Swarm Intelligence manuscript No.
(will be inserted by the editor)

Training Feedforward Neural Networks with Dynamic Particle Swarm Optimisation

A. S. Rakitianskaia · A. P. Engelbrecht

Received: date / Accepted: date

Abstract Particle swarm optimisation (PSO) has been successfully applied to train feedforward neural networks (NNs) in static environments. Many real-world problems to which NNs are applied are dynamic in the sense that the underlying data distribution changes over time. In the context of classification problems, this leads to concept drift where decision boundaries may change over time. This article investigates the applicability of dynamic PSO algorithms as NN training algorithms under the presence of concept drift.

Keywords Swarm Intelligence · Particle Swarm Optimization · Neural Networks · Dynamic Environments · Classification · Concept Drift

1 Introduction

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, and the like. 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, or new optima may emerge. Alternatively, for classification problems, decision boundaries that separate different classes may change over time. In both cases, the task of the optimisation algorithm is not only to find the optimal solution, but also to detect environment changes and promptly adapt to them, which might entail dismissing the old solution entirely. To be applicable in dynamic environments, optimisation algorithms

A. S. Rakitianskaia
E-mail: myearwen@gmail.com

A. P. Engelbrecht
E-mail: engel@cs.up.ac.za

developed for static environments have to be changed in order to include these extra targets.

An example of successfully adapting an algorithm originally developed for static environments to dynamic environments is particle swarm optimisation (PSO), a population-based optimisation technique that models social behaviour of a bird flock (Kennedy and Eberhart, 1995). Due to the success of the PSO in static environments, numerous variants of PSO that cater for dynamic changes have been developed to date, including the simple restarting PSO (Eberhart and Shi, 2001), the charged PSO based on electrostatic principles (Blackwell and Bentley, 2002), and the quantum PSO based on the model of an atom (Blackwell and Branke, 2006), amongst others.

However, one of the oldest fields of CI research, namely neural networks (NNs) (Bishop, 1995; Dreyfus, 2005; Zurada, 1992) – 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 that employ gradient descent are implicitly dynamic (Ismail, 2001; Kuncheva, 2004; Rokach, 2010), 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 (Street and Kim, 2001; Tsymbal, 2004; Tsymbal et al, 2008). 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 environment changes.

NNs are widely used in real life (Cournane and Hunt, 2004; Watanasusin and Sanguansintukul, 2011; Zhao et al, 2011), and it is necessary to ensure that NNs can be effectively trained in dynamic environments. PSO has been successfully applied to NN training before (Kennedy et al, 2001; Engelbrecht and Ismail, 1999; Gudise and Venayagamoorthy, 2003; Van Den Bergh and Engelbrecht, 2000), and in this work the applicability of dynamic PSOs 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 back propagation (Werbos, 1974) and a few popular dynamic PSOs on different dynamic classification problems is analysed.

The rest of the paper is organised as follows. Section 2 outlines the static PSO algorithm. Section 3 briefly discusses NNs and back propagation. Section 4 discusses dynamic environments and the dynamic PSOs used in this study. Section 5 discusses the existing approaches to NN training in dynamic environments. Section 6 presents the empirical study conducted. Section 7 provides a summary of the conclusions arrived at in this study, and lists possible future research directions.

2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a nature-inspired population-based optimisation technique. PSO operates on a set (referred to as a swarm) of particles, where every particle represents a candidate solution to the optimisation problem.

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

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

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

$$\mathbf{v}_y(t) = \omega\mathbf{v}_y(t-1) + c_1\mathbf{r}_1(\mathbf{x}_{pbest,y}(t) - \mathbf{x}_y(t)) + c_2\mathbf{r}_2(\mathbf{x}_{nbest,y}(t) - \mathbf{x}_y(t)) \tag{2}$$

where ω is the inertia weight (Shi and Eberhart, 1999), 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)) and *social* (third term of Equation (2)) components; $\mathbf{r_1}$ and $\mathbf{r_2}$ are vectors with each component sampled from a uniform distribution $U(0,1)$; $\mathbf{x}_{pbest,y}(t)$ is the personal best of particle y, or, in other words, the best position encountered by this particle so far; similarly, $\mathbf{x}_{nbest,y}(t)$ 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 $\mathbf{V}_{max}$ (Shi and Eberhart, 1998) 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. $\mathbf{V}_{max}$ is enforced by restricting $\mathbf{v}_y(t)$ per dimension:

$$v_{yl}(t) = \begin{cases} V_{max,l} & \text{if } v_{yl}(t) > V_{max,l} \\ -V_{max,l} & \text{if } v_{yl}(t) < -V_{max,l} \\ v_{yl}(t) & \text{otherwise} \end{cases} \tag{3}$$

Various PSO neighbourhood topologies have been proposed in the literature and applied in practice. A particle's neighbourhood is determined topologically rather than spatially, meaning that the distance between particles is determined by particle's indices and not the actual position in the search space (Kennedy, 1999). The structure and size of the neighbourhood determines the way in which information is shared between the particles. Thus, choosing an appropriate neighbourhood topology is crucial for the overall efficiency of the optimisation process.

The Von Neumann topology was first introduced by Kennedy and Mendes (2002). 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. Peer et al (2003) showed that the Von Neumann neighbourhood topology maintains swarm diversity 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 by Li and Khanh (2003) 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 (Li and Khanh, 2003). The Von Neumann neighbourhood topology was used in all experiments conducted for this study.

3 Neural Networks

A neural network is a simple mathematical model inspired by the learning mechanisms of the human brain. NNs are able to carry out tasks such as pattern recognition, classification, and function approximation (Dreyfus, 2005; Hassoun, 1995; Patterson, 1998; Rojas, 1996). A NN is essentially a collection of interconnected neurons aligned in layers. It was theoretically proved in Lawrence et al (1996); Jinli and Zhiyi (2000) that a NN can represent any non-linear mapping between the input space and the target space, provided the hidden layer has enough neurons. The NN itself is just a structure capable of representing a function, requiring to be trained on a problem in order to learn the mapping between input space and output space. Training can be supervised, unsupervised or reinforced. This paper deals with supervised NNs only. Such NNs work on a set of data patterns, where each pattern is a vector of problem inputs and the corresponding targets. Given a set of data patterns with known targets, a NN is trained to learn the mapping between the inputs and the targets. A trained NN is then capable of accurately approximating outputs for the data patterns it has never seen before. Such ability is known as the ability to generalise. The generalisation ability of a NN can deteriorate if the data set is not representative of the mapping to be learned, contains noise, if the NN is trained for too long, and if there are too many weights in the NN. The deterioration of the generalisation ability of the NN due to learning unnecessary information is known as overfitting. A NN that can not generalise has no practical use, since such NN will only be able to predict the outputs for the previously seen patterns. For a more extensive discussion of overfitting, refer to (Bishop, 1995; Blackwell and Chen, 2009).

The rest of this Section is dedicated to NN training algorithms applied in this study. Section 3.1 discusses back propagation. Section 3.2 explains how PSO can be applied to NN training, and outlines the advantages of PSO compared to back propagation.

3.1 Back Propagation

Many different supervised training algorithms exist. The most commonly used and popular algorithm is gradient descent back propagation (BP) (Werbos, 1974), which iteratively adjusts the NN weights to decrease the resulting NN output error.

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

BP is essentially a hill-climbing algorithm, and 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 of NNs. Population-based algorithms such as PSO (Engelbrecht and Ismail, 1999; Kennedy et al, 2001) address this problem by starting the search from multiple random initial positions.

3.2 Particale Swarm Optimiser Training

In order to train a NN using PSO, a fitness function has to be defined, which is usually simply the MSE. An appropriate representation of candidate solutions also 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 and biases in the NN. The PSO is then used, as discussed in Section 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 an effective NN training algorithm (Engelbrecht and Ismail, 1999; Gudise and Venayagamoorthy, 2003; Kennedy et al, 2001; Mendes et al, 2002; Van Den Bergh and Engelbrecht, 2000). PSO outperformed BP on a selection of classification, function approximation, and prediction problems. The advantages that PSO offers in comparison with BP 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 (Mendes et al, 2002).

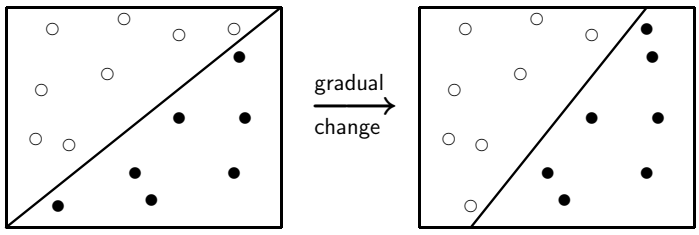
The major disadvantages of PSO, as compared to BP, are slower speed of convergence and more algorithm parameters to optimise before optimal performance can be expected.

4 Dynamic Environments

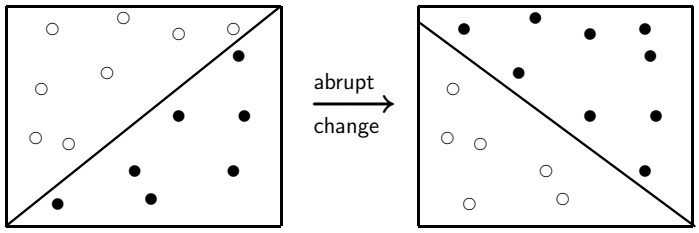
This section focuses on dynamic classification problems, also referred to as classification problems with concept drift, and the relevant dynamic optimisation algorithms. Section 4.1 discusses the phenomena of concept drift and its properties. Section 4.2 lists the existing approaches to concept drift. Section 4.3 discusses the dynamic PSO algorithms applied to train NNs for problems with concept drift in this study.

4.1 Concept Drift

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. In case of classification, 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, i.e. the mapping between inputs and targets will change. This phenomena is referred to as concept drift (Schlimmer and Granger, 1986). 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. The severity of concept drift may vary from gradual changes, when the decision boundaries shift slowly over time, to abrupt changes, when the old boundaries are abruptly replaced by new boundaries (Tsymbal, 2004). This property of concept drift is referred to as spatial severity. Figure 1 illustrates spatial severity of changes as applied to concept drift. Decision boundaries may change continuously, at regular time intervals, or unpredictably. This property of concept drift is referred to as temporal severity, or frequency of change.



(a) Gradual decision boundary change



(b) Abrupt decision boundary change

Fig. 1 Spatial severity applied to concept drift

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. 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.

4.2 Optimisation in the Presence of Concept Drift

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 (Klinkenberg, 2004; Last, 2002; Widmer and M., 1996). Three major categories of concept drift handling techniques can be distinguished, namely instance selection, instance weighting, and ensemble learning (Tsymbal et al, 2008).

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 (Last, 2002; Widmer and M., 1996). The goal of the learner is to discover 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 (Widmer and M., 1996). More complex forms of instance selection that delete noisy, irrelevant and redundant instances have also been developed (Delany et al, 2005; Last, 2002). The windowing approach to instance selection applied in this study is described in more detail in Section 6.

Instance weighting is sometimes used instead of instance selection with the learning algorithms that have the ability to process weighted instances (Klinkenberg, 2004). An example of such algorithms are support vector machines (SVMs) (Cortes and Vapnik, 1995). 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 (Rokach, 2010; Tsymbal, 2004). 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 (Tsymbal, 2004). 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 (Chu et al, 2004; Rokach, 2010; Street and Kim, 2001; Tsymbal, 2004; Tsymbal et al, 2008). 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.

4.3 Dynamic Particle Swarm Optimisation

A number of PSO algorithms for dynamic environments have been developed. These include the reinitialising PSO (Hu and Eberhart, 2002), the charged PSO based on electrostatic principles (Blackwell and Bentley, 2002), and the quantum PSO based on the model of an atom (Blackwell and Branke, 2006), amongst others. 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 environment changes.

Both phases are discussed below in the context of PSO.

4.3.1 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 a *sentry*, which is either a dedicated particle or a fixed point in the search space (Carlisle and Dozier, 2000; Carlisle and Dozler, 2002). The only difference between a normal particle and a sentry particle is that the sentry particles keep a record of their previous fitness values. In the beginning 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. The number of sentry particles to use in order to efficiently detect changes is problem dependent.

4.3.2 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 Engelbrecht (2005), 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 (Clerc and Kennedy, 2002; Van Den Bergh, 2002) 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), 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 (Engelbrecht, 2005).

A number of PSO variations have been developed, differing in the way that the above issues are addressed. A review of 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 (Hu and Eberhart, 2002). Reinitialisation ratio is problem dependent and should be chosen empirically.

Charged PSO: The charged PSO (Blackwell and Bentley, 2002) 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. Repelling forces are introduced by adding an acceleration term to the standard velocity equation (refer to Blackwell and Bentley (2002)). 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.

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 (Blackwell and Branke, 2006) 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.

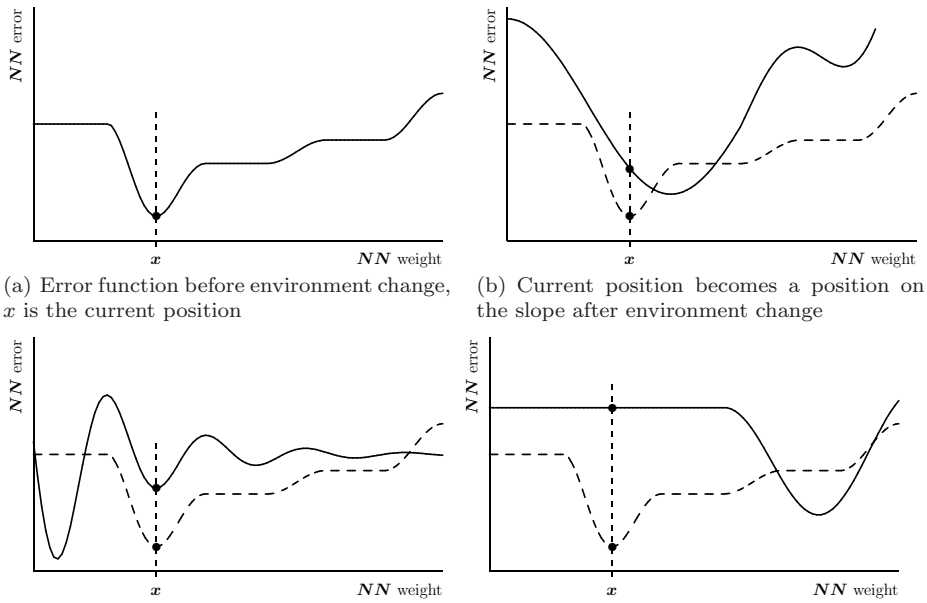
The problem of outdated memory is again addressed by complete re-evaluation of the swarm memory.

5 Neural Networks in Dynamic Environments

Back propagation is a gradient descent method, which implies a hill-climbing approach to training (Werbos, 1974). 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.

Implicit dynamism, however, does not eliminate the premature convergence problem. The error function hypersurface may have flat regions where gradient descent is inefficient, and local minima where the algorithm may get stuck (Fogel et al, 1990; Gallagher, 2000; Gudise and Venayagamoorthy, 2003; Hush et al, 1992). 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 2(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 2(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 2(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 2(d) illustrates a scenario where the current



(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

Fig. 2 Back Propagation in dynamic environments

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 (Fogel et al, 1990; Gudise and Venayagamoorthy, 2003; Porto and Fogel, 2002), 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) and reinitialising back propagation (RBP). RBP applied the standard BP algorithm between environment changes, and completely reinitialised NN weights and biases whenever an environment change occured. RBP was used to test the supposition, made in Chu et al (2004); Rokach (2010); Street and Kim (2001); Tsymbal (2004), 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 study suggested applying dynamic PSO to train NNs in the presence of concept drift.

6 Empirical Analysis

This section provides a description of the experimental procedure followed and experimental results obtained for this study. 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. Namely, the reinitialising PSO (RPSO), the charged PSO (CPSO), and the quantum PSO (QPSO). 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 (Duhain, 2011). The three algorithms listed above were chosen based on their relative popularity. The RPSO was chosen as the most natural, naive way of adapting the standard PSO to dynamic environments. The CPSO and the QPSO 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 Blackwell and Branke (2006); Blackwell and Bentley (2002); Blackwell (2005); Li et al (2008); Rakitianskaia and Engelbrecht (2008, 2009)). The performance of dynamic PSO algorithms is compared to the performance of BP and RBP, and the behaviour of these algorithms in the presence of concept drift of varying spatial and temporal severity is investigated on five different dynamic classification problems.

The rest of the section is structured as follows: Section 6.1 describes the experimental procedure followed. Section 6.1 discusses the data sets used. Performance measures are described in Section 6.2. Experimental results are presented in Section 6.3.

6.1 Experimental Procedure

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

6.1.1 Parameter Optimisation

An iterative approach to algorithm parameter optimisation was used. Algorithm 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. The discrete parameter value ranges used for BP and RBP are listed in Table 1, and the discrete parameter value ranges used for

the dynamic PSO algorithms are listed in Table 2. Optimal parameters obtained for each problem are listed in Table 3. All NNs used a single hidden layer. For all dynamic PSO 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 Eberhart and Shi (2000), 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. For the CPSO and the QPSO, 50% particles were labelled charged and quantum, respectively.

Table 1 Parameter Ranges for BP and RBP

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

Table 2 Parameter Ranges for the dynamic PSOs

Parameter	**Range**
V_{max}	$\{0.1,0.5,1,2,5,10,20,+\infty\}$
Swarm Size S_P	$\{15,20,30,50\}$
NN weight initialisation range w_{int}	$\{[-1,1],[-2,2],[-3,3],[-4,4],[-5,5]\}$
RPSO: Reinitalisation Ratio ϱ	$\{0.25,0.5,0.75,1.0\}$
CPSO: Charge Magnitude Q	$\{0.1,0.3,1,5,10,20\}$
QPSO: Radius R	$\{1,1.5,2,3,5,10\}$

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. In this study, the size of the window was set to M, thus

Table 3 Optimal Parameter Values

		SEA Concepts	Moving Hyperplane	Dynamic Sphere	Sliding Thresholds	Electricity Pricing
BP	w_{int}	$[-5; 5]$	$[-2, 2]$	$[-3, 3]$	$[-1, 1]$	$[-1, 1]$
	η	0.1	0.1	0.1	0.1	0.1
	α	0.7	0.7	0.3	0.1	0.7
RBP	w_{int}	$[-5; 5]$	$[-2, 2]$	$[-3, 3]$	$[-1, 1]$	$[-1, 1]$
	η	0.1	0.1	0.1	0.1	0.1
	α	0.7	0.7	0.3	0.2	0.7
RPSO	w_{int}	$[-3; 3]$	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$
	V_{max}	0.5	0.5	1	1	2
	S_P	50	30	50	50	30
	ϱ	0.25	0.5	0.5	0.75	0.25
CPSO	w_{int}	$[-3; 3]$	$[-2, 2]$	$[-5, 5]$	$[-1, 1]$	$[-1, 1]$
	V_{max}	0.5	0.5	0.1	2	5
	S_P	30	50	50	50	30
	Q	0.3	20	5	20	0.1
QPSO	w_{int}	$[-1; 1]$	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$	$[-3, 3]$
	V_{max}	0.5	0.1	0.1	2	5
	S_P	50	30	20	50	50
	R	1.5	1	1.5	2	1.5

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 3. 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 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 4. Conflicting boundaries make dynamic adaptation more challenging for the training algorithms.

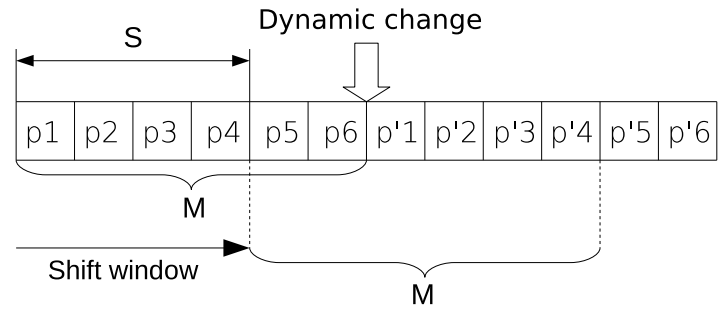


Fig. 3 Introducing new decision boundaries by window shifts

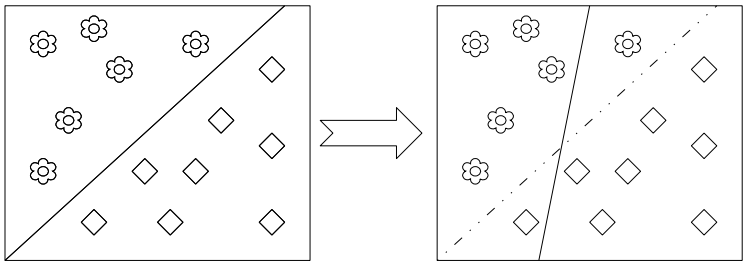


Fig. 4 Conflicting boundaries

Two major characteristics of a dynamic problem are: (a) spatial severity of changes, and (b) temporal severity of changes, as described in Section 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:

- 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.

Since an exhaustive evaluation of the considered training algorithms under all possible combinations of values of the above two variables 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 section.

For every scenario considered, every training algorithm had to traverse the entire data set. 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

$$T = F * \frac{P - P_w}{S} + F \tag{4}$$

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 S is the step size. The collective mean error results and the corresponding standard deviation values reported in this work were obtained after the number of iterations as given by equation (4). The window size was problem-dependent, and the values of both F and S were determined by the scenario in use.

6.2 Measuring NN Performance in Dynamic Environments

In dynamic environments, 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. This is why Morrison (2003) suggested that a representative performance measure in a dynamic environment should reflect algorithm performance "across the entire range of landscape dynamics". Morrison (2003) proposed that the collective mean fitness, or the average over all previous fitness values, be used as given by:

$$F_{mean}(T) = \frac{\sum_{t=1}^{T} F(t)}{T} \tag{5}$$

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 field, and refers to an applicable measure that reflects the quality of the current solution. In case of NN training, the 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 (1990) that minimisation of the MSE consequently minimises the probability of misclassification. Thus, MSE does not loose its meaning when classification problems are concerned.

A study of overfitting is outside the scope of this work, thus no measures were taken to prevent overfitting of training data. Counter-overfitting techniques developed to date were designed for static problems (Lau, 1994; Williams, 1995; Zhang, 2005), 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. The

generalisation error was nonetheless recorded throughout the experiments and reported in the analysis that follows.

All reported results are averages over 30 independent simulations. The next section describes the data sets used in the experiments.

6.3 Dynamic 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.

6.3.1 SEA Concepts

This problem was adopted from Street and Kim (2001); Tsymbal et al (2008). The data set consisted of 10 000 patterns, obtained by randomly generating 10 000 3-dimensional points, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10000}\}, \mathbf{x}_i \in [0; 10)^3, i = 1, \ldots, 10\ 000$. The generated points were divided into four equal concept blocks, 2500 points each, as illustrated in Figure 5. In each block, the class label of each data point $\mathbf{x}$ was determined as follows:

$$Classification(\mathbf{x}) = \begin{cases} Class\ A & \text{if } x_1 + x_2 \leq \theta \\ Class\ B & \text{otherwise,} \end{cases} \tag{6}$$

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.

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. 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:

$$n_w = (I + 1)J + (J + 1)K \tag{7}$$

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 (7), 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 4 were applied to the SEA concepts problem. As shown in Table 4, the values for both the frequency of change and the step size increase non-linearly, because the influence of parameter

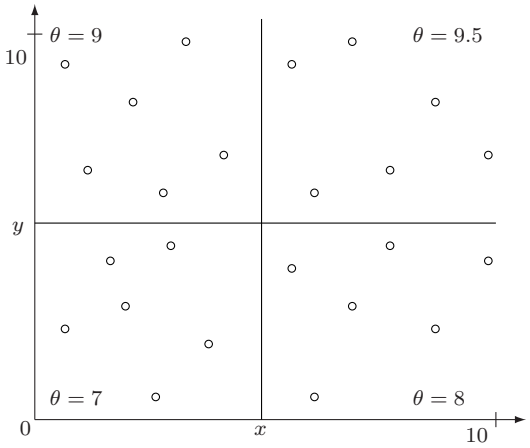


Fig. 5 SEA Concepts

values was expected to be stronger for small values. Change frequencies of 10, 50, 100, and 250 iterations were considered. Different levels of spatial severity were simulated by shifting the sliding window by 50, 100, 500, 1000, and 2500 patterns per step.

Table 4 Dynamic Scenarios for the SEA Concepts Problem

Frequency (F)	Step Size (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

6.3.2 Moving Hyperplane

The hyperplane is given by

$$\sum_{i=1}^{n} a_i x_i + c = a_0,$$

where n is the number of dimensions over which the hyperplane is defined, a_i for $i = 1, 2, \ldots, n$ are linear coefficients, and c is a constant. All points satisfying $\sum_{i=1}^{n} a_i x_i + c > a_0$ are labelled as class A, and all points satisfying $\sum_{i=1}^{n} a_i x_i + c \leq a_0$ 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, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{1000}\}$, was randomly generated such that $\mathbf{x}_j \in [0, 1]^{10}$, $j = 1, \ldots, 1000$. 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.

A NN with 10 input units, 6 hidden units, and a single output unit was used for the moving hyperplane problem. According to equation (7), 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. Parameter settings corresponding to each dynamic scenario are listed in Table 5.

Table 5 Dynamic Scenarios for the Moving Hyperplane

Frequency (F)	Step Size (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

6.3.3 Dynamic Sphere

This dynamic classification problem was obtained by generating a 3-dimensional hypersphere and using it to divide the space into two mutually exclusive classes. The hypersphere is given by

$$\sum_{i=1}^{n}(x_i + b_i) = R^2, \tag{8}$$

where R is the radius of the sphere, and $\mathbf{b}$ 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, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{1000}\}$ was randomly generated such that $\mathbf{x}_j \in [0, 1]^3,\ j = 1, \ldots, 1000$. The sphere was generated N times by randomising its centre point, $\mathbf{b} \in [0, 1]^3$, 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 (7).

Sixteen different dynamic scenarios as described in Section 6.3.2 and outlined in Table 5 were applied to the dynamic sphere problem. Parameter settings corresponding to each dynamic scenario are listed in Table 5.

6.3.4 Sliding Thresholds

For this problem, the Cartesian space was subdivided into three classes by means of two parallel linear thresholds. The two parallel linear thresholds, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, are given by

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

where t_1 and t_2 are constant values, therefore $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ 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:

$$Classification(\mathbf{x}) = \begin{cases} Class\ A & \text{if } x_1 \leq t_1 \\ Class\ B & \text{if } x_1 \geq t_2 \\ Class\ C & \text{otherwise} \end{cases} \tag{9}$$

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, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{1000}\}$ was randomly generated such that $\mathbf{x}_j \in [0, 1]^2,\ j = 1, \ldots, 1000$, and each point $\mathbf{x}_j$ was assigned a class value according to equation (9). 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 (7), 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 6.3.2 and outlined in Table 5 were considered for the sliding thresholds problem. 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.

6.3.5 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 Harries (1999).

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 (Harries, 1999).

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 (7), 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 6.3.2 and outlined in Table 5 were applied to the electricity pricing problem.

6.4 Analysis of Empirical Data

Table 6 lists the average ranks obtained by the considered algorithms under the five considered problems. Algorithms were ranked based on their collective mean E_T and E_G values reported in Appendix A, taking p-values reported in Appendix B into account. It follows from the overall average ranks in Table 6 that, on average, all three dynamic PSO algorithms obtained a higher rank than both BP and RBP. Thus, it can immediately be concluded that the dynamic PSOs proved to be a viable alternative to BP and RBP. RBP obtained the lowest average rank, thus proving to be the least successful approach to NN training on problems with concept drift. Figure 6(a) illustrates the E_G profiles obtained by the algorithms on the electricity pricing problem under infrequent gradual changes. Figure 6(a) illustrates that RBP's error not only fluctuated severely, but also failed to reach a minimum E_G as obtained by BP and the dynamic PSOs. 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

Table 6 Algorithm Ranking for the SEA Concepts Problem

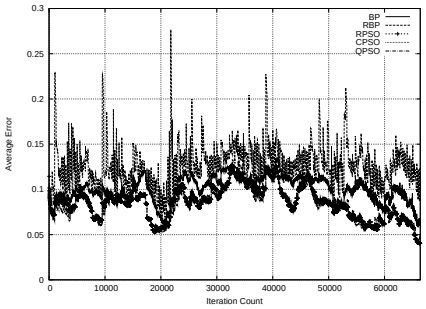
Problem		BP	RBP	RPSO	CPSO	QPSO
SEA	R(E_T)	2.925	4.775	**2.2**	2.9	**2.2**
Concepts	R(E_G)	2.6	4.625	2.675	**2.55**	**2.55**
Moving	R(E_T)	4	**2.09375**	3.1875	2.5	3.21875
Hyperplane	R(E_G)	2.90625	4.71875	2.75	**2.09375**	2.53125
Dynamic	R(E_T)	3.6875	**1.34375**	2.0625	3.6875	4.21875
Sphere	R(E_G)	**2.4375**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

environment changed. However, previously learned information remained useful when changes were spatially gradual (i.e. only a few new patterns were added to the sliding window), or when the new patterns were derived from the previous patterns (e.g. electricity pricing problem). Thus, the algorithms which made use of previously learned information had an advantage over RBP.

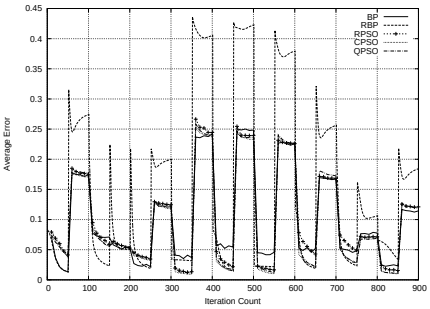
The ranks in Table 6 also show that RBP's E_T rank was often higher than the corresponding E_G rank, indicating that RBP did not generalise well. Figure 6(b) illustrates E_G profiles obtained by the algorithms on the moving hyperplane problem, under a semi-abrupt scenario, where exactly one half of the sliding window was replaced by new patterns whenever a change occurred. Figure 6(b) illustrates that RBP performed comparably well to other algorithms when the sliding window contained a single concept (complete replacement of patterns inside the window), but failed to generalise when two bordering and possibly conflicting concepts were present in the sliding window. Thus, RBP exhibited a strong sensitivity to the presence of stale data inside the sliding window. Poor generalisation of RBP can again be attributed to RBP's "forgetfulness": lack of memory of previous solutions prevented RBP from discerning between old and new data patterns.

Figure 6(c) illustrates a scenario under which both BP and RBP failed to optimise decision boundaries between environment changes. As discussed in Section 5, BP will fail to adapt to an environment change if the change of the error function landscape causes the current position of the NN weight vector to be in an unfruitful region such as a plateau of a local minimum. The failure of BP to adapt to changes illustrated in Figure 6(c) indicates that these changes trapped BP in a region from which BP could not escape. RBP was reinitialised after every change, thus RBP did not become trapped in such unfruitful regions as easily as BP. However, Figure 6(c) illustrates that RBP was also susceptible to stagnation between changes, and performed significantly worse than the dynamic PSOs. 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 located in an unfruitful region of the search space by reinitialisation.

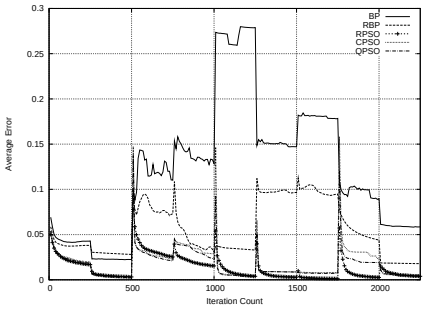
Table 6 indicates that RPSO obtained the top rank on the sliding thresholds problem, where, as explained in Section 6.3.4, multiple conflicting boundaries



(a) Electricity Pricing, E_G for D2



(b) Moving Hyperplane, E_G for B3

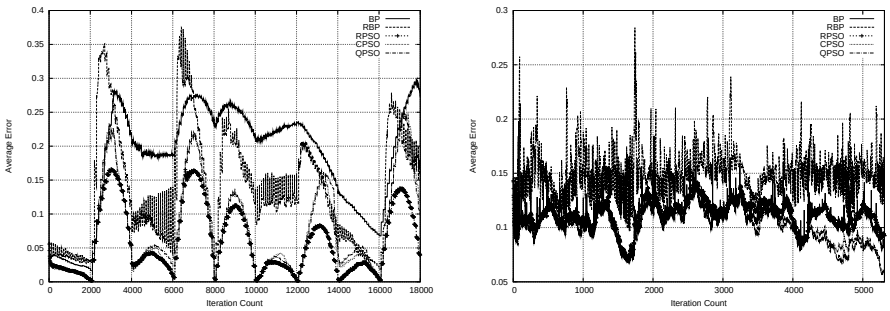


(c) Sliding Thresholds, E_G for D4

Fig. 6 Generalisation error profiles over time

could have been present. Figure 7(a) illustrates algorithm E_G profiles obtained for the sliding thresholds problem under scenario C1, and indicates that the RPSO's E_G profile peaked lower than that of the other algorithms. 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 happen to be chosen for reinitialisation. Weaker dependence on memory helps the RPSO to promptly "unlearn" obsolete information after every environment change, which proved useful on a problem where multiple conflicting boundaries were present.

Table 6 shows that CPSO and QPSO performed superior to RPSO on problems where the ability to preserve previously learned information was more important than the ability to promptly "unlearn" the stale data. Examples of such problems are the electricity pricing problem, where new electricity prices were derived from old electricity prices, and the SEA concepts problem, where no conflicting boundaries were present due to the fact that concept blocks were mutually exclusive. Figure 7(b) illustrates algorithm E_G profiles obtained for the electricity pricing under a gradual temporally severe scenario. Here, the CPSO and the QPSO were visibly more apt at tracking the moving optima than the RPSO. Under frequent changes, the RPSO did not have enough time to properly exploit the found optima, whereas the CPSO and the QPSO used their memory of previous solutions to promptly optimise the solution at hand.



(a) Sliding Thresholds, E_G for C1 (b) Electricity Pricing, E_G for A1

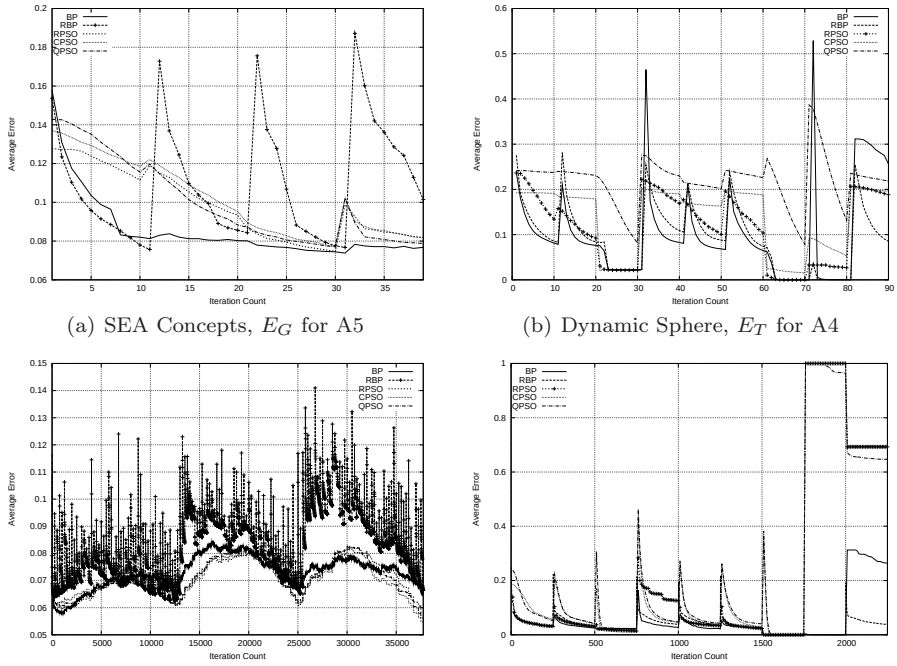
Fig. 7 Generalisation error profiles over time

Table 7 lists the best performing algorithms under various sliding window step sizes (i.e. spatial severity) considered for the five dynamic classification problems. Table 7 shows that the dynamic PSOs were more successful under gradual to semi-gradual scenarios, while BP and RBP were more successful under abrupt scenarios. An important advantage of the hill-climbing approach to NN training is the fast speed of convergence, and it is precisely this property which allowed BP and RBP to outperform the dynamic PSOs under abrupt changes. Figure 8(a) illustrates that under an abrupt scenario, the dynamic PSOs took longer to converge than both BP and RBP: the three dynamic PSOs took 25 iterations to reach the minimum attained by BP and RBP in 10 iterations. Figure 8(b) illustrates another abrupt scenario where the dynamic PSOs exhibited no signs of stagnation, but were unable to match the convergence speed of the hill-climbers. Thus, slow convergence speed of the dynamic PSOs often made these algorithms less efficient than BP under abrupt changes.

Table 7 Best Algorithms under varying Spatial Severity

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	-

However, figures 6(a) and 8(c) illustrate that, once the dynamic PSOs reached a minimum, the dynamic PSOs maintained the obtained minimum better than BP or RBP: under gradual scenarios, the dynamic PSOs were able to reach a lower minimum error than that obtained by BP and RBP. The dynamic PSOs also tracked gradual changes more closely than the hill-climbers. Gradual changes imply that little new data is added to the sliding window after every change, thus the new optimum is 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, while BP and RBP had to climb downhill towards the



(a) SEA Concepts, E_G for A5

(b) Dynamic Sphere, E_T for A4

(c) SEA Concepts, E_G for D1

(d) Dynamic Sphere, E_G for D4

Fig. 8 Training and Generalisation error profiles over time

changed optimum. The ability to explore a wider area around the current solution made the dynamic PSOs more apt than the hill-climbers at tracking gradual changes.

Table 8 lists the best performing algorithms under various change frequencies (i.e. temporal severity). Table 8 shows that BP was more successful under frequent changes, which corresponds to the already made observation that BP and RBP converged faster than the dynamic PSOs. However, BP came as the best performer only three times out of 20, as Table 8 shows, thus slower convergence did not prevent the dynamic PSOs from outperforming BP and RBP under most scenarios considered.

Table 8 also shows that BP was the best performer on the dynamic sphere problem under the least temporally severe scenarios, although the dynamic PSOs performed best under other change frequencies considered for the same problem. This indicates that although the dynamic PSOs typically took longer than BP to exploit a fruitful area, exploiting a fruitful area for too long can indeed result in poor performance due to over-exploitation. Figure 8(d) illustrates the algorithms' E_G profiles obtained on the dynamic sphere problem under abrupt infrequent changes. Figure 8(d) illustrates that, after iteration 1500, all dynamic PSOs obtained $E_G \approx 0$, and produced a very high error after the next environment change. Low temporal severity allowed the dynamic PSOs not only to find a good solution, but also to over-exploit it, leading the swarms to an unfruitful search space region (such as a plateau or a local minimum), from which the swarms struggled

Table 8 Best Algorithms under varying Temporal Severity

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

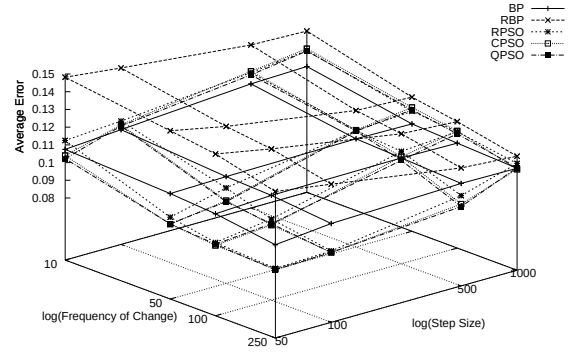
to escape after a change in the error landscape. The fact that BP did not exhibit over-exploitation under the same scenario indicates that the dynamic PSOs are more prone to over-exploitation than BP when left training for too long.

In general, it can be observed that the dynamic PSOs exhibited a stronger sensitivity to such characteristics of dynamic environments as spatial severity and temporal severity. Figures 9(a) and 9(b) illustrate the collective mean E_G values obtained by the algorithms under various dynamic scenarios for the electricity pricing problem and the moving hyperplane problem, respectively. In Figures 9(a) and 9(b), the performance of the dynamic PSOs deteriorated as both the temporal severity and the spatial severity increased. In Figure 9(b), the performance of dynamic PSOs also deteriorated under gradual scenarios of low temporal severity, because the dynamic PSOs were allowed to train for too long and thus over-exploited. In both Figures 9(a) and 9(b), BP produced more robust results than the dynamic PSOs: the BP error was less affected by the extent of spatial 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 scenario-specific optimisation than BP and RBP, but have a high potential to outperform the hill-climbers when properly optimised.

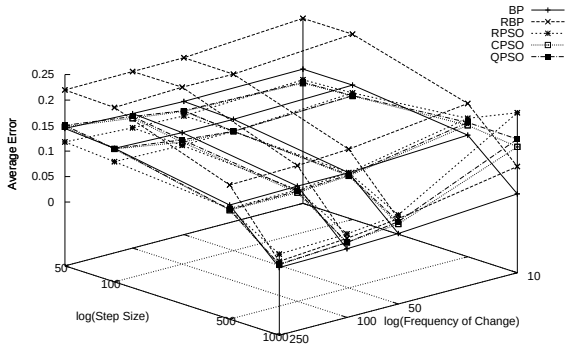
Figure 9(c) illustrates the collective mean E_G values obtained by the algorithms under various dynamic scenarios for the sliding thresholds problem. Figure 9(c) illustrates that the dynamic PSOs outperformed both BP and RBP under most scenarios considered. Superior performance of the dynamic PSOs on a problem where multiple decision boundaries were present indicates that the dynamic PSOs were less sensitive to the total number of decision boundaries, as well as the presence of stale data inside the sliding window. Thus, the dynamic PSOs may be expected to outperform BP on more rugged NN error surfaces.

7 Conclusions and Future Work

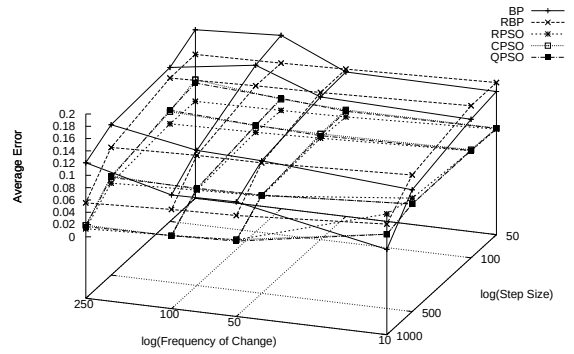
The main objective of this paper was to show that dynamic PSO algorithms have potential to be efficiently applied to NN training in the presence of concept drift. An experimental procedure was designed to compare the training algorithms on a representative selection of dynamic classification problems 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



(a) Average Generalisation Error Results for Electricity Pricing



(b) Average Generalisation Error Results for Moving Hyperplane



(c) Average Generalisation Error Results for Sliding Thresholds

Fig. 9 Average generalisation error results across all dynamic scenarios considered

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: the RPSO, the CPSO, and the QPSO.

The obtained empirical results showed the dynamic PSOs to be viable NN training algorithms in the context of dynamic classification problems. The dynamic PSOs exploited and tracked decision boundaries better than BP and RBP, allowing the dynamic PSOs to outperform BP and RBP under scenarios exhibiting infrequent to moderately infrequent gradual changes. BP and RBP converged faster than the dynamic PSOs, which enabled BP to outperform the dynamic PSOs under frequent abrupt changes. RBP exhibited the worst performance under most scenarios for most problems, thus showing that a complete reinitialisation of NN weights is an inefficient approach to handling concept drift. The dynamic PSOs exhibited stronger sensitivity to the extent of temporal and spatial severity than BP and RBP. Therefore, the dynamic PSOs require scenario-specific optimisation. However, the 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. Thus, 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).

Further research will include a study of overfitting behaviour exhibited by the dynamic PSOs applied to NN training under concept drift. The applicability of dynamic PSO NN training to dynamic problems other than classification problems, e.g. dynamic function approximation, also needs to be investigated. The dependence of the dynamic PSO performance on choosing optimal algorithm parameters can be lessened by developing self-adapting parameter strategies for dynamic environments. It would also be interesting to apply other dynamic population-based techniques, such as dynamic genetic algorithms, to NN training in dynamic environments. The most popular approach to concept drift handling, namely ensemble learning, can also be improved by employing dynamic PSOs as training algorithms for the single classifiers involved.

References

Bishop CM (1995) Neural Networks for Pattern Recognition. Oxford University Press

Blackwell TM (2005) Particle swarms and population diversity. Soft Computing-A Fusion of Foundations, Methodologies and Applications 9(11):793–802

Blackwell TM, Bentley PJ (2002) Dynamic search with charged swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers Inc., pp 19–26, URL `http://0-portal.acm.org.innopac.up.ac.za/citation.cfm?id=646205.682961`

Blackwell TM, Branke J (2006) Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10(4):459–472

Blackwell WJ, Chen FW (2009) Neural networks in atmospheric remote sensing. Artech House Publishers, URL http://books.google.co.za/books?id=Ija-H2uxejwC

Carlisle A, Dozier G (2000) Adapting particle swarm optimization to dynamic environments. In: Proceedings of the International Conference on Artificial Intelligence, vol 1, pp 429–434

Carlisle A, Dozler G (2002) Tracking changing extrema with adaptive particle swarm optimizer. In: Proceedings of the 5th Biannual World Automation Congress, IEEE, vol 13, pp 265–270

Chu F, Wang Y, Zaniolo C (2004) An adaptive learning approach for noisy data streams. In: Proceedings of the IEEE International Conference on Data Mining, pp 351–354

Clerc M, Kennedy J (2002) The particle swarm – explosion, stability and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1):58–73

Cortes C, Vapnik V (1995) Support-vector networks. Machine learning 20(3):273–297

Cournane A, Hunt R (2004) An analysis of the tools used for the generation and prevention of spam. Computers & Security 23(2):154–166

Delany SJ, Cunningham P, Tsymbal A, Coyle L (2005) A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems 18(4-5):187–195

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7:1–30

Dreyfus G (2005) Neural networks: methodology and applications. Springer, URL http://books.google.co.za/books?id=g2J4J2bLgRQC

Duhain JGOL (2011) Particle swarm optimisation in dynamically changing environments - an empirical study. Master's thesis, University of Pretoria, Pretoria, South Africa

Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol 1, pp 84–88

Eberhart RC, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol 1, pp 94–100

Engelbrecht AP (2005) Fundamentals of Computational Swarm Intelligence. John Wiley & Sons

Engelbrecht AP, Ismail A (1999) Training product unit neural networks. Stability and Control: Theory and Applications 2(1–2):59–74

Fogel DB, Fogel LJ, Porto VW (1990) Evolving neural networks. Biological Cybernetics 63(6):487–493

Gallagher MR (2000) Multi-layer perceptron error surfaces: Visualization, structure and modelling. PhD thesis, University of Queensland, St Lucia 4072, Australia

Gudise VG, Venayagamoorthy GK (2003) Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In:

Proceedings of the IEEE Swarm Intelligence Symposium, pp 110–117
Harries M (1999) Splice-2 comparative evaluation: Electricity pricing. Tech. Rep. UNSW-CSE-TR-9905, Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney 2052, Australia
Hassoun MH (1995) Fundamentals of artificial neural networks. Bradford books, MIT Press, URL http://books.google.co.za/books?id=Otk32Y3QkxQC
Hu X, Eberhart RC (2002) Adaptive particle swarm optimization: Detection and response to dynamic systems. In: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, vol 2, pp 1666–1670
Hush DR, Horne B, Salas JM (1992) Error surfaces for multilayer perceptrons. IEEE Transactions on Systems, Man and Cybernetics 22(5):1152–1161
Ismail A (2001) Training and optimization of product unit neural networks. Master's thesis, University of Pretoria, Pretoria, South Africa
Jinli M, Zhiyi S (2000) Application of combined neural networks in nonlinear function approximation. In: Proceedings of the World Congress on Intelligent Control and Automation, 2, pp 839–841
Kennedy J (1999) Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol 3, pp 1931–1938
Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, vol IV, pp 1942–1948
Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: Proceedings of the Congress on Evolutionary Computation, Honolulu, Hawaii, pp 407–412
Kennedy J, Eberhart RC, Shi Y (2001) Swarm Intelligence. Morgan Kaufmann Publishers, USA
Klinkenberg R (2004) Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis 8(3):281–300
Kuncheva L (2004) Classifier ensembles for changing environments. In: Roli F, Kittler J, Windeatt T (eds) Multiple Classifier Systems, Lecture Notes in Computer Science, vol 3077, Springer Berlin / Heidelberg, pp 1–15
Last M (2002) Online classification of nonstationary data streams. Intelligent Data Analysis 6(2):129–147, URL http://dl.acm.org/citation.cfm?id=1293986.1293988
Lau R (1994) Adaptive statistical language modeling. Master's thesis, MIT Department of Electrical Engineering and Computer Science
Lawrence S, Tsoi AC, Back AD (1996) Function approximation with neural networks and local methods: bias, variance and smoothness. In: Proceedings of the Australian Conference on Neural Networks, pp 16–21
Li F, Meng QH, Bai S, Li JG, Popescu D (2008) Probability-PSO algorithm for multi-robot based odor source localization in ventilated indoor environments. Lecture Notes in Computer Science, vol 5314, Springer Berlin / Heidelberg, pp 1206–1215
Li X, Khanh HD (2003) Comparing particle swarms for tracking extrema in dynamic environments. In: Proceedings of the Congress on Evolutionary Computation, vol 3, pp 1772–1779
Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Annals of Mathematical Statistics 18(1):50–

60

Mendes R, Cortez P, Rocha M, Neves J (2002) Particle swarms for feedforward neural network training. In: Proceedings of the International Joint Conference on Neural Networks, IEEE, Honolulu, USA, vol 2, pp 1895–1899

Morrison RW (2003) Performance measurement in dynamic environments. In: Branke J (ed) Proceedings of the Genetic and Evolutionary Computation Conference Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp 5–8

Patterson DW (1998) Artificial Neural Networks: Theory and Applications, 1st edn. Prentice Hall, USA

Peer ES, van den Bergh F, Engelbrecht AP (2003) Using neighborhoods with guaranteed convergence PSO. In: Proceedings of the IEEE Swarm Intelligence Symposium, Indianapolis, USA, pp 235–242

Porto VW, Fogel DB (2002) Alternative neural network training methods [Active sonar processing]. IEEE Expert 10(3):16–22

Rakitianskaia A, Engelbrecht AP (2008) Cooperative charged particle swarm optimiser. In: Proceedings of the IEEE Congress on Evolutionary Computation, Hong Kong, pp 933–939

Rakitianskaia A, Engelbrecht AP (2009) Training neural networks with PSO in dynamic environments. In: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, pp 667–673

Rojas R (1996) Neural networks: a systematic introduction. Springer-Verlag, URL `http://books.google.co.za/books?id=txsjjYzFJS4C`

Rokach L (2010) Pattern classification using ensemble methods. Series in machine perception and artificial intelligence, World Scientific, URL `http://books.google.co.za/books?id=4qnUwdoaVbsC`

Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Machine Learning 1(3):317–354, DOI {10.1007/BF00116895}, URL `http://dx.doi.org/10.1007/BF00116895`

Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: Porto V, Saravanan N, Waagen D, Eiben A (eds) Evolutionary Programming VII, Lecture Notes in Computer Science, vol 1447, Springer Berlin / Heidelberg, pp 591–600, URL `http://dx.doi.org/10.1007/BFb0040810`

Shi Y, Eberhart RC (1999) Empirical study of particle swarm optimisation. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol 3, pp 1945–1950

Street WN, Kim YS (2001) A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, ACM, San Francisco, California, pp 377–382

Tsymbal A (2004) The problem of concept drift: Definitions and related work. Tech. Rep. TCD-CS-2004-15, Trinity College, Dublin

Tsymbal A, Pechenizkiy M, Cunningham P, Puuronen S (2008) Dynamic integration of classifiers for handling concept drift. Information Fusion 9(1):56–68

Van Den Bergh F (2002) An analysis of particle swarm optimizers. Doctoral dissertation, Department of Computer Science Scandinavica, University of Pretoria, Pretoria, South Africa

Van Den Bergh F, Engelbrecht AP (2000) Cooperative learning in neural networks using particle swarm optimizers. South African Computer Journal 26:84–90

Wan EA (1990) Neural network classification: A bayesian interpretation. IEEE Transactions on Neural Networks 1(4):303–305

Watanasusin N, Sanguansintukul S (2011) Classifying chief complaint in ear diseases using data mining techniques. In: International Conference on Digital Content, Multimedia Technology and its Applications, pp 149–153

Werbos PJ (1974) Beyond regression: New tools for prediction and analysis in the behavioural sciences. PhD thesis, Harvard University, Boston, USA

Widmer G, M K (1996) Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1):69–101

Williams PM (1995) Bayesian regularization and pruning using a laplace prior. Neural Computation 7(1):117–143

Zhang T (2005) Class-size independent generalization analysis of some discriminative multi-category classification. Advances in Neural Information Processing Systems 17:1625–1632

Zhao J, Chen M, Luo Q (2011) Research of intrusion detection systems based on neural networks. In: International Conference on Communication Software and Networks, IEEE, pp 174–178

Zurada JM (1992) Introduction to artificial neural systems. West, URL `http://books.google.co.za/books?id=wLFQAAAAMAAJ`

A Accuracy

This appendix reports the collective mean E_T and E_G values obtained by the algorithms for the five dynamic classification problems considered. Tables 9 and 10 list E_T and E_G obtained for the SEA concepts problem. Table 11 lists E_T and E_G obtained for the moving hyperplane problem. Table 12 lists E_T and E_G obtained for the dynamic sphere problem. Table 13 lists E_T and E_G obtained for the sliding thresholds problem. Table 14 lists E_T and E_G obtained for the electricity pricing problem.

B Statistical significance

Whenever experimental results were compared, the two-tailed non-parametric Mann-Whitney U test (Mann and Whitney, 1947) was used to determine whether the difference in algorithm performance was statistically significant. The choice of the significance test is based on Demšar (2006), 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 (printed in bold in the tables that follow). For the sake of convenience, all p-values were bounded below by 0.0001. Tables 15 and 16 list the p-values obtained for the SEA concepts problem. Tables 17 and 18 list the p-values obtained for the moving hyperplane problem. Tables 19 and 20 list the p-values obtained for the dynamic sphere problem. Tables 21 and 22 list the p-values obtained for the sliding thresholds problem. Tables 23 and 24 list the p-values obtained for the electricity pricing problem.

Table 9 SEA Concepts Results for Scenarios A1 to B5

Scenario			BP	RBP	RPSO	CPSO	QPSO
A1	E_T	Mean	0.074272	0.11765	0.073435	0.074025	0.073285
		Std. Dev.	0.00206	0.004987	0.000793	0.000728	0.000881
	E_G	Mean	0.076167	0.117498	0.074669	0.074008	0.074031
		Std. Dev.	0.004317	0.007621	0.004909	0.004829	0.004996
A2	E_T	Mean	0.075578	0.117573	0.074902	0.075401	0.074578
		Std. Dev.	0.002115	0.006597	0.000935	0.000924	0.000829
	E_G	Mean	0.074614	0.118017	0.074139	0.076178	0.075184
		Std. Dev.	0.004726	0.006797	0.006028	0.005241	0.00473
A3	E_T	Mean	0.080557	0.118469	0.081363	0.081821	0.081712
		Std. Dev.	0.012953	0.016798	0.001426	0.002034	0.001563
	E_G	Mean	0.082473	0.116704	0.083129	0.08162	0.080766
		Std. Dev.	0.012549	0.017312	0.00422	0.003807	0.003235
A4	E_T	Mean	0.081234	0.117146	0.089078	0.090294	0.089353
		Std. Dev.	0.012918	0.023963	0.003179	0.003153	0.002067
	E_G	Mean	0.082579	0.115934	0.090288	0.090222	0.091455
		Std. Dev.	0.011709	0.023393	0.00601	0.004523	0.005
A5	E_T	Mean	0.085228	0.117479	0.097821	0.101398	0.099188
		Std. Dev.	0.036716	0.023246	0.005196	0.005385	0.005198
	E_G	Mean	0.086351	0.109948	0.098403	0.101769	0.099901
		Std. Dev.	0.034788	0.019195	0.006403	0.006693	0.00484
B1	E_T	Mean	0.074078	0.087951	0.071208	0.071717	0.071246
		Std. Dev.	0.001688	0.00329	0.000677	0.000875	0.000872
	E_G	Mean	0.074755	0.096313	0.072236	0.071091	0.071703
		Std. Dev.	0.005007	0.00589	0.005498	0.004605	0.004318
B2	E_T	Mean	0.073412	0.087863	0.071659	0.07204	0.071649
		Std. Dev.	0.001436	0.003877	0.00059	0.000625	0.000882
	E_G	Mean	0.074548	0.095976	0.073009	0.073066	0.073669
		Std. Dev.	0.005521	0.005494	0.005974	0.00437	0.005507
B3	E_T	Mean	0.07414	0.086878	0.07347	0.073609	0.073475
		Std. Dev.	0.00119	0.007596	0.000578	0.000684	0.000735
	E_G	Mean	0.074036	0.092688	0.073569	0.074822	0.073081
		Std. Dev.	0.003998	0.0074	0.003802	0.003874	0.00474
B4	E_T	Mean	0.074784	0.09203	0.075825	0.076371	0.075958
		Std. Dev.	0.001933	0.015599	0.000901	0.001175	0.000959
	E_G	Mean	0.074867	0.097183	0.077073	0.077229	0.075926
		Std. Dev.	0.005506	0.015191	0.005139	0.005933	0.005911
B5	E_T	Mean	0.072648	0.083178	0.073692	0.074436	0.073581
		Std. Dev.	0.016249	0.008553	0.001728	0.001995	0.001283
	E_G	Mean	0.072121	0.083807	0.075357	0.074777	0.074403
		Std. Dev.	0.017849	0.009732	0.005735	0.005022	0.004463

Table 10 SEA Concepts Results for Scenarios C1 to D5

Scenario			BP	RBP	RPSO	CPSO	QPSO
C1	E_T	Mean	0.073506	0.077631	0.070671	0.07088	0.070579
		Std. Dev.	0.001122	0.002366	0.000646	0.000762	0.00085
	E_G	Mean	0.072902	0.086025	0.071146	0.071306	0.072789
		Std. Dev.	0.003983	0.005988	0.004587	0.004753	0.005814
C2	E_T	Mean	0.073749	0.077723	0.070901	0.071195	0.071195
		Std. Dev.	0.001783	0.002391	0.000572	0.000732	0.000575
	E_G	Mean	0.072401	0.086938	0.071495	0.072334	0.071782
		Std. Dev.	0.005544	0.005286	0.005306	0.004217	0.004171
C3	E_T	Mean	0.074033	0.079415	0.072101	0.072522	0.072109
		Std. Dev.	0.001444	0.006442	0.000826	0.00072	0.000685
	E_G	Mean	0.074472	0.088763	0.072631	0.072996	0.072983
		Std. Dev.	0.004278	0.010017	0.005729	0.004072	0.004936
C4	E_T	Mean	0.076934	0.07673	0.074013	0.074196	0.073903
		Std. Dev.	0.014302	0.005086	0.000879	0.001033	0.000713
	E_G	Mean	0.0765	0.081602	0.074156	0.074275	0.07515
		Std. Dev.	0.01565	0.007463	0.00472	0.004435	0.004517
C5	E_T	Mean	0.067721	0.071498	0.069651	0.070355	0.069229
		Std. Dev.	0.004688	0.001767	0.001247	0.001678	0.001408
	E_G	Mean	0.069545	0.071627	0.070552	0.071822	0.06982
		Std. Dev.	0.004448	0.005087	0.004322	0.00472	0.004534
D1	E_T	Mean	0.073396	0.073543	0.070056	0.070774	0.070851
		Std. Dev.	0.001278	0.001375	0.000934	0.001174	0.001382
	E_G	Mean	0.072911	0.083022	0.070433	0.070202	0.07095
		Std. Dev.	0.00406	0.004974	0.004267	0.003386	0.005426
D2	E_T	Mean	0.07305	0.073585	0.070082	0.070429	0.07056
		Std. Dev.	0.001589	0.00157	0.000753	0.000892	0.000799
	E_G	Mean	0.073786	0.082509	0.071375	0.07152	0.07096
		Std. Dev.	0.004696	0.006086	0.004665	0.004626	0.00482
D3	E_T	Mean	0.073626	0.074375	0.070449	0.071185	0.070834
		Std. Dev.	0.001459	0.005637	0.000807	0.000702	0.000589
	E_G	Mean	0.073364	0.08254	0.073815	0.071683	0.070073
		Std. Dev.	0.005627	0.007563	0.005382	0.004437	0.003652
D4	E_T	Mean	0.072973	0.073241	0.072299	0.072747	0.072287
		Std. Dev.	0.001673	0.002844	0.000698	0.000734	0.000805
	E_G	Mean	0.075229	0.077941	0.072563	0.07269	0.073863
		Std. Dev.	0.005077	0.005381	0.004719	0.004979	0.005888
D5	E_T	Mean	0.065106	0.071554	0.065198	0.065742	0.065092
		Std. Dev.	0.004135	0.010663	0.001192	0.001147	0.000992
	E_G	Mean	0.064206	0.06982	0.066019	0.067239	0.066777
		Std. Dev.	0.005316	0.013026	0.006596	0.005103	0.004757

Table 11 Moving Hyperplane Results for Scenarios A1 to D4

Scenario			BP	RBP	RPSO	CPSO	QPSO
A1	E_T	Mean	0.155488	0.114547	0.112531	0.107024	0.108173
		Std. Dev.	0.002328	0.001191	0.000872	0.001584	0.00201
	E_G	Mean	0.132911	0.237748	0.117451	0.110943	0.112485
		Std. Dev.	0.004695	0.004665	0.003418	0.004261	0.003668
A2	E_T	Mean	0.156778	0.113846	0.11839	0.111091	0.113791
		Std. Dev.	0.002442	0.001363	0.001208	0.001421	0.001672
	E_G	Mean	0.138453	0.238293	0.123117	0.117111	0.117842
		Std. Dev.	0.005059	0.006099	0.004598	0.004598	0.004111
A3	E_T	Mean	0.13741	0.103456	0.134078	0.12389	0.129902
		Std. Dev.	0.001797	0.002951	0.003818	0.0039	0.005268
	E_G	Mean	0.113383	0.175411	0.146382	0.132682	0.138235
		Std. Dev.	0.002645	0.006053	0.009233	0.005736	0.007295
A4	E_T	Mean	0.033327	0.08689	0.101326	0.106926	0.121284
		Std. Dev.	0.003651	0.005457	0.005489	0.008617	0.01013
	E_G	Mean	0.029691	0.083383	0.188668	0.121909	0.137007
		Std. Dev.	0.004585	0.00689	0.053221	0.010094	0.012799
B1	E_T	Mean	0.1504	0.067478	0.101187	0.107609	0.108914
		Std. Dev.	0.0020	0.000596	0.00147	0.004842	0.004429
	E_G	Mean	0.136252	0.221859	0.10774	0.116852	0.11807
		Std. Dev.	0.00552	0.005365	0.003387	0.006497	0.006523
B2	E_T	Mean	0.150329	0.067098	0.103442	0.103015	0.104123
		Std. Dev.	0.001864	0.000852	0.001701	0.002516	0.002512
	E_G	Mean	0.132623	0.220962	0.108933	0.109161	0.108856
		Std. Dev.	0.005332	0.006585	0.003523	0.004026	0.003687
B3	E_T	Mean	0.131567	0.056117	0.095947	0.091132	0.093299
		Std. Dev.	0.001894	0.001308	0.001563	0.001667	0.001708
	E_G	Mean	0.100828	0.14698	0.101191	0.094755	0.09686
		Std. Dev.	0.003609	0.005182	0.0039	0.003897	0.003581
B4	E_T	Mean	0.011053	0.04116	0.038352	0.029461	0.032953
		Std. Dev.	0.000871	0.001592	0.003989	0.003598	0.004226
	E_G	Mean	0.013329	0.042341	0.049599	0.031942	0.03649
		Std. Dev.	0.001963	0.003982	0.022523	0.004398	0.005777
C1	E_T	Mean	0.149038	0.05963	0.098834	0.115017	0.119668
		Std. Dev.	0.002822	0.000607	0.001292	0.005976	0.006391
	E_G	Mean	0.138761	0.221091	0.11088	0.129655	0.133049
		Std. Dev.	0.007095	0.006619	0.004162	0.008136	0.007534
C2	E_T	Mean	0.149106	0.059685	0.100531	0.105563	0.109364
		Std. Dev.	0.001946	0.001031	0.001239	0.004117	0.005167
	E_G	Mean	0.132797	0.221858	0.107885	0.114033	0.118749
		Std. Dev.	0.005508	0.005652	0.003989	0.004939	0.00727
C3	E_T	Mean	0.128096	0.046808	0.088614	0.084396	0.086369
		Std. Dev.	0.001797	0.001316	0.001428	0.001449	0.001536
	E_G	Mean	0.10101	0.141273	0.093955	0.08831	0.091123
		Std. Dev.	0.003321	0.004968	0.004054	0.00282	0.003255
C4	E_T	Mean	0.007695	0.0294	0.026101	0.019177	0.019719
		Std. Dev.	0.000638	0.001623	0.003031	0.003232	0.002443
	E_G	Mean	0.009819	0.030122	0.038808	0.022115	0.021954
		Std. Dev.	0.001473	0.00328	0.028168	0.002804	0.003473
D1	E_T	Mean	0.148513	0.056655	0.098873	0.127313	0.130179
		Std. Dev.	0.00199	0.00071	0.001553	0.002895	0.003944
	E_G	Mean	0.144373	0.219934	0.118008	0.14769	0.150337
		Std. Dev.	0.007129	0.005605	0.004376	0.007511	0.007289
D2	E_T	Mean	0.14725	0.056406	0.098709	0.121175	0.122189
		Std. Dev.	0.002158	0.00071	0.001347	0.004601	0.004256
	E_G	Mean	0.136828	0.216878	0.110697	0.135658	0.136576
		Std. Dev.	0.006315	0.006479	0.004437	0.006594	0.007123
D3	E_T	Mean	0.126362	0.041062	0.083174	0.082042	0.082775
		Std. Dev.	0.002513	0.000895	0.001437	0.004411	0.002679
	E_G	Mean	0.098811	0.138068	0.091993	0.088738	0.090142
		Std. Dev.	0.003342	0.004618	0.009105	0.00514	0.004242
D4	E_T	Mean	0.004087	0.017219	0.015818	0.00995	0.011061
		Std. Dev.	0.000332	0.001044	0.002082	0.001949	0.001863
	E_G	Mean	0.007678	0.019877	0.034009	0.013311	0.014333
		Std. Dev.	0.001513	0.003026	0.028019	0.002892	0.002704

Table 12 Dynamic Sphere Results for Scenarios A1 to D4

Scenario			BP	RBP	RPSO	CPSO	QPSO
A1	E_T	Mean	0.171713	0.131876	0.113942	0.119896	0.120057
		Std. Dev.	0.00311	0.002916	0.001718	0.002418	0.00155
	E_G	Mean	0.135771	0.199343	0.115899	0.121289	0.120964
		Std. Dev.	0.004825	0.005105	0.003382	0.00364	0.003453
A2	E_T	Mean	0.172275	0.130808	0.11798	0.133813	0.136196
		Std. Dev.	0.002906	0.003701	0.001374	0.005329	0.004216
	E_G	Mean	0.140283	0.198831	0.120919	0.13525	0.137025
		Std. Dev.	0.004329	0.005248	0.003699	0.004943	0.006435
A3	E_T	Mean	0.159435	0.122616	0.13249	0.176709	0.20885
		Std. Dev.	0.004758	0.008421	0.004636	0.004776	0.006293
	E_G	Mean	0.132337	0.161899	0.172753	0.179689	0.208684
		Std. Dev.	0.007327	0.00782	0.0356	0.004688	0.008191
A4	E_T	Mean	0.076418	0.093061	0.115917	0.137869	0.21962
		Std. Dev.	0.009425	0.010648	0.006892	0.009361	0.004297
	E_G	Mean	0.076587	0.08518	0.270092	0.142018	0.220084
		Std. Dev.	0.010715	0.01001	0.041104	0.010995	0.005145
B1	E_T	Mean	0.168007	0.105521	0.105274	0.109405	0.110216
		Std. Dev.	0.002584	0.000871	0.001257	0.001918	0.001549
	E_G	Mean	0.132217	0.182805	0.109222	0.115289	0.114269
		Std. Dev.	0.004245	0.00569	0.003762	0.00367	0.003857
B2	E_T	Mean	0.166557	0.105463	0.106646	0.11202	0.112497
		Std. Dev.	0.002542	0.001317	0.001065	0.001841	0.003126
	E_G	Mean	0.131353	0.181257	0.109564	0.114478	0.116895
		Std. Dev.	0.004144	0.005014	0.003063	0.004462	0.004536
B3	E_T	Mean	0.145928	0.090385	0.111102	0.140143	0.147884
		Std. Dev.	0.003437	0.002677	0.005848	0.00421	0.006497
	E_G	Mean	0.11083	0.142071	0.226889	0.140779	0.150163
		Std. Dev.	0.005273	0.004415	0.04717	0.005707	0.008159
B4	E_T	Mean	0.060173	0.056284	0.07264	0.196929	0.216473
		Std. Dev.	0.01075	0.004055	0.005138	0.013914	0.013488
	E_G	Mean	0.061217	0.054775	0.233869	0.197626	0.218016
		Std. Dev.	0.012189	0.004384	0.009101	0.014396	0.016347
C1	E_T	Mean	0.16711	0.099321	0.103326	0.11267	0.107417
		Std. Dev.	0.002519	0.000977	0.000945	0.026986	0.00243
	E_G	Mean	0.132319	0.177713	0.109139	0.116969	0.111008
		Std. Dev.	0.004788	0.005864	0.003655	0.024469	0.003637
C2	E_T	Mean	0.166076	0.098423	0.104653	0.110679	0.110652
		Std. Dev.	0.002259	0.000873	0.001422	0.003285	0.003651
	E_G	Mean	0.132456	0.177158	0.108582	0.11461	0.11474
		Std. Dev.	0.004116	0.005948	0.003781	0.003748	0.005502
C3	E_T	Mean	0.143071	0.082777	0.109036	0.1242	0.130536
		Std. Dev.	0.002946	0.002989	0.002281	0.003351	0.005925
	E_G	Mean	0.106784	0.136712	0.240324	0.125316	0.132597
		Std. Dev.	0.005094	0.005531	0.013597	0.005589	0.008417
C4	E_T	Mean	0.058047	0.045874	0.063607	0.200857	0.209843
		Std. Dev.	0.007239	0.002774	0.004789	0.015219	0.014711
	E_G	Mean	0.058775	0.046078	0.228473	0.200924	0.210977
		Std. Dev.	0.00914	0.004682	0.013786	0.014838	0.017679
D1	E_T	Mean	0.165768	0.091211	0.101189	0.244169	0.246038
		Std. Dev.	0.002174	0.000851	0.001321	0.029981	0.020945
	E_G	Mean	0.132933	0.173511	0.10672	0.248832	0.251531
		Std. Dev.	0.003737	0.006563	0.004144	0.031562	0.02237
D2	E_T	Mean	0.164773	0.090866	0.101856	0.242995	0.247662
		Std. Dev.	0.002493	0.001116	0.001426	0.036794	0.026591
	E_G	Mean	0.13038	0.172943	0.107513	0.246834	0.251565
		Std. Dev.	0.005265	0.005622	0.004631	0.037879	0.027257
D3	E_T	Mean	0.140808	0.073325	0.102763	0.236357	0.25326
		Std. Dev.	0.002707	0.001552	0.001444	0.041818	0.001802
	E_G	Mean	0.105287	0.129212	0.240959	0.238866	0.254952
		Std. Dev.	0.004103	0.004556	0.008675	0.04002	0.006693
D4	E_T	Mean	0.054159	0.034517	0.056596	0.240363	0.238631
		Std. Dev.	0.007395	0.00331	0.008	0.006358	0.019632
	E_G	Mean	0.056111	0.035871	0.226403	0.241697	0.240392
		Std. Dev.	0.008284	0.004875	0.023402	0.007598	0.019668

Table 13 Sliding Thresholds Results for Scenarios A1 to D4

Scenario			BP	RBP	RPSO	CPSO	QPSO
A1	E_T	Mean	0.153493	0.103516	0.070907	0.072955	0.071743
		Std. Dev.	0.015032	0.000502	0.000673	0.004674	0.003221
	E_G	Mean	0.136737	0.147665	0.071907	0.073412	0.073101
		Std. Dev.	0.014298	0.003263	0.002433	0.005112	0.00487
A2	E_T	Mean	0.142729	0.102635	0.074074	0.074585	0.074158
		Std. Dev.	0.009063	0.00077	0.000665	0.003901	0.003075
	E_G	Mean	0.125513	0.147616	0.074277	0.075893	0.074692
		Std. Dev.	0.011567	0.004737	0.003028	0.004829	0.00366
A3	E_T	Mean	0.114955	0.09379	0.082315	0.072066	0.071948
		Std. Dev.	0.009776	0.001615	0.002789	0.002915	0.00227
	E_G	Mean	0.098212	0.12228	0.084629	0.075276	0.075056
		Std. Dev.	0.012554	0.004462	0.005339	0.00405	0.003118
A4	E_T	Mean	0.049715	0.080386	0.066253	0.058537	0.057151
		Std. Dev.	0.004035	0.002757	0.004782	0.005081	0.005216
	E_G	Mean	0.045486	0.080073	0.096665	0.063731	0.063549
		Std. Dev.	0.005378	0.005149	0.024581	0.007158	0.007882
B1	E_T	Mean	0.151092	0.094176	0.061455	0.072534	0.070514
		Std. Dev.	0.020754	0.00101	0.001925	0.005605	0.004996
	E_G	Mean	0.13528	0.139925	0.062192	0.073516	0.071295
		Std. Dev.	0.017375	0.003195	0.002846	0.006696	0.005128
B2	E_T	Mean	0.149287	0.093804	0.06368	0.071221	0.068427
		Std. Dev.	0.017802	0.001313	0.001258	0.008886	0.004953
	E_G	Mean	0.132267	0.139525	0.065144	0.072218	0.069381
		Std. Dev.	0.014432	0.004149	0.002031	0.00837	0.00564
B3	E_T	Mean	0.132411	0.083283	0.058312	0.059298	0.058643
		Std. Dev.	0.011119	0.003031	0.00091	0.003277	0.00301
	E_G	Mean	0.115916	0.11243	0.058575	0.059405	0.060084
		Std. Dev.	0.010667	0.004419	0.002254	0.00457	0.003505
B4	E_T	Mean	0.085928	0.066596	0.024424	0.024341	0.022029
		Std. Dev.	0.029466	0.005977	0.001652	0.020914	0.004917
	E_G	Mean	0.08665	0.064456	0.024827	0.025131	0.022903
		Std. Dev.	0.031776	0.007366	0.003369	0.020715	0.005061
C1	E_T	Mean	0.200552	0.092587	0.059751	0.076702	0.077463
		Std. Dev.	0.11802	0.00123	0.00178	0.004779	0.005842
	E_G	Mean	0.182108	0.136819	0.059944	0.078222	0.079614
		Std. Dev.	0.112807	0.003454	0.002863	0.005915	0.006852
C2	E_T	Mean	0.158981	0.091914	0.061345	0.071107	0.072182
		Std. Dev.	0.038679	0.001601	0.002197	0.006099	0.006226
	E_G	Mean	0.171066	0.137699	0.061962	0.072751	0.073344
		Std. Dev.	0.107484	0.003701	0.003968	0.00658	0.00678
C3	E_T	Mean	0.138621	0.082158	0.054092	0.056934	0.056913
		Std. Dev.	0.017921	0.004326	0.000935	0.003381	0.004078
	E_G	Mean	0.120052	0.111868	0.055278	0.057185	0.058309
		Std. Dev.	0.016573	0.005933	0.002278	0.003319	0.004474
C4	E_T	Mean	0.08482	0.061294	0.018883	0.018289	0.018509
		Std. Dev.	0.026541	0.00619	0.002162	0.006454	0.004962
	E_G	Mean	0.085047	0.061548	0.019303	0.018742	0.018866
		Std. Dev.	0.026804	0.008117	0.003366	0.007034	0.005452
D1	E_T	Mean	0.190979	0.089317	0.056782	0.089642	0.086245
		Std. Dev.	0.116883	0.001316	0.001141	0.009767	0.005966
	E_G	Mean	0.174311	0.134207	0.057856	0.093216	0.087575
		Std. Dev.	0.114016	0.003126	0.002018	0.011032	0.006176
D2	E_T	Mean	0.16679	0.088831	0.058099	0.076492	0.078824
		Std. Dev.	0.082548	0.001458	0.00136	0.005072	0.00572
	E_G	Mean	0.150545	0.133354	0.058946	0.078749	0.08096
		Std. Dev.	0.081005	0.004312	0.002143	0.005938	0.005934
D3	E_T	Mean	0.161774	0.078363	0.049395	0.059674	0.058129
		Std. Dev.	0.044374	0.004173	0.000909	0.007093	0.006498
	E_G	Mean	0.144836	0.107945	0.04987	0.061183	0.059113
		Std. Dev.	0.043317	0.005508	0.001521	0.007908	0.007565
D4	E_T	Mean	0.118543	0.055971	0.012676	0.017977	0.016372
		Std. Dev.	0.070682	0.006565	0.000938	0.009832	0.00934
	E_G	Mean	0.120698	0.055365	0.013223	0.018939	0.016812
		Std. Dev.	0.071051	0.008704	0.001815	0.009866	0.009346

Table 14 Electricity Pricing Results for Scenarios A1 to D4

Scenario			BP	RBP	RPSO	CPSO	QPSO
A1	E_T	Mean	0.107272	0.130724	0.1123	0.103669	0.101964
		Std. Dev.	0.001239	0.000274	0.000892	0.005592	0.00475
	E_G	Mean	0.107447	0.148172	0.112489	0.103975	0.102248
		Std. Dev.	0.001796	0.001327	0.002648	0.005727	0.005023
A2	E_T	Mean	0.110246	0.132773	0.11512	0.112274	0.111304
		Std. Dev.	0.001341	0.000297	0.001249	0.002701	0.00308
	E_G	Mean	0.109761	0.144535	0.114578	0.112519	0.111775
		Std. Dev.	0.002554	0.00125	0.001907	0.003538	0.003462
A3	E_T	Mean	0.115424	0.135959	0.121239	0.120312	0.11953
		Std. Dev.	0.001046	0.000425	0.001295	0.001869	0.001611
	E_G	Mean	0.115071	0.136993	0.121328	0.122126	0.120109
		Std. Dev.	0.001993	0.001513	0.002863	0.00201	0.002337
A4	E_T	Mean	0.11685	0.137511	0.123221	0.123785	0.122252
		Std. Dev.	0.001113	0.000552	0.001571	0.001418	0.001398
	E_G	Mean	0.116022	0.136005	0.12502	0.126081	0.124756
		Std. Dev.	0.003052	0.001683	0.002156	0.002805	0.002274
B1	E_T	Mean	0.102724	0.111686	0.090001	0.086432	0.085533
		Std. Dev.	0.00105	0.000244	0.001818	0.001352	0.001075
	E_G	Mean	0.104494	0.13992	0.091167	0.087266	0.08741
		Std. Dev.	0.002749	0.001823	0.00302	0.003113	0.002577
B2	E_T	Mean	0.104079	0.114145	0.097767	0.090506	0.090119
		Std. Dev.	0.001168	0.000307	0.003733	0.001881	0.001454
	E_G	Mean	0.105228	0.133539	0.098844	0.091683	0.09106
		Std. Dev.	0.002769	0.002286	0.003975	0.003021	0.001837
B3	E_T	Mean	0.106364	0.118998	0.110811	0.110661	0.1091
		Std. Dev.	0.001252	0.000255	0.001183	0.002022	0.002018
	E_G	Mean	0.10602	0.121973	0.110738	0.110967	0.110334
		Std. Dev.	0.001793	0.002099	0.002312	0.002817	0.003032
B4	E_T	Mean	0.104404	0.119889	0.111163	0.112023	0.1101
		Std. Dev.	0.001763	0.000466	0.001441	0.001367	0.000844
	E_G	Mean	0.105596	0.120727	0.112698	0.114921	0.113525
		Std. Dev.	0.003542	0.002529	0.002586	0.002237	0.002294
C1	E_T	Mean	0.100309	0.107059	0.085203	0.083106	0.082736
		Std. Dev.	0.002084	0.00025	0.001031	0.001323	0.001698
	E_G	Mean	0.102539	0.136312	0.086372	0.084711	0.085369
		Std. Dev.	0.00303	0.002022	0.002087	0.002284	0.002352
C2	E_T	Mean	0.101966	0.109329	0.088946	0.086329	0.08598
		Std. Dev.	0.001622	0.00024	0.001818	0.001865	0.001284
	E_G	Mean	0.104083	0.13032	0.090749	0.087324	0.0881
		Std. Dev.	0.003152	0.002253	0.002938	0.002207	0.002133
C3	E_T	Mean	0.103866	0.113369	0.107534	0.10435	0.102471
		Std. Dev.	0.001315	0.000367	0.001977	0.004578	0.004847
	E_G	Mean	0.104704	0.118371	0.108407	0.105365	0.103366
		Std. Dev.	0.002472	0.002558	0.003185	0.005218	0.005688
C4	E_T	Mean	0.100393	0.114362	0.107722	0.108165	0.1067
		Std. Dev.	0.001628	0.000416	0.001031	0.001332	0.001677
	E_G	Mean	0.104217	0.116367	0.110712	0.111114	0.109474
		Std. Dev.	0.002812	0.002244	0.002297	0.002527	0.002684
D1	E_T	Mean	0.094779	0.100251	0.081339	0.080573	0.08026
		Std. Dev.	0.003527	0.000249	0.001314	0.002001	0.001216
	E_G	Mean	0.097618	0.127633	0.084291	0.083854	0.083442
		Std. Dev.	0.00474	0.001818	0.002058	0.002978	0.002134
D2	E_T	Mean	0.099172	0.102321	0.083684	0.082523	0.081744
		Std. Dev.	0.002479	0.000283	0.001233	0.001528	0.001526
	E_G	Mean	0.100719	0.122955	0.085278	0.084428	0.084446
		Std. Dev.	0.002892	0.002114	0.00214	0.002947	0.002468
D3	E_T	Mean	0.10169	0.105676	0.095168	0.090168	0.088658
		Std. Dev.	0.001228	0.000334	0.004108	0.003092	0.001317
	E_G	Mean	0.102847	0.111569	0.095952	0.09121	0.089816
		Std. Dev.	0.002181	0.002374	0.005246	0.003668	0.003046
D4	E_T	Mean	0.099187	0.106452	0.102084	0.098671	0.098989
		Std. Dev.	0.001916	0.000427	0.001793	0.004797	0.003366
	E_G	Mean	0.102562	0.109373	0.105275	0.101999	0.102871
		Std. Dev.	0.003234	0.002192	0.003199	0.006492	0.004776

Table 15 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**	0.2794	**0.0001**	**0.0020**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0162**	**0.0007**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	0.2927	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.6653	**0.0001**
D	0.4761	0.2300	0.2601	0.4231	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.0654	**0.0110**
BP vs CPSO						BP vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	**0.0025**	0.9593	**0.0001**	**0.0005**	**0.0001**	**0.0001**	0.06113	**0.0001**	**0.0012**	**0.0001**
B	**0.0001**	**0.0001**	0.0571	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0207**	**0.0003**	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	0.9360	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.5134	**0.0001**
D	**0.0001**	**0.0001**	**0.0001**	0.7528	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0462**	**0.0076**
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**	**0.0021**
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**	0.1091	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.2601	**0.0224**
D	**0.0001**	**0.0001**	**0.0001**	0.8434	**0.0001**	**0.0001**	**0.0001**	**0.0027**	0.1124	**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**	**0.0002**	**0.0080**	**0.0332**	0.2728	0.1462	**0.0183**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0142**	**0.0274**	0.4231	0.0878	0.1774
C	**0.0001**	**0.0001**	**0.0001**	0.0995	**0.0001**	0.2539	0.0850	0.0723	0.6440	0.0772
D	**0.0001**	**0.0001**	**0.0001**	0.8148	**0.0001**	**0.0263**	0.1462	**0.0002**	**0.0162**	0.1973
RPSO vs QPSO						CPSO vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	0.4404	0.2728	0.1973	0.5134	0.3353	**0.0005**	**0.0002**	0.7865	0.2996	0.0699
B	0.9709	0.9826	0.8664	0.5326	0.5720	**0.0332**	0.06994	0.8091	0.2132	0.0677
C	0.9360	0.0688	0.8549	0.6283	0.1973	0.3353	0.9709	**0.0479**	0.4492	**0.0084**
D	**0.0187**	**0.0234**	**0.0371**	0.9942	0.7306	0.7416	0.5040	**0.0263**	**0.0184**	0.1266

Table 16 Mann-Whitney U p-values obtained for the average generalisation 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**	0.1230	0.0514	**0.0003**	**0.0022**	**0.0001**
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0371**	0.4492	0.6024	0.1124	**0.0001**
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.1727	0.0935	0.6335	0.1727	1.0058	0.4146
D	**0.0001**	**0.0001**	**0.0001**	0.0906	**0.0136**	**0.0308**	**0.0400**	0.5720	0.0699	0.1194
BP vs CPSO						BP vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	0.0677	0.6024	**0.0032**	**0.0006**	**0.0001**	0.0514	0.2132	**0.0120**	**0.0002**	**0.0001**
B	**0.0034**	0.5134	0.2996	0.1547	**0.0002**	**0.0225**	0.6440	0.3428	0.6760	**0.0001**
C	0.1973	0.9826	0.2418	0.7640	0.1194	0.8664	0.6024	0.2478	0.4853	0.9942
D	**0.0084**	0.1026	0.3353	0.0935	**0.0155**	0.1091	**0.0319**	**0.0225**	0.2358	**0.0080**
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**	0.1026
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**	**0.0001**	0.5820	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.5720
D	**0.0001**	**0.0001**	**0.0001**	**0.0002**	0.3737	**0.0001**	**0.0001**	**0.0001**	**0.0003**	0.8320
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**	**0.0225**	0.6653	0.2078	0.1871	0.6440	0.0611
B	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.3136	0.9360	0.1727	0.8780	0.9476
C	**0.0001**	**0.0001**	**0.0001**	**0.0001**	0.3580	0.8664	0.7416	0.7528	0.7085	0.3428
D	**0.0001**	**0.0001**	**0.0001**	**0.0063**	0.8091	0.8664	0.7195	0.1194	0.9244	0.5620
RPSO vs QPSO						CPSO vs QPSO				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
A	0.6024	0.4062	**0.0479**	0.5820	0.3898	0.9593	0.4146	0.5040	0.2300	0.1401
B	0.8434	0.7978	0.5922	0.2794	0.8320	0.3580	0.8205	0.0850	0.5230	0.6546
C	0.1547	0.9593	0.8549	0.4063	0.6127	0.1680	0.7306	0.9011	0.5922	0.1547
D	0.6546	0.8780	**0.0020**	0.5423	0.6024	0.4492	0.6546	0.1973	0.5620	1.0058

Table 17 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 18 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 19 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 20 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 21 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 22 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

Table 23 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 24 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