*Neural, Parallel & Scientific Computations* 15 (2007) 137-164

# Recursive Percentage based Hybrid Pattern Training for Supervised Learning

Kiruthika Ramanthan[1] and Sheng Uei Guan[2]
[1]Data Storage Institute, Singapore 117608
[2]School of Engineering and Design, Brunel University.

**Abstract**
Supervised learning algorithms, often used to find the I/O relationship in data, have the tendency to be trapped in local optima as opposed to the desirable global optima. In this paper, we discuss the Recursive Percentage based Hybrid Pattern (RPHP) learning algorithm. The algorithm uses Real Coded Genetic Algorithm based global and local searches to find a set of pseudo global optimal solutions. Each pseudo global optimum is a local optimal solution from the point of view of all the patterns but globally optimal from the point of view of a subset of patterns. Together with RPHP, a Kth nearest neighbor algorithm is used as a second level pattern distributor to solve a test pattern. We also show theoretically the condition under which finding several pseudo global optimal solutions requires a shorter training time than finding a single global optimal solution. As the difficulty of curve fitting problems is easily estimated, we verify the capability of the RPHP algorithm against them and compare the RPHP algorithm with three counterparts to show the benefits of hybrid learning and active recursive subset selection. The RPHP shows a clear superiority in performance. We conclude our paper by identifying possible loopholes in the RPHP algorithm and proposing possible solutions.

**Keywords** - evolutionary algorithms, task decomposition, hybrid learning, pattern learning, subset finding, data oriented training

## 1. INTRODUCTION

Imagine a situation where a teacher is to teach a group of students (the population) a set of problems (the tasks). In a normal classroom, the teacher would repeatedly teach the various problems until either the teacher is satisfied with the results or the students are unable to learn any more tasks. Usually, the aptitude of the students is limited and without a very good teacher, it is unlikely that they will learn all the tasks. On the other hand, although very good teachers are hard to find, it is necessary that the class learn all the problems.

Therefore, a new solution is proposed. The teacher now teaches all the problems until some students (Group A) in the class learn some of the problems (Set A) (The students do not have to learn these problems perfectly; they can make some errors during the learning). Now the teacher isolates these Group A students and allows them to learn set A problems alone.

The teacher now blanks the memory of the remaining students and focuses their attention on the remaining problems. As the students' memories are reinitialized, their previous lack of knowledge will not come into account when dealing with the remaining problems. The teacher teaches the remaining problems until a group of students (Group

Received June 12, 2006
1061-5369 $15.00 © Dynamic Publishers, Inc.

B) learns some of them (Set B). The teacher now isolates Group B and lets them learn set B problems until the problems are perfected.

This process is repeated until there are too few problems left to allow further decomposition. The class is then set to learn these remaining problems in the best way possible. The teacher therefore induces a team effort, such that, as a team, the class is able to solve the problems better than an individual student would. She isolates students showing aptitude for a set of problems by allowing them to focus their attention on those problems in particular and not worry about the problems that they find difficult. These problems will still be learnt... there are other students who will show aptitude for these problems.

The teacher's job is therefore simplified. The students' job is simplified as well, since they only have to learn those problems they find easy and can therefore solve them faster and better.

This is the concept of RPHP algorithm. The pool of solutions (population), along with the teacher, uses a genetic algorithm based global search to decompose the datasets recursively. Each data subset is considered simple by the solution S that has learnt it to certain extent. This solution S now concentrates on specializing on the decomposed dataset and learns it perfectly using a local training technique (based on genetic algorithms or neural networks). The set of solutions S obtained can therefore solve any problem in the domain as long as we know which subset of problems the new problem belongs to. In RPHP, this subset identification is performed using the *K*th nearest neighbor algorithm. We also make use of extensive validation and early stopping techniques to ensure that overtraining is avoided.

The RPHP algorithm displays the following salient properties

1. Pattern training and error training are focused on, as opposed to error training only. Error training alone has the disadvantage of over-training of some patterns while other patterns can be left untrained or under-trained.
2. Since all patterns receive equal attention in training, there is a higher possibility of obtaining better training, as well as generalization accuracy.
3. Since the RPHP algorithm is also clustering based, adding new training patterns after training is complete will simply result in the development of new clusters to deal with the new patterns, instead of complete retraining.
4. As progressively fewer samples are learnt in each recursion, the training time required for each epoch is reduced.
5. As the difficulty of the training patterns increases progressively with each recursion (from the point of view of the students), the population focuses more on the difficult samples
6. The recursions stop when the number of samples reduces to a small amount, avoiding the likelihood of overtraining.
7. The combination of global and local training reduces the possibility of the KNN pattern distributor error. This is done by explicitly learning each pattern by one sub-solution and by implicitly learning it by one or more solutions, creating some cushion against the error of the pattern distributor This property is further discussed in section 2.5
8. The decomposition algorithm takes the problem structure into account, while being problem independent at the same time. Therefore, this process is more natural than other data decomposition techniques described below.
9. The RPHP algorithm also reduces dependence on several training parameters often introduced in other hybrid algorithms and subset finding algorithms. These include

the subset size, the degree of error tolerance $\varsigma$ and the number of epochs to be trained before the training mode can change from global to local.

### 1.1. Related Literature Review

The RPHP algorithm can be summarized as an efficient combination of three types of weak learning algorithms – hybrid learning, task decomposition, and pattern distribution. In this section, we will briefly review related work in these three fields and discuss their respective strengths and weaknesses.

#### 1.1.1. Hybrid learning

Global search algorithms such as genetic algorithms (Goldberg, 1989) and genetic programming (Riolo and Worzel, 2003) have been used in conjunction with local search techniques such as backpropagation (Rumelhart et al., 1986), dynamic backpropagation (Ash, 1987; Marchesi et al., 1990) and decision tree algorithms (Quinlan, 1996; Quinlan, 1986; Breiman, 1994). Combinations of global and local search algorithms have been proposed and tested successfully in the literature. Carvalho and Freitas (2004) proposed and verified that a hybrid GA/ decision tree algorithm performed better than the local training algorithm for 16 out of 21 classification problems. In another paper, Andreas et al. (2002) proposed the use of a genetic algorithm to optimize adaptive neural networks for exchange rates forecasting. Other applications of genetic algorithms to improve neural networks include friction compensation (Chaitanya and Zalzala, 2000), function approximation (Rovithakis, 2004), antennas and propagation (Devi et al., 2003), image segmentation (Dokur, 2002) and classification (Pendharkar and Rodger, 1999) . However, besides neural networks and decision trees, the global search ability of genetic algorithms have been widely applied to supplement training algorithms such as Particle Swarm Optimization (Juang, 2004) and simulated annealing (Aydin and Fogarty, 2004; Mahfoud and Goldberg, 1992)). A comprehensive, if old, survey of hybrid neural network and evolutionary algorithms can be found in Yao (1993). The research in hybrid GA neural networks however, is still hot, with much recent work in the field (Cantu-Paz and Kamnath, 2005; Illonen et al., 2003).

Since genetic algorithms can be oriented towards both global and local search depending on its tuning parameters, hybrid combinations of global and local search in genetic algorithms have been proposed in various papers, beginning with the Lamarckian evolution (Ackley and Littmann, 1993). A recent work by Vasconcelos et al. (2001) proposed the Linear Interpolation algorithm, which automatically adjusted the degree of global and local search in a GA based optimization problem. In order to simplify the theory of the algorithm proposed, we will focus, in this paper, specifically on hybrids of global and local search using real coded genetic algorithms.

While all the above papers concluded that hybrid algorithms are generally better as they appear to combine the breadth of global search with the hill climbing of a local search, the reason why hybrid algorithms work is still predominantly a black box. A notable question that arises is when to change from global to local search and vice versa. A number of algorithms, including (Dokur, 2002; Pendharkar, 1999, Yasunaga et al., 1999) use the number of generations of global search as a criterion for switching between global and local search.

In Linear Interpolation (Vasconcelos et al., 2001), however, the genetic diversity of the population is used to determine the probability of global search and the probability of local search. While these switchover methods suffice for optimization applications, the performance of a supervised learning algorithm depends on the patterns learnt. In an

earlier work (Guan and Ramanathan, 2005), we showed empirically that a good point of switchover between global and local search would be when approximately 50% of the patterns are learnt by the system. We also showed, by experiments, that using global search to learn 50% of the patterns increases the probability of finding a global optimal neighborhood. The RPHP algorithm will use this property that was observed in (Guan and Ramanathan, 2005).

### 1.1.2. Task decomposition

Many papers have been written on the possibility of using a subset of training patterns for training instead of the whole dataset. A notable work by Foody (1998) argued that the classifier structure can be determined by the border patterns (i.e, those whose Mahalanobhis distances are close to patterns of other classes), while the core patterns can be discarded. Another paper by Gathercole et al. (1994) assigns a difficulty value to a pattern based on how often the best performing chromosome learns it and then using this probability to select a subset. The topology based dynamic selection (Lasarcyck et al., 2004) again selects subsets of training patterns based on their difficulty. The difficulty of the pattern is determined by whether a pattern can be learnt with an accuracy of $\xi$. More and more "difficult" patterns are chosen until a desired subset size is reached.

The theory behind these approaches is that when training emphasis is given to the difficult patterns, it is possible to obtain an accurate classifier. However, neither method outlines a method for finding the size of a subset or the preferred learning accuracy, $\xi$. Moreover, it is likely that information (and generalization capability) can be lost when the core or "easy" patterns are ignored when training.

Several methods have also been proposed to emphasize on learning difficult patterns while still learning the easy patterns. Techniques include incremental learning with active data selection (Zhang and Cho, 1998) and repeated learning of difficult patterns (Stoianov et al., 2001).

While the above algorithms in the literature were shown to be effective, they still try to increase the probability of finding one good solution. On the other hand, it is possible to split the training set into several smaller subsets, find a solution for each subset, and put them all together. This way, although emphasis is given to difficult patterns, the easy patterns are not ignored. Many recent algorithms implement this subset selection by decomposing the data manually according to class labels (Guan et al., 2003; Fu et al., 2001). The assumption is that a two-class problem is easier to solve than a K class problem. Therefore a K-class problem is divided into K two-class problems and each sub problem is solved by using a separate classifier.

There are two fundamental problems with this approach. Firstly, it is not necessarily true that a two-class problem is simpler than a K-class problem. For example, the two spiral problem (2 classes), is inarguably more complicated than the Iris (3-class) problem. Therefore, this approach is not foolproof. Secondly, the separation of data is manual and based on class labels. Therefore the algorithm itself can only be applied to classification problems.

The multisieving algorithm (Lu et al., 1995) aims to combat these issues. In the multisieving algorithm, the neural network is trained using all the available data until stagnation occurs. At that point, all the patterns which produce valid outputs, i.e., $|D_j - O_j| < \xi$, where $\xi$ is an error tolerance, are considered learnt and therefore isolated along with their corresponding network. The remaining (unlearnt) patterns are

further trained using another network and the process is repeated until all the patterns are learnt. Using the two spiral problem, the authors showed the validity of their approach.

Although the multi-sieving approach combats the problems of class based decomposition, the following issues remain unsolved: Firstly, the algorithm is not completely adapted to the problem topology since its performance depends on the error tolerance $\xi$, which is a predefined value. A low value of $\xi$ could lead to overtraining while too large an $\xi$ possibly results in patterns not being learnt near optimal. The parameter therefore has to be set to an accurate value.

Furthermore, the algorithm described did not provide for generalization but only for 100% training accuracy to the limit of the error tolerance. Moreover, there is no documentation on how to best deal with a test pattern, other than introduce it into the various modules until one of the modules produces an output of 1. Further, the generalization accuracy of the algorithm was not reported. Our survey showed that, while the multisieving algorithm has been widely cited in the literature, no follow-up work has been done to improve the algorithm.

### 1.1.3. Pattern distribution

When data is decomposed into several subsets as discussed in section 1.1.2, and each subset trained and represented by a separate network, there appears the problem of system generalization accuracy. How do we best assign an unseen pattern into one of the K subsets? In Lu et al. (1995) the authors proposed that a pattern be presented to each network in turn until one of the networks produces an output of 1. However, this procedure produces a bias on the result of the first network. A pattern distributor is therefore introduced (Guan et al., 2003) to solve this problem. The input patterns and the index of the subset are used to train another network which acts as a second layer classifier. This classifier classifies the patterns into subsets, which in turn provide the solution to the problem. However, the pattern distributor, in itself, produces a classification error which must be taken into account.

The RPHP algorithm can be generally overviewed as a robust generalization of hybrid genetic algorithms, topology based subset finding, multisieving and pattern distribution. As we proceed with this paper, we shall explain the structure of the RPHP algorithm and how it overcomes the disadvantages of the various algorithms outlined above. The rest of the paper is organized as follows: Section 2 presents related theory and introduces the RPHP algorithm. In section 3, we present theoretical analysis of the effectiveness of the RPHP algorithm. In section 4, we describe the problems considered and present the simulation results. While the RPHP algorithm improves the training results, it has two associated problems. These are discussed in Section 5. Section 5 also suggests the RPHP2 algorithm, which solves these problems. Section 6 presents the conclusions of our research and the future work we wish to undertake with regards to the RPHP algorithm.

## 2. RPHP ALGORITHM AND RELATED THEORY

### 2.1. The RPHP algorithm overview

The RPHP training algorithm is a supervised learning algorithm designed to improve training time and generalization accuracy when finding the global optimal solution of a sufficiently complex surface. The algorithm begins with fitting a solution to all the training patterns.

### 2.1.1. Training

1. As we are only looking for a partial solution fast, we use genetic algorithms to perform the global search across the solution space with all the available training patterns.
2. We continue training until one of the following two conditions are satisfied: a). There is stagnation or b) 50% of the patterns are learnt. The condition of 50% patterns learnt has been imposed based on our earlier work (Guan and Ramanathan, 2005). The reason for implementing this condition is explained in section 2.5.
3. At this stage, we use a condition similar to that in Lasarzyck (2004) and the multisieving network (Lu et al., 1995) to identify learnt patterns, i.e., a pattern is considered learnt if $|D_j - O_j| < \xi$. More formally, we can define the percentage of total patterns learnt as

$$L_g = \frac{1}{T}\sum_{i=0}^{T}\delta\left\{\left[\frac{1}{O}\sum_{j=0}^{O}\phi\left[\xi - |D_{i,j} - O_{i,j}|\right]\right] - 1\right\} \quad (1)$$

   In the above equation, $\xi$ is the predefined error tolerance and $\delta(x)$ is the unit impulse function. $\phi(x)$ is the unit step function. Note that although, similar to the multisieving algorithm, a tolerance $\xi$ is used to identify learnt patterns, the arbitrarily set value of $\xi$ for RPHP does not affect the performance of the algorithm, as explained later.
4. The dataset is now split into learnt and unlearnt patterns. With the unlearnt patterns, we repeat steps 1 to 3.
5. Since the learnt patterns are only learnt up to a tolerance $\xi$, we use a local search oriented genetic algorithm to find the nearest optima for the learnt patterns. This optima is called the ***pseudo-global optimal solution,*** and is found using a validation set of data to prevent over training and to overcome the dependence of the algorithm on $\xi$. The concept of *pseudo global optima* is explained in the next section. The local search oriented genetic algorithm is used because of illustration simplicity and can easily be replaced by any hill climbing technique.

As the number of patterns in a data subset is small, especially as the number of recursions increases, it is possible for the pseudo global optimal solution to overfit the data in the subset. In order to avoid this possibility, we use a validation dataset. The validation dataset is used along with the training data to detect generalization loss using an algorithm in (Guan and Li, 2002). The algorithm details are given in the appendix.

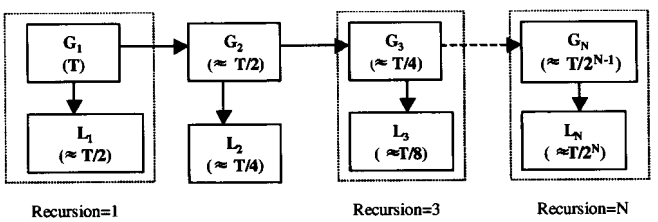


**Figure 1. Recursive Data Decomposition employed by RPHP**

The data decomposition technique of the RPHP algorithm can be best described by Figure 1. During the first recursion, the entire training set (size *T*) is learnt using global training until *T/2* patterns are learnt or until stagnation occurs. Only the learnt patterns are then further learnt locally, with measures to prevent overtraining. This ensures the finding of a *pseudo global optimal solution.* The second recursion repeats the same procedure with the unlearnt patterns. The process repeats until the total number of patterns in a given recursion (Recursion *N*) is too small, in which case, local training is applied to the whole dataset to learn the remaining patterns to the best possible extent. Figure 2 gives the detailed pseudocode of the RPHP algorithm

```
Train (i)
{       Globally train the dataset Gi using a new set of chromosomes
        IF approximately 50% of the patterns are learnt OR stagnation occurs
        {       1. Identify the learnt patterns
                2. Split Gi into Li (consisting of the learnt patterns) and Gi+1 (consisting of the unlearnt patterns)
                3. Locally train the existing solutions using data set Li. The procedure is validated using dataset V1
                4. IF local training is complete (stagnation OR generalization loss)
                        IF Gi has too few patterns
                        {       a. Copy Gi to Li
                                b. Locally train Li until Generalization loss OR stagnation
                                c. STORE solution i
                                d. END Training
                        }
                        ELSE
                        {       FREEZE solution i
                                Train (i+1)
                        }}}
```

**Figure 2. Detailed pseudocode of the RPHP1 algorithm**

### 2.1.2. Testing

Testing in the RPHP algorithm is implemented using a Kth nearest neighbor (Wong and Lane, 1983) based pattern distributor. KNN was used to implement the pattern distributor because of the ease in its implementation. At the end of the RPHP training phase, we have *N* subsets of data. A given test pattern is matched with its nearest neighbor. If the neighbor belongs to subset *i*, the pattern is also deemed as belonging to subset *i* the solution for subset *i* is used to find the output of the pattern. This process is illustrated by Figure 3.

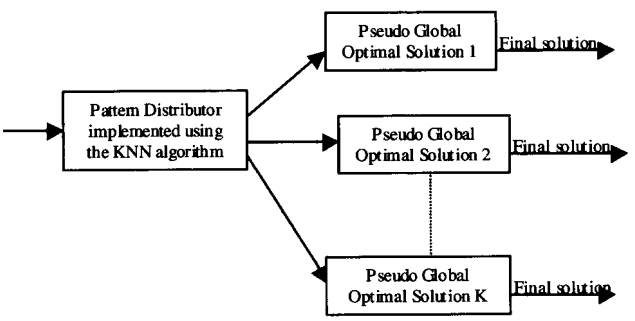


**Figure 3. The two level RPHP problem solver**

### 2.2. Terms and Definitions

In order to demonstrate effectively the theoretical implications and effectiveness of the RPHP algorithm, we use the simplest forms of the various algorithms proposed in the literature. As many new algorithms proposed black boxes, we decided to use fundamental

principles and algorithms to justify our techniques. Simulations on the two spiral dataset were carried out to demonstrate that the RPHP algorithm can also be built on top of newer algorithms. In this section we will define the terminology used in this paper.

### 2.2.1. Global and local training

We define global and local search based on the fact that global search searches for fitter individuals through the whole search space while local search searches within a given neighborhood.

Global training is therefore implemented with single point crossover and mutation with a large random change in one or more elements of the chromosome. Local training is implemented by a mutation only genetic algorithm. The chromosome is changed by a small random $\delta$, where $\delta$ is a normally distributed variable.

### 2.2.2. Global optimal neighbourhood

Figure 4 illustrates the idea of a global optimal neighborhood. It is defined as the largest continuous segment of the error against x curve where the value of error is monotonically increasing for $x_g+$ or $x_g-$. The neighborhood is bounded by $x_{gmax}$ and $x_{gmin}$.

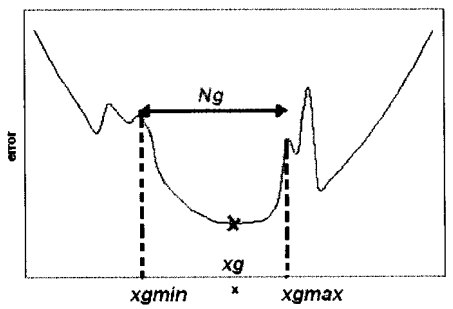


**Figure 4. A global optimal neighborhood**

### 2.2.4. Chromosome encoding

Since this paper aims to cast some theoretical insight to the behavior and advantages of RPHP, it was necessary to find the best possible experimental algorithms to simulate the theories. The problems considered in this paper are curve fitting problems, i.e., problems whose I/O data and the general form of the relationship function are known, but the actual coefficients are unknown. The elements in the real coded genetic algorithm represent one of the several unknown coefficients.

For generalization and comparison purposes, we have also included simulation results from the two spiral classification problem. In that example, a messy genetic algorithm (Goldberg et al., 1991) is used to evolve both the topology and the weights of the neural network simultaneously. This represented the global search phase. The local search was carried out by performing back propagation on the best chromosome. The chromosome thus encodes the weights and architecture of the neural network.

## 2.3. Pseudo global optima

The better performance of the RPHP algorithm can be attributed to the fact that RPHP aims to find several pesudo-global solutions as opposed to a single global solution. We define a pseudo-global optimal solution as follows:

***Definition 1:*** *A pseudo-global optima is a global optima when viewed from the perspective of a subset of training patterns, but could be a local (or global) optima when viewed from the perspective of all the training patterns.*

To illustrate the connection between pseudo global optima and the RPHP algorithm, we present the argument below:

Consider the use of the RPHP algorithm to model a function $\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{w})$, where **x** and **y** are the feature and output values respectively and **f** is a vector function of a vector variable **x**. **w** is the set of values to be optimized. Given $N$ training patterns, the training error at any point of time is given by

$$E = \sum_{i=1}^{N} \left\| \mathbf{y}_i - \hat{\mathbf{f}}_i(\mathbf{x}_i) \right\| = N_{learnt}\varepsilon_{learnt} + N_{unlearnt}\varepsilon_{unlearnt} = E_{learnt} + E_{unlearnt} \tag{2}$$

where $\hat{\mathbf{f}}_i(\mathbf{x}_i)$ is an approximation to $\mathbf{f}(\mathbf{x})$. Further, we know that at any given point, E can be split into $E_{learnt}$ and $E_{unlearnt}$, which represents the errors due to the learnt and unlearnt patterns respectively. By definition of learnt and unlearnt patterns $\varepsilon_{learnt} \leq \xi < \varepsilon_{unlearnt}$, where $\xi$ is the error tolerance Also, as we approach the optimal points, $E_{learnt} \to 0$.

Also, consider that at the end of global training, all the learnt patterns have an error less than the error tolerance $\xi$, i.e.

$$E = E_{learnt} + E_{unlearnt} < \xi N_{learnt} + E_{unlearnt} \tag{3}$$

RPHP splits up the training patterns after global training of recursion k ($G_k$) such that the local training $L_k$ of recursion k is carried out with $N_{learnt}$ patterns and the step $G_{k+1}$ with $N_{unlearnt}$ patterns. The value of $E_{unlearnt}$ is therefore constant during $L_k$, i.e. for any given local training epoch,

$$E_{total} = E_{learnt} + C \tag{4}$$

Figure 5 illustrates how the RPHP algorithm, a single staged hybrid algorithm such as the linear interpolation algorithm (Vasconcelos et al., 2001) or the Lamarckian evolution (Ackley et al., 1993), and the multisieving algorithm (Lu et al., 1995)) find their solutions. The graph shows a hypothetical one-dimensional error surface to be optimized with respect to w. Assume that at the end of the global training phase, solution $S_g$ has an error value $E_g$ computed according to Equation 2.

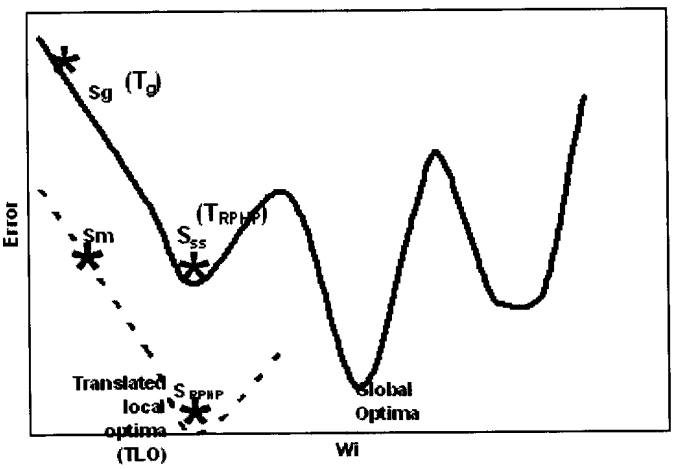


**Figure 5. Illustration of solutions found by (i) RPHP ($S_{RPHP}$) (ii) Single-staged hybrid search ($S_{SS}$) (iii) Multisieving algorithm ($S_m$)**

A single-staged hybrid search algorithm, at this stage, will either try to search for an optima or, if the probability of finding the optima is too small, climb the hill and reach the local optima marked by $S_{ss}$.

However, by virtue of equation 4, $E_{unlearnt}$ is a constant value C. The error curve (represented by the dotted line), is just a vertically translated copy of the part of the original curve, which is of interest to us.

Now, if we consider the multisieving algorithm (Lu et al., 1995), the data splitting depends on the error tolerance of learnt patterns $\xi$, as defined in equation 1. The solution $S_g$, in itself, is found by a local training algorithm. The final solution $S_m$ is a vertically translated $S_g$.

On the other hand, we can see from Figure 5, that the translated local optima (TLO) due to the splitting of patterns is more optimal than the other optimal solutions, i.e,

$$E_{TLO} \leq E_{global} \qquad (5)$$

This is by virtue of the fact that $E_{learnt} \rightarrow 0$ as we approach the optimal point. Further, from Equation 4, $\partial E_{total}/\partial_w \rightarrow 0$ as $\partial E_{learnt}/\partial_w \rightarrow 0$ Therefore, the solution found by the RPHP algorithm is a pseudo global optima, *i.e, it could be a local optima but it appears global from the perspective of a pattern subset.*

It must be noted that the solution found by the multisieving algorithm, $S_m$, can only be equal to the translated local optima TLO if the error tolerance $\xi$ is set to optimum values. However, due to local training at the end of each recursion, the solution adapts itself accordingly, regardless of the error tolerance $\xi$. Finding a pseudo global optima therefore reduces the dependence of the algorithm on the error tolerance of learnt patterns $\xi$. It is also the natural optima based on the data subset.

***Note:*** Since early stopping is implemented during local training so as to prevent overtraining, the optima found by RPHP may not necessarily be $S_{RPHP}$, but in the vicinity of $S_{RPHP}$.

**2.5. The advantage of the RPHP pattern distributor**

As described in section 2.1.2, testing with the RPHP algorithm is implemented using the Kth nearest neighbor based pattern distributor. The pattern distributor processes the incoming test pattern and checks to see which pseudo global optimal solution can fit it best. The solution chosen is switched on while the others are switched off. The chosen solution is applied to the test pattern and the final output for the given test pattern obtained.

The error of the system is, therefore, the combination of the errors of the pattern distributor and the error of the RPHP system. The KNN algorithm has a fairly high accuracy but patterns that are close to the neighborhood boundary (i.e., points marking class separations) are more susceptible to errors. Moreover, the distribution mechanism of RPHP is automatic, and therefore, its properties and results cannot be predicted.

The hypothesis of 50% PHP, earlier proposed (Guan and Ramanthan, 2005), is used in order to have a better control over the pattern distribution. This is explained below:

If the error probability of the pattern distributor is $P_{e,PD}$, the number of test patterns classified wrongly by the pattern distributor is $T_{ts}P_{e,PD}$, where $T_{ts}$ is the total number of testing patterns The total number of test patterns being predicted wrongly by a RPHP system is therefore,

$$T_{ts,e} = T_{ts}\sum_{i=1}^{K}\left[1-(1-P_{PD,i})(1-P_{e,i})\right] \qquad (6)$$

where $P_{PD,i}$ is the probability of wrong classification of the KNN algorithm for the $i$th solution, K is the total number of solutions resulting from the RPHP algorithm and $P_{e,i}$ is the probability of wrong prediction of the pseudo global optimal solution $i$, where $i \in [1, K]$.

If we express $P_{PD,i}$ as a ratio of wrongly classified patterns to solution $i$, i.e, $P_{PD,i} = {}^{T_{PD,i}}\!/_{T_{ts}}$, we can simplify equation 6 as given below:

$$T_{ts,e} = \sum_{i=1}^{K}T_{PD,i} + T_{ts}\sum_{i=1}^{K}P_{e,i} + \sum_{i=1}^{K}T_{PD,i}P_{e,i} \qquad (7)$$

The first term of equation 7 is the total error of the pattern distributor and depends on the accuracy of the KNN algorithm. However, considering the last two terms of the equation and expressing $P_{e,i}$ as a ratio of the number of test patterns that are wrongly solved by the pseudo global optima i to the number of test patterns presented to the solution i, we obtain $P_{e,i} = {}^{T_{e,i}}\!/_{T_i}$.

From Figure 5, we can see that the number of patterns $T_{RPHP}$, learnt by the RPHP solution $S_i$ is such that $T_{RPHP} \geq T_g$, meaning that the larger the distance between $S_g$ and $S_{ss}$, the larger the difference between $T_{RPHP}$ and $T_g$ and more than one pseudo global optima is representative of a given pattern, i.e. $P_{e,i,RPHP} = {}^{T_{e,i}}\!/_{T_{RPHP}} \leq {}^{T_{e,i}}\!/_{T_g}$. This improves the test accuracy, as can be seen from equation 7.

*In some of our earlier experiments on PHP (Guan and Ramanathan, 2005), we observed that implementing 50% PHP results in a large percentage of chromosomes in the global optimal or the near global optimal neighborhood,* implying that $T_{RPHP}$-$T_g$ will be larger. Since we are using 50% PHP training as a switching point for local optimization and recursion, we are minimizing $T_{ts.e}$ and therefore cushioning the error of the pattern distributor. This is a unique feature of the percentage and topology based training which cannot be implemented on a class based or manual task decomposition algorithm (Guan et al., 2003; Fu et al., 2001; Lu et al., 1995; Guan and Li, 2002).

## 3. THE RPHP EFFICIENCY MODEL

In order to illustrate the advantage that RPHP has over other algorithms with respect to training time, we assume that, for all $i \in \alpha$, each ε vs $w_i$ curve has G globally optimal solutions and L locally optimal solutions, where ε is the training error. Given that the problem can be solved with $\alpha$ independent parameters, i.e., a chromosome with $\alpha$ elements, we make the following assumptions to simplify our analysis:

1. There is at least one global optimal solution for the problem considered. i.e., for all *i*, each ε vs $w_i$ curve, must have at least one global optimal solution.
2. All the global optima occur with probabilities $P_g$ and the local optima occur with probabilities $P_l$, i.e., for a single dimension,

$$GP_g + LP_l = 1 \tag{8}$$

A global optimal solution occurs only if the values of training error are optimum for all dimensions. The total probability of finding a global optimal solution in a $\alpha$-dimensional error space is therefore the product of $GP_g$, over all the i dimensions, $P_{gs} = \prod_{i=1}^{\alpha}\left[(GP_g)_i\right]$.

From Definition 1, since pseudo optimal solutions only require the presence of a local minima, the probability of finding a pseudo globally optimal solution is

$$P_{pgs} = P_{gs} + P_{ls} = 1 \tag{9}$$

From equation 9, we know that a pseudo global optimal solution will always be found, for any problem. Therefore $P_{gs} \le P_{pgs}$. We can therefore deduce that

$$N_{gs} \ge N_{pgs} \tag{10}$$

Where $N_{gs}$ and $N_{pgs}$ are the number of epochs of learning required to find a global solution and a pseudo global solution respectively.

For the RPHP algorithm, we require K recursions to decompose the problem. The following considerations have to be taken into account to find the best value of K.

*Condition 1: Training Accuracy*

For good training accuracy, every pattern, no matter how difficult, needs to be learnt. In order to make sure that every pattern is learnt, K recursions are required, where $K = ceil(\log_2(T))$.

*Condition 2: Generalization accuracy*

In order to ensure generalization accuracy and to filter out noise components, we need to make sure that there are enough training patterns in the *K*th recursion. For this, we use the rule of thumb advocated by in Haykins (1999) for the number of training patterns

required for good generalization accuracy in a problems. $K$ is therefore set such that the number of training patterns in the $K$th recursion is greater than 10 times the number of free parameters $\alpha$.

The ideal RPHP situation (assuming no stagnation) follows 50% division in patterns. This gives us a geometric reduction in training patterns i.e,

$T_1 = T, T_2 = T/2, T_3 = T/4 \ldots\ldots T_K = T/2^{K-1}$

Therefore, for optimal training we require $\frac{T}{2^{K-1}} > 10\alpha$

Solving for K, we obtain

$$K < \log_2\left(\frac{T}{10\alpha}\right) + 1$$

Therefore,

$$K = ceil\left(\log_2\left(\frac{T}{10\alpha}\right)\right) \qquad (11)$$

Conditions 1 and 2, give us an ideal value of K as given by equation 11. Note that if the training is stopped by stagnation, it means that there is little correlation between the training, testing and validation data. In this case, more training patterns are required for the reliability of the solution, and a smaller K is advocated.

*Condition 3: Training time*

As K recursions are required for RPHP to give accurate training results, we can say that RPHP training is faster than other methods that focus on finding a single global optima listed in the survey (Carvalho and Freitas, 2004; Rovithakis et al., 2004; Vasconcelos et al., 2001; Guan and Ramanathan, 2005; Yasunaga et al., 1999) if the number of epochs required to find a global optimal solution is greater than the number of epochs required to find $K$ local optimal solutions.

$$N_{gs} > \sum_{i=0}^{K} N_{pgs,i} \qquad (12)$$

where $K$ is as given in equation 11. We assume that the number of epochs required to obtain an optimal solution is inversely proportional to the probability of obtaining that solution, i.e., $P = \beta/N$, where P is the probability of finding the optima and N is the number of epochs required to find the optima and $\beta$ is the proportionality constant. We further assume that for a problem with G global optima and L local optima, $\beta_1 = \beta_2 = \ldots\ldots = \beta_{G+L}$, i.e, the probability of finding a pseudo global optima is equal to or greater than the probability of finding a global optima.

We hypothesize that for inequality 12 to be true, inequality13 should be satisfied

$$P_{gs} < \frac{1}{K} \qquad (13)$$

***Theorem 1.*** *For the RPHP technique to find K pseudo optimal solutions and be more efficient than a technique to find the single global optimal solution, the probability of finding the global optimal solution must be smaller than 1/K.*

**Proof: Apagoge is used.**

We prove the validity of inequality 13 by assuming that the opposite is true, ie, $P_{gs} \geq 1/K$. This means that the probability of finding a global minimum is not very difficult. Therefore,

$\frac{1}{N_{gs}} \geq \frac{1}{\sum_{i=1}^{K} N_{pgs}}$, resulting in $N_{gs} \leq \sum_{i=0}^{K} N_{pgs,i}$, i.e., classical GA will perform faster than the RPHP. Therefore, for RPHP to be faster than classical GA, inequality 13 must hold.

***Implications of Theorem 1***

From the experiment results, we can observe that the number of recursions is usually small. Therefore, $P_{gs}$ is often less than *1/K* and the RPHP algorithm solves the problem in fewer epochs. On the other hand, the number of training patterns required for RPHP to be successful in generalization is determined by equation 11. This means that RPHP requires more training patterns than single-staged algorithms. No guarantees can be made on generalization accuracy if the number of training patterns is too small. RPHP will then try to find the best solution based on the training and validation data.

## 4. RPHP1 EXPERIMENTS AND RESULTS

### 4.1. Problems considered and experimental setup

Four curve fitting problems and one classification problem (two spiral) were chosen as benchmark problems.

#### 4.1.1. Curve fitting problems

The RPHP algorithm was set to solve for the coefficients of each of the problem equations below. A normally distributed noise $\xi$ was added to the training, testing and validation sets in order to test the generalization capability of the RPHP algorithm. All the problem definitions were obtained from the non-linear regression repository (NIST, 2000) and in order to achieve the sufficient number of training patterns, data was artificially generated using the problem definitions and added to the data in the repository.

To obtain the training, testing and validation datasets, the dataset was randomly split into three parts in the ratio of 2:1:1.

The problems were chosen according to varying degrees of difficulty, the LINEAR problem being the easiest and the HAHN problem being the most difficult. The difficulty of a given problem is measured by the $P_{gs}$ value of the problem (i.e, the probability of finding a global optimal neighborhood). The pseudocode for evaluating the $P_{gs}$ value of a curve fitting problem is given in Figure 6. Note that this value of $P_{gs}$ can only be obtained when the problem structure is fixed and the ideal values of independent parameters are known. The values used here are therefore simply measures of difficulty and cannot be found in real world problems.

The procedure in Figure 6 will give us the error vs $w_i$ curve for a single dimension. From the curve, the value of $P_{gs}$ for the dimension *i* is given by:

$$P_{gs,i} = \frac{\sum_{j=1}^{G} w_{j,g,\max} - w_{j,g,\min}}{w_{j,\max} - w_{j,\min}} \qquad \textbf{(14)}$$

where G is the number of global optima in the dimension i. The value of $P_{gs}$ for the whole problem with α dimensions is therefore $P_{gs} = \prod_{i=1}^{\alpha} P_{gs,i}$

1. Assume the problem has *K* independent parameters. Work with a single chromosome.
2. Initialize all the chromosome element values to optimal values.
3. For all *i*, where *i*<=*N*,
   a. Keep all other element values constant
   b. Vary the value of *i* in small steps to cover the range of value(*i*)
   c. Evaluate for each *i*

$$E = \frac{1}{T}\sum_{i=0}^{T}\sum_{j=0}^{O}(D_{i,j} - O_{i,j})^2$$

   d. Plot the graph of E against value(*i*)

**Figure 6. Pseudo code to evaluate the value of $P_{gs}$ for a given problem**

*Problem 1. LINEAR*

The LINEAR problem is the control problem and is used to show that when $P_{gs}$ is large, i.e, when inequality 13 is not satisfied, the use of RPHP is not recommended. The LINEAR problem is a classic linear regression problem and is defined in Equation 15, with the values of $w_1$, $w_2$ and $w_3$ as 40, 0.005 and 0.2516. The $P_{gs}$ value of the LINEAR problem is 1.0 when 500 training patterns were used.

$$y = w_1 + w_2 x + w_3 x^2 + \xi \qquad (15)$$

*Problem 2: ENSO*

The ENSO problem is a benchmark problem used in non linear regression and curve fitting problems. The data are monthly averaged atmospheric pressure differences. The *x* values are the time, while the *y* values are the atmospheric pressure. Equation 16 defines the problem. The problem was trained with 500 training patterns.

$$y = A_1 + A_2\cos\left(\frac{2\Pi x}{12}\right) + A_3\sin\left(\frac{2\Pi x}{12}\right) + A_5\cos\left(\frac{2\Pi x}{A_4}\right) + A_6\sin\left(\frac{2\Pi x}{A_4}\right) + A_8\cos\left(\frac{2\Pi x}{A_7}\right) + A_9\sin\left(\frac{2\Pi x}{A7}\right) + \xi \qquad (16)$$

Table 3 shows the optimal values for the coefficients $A_1$ to $A_9$

Table 1. Coefficient values for the ENSO problem

| Index | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ | $A_7$ | $A_8$ | $A_9$ |
|---|---|---|---|---|---|---|---|---|---|
| Value | 10.51 | 3.07 | 0.53 | 44.31 | -1.62 | 0.525 | 26.89 | 0.212 | 1.49 |

The ENSO problem can be considered to be of medium difficulty. The value of $P_{gs}$ for the ENSO problem is 0.01.

*Problem 3: GAUSS*

The probability that this problem will find a global optimum is approximately $3.5 \times 10^{-4}$. Equation 17 defines the problem and Table 2 lists the coefficients. 250 patterns were used to train the problem.

$$y = B_1 \exp(-B_2 x) + B_3 \exp[-(x - B_4)^2 / B_5^2] + B_6 \exp[-(x - B_7)^2 / B_8^2] + \xi \quad (17)$$

Table 2. Coefficient values for the GAUSS problem

| Index | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ | $B_6$ | $B_7$ | $B_8$ |
|---|---|---|---|---|---|---|---|---|
| Value | 98.778 | $1.04*10^{-2}$ | 100.489 | 67.481 | 23.129 | 71.994 | 178.998 | 18.389 |

*Problem 4: HAHN*

The HAHN equation is the result of a NIST study involving the thermal expansion of copper. y is the coefficient of thermal expansion, and x is temperature in degrees Kelvin. 1000 training patterns were used in training. The $P_{gs}$ value for the HAHN problem is $2.04x10^{-5}$. Equation 18 defined the problem. The coefficient values are given in Table3.

$$y = \frac{w_1 + w_2 x + w_3 x^2 + w_4 x^3}{1 + w_5 x + w_6 x^2 + w_7 x^3} + \xi \quad (18)$$

Table 3. Coefficient values for the HAHN problem

| Index | $W_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ |
|---|---|---|---|---|---|---|---|
| Value | 1.078 | $-1.266x10^{-1}$ | $4.087x10^{-3}$ | $-1.43x10^{-6}$ | $-5.76x10^{-3}$ | $-1.23x10^{-7}$ | $2.40x10^{-4}$ |

### 4.1.2. The Two spiral problem

Simulations were carried out on the two-spiral algorithm in order to illustrate the advantage that RPHP has in terms of a) Evolutionary search, b) Pseudo global optima, c) Recursive subset search. Results are compared with those obtained by the multisieving algorithm (Lu et al., 1995), which implements only recursive subset search, and the Topology-based Subset Selection (Lasarzyck et al., 2004), which implements single subset finding based on evolutionary algorithms. Both algorithms were discussed in our literature survey. The two spiral dataset was chosen because it is considered a fairly difficult problem and because it was the only experiment referred to by *Lu et al*, in the multisieving algorithm (Lu et al., 1995).

The dataset consists of 194 patterns, which were decomposed into sets of 2:1:1 for comparison with the multisieving algorithm. To ensure a fair comparison to the TSS algorithm (Lasarcyck et al., 2004), test and validation datasets of 192 patterns each were constructed by choosing points next to the original points in the dataset as mentioned in the TSS paper.

A messy genetic algorithm with variable length chromosomes was implemented based on Goldberg's messy algorithms (Goldberg, 1991). Each chromosome encoded the weights and structure of a two-layered feedforward neural network. During each recursion, the best network obtained from global training was used to perform the local training. Local training was simply performed by training the resulting neural network using the learnt patterns and the back propagation algorithm.

### 4.1.3. Experimental parameters

The following table summarizes the problem information and training parameters. The training parameters were kept constant through simulations on all the datasets.

Table 4. Experimental Parameters used in the RPHP algorithm

| **Parameter** | | | | | | **Value** |
|---|---|---|---|---|---|---|
| Crossover Probability, $P_c$ | | | | | Global search | 0.9 |
| | | | | | Local search | 0.1 |
| Small change Mutation Probability, $P_m$ | | | | | Global search | 0.1 |
| | | | | | Local search | 0.9 |
| Large change mutation probability | | | | | Global search | 0.7 |
| | | | | | Local search | 0.0 |
| Pattern learning tolerance for global training $\xi$ | | | | | | 0.1 |
| Population size | | | | | | 50 |
| NN learning rate (For two spiral problem) | | | | | | $10^{-2}$ |
| Generalization loss tolerance (Ref Appendix) | | | | | | 1.5 |
| Number of neighbors considered for KNN pattern distributor | | | | | | 1 |
| **Problem ID** | | **Linear** | **ENSO** | **Gauss** | **HAHN** | **Two-spiral** | |
| Problem type | | Curve fitting | | | | Classification | |
| Number of variables | | 3 | 9 | 8 | 7 | 2 real valued inputs, 2 classes. NN solved | |
| Fitness function | | $E = \frac{1}{T}\sum_{i=0}^{T}\sum_{j=0}^{O}(D_{i,j} - O_{i,j})^2$ | | | | | |
| Number of Patterns | Training | 500 | 500 | 250 | 1000 | 194 | |
| | Testing | 250 | 250 | 125 | 500 | 192 | Test and validation sets used to compare with TBS and multisieving |
| | Validation (V1+V2) | 250 | 250 | 125 | 500 | 192 | |
| Number of recursions | | 2 | 3 | 3 | 4 | 3 | |

### 4.2. Experimental results

#### 4.2.1. RPHP vs. single staged hybrid training for curve fitting

The graphs below show the typical performance of RPHP, PHP (Guan and Ramanthan, 2005) and the linear interpolation hybrid algorithm (Vasconcelos et al., 2001). We chose the Linear Interpolation algorithm for comparison as it is one of the more recent hybrid algorithms developed using evolutionary algorithms only. Each simulation was carried out 15 times. Table 5 gives the average values and standard deviations of the generalization accuracy of each experiment.

From the training graph of the linear problem, we can observe that RPHP takes a longer training time than PHP and LI. This is expected, as the LINEAR problem does not

satisfy the condition in inequality 13. All the solutions reach the global optimum and the generalization accuracy is 100%.

The simulation results of the other problems show that the RPHP algorithm has a lower training error than both PHP and Linear Interpolation. In particular, both the GAUSS and the HAHN problems have a very small value of $P_{gs}$. It is observed from Figure 7 that the number of training epochs when using RPHP is much lower than the results of using Single staged hybrid training.

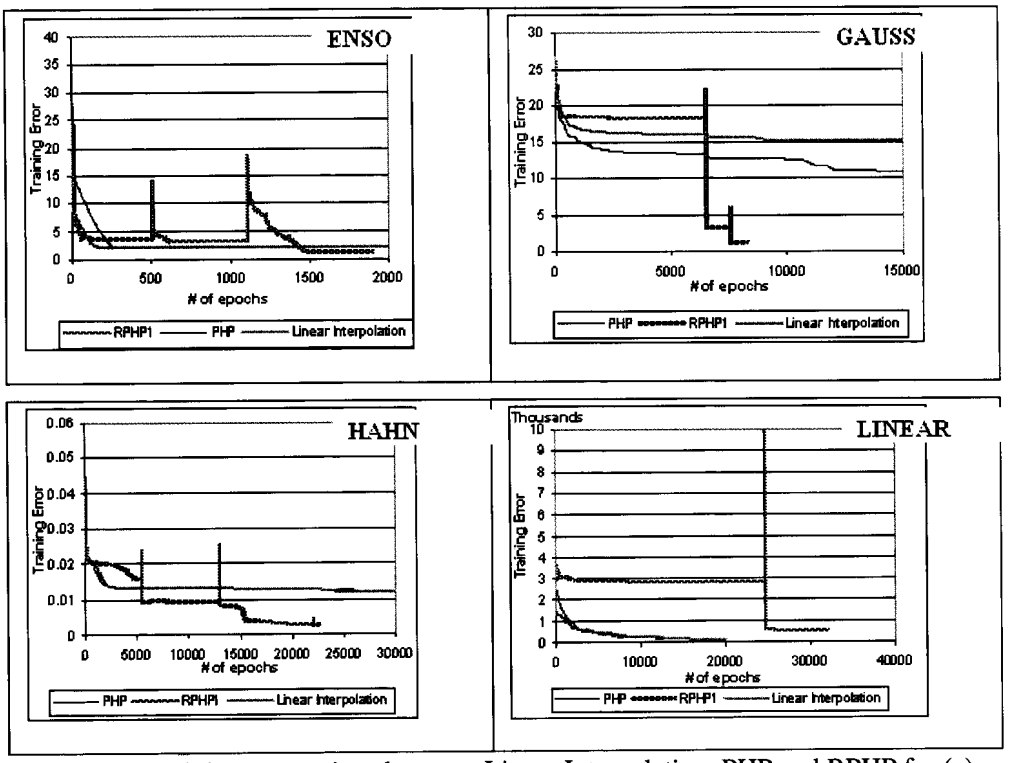


Figure 7. Training comparison between Linear Interpolation, PHP and RPHP for (a) ENSO (b) GAUSS (c) HAHN and (d) LINEAR

Table 5. Comparison of Generalization accuracy values between Linear Interpolation, PHP and RPHP

| *Problem Name* | **Linear Interpolation Accuracy** | | **PHP Accuracy** | | **RPHP Accuracy** | |
|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ |
| **ENSO** | 0.74 | 0.132 | 0.74 | 0.11 | 0.85 | 0.060 |
| **GAUSS** | 0.64 | 0.214 | 0.719 | 0.151 | 0.87 | 0.089 |
| **HAHN** | 0.404 | 0.091 | 0.451 | 0.094 | 0.680 | 0.063 |
| **LINEAR** | 1.0 | 0 | 1.0 | 0 | 1.0 | 0 |

### 4.2.2. Studies on the two spiral problem

According to Lu et al., (1995), the multisieving algorithm achieved 100% training accuracy. However, with the small number of training patterns, 100% training accuracy

does not hold much value unless accompanied by equal generalization capability. We therefore compare the data splitting mechanisms of the multisieving algorithm (as reported in Lu et al. (1995)) and the RPHP algorithm. This will show us the clear advantage of using evolutionary algorithms as a base for data decomposition. Figure 8(a) shows the original training data of the two spiral set and the decomposition of the data by the RPHP and the multisieving algorithm 8(b). The lines show separation between the two classes of the two spiral problem.

It is observed that the global search implicitly finds separable sets of data, i.e., compared to the original dataset, the decomposed datasets are more separable and hence more suited for backpropagation training. The separation is better defined with the RPHP algorithm than with the multisieving algorithm

As mentioned in section 4.1.2, the generalization accuracy of the RPHP solution when applied to the two spiral dataset was evaluated based on the accuracy of artificially generated 192 test patterns in a method similar to that in the TBS algorithm (Lasarcyck et al., 2004). Both the TBS algorithm and the RPHP algorithm validates the solution based on a 192-pattern validation data set, generated in the same way as the test set. Table 6 compares the results of the constructive backpropagation(CBP) algorithm (Lehtokangas, 1999), TBS, multisieving (Lu et al., 1995) and RPHP in terms of the mean classification accuracy and the 95% confidence interval of the solutions

Table 6. Mean classification accuracy and 95% confidence interval on the test set for the two-spiral problem.

| **Algorithm** | **RPHP** | **CBP** | **TBS** | **Multisieving** |
|---|---|---|---|---|
| **Mean Generalization accuracy** | 0.8892 | 0.5062 | 0.72 | 0.7639 |
| **95% confidence interval** | [-0.0031,+0.0031] | [-0.0014,+0.0013] | [0.029,+0.015] | [-0.093,+0.093] |

The superior performance of RPHP both in terms of data decomposition as well as in terms of generalization accuracy reinforces the advantage of evolutionary search for learnt patterns, pseudo global optima and recursive subset selection.

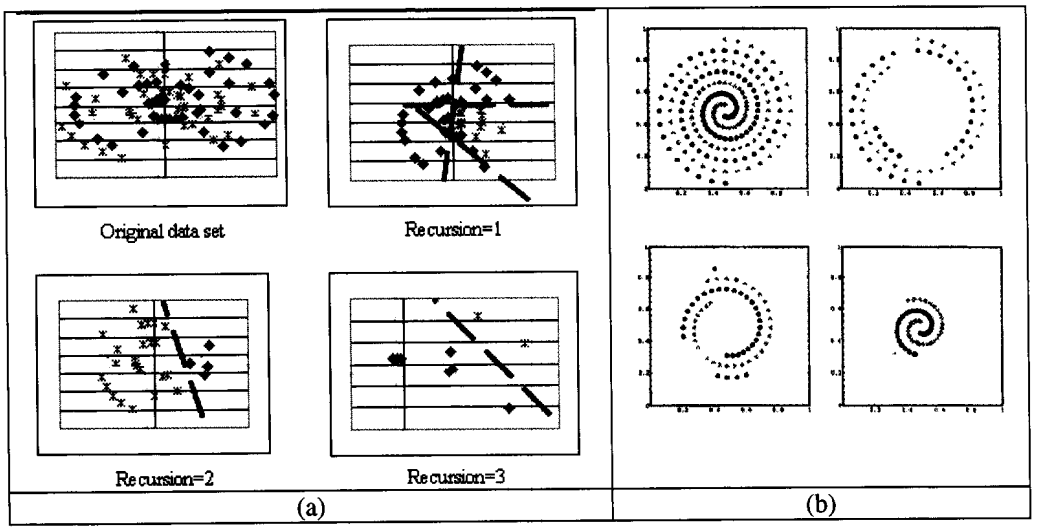


Figure 8. (a) One instance of the RPHP decomposition of the two-spiral dataset. (b). Decomposition of the two spiral data by the multi sieving algorithm (Lu et al, 1995)

## 5. CAN WE IMPROVE RPHP?

### 5.1. The necessity for improving the RPHP algorithm

From the training graphs in Figure 7, we can observe that each of the graphs display a trend as shown in Figure 9.

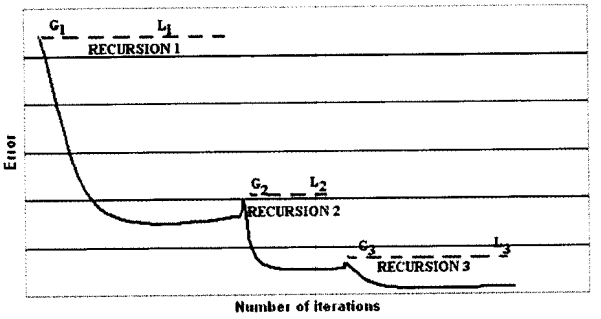


Figure 9. Expected behavior of the RPHP1 algorithm (training error vs number of epochs)

In Figure 9, the expected global and local optimization curves for recursion $k$ are marked as $G_k$ and $L_k$ respectively. We can observe, in each case, a marked increase in the training error at the end of each recursion due to the introduction of a new population. While this characteristic does not affect the final result, we feel that the algorithm can be improved to make the training smoother.

Secondly, though the RPHP1 algorithm resulted in faster training and good generalization accuracy, experimentally, it was found that the algorithm tended to over-

train the solutions at times. The theoretical speculation for this behavior of RPHP is given later in this section. The possibility of RPHP1 to over train is also attributed to the PHP component of the algorithm and its dependence on error tolerance.

### 5.2. The RPHP2 algorithm

RPHP2 is proposed to solve this problem using a simple pattern validation component at the end of local training. For implementing RPHP2, the validation set is split into two parts, V1 and V2. V1 acts to prevent overtraining while V2 acts as a pattern validator, so as to ensure that overtraining has not taken place in the global training phase itself. The RPHP2 algorithm is similar in form to the RPHP algorithm with the following pseudocode implemented at the end of local training in each recursion.

*Pattern Test on the training data (G1), V1 and V2*
*IF [Accuracy($V_2$)<1/2(Accuracy($V_1$)+Accuracy($G_1$)- $\delta$]*
*{*
*a. Increase the error tolerance $\varepsilon$ by $\zeta$*
*b. Repeat that Recursion i.e., Global training*
*}*

In the pseudocode above, Accuracy(X) refers to the percentage of patterns in the dataset X correctly predicted by the solution. For all the experiments carried out, the value of $\zeta$ and $\delta$ were both set to 5%.
Further, RPHP2 inherits the best chromosome at the end of each recursion. All other chromosomes are randomly generated. This reduces the increase in the training error at the end of each recursion.

### 5.3. Over training and RPHP2

The RPHP training is based on the PHP algorithm, the performance of which requires the accurate setting of the error tolerance ($\xi$) values. Inaccurate values of optimal tolerance (a problem dependent value) can result in over-training of the problem. The use of pseudo global optima prevents inaccuracies due to too large values of $\xi$. However, over-training can result if $\xi$ is initially set to a value too low. RPHP2 implements continuous error validation as well as pattern validation at the end of each recursion. The pattern validation protects RPHP2 against possible overtraining due to low values of $\xi$.

Consider the hypothetical situation described by Figure 10, showing the training error and validation error with the number of epochs. If we let optimal tolerance for best possible training to be $\xi_o$, we define solutions A and B such that $\xi_A > \xi_o$ and $\xi_B < \xi_o$

**Case 1:** $\xi_A > \xi_o$

In this case, global training will result in solution A and if we condition 50% PHP (Guan et al., 2005), we obtain the training and testing pattern accuracy as given by equations 19 and 20 respectively.

$$L_{g,tr} = \frac{1}{T}\sum_{i=0}^{T}\delta\left\{\left[\frac{1}{O}\sum_{j=0}^{O}\phi\left[\xi_A - (D_{i,j} - O_{i,j})^2\right]\right] - 1\right\} \approx 0.5 \quad \textbf{(19)}$$

$$L_{g,ts} = \frac{1}{Ts}\sum_{i=0}^{Ts}\delta\left\{\left[\frac{1}{O}\sum_{j=0}^{O}\phi\left[\xi_A - (D_{i,j} - O_{i,j})^2\right]\right] - 1\right\} = L_{g,tr} - \mu_{g,A} \quad \textbf{(20)}$$

Where $\mu_{g,1}$ is the accuracy difference in the percentage of patterns learnt between the training and the testing sets. Local training and continuous error validation is now set to optimize the solution to obtain O. If $\xi_A > \xi_o$, then $L_{tot,ts} > L_{g,ts}$ and the following relationship holds $\mu_{tot,A} = L_{tot,tr} - L_{tot,ts} = \mu_o \leq \mu_{g,A}$

We say that no pattern generalization loss has occurred. And the optimal solution O has been found.

**Case 2:** $\xi_B < \xi_o$

In this case, global training results in solution B. After local training, we obtain $L_{g,ts} = L_{g,tr} - \mu_{g,B}$. However, with solution B, there has already been a fair amount of generalization loss. Therefore, with error validation based termination, the best solution is therefore, B and $\mu_{tot,B} = \mu_{g,B} > \mu_o$ where $\mu_o$ is the accuracy loss allowance for optimal training. RPHP1 therefore exhibits possible over training capability at low $\xi$ values.

RPHP2 implements pattern validation and global retraining if there is large difference in the pattern accuracy of the training set and the pattern accuracy of the validation set.

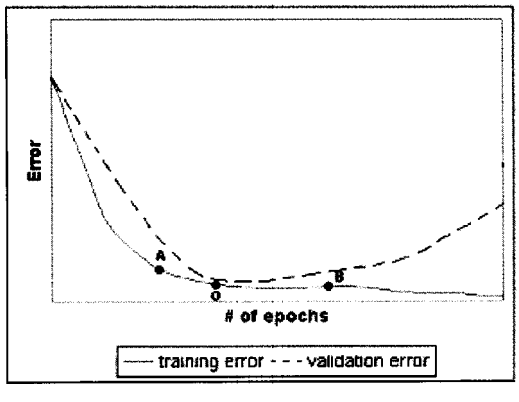


Figure 10. Hypothetical situation showing typical behaviors of training and validation errors with # of epochs

### 5.4. Experimental results on the use of RPHP2

The ability of the RPHP2 algorithm to provide better training is clearly observed from Figure 11. RPHP2 displays lower increase in training error at the end of each recursion when compared to RPHP (RPHP1), resulting in some turbulence in training. This is attributed to the pattern validated component of RPHP2. Table 7 compares the generalization accuracies of RPHP1 and RPHP2. It is observed that the implementation

of the pattern validated component results in a more reliable algorithm – the standard deviation of generalization accuracy is much lower in the case of RPHP2.

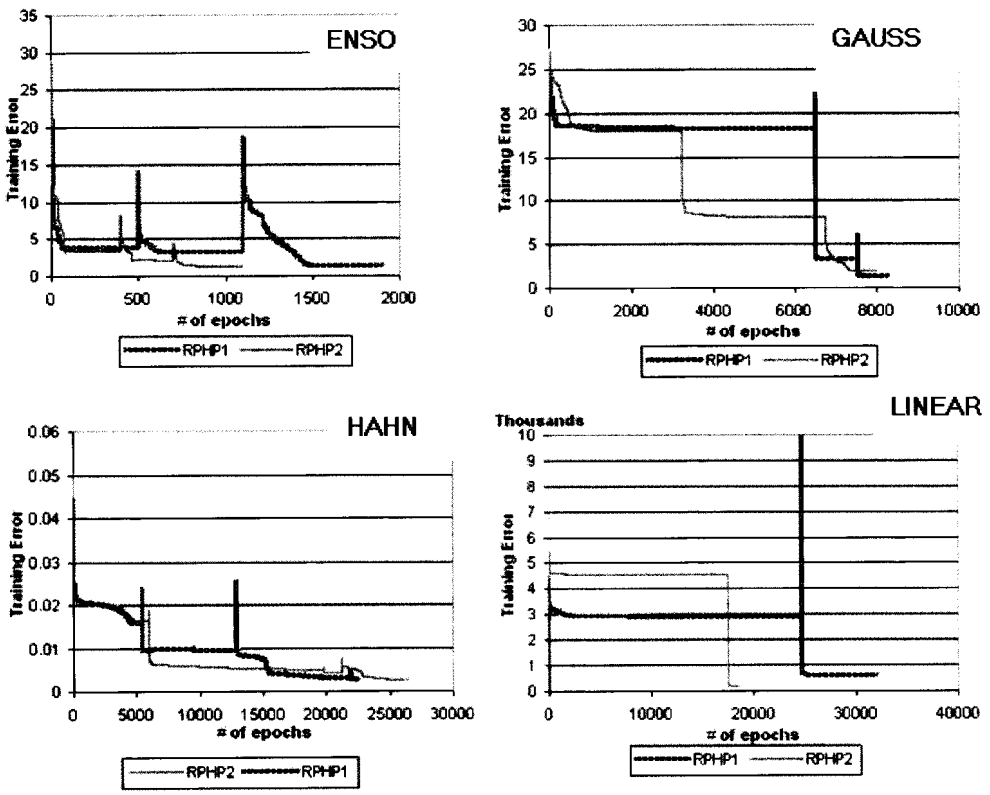


Figure 11. Training comparison between RPHP1 and RPHP2 for (a) ENSO (b)GAUSS (c)HAHN and (d) LINEAR

Table 7. Comparison of Generalization accuracy values between PHP and RPHP

| *Problem Name* | **RPHP1 Accuracy** | | **RPHP2 Accuracy** | |
|---|---|---|---|---|
| | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ |
| **LINEAR** | 1.0 | 0 | 1.0 | 0 |
| **ENSO** | 0.85 | 0.060 | 0.85 | 0.041 |
| **GAUSS** | 0.87 | 0.089 | 0.921 | 0.069 |
| **HAHN** | 0.680 | 0.063 | 0.711 | 0.055 |

## 6. DISCUSSIONS, CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced two versions of the Recursive Percentage based Hybrid Pattern training algorithm (RPHP and RPHP2). Both algorithms use the concept of pseudo global optimal solutions to find several suitable local optimal solutions. Each local optimal solution acts as a global optimal solution from the point of view of a chosen

set of patterns. Theoretically, we have shown that RPHP performs better than traditional methods as long as the probability $P_{gs}$ of finding a global optimal solution is less than 1/K where K is the number of recursions. The use of PHP (Guan et al., 2005) in RPHP is hypothesized to act as a cushion against pattern distributor error. We also observed that the use of evolutionary search in task decomposition results in more separable subsets. From experimental simulations we can conclude that the RPHP2 algorithm is superior to traditional subset finding and hybrid algorithms in terms of both generalization accuracy and training time.

While experimental results illustrate that the RPHP algorithm obtains better generalization accuracy, several problems still need to be dealt with. Firstly, the data that has been used in the paper, with the exception of the two-spiral dataset, is suited for curve fitting problems. Further, in order to create the necessary number of patterns, we had generated "artificial" data using a noise factor. One important future work of the algorithm would be to explore the training possibilities for classification as well as for problems with insufficient data.

Secondly, the big question remains as to why the introduction of global search during training improves the results when compared solely with the use of local search based recursive algorithms such as the multisieving algorithm. While intuitively the answer can be attributed to the fact that global search algorithms span the search space more effectively, a mathematical proof of performance improvement attributed to global search needs to be established.

Finally, an important precaution to be taken during training is "over decomposition". It is possible that the training may isolate just a small portion of the training set, which can eventually get over trained. This problem is not present in classical training and remains to be solved.

Further work on the RPHP algorithm would include hybrids of TBS and RPHP as well as a deeper study on its application to neural networks.

## REFERENCES

1. Ash T (1989), Dynamic node creation in backpropagation networks, ICS Report 8901, UCSD.
2. Ackley, D.H. and Littmann M.L (1993). A Case for Lamarckian Evolution. Artificial Life III Edited by Langton C. G, MA. Addison Wesley, pp 3-10.
3. Andreou A S, Efstratios F, Spiridon G, Likothanassis D (2002), Exchange-Rates Forecasting: A Hybrid Algorithm Based on Genetically Optimized Adaptive Neural Networks, Computational Economics, 20(3), pp191-210
4. Aydin M E, Fogarty T C (2004), A distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimisation Problems, Journal of Heuristics, 10 (3), pp 269-292
5. Breiman L (1984), Classification and regression trees, Wadsworth International Group.
6. Cantu-Paz, E. and Kamath C (2005), An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, pp 915-927
7. Carvalho D R, Freitas A A (2004), A hybrid decision tree/ genetic algorithm method for data mining, Information Sciences: an International Journal, 163(1-3), pp 13-35

8. Chaiyaratna. N, Zalzala. AMS (2000), Hybridization of neural networks and a genetic algorithm for friction compensation, Proceedings of the 2000 congress on Evolutionary Computation, 1.
9. Devi. S, Panda D.C, Pattnaiik, S.S, Khuntia, B, Neog, D.K (2003), Initializing Artificial neural networks by genetic algorithm to calculate the resonant frequency of single shortpost rectangular patch antenna, IEEE Antennas and Propagation Society International Symposium, 3, pp 144-147.
10. Dokur Z(2002), Segmentation of MR and CT images using Hybrid Neural Network Trained by Genetic Algorithms, Neural Processing Letters, 16(3), pp211-225
11. Foody G M (1998), Issues in training set selection and refinement for classification by a feedfoward neural network, Geoscience and Remote Sensing Symposium Proceeding
12. Fu H C, Lee Y P, Cheng C C, Pao H T (2001), Divide and Conquer learning and modular percepteron networks, IEEE transactions on neural networks, 12(2), pp 250-263
13. Gathercole C, Ross P, Bridge S (1994), Dynamic training subset selection for supervised learning in genetic programming, Parallel Problem Solving from Nature, pp312-321
14. Goldberg D E (1989), Genetic Algorithms in Search, Optimization and Machine Learning: Addition Wesley.
15. Goldberg, D.E., Deb, K., and Korb, B (1991), Don't worry, be messy, Proceedings of the Fourth International Conference in Genetic Algorithms and their Applications, edited by R. Belew and L. Booker pp24-30.
16. Guan S U, Ramanathan K (2005), A Lateral Symmetry approach to Percentage based Hybrid Pattern (PHP) Training, Journal of Intelligent Systems, 13(2), pp 123-150.
17. Guan S U and Zhu F (2004), Class Decomposition for GA-based Classifier Agents – A Pitt Approach, IEEE Transactions on Systems, Man, and Cybernetics B, 34(1), pp381-392
18. Guan S U and Li S (2002), Parallel Growing and Training of Neural Networks Using Output Parallelism, IEEE Trans. on Neural Networks, 13(3), pp 542 -550
19. Guan S U, Neo T and Bao C (2004), Task Decomposition using Pattern distributor, Journal of Intelligent Systems, .
20. Haykins, Simon (1999), Neural Networks, Prentice Hall, Chapter 4, pp 208
21. Ilonen, J., Kamarainen, J.-K., Lampinen, J. (2003), Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters 7, 1, pp 93-105
22. Juang CF (2004) , A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design, IEEE Transactions on Systems, Man and Cybernetics, Part B, 34(2) pp 997-1006.
23. Lasarzyck CWG, Dittrich P, Banzhaf W (2004), Dynamic subset selection based on a Fitness Case Topology, Evolutionary Computation, pp 223-242.
24. Lu B L, Ito K, Kita H, Nishikawa Y (1995), Parallel and modular multi-sieving neural network architecture for constructive learning. In Proceedings of the 4th International Conference on Artificial Neural Networks. 409, pp 92-97.
25. M. A. Wong and T. Lane (1983). A kth nearest neighbour clustering procedure. Journal of the Royal Statistical Society (B), 45(3), pp362-368.

26. M. Lehtokangas (1999), Modelling with constructive backpropagation, Neural Networks, 12, pp 707-716
27. Mahfoud, S.W. and Goldberg D E (1992). A genetic algorithm for parallel simulated annealing, Parallel Problem Solving From Nature 2, Edited by Männer, R. and B. Manderick , pp. 301-310.
28. Marchesi M, Orlandi O, Piazza F, Pignotti G, Uncini A (1990), Dynamic topology neural network, Parallel Architectures and Neural Networks III, Edited by Caianiello E R, World Scientific, pp. 107-. 115
29. National Institute of Standards and Technology (2000), Statistical reference datasets, http://www.itl.nist.gov/div898/strd/index.html
30. Pendharkar P C, Rodger J A (1999), An empirical study of non-binary genetic algorithm based neural approaches for classification, Proceeding of the 20th international conference on Information Systems, pp155-165.
31. Quinlan J R (1986), Introduction of Decision trees, Machine learning, 1, pp81-106.
32. Quinlan J R (1996), Improved Use of Continuous Attributes in C4.5, Journal of Artificial Intelligence Research, 4, pp77-90
33. Riolo R, Worzel B (2003), Genetic programming theory and practice, Boston: Kluwer Academic
34. Rovithakis, G.A, Chalkiadakis. I, Zeravakis, M.E, (2004) High – Order Neural Network Structure Selection for Function Approximation Applications using Genetic Algorithms, IEEE Transactions on Systems, Man and Cybernetics, 34(1), pp 150-158.
35. Rumelhart, D. Hinton, G. and Williams, R (1986). Learning internal representations by error propagation., Parallel Distributed Processing, edited by D. Rumelhart and J. McClelland, 1 MIT Press, pp318-362
36. Stoianov I, Stowe L and Nerbonne J (2001), Connectionist learning to read aloud and correlate to human data, 21st Annual meeting of the cognitive science society, pp 706-711
37. Vasconcelos J.A., Ramirez J.A., Takahashi RHC, and R.R. Saldanha (2001), Improvements in Genetic algorithms, IEEE Transactions on Magnetics, 37 (5), pp3414-3417.
38. Yasunaga, M.; Yoshida, E.; Yoshihara, I (1999), Parallel back-propagation using genetic algorithm: real-time BP learning on the massively parallel computer CP-PACS, International Joint Conference on Neural Networks, 6, pp4175-4180.
39. Yao X (1993), A review of evolutionary artificial neural networks, International Journal of Intelligent Systems, 8(4), 539-567
40. Zhang BT, Cho DY (1998), Genetic Programming with Active Data selection, Simulated Evolution and Learning, pp 146-153

## APPENDIX

In order to prevent over- or under-training of a neural network, a validation set of data is used to terminate the network training. The total training error of a neural network is defined based on the difference between the desired and the obtained outputs of the network as shown below:

$$E_{train} = \frac{1}{2}\sum_{p}\sum_{k}(D_{pk(train)} - O_{pk(train)})^2$$

The network's validation error is therefore $E_{val}(n) = \frac{1}{2}\sum_{p}\sum_{k}(D_{pk(val)}(n) - O_{pk(val)}(n))^2$

The total network error given by $E_{tot} = E_{train} + E_{val}$

The value $E_{opt}(n)$ is defined to be the lowest validation set error obtained in epochs up to epoch n, i.e, $E_{opt}(n) = \min_{n' \le n} E_{tot}(n')$

The *generalization loss* at epoch n is defined as the relative increase of the total error over the minimum so far.

$$GL(n) = \left(\frac{E_{tot}(n)}{E_{opt}(n)} - 1\right)$$

The validation set termination criterion is set such that a high generalization loss will result in termination of the training. This method is specifically designed to reduce the possibility of loss of generalization accuracy due to over-training.