

# Comparing Different Optimisation Strategies for Nonlinear Mapping and Mapping Large Datasets using Neural Networks

Auralia I. Edwards and Andries P. Engelbrecht  
Department of Computer Science  
University of Pretoria  
South Africa  
aedwards@cs.up.ac.za and engel@cs.up.ac.za  
+27833268253

**Abstract**—Reducing the dimensionality of high-dimensional data allows easier visualisation of data, facilitating more efficient extraction of knowledge. Nonlinear mapping methods transform data existing in high-dimensional space into a lower-dimensional space such that the topological characteristics of the high-dimensional data are preserved. Recent work [6] [4] proposed a particle swarm optimisation algorithm to perform nonlinear mapping. Experimental results show a comparison between a number of optimisation algorithms in performing nonlinear mapping. Nonlinear mapping methods were designed to perform mapping on datasets that contain a small number of data points. To map large datasets, such as fault management systems in telecommunication networks, a different strategy is required. A proposed method that uses a Neural Network to perform nonlinear mapping on larger datasets is discussed.

## I. INTRODUCTION

Mapping high-dimensional data to a lower-dimensional space, such that the dissimilarities between data points in the lower dimension correspond to the dissimilarities in the higher space, enables the user to interpret the underlying structure of the data more easily.

Current nonlinear mapping methods are computationally intensive and have not always been renowned for obtaining an acceptable lower-dimensional configuration within a small number of iterations. Using different strategies could enable a lower-dimensional configuration to be found in fewer iterations.

Previous work [6] [4] has illustrated that the Particle Swarm Optimisation (PSO) algorithm developed by Kennedy and Eberhart [12] is an adequate algorithm to map high-dimensional datasets to a lower-dimensional space efficiently. Each particle in the swarm represented a possible lower-dimensional mapping of the high-dimensional space, where the objective of the particle was to minimise the nonlinear mapping's error function. A PSO approach obtained information concerning the search space from the swarm, thus directing the search towards optimal values.

Sincere thanks for the necessary funds from the Centre of Excellence for making this research possible.

There are two objectives to this paper. The first is to compare different optimisation algorithms that minimise the error criterion of a nonlinear mapping method. Current nonlinear mapping methods were not designed to map large datasets. The second objective is to give a proposed algorithm to map datasets containing a large number of vectors to a smaller dimensional space.

The remainder of the paper is organised as follows: Section II provides the background and problem definition of nonlinear mapping. Section III briefly discusses the optimisation algorithms. Section IV describes the algorithm and various datasets used for the experiments. Experimental results are presented and discussed in section V. Extracting useful information to understand data has been done in variety of ways, for example discovering rules or clusters of data or by projecting data to lower-dimensional spaces in which visualization is easier. Data mining algorithms have been applied to variety of applications, e.g. wireless telecommunication industry. A brief look at a proposed solution for mapping large datasets to a smaller dimension is described in section VI. Finally, section VII concludes this paper.

## II. NONLINEAR MAPPING

### A. Background

There are three aspects within nonlinear mapping that influence the performance of the mapping process:

- 1) The nonlinear mapping method [3], [16], [15], [19], which refer to the optimisation criteria or objective functions.
- 2) The dissimilarity measurement, which calculates the difference in the topological relationships between the vectors, for example, Euclidean distance.
- 3) The optimisation algorithm that finds a lower-dimensional configuration of points, by updating these points, so that the underlying structure of the dataset is preserved.

The focus of this paper is on 3). Gradient descent, genetic algorithms [26], PSO and PSO modifications such as the

guaranteed convergence PSO (GCPSO) and “mutating” PSO [8] are compared.

### B. Problem Definition

Nonlinear mapping is a projection from a high-dimensional input space to a lower-dimensional output space, where the distance between vectors in the input space is preserved. To determine the accuracy of a projection from the higher dimensional space to the lower dimensional space, an error is calculated, known as the objective function. The objective function quantifies the difference between these two spaces.

Two datasets represent two matrices containing the same number of vectors, namely  $n$ . Let  $m$  be the dimension of the high-dimensional dataset, and  $d$  the dimension of the lower-dimensional dataset, where  $m \gg d$ . For the  $m \times n$  matrix, each row is represented as an  $m$ -dimensional vector.

A lower left triangular matrix is calculated to represent a topological distance between these two matrices. The distance matrix  $\mathbf{M}$  is represented as:

For all  $i, j$  in  $\mathbf{M}$  where  $i > j$ :

$$m_{i,j} = \text{dist}(\mathbf{v}_i, \mathbf{w}_j) \quad (1)$$

$m_{i,j}$  denotes the distance between two vectors at the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column respectively, and  $\text{dist}(\mathbf{v}_i, \mathbf{w}_j)$  denotes the Euclidean distance

## III. OPTIMISATION ALGORITHMS

A number of optimisation methods have been applied to nonlinear mapping [2]. Each use a different method to obtain the optimal solution for the problem. This section describes the algorithms enabling the user to differentiate between them.

### A. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is an evolutionary optimisation technique first introduced in 1995 by Kennedy and Eberhart [12]. PSO method consists of a swarm of particles that simulates animal behaviour, such as a swarm of bees or school of fish, in that particles interact and help each other to find an optimal solution within the search space. In nonlinear mapping, each particle represents a candidate solution in the lower dimensional search space [18].

The particles’ positions are randomly initialised in the search space. The solutions found by the particles are evaluated by a fitness function that has to be optimised. By utilising social behaviour, the particles find the global best solution by simply adjusting the position of each individual. The position is adjusted by dynamically altering the velocity of each particle, according to its own flying experience and the flying experience of the other particles in the search space. The  $i^{\text{th}}$  particle’s new position is calculated as [22]:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \quad (2)$$

where  $\mathbf{v}_i(t)$  represents the particle’s velocity and  $\mathbf{x}_i(t)$  the current position. If  $d$  is the dimension index, the velocity is updated according to [22]:

$$\begin{aligned} v_{i,d}(t+1) = & wv_{i,d}(t) + \\ & c_1r_{1,d}(t)(\mathbf{y}_{i,d}(t) - x_{i,d}(t)) + \\ & c_2r_{2,d}(t)(\hat{\mathbf{y}}_d(t) - x_{i,d}(t)) \end{aligned} \quad (3)$$

where  $w$  is the inertia weight between values 0 and 1, which determines how much of the particle’s previous velocity is preserved,  $c_1$  and  $c_2$  are acceleration constants,  $r_1$  and  $r_2$  are random variables sampled from  $U(0, 1)$ ,  $\mathbf{y}_i$  is the personal best position found by the  $i^{\text{th}}$  particle and  $\hat{\mathbf{y}}$  is the best position found by the entire swarm thus far (assuming *gbest* PSO). The second term, known as the cognitive component, models the particle’s personal knowledge about the search space. The third term is the social component since information is obtained by interacting with other particles within the swarm.

Neighbourhood structures, represented by the social component of equation (3), determine the scope of information exchange between particles. Detailed information concerning the different topologies namely, Star, Ring and Von Neumann can be found in [6], [7], [13], [4], [5].

### B. The Guaranteed Convergence PSO (GCPSO)

The PSO has been known to suffer from premature convergence, where the particles cannot escape local minima. The GCPSO, introduced by Van den Bergh [22], addressed this problem. A new position update equation for the global best particle is given by

$$x_{i,d}(t+1) = wv_{i,d}(t) + \hat{y}_d(t) + p(t)(1 - 2r_2(t)) \quad (4)$$

which is achieved by modifying the velocity update equation for the global best particle, so that

$$v_{i,d}(t+1) = wv_{i,d}(t) - x_{i,d}(t) + \hat{y}_d(t) + p(t)(1 - 2r_{2,d}(t)) \quad (5)$$

where  $\mathbf{r}_d$  is a sequence of uniform random numbers sampled from  $U(-1, 1)$  and  $p(t)$  is a scaling factor. More details information describing these equations can be found in [5].

### C. “Mutating” PSO

As the GCPSO was designed to improve the PSO, so was the “mutating” PSO. One of the major concerns with standard PSOs is the lack of diversity that occurs once particles start to converge to a solution. The “mutating” PSO incorporates a mutation operator that effects the particle position update, possibly allowing the particle to escape local optima.

The “mutating” PSO employs mutation, a concept originally introduced in genetic algorithms [26], to maintain diversity allowing the search space to be fully explored leading to potentially better solutions. If the mutation rate is too large, then the optimisation process finds it difficult to converge. On the contrary, small mutation values may lead to premature convergence. A good approach would be to have

an initial large mutation rate that decreases with increase in number of iterations.

The *gbest* and *lbest* are two hybrid PSO algorithms, developed by Esquivel and Coello Coello [8], that both incorporate a mutation operator (The pseudo-code for both algorithms are given in [8]).

The mutation operator effects the position update of the particle.

1) *Mutation operator*: The nonuniform mutation operator that was introduced by Michalewicz [8], [26] is used within the “mutating” PSO algorithms.

Another aspect that affects the particle position is the random value  $r$  discussed in the above equation. The random value is obtained by using the Cauchy distribution <sup>1</sup>. The Cauchy distribution has replaced the Gaussian distribution in many practical applications [25], [24].

#### D. Real-Valued Genetic Algorithm (RGA)

John Holland [9] introduced Genetic Algorithms, that consist of a population of potential solutions that are evolved through generations by selection and crossover. A GA differs from a PSO in that the solutions within the GA compete against one another (“Survival of the fittest”) as opposed to particles working together to find the best solution.

A real-valued Genetic Algorithm (RGA) [2], each individual in the population is a vector (similar to the particle in the PSO) representing the potential solution of the lower-dimensional space.

A RGA differs from the GA in the implementation of the crossover and mutation operators.

1) *Crossover*: For the first few generations, the individuals within the search space are sparsely distributed. Towards the end of the process, a different crossover scheme was used to enhance the searching efficiency. Information concerning the implementation of these schemes are described in [2].

2) *Mutation*: Mutation greatly influences the GA algorithm, since it allows the individuals to escape local minima. The selection procedure randomly selects individuals of the current population for development of the next generation. The crossover procedure creates two new individuals (children) by combining the selected individuals (parents) using a crossover operator. The mutation procedure randomly modifies the genes of an individual, allowing diversity to remain within the population. One of the biggest problems concerning GAs is that, given the common genetic operators, finding the global optimum within a reasonable time frame is difficult. The RGA introduced an *orientated mutation* [2] to reduce computational time but was prone to premature convergence. To resolve this, another operator, the *immigration operator* was also implemented. This operator, maintained diversity within the population, by selecting new individuals from a different region in the search space to replace old individuals.

<sup>1</sup><http://www.answers.com/topic/cauchy-distribution>

## IV. EXPERIMENTAL PROCEDURE

The objective of this paper is to investigate the performance of PSO compared to other optimisation algorithms to solve the nonlinear mapping problems. The scope of this paper does not include comparison of different nonlinear mapping methods.

Section IV-A provides an overview of the algorithms used in the experiments. Section IV-B describes the error functions used to determine particle fitness. Finally, the datasets used in the experiments are discussed in section IV-C.

### A. Algorithms

Each particle in the swarm represents an  $n \times d$  array in the solution space, where  $n$  is the number of vectors and  $d$  is the dimension of the lower-dimensional output space. All experiments were implemented using the Computational Intelligence Library (CILib) framework <sup>2</sup>. The GCP SO algorithm [6] is repeated 30 times so that the average, standard deviation and a 95 % confidence interval of mapping error values can be obtained.

The PSO parameter values, proposed by Van den Bergh [22] are listed in table I

TABLE I  
PSO PARAMETERS

Parameter	Value
$c_1 = c_2$	1.49618
inertia ( $w$ )	0.729844
$s_c = f_c$	5
size ( <i>lbest</i> )	2

Those used by Coello Coello (table II) have been used.

TABLE II  
“MUTATING” PSO PARAMETERS

Parameter	Value
$c_1 = c_2$	1.3
inertia ( $w$ )	0.3
model	gbest
size	20 - 50

In addition to the parameters listed in table above, 40 particles were used and each algorithm was executed for 5000 iterations. Further information on the implementation of RGA algorithm can be found in [2]. This paper did not optimise parameter values.

### B. Determining Fitness

The fitness function determines how well the representation in  $d$ -dimensional space approximates the topological structure of the given data in  $m$ -dimensional space. Four different error functions are used as fitness functions:

- *Kruskal’s Stress function*:

$$E_{Kruskal} = \sqrt{\frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n (\sigma_{ij} - d_{ij})^2}{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \sigma_{ij}^2}} \quad (6)$$

<sup>2</sup><http://cilib.sourceforge.net>

## V. RESULTS

- *Sammon's Error function:*

$$E_{Sammon} = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{(\sigma_{ij} - d_{ij})^2}{\sigma_{ij}}}{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \sigma_{ij}} \quad (7)$$

- *Curvilinear Component Analysis (CCA):*

$$E_{CCA} = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (d_{ij} - \sigma_{ij})^2 F(\sigma_{ij}) \quad (8)$$

- *Component Distance Analysis (CDA):*

CDA is a variant of CCA. The error function becomes:

$$E_{CDA} = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\eta_{ij} - \sigma_{ij})^2 F(\sigma_{ij}) \quad (9)$$

where  $d_{ij}$  is the Euclidean distance between vector  $i$  and vector  $j$  in the  $m$ -dimensional space and  $\sigma_{ij}$  is the distance between vector  $i$  and vector  $j$  in the  $d$ -dimensional space.

In equation (9),  $\eta_{ij}$  represents the curvilinear distance [4], or geodistic distance, between vector  $i$  and vector  $j$ . In equations (8) and (9),  $F$  is a decreasing Exponential function ( $e^{-x}$ ) [3].

### C. Datasets

Four datasets were used to investigate the performance of a number of optimisation algorithms. The datasets chosen in this analysis have been used in previous studies [6], [5] [2].

1) *Swiss Roll (Dataset A):* A 3-dimensional dataset containing 50 vectors was reduced to a dimension of 2. The best results obtained from the GCPSO [6] are compared to those obtained using the “mutating” PSO.

2) *Helix (Dataset B):* A 4-dimensional dataset containing 35 vectors was reduced to a dimension of 2. The dataset was used to compare the GCPSO, “mutating” GCPSO and the “mutating” PSO.

3) *Clusters (Dataset C):* A 4-dimensional dataset of two unimodal Gaussian clusters was simulated. Each cluster contains 30 samples from a normal population with mean vector  $\mathbf{m}_i$  and covariance matrix  $\sum_i$ . This dataset illustrated two independent clusters. It is important that the mapped dataset should also have two separated clusters. The RGA algorithm is compared to the GCPSO and “mutating” PSO using this dataset.

4) *Cataract Lenses (Dataset D):* Senine cataracts are the major cause of poor vision amongst elders. Knowing what the problems are lead to finding the cure of cataracts. This dataset consists of 30 samples of human lenses originally used in Xu [23]. A lense consists of 3 parts [23]: the central nucleus is surrounded by the cortex, and then by the capsular membrane. The dataset contains 3 classes. The first class is the Senine cataract Lense containing 15 vectors. The second class is the “Nuclei from the cataract Lenses” that has 5 vectors. The last 10 vectors belong to the third class, Normal Lense.

The RGA algorithm is compared to the GCPSO and “mutating” PSO.

The results obtained by different PSOs are discussed in section V-A. The PSO optimisation algorithm is compared to gradient descent and the RGA in sections V-B and V-C, respectively.

### A. Comparing PSOs and PSO modifications

Referring to table IV, the PSO that uses a mutating particle shows the best results. When the GCPSO used the mutating particle, poor solutions were found. The GCPSO with mutating particle had shown significant decrease in standard deviation. However, the “mutating” PSO gave the best average results. Without the mutation operator, the GCPSO obtained smaller standard deviations (refer to table IV).

Sammon's Mapping was used as the objective function since it showed good results from previous work [6]. The “mutating” PSO took longer to converge using the helix dataset. During initial iterations, the “mutating” PSO did not minimise the fitness criterion as well as the GCPSO (*gbest* topology). Only in the last few iterations, did the “mutating” PSO show a significant decrease resulting in an overall smaller error.

TABLE III

DATASET A MAPS FROM 3 TO 2 DIMENSIONS USING CDA AS THE OBJECTIVE FUNCTION. GCPSO AND “MUTATING” PSO ARE COMPARED [5].

Method	Results
GCPSO (Von Neumann)	$1.16 \times 10^{-7} \pm 1.15 \times 10^{-7}$
“Mutating” PSO ( <i>gbest</i> )	$1.8 \times 10^{-37} \pm 1.7 \times 10^{-37}$

Referring to table III, the GCPSO from previous work [6] illustrated *Von Neumann* to be the superior topology, since it produced the lowest errors and smallest confidence intervals. The “mutating” PSO, has previously shown to give better results than the GCPSO on classical multimodal optimisation problems [8]. The superior performance of the “mutating” PSO is clearly illustrated for the Swiss Roll dataset (refer to table III).

### B. Gradient descent and PSO

Referring to previous work [6], Gradient Descent needed a larger number of iterations to obtain the correct helix structure. During initial iterations, vectors within the gradient descent algorithm were closely positioned. On the other hand, particles within the PSO were scattered across the search space. The two algorithms differ in that the PSO explores global optimal solutions through exploiting the particle's and swarm's memory. Referring to table V, the result of 400.68 obtained for dataset D using gradient descent was unsatisfactory.

TABLE IV  
DATASET B MAPS FROM 4 TO 2 DIMENSIONS USING SAMMON'S MAPPING AS THE OBJECTIVE FUNCTION. PSO AND MODIFIED PSO ALGORITHMS ARE BEING COMPARED [5].

		Gbest	Lbest	Von Neumann
GCPSO	Ave	$1.32 \times 10^{-3}$	$1.291 \times 10^{-3}$	$1.173 \times 10^{-3}$
	Std Dev	$6.40 \times 10^{-5}$	$1.57 \times 10^{-4}$	$8.90 \times 10^{-5}$
GCPSO (Mutating Particle)	Ave	$5.575 \times 10^{-3}$	$2.316 \times 10^{-1}$	$5.893 \times 10^{-2}$
	Std Dev	$2.224 \times 10^{-2}$	$7.564 \times 10^{-2}$	$6.366 \times 10^{-3}$
PSO (Mutating Particle)	Ave	$9.32 \times 10^{-4}$	$9.91 \times 10^{-4}$	
	Std Dev	$1.01 \times 10^{-4}$	$6.00 \times 10^{-4}$	

### C. PSO and RGA

Dataset C and D are used to analyse the difference between the RGA, PSO and steepest gradient descent. Chen, Z. et al. [2] showed that the proposed RGA alleviated the problem of premature convergence and found acceptable lower dimensional-mappings with high efficiency.

In [2], it was shown that the two distinct clusters were not differentiable when using a steepest gradient algorithm or a standard GA. The RGA was able to separate the clusters. Similarly, the GCPSO showed a clear distinction between the two clusters ([6], [5]). The PSO successfully projected the clusters from a four-dimensional space to a two-dimensional space while preserving the original structure.

The GCPSO correctly mapped dataset D into 3 classes with Error of 0.006 Sammon's Mapping. The fitness results for the RGA and gradient descent, as summarised in table V were taken from [2]. The GCPSO and "mutating" PSO gave smaller fitness values than the RGA. The objective function that was minimised can be found in [2].

TABLE V  
DATASET D. COMPARING OPTIMISING ALGORITHMS USING THE OBJECTIVE FUNCTION FROM [2], [5]

Optimisation Algorithm	Smallest Fitness Error
Steepest Gradient Descent	400.68
RGA	248.13
GCPSO	89.13
"Mutating" PSO	48

## VI. MAPPING LARGE DATASETS

There are a number of domains where large volumes of data are stored in databases. An example would be a telecommunication network containing fault management data in the order  $10^9$ .

Mapping datasets such as the face, hands or fingerprint datasets, that contain thousands of data points, using current nonlinear mapping techniques is not feasible. To obtain an acceptable lower-dimensional configuration requires thousands of iterations. Nonlinear mapping methods were designed to perform mapping on datasets containing only a few data points, since they are computationally expensive.

A Neural Network (NN) [11] is a mathematical model representing a network of independent processing units called

"neurons" arranged in one or more layers. These artificial neurons are approximations of neurons found in the brain.

Every connection between neurons has a weight value associated with it that indicates the connected input neuron's "importance" to the processing neuron. A Neuron is essentially a pipeline of functions that combine multiple inputs and weights with a activation function, and then provides an output via an output function.

The action of the neurons are reliant on perceived input values and the continual refinement of the weight values associated with the interconnecting neurons defines the learning action. The NN adapts the weights of the connections between neurons so that the final output activations are correct. NNs are massively parallel, which make them efficient, robust and fault tolerant. NNs are used for real world applications such as data analysis, pattern recognition, financial modelling and control systems. One of the biggest advantage NN have over nonlinear mapping methods are that they can learn and generalise from a set of training data enabling them not only to be able to map larger datasets but to project unknown points [21].

One of the biggest drawback to the techniques that are studied is that new, previously unseen data cannot be projected. There have been a number of NN implementations to perform nonlinear mapping on large datasets [14][17]. One approach uses fuzzy clustering in conjunction with NN [1][10]. A similar approach that integrates *k-means* clustering, PSO mapping and NN is proposed (refer to algorithm 1). *K-means clustering* is performed on the large dataset to obtain a number of clusters. A *stratified sampling* technique selects a number of vectors from each of these clusters so that a small number of vectors can be mapped. The PSO maps these vectors to a lower-dimensional space. The NN is able to learn this mapping since the input and output vectors have been established. The NN is trained and tested enabling the rest of the dataset to be mapped.

Please note that this algorithm has been tested using Dataset C containing 2000 vectors. The results are not included in this paper's scope.

## VII. CONCLUSION AND FINAL REMARKS

It is important to know how each of the optimisation algorithms behave as this affects the number of iterations required to find a lower-dimensional configuration. For example, the

---

**Algorithm 1** Large Dataset Proposed Method

---

A dataset containing  $x$  number of vectors,  $x > 200$   
Perform *k-means* clustering [20] on the dataset  
Perform stratified sampling of each cluster  
Taking only a percentage of vectors in each cluster. Apply the PSO to perform nonlinear mapping on each of the clusters  
Train and test the NN on the input and output vectors of each cluster  
Obtain the lower-dimensional vectors of the rest of the dataset by passing the high-dimensional data through the NN

---

PSO algorithm returned a result in a shorter computational time when compared to the gradient descent algorithm for the helix dataset [6]. PSO differs from gradient descent in that it uses social behaviour to find optimal solutions.

The mutation operator was introduced to the PSO algorithm to generate the diversity that could possibly prevent problems such as getting trapped in local optima. For the problems in this paper, the *gbest* model (“mutating” PSO) had the best overall performance of all the algorithms compared. The mutation operator had shown an improvement on performance. The “mutating” PSO was successful in mapping the datasets to a lower dimension but took longer to converge, referring to previous work [6], [5]. A tradeoff between speed and accuracy arose. Another remark is that although the “mutating” PSO may have given very good results, higher standard deviations show that the results obtained (helix dataset) [5], were not consistent (table IV).

The PSO is algorithmically more simple than the RGA. The RGA typically requires three major operators: selection, crossover, and mutation.

Future work will include a discussion, results and conclusions on the algorithm for mapping large datasets using Neural Networks. The Self-Organising Map (SOM), which is a type of NN, has previously been used in analysis and monitoring of telecommunication systems. Applications include adaptive resource allocation in telecommunication networks. SOMs will be included in future work.

#### ACKNOWLEDGMENT

Developers of the Cilib framework are hereby acknowledged. This research was funded by Telkom SA.

#### REFERENCES

- [1] J.K. Agrafiotis, V.S. Lobanov, and DN Rossokhin. Nonlinear Mapping of Massive Data Sets by Fuzzy Clustering and Neural Networks. *Journal of Computational Chemistry*, 22(4):373–386, 2001.
- [2] Z. Chen, J. Jiang, Y. Li, and R. Yu. Nonlinear mapping using real-valued genetic algorithm. *Chemometrics and intelligent laboratory systems*, 1999.
- [3] Demartines and J. P. Hérault. Curvilinear Component Analysis: A Self-Organising Neural Network for Nonlinear Mapping of Data Sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, January 1997.
- [4] A.I. Edwards and A.P. Engelbrecht. Nonlinear mapping using particle swarm optimisation in security based applications. *Satnac 2005*, 2005.
- [5] A.I. Edwards and A.P. Engelbrecht. Comparing particle swarm optimisation and genetic algorithms for nonlinear mapping. *WCCI IEEE*, 2006.
- [6] A.I. Edwards, A.P. Engelbrecht, and N. Franken. Nonlinear mapping using particle swarm optimisation. *Computational Evolutionary Conference*, 2005.
- [7] A. P. Engelbrecht. *Computational Intelligence: An Introduction*, chapter 6, page 191. Wiley and Sons, 2002.
- [8] S.C. Esquivel and C.A. Coello Coello. On the use of particle swarm optimization with multimodal functions. *The 2003 Congress on Evolutionary Computation*, 2:1130–1136, Dec 2003.
- [9] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [10] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, page 264, September 1999.
- [11] M.I. Jordan. Neural networks. Technical report, Massachusetts Institute of Technology, 1996.
- [12] J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. *Proceedings of International Conference on Neural Networks*, pages 1942–1948, 1995.
- [13] J. Kennedy and R. Mendes. Population Structure and Particle Swarm performance. *In Proceedings of Congress of Evolutionary Computation, Hawaii USA*, 2002.
- [14] M.A. Kraaijveld, J. Mao, and A.K. Jain. A nonlinear projection method based on kohonen’s topology preserving maps. *IEEE Transactions on Neural Networks*, 6(3):548–559, 1995.
- [15] J.B. Kruskal. *Nonmetric multidimensional scaling: A numerical method*, chapter 29, pages 115–129. Psychometrika, 1964.
- [16] J.A. Lee, A. Lendasse, N. Donckers, and M. Verleysen. A Robust Nonlinear Projection Method. *European Symposium on Artificial Neural Networks Bruges (Belgium)*, 8:13–20, April 2000.
- [17] B. Lerner, H. Guterman, M. Aladjem, I. Dinstein, and Y. Romem. Feature extraction by neural network nonlinear mapping for pattern classification. *Proceedings of ICPR IEEE*, D:320–324, 1996.
- [18] E.S. Peer, F. van den Bergh, and A.P. Engelbrecht. Using Neighborhoods with the Guaranteed Convergence PSO. *Proceedings of the 2003 Swarm Intelligence Symposium*, pages 235–242, 24–26 April 2003.
- [19] J.W. Sammon. A Nonlinear Mapping algorithm for data structure analysis. *IEEE Trans. Computer*, 18(5):401–409, 1969.
- [20] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical report, University of Minnesota, 2000.
- [21] M.E. Tipping. *Topographic Mappings and Feed-Forward Neural Networks*. PhD thesis, University of Aston in Birmingham, 1996.
- [22] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Department of Computer Science, 2002.
- [23] L. Xu. Multivariate classification based on metal contents in human senile cataract lenses. *Analytica Chimica Acta*, pages 11–15, 1991.
- [24] J.M. Yang, J.T. Horng, and Kao. C.K. A genetic algorithm with adaptive mutations and family competition for training neural networks. *International Journal of Neural Systems*, 10:333–352, 2000.
- [25] X. Yao and Y. Liu. Fast evolutionary programming. *Proceeding on Fifth Annual Conference on Evolutionary Programming*, pages 451–460, 1996.
- [26] Z. Michalewicz. *Genetic Algorithms and Data structures = Evolution Programs*. pringer-Verlag, third edition, 1996.

**Auralia I. Edwards** received the Hons-B.Sc degree, with distinction, in Computer Science in 2004 and a B.Sc degree, with distinction, in Computer Science in 2003. She started her M.Sc degree in Computer Science at the University of Pretoria in 2005.

**Andries P. Engelbrecht** received the B.Sc, Hons-B.Sc, M.Sc and PhD in Computer Science at the University of Stellenbosch in 1990, 1992, 1994, 1999 respectively. He is currently a professor in Computer Science at the University of Pretoria.