0.071221 ± 0.008886	0.072218 ± 0.00837	1.01254 ± 0.045516	6.54465 ± 0.58564
Quantum PSO	0.068427 ± 0.004953	0.069381 ± 0.00564	0.991213 ± 0.061567	6.59418 ± 0.739838
	B3 *(frequency: 50, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.132411 ± 0.011119	0.115916 ± 0.010667	0.905601 ± 0.068225	n/a
Reinit. Back Propagation	0.083283 ± 0.003031	0.11243 ± 0.004419	1.3706 ± 0.119529	n/a
Reinitialising PSO	0.058312 ± 0.00091	0.058575 ± 0.002254	1.01552 ± 0.076406	5.3119 ± 0.550175
Charged PSO	0.059298 ± 0.003277	0.059405 ± 0.00457	1.0041 ± 0.067167	8.84716 ± 0.816739
Quantum PSO	0.058643 ± 0.00301	0.060084 ± 0.003505	1.01826 ± 0.079305	8.69489 ± 0.791932
	B4 *(frequency: 50, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.085928 ± 0.029466	0.08665 ± 0.031776	1.01553 ± 0.074648	n/a
Reinit. Back Propagation	0.066596 ± 0.005977	0.064456 ± 0.007366	0.949584 ± 0.121842	n/a
Reinitialising PSO	0.024424 ± 0.001652	0.024827 ± 0.003369	1.00775 ± 0.133446	6.26526 ± 1.06991
Charged PSO	0.024341 ± 0.020914	0.025131 ± 0.020715	1.03728 ± 0.131702	9.74551 ± 0.925972
Quantum PSO	0.022029 ± 0.004917	0.022903 ± 0.005061	1.04336 ± 0.137933	9.37184 ± 0.971857

Table 6.47 summarises the mean fitness results obtained under scenarios B1 to B4, as well as the average ρ_F values, and average swarm diversity, where applicable. As the ρ_F values in Table 6.47 indicate, all dynamic PSOs exhibited minor overfitting under B1 to B4. BP did not overfit under scenarios B1 to B3. However, the standard deviation values of E_T and E_G obtained by BP indicate that BP showed less robust performance than RBP and the dynamic PSOs.

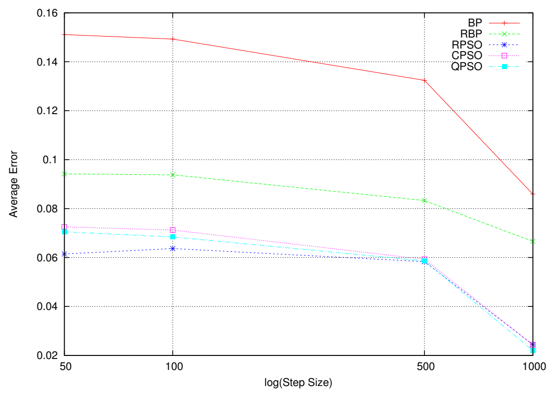
The algorithms were ranked in terms of collective mean E_T and E_G values obtained for B1 to B4, taking the p-values in Tables 6.44 and 6.45 into account. The obtained ranks are listed in Table 6.48. Collective mean E_T and E_G values are illustrated in Figure 6.65. Figure 6.65 illustrates that BP was significantly outperformed by all other algorithms in terms of E_T. BP was also significantly outperformed by all algorithms except RBP under B1 and B2 in terms of E_G, and by all algorithms under B3 and B4. Table 6.48 confirms that the three dynamic PSOs significantly outperformed both BP and RBP under B1 to B4. The RPSO managed to significantly outperform both the CPSO and the QPSO: Figure 6.62 illustrates that the RPSO's E_T and E_G peaks were lower than the corresponding CPSO's and QPSO's error peaks. Since the sliding thresholds problem contained more decision boundaries than the other problems considered, and the conflicting boundaries problem was more severe, the ability of the RPSO to quickly "unlearn" stale data proved useful under gradual scenarios. No statistically significant difference was observed between the dynamic PSOs under abrupt scenarios B3 and B4.

Average ranks reported in Table 6.48 indicate that the RPSO was the best performer under B1 to B4. The algorithm's ability to "unlearn" stale information proved crucial for the problem where multiple conflicting boundaries were present.

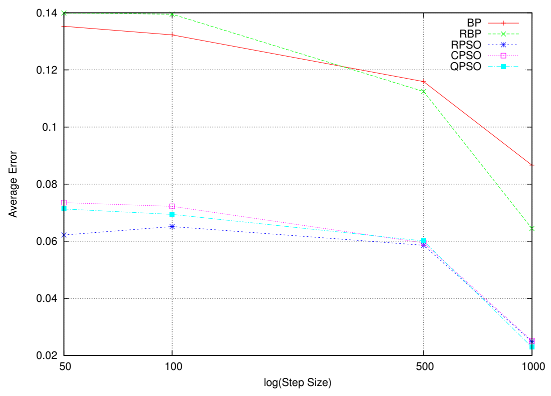
Scenarios C1 to C4: Figures 6.66 and 6.67 illustrate the E_T and E_G profiles over time obtained by the five algorithms for C1 to C4. Figure 6.66 illustrates that CPSO's and QPSO's E_T and E_G peaked higher under gradual scenarios C1 and C2 than under B1 and

Table 6.48: Sliding Thresholds Algorithm Ranking for Scenarios B1 to B4

Algorithm	B1		B2		B3		B4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	4	5	4	5	5	5	5	5	4.5
RBP	4	5	4	5	4	4	4	4	4	4.5
RPSO	1	1	1	1	2	2	3	1.5	1.75	1.375
CPSO	2.5	2.5	2.5	2.5	2	2	1.5	3	2.125	2.5
QPSO	2.5	2.5	2.5	2.5	2	2	1.5	1.5	2.125	2.125



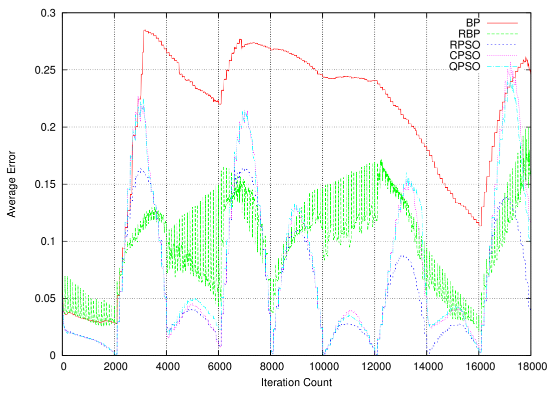
(a) Average Training Error Results



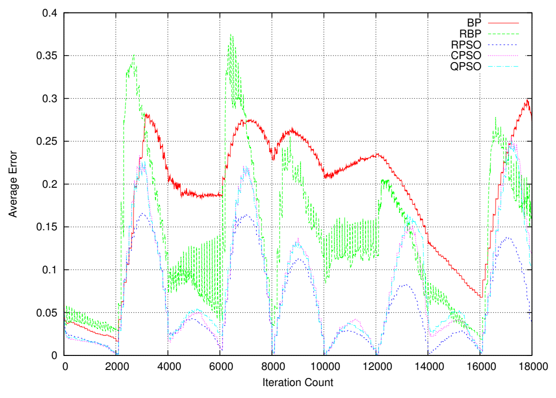
(b) Average Generalisation Error Results

Figure 6.65: Average Error results for Sliding Thresholds, scenarios B1 to B4

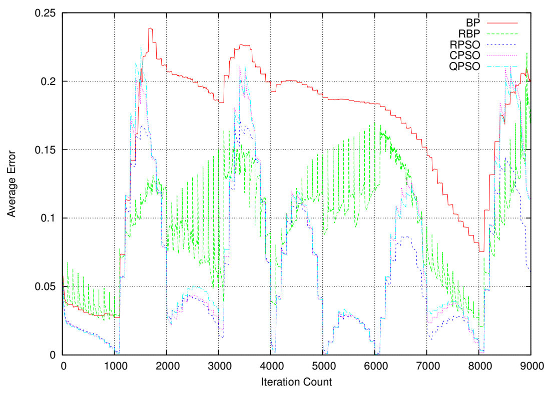
B2 (see Figure 6.62). Scenarios C1 to C2 simulated changes of lower spatial severity than B1 to B4. Thus, the algorithms were given more time to converge between environment changes, and refine the found solutions. However, every environment change yielded some of the previously learned information obsolete, and the algorithms had to promptly adjust the learned model accordingly. Figure 6.68 illustrates that, although both the CPSO and the QPSO preserved diversity throughout the algorithm run under all scenarios, the RPSO reached higher diversity levels under gradual scenarios C1 and C2. The CPSO's diversity was maintained by the repelling forces of the charged particles only, and the QPSO's quantum cloud was randomised in a small radius around the global best, and the gradual changes were easier to adjust to than the abrupt changes. Thus, the CPSO and



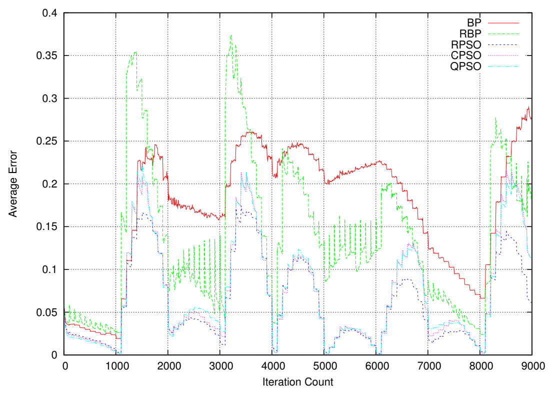
(a) E_T for C1



(b) E_G for C1



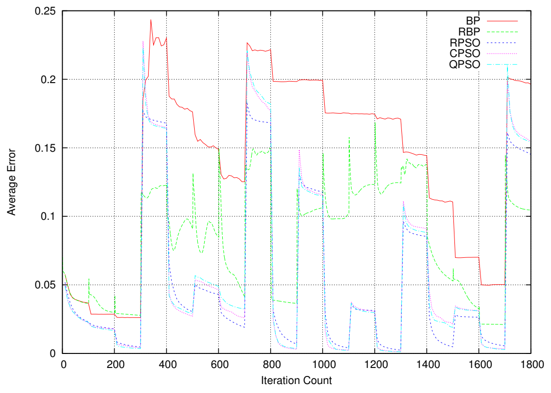
(c) E_T for C2



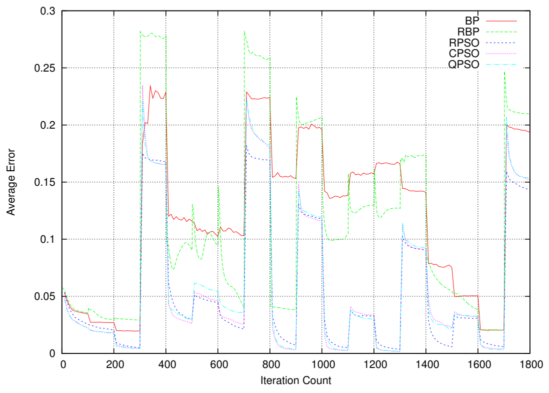
(d) E_G for C2

Figure 6.66: Training and Generalisation Error results for Sliding Thresholds, scenarios C1 to C2

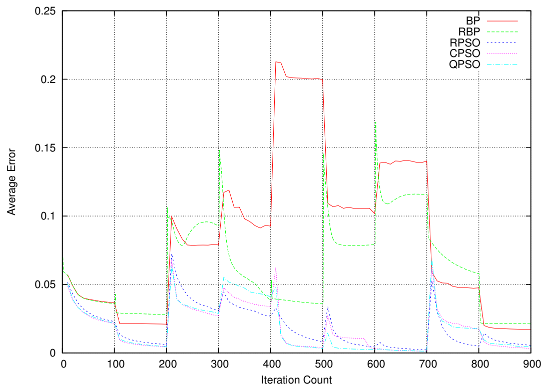
the QPSO converged on a small area around global best, and had to spend more time regaining diversity after a change. The RPSO converged to an even smaller area between environment changes due to the lack of a diversity-preserving mechanism. However, the RPSO regained diversity instantly after every environment change by randomising a percentage of particles in the given weight initialisation interval. In case of the sliding thresholds problem, 75% of the swarm was reinitialised by the RPSO, thus only 25% of the swarm memory was preserved, and the other 75% were scattered around the search space, allowing the RPSO to efficiently explore. Figure 6.68 illustrates that the CPSO



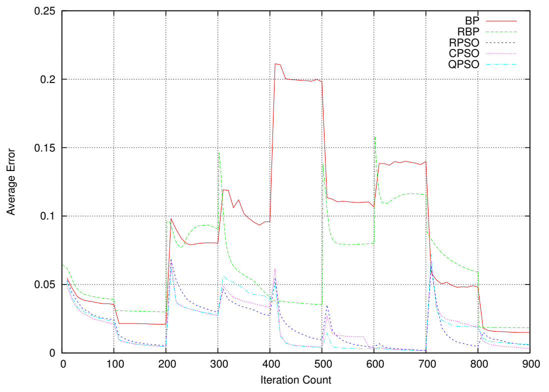
(a) E_T for C3



(b) E_G for C3



(c) E_T for C4



(d) E_G for C4

Figure 6.67: Training and Generalisation Error results for Sliding Thresholds, scenarios C3 to C4

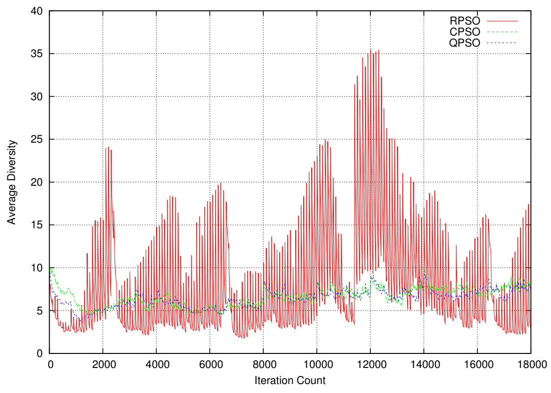
and the QPSO maintained diversity better under abrupt scenarios C3 and C4, where these algorithms struggled to converge to a small stable diversity due to the abruptness of changes.

Figure 6.67 illustrates that neither BP nor RBP has benefited from the decreased temporal severity: both these algorithms stagnated early between most environment changes. The NN error landscape of the sliding thresholds proved hard to optimise for the hill-climbing NN training algorithms.

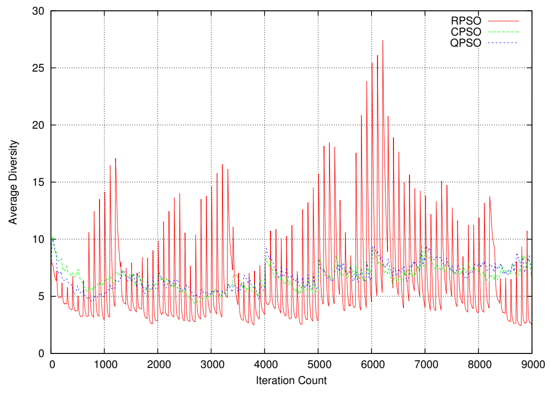
Collective mean E_T, E_G, and ρ_F values, as well as the average swarm diversity was

Table 6.49: Sliding Thresholds Results for Scenarios C1 to C4

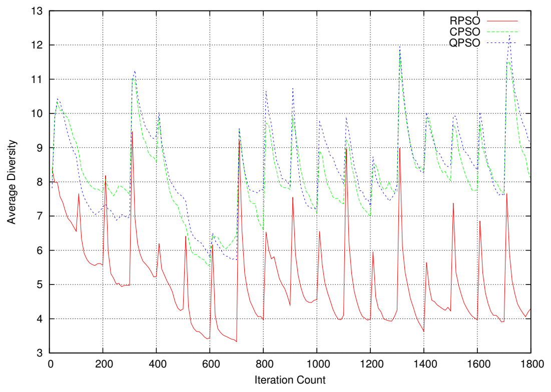
Algorithm \ **Scenario**	**C1** *(frequency: 100, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.200552 ± 0.11802	0.182108 ± 0.112807	0.918696 ± 0.046722	n/a
Reinit. Back Propagation	0.092587 ± 0.00123	0.136819 ± 0.003454	1.54156 ± 0.089565	n/a
Reinitialising PSO	0.059751 ± 0.00178	0.059944 ± 0.002863	1.03563 ± 0.063937	7.59927 ± 1.05103
Charged PSO	0.076702 ± 0.004779	0.078222 ± 0.005915	1.0251 ± 0.063742	6.59631 ± 0.522951
Quantum PSO	0.077463 ± 0.005842	0.079614 ± 0.006852	1.03078 ± 0.057624	6.39715 ± 0.698927
	C2 *(frequency: 100, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.158981 ± 0.038679	0.171066 ± 0.107484	1.06294 ± 0.513781	n/a
Reinit. Back Propagation	0.091914 ± 0.001601	0.137699 ± 0.003701	1.55542 ± 0.079222	n/a
Reinitialising PSO	0.061345 ± 0.002197	0.061962 ± 0.003968	1.02074 ± 0.065403	6.07764 ± 0.667674
Charged PSO	0.071107 ± 0.006099	0.072751 ± 0.00658	1.03272 ± 0.068002	6.64967 ± 0.938283
Quantum PSO	0.072182 ± 0.006226	0.073344 ± 0.00678	1.00832 ± 0.05989	6.71768 ± 0.770835
	C3 *(frequency: 100, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.138621 ± 0.017921	0.120052 ± 0.016573	0.875654 ± 0.058455	n/a
Reinit. Back Propagation	0.082158 ± 0.004326	0.111868 ± 0.005933	1.33506 ± 0.09479	n/a
Reinitialising PSO	0.054092 ± 0.000935	0.055278 ± 0.002278	1.03434 ± 0.093407	5.04578 ± 0.735815
Charged PSO	0.056934 ± 0.003381	0.057185 ± 0.003319	1.0064 ± 0.091524	8.21256 ± 0.799795
Quantum PSO	0.056913 ± 0.004078	0.058309 ± 0.004474	1.02097 ± 0.071611	8.36473 ± 0.84815
	C4 *(frequency: 100, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.08482 ± 0.026541	0.085047 ± 0.026804	0.993225 ± 0.090387	n/a
Reinit. Back Propagation	0.061294 ± 0.00619	0.061548 ± 0.008117	1.02942 ± 0.150179	n/a
Reinitialising PSO	0.018883 ± 0.002162	0.019303 ± 0.003366	1.0102 ± 0.132476	6.06824 ± 1.00838
Charged PSO	0.018289 ± 0.006454	0.018742 ± 0.007034	1.00383 ± 0.128576	8.86321 ± 1.00175
Quantum PSO	0.018509 ± 0.004962	0.018866 ± 0.005452	1.01503 ± 0.149493	8.81854 ± 1.07132



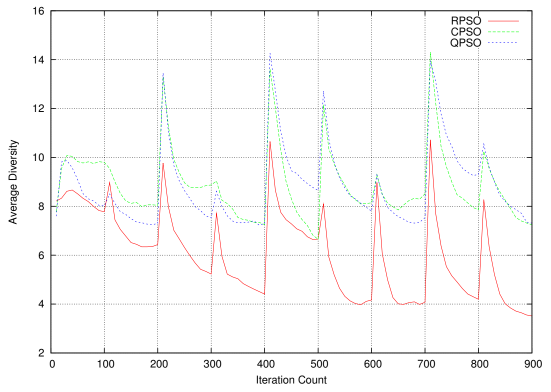
(a) Average Diversity Results for Scenario C1



(b) Average Diversity Results for Scenario C2



(c) Average Diversity Results for Scenario C3



(d) Average Diversity Results for Scenario C4

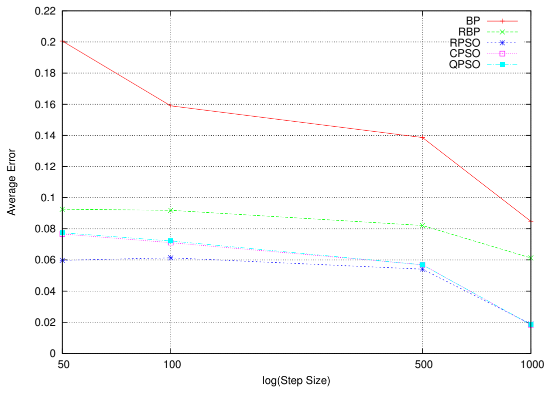
Figure 6.68: Average Diversity results for Sliding Thresholds, scenarios C1 to C4

reported in Table 6.49. The ρ_F values in Table 6.49 indicate that all algorithms except BP were prone to overfitting under C1 to C4, especially RBP, which produced the largest ρ_F values under C1 to C4. The susceptibility of RBP to overfit under scenarios with conflicting boundaries was further facilitated by low temporal severity, which allowed all algorithms to train for too long under C1 to C4. BP produced a $\rho_F > 1$ only under gradual scenario C2. However, the standard deviations of E_T and E_G obtained by BP indicate that BP performed less robustly than the other algorithms considered.

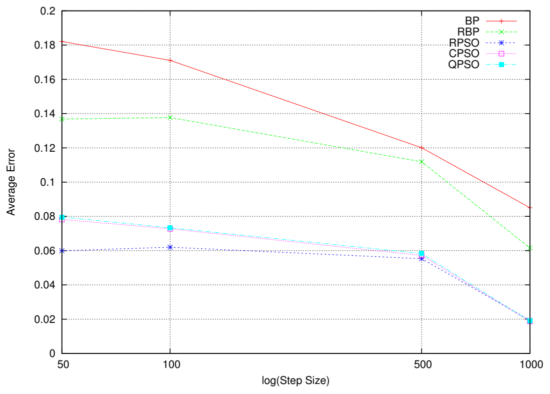
Low average diversity values obtained by the dynamic PSOs under abrupt scenarios C3 and C4 confirm that the dynamic PSO algorithms used the extra iterations between

Table 6.50: Sliding Thresholds Algorithm Ranking for Scenarios C1 to C4

Algorithm	C1		C2		C3		C4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	5	5	5	5	5	5	5	5	5
RBP	4	4	4	4	4	4	4	4	4	4
RPSO	1	1	1	1	1	1	2	2	1.25	1.25
CPSO	2.5	2.5	2.5	2.5	2.5	2.5	2	2	2.375	2.375
QPSO	2.5	2.5	2.5	2.5	2.5	2.5	2	2	2.375	2.375



(a) Average Training Error Results

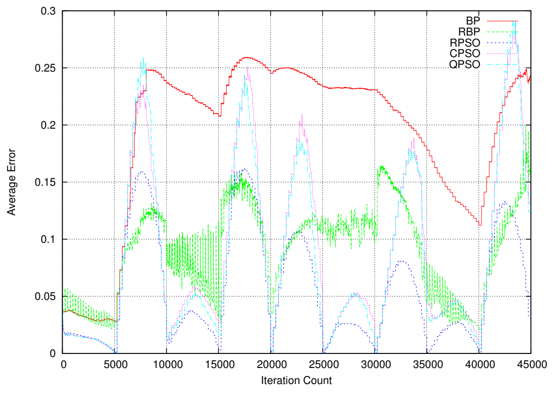


(b) Average Generalisation Error Results

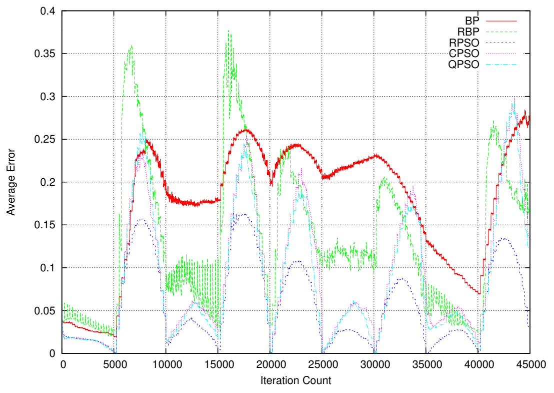
Figure 6.69: Average Error results for Sliding Thresholds, scenarios C1 to C4

changes to converge to a smaller area around a solution.

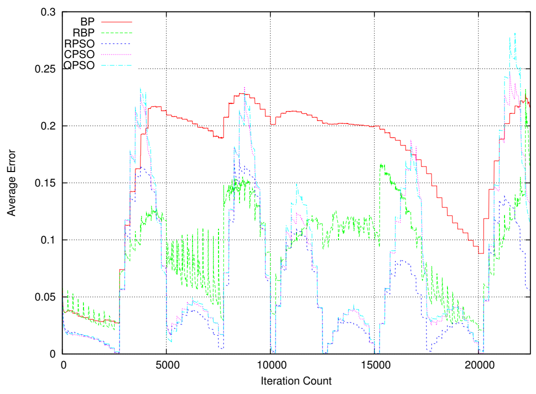
The algorithms were ranked based on the collective mean E_T and E_G values, taking the p-values in Tables 6.44 and 6.45 into account. The obtained ranks, as well as the average ranks, are reported in Table 6.50. Collective mean E_T and E_G values are also illustrated in Figure 6.69. Figure 6.69 and Table 6.50 show that BP was significantly outperformed by all other algorithms under C1 to C4. RBP outperformed BP, but was significantly outperformed by the three dynamic PSOs under C1 to C4. The RPSO significantly outperformed the CPSO and the QPSO under scenarios C1 to C3, and no statistically significant difference was observed between the three dynamic PSOs under C4: as already observed, weak dependence on the swarm memory of the RPSO helped



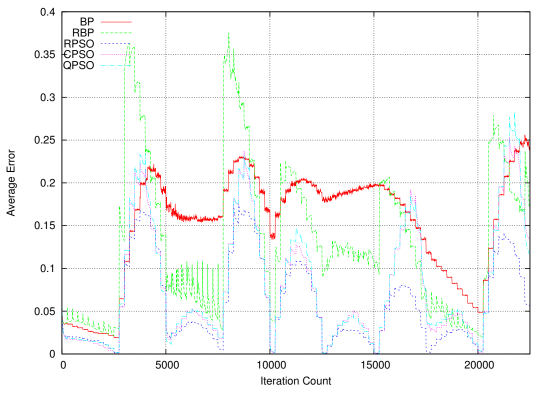
(a) E_T for D1



(b) E_G for D1



(c) E_T for D2

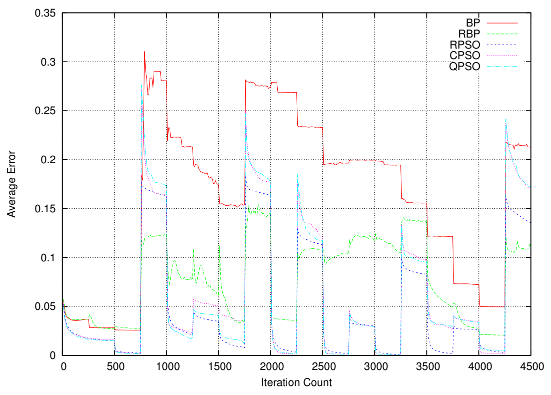


(d) E_G for D2

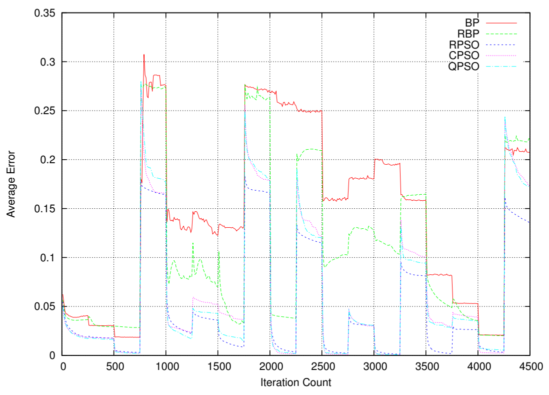
Figure 6.70: Training and Generalisation Error results for Sliding Thresholds, scenarios D1 to D2

this algorithm to promptly "forget" stale data, which proved useful on a problem where multiple conflicting boundaries were present. The average ranks in Table 6.50 indicate that the RPSO was the best performer under C1 to C4, and BP showed the worst performance.

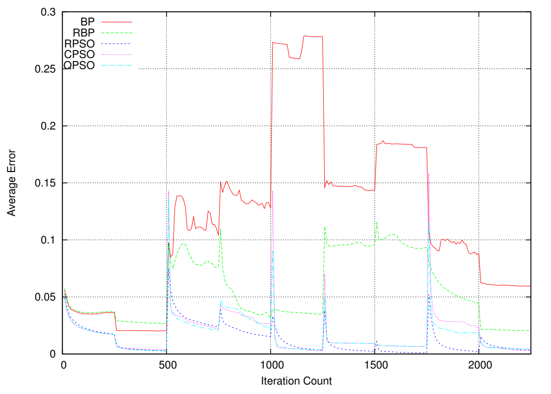
Scenarios D1 to D4: Figures 6.70 and 6.71 illustrate the progression of E_T and E_G over time as obtained by the different algorithms under D1 to D4. Figures 6.70 illustrates that the CPSO's and the QPSO's error profiles peaked even higher under



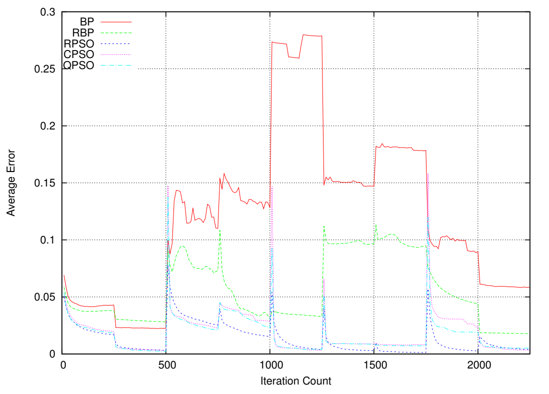
(a) E_T for D3



(b) E_G for D3



(c) E_T for D4

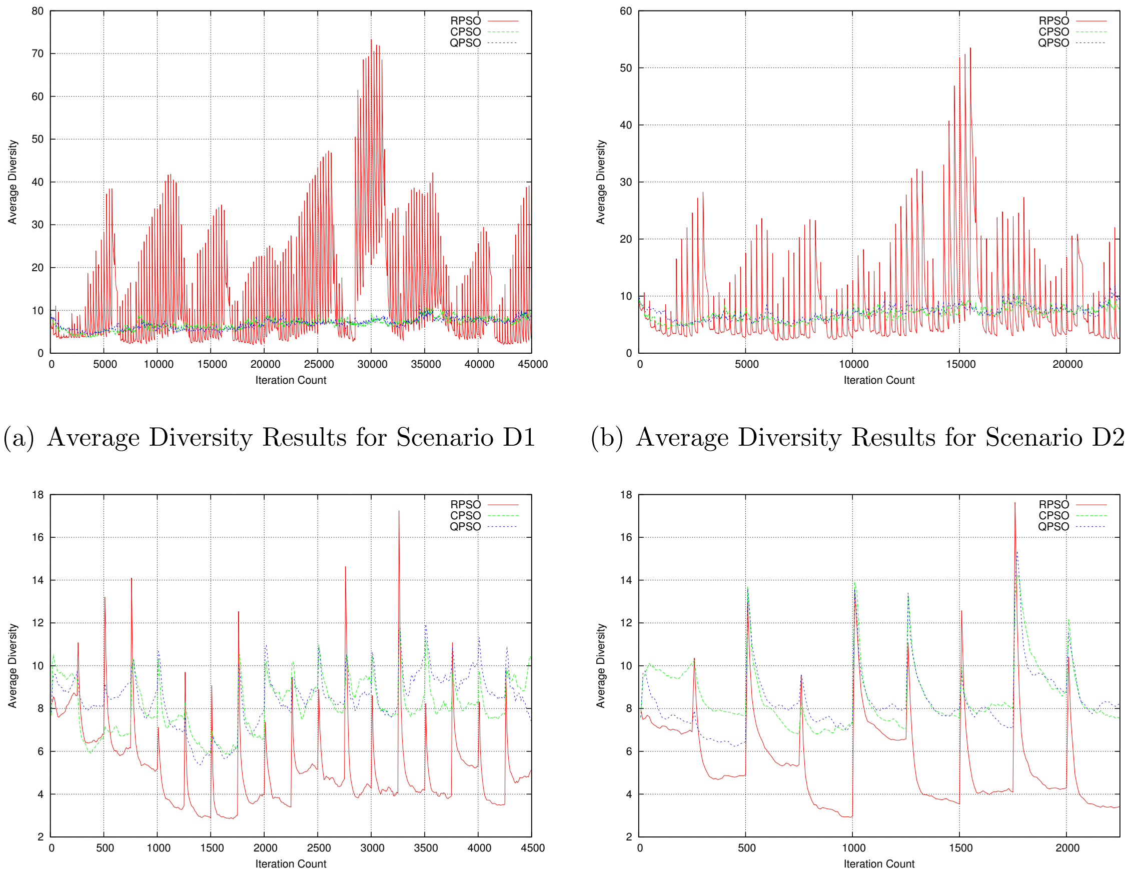


(d) E_G for D4

Figure 6.71: Training and Generalisation Error results for Sliding Thresholds, scenarios D3 to D4

gradual scenarios D1 and D2 than under C1 to C2. Scenarios D1 to D2 simulated the least temporally severe environment changes, with the frequency of changes equal to 250 iterations: memory-preserving CPSO and QPSO exhibited overfitting due to training for too long between environment changes. The RPSO remained overfitting-resistant under D1 to D4.

Under abrupt changes, as Figure 6.71 illustrates, the CPSO and the QPSO also suffered from overfitting: higher error peaks were produced after environment changes under D3 and D4 than under C3 and C4. However, when no conflicting boundaries were



(a) Average Diversity Results for Scenario D1 (b) Average Diversity Results for Scenario D2

(c) Average Diversity Results for Scenario D3 (d) Average Diversity Results for Scenario D4

Figure 6.72: Average Diversity results for Sliding Thresholds, scenarios D1 to D4

present (scenario D4), both the CPSO and the QPSO managed to promptly find and exploit the new solution (refer to Figures 6.71(c) and 6.71(d)).

Figures 6.70 and 6.71 illustrate that BP and RBP exhibited similar behaviour under D1 to D4 as under C1 to C4.

The progression of swarm diversity over time, illustrated in Figure 6.72, shows that all three dynamic PSOs reached a stable diversity level between environment changes. Figure 6.72 illustrates that the dynamic PSOs preserved diversity without exhibiting divergent behaviour.

Table 6.51 summarises the collective mean E_T and E_G, ρ_F, and swarm diversity val-

Table 6.51: Sliding Thresholds Results for Scenarios D1 to D4

Scenario / Algorithm	**D1** *(frequency: 250, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.190979 ± 0.116883	0.174311 ± 0.114016	0.912823 ± 0.049653	n/a
Reinit. Back Propagation	0.089317 ± 0.001316	0.134207 ± 0.003126	1.57525 ± 0.086802	n/a
Reinitialising PSO	0.056782 ± 0.001141	0.057856 ± 0.002018	1.02831 ± 0.064099	11.6523 ± 2.22066
Charged PSO	0.089642 ± 0.009767	0.093216 ± 0.011032	1.04981 ± 0.069186	6.81867 ± 0.627124
Quantum PSO	0.086245 ± 0.005966	0.087575 ± 0.006176	1.03874 ± 0.065755	6.79253 ± 0.852297
	D2 *(frequency: 250, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.16679 ± 0.082548	0.150545 ± 0.081005	0.901706 ± 0.060186	n/a
Reinit. Back Propagation	0.088831 ± 0.001458	0.133354 ± 0.004312	1.57991 ± 0.118443	n/a
Reinitialising PSO	0.058099 ± 0.00136	0.058946 ± 0.002143	1.02881 ± 0.063062	7.3771 ± 1.28765
Charged PSO	0.076492 ± 0.005072	0.078749 ± 0.005938	1.01142 ± 0.057068	6.80669 ± 0.747515
Quantum PSO	0.078824 ± 0.00572	0.08096 ± 0.005934	1.02818 ± 0.068527	7.10211 ± 0.917528
	D3 *(frequency: 250, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.161774 ± 0.044374	0.144836 ± 0.043317	0.920468 ± 0.076284	n/a
Reinit. Back Propagation	0.078363 ± 0.004173	0.107945 ± 0.005508	1.35879 ± 0.079319	n/a
Reinitialising PSO	0.049395 ± 0.000909	0.04987 ± 0.001521	1.03178 ± 0.086033	5.14095 ± 0.855502
Charged PSO	0.059674 ± 0.007093	0.061183 ± 0.007908	1.023 ± 0.078771	8.09304 ± 0.92506
Quantum PSO	0.058129 ± 0.006498	0.059113 ± 0.007565	1.03152 ± 0.089961	8.44183 ± 0.869567
	D4 *(frequency: 250, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.118543 ± 0.070682	0.120698 ± 0.071051	1.08199 ± 0.090834	n/a
Reinit. Back Propagation	0.055971 ± 0.006565	0.055365 ± 0.008704	1.00602 ± 0.118194	n/a
Reinitialising PSO	0.012676 ± 0.000938	0.013223 ± 0.001815	1.0223 ± 0.174777	5.50318 ± 0.954823
Charged PSO	0.017977 ± 0.009832	0.018939 ± 0.009866	1.07967 ± 0.183565	8.58003 ± 1.12658
Quantum PSO	0.016372 ± 0.00934	0.016812 ± 0.009346	1.0556 ± 0.172341	8.32965 ± 0.875176

Table 6.52: Sliding Thresholds Algorithm Ranking for Scenarios D1 to D4

Algorithm	D1		D2		D3		D4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	5	5	5	5	5	5	5	5	5	5
RBP	3.5	4	4	4	4	4	4	4	3.875	4
RPSO	1	1	1	1	1	1	2	2	1.25	1.25
CPSO	3.5	2.5	2.5	2.5	2.5	2.5	2	2	2.625	2.375
QPSO	2	2.5	2.5	2.5	2.5	2.5	2	2	2.25	2.375

ues obtained by the algorithms under D1 to D4. Table 6.51 shows that once again all algorithms except BP were susceptible to minor overfitting under D1 to D3. However, BP produced the highest ρ_F value under abrupt scenario D4. Thus, BP was most susceptible to overfitting given enough iterations between changes, even when no conflicting boundaries were present.

Table 6.52 lists the algorithm ranks based on collective mean E_T and E_G values, taking the p-values reported in Tables 6.44 and 6.45 into account. Table 6.52 shows that the algorithms exhibited similar trends under D1 to D4 as under C1 to C4, and the average ranks indicate that the RPSO was again the best, while BP came last.

All scenarios: The average training errors produced by the four algorithms under the 16 different dynamic scenarios are illustrated in Figure 6.73, and the average generalisation errors are illustrated in Figure 6.74.

Figure 6.73 illustrates that BP produced inferior E_T values compared to all other algorithms under all scenarios except the most abrupt and the least temporally severe scenario A4. Environment changes exhibited under A4 were abrupt and frequent enough to prevent BP from getting trapped in an unfruitful area of the search space. Both training and generalisation performance of BP, as figures 6.73 and 6.74 illustrate, deteriorated as the changes became less temporally or spatially severe. Any exploitable changes led BP to such regions of the error landscape from which BP could not escape

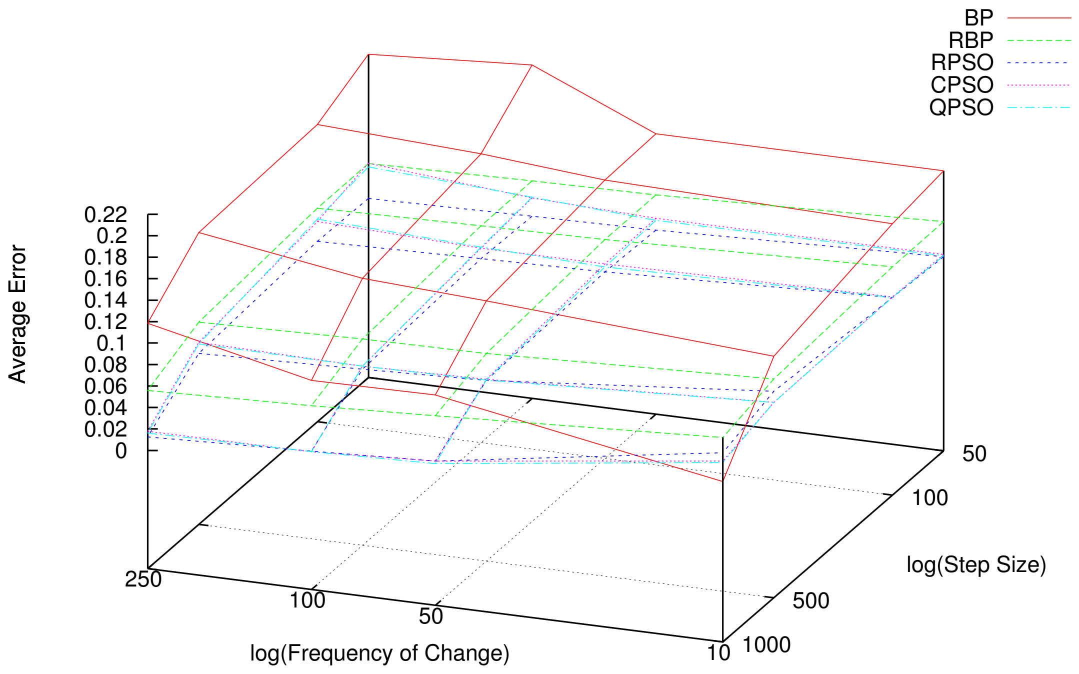


Figure 6.73: Average Training Error Results for Sliding Thresholds

due to its hill-climbing approach to NN training. RBP performed significantly better than BP, especially under changes of lower temporal severity: complete reinitialisation of weights required more iterations for the algorithm to converge on a solution.

Figures 6.73 and 6.74 illustrate that the dynamic PSOs significantly outperformed both BP and RBP in terms of both E_T and E_G under all scenarios except the most abrupt and temporally severe scenario A4, where BP was the top performer. Population-based algorithms converge slower than hill-climbing algorithms, thus the dynamic PSOs required more algorithm iterations to converge around a solution than BP. Under scenario A4, the exhibited changes were too drastic for the dynamic PSOs to be able to adapt to in 10 iterations.

Figures 6.73 and 6.74 illustrate that the RPSO was outperformed by the CPSO and

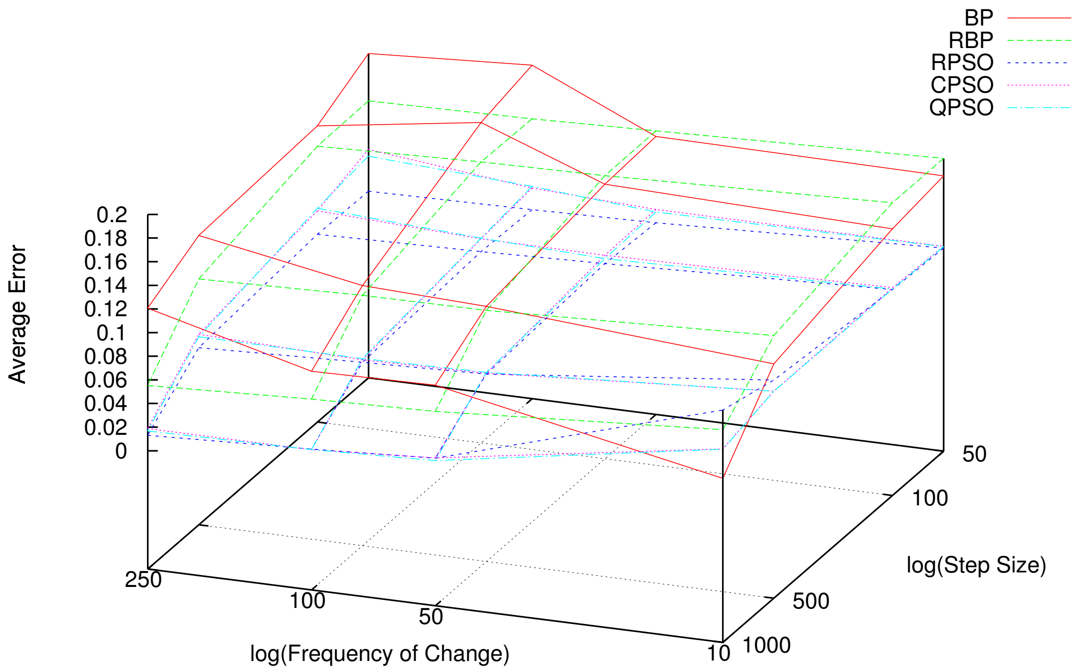


Figure 6.74: Average Generalisation Error Results for Sliding Thresholds

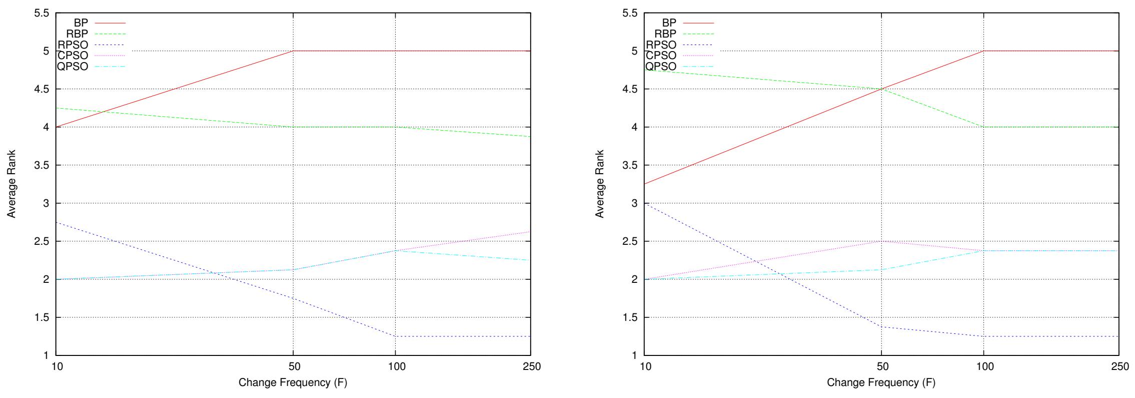
the QPSO under frequent abrupt changes, and outperformed both the CPSO and the QPSO under gradual infrequent changes. The diversity of the RPSO fluctuated more severely than the diversity of the CPSO and the QPSO, which prevented the RPSO from efficiently exploiting under A4, and at the same time made the RPSO more resistant to overfitting than both the CPSO and the QPSO under infrequent gradual changes.

The average ranks of the algorithms obtained for scenarios A1 to A4, B1 to B4, C1 to C4, and D1 to D4, as well as the overall average algorithm ranks, are reported in Table 6.53. The progression of algorithm ranks over different change frequencies is illustrated in Figure 6.75.

Figure 6.75 illustrates that BP, the CPSO, and the QPSO deteriorated in ranking as the temporal severity decreased. The RPSO and RBP, on the other hand, improved their

Table 6.53: Sliding Thresholds Average Algorithm Ranking for Scenarios A to D

Algorithm	Average R(A)		Average R(B)		Average R(C)		Average R(D)		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	3.25	5	4.5	5	5	5	5	4.75	4.4375
RBP	4.25	4.75	4	4.5	4	4	3.875	4	4.03125	4.3125
RPSO	2.75	3	1.75	1.375	1.25	1.25	1.25	1.25	1.75	1.71875
CPSO	2	2	2.125	2.5	2.375	2.375	2.625	2.375	2.28125	2.3125
QPSO	2	2	2.125	2.125	2.375	2.375	2.25	2.375	2.1875	2.21875



(a) Average Rank in terms of E_T (b) Average Rank in terms of E_G

Figure 6.75: Average Rank results for Sliding Thresholds

ranks with an increase in temporal severity. Both RBP and RPSO completely or partially restarted the search for an optimal solution after every environment change, thus these algorithms required a larger number of iterations to exploit any fruitful regions found than the algorithms that retained more memory about the previous solution. BP, the CPSO, and the QPSO relied on previously discovered information, and thus performed their best under frequent changes. However, infrequent changes allowed these algorithms to over-exploit, thus yielding overfitting behaviour.

The overall average ranks in Table 6.53 indicate that BP showed the worst performance on the sliding thresholds problem, and the RPSO showed the best performance compared to the other algorithms considered. The QPSO came second best, and the CPSO came third in terms of the average E_T and E_G rank. The CPSO was the only dynamic PSO considered which did not reinitialise particles, relying instead on the diversity-preserving forces of the charged particles. However, the sliding thresholds problem required the training algorithms to deal with a larger number of decision boundaries than the other problems considered, thus the ability to promptly "unlearn" the stale information was crucial. The CPSO was the most memory-dependent, and the RPSO was least memory-dependent; thus the CPSO was the least successful, and the RPSO was the most successful dynamic PSO approach to the sliding thresholds problem.

6.2.5 Electricity Pricing

This problem used a real-life data set based on the electricity market in the Australian state of New South Wales. The data set was adopted from [45].

Domain Description

Prices in the electricity market are determined by matching the present demand for electricity with the least expensive combination of electricity from all available power stations. Both market prices and electricity price schedules published by each power station are frequently recalculated and updated.

Market prices depend on both demand and supply of electrical power. Significant factors affecting the demand are season, weather, time of day and central business district population density. The supply is affected mainly by the number of active electricity generators. Thus, this environment is subject to both regular long-term changes, such as seasonal changes, and irregular short-term changes, such as weather fluctuations [45].

Problem Definition

A dynamic classification problem was constructed based on the changing electricity market price. Six parameters on which the price is dependent were identified. These include day of week, time of day, and electricity demand estimates. Parameter values were recorded every half an hour, from 7 May 1996 to 5 December 1998. In this manner, a data set of 27 552 samples was recorded.

Each pattern was labelled as either class A or class B. The class label identified whether the current price is higher (class A) or lower (class B) than a moving average price over the last 24 hours. The task of the NN was thus to predict whether the price will go up or down based on the given input values.

Dynamic environments were simulated by sliding a window over the data set. The original temporal order of the data was preserved, and the patterns were shuffled only inside the window. The size of the window was set to 1000 patterns. A NN with 6 input units, 6 hidden units and 1 output unit was trained on the electricity pricing problem. According to equation (6.3), the total number of weights and biases, corresponding to the dimensionality of the problem, was equal to 56.

Sixteen different dynamic scenarios as described in Section 6.2.2 and outlined in Table 6.17 were applied to the electricity pricing problem.

Parameter Optimisation

All algorithm parameters were optimised according to the procedure described in Section 6.1.3. Corresponding optimal parameters discovered are shown in Table 6.54.

Analysis of Empirical Data

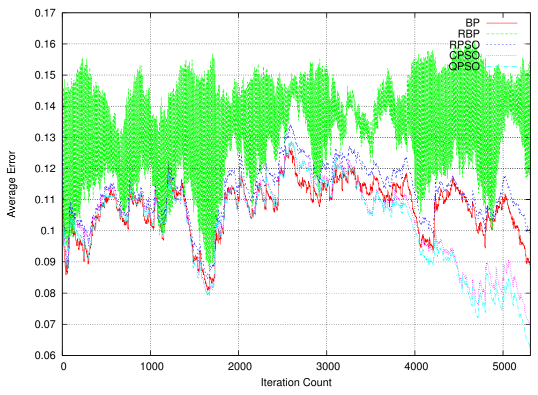
The number of iterations required to traverse the entire data set under every dynamic scenario considered was calculated using equation (6.4). The number of patterns was equal to 27 552, the window size was fixed to 1000 patterns, and both the step size and the frequency of changes were determined by the scenario in use.

Table 6.54: Optimal Parameters for the Electricity Pricing problem

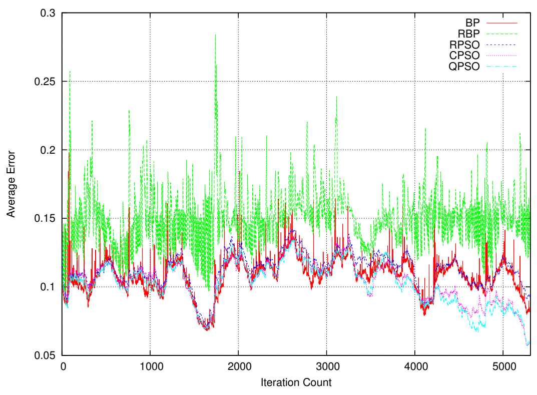
Algorithm	**Parameters**			
Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-1, 1]$	0.1	0.7	
Reinitialising Back Propagation	Weight Interval	Learning Rate	Momentum	
	$[-1, 1]$	0.1	0.7	
Reinitialising PSO	Weight Interval	V_{max}	Swarm Size	Reinitialisation Ratio
	$[-1, 1]$	2	30	0.25
Charged PSO	Weight Interval	V_{max}	Swarm Size	Charge Magnitude
	$[-1, 1]$	5	30	0.1
Quantum PSO	Weight Interval	V_{max}	Swarm Size	Cloud Radius
	$[-3, 3]$	5	50	1.5

Scenarios A1 to A4: Figures 6.76 and 6.77 illustrate the progression of the training and generalisation error over time as obtained by the algorithms under scenarios A1 to A4. Figures 6.76(a) and 6.76(b) illustrate that the CPSO and the QPSO exploited better than the other algorithms considered under gradual scenario A1. Scenarios A1 to A4 simulated frequent changes (every 10 iterations). Hence, the ability to efficiently track the found optima was more important than the ability to find new optima under the most gradual scenario A1. The dynamic PSOs showed their ability to exploit better than BP and RBP on other problems considered, and both the CPSO and the QPSO retain memory better than RPSO; thus, the CPSO and the QPSO have a stronger ability to track the moving optima.

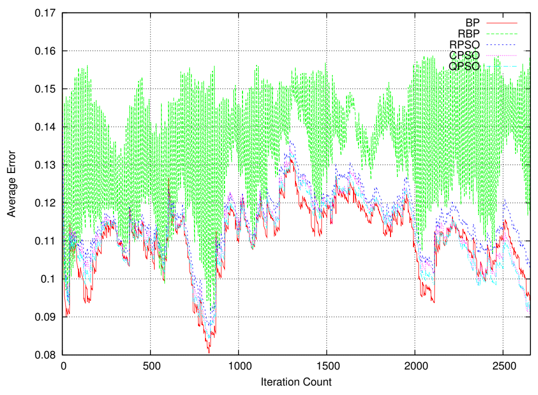
Figure 6.76 illustrates that BP's and RBP's E_G errors peaked higher after gradual environment changes than the corresponding dynamic PSO errors. Both BP and RBP



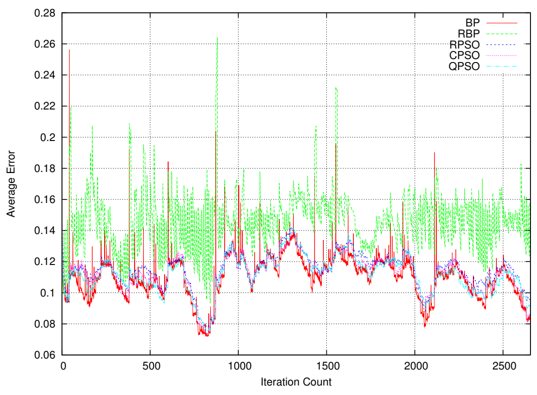
(a) E_T for A1



(b) E_G for A1



(c) E_T for A2



(d) E_G for A2

Figure 6.76: Training and generalisation error results for Electricity Pricing, scenarios A1 to A2

moved down the steepest slope of the error function gradient from a single point in the weight space. The dynamic PSOs, on the other hand, used a swarm of potential solutions, i.e. weight vectors represented by particles. Therefore, a number of different weight vectors were considered at every iteration, and the dynamic PSOs had a higher chance of instantly locating a better, more up-to-date solution just by re-evaluating every particle's fitness after an environment change and choosing a new global best particle from the swarm.

Figures 6.76 and 6.77 illustrate that RBP fluctuated a lot in terms of both E_T and

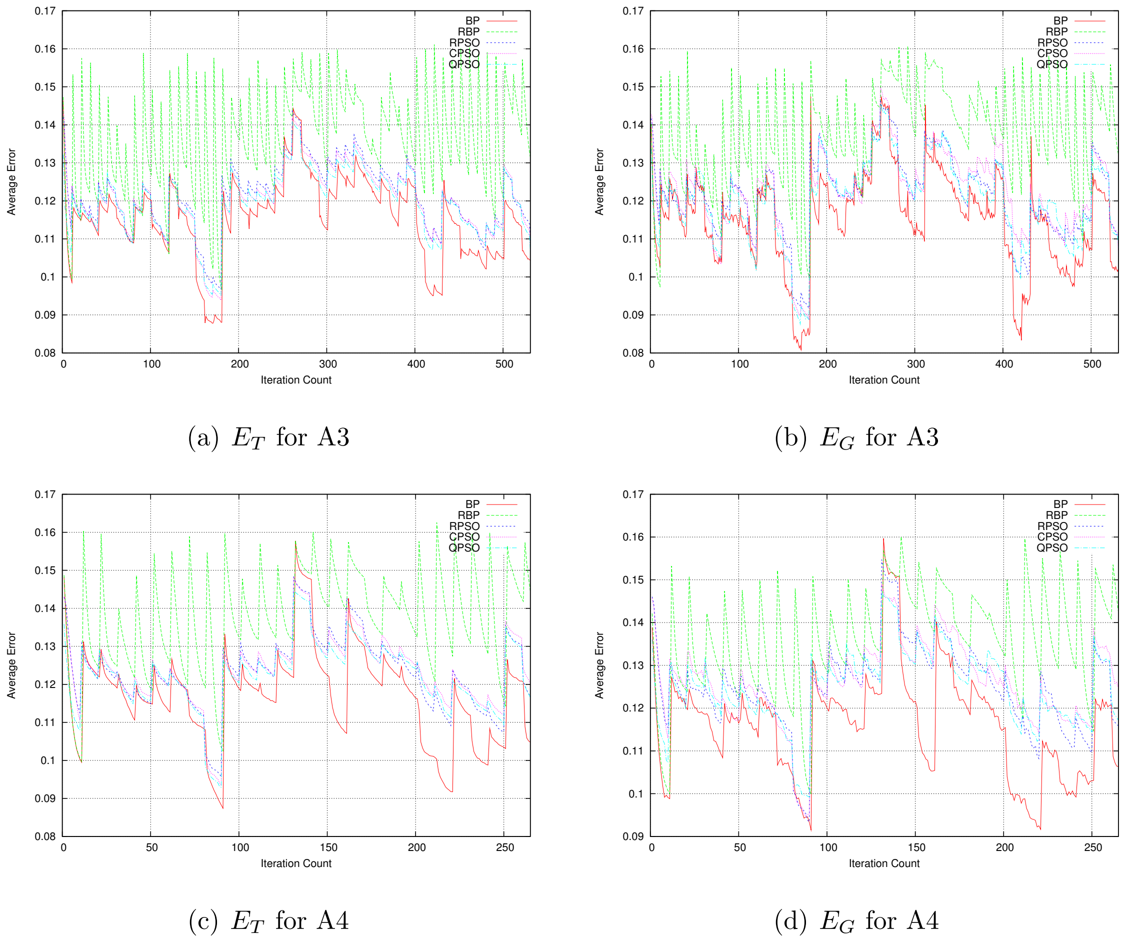
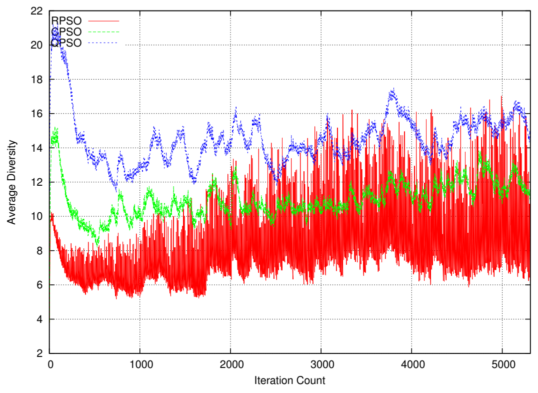


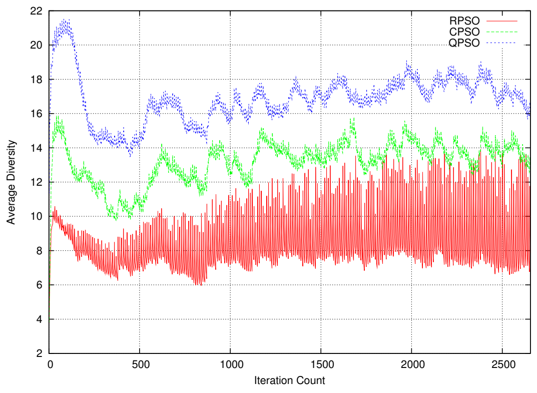
Figure 6.77: Training and generalisation error results for Electricity Pricing, scenarios A3 to A4

E_G, and never had a chance to converge on a good solution due to the high frequency of changes: a complete reinitialisation of NN weights was not an effective approach for the electricity pricing problem under scenarios of high temporal severity.

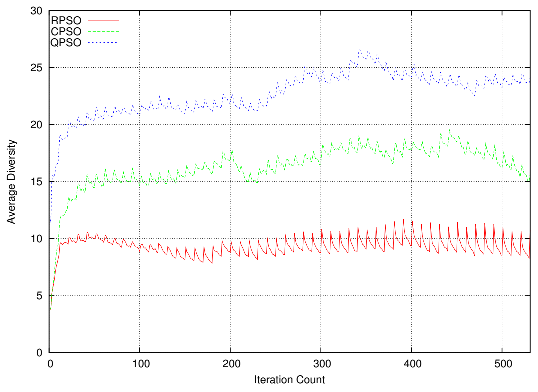
Figure 6.77 illustrates that the dynamic PSOs also struggled to converge on a solution under the change frequency of 10 iterations proved: the E_T and E_G values obtained by the dynamic PSOs showed no sign of stagnation (i.e., the E_T and E_G decreased between the changes), but the dynamic PSOs did not have enough time to converge to similar or better error values than that obtained by BP. The swarm diversity profiles of the



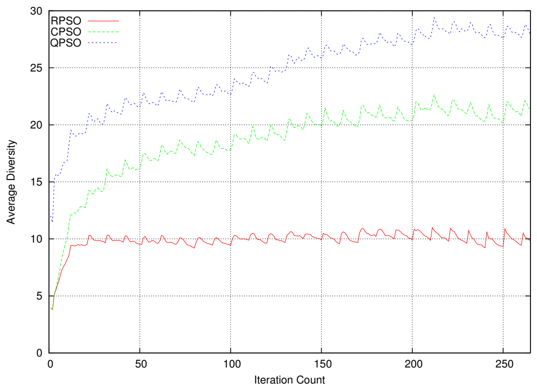
(a) Average Diversity Results for Scenario A1



(b) Average Diversity Results for Scenario A2



(c) Average Diversity Results for Scenario A3



(d) Average Diversity Results for Scenario A4

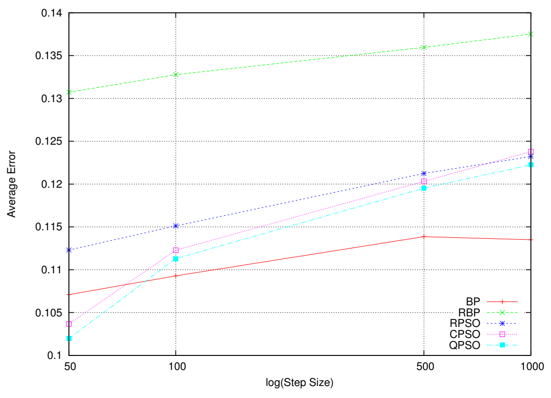
Figure 6.78: Average Diversity results for Electricity Pricing, scenarios A1 to A4

dynamic PSOs, illustrated in Figure 6.78, confirm that the dynamic PSO's diversity did not reach a stable level, and increased under scenarios A1 to A4 for the three dynamic PSOs considered. The RPSO was the only dynamic PSO which did not apply a diversity-preserving technique, hence the RPSO's diversity level was lower than that of the CPSO and the QPSO under abrupt scenarios A3 and A4.

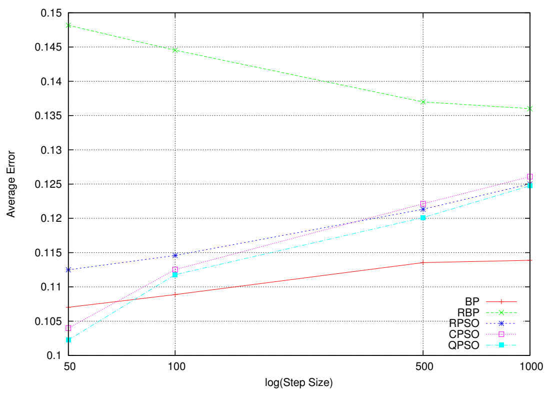
Collective mean E_T and E_G values, ρ_F values, and average swarm diversity values, where applicable, are reported in Table 6.55. The ρ_F values in Table 6.55 show that all algorithms exhibited some overfitting under scenarios A1 to A4, except for BP and RBP, which overfitted under A1 to A3, but produced $\rho_F < 1$ under the most abrupt scenario

Table 6.55: Electricity Pricing Results for Scenarios A1 to A4

Scenario / Algorithm	Training Error	Generalisation Error	ρ_F	Swarm Diversity
	A1 *(frequency: 10, step size: 50)*			
Back Propagation	0.107272 ± 0.001239	0.107447 ± 0.001796	1.00983 ± 0.026576	n/a
Reinit. Back Propagation	0.130724 ± 0.000274	0.148172 ± 0.001327	1.16051 ± 0.016344	n/a
Reinitialising PSO	0.1123 ± 0.000892	0.112489 ± 0.002648	1.0107 ± 0.033401	8.2108 ± 0.845223
Charged PSO	0.103669 ± 0.005592	0.103975 ± 0.005727	1.00974 ± 0.031274	10.95 ± 1.83442
Quantum PSO	0.101964 ± 0.00475	0.102248 ± 0.005023	1.00493 ± 0.030365	14.5312 ± 1.84741
	A2 *(frequency: 10, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.110246 ± 0.001341	0.109761 ± 0.002554	1.00528 ± 0.028872	n/a
Reinit. Back Propagation	0.132773 ± 0.000297	0.144535 ± 0.00125	1.11496 ± 0.019787	n/a
Reinitialising PSO	0.11512 ± 0.001249	0.114578 ± 0.001907	1.00387 ± 0.03096	8.41834 ± 0.602839
Charged PSO	0.112274 ± 0.002701	0.112519 ± 0.003538	1.00778 ± 0.022714	13.1445 ± 1.60485
Quantum PSO	0.111304 ± 0.00308	0.111775 ± 0.003462	1.00433 ± 0.035532	16.8054 ± 1.71678
	A3 *(frequency: 10, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.115424 ± 0.001046	0.115071 ± 0.001993	1.00078 ± 0.027635	n/a
Reinit. Back Propagation	0.135959 ± 0.000425	0.136993 ± 0.001513	1.00626 ± 0.016401	n/a
Reinitialising PSO	0.121239 ± 0.001295	0.121328 ± 0.002863	1.00782 ± 0.025264	9.27923 ± 0.604357
Charged PSO	0.120312 ± 0.001869	0.122126 ± 0.00201	1.01519 ± 0.023311	16.3026 ± 1.63845
Quantum PSO	0.11953 ± 0.001611	0.120109 ± 0.002337	1.00651 ± 0.025085	22.6831 ± 3.15953
	A4 *(frequency: 10, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.11685 ± 0.001113	0.116022 ± 0.003052	0.988718 ± 0.027543	n/a
Reinit. Back Propagation	0.137511 ± 0.000552	0.136005 ± 0.001683	0.98745 ± 0.018722	n/a
Reinitialising PSO	0.123221 ± 0.001571	0.12502 ± 0.002156	1.0163 ± 0.024761	9.87126 ± 0.540778
Charged PSO	0.123785 ± 0.001418	0.126081 ± 0.002805	1.01991 ± 0.025703	18.5897 ± 2.42353
Quantum PSO	0.122252 ± 0.001398	0.124756 ± 0.002274	1.01767 ± 0.025868	24.6344 ± 2.51626



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.79: Average Error results for Electricity Pricing, scenarios A1 to A4

A4. In addition to the typical causes of overfitting, such as training for too long or noise in the training data, dynamic classification problems introduce an extra possible cause of overfitting; namely, the presence of stale data in the sliding window. The fact that the ρ_F value of both BP and RBP decreased as the step sizes of the sliding window increased from 50 to 1000, indicates that smaller step size retained a larger amount of stale data inside the sliding window, and both BP and RBP fitted the stale data.

Increasing average swarm diversity in Table 6.55 indicates that the dynamic PSOs had to explore more as the changes became more abrupt, and that a stable diversity level was not reached by the swarms under frequently severe scenarios A1 to A4.

The algorithms were ranked based on their average E_T and E_G values, taking the p-values in Tables 6.56 and 6.57 into account. The obtained ranks, as well as the average ranks, are reported in Table 6.58. Average E_T and E_G values obtained by the algorithms under A1 to A4 are illustrated in Figure 6.79. Figure 6.79 illustrates that E_T of all algorithms deteriorated with an increase in change abruptness, indicating that abrupt changes were harder to adapt to under the temporal severity simulated by A1 to A4. However, Figure 6.79 illustrates that RBP's E_G decreased as the step size (i.e. spatial severity) increased, even though the training error of all algorithms deteriorated with

Table 6.56: Mann-Whitney U p-values obtained for the average training error comparisons on the Electricity Pricing problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.018329**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.003532**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	0.072293	**0.0001**	**0.0001**	**0.0001**	0.982572	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	0.831957	**0.0001**	**0.0001**	**0.0001**	0.52295
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.012515**	0.074711
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.866438	**0.021582**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.003358**	0.313557
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.082354	**0.001155**	**0.0001**	**0.004103**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.024336**	0.17034	0.19217	0.069934	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.001443**	**0.006965**	0.485326	**0.001092**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.292719	0.62306	0.177408	**0.0001**
D	**0.010038**	**0.0001**	**0.0001**	**0.0001**	0.532582	0.063242	0.082358	0.912747

Table 6.57: Mann-Whitney U p-values obtained for the average generalisation error comparisons on the Electricity Pricing problem with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

	BP vs RBP				BP vs RPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.003034**
	BP vs CPSO				BP vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.010973**	**0.000688**	**0.0001**	**0.0001**	**0.0001**	**0.0017**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	0.099517	**0.0001**	**0.0001**	**0.0001**	0.335329	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	0.953455	**0.0001**	**0.0001**	**0.0001**	0.582035
	RBP vs RPSO				RBP vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
	RBP vs QPSO				RPSO vs CPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.008379**	0.260082	0.112435
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.602392	**0.000312**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.004103**	**0.0001**	**0.016175**	0.224322
D	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.571977	0.241747	**0.0001**	**0.0479**
	RPSO vs QPSO				CPSO vs QPSO			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
A	**0.0001**	**0.0001**	**0.049631**	0.924365	0.230029	0.397957	**0.000426**	0.055132
B	**0.0001**	**0.0001**	0.552109	0.159042	0.854916	0.494588	0.414639	**0.016867**
C	0.11586	**0.000164**	**0.000227**	0.093511	0.142132	0.074711	0.163495	**0.009599**
D	0.096477	0.365817	**0.0001**	**0.043005**	0.552109	1.00578	0.213209	0.592174

Table 6.58: Electricity Pricing Algorithm Ranking for Scenarios A1 to A4

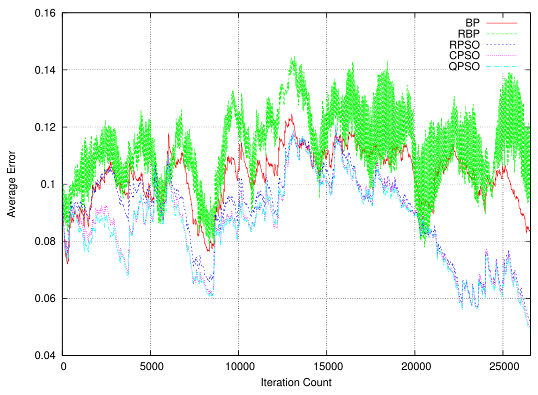
Algorithm	A1		A2		A3		A4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	3	3	1	1	1	1	1	1	1.5	1.5
RBP	5	5	5	5	5	5	5	5	5	5
RPSO	4	4	4	4	4	3.5	3.5	3	3.875	3.625
CPSO	1.5	1.5	2.5	2.5	2.5	3.5	3.5	3	2.5	2.625
QPSO	1.5	1.5	2.5	2.5	2.5	2	2	3	2.125	2.25

increased abruptness. Decrease in RBP's E_G confirms that larger step sizes minimised the amount of stale data inside the sliding window. RBP re-started itself at every iteration, completely erasing all memory of previously found solutions. Hence, RBP failed to discern between up-to-date and stale data.

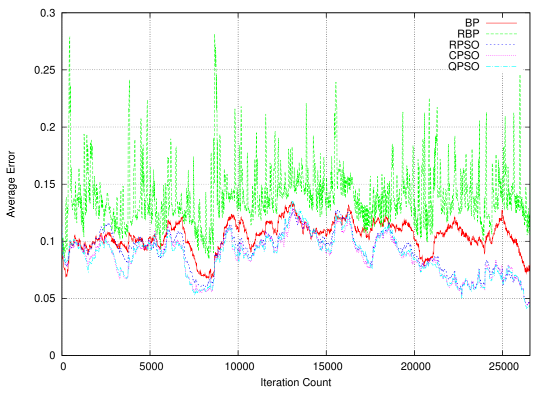
Figure 6.79 illustrates that RBP was outperformed by all other algorithms under scenarios A1 to A4. BP, due to its quicker convergence, outperformed the dynamic PSOs under scenarios A2 to A4. Under the most gradual scenario A1, BP outperformed the RPSO, but was outperformed by the CPSO and the QPSO. Table 6.58 shows that no statistically significant difference was observed between the CPSO and the QPSO under most scenarios, and the RPSO was outperformed by these two algorithms under scenarios A1 to A3. The RPSO performed inferior to the other dynamic PSOs because the RPSO discarded too much of the previously learned information every time the environment changed, which proved inefficient on the electricity pricing problem, where new electricity prices were derived from previous electricity prices.

Average ranks in Table 6.58 indicate that BP was the best performer under A1 to A4, and RBP showed the worst performance. Faster convergence of BP made this algorithm the most efficient approach under the temporally severe scenarios A1 to A4.

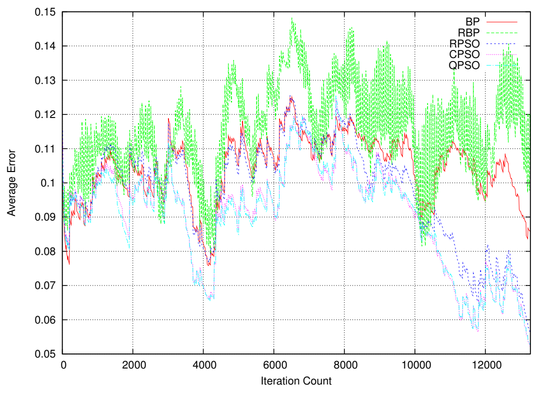
Scenarios B1 to B4: Figures 6.80 and 6.81 illustrate the progression of E_T and E_G over time as obtained by the five algorithms considered under scenarios B1 to B4. Figure



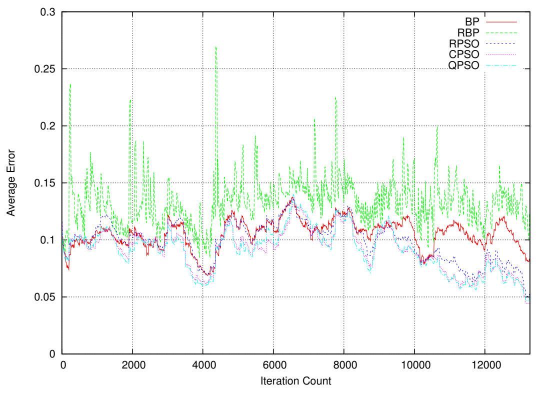
(a) E_T for B1



(b) E_G for B1



(c) E_T for B2

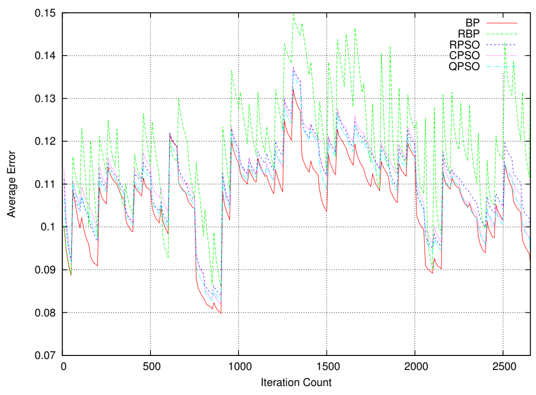


(d) E_G for B2

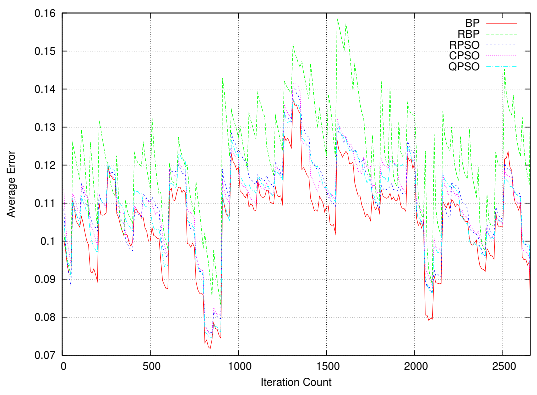
Figure 6.80: Training and Generalisation Error results for Electricity Pricing, scenarios B1 to B2

6.80 illustrates that, given enough time between environment changes (50 instead of 10), the dynamic PSOs managed to outperform both BP and RBP under both B1 and B2. All three dynamic PSOs showed a better ability to exploit under gradual environment changes than BP and RBP. It also becomes visible from Figure 6.80 that RBP fluctuated less as the temporal severity decreased, since RBP had more time to converge between the changes.

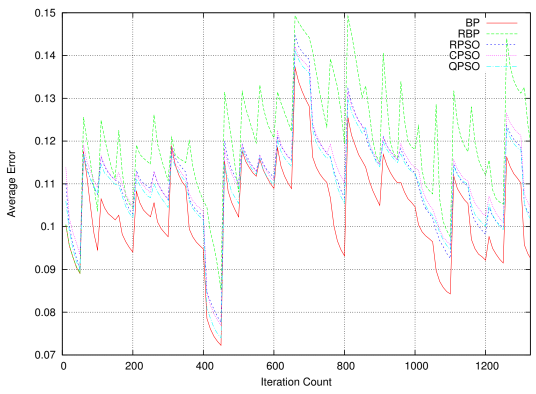
Figure 6.81 illustrates that both RBP and the dynamic PSOs also benefited from the decrease of temporal severity under abrupt scenarios: given more time between the



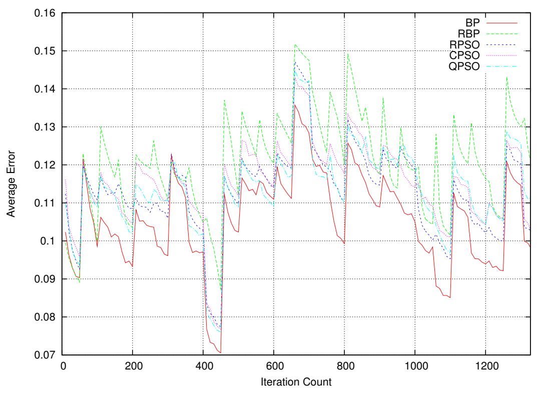
(a) E_T for B3



(b) E_G for B3



(c) E_T for B4

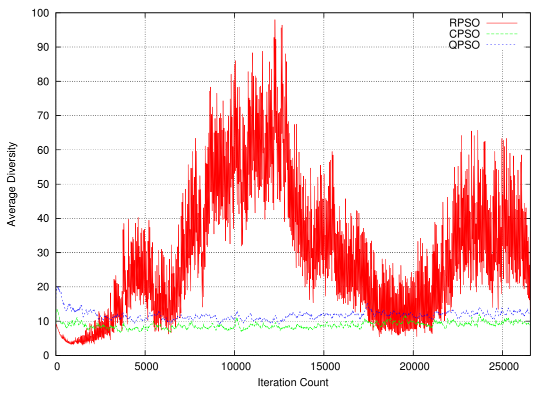


(d) E_G for B4

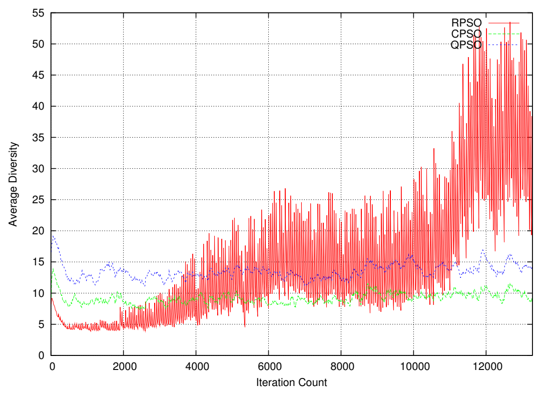
Figure 6.81: Training and Generalisation Error results for Electricity Pricing, scenarios B3 to B4

changes, these algorithms managed to exploit better and reach lower minimum errors than under A3 and A4. Neither RBP nor the dynamic PSOs showed any signs of stagnation, thus further performance improvement is expected with further temporal severity decrease.

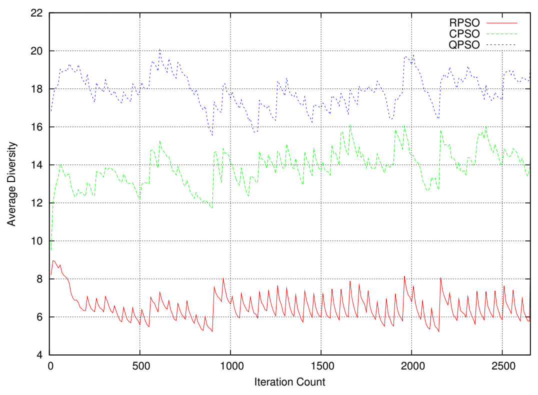
Swarm diversity profiles, illustrated in Figure 6.82, show that both the CPSO and the QPSO reached a stable diversity level under gradual scenarios B1 and B2, and maintained that diversity level throughout the algorithm run. The RPSO's diversity, however, fluctuated violently under B1 and B2. The RPSO has a weak dependence



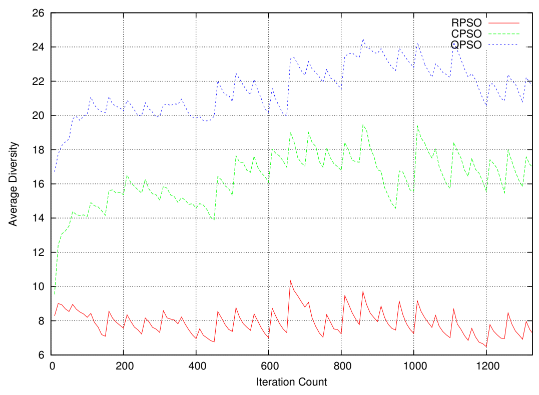
(a) Average Diversity Results for Scenario B1



(b) Average Diversity Results for Scenario B2



(c) Average Diversity Results for Scenario B3



(d) Average Diversity Results for Scenario B4

Figure 6.82: Average Diversity results for Electricity Pricing, scenarios B1 to B4

on memory due to particle reinitialisation, which is good for exploration, but bad for exploitation, and too little exploitation may result in swarm divergence. Indeed, Figure 6.82 illustrates that the diversity of the RPSO not only fluctuated violently, but also reached higher maximum diversity values than under A1 to A2, which may be interpreted as a sign of divergent behaviour. Another possible cause of divergence is the use of the sigmoid activation function in the neurons. This topic, however, is out of the scope of the present study, and is further investigated by van Wyk and Engelbrecht [119].

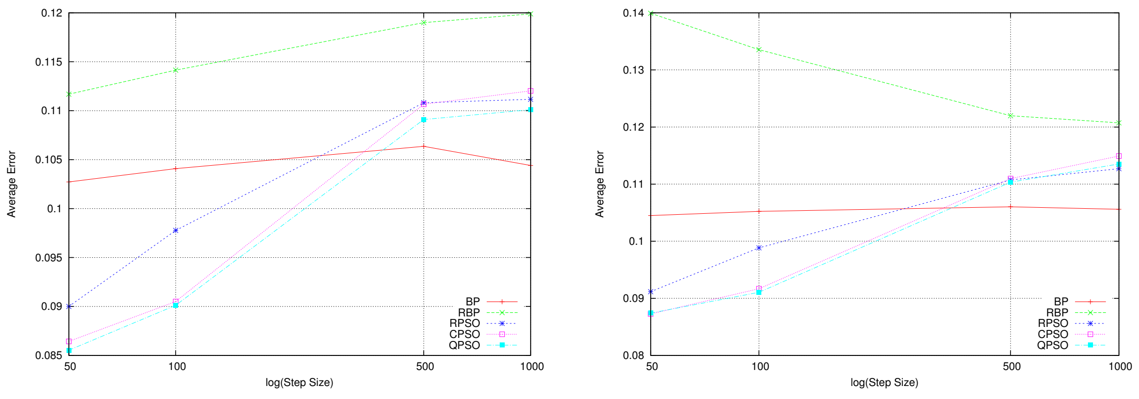
Collective mean E_T and E_G, ρ_F, and swarm diversity values are reported in Table 6.59. The ρ_F values in Table 6.59 indicate that all algorithms exhibited minor overfitting

Table 6.59: Electricity Pricing Results for Scenarios B1 to B4

Algorithm \ **Scenario**	**B1** *(frequency: 50, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.102724 ± 0.00105	0.104494 ± 0.002749	1.01968 ± 0.035915	n/a
Reinit. Back Propagation	0.111686 ± 0.000244	0.13992 ± 0.001823	1.27998 ± 0.031875	n/a
Reinitialising PSO	0.090001 ± 0.001818	0.091167 ± 0.00302	1.01782 ± 0.037745	29.7219 ± 7.42257
Charged PSO	0.086432 ± 0.001352	0.087266 ± 0.003113	1.02825 ± 0.042534	8.83245 ± 1.11529
Quantum PSO	0.085533 ± 0.001075	0.08741 ± 0.002577	1.04178 ± 0.036461	11.7599 ± 1.82975
	B2 *(frequency: 50, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.104079 ± 0.001168	0.105228 ± 0.002769	1.01372 ± 0.035939	n/a
Reinit. Back Propagation	0.114145 ± 0.000307	0.133539 ± 0.002286	1.20129 ± 0.032211	n/a
Reinitialising PSO	0.097767 ± 0.003733	0.098844 ± 0.003975	1.02117 ± 0.032905	14.8794 ± 6.11539
Charged PSO	0.090506 ± 0.001881	0.091683 ± 0.003021	1.02043 ± 0.036611	9.19276 ± 0.934826
Quantum PSO	0.090119 ± 0.001454	0.09106 ± 0.001837	1.02248 ± 0.032973	13.4702 ± 1.876
	B3 *(frequency: 50, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.106364 ± 0.001252	0.10602 ± 0.001793	0.998114 ± 0.03768	n/a
Reinit. Back Propagation	0.118998 ± 0.000255	0.121973 ± 0.002099	1.03134 ± 0.028644	n/a
Reinitialising PSO	0.110811 ± 0.001183	0.110738 ± 0.002312	1.00479 ± 0.02451	6.51035 ± 0.437222
Charged PSO	0.110661 ± 0.002022	0.110967 ± 0.002817	1.00327 ± 0.030723	13.8621 ± 1.40065
Quantum PSO	0.1091 ± 0.002018	0.110334 ± 0.003032	1.00959 ± 0.025336	17.8924 ± 1.93196
	B4 *(frequency: 50, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.104404 ± 0.001763	0.105596 ± 0.003542	1.00955 ± 0.035832	n/a
Reinit. Back Propagation	0.119889 ± 0.000466	0.120727 ± 0.002529	1.00662 ± 0.033486	n/a
Reinitialising PSO	0.111163 ± 0.001441	0.112698 ± 0.002586	1.01501 ± 0.027651	7.90617 ± 0.495408
Charged PSO	0.112023 ± 0.001367	0.114921 ± 0.002237	1.02394 ± 0.027742	16.3018 ± 1.80682
Quantum PSO	0.1101 ± 0.000844	0.113525 ± 0.002294	1.02403 ± 0.030674	21.5517 ± 1.80445

Table 6.60: Electricity Pricing Algorithm Ranking for Scenarios B1 to B4

Algorithm	B1		B2		B3		B4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	4	4	4	1	1	1	1	2.5	2.5
RBP	5	5	5	5	5	5	5	5	5	5
RPSO	3	3	3	3	3.5	3	3	2.5	3.125	2.875
CPSO	2	1.5	1.5	1.5	3.5	3	4	4	2.75	2.5
QPSO	1	1.5	1.5	1.5	2	3	2	2.5	1.625	2.125



(a) Average Training Error Results (b) Average Generalisation Error Results

Figure 6.83: Average Error results for Electricity Pricing, scenarios B1 to B4

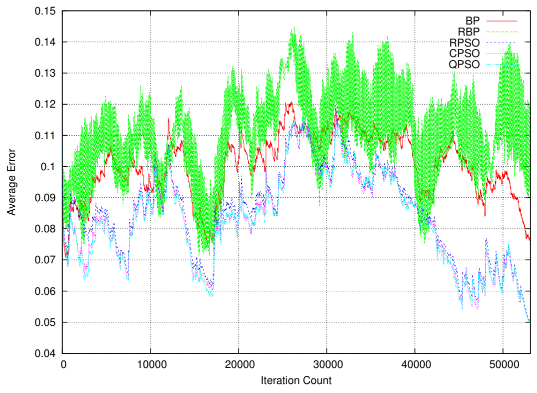
under scenarios B1 to B4, and RBP was again most susceptible to overfitting under gradual scenarios B1 and B2. Average diversity values in Table 6.59 confirm that the RPSO's average diversity under B1 and B2 increased compared to A1 and A2. RPSO's average diversity under B3 and B4, on the other hand, decreased compared to A3 and A4. Thus, the RPSO exhibited divergent behaviour under gradual scenarios, and convergent behaviour under abrupt scenarios: due to its weak dependence on memory, the RPSO failed under scenarios where memory preservance was more important, and succeeeded under scenarios where prompt "unlearning" was crucial. The CPSO's and the QPSO's average diversity under B1 to B4 decreased compared to A1 to A4. Thus, the CPSO and the QPSO exhibited convergent behaviour under all scenarios considered.

The algorithms were ranked based on their collective mean E_T and E_G values, taking the p-values in Tables 6.56 and 6.57 into account. The resulting ranks, as well as the average ranks, are listed in Table 6.60. Collective mean E_T and E_G values are also illustrated in Figure 6.83. Figure 6.83 illustrates that RBP was significantly outperformed by all other algorithms under scenarios B1 to B4. BP was significantly outperformed by the dynamic PSOs under gradual scenarios B1 and B2, and BP significantly outperformed the dynamic PSOs under abrupt scenarios B3 and B4. In fact, concept drift exhibited by real life problems is more often scarce and gradual than frequent and abrupt [114], thus the efficient performance of the dynamic PSOs on gradually changing environments can indeed be useful in real life applications.

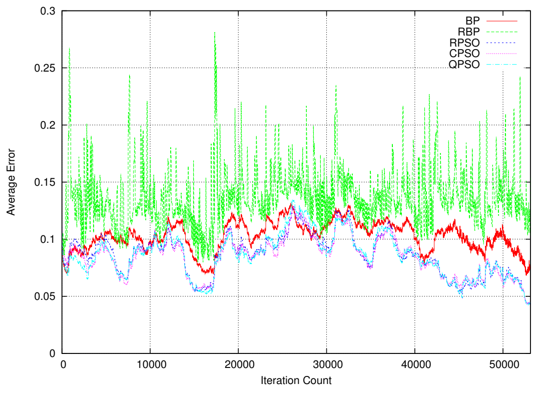
Average ranks in Table 6.60 show that out of the three dynamic PSOs, the QPSO obtained the highest average rank, the CPSO came second, and the RPSO came last. The QPSO reinitialised the quantum cloud at every algorithm iteration, i.e., the QPSO refreshed the swarm memory faster than the CPSO, and also explored wider. The diversity graphs illustrated in Figure 6.82 confirm that the QPSO converged to a larger area than the CPSO. Thus, the QPSO explored better, which proved beneficial on the electricity pricing problem. Table 6.60 shows that the RPSO exhibited the worst performance. The RPSO did not preserve previously learned information as well as the other algorithms did. Also, the RPSO did not implement any diversity-preserving techniques between environment changes, which allowed the RPSO to converge too much between the environment changes.

The average ranks in Table 6.60 indicate that RBP showed the worst performance under scenarios B1 to B4, and the QPSO performed best.

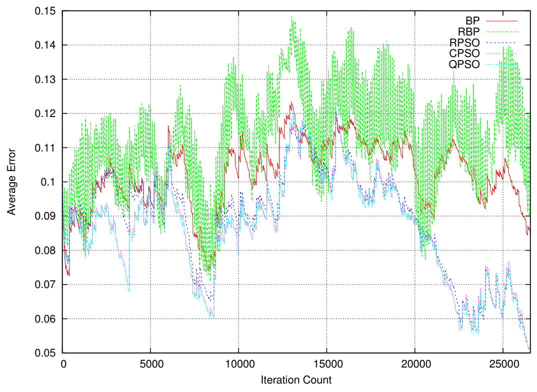
Scenarios C1 to C4: The progression of E_T and E_G over time as obtained by the training algorithms under scenarios C1 to C4 is illustrated in Figures 6.84 and 6.85. Figure 6.84 illustrates that the three dynamic PSOs have again performed better than



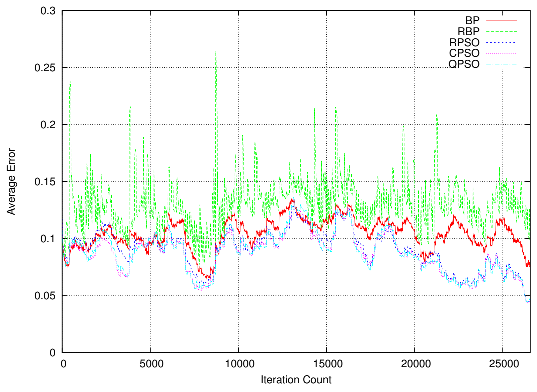
(a) E_T for C1



(b) E_G for C1



(c) E_T for C2

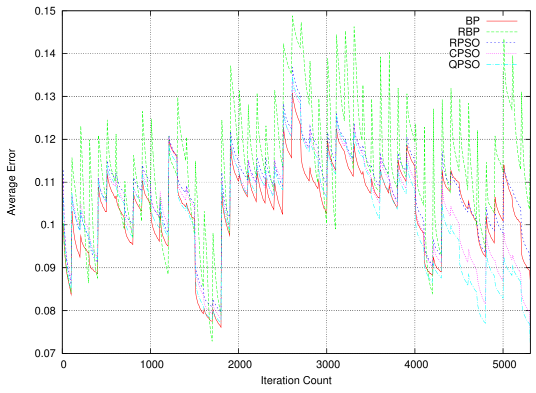


(d) E_G for C2

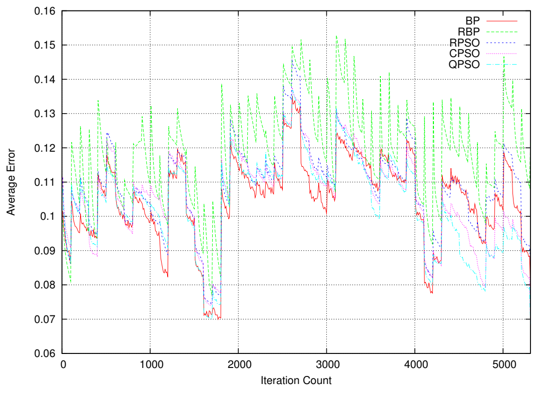
Figure 6.84: Training and generalisation error results for Electricity Pricing, scenarios C1 to C2

both RBP and BP under gradual scenarios by reaching lower minimum error values throughout the algorithm run. RBP once again struggled to compete with the other algorithms considered: the electricity pricing problem used a real-life data set with decision boundaries of more complex and irregular shapes than the ones generated for the moving hyperplane problem or the sliding thresholds problem. Completely reinitialising the NN weights and re-discovering the decision boundaries after every environment change proved impractical in a real-life context.

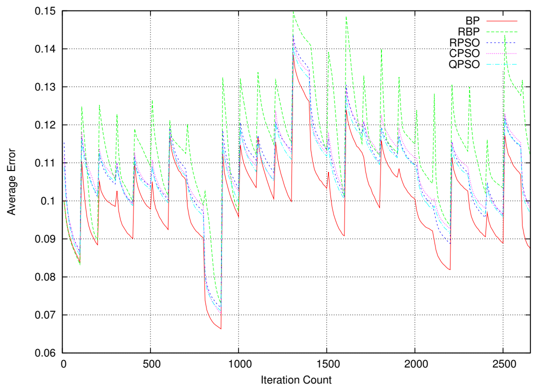
Figure 6.85 illustrates that the dynamic PSOs have improved in performance, given



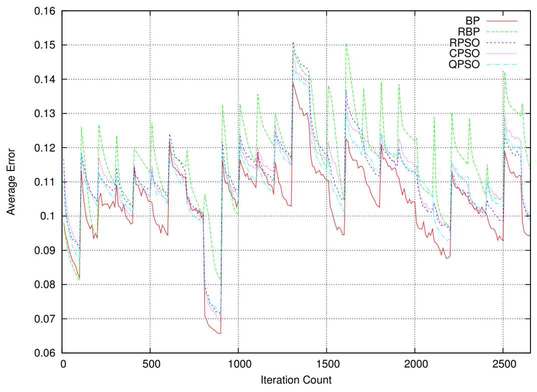
(a) E_T for C3



(b) E_G for C3



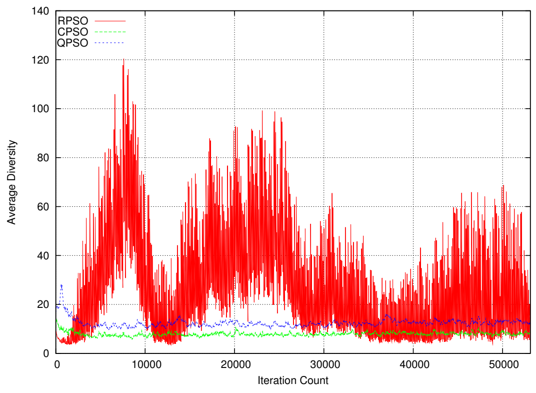
(c) E_T for C4



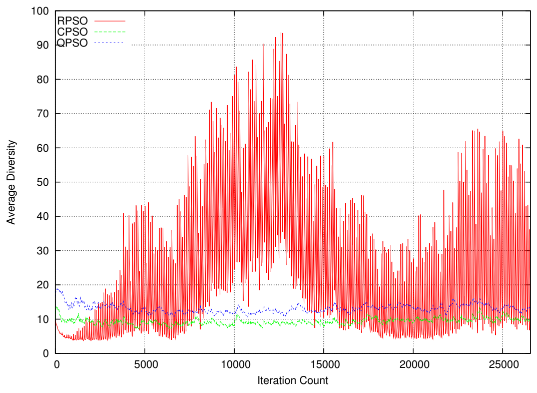
(d) E_G for C4

Figure 6.85: Training and generalisation error results for Electricity Pricing, scenarios C3 to C4

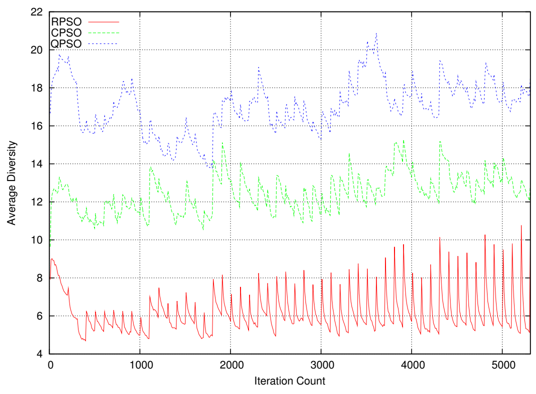
more time to converge between environment changes (change frequency equal to 100 iterations). Figures 6.85(a) and 6.85(b) illustrate that the dynamic PSOs performed worse than BP at the beginning of the algorithm run, caught up with BP around iteration 3500, and then outperformed BP at the end of the algorithm run under the abrupt scenario C3. Figure 6.86 illustrates the progression of swarm diversity over time as obtained for scenarios C1 to C4. Figure 6.86(c) illustrates that none of the dynamic PSOs considered managed to reach a stable diversity level under scenario C3. Thus, the inferior performance of the dynamic PSOs compared to BP can be attributed to slow



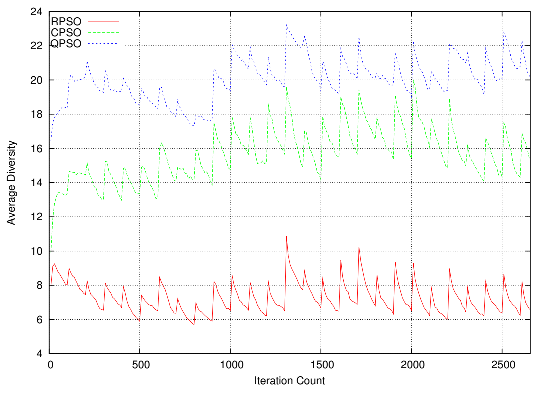
(a) Average Diversity Results for Scenario C1



(b) Average Diversity Results for Scenario C2



(c) Average Diversity Results for Scenario C3

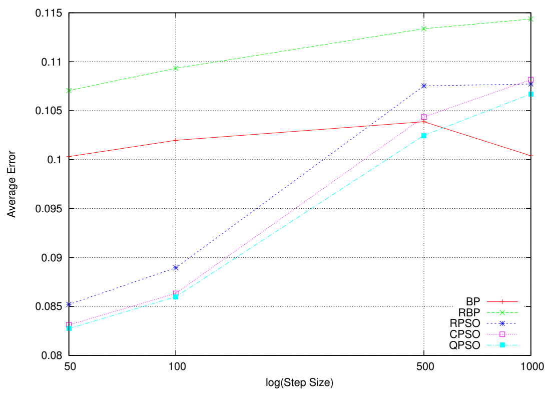


(d) Average Diversity Results for Scenario C4

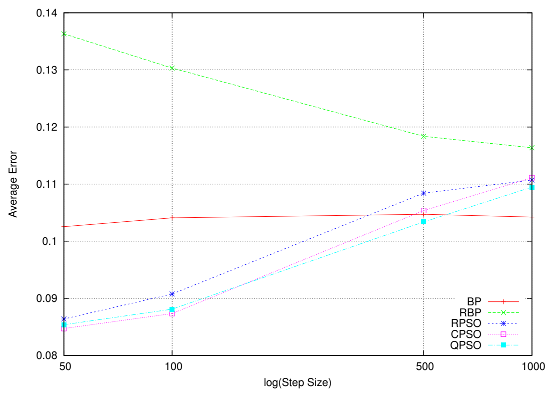
Figure 6.86: Average Diversity results for Electricity Pricing, scenarios C1 to C4

convergence speed of the dynamic PSOs. Real-life problems, however, often require the algorithms to deal with continuous data sets, thus the success of a dynamic algorithm in the long run is more important than some overhead at the beginning of the algorithm run.

Figure 6.86 illustrates that the CPSO and the QPSO reached a stable diversity level under gradual scenarios C1 and C2, but not under the abrupt scenarios C3 and C4. Thus, 100 iterations between environment changes was not enough for the CPSO and the QPSO to exploit the good regions of the search space when the changes were abrupt. The RPSO again showed signs of divergent behaviour by reaching even a higher maximum diversity



(a) Average Training Error Results



(b) Average Generalisation Error Results

Figure 6.87: Average Error results for Electricity Pricing, scenarios C1 to C4

level under scenarios C1 to C4 than under B1 to B4.

Table 6.61 lists the collective mean E_T, E_G, ρ_F, and swarm diversity values obtained by the five algorithms under scenarios C1 to C4. The ρ_F values in Table 6.61 indicate that all algorithms exhibited some overfitting under scenarios C1 to C4. RBP was most prone to overfitting under gradual scenarios, due to slow replacement of stale data inside the sliding window under these scenarios. BP was more prone to overfitting than the dynamic PSOs under abrupt scenarios, due to the faster convergence speed of BP.

The algorithms were ranked based on their collective mean E_T and E_G values, taking the p-values in Tables 6.56 and 6.57 into account. The obtained ranks, along with the average ranks, are reported in Table 6.62. Collective mean E_T and E_G values obtained for C1 to C4 are illustrated in Figure 6.87. Figure 6.87 illustrates that RBP once again came last in terms of both E_T and E_G under all scenarios considered. BP was significantly outperformed by the dynamic PSOs under gradual scenarios C1 and C2, but performed significantly better than the dynamic PSOs under the most abrupt scenario C4, as Figure 6.87 illustrates. Table 6.62 shows that no statistically significant difference was observed between BP, the CPSO, and the QPSO under abrupt scenario C3. While BP was little affected by the extent of temporal severity, the dynamic PSOs showed

Table 6.61: Electricity Pricing Results for Scenarios C1 to C4

Algorithm \ Scenario	**C1** *(frequency: 100, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.100309 ± 0.002084	0.102539 ± 0.00303	1.02636 ± 0.040185	n/a
Reinit. Back Propagation	0.107059 ± 0.00025	0.136312 ± 0.002022	1.29926 ± 0.026974	n/a
Reinitialising PSO	0.085203 ± 0.001031	0.086372 ± 0.002087	1.03924 ± 0.043862	27.0377 ± 5.5646
Charged PSO	0.083106 ± 0.001323	0.084711 ± 0.002284	1.04486 ± 0.036309	7.92476 ± 1.35528
Quantum PSO	0.082736 ± 0.001698	0.085369 ± 0.002352	1.04755 ± 0.041873	12.4046 ± 2.04719
	C2 *(frequency: 100, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.101966 ± 0.001622	0.104083 ± 0.003152	1.02786 ± 0.038175	n/a
Reinit. Back Propagation	0.109329 ± 0.00024	0.13032 ± 0.002253	1.21879 ± 0.028532	n/a
Reinitialising PSO	0.088946 ± 0.001818	0.090749 ± 0.002938	1.02782 ± 0.038223	23.5604 ± 4.87848
Charged PSO	0.086329 ± 0.001865	0.087324 ± 0.002207	1.01874 ± 0.031964	9.36062 ± 1.26703
Quantum PSO	0.08598 ± 0.001284	0.0881 ± 0.002133	1.03668 ± 0.032632	13.1008 ± 1.46531
	C3 *(frequency: 100, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.103866 ± 0.001315	0.104704 ± 0.002472	1.01035 ± 0.030989	n/a
Reinit. Back Propagation	0.113369 ± 0.000367	0.118371 ± 0.002558	1.04224 ± 0.03249	n/a
Reinitialising PSO	0.107534 ± 0.001977	0.108407 ± 0.003185	1.00866 ± 0.036487	6.14009 ± 0.558376
Charged PSO	0.10435 ± 0.004578	0.105365 ± 0.005218	1.00946 ± 0.030144	12.475 ± 1.13447
Quantum PSO	0.102471 ± 0.004847	0.103366 ± 0.005688	1.00685 ± 0.027117	17.0299 ± 1.52774
	C4 *(frequency: 100, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.100393 ± 0.001628	0.104217 ± 0.002812	1.03391 ± 0.038747	n/a
Reinit. Back Propagation	0.114362 ± 0.000416	0.116367 ± 0.002244	1.01491 ± 0.030828	n/a
Reinitialising PSO	0.107722 ± 0.001031	0.110712 ± 0.002297	1.02851 ± 0.031989	7.26793 ± 0.529495
Charged PSO	0.108165 ± 0.001332	0.111114 ± 0.002527	1.02435 ± 0.027656	15.6542 ± 1.39795
Quantum PSO	0.1067 ± 0.001677	0.109474 ± 0.002684	1.01869 ± 0.026249	20.0419 ± 1.69383

Table 6.62: Electricity Pricing Algorithm Ranking for Scenarios C1 to C4

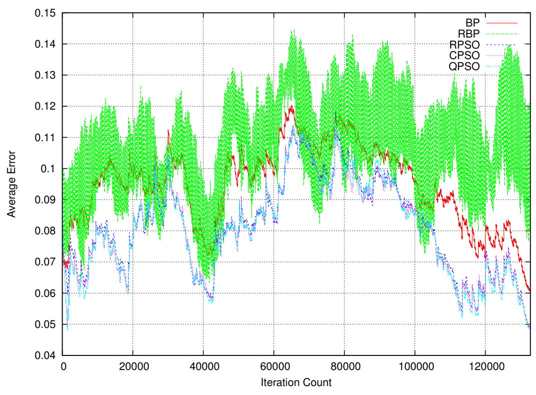
Algorithm	C1		C2		C3		C4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	4	4	4	2	2	1	1	2.75	2.75
RBP	5	5	5	5	5	5	5	5	5	5
RPSO	3	2	3	3	4	4	3.5	3	3.375	3
CPSO	1.5	2	1.5	1.5	2	2	3.5	3	2.125	2.125
QPSO	1.5	2	1.5	1.5	2	2	2	3	1.75	2.125

stable improvement in performance as the change frequency decreased from scenarios A1 to A4, to C1 to C4.

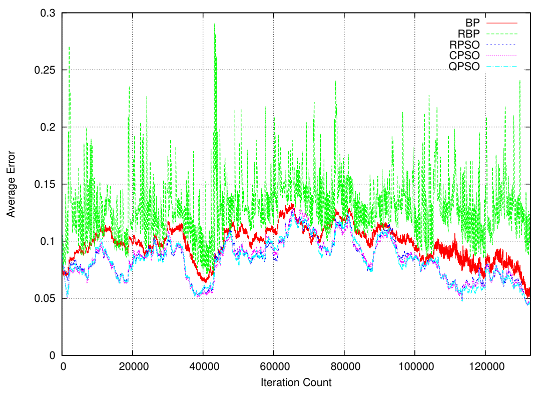
Average rank values in Table 6.62 show that the CPSO and the QPSO obtained the same rank in terms of E_G, and both these algorithms obtained a higher rank than BP. The CPSO and the QPSO were the best performers under scenarios C1 to C4, and RBP showed the worst performance.

Scenarios D1 to D4: Figures 6.88 and 6.89 illustrate the E_T and E_G profiles over time obtained by the algorithms for scenarios D1 to D4. Figure 6.88 illustrates similar trends as were observed under scenarios C1 and C2: the dynamic PSOs performed visibly better than both BP and RBP. It becomes visible in Figures 6.88(a) and 6.88(c) that RBP often reached a lower minimum E_T than BP, and sometimes the RBP's E_T was even lower than the minimum E_T of the dynamic PSOs. The RBP reinitialised all NN weights after every change, thus RBP had a chance of escaping the unfruitful search space areas where the other algorithms resided due to their memory of previous solutions. However, RBP failed to support good E_T values by similarly good E_G values, as Figures 6.88(b) and 6.88(d) illustrate: RBP, having no memory of previous solutions, failed to discern between the stale and up-to-date data in the sliding window, which resulted in the observed overfitting behaviour.

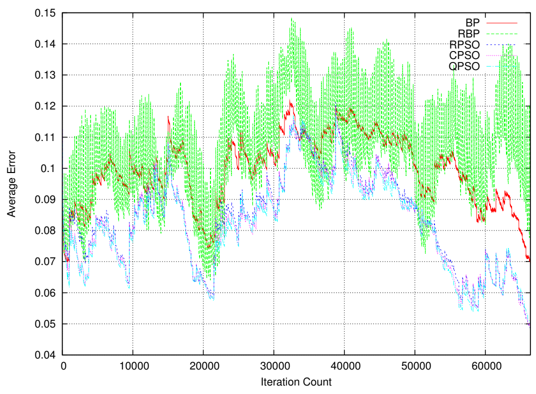
Figure 6.89 illustrates that the dynamic PSOs, given enough time to converge, man-



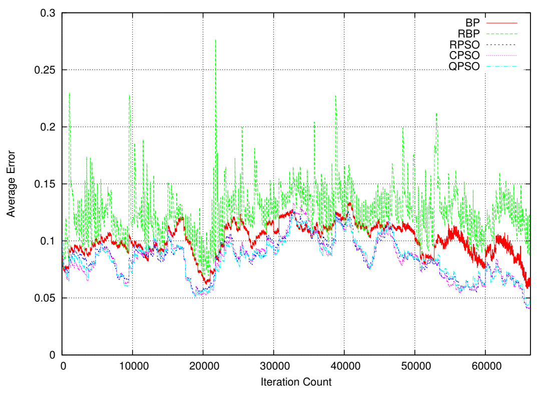
(a) E_T for D1



(b) E_G for D1



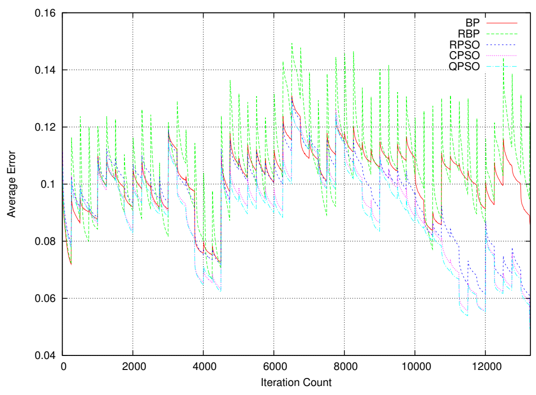
(c) E_T for D2



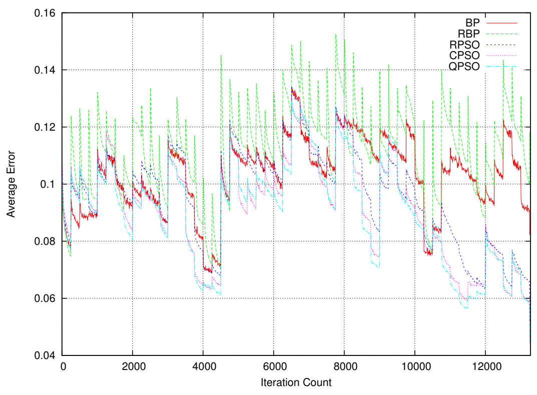
(d) E_G for D2

Figure 6.88: Training and generalisation error results for Electricity Pricing, scenarios D1 to D2

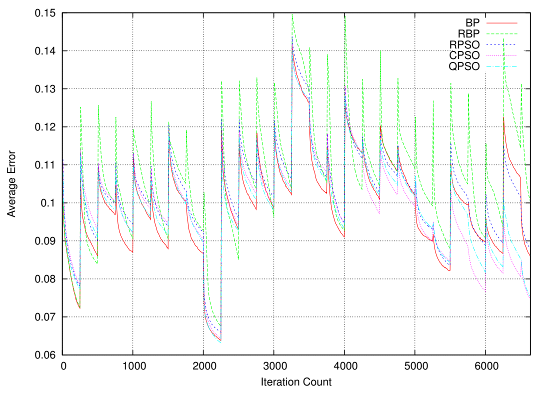
aged to perform as well as BP, and even to outperform BP under abrupt scenarios. The superiority of the dynamic PSOs compared to BP is quite apparent in Figures 6.89(a) and 6.89(b). Under the most abrupt scenario D4, as Figures 6.89(c) and 6.89(d) illustrate, the dynamic PSOs were inferior to BP in the beginning of the algorithm run, and then outperformed BP in the second half of the algorithm run. Both E_T and E_G were stably decreasing between environment changes simulated by scenario D4. Thus, the dynamic PSOs exhibited no signs of stagnation. Hence, the dynamic PSOs are likely to outperform BP in the long run on a problem similar to electricity pricing, no matter



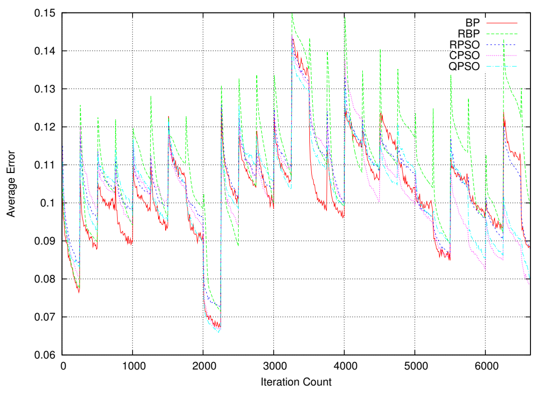
(a) E_T for D3



(b) E_G for D3



(c) E_T for D4



(d) E_G for D4

Figure 6.89: Training and generalisation error results for Electricity Pricing, scenarios D3 to D4

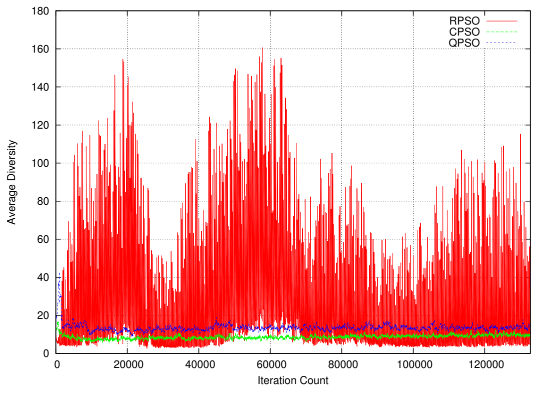
whether changes are abrupt or gradual. The dynamic PSOs indeed converge slower than BP, but exploit and track the optima better when given enough iterations to do so.

Scenarios D1 to D4 allowed 250 algorithm iterations between environment changes, which proved enough for the dynamic PSOs: Figure 6.90 illustrates the swarm diversity progression over time for scenarios D1 to D4, and shows that the CPSO and the QPSO reached a stable diversity level under scenarios D1 to D4. The RPSO again exhibited signs of divergence under scenarios D1 to D3.

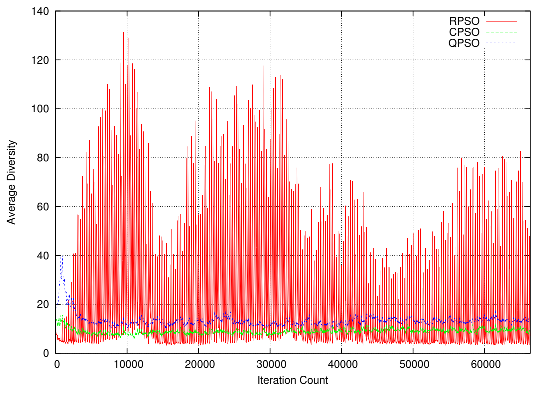
Table 6.63 summarises the collective mean E_T, E_G, ρ_F, and swarm diversity values

Table 6.63: Electricity Pricing Results for Scenarios D1 to D4

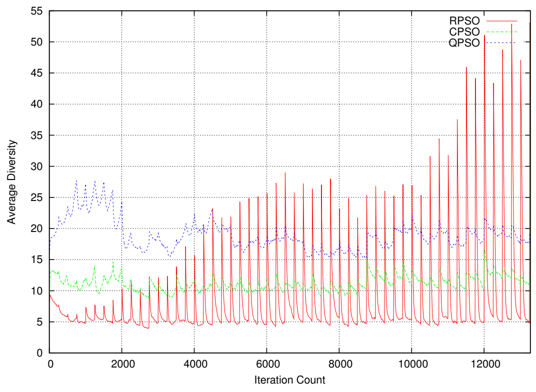
Scenario / **Algorithm**	**D1** *(frequency: 250, step size: 50)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.094779 ± 0.003527	0.097618 ± 0.00474	1.03968 ± 0.041908	n/a
Reinit. Back Propagation	0.100251 ± 0.000249	0.127633 ± 0.001818	1.29801 ± 0.035975	n/a
Reinitialising PSO	0.081339 ± 0.001314	0.084291 ± 0.002058	1.05428 ± 0.035383	26.7539 ± 9.63906
Charged PSO	0.080573 ± 0.002001	0.083854 ± 0.002978	1.05752 ± 0.051231	8.61006 ± 1.08301
Quantum PSO	0.08026 ± 0.001216	0.083442 ± 0.002134	1.06523 ± 0.032704	13.4439 ± 1.72592
	D2 *(frequency: 250, step size: 100)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.099172 ± 0.002479	0.100719 ± 0.002892	1.02128 ± 0.033658	n/a
Reinit. Back Propagation	0.102321 ± 0.000283	0.122955 ± 0.002114	1.23313 ± 0.04248	n/a
Reinitialising PSO	0.083684 ± 0.001233	0.085278 ± 0.00214	1.03236 ± 0.044444	20.7362 ± 8.35509
Charged PSO	0.082523 ± 0.001528	0.084428 ± 0.002947	1.03085 ± 0.042739	9.00562 ± 1.57632
Quantum PSO	0.081744 ± 0.001526	0.084446 ± 0.002468	1.04553 ± 0.048728	13.4092 ± 1.50571
	D3 *(frequency: 250, step size: 500)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.10169 ± 0.001228	0.102847 ± 0.002181	1.00467 ± 0.029516	n/a
Reinit. Back Propagation	0.105676 ± 0.000334	0.111569 ± 0.002374	1.06206 ± 0.034034	n/a
Reinitialising PSO	0.095168 ± 0.004108	0.095952 ± 0.005246	1.00637 ± 0.029907	8.17592 ± 2.19789
Charged PSO	0.090168 ± 0.003092	0.09121 ± 0.003668	1.01716 ± 0.02538	11.2166 ± 1.4259
Quantum PSO	0.088658 ± 0.001317	0.089816 ± 0.003046	1.01049 ± 0.039784	18.9432 ± 2.28585
	D4 *(frequency: 250, step size: 1000)*			
	Training Error	Generalisation Error	ρ_F	Swarm Diversity
Back Propagation	0.099187 ± 0.001916	0.102562 ± 0.003234	1.02886 ± 0.045493	n/a
Reinit. Back Propagation	0.106452 ± 0.000427	0.109373 ± 0.002192	1.03166 ± 0.032414	n/a
Reinitialising PSO	0.102084 ± 0.001793	0.105275 ± 0.003199	1.0353 ± 0.034309	6.60816 ± 0.527193
Charged PSO	0.098671 ± 0.004797	0.101999 ± 0.006492	1.02987 ± 0.037575	14.7018 ± 1.07111
Quantum PSO	0.098989 ± 0.003366	0.102871 ± 0.004776	1.04205 ± 0.038577	21.2345 ± 1.76094



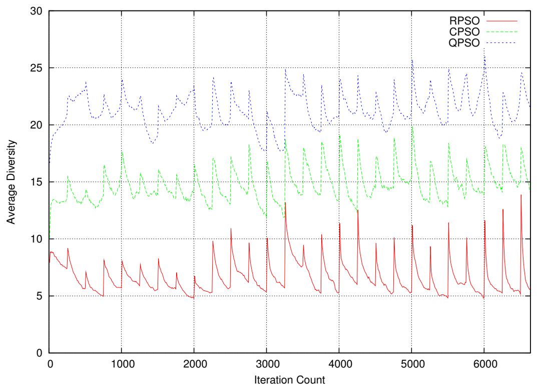
(a) Average Diversity Results for Scenario D1



(b) Average Diversity Results for Scenario D2



(c) Average Diversity Results for Scenario D3



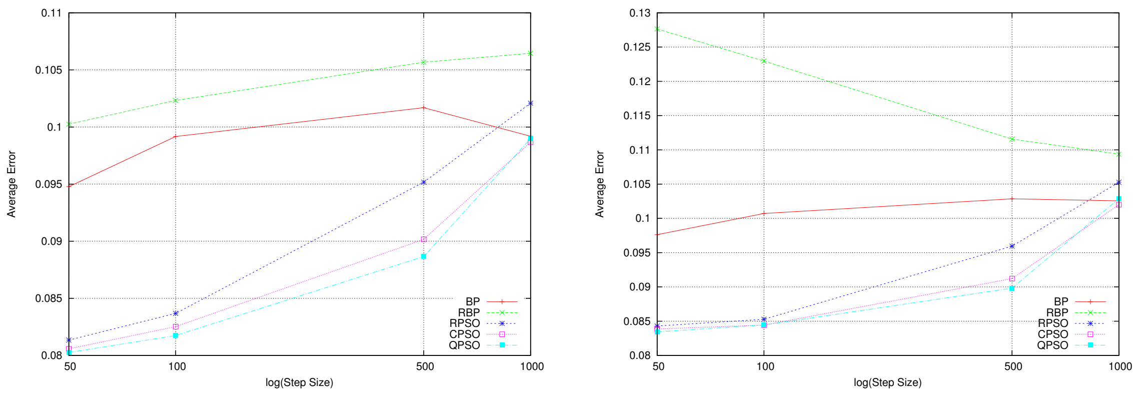
(d) Average Diversity Results for Scenario D4

Figure 6.90: Average Diversity results for Electricity Pricing, scenarios D1 to D4

obtained by the algorithms under scenarios D1 to D4. The ρ_F values in Table 6.63 indicate that all algorithms were again subject to overfitting under scenarios D1 to D4. Collective mean E_T and E_G values are illustrated in Figure 6.91, and algorithm ranks, along with average ranks, are listed in Table 6.64. Figure 6.91 illustrates that all algorithms deteriorated in performance as the spatial severity of changes increased. Changes of high spatial severity replaced more patterns inside the sliding window, adding more new information, thus bigger changes had to be made to the learned model. It becomes evident from Figure 6.91 that BP was less sensitive to the level of spatial severity than the dynamic PSOs, since the collective mean E_T and E_G values obtained

Table 6.64: Electricity Pricing Algorithm Ranking for Scenarios D1 to D4

Algorithm	D1		D2		D3		D4		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	4	4	4	4	4	4	2	2	3.5	3.5
RBP	5	5	5	5	5	5	5	5	5	5
RPSO	2	2	3	2	3	3	4	4	3	2.75
CPSO	2	2	1.5	2	1.5	1.5	2	2	1.75	1.875
QPSO	2	2	1.5	2	1.5	1.5	2	2	1.75	1.875



(a) Average Training Error Results (b) Average Generalisation Error Results

Figure 6.91: Average Error results for Electricity Pricing, scenarios D1 to D4

by BP under D1 to D4 covered a smaller numerical interval than the corresponding errors produced by the dynamic PSOs. BP is a hill climber, thus the degree of BP's success depended entirely on the landscape of the error function and the position of the weight vector on that landscape after an environment change. The dynamic PSOs with their population-based approach to NN training also depended on the specific characteristics of the swarm such as the current swarm diversity and current particle velocities, in addition to the positions of the particles in the weight space. The population-based approach of the dynamic PSOs was often superior to BP. However, the dynamic PSOs proved more sensitive to specific characteristics of dynamic environments, such as spatial and temporal severity.

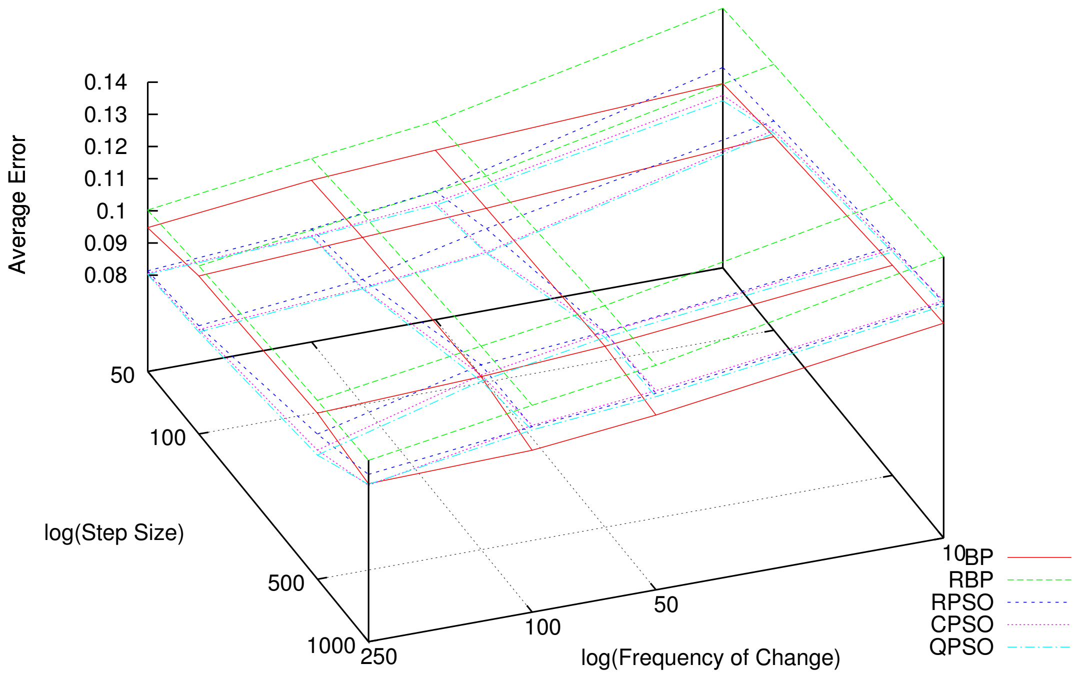


Figure 6.92: Average Training Error Results for Electricity Pricing

Table 6.64 shows that all three dynamic PSOs significantly outperformed BP under scenarios D1 to D3, and no statistically significant difference was observed between the CPSO, the QPSO, and BP under the most abrupt scenario D4. The CPSO and the QPSO obtained the highest average E_T and E_G ranks, and RBP once again performed worst.

All scenarios: The average training errors produced by the five algorithms under the 16 different dynamic scenarios are illustrated in Figure 6.92, and the average generalisation errors are illustrated in Figure 6.93.

Figures 6.92 and 6.93 illustrate that BP was less sensitive to dynamic scenario characteristics than the other algorithms considered: training and generalisation errors pro-

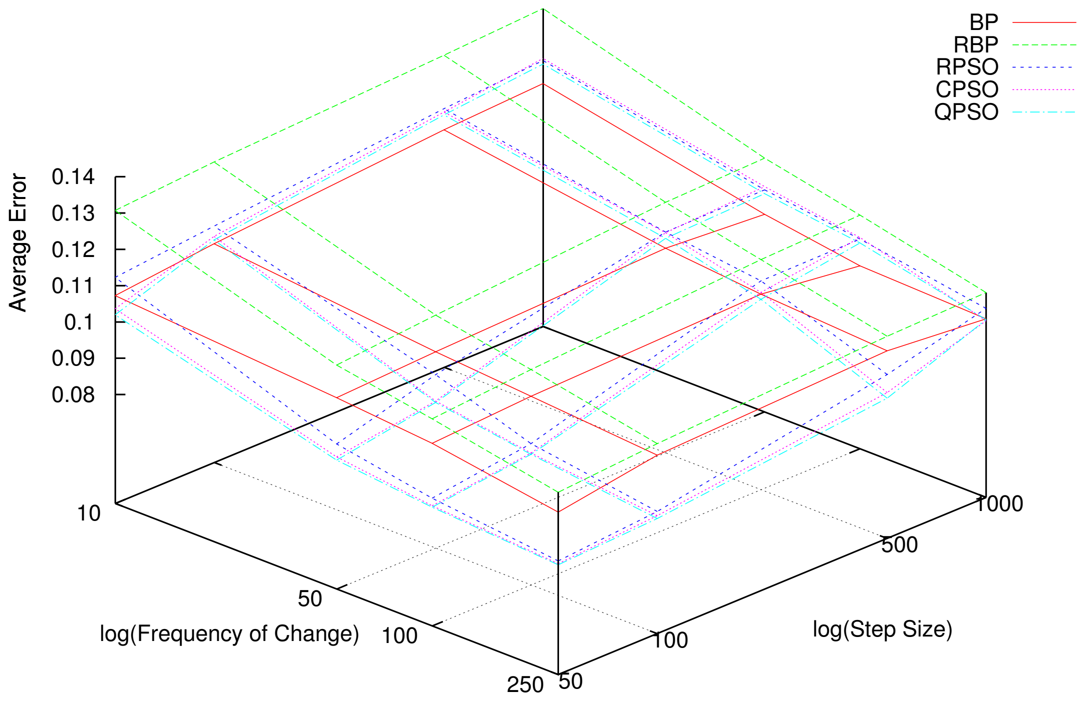


Figure 6.93: Average Generalisation Error Results for Electricity Pricing

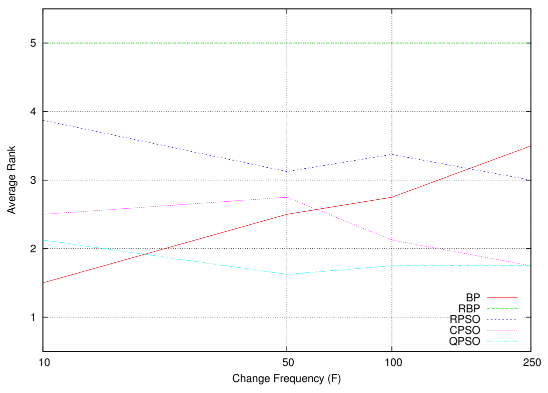
duced by BP were little affected by the changes in spatial and temporal severity.

Figures 6.92 and 6.93 illustrate that the performance of RBP improved as the temporal severity decreased: RBP was restarted after every environment change, thus RBP required more time to converge than all other algorithms considered. Even though RBP improved under low temporal severity, this algorithm still showed the worst performance under all scenarios considered: completely discarding previously learned information was not an efficient approach to the electricity pricing problem, since new electricity prices were always derived from old electricity prices.

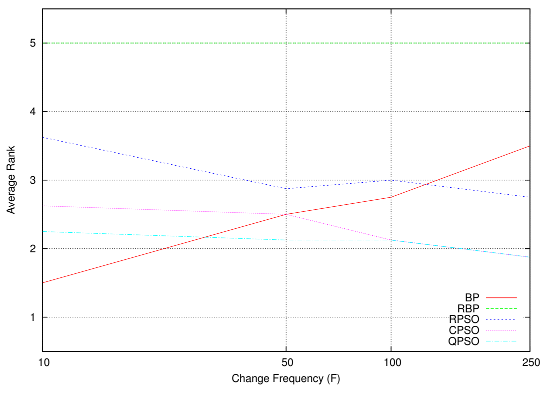
The dynamic PSOs outperformed both BP and RBP under gradual changes, as well as under changes of low temporal severity. Figures 6.92 and 6.93 illustrate that the performance of all dynamic PSOs deteriorated as either spatial severity increased or

Table 6.65: Electricity Pricing Average Algorithm Ranking for Scenarios A to D

Algorithm	Average R(A)		Average R(B)		Average R(C)		Average R(D)		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	1.5	1.5	2.5	2.5	2.75	2.75	3.5	3.5	2.5625	2.5625
RBP	5	5	5	5	5	5	5	5	5	5
RPSO	3.875	3.625	3.125	2.875	3.375	3	3	2.75	3.34375	3.0625
CPSO	2.5	2.625	2.75	2.5	2.125	2.125	1.75	1.875	2.28125	2.28125
QPSO	2.125	2.25	1.625	2.125	1.75	2.125	1.75	1.875	1.8125	2.09375



(a) Average Rank in terms of E_T



(b) Average Rank in terms of E_G

Figure 6.94: Average Rank results for Electricity Pricing

temporal severity increased. The electricity pricing problem proved to be nontrivial: the dynamic PSOs required more time to converge on a good solution between changes, and struggled when the frequency of changes was not proportional to the spatial severity of changes. The dynamic PSOs consistently outperformed the hill-climbing approaches when given enough time to converge between changes.

Average ranks obtained by the algorithms under scenarios A1 to A4, B1 to B4, C1 to C4, and D1 to D4 are reported in Table 6.65 along with the overall average ranks. The average ranks in terms of E_T and E_G are also visualised in Figure 6.94. Figure 6.94 illustrates that, out of the three dynamic PSOs, the QPSO obtained the highest rank under most frequencies considered, the CPSO came second, and the RPSO showed the worst performance. The RPSO was least dependent on the memory of previous solutions, which, as already explained, was not a good approach to the electricity pricing problem, because new prices always depended on the old prices. Figure 6.94 illustrates that the QPSO was superior to the CPSO under changes of high temporal severity, and the two algorithms were ranked as equals under the least temporally severe changes. Hence, the QPSO performed with a higher degree of success when prompt exploitation was necessary. The QPSO's quantum cloud uniformly covers the area in the immediate proximity of the global best particle, which helps the swarm to exploit the area around the global best. The CPSO, on the other hand, preserves swarm diversity with its electrostatic principles, but does not specifically promote exploitation of fruitful search space regions.

The overall average ranks in Table 6.65 indicate that the QPSO obtained the highest average rank, the CPSO came second, BP came third, followed by the RPSO, and RBP obtained the lowest rank. Thus, the least memory-preserving algorithms came last, and BP was outperformed by two dynamic PSOs on the electricity pricing problem.

6.3 Summary

The empirical study conducted for this dissertation was presented in this chapter. After discussing the performance measurements used in this study, a discussion on simulating dynamic classification problems was provided, and the parameter optimisation process was described in detail. The rest of the chapter was dedicated to the dynamic classification problems considered in this study, and the experimental results obtained.

Table 6.66 summarises the overall average ranks obtained by the algorithms for the five different dynamic classification problems considered. The highest rank for each problem is given in bold. Overall average algorithm ranks for the five problems are also listed in Table 6.66. Table 6.66 shows that the dynamic PSOs proved to be a viable

Table 6.66: Average Algorithm Ranking

Algorithm	SEA Concepts		Moving Hyperplane		Dynamic Sphere		Sliding Thresholds		Electricity Pricing		Average Rank	
	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$	$R(E_T)$	$R(E_G)$
BP	2.925	2.6	4	2.90625	3.6875	**2.4375**	4.75	4.4375	2.5625	2.5625	3.585	2.98875
RBP	4.775	4.625	**2.09375**	4.71875	**1.34375**	3.25	4.03125	4.3125	5	5	3.44875	4.38125
RPSO	**2.2**	2.675	3.1875	2.75	2.0625	2.71875	**1.75**	**1.71875**	3.34375	3.0625	2.50875	2.585
CPSO	2.9	**2.55**	2.5	**2.09375**	3.6875	3	2.28125	2.3125	2.28125	2.28125	2.73	2.4475
QPSO	**2.2**	**2.55**	3.21875	2.53125	4.21875	3.59375	2.1875	2.21875	**1.8125**	**2.09375**	2.7275	2.5975

Table 6.67: Best performing algorithms under varying spatial severity for the five considered dynamic classification problems

Step Size	**50**	**100**	**500**	**1000**	**2500**
SEA Concepts	RPSO	RPSO, QPSO	QPSO	BP	BP
Moving Hyperplane	RPSO	RPSO	CPSO	BP	-
Dynamic Sphere	RPSO	RPSO	BP	RBP	-
Sliding Thresholds	RPSO	RPSO	RPSO, CPSO, QPSO	QPSO	-
Electricity Pricing	QPSO	CPSO, QPSO	BP	BP	-

alternative to hill-climbing training algorithms such as BP and RBP: all three dynamic PSOs obtained a higher overall rank than both BP and RBP. Table 6.66 shows that BP managed to outperform the dynamic PSOs only on the dynamic sphere problem, which, as discussed in Section 6.2.3, is a rather trivial problem.

Table 6.66 also indicates that, out of the three dynamic PSOs considered, the RPSO produced the highest rank for problems where prompt "unlearning" of stale data was important (i.e. the dynamic sphere and sliding thresholds problems). The CPSO and the QPSO, on the other hand, performed superior to the RPSO for problems where preservation of memory was more important, i.e. where the new boundaries were derived from the old boundaries (electricity pricing), or where no conflicting boundaries were present (SEA concepts). Hence, the choice of the best dynamic PSO remains problem-specific.

Table 6.68: Best performing algorithms under varying temporal severity for the five considered dynamic classification problems

Change Frequency	**A** *(10)*	**B** *(50)*	**C** *(100)*	**D** *(250)*
SEA Concepts	BP	RPSO, QPSO	RPSO, QPSO	RPSO, QPSO
Moving Hyperplane	CPSO	CPSO	CPSO	RPSO
Dynamic Sphere	RPSO	CPSO	CPSO	BP
Sliding Thresholds	CPSO, QPSO	RPSO	RPSO	RPSO
Electricity Pricing	BP	QPSO	QPSO	CPSO, QPSO

Table 6.67 lists the best performing algorithms under various sliding window step sizes (i.e. spatial severity) considered for the five dynamic classification problems. Table 6.67 shows that the dynamic PSOs were more successful under gradual to semi-gradual scenarios, and BP and RBP were more successful under abrupt scenarios. It was observed throughout the empirical analysis carried out in this chapter that the dynamic PSOs exploited and tracked the decision boundaries better than BP and RBP. Hence, the dynamic PSOs were more successful under scenarios where new information arrived in small portions. Table 6.68 lists the top performing algorithms under various change frequencies (i.e. temporal severity). BP and RBP converged faster than the dynamic PSOs, and were much more sensitive to the presence of stale data. Thus, as Tables 6.67 and 6.68 show, BP and RBP were more successful under frequent and abrupt scenarios, where changes were either drastic, or had to be adjusted to in a short period of time, and where the stale data was promptly discarded.

Table 6.66 shows that RBP obtained the lowest overall E_G rank among the considered algorithms. Indeed, a complete restart of the training algorithm proved to be an inefficient approach to NN training in dynamic environments.

It was also observed for all considered problems that the dynamic PSOs exhibited a stronger sensitivity to such characteristics of dynamic environments as spatial severity and temporal severity. The dynamic PSO algorithms involve more parameter optimisation than the hill-climbing BP and RBP, since, in addition to the standard PSO parameters, parameters specific to each dynamic PSO also require optimisation. Hence, the dynamic PSOs require more fine-tuning than BP and RBP, but have a high potential to outperform the hill-climbers when properly optimised. The dynamic PSOs exhibited higher capacity for learning than BP and RBP even under abrupt changes for the sliding thresholds problem, where multiple decision boundaries were present.

RBP was subject to overfitting under all gradual scenarios where conflicting boundaries were present. BP overfitted whenever the temporal severity allowed BP to train for too long. The dynamic PSOs, however, exhibited minor overfitting under most scenarios considered. The tendency of the dynamic PSOs to overfit remains a topic for future research, since multiple factors may be contributing to such behaviour; for example, the sigmoid activations function used [119], the values of V_{max}, and the diversity-preserving mechanisms which never allow complete convergence, yielding continuous refinement of already good enough solutions.

The next chapter summarises the findings of this study and lists possible topics for future research.

Chapter 7

Conclusions

When a love comes to an end, weaklings cry, efficient ones instantly find another love, and the wise already have one in reserve.

Oscar Wilde

This chapter summarises the work presented in this thesis and all the important findings derived from the empirical study conducted. Section 7.1 presents the conclusions, and Section 7.2 lists possible topics for future research.

7.1 Summary of Conclusions

The main objective of this thesis was to show that dynamic PSO algorithms have potential to be efficiently applied to NN training in the presence of concept drift. Different dynamic PSO algorithms were applied to NN training on a selection of dynamic classification problems under a representative selection of dynamic scenarios in order to examine the applicability of dynamic PSOs as dynamic NN training algorithms. This chapter summarises the major observations and contributions made in the course of this study.

This study involved three CI research fields, namely PSO, NNs, and concept drift. The research started with a discussion of the PSO algorithm, as well as the algorithm parameters that have an influence on the performance of PSO. NNs were discussed next, with a thorough overview of back propagation and the algorithm parameters that have an influence on the performance of back propagation. The applicability of PSO as a NN training algorithm was discussed, and the advantages and disadvantages of PSO compared to back propagation were outlined.

Although PSO was applied to NN training with a high degree of success before, it was usually assumed that the training data was static. The falsity of such assumption was substantiated with some real-life examples of dynamic NN training problems. Various properties of dynamic environments were discussed next, and it was determined that dynamic classification problems, or problems with concept drift, are a subset of the larger set of dynamic optimisation problems.

PSO is capable of adjusting to the changing objective function landscape until the swarm reaches convergence. Loss of swarm diversity, as well as outdated swarm memory, prevent the standard PSO from being efficient in dynamic environments. Some popular dynamic PSO approaches that address these two issues were discussed, namely RPSO, CPSO, and QPSO.

The applicability of BP as a NN training algorithm under concept drift was discussed next. A selection of hypothetical scenarios illustrated that, although BP is indeed implicitly dynamic, BP still has significant limitations. It was suggested that dynamic PSO algorithms be used to train NNs under concept drift as an alternative to BP. Possible advantages of the dynamic PSOs applied to train NNs under concept drift were discussed. The problem of overfitting, architecture selection, and algorithm parameter optimisation in the context of concept drift was also discussed.

An experimental procedure was designed to compare the training algorithms on a representative selection of dynamic classification problems under a representative selection of dynamic environments. A sliding window was used for pattern selection, and dynamic environments of varying spatial and temporal severity were simulated by adjusting the step size of the sliding window and the number of algorithm iterations between the sliding window shifts, respectively.

Four artificially generated dynamic classification problems and one real-life data set with concept drift were used in the experiments. The problems differed in terms of dimensionality, decision boundary shape, total number of decision boundaries, and the probability of encountering conflicting decision boundaries in the sliding window data.

BP was compared to RBP, as well as to the three dynamic PSO algorithms previously mentioned: the RPSO, the CPSO, and the QPSO.

The major findings of the empirical study conducted can be summarised as follows:

- The dynamic PSOs are viable NN training algorithms in the context of dynamic classification problems.

- The dynamic PSOs exploited and tracked decision boundaries better than BP and RBP. Hence, the dynamic PSOs outperformed BP and RBP under scenarios exhibiting infrequent to moderately infrequent gradual changes.

- BP and RBP converged faster than the dynamic PSOs. Hence, BP outperformed the dynamic PSOs under frequent abrupt changes.

- RBP exhibited the worst performance under most scenarios for most problems. Hence, a complete reinitialisation of NN weights is an inefficient approach to handling concept drift.

- The dynamic PSOs outperformed BP and RBP on a real-life data set with concept drift.

- The dynamic PSOs exhibited stronger sensitivity to the extent of temporal and spatial severity than BP and RBP. Hence, the dynamic PSOs require scenario-specific optimisation.
- Dynamic PSOs were less sensitive to the total number of decision boundaries or the presence of stale data in the sliding window than BP and RBP. Hence, the dynamic PSOs are expected to perform better than hill-climbers on rugged NN error surfaces.
- The RPSO performed better than the CPSO and the QPSO on problems which required prompt "unlearning" after a change (*e.g.* conflicting data present in the sliding window, numerous decision boundaries). The CPSO and the QPSO performed better than the RPSO on problems which required previously learned information to be preserved (*e.g.* no conflicting boundaries present, new decision boundaries derived from old decision boundaries).

7.2 Future Work

A number of possible future research topics have emerged from this study, and are briefly discussed below.

Overfitting Study

The experiments conducted for this study indicated that the dynamic PSOs are susceptible to minor overfitting. Possible causes of overfitting in dynamic PSOs are values chosen for V_{max}, the sigmoid activation function used in the neurons, and preserved diversity. All existing techniques to counteract overfitting were developed with static environments in mind. Adapting the existing techniques and developing new techniques to prevent overfitting in dynamic environments is an important topic for future research.

Correlation Between Problem Types and Dynamic Training Algorithms

The conclusions above suggested that dynamic classification problem attributes, such as the number of decision boundaries and the correlation between old and new boundaries, determine the optimal NN training algorithm for each problem. That is, the optimal training algorithm is problem-specific. Discovering the connection between particular dynamic problem parameters and corresponding optimal NN training algorithms would significantly help the progress of dynamic optimisation in general, and would aid practitioners to choose the most appropriate dynamic NN training algorithms for real-life dynamic optimisation problems without the wearisome exhaustive empirical testing.

Adaptive Algorithm Parameters

The dynamic PSO algorithms used in this study, as well as BP and RBP, were optimised once for each problem, afterwards the algorithm parameters remained static throughout the algorithm runs. Existing adaptive parameter strategies were developed with static environments in mind, thus they can not be blindly applied to dynamic problems. The efficiency of existing dynamic NN training algorithms can be improved by considering adaptive algorithm parameters, either by adapting the existing strategies accordingly, or by developing new self-adaptive parameter strategies specific to dynamic environments.

Dynamic Architecture Selection

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 problem with a fluctuating number of decision boundaries 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.

Active Search Space Bounds for PSO Training

The search space of the PSO is unbounded, and, although the NN weight space is not bounded either, the studies of NN error surfaces suggest that error surfaces tend to have large flat regions asymptotically approaching infinity [38, 55]. Unbounded exploration exhibited by the swarm is thus likely to lead to unfruitful search space regions. An investigation of NN error surfaces is required in order to obtain any guidelines to the possible bounds of the active search space.

Triggering Agents

Mining continuous data streams is usually carried out by continuously training a learner on the data. However, the pitfall of continuous training is the increased possibility of overfitting, since the training algorithm may train for too long between environment changes. A possible alternative to continuous training would be to have a separate change-detecting agent that would trigger the training algorithm when a change occurs, and suspend training when signs of overfitting are observed.

Dynamic Function Approximation

This study was dedicated to classification problems with concept drift. Another application area of dynamic PSO training would be to dynamic function approximation.

Other Population-Based Algorithms Applied to Dynamic NN Training

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 EC. It has also been applied to NN training before, and it would be interesting to determine the applicability and efficiency of dynamic EC algorithms to dynamic NN training.

Ensemble Learning

Ensemble learning is the most popular approach to handle concept drift to date. The performance of ensemble learning in the presence of concept drift could be significantly improved through the use of dynamic training algorithms in the classifiers, such as the dynamic PSOs. Using a dynamic PSO to train each NN in the NN ensemble would make every NN adjust to concept drift automatically, thus improving the overall performance of the ensemble. Application of dynamic training algorithms to ensemble learning in the presence of concept drift should be further investigated.

Bibliography

[1] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press, 1999.

[2] T. Bäck. On the behavior of evolutionary algorithms in dynamic environments. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 446–451. IEEE, 1998.

[3] E. Barnard and J. E. W. Holm. A comparative study of optimization techniques for backpropagation. *Neurocomputing*, 6(1):19–30, 1994.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[5] T. M. Blackwell. Particle swarms and population diversity. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(11):793–802, 2005.

[6] T. M. Blackwell. Particle swarm optimization in dynamic environments. In Shengxiang Yang *et al.*, editor, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 29–49. Springer Berlin / Heidelberg, 2007.

[7] T. M. Blackwell and P. J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 19–26. Morgan Kaufmann Publishers Inc., 2002.

[8] T. M. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, 3005:489–500, 2004.

[9] T. M. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

[10] W. J. Blackwell and F. W. Chen. *Neural networks in atmospheric remote sensing*. Artech House Publishers, 2009.

[11] J. Branke. Evolutionary algorithms for neural network design and training. Technical Report 322, University of Karlsrühe, Institute AIFB, January 1995.

[12] J. Branke. *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[13] J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck. A multi-population approach to dynamic optimization problems. *Adaptive computing in design and manufacturing*, 2000:299, 2000.

[14] Z. W. Cao, L. Y. Han, C. J. Zheng, Z. L. Ji, X. Chen, H. H. Lin, and Y. Z. Chen. Computer prediction of drug resistance mutations in proteins. *Drug Discovery Today*, 10(7):521–529, 2005.

[15] A. Carlisle and G. Dozier. Adapting particle swarm optimization to dynamic environments. In *Proceedings of the International Conference on Artificial Intelligence*, volume 1, pages 429–434, 2000.

[16] A. Carlisle and G. Dozier. Tracking changing extrema with particle swarm optimizer. Technical Report CSSE01-08, Auburn University, Auburn, Alabama, 2001.

[17] 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.

[18] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. In *IEE Proceedings on Circuits, Devices and Systems*, volume 139, pages 301–310. Institution of Electrical Engineers, 1992.

[19] F. Chu, Y. Wang, and C. Zaniolo. An adaptive learning approach for noisy data streams. In *Proceedings of the IEEE International Conference on Data Mining*, pages 351–354, November 2004.

[20] M. Clerc. *Particle swarm optimization*. Wiley-ISTE, 2006.

[21] M. Clerc and J. Kennedy. The particle swarm – explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, February 2002.

[22] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, D. C., 1990.

[23] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[24] G. W. Cottrell. Extracting features from faces using compression networks: Face, identity, emotion, and gender recognition using holons. In *Proceedings of the Connectionist Models Summer School*, pages 328–337, San Mateo, USA, 1990.

[25] A. Cournane and R. Hunt. An analysis of the tools used for the generation and prevention of spam. *Computers & Security*, 23(2):154–166, March 2004.

[26] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18(4-5):187–195, 2005.

[27] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[28] G. Dreyfus. *Neural networks: methodology and applications.* Springer, 2005.

[29] R. Durbin and D. E. Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1(1):133–142, 1989.

[30] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 84–88, 2000.

[31] R. C. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 94–100, 2001.

[32] A. P. Engelbrecht. *Computational Intelligence: An Introduction.* John Wiley & Sons, Chichester, December 2002.

[33] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons, 2005.

[34] A. P. Engelbrecht and A. Ismail. Training product unit neural networks. *Stability and Control: Theory and Applications*, 2(1–2):59–74, 1999.

[35] D. B. Fogel, L. J. Fogel, and V. W. Porto. Evolving neural networks. *Biological Cybernetics*, 63(6):487–493, 1990.

[36] A. S. Fraser. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10:484–491, 1957.

[37] B. Fritzke. Incremental learning of local linear mappings. In *Proceedings of the International Conference on Artificial Neural Networks*, volume 95, pages 217–222, Paris, France, October 1995.

[38] M. R. Gallagher. *Multi-layer Perceptron Error Surfaces: Visualization, Structure and Modelling.* PhD thesis, University of Queensland, St Lucia 4072, Australia, 2000.

[39] A. Gasper and P. Collard. From gas to artificial immune systems: improving adaptation in time dependent optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3. IEEE, 1999.

[40] J. Ghosh and Y. Shin. Efficient higher-order neural networks for classification and function approximation. *International Journal on Neural Systems*, 3(4):323–350, 1992.

[41] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[42] J. J. Grefenstette. *Genetic algorithms for changing environments.* Navy Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1992.

[43] J. J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3. IEEE, 1999.

[44] V. G. Gudise and G. K. Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 110–117, April 2003.

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

[46] P. B. Harrington. Sigmoid transfer functions in backpropagation neural networks. *Analytical Chemistry*, 65(15):2167–2168, 1993.

[47] M. H. Hassoun. *Fundamentals of artificial neural networks*. Bradford books. MIT Press, 1995.

[48] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1994.

[49] Y. Hirose, K. Yamashita, and S. Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4(1):61–66, 1991.

[50] J. E. W. Holm and E. C. Botha. Leap-frog is a robust algorithm for training neural networks. *Network: Computation in Neural Systems*, 10(1):1–13, 1999.

[51] G. C. Homans. Social behavior as exchange. *The American Journal of Sociology*, 63(6):597–606, 1958.

[52] T. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[53] 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.

[54] 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. IEEE, 2002.

[55] 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, October 1992.

[56] A. Hussain, J. J. Soraghan, and T. S. Durbani. A new neural network for nonlinear time-series modelling. *NeuroVest Journal*, pages 16–26, January 1997.

[57] J. Ilonen, J. K. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, 2003.

[58] M. S. Islam, S. M. Khaled, K. Farhan, M. A. Rahman, and J. Rahman. Modeling spammer behavior: Naive bayes vs. artificial neural networks. In *Proceedings of the International Conference on Information and Multimedia Technology*, pages 52–55. IEEE, 2009.

[59] A. Ismail. Training and optimization of product unit neural networks. Master's thesis, University of Pretoria, Pretoria, South Africa, November 2001.

[60] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307, 1988.

[61] D. J. Janson and J. F. Frenzel. Training product unit neural networks with genetic algorithms. *IEEE Expert*, 8(5):26–33, 1993.

[62] M. Jinli and S. Zhiyi. Application of combined neural networks in nonlinear function approximation. In *Proceedings of the World Congress on Intelligent Control and Automation*, number 2, pages 839–841, 2000.

[63] J. Kennedy. The behavior of particles. In V. W. Porto, N. Saravanan, and D. Waagen, editors, *Proceedings of the International Conference on Evolutionary Programming*, pages 581–589, 1998.

[64] J. Kennedy. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 1931–1938, 1999.

[65] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Perth, Australia, 1995.

[66] J. Kennedy and R. C. Eberhart. *New Ideas in Optimization*, chapter The Particle Swarm: Social Adaptation in Information-Processing Systems, pages 379–387. McGraw-Hill, 1999.

[67] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, USA, 2001.

[68] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the Congress on Evolutionary Computation*, pages 407–412, Honolulu, Hawaii, 2002.

[69] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[70] T. Krink, J. S. Vesterstrom, and J. Riget. Particle swarm optimisation with spatial particle extension. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1474–1479, 2002.

[71] T. Y. Kwok and D. Y. Yeung. Constructive feedforward neural networks for regression problems: A survey. Technical Report HKUST-CS95-43, Department of Computer Science, The Hong Kong University of Science & Technology, 1995.

[72] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43, 1990.

[73] B. Larder, D. Wang, A. Revell, J. Montaner, R. Harrigan, F. De Wolf, J. Lange, S. Wegner, L. Ruiz, M.J. Pérez-Elías, et al. The development of artificial neural networks to predict virological response to combination HIV therapy. *Antiviral therapy*, 12(1):15, 2007.

[74] M. Last. Online classification of nonstationary data streams. *Intelligent Data Analysis*, 6(2):129–147, April 2002.

[75] R. Lau. Adaptive statistical language modeling. Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1994.

[76] S. Lawrence, A. C. Tsoi, and A. D. Back. Function approximation with neural networks and local methods: bias, variance and smoothness. In *Proceedings of the Australian Conference on Neural Networks*, pages 16–21, 1996.

[77] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[78] F. Li, Q. H. Meng, S. Bai, J. G. Li, and D. Popescu. Probability-PSO algorithm for multi-robot based odor source localization in ventilated indoor environments. volume 5314 of *Lecture Notes in Computer Science*, pages 1206–1215. Springer Berlin / Heidelberg, 2008.

[79] X. Li and H. D. Khanh. Comparing particle swarms for tracking extrema in dynamic environments. In *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1772–1779, 2003.

[80] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation*, 11(7):1769–1796, 1999.

[81] D. P. Mandic and J. A. Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Adaptive and learning systems for signal processing, communications, and control. John Wiley, 2001.

[82] 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.

[83] R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle swarms for feedforward neural network training. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1895–1899, Honolulu, USA, 2002. IEEE.

[84] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[85] R. W. Morrison. Performance measurement in dynamic environments. In J. Branke, editor, *Proceedings of the Genetic and Evolutionary Computation Conference Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 5–8, 2003.

[86] R. W. Morrison. *Designing evolutionary algorithms for dynamic environments*. Natural computing series. Springer, 2004.

[87] O. Olorunda and A. P. Engelbrecht. Measuring exploration/exploitation in particle swarm using swarm diverity. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1128–1134, Hong Kong, 2008.

[88] A. E. Olsson. *Particle Swarm Optimization: Theory, Techniques and Applications*. Nova Science Publishers Inc, 2011.

[89] G. B. Orr and T. K. Leen. Momentum and optimal stochastic search. In M. C. Mozer, P. Smolensky, D. S. Touretzky, Elman J. L., and A. S. Weigend, edi-

tors, *Proceedings of the Connectionist Models Summer School*, Erlbaum Associates, 1993.

[90] K. E. Parsopoulos and M. N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Reference, 2010.

[91] D. W. Patterson. *Artificial Neural Networks: Theory and Applications*. Prentice Hall, USA, 1st edition, 1998.

[92] B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 365–372. IEEE, 1989.

[93] E. S. Peer, F. van den Bergh, and A. P. Engelbrecht. Using neighborhoods with guaranteed convergence PSO. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 235–242, Indianapolis, USA, 2003.

[94] R. Poli, J. Kennedy, and T. Blackwell. *Particle Swarms: The Second Decade*. Hindawi Publishing Corporation, 2008.

[95] V. W. Porto and D. B. Fogel. Alternative neural network training methods [Active sonar processing]. *IEEE Expert*, 10(3):16–22, 2002.

[96] L. V. Qiang and Y. U. Jin-shou. Self-organizing feature map neural network based on particle swarm optimizer and its application. *Control and Decision*, 20(10):1115, 2005.

[97] Z. Qin, J. Chen, Y. Liu, and J. Lu. Evolving RBF neural networks for pattern classification. In Y. Hao, J. Liu, Y. Wang, Y. Cheung, H. Yin, L. Jiao, J. Ma, and Y. C. Jiao, editors, *Computational Intelligence and Security*, volume 3801 of *Lecture Notes in Computer Science*, pages 957–964. Springer Berlin / Heidelberg, 2005.

[98] A. Rakitianskaia and A. P. Engelbrecht. Cooperative charged particle swarm optimiser. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 933–939, Hong Kong, 1–6 June 2008.

[99] A. Rakitianskaia and A. P. Engelbrecht. Training neural networks with PSO in dynamic environments. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 667–673. IEEE, 2009.

[100] A. Röbel. The dynamic pattern selection algorithm: Effective training and controlled generalization of backpropagation neural networks. Technical report, Technische Universität Berlin, 1994.

[101] R. Rojas. *Neural networks: a systematic introduction*. Springer-Verlag, 1996.

[102] L. Rokach. *Pattern classification using ensemble methods*. Series in machine perception and artificial intelligence. World Scientific, 2010.

[103] C. Romero, M. G. Valdez, and A. Alanis. A comparative study of machine learning techniques in blog comments spam filtering. In *The Proceedings of the International Joint Conference on Neural Networks*, pages 1–7. IEEE.

[104] D. Saad. *On-line learning in neural networks*. Publications of the Newton Institute. Cambridge University Press, 1998.

[105] J. C. Schlimmer and R. H. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, 1986.

[106] M. Settles, B. Rodebaugh, and T. Soule. Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. In *Proceedings of the Genetic and Evolutionary Computation Congress*, pages 148–149. Springer, 2003.

[107] R. S. Sexton, R. E. Dorsey, and J. D. Johnson. Optimization of neural networks: a comparative analysis of the genetic algorithm and simulated annealing. *European Journal of Operational Research*, 114(3):589–601, 1999.

[108] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In V. Porto, N. Saravanan, D. Waagen, and A. Eiben, editors, *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin / Heidelberg, 1998.

[109] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 1945–1950, 1999.

[110] J. Sietsma and R. J. F. Dow. Neural net pruning - why and how. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 325–333, San Diego, USA, 24–27 July 1988.

[111] W. N. Street and Y. S. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 377–382, San Francisco, California, 2001. ACM.

[112] H. Tang, K. C. Tan, and Y. Zhang. *Neural networks: computational models and applications*. Studies in computational intelligence. Springer, 2007.

[113] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Proceedings of the Congress on Evolutionary Computation*, volume 3. IEEE, 1999.

[114] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical Report TCD-CS-2004-15, Trinity College, Dublin, 2004.

[115] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections. In *Proceedings of the IEEE Symposium on Computer-Based Medical Systems*, pages 679–684. IEEE, 22–23 June 2006. DOI online: https://www.cs.tcd.ie/research_groups/mlg/kdp/publications/Tsymbal_CBMS2006.pdf.

[116] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68, 2008.

[117] F. Van den Bergh. *An Analysis of Particle Swarm Optimizers*. Doctoral dissertation, Department of Computer Science Scandinavica, University of Pretoria, Pretoria, South Africa, 2002.

[118] F. Van den Bergh and A. P. Engelbrecht. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26:84–90, 2000.

[119] A. B. van Wyk and A. P. Engelbrecht. Overfitting by PSO trained feedforward neural networks. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[120] A. Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1(1):39–46, 1989.

[121] E. A. Wan. Neural network classification: A bayesian interpretation. *IEEE Transactions on Neural Networks*, 1(4):303–305, 1990.

[122] D. Wang and B. Larder. Enhanced prediction of lopinavir resistance from genotype by use of artificial neural networks. *Journal of Infectious Diseases*, 188(5):653, 2003.

[123] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ACM international conference on Knowledge discovery and data mining*, pages 226–235, Washington, D.C., 2003. ACM.

[124] N. Watanasusin and S. Sanguansintukul. Classifying chief complaint in ear diseases using data mining techniques. In *International Conference on Digital Content, Multimedia Technology and its Applications*, pages 149–153, August 2011.

[125] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Proceedings of the Congress on Evolutionary Computation*, volume 3. IEEE, 1999.

[126] B. Wenerstrom and C. Giraud-Carrier. Temporal data mining in dynamic feature spaces. In *Proceedings of the International Conference on Data Mining*, pages 1141–1145, 2006.

[127] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*. PhD thesis, Harvard University, Boston, USA, 1974.

[128] 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.

[129] G. Widmer and Kubat M. Effective learning in dynamic environments by explicit context tracking. In *Proceedings of the European Conference on Machine Learning*, pages 227–243. Springer-Verlag, 1993.

[130] G. Widmer and Kubat M. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[131] P. M. Williams. Bayesian regularization and pruning using a laplace prior. *Neural Computation*, 7(1):117–143, 1995.

[132] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[133] X. Xiao, E. R. Dow, R. Eberhart, Z. B. Miled, and R. J. Oppelt. Gene clustering using self-organizing maps and particle swarm optimization. In *Proceedings of the International Parallel and Distributed Processing Symposium*, page 10. IEEE, 2003.

[134] S. Yang, Y. S. Ong, and Y. Jin. *Evolutionary computation in dynamic and uncertain environments.* Springer, 2007.

[135] Y. Yang and S. Elfayoumy. Anti-spam filtering using neural networks and baysian classifiers. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation*, pages 272–278. IEEE, 2007.

[136] X. Yao. Evolution of connectionist networks. In *Proceedings of the International Symposium on AI, Reasoning & Creativity*, pages 49–52, Queensland, Australia, 1991.

[137] X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.

[138] X. H. Yu and G. A. Chen. Efficient backpropagation learning using optimal learning rate and momentum. *Neural Networks*, 10(3):517–527, 1997.

[139] X. H. Yu, G. A. Chen, and S. X. Cheng. Dynamic learning rate optimization of the backpropagation algorithm. *IEEE Transactions on Neural Networks*, 6(3):669–677, 1995.

[140] T. Zhang. Class-size independent generalization analysis of some discriminative multi-category classification. *Advances in Neural Information Processing Systems*, 17:1625–1632, 2005.

[141] J. Zhao, M. Chen, and Q. Luo. Research of intrusion detection systems based on neural networks. In *International Conference on Communication Software and Networks*, pages 174–178. IEEE, 2011.

[142] J. M. Zurada. *Introduction to artificial neural systems.* West, 1992.

Appendix A

Acronyms

This appendix lists the acronyms used throughout this thesis. Acronyms are listed alphabetically, and typeset in bold. Each acronym's associated meaning is provided alongside:

BP	Back Propagation
CI	Computational Intelligence
CPSO	Charged PSO
EC	Evolutionary Computation
FFNN	Feedforward Neural Network
GA	Genetic Algorithm
MSE	Mean Squared Error
NN	Neural Network
PSO	Particle Swarm Optimisation
QPSO	Quantum PSO

RBP	Reinitialising Back Propagation
RPSO	Reinitialising PSO
SSE	Sum Squared Error

Appendix B

Symbols

This appendix lists the mathematical symbols used throughout this dissertation, and their definitions. The symbols used within each chapter are listed under separate sections. Each section lists only newly introduced symbols.

B.1 Chapter 2: Particle Swarm Optimisation

ω	Inertia weight
c_1	Cognitive coefficient
c_2	Social coefficient
l	Dimension index
n	Search space dimensionality
S_P	Total number of particles in a swarm
$P(t)$	Particle swarm
$\vec{r_1}, \vec{r_2}$	Random components
t	Time step
$\vec{v}_y(t)$	Velocity of particle y at time step t

$\vec{V}_{max}$	Maximum velocity
$U(0,1)$	Uniform random number distribution
$\vec{x}_y(t)$	Position of particle y at time step t
$\vec{x}_{gbest}(t)$	Global best position in a particle neighbourhood
$\vec{x}_{pbest}(t)$	Personal best position of a particle
X^n	Search space of a n-dimensional problem
y	Particle index

B.2 Chapter 3: Artificial Neural Networks

α	The momentum term in back propagation
η	The learning rate of back propagation
θ	The bias threshold
D	Set of NN data patterns
D_T	Set of NN training data patterns
D_G	Set of NN generalisation data patterns
E_p	NN error per patern
E_T	NN error on the training set
E_G	NN error on the generalisation set
f_{AN}	Activation function of a neuron
f_{o_k}	Activation function of the output unit k
f_{u_j}	Activation function of the hidden unit j
$fanin$	The number of connections leading to a neuron
I	Total number of input units
i	Index of an input unit
J	Total number of hidden units

j	Index of a hidden unit
K	Total number of output units
k	Index of an output unit
net	Weighted sum of inputs
o_k	k^{th} output unit
$o_{k,p}$	Output value produced by the k^{th} output unit on pattern p
P	Total number of data patterns
P_G	Total number of generalisation data patterns
P_T	Total number of training data patterns
p	Data pattern index
$\mathbb{R}^I$	Set of all real numbers in I dimensions
$t_{k,p}$	k^{th} target value of pattern p
$\vec{t}_p$	Target vector
u_j	Output of the j^{th} hidden unit
v_{ji}	Weight between the i^{th} input unit and j^{th} hidden unit
Δv_{ji}	Adjustment of the weight between i^{th} input unit and j^{th} hidden unit
W	NN weight vector, in other words, a set of all NN weights
w_{kj}	Weight between the j^{th} hidden unit and k^{th} output unit
Δw_{kj}	Adjustment of the weight between j^{th} hidden unit and k^{th} output unit
z_i	Output of the i^{th} input unit
$\vec{z}_p$	Input vector

B.3 Chapter 4: Dynamic Environments

$\varpi(t)$	Vector of time-dependent control parameters of the objective function
ω	Inertia weight

$\vec{a}_y$	Acceleration term of the charged PSO velocity update equation
$\vec{d}_{yg}(t)$	Distance between the y^{th} particle and the g^{th} particle at time step t
Q_y	Charge of particle y
R_c	Core radius
R_p	Perception limit
r_{cloud}	Radius of the quantum cloud
$\vec{x}^*(t)$	Optimum found at time step t

B.4 Chapter 6: Experimental Results

θ	Threshold value separating classes in the SEA classification problem
ρ_F	Generalisation factor
a_l	l^{th} linear coefficient
$\vec{b}$	Center of the sphere for the Dynamic Sphere problem
c	Constant value
d	Swarm diversity
F	Change frequency
$F(T)$	Algorithm fitness after iteration t
$F_{mean}(T)$	Collective mean fitness over T iterations
H_0	Null hypothesis
H_1	Alternative hypothesis
T	Total number of iterations
μ_1	Mean of the first sample being compared
μ_2	Mean of the second sample being compared
M	Number of data points
m	Data point

N	Number of environment changes
n_w	Total number of NN weights
P	Number of data patterns
P_w	Sliding window size
R	Sphere radius for the Dynamic Sphere problem
S	Step size by which the sliding window shifts
t_1, t_2	Scalar thresholds used for the Sliding Thresholds problem
T	Total number of iterations
V_{max}	Scalar maximum velocity
$\mathbf{x}_y$	Particle position y
$\bar{\mathbf{x}}$	The swarm center

Appendix C

Derived Publications

This appendix lists the publications derived from the work presented in this thesis.

- A. Rakitianskaia and Andries P. Engelbrecht. Cooperative Charged Particle Swarm Optimiser, *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 933–939, Hong Kong, 2008.
- A. Rakitianskaia and Andries P. Engelbrecht. Training Neural Networks with PSO in Dynamic Environments, *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 667–673, Trondheim, Norway, 2009.