Novel Algorithms for Multi-Objective Search and their application in Multi-Objective Evolutionary Neural Network Training.

Submitted by Jonathan Edward Fieldsend, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science, June 2003.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other university.

....

0.1 Abstract

In a wide variety of application areas there is a need to trade-off competing objectives to achieve a resolution to a specific problem. Where the interdependencies of these objectives are unknown (as is often the case), this involves the searching and storing of a set of potential problem solutions which, without objective preference knowledge, cannot be said to be any better or worse than other members of the set. Many studies over the last 18 years have investigated ways to improve the efficiency of the search process for these solutions; principally through the tools of evolutionary computation. This increased efficiency has been manifest in the discovery of the best set of solutions in a fewer number of function (problem) evaluations; the fewer the function evaluations, the lower the computational cost and the better the search process is judged to perform. A number of studies have focused on the additional computation complexity of some of the more advanced methods, however one major cause of realised run-time has been largely ignored. The current approach in the literature is to store potential solutions in a linear list. This means that a large proportion of an optimiser's time can be actually spent comparing stored solutions with new solutions, as opposed to function evaluation of solutions, or the search process itself. The first part of this work confronts this problem by developing new data structures for the representation of multi-dimensional points, which can be used in multi-objective search processes. These new data structures are shown to be significantly faster than linear lists and operational proofs are also derived that evinces this. The second part of the thesis is concerned with the benefits of these new data structures to multi-objective search beyond their application within a general framework - to their specific use in facilitating novel optimisation techniques. This is shown with the development and empirical validation of a multi-objective particle swarm optimisation model. The final part of the work is concerned with the development of a multi-objective evolutionary neural network framework. Until this point the technique of choice in this field has been the linear weighting method, whose shortcomings have been amply demonstrated in the general multi-objective optimisation field. This section therefore transfers the recent advances in multi-objective optimisation to a neural network training framework, and develops novel generalisation techniques to deal with the unique properties of multi-objective error minimisation that are not apparent in the uni-objective case. Empirical validation is provided in terms of test problems from the literature and an extensive financial forecasting application.

for my family

0.2 Acknowledgements

No work is that of a single person - but rather some amalgamation of that person and their experiences, here I would like to acknowledge those people who have in some part provided those experiences that have propelled me in different directions during my PhD research, helped me to think in different ways and enabled me to stay sane during the inevitable lows as well as highs.

From the academic community at the department I would like to thank my supervisors Sameer Singh and Zheng Yang as well as my Thesis chair Derek Partridge. I would also like to thank Richard Everson for our many discussions on multi-objective theory and generous collaboration in the development of the dominated and non-dominated trees.

I have been privileged to enjoy the company of a number of excellent PhD students during my time at Exeter, firstly Keir Bovis who was forced to sit next to me for 3 years, fostered my LINUX initiation, was free in the exchange of ideas, as well being persuasive of the benefit of the Imperial Free House as a solution to the most persistent problems. In addition I would like also thank Michelle Fisher, Tim Hodgson, Julia Wallace, Kevin Smith and Alex Schmolck - all of whose conversations over lunch have provided an interesting insight into life, the universe, and the best source of carbohydrates on campus.

I would also like to thank Morgan Adams, Demetra Arsalidou, Sarah Bidgood, Daniel Fraser, Rinske Goettsch, Keith Langmead, Carly Mays, Tom Milburn, Nick Murison, Brynmor Morris, Ming Peng, Christina Pössel, Marika Wedlock, James Wheeler and Maria Varikou for their friendship and support over the last few years.

I would finally like to thank Soon-Thiam Khu and Xin Yao for agreeing to examine my thesis.

Contents

List of Figures

List of Tables

List of Algorithms

0.3 Nomenclature

$\mathbf{c}$	Composite point.
D	Number of objective dimensions.
$\mathcal{E}$	Set of estimated Pareto optimal solutions.
F	Archive of non-dominated solutions maintained by a multi-objective algorithm.
g	Index of global best solution maintained by PSO in the uni-objective domain.
$\mathbf{L}$	(Hyper-) Set of best previous solutions found by particles in multi-objective PSO (local best).
L	Length of (non-) dominated tree.
M	Number of solutions in F.
n	Number of parameters in decision vector.
P	Set of best previous solutions found by particles in uni-objective PSO.
$\mathcal{P}$	The true Pareto set of solutions.
$\mathbf{q}$	Query point.
$\mathcal{T}$	Dominated tree.
$\mathbf{p}, \mathbf{u}, \mathbf{v}, \mathbf{x}$	Decision vectors (solutions/particles).
V	Set of particle velocities in PSO.
$\mathcal{V}$	Volume measure, comparing different $\mathcal{E}$.
$\mathcal{V}^{\mathcal{P}}$	Volume measure, comparing $\mathcal{P}$ and an $\mathcal{E}$.
X	Set of solutions.

0.4 Author Declaration

This Thesis includes works from number of papers published by the author (or submitted for publication) during the period of his research toward a PhD at the University of Exeter. The papers and the Chapters in which work from them appears are as follows:

- Chapters 4 & 5 include work from the study "*Using Unconstrained Elite Archives for Multi-Objective Optimisation*" in IEEE Transactions on Evolutionary Computation, 7(3), 2003, by J.E. Fieldsend, R.M. Everson and S. Singh,
- Chapter 7 includes work from the study "*A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence*" in proceedings of the UK Workshop on Computational Intelligence (UKCI'02) by J.E. Fieldsend and S. Singh.
- Chapter 8 includes work from the study "*On the selection of gbest, lbest and pbest individuals, the use of turbulence and the impact of inertia in multi-objective PSO*" (submitted to IEEE Transactions on Evolutionary Computation, July 2002) by J.E. Fieldsend and S. Singh.
- Chapter 11 includes work from the study "*Pareto Multi-Objective Non-Linear Regression Modelling to Aid CAPM analogous Forecasting*", in proceedings of the IEEE International Joint Conference on Neural Networks, part of the World Congress on Computational Intelligence, WCCI 2002 by J.E. Fieldsend and S. Singh.
- Chapter 11 also includes work from the study *"Pareto Evolutionary Neural Networks"* (submitted to IEEE Transactions on Neural Networks, June 2003), by J.E. Fieldsend and S. Singh.

Chapter 1

Introduction

1.1 Motivation

Frequently, when looking for a potential solution to a problem, one must trade-off a number of competing objectives. Under normal circumstances these may be binary trade-offs - a proper breakfast versus catching the earlier bus, or more continuous trade-offs; how much capital to save for a 'rainy day' versus how much to spend and enjoy now. These types of decisions are commonplace - and are arguably far more apparent than the simple single-objective problems a large proportion of optimisation academia are concerned with. The two examples previously given are quite easy to represent, *ceteris paribus*, the consumption of breakfast will directly lead to the missing of the bus, and the expenditure of capital on a nice new car will directly lead to the number in the bank account shrinking (or the red number rising!).

However in a number of domains the inter-relationships of a set of parameters which effect the outcome of a process are unknown - empirical observations being the only way to ascertain/approximate them. Two general techniques have received much interest in the later half of the 20th century in order to facilitate the resolution of these types of problem; evolutionary computation (EC) and neural networks (NNs), which both belong to a group of technologies commonly referred to as computational intelligence (CI). EC has gained in popularity due to its ability to search for global solutions in high dimensional parameter space, which may exhibit complex interactions in relation to their relationship to the evaluation of the parameter set. NNs in turn have received much attention as a 'universal approximator', that is the ability (given sufficient network size) to represent any deterministic function mapping $f(\mathbf{a}) \mapsto \mathbf{b}$, where $\mathbf{a}$ is a scalar or vector of inputs and $\mathbf{b}$ is a scalar or vector of desired outputs. More recently, research has been undertaken on combining the two approaches, using the methods of EC to optimise (train) NN approximators (evolutionary neural networks (ENNs)), in the situation where the function approximation is evaluated in terms of a single error term (Alba *et al.* , 1993; Angeline *et al.* , 1994; Baluja, 1996; Belfore & Arkadan, 1997; van den Bergh & Engelbrecht, 2001b; van den Bergh & Engelbrecht, 2000; van den Bergh, 1999; Berlanga *et al.* , 1999; Chen & Weigand, 1992; Coit & Smith, 1996; Conradie *et al.* , 2002b; Conradie *et al.* , 2002a; de Garis, 1991; Dominic *et al.* , 1992; Dracopoulos & Jones, 1995; Engelbrecht & Ismail, 1999; Fang & Xi, 1997; Fogel *et al.* , 1995; Greenwoood, 1997; Greenwood, 1997; Hansen & Meservy, 1996; Huang & Huang, 1997; Hung & Adeli, 1994; Ishigami *et al.* , 1995; Ismail & Engelbrecht, 2000; Ismail & Engelbrecht, 1999; Janson & Frenzel, 1992; Janson & Frenzel, 1993; Koza & Rice, 1992; Kupinski & Anastasio, 1999; Lee & Jang, 1996; Liu & Yao, 1998; Lopez *et al.* , 1999; McDonnell & Waagen, 1994; Maniezzo, 1994; Maricic, 1991; Marin & Sandoval, 1993; Marti, 1992; Merelo *et al.* , 1993; Olmez, 1997; O'Neil, 1992; Park *et al.* , 1995; Porto *et al.* , 1995; Prados, 1992; Saravanan & Fogel, 1998; Schaffer *et al.* , 1990; Spofford & Hintz, 1991; Srinivas & Patnaik, 1991; Topchy *et al.* , 1997; Vico & Sandoval, 1991; White, 1993; Whitehead & Choate, 1996; Wieland, 1992; Yao *et al.* , 1996; Yao & Liu, 1997; Yao & Liu, 1998; Yao, 1999; Zaus & Roland, 1991; Zhang & Shoa, 2001; Zhang & Veenker, 1991; Zitar & Hassoun, 1995).

The vast majority of EC and NN applications are concerned with problems formulated such that they have a single objective. However, as alluded to previously, many (if not the vast majority) of situations encountered in real life involve the trading-off of one objective against another (or more). This is also true of processes in the design of a product or the approximation of a functional relationship between signals where EC and NNs have their central applications. For instance the manufacturer of a product may wish it to be as cheap to produce as possible, but also of high performance. In the approximation of a functional relationship there may be more than one competing measurement that the practitioner is interested in minimising (for example mean versus maximum Euclidean error (Lo & Bassu, 2002a)).

Methods to resolve these types of problem have been developed since 1985 in the EC domain, under the guise of multi-objective evolutionary algorithms (MOEAs) which return a set of solutions describing the trade-off front (Borges & Hansen, 1998; Coello, 1999; Deb, 1999; Deb *et al.* , 2000; Deb *et al.* , 2001; Deb, 2001; Ehrgott & Gandibleux, 2000; Everson *et al.* , 2002; Fieldsend & Singh, 2002b; Fieldsend *et al.* , 2003; Fonseca & Fleming, 1995; Fonseca & Fleming, 1993; Hajela & Lin, 1992; Hanne, 2000; Hanne, 1999; Horn *et al.* , 1994; Knowles & Corne, 2000; Knowles & Corne, 1999; Kupinski & Anastasio, 1999; Laumanns *et al.* , 2000; Laumanns *et al.* , 2001; Zitzler *et al.* , 2001; Mostaghim *et al.* , 2001; Mostaghim *et al.* , 2002; Murata & Ishibuchi, 1995; Parks & Miller, 1998; Ray & Liew, 2002; Schaffer, 1985; Srinivas & Deb, 1995; Veldhuizen & Lamont, 2000a; Veldhuizen & Lamont, 2000b; Zitzler *et al.* , 2000; Zitzler & Thiele, 1999; Zitzler, 1999). However traditional 'gradient descent' techniques are still the training technique of choice for multi-objective neural networks (MONNs) which return a single model.

1.1.1 Problems in evolutionary multi-objective search

The classical approach to resolving the multi-objective optimisation problem has been through first defining a *relative preference vector* (Deb, (2001)), creating a composite function, and finding a single trade-off optimal solution through the use of a single -objective optimiser. However this approach is very sensitive to the allocation of the preference vector, slight variations of values will in all probability lead to different solutions being returned. Unfortunately, as the preference vector is generated by a mixture of qualitative as well as quantitative information, which may be difficult to define, the classical approach has the potential of being highly subjective. As such this thesis will focus on more recent advances in the field, that of evolutionary multi-objective optimisation. The vast majority of these multi-objective optimisation methods assume that the interdependencies of the objectives being optimised are unknown. As such, although a preference may be known with regard to how the objectives should be ranked (or indeed weighted), optimisation and search is undertaken assuming that all objectives are equally important.[1] This therefore necessitates the searching and storing of a set of potential problem solutions which cannot be said to be any better or worse than other members of the set. Several studies over the last 18 years have investigated ways to improve the efficiency of the search process for this set of solutions; principally through the tools of EC. This increased efficiency has been manifest in the discovery of the best set of solutions in a fewer number of function (problem) evaluations - the fewer the function evaluations, the lower

[1]A full discussion as to why prior objective weightings cannot be used usefully within the optimisation process if the inter-relationships of objectives is unknown is provided later in Part III.

the computational cost and the better the search process is judged to perform. A number of studies have focused on the additional computation complexity of some of the more advanced methods (e.g. Knowles & Corne (2000) and Coello & Lechunga (2002)), however one major cause of computational cost has been largely ignored. The current approach in the literature is to store sets of potential solutions in a linear list; meaning a large proportion of an optimiser's time can be actually spent comparing stored solutions with new solutions, as opposed to function evaluation of solutions, or the search process itself. If a method where developed which could store these multi-dimensional points such that comparison of a point to the set was significantly faster than the linear list method (including of course the additional maintenance cost of this structure) then a significant influence on the computational cost of *all* multi-objective optimisation methods will have been reduced. In addition, it is conceivable that not only would this new data structure improve the efficiency of general multi-objective search, but that it may also enable the development of new multi-objective optimisation algorithms as well.

1.1.2 Problems with the current approach to multi-error neural network training

Currently the technique of choice in the field of multi-error NN training has been the linear weighting method propagated by gradient descent techniques (Moya & Hush, 1996; Wang & Wahl, 1997; Wen & Lee, 1998; Yao & Tan, 2000). However, this approach has a number of shortcomings that have been amply demonstrated in the general multi-objective optimisation field. These include an inability to find points on a non-convex surface (Deb, 2001), and (if the technique is being used to emulate a uni-objective method), a very restrictive requirement to know *a priori* the inter-relationships of the competing errors (Fieldsend & Singh, 2002b). Given the success of uni-objective EC techniques in NN training, and the preeminent use of EC methods in the general area of multi-objective optimisation it would seem reasonable to use the more advanced MOEA methods to train NNs in the multi-error case. However, for this to succeed the problem of generalisation in the multi-error[2] estimation domain needs to be addressed; the problem of over-fitting in multi-error NN training has been ignored in the literature. This has most probably been due to the recent emergence of the field itself. Of the published papers in the area that the author has found, there are only two

[2]In NN training, the terms multi-error and multi-objective are interchangeable in this work, as it is assumed that error terms are the objectives to be optimised.

that actually use a MOEA for multi-error NN training. The first study used a synthetic problem, Kupinski & Anastasio (1999), which due to its design may experience only very minimal over-fitting. The second one is by the author himself, Fieldsend & Singh (2002b), where the empirical application of a well known MOEA to a financial forecasting problem demonstrated that a large proportion of optimal set members with respect to the training data, may be suboptimal on the test data (compared with other set members).

Given the problems outlined above, the objectives of the thesis are now defined.

1.2 Objectives

The general objectives of this thesis are as follows;

- The development of novel data structures to enable the storing, searching and updating of a large number of *non-dominated* solutions to multi-objective problems, faster than the current linear order method used by the literature; that can be used by all kinds of multi-objective/multi-criterion optimisation techniques.
- The development of new multi-objective optimisation techniques, that were not viable previously, dependent upon the new data structures.
- To empirically test the validity of multi-objective evolution of NNs, where an unbounded set of networks is maintained that lie across the error trade-off front, as opposed to the single model (linear sum) training currently in the literature.
- To present empirical results which support each of these steps, both on well-known existing test functions/data and new application data.

1.3 Contributions

The contribution of this thesis are as follows:

- The theoretical development of two novel data structures (called dominated and non-dominated trees) with details of their operation, including time computational complexity, for the storing of multi-dimensional points. These new structures are not only empirically demonstrated to perform significantly faster than linear lists in the active archiving of *non-dominating* sets for their use by multi-objective optimisation methods, but in addition their use in the storing of general sets of multi-dimensional points is also highlighted.

- The generation of a multi-objective variant of the recent particle swarm optimisation algorithm which is based on the properties of the new data structures. It is demonstrated that this new algorithm more closely transfers the particle swarm heuristic to the multi-objective domain than its peers which have been developed by other researchers during the course of this thesis, and it is also empirically demonstrated to be superior to the best performing of these competing multi-objective particle swarm optimisers on a number of well known test problems.
- The formulation of a multi-objective evolutionary neural network (MOENN) architecture by the synthesis of approaches previously developed in MOEAs, the ENN, and the multi-objective neural network (MONN) literatures. This architecture not only encompasses sets of function approximators with heterogeneous error properties but heterogeneous parameterisation. New methods for the improvement of MOENN generalisation are developed and empirically validated on existing test data and new financial forecasting problems.

1.4 Thesis overview

The thesis is structured as follows. Next is a short chapter introducing the basic concepts and methods of NNs and EC for the reader who is unfamiliar with these techniques. Following this is the main body of the thesis, which covers the three topics described in Sections 1.2 and 1.3; the development of new data structures to improve the efficiency of general multi-objective optimisation; the application of these new data structures in the generation of new multi-objective optimisation methods; the synthesis of a general MOENN training framework and new generalisation methods to confront the unique problems of multi-objective training. Each of these topics are tackled in turn in Parts I, II and III of the thesis, and each of these parts is split into three chapters. The first chapter of each part provides a general review of the topic area and state-of-the-art techniques, the next chapter provides the theoretic framework of the novel research undertaken in the area as part of this thesis and the third chapter provides rigorous empirical support for the new techniques introduced. The thesis concludes with a discussion of the various results obtained, applicability of proposed methods, and possible future research directions in the area.

Chapter 2

Artificial Neural and Evolutionary Computation

In this chapter the basic design of the CI methods of NNs (in the form of multi-layer perceptions) and EC (in its two most popular forms - genetic algorithms and evolution strategies) will be described. This is necessary due to focus of this body of work on methods using these models. Readers who are already comfortable with the basic premise and operation of these approaches however can feel free to move directly to Part I.

2.1 The multi-layer perceptron

The original motivation behind artificial NNs was the observation that the human brain computes in a completely different manner than the standard digital computer (Haykin, 1999), which enables it to perform tasks such as pattern recognition and motor control far faster and more accurately than standard computation. This ability is derived from the fact that the human brain is complex, nonlinear and parallel, and has the additional ability to adapt to the environment it finds itself in (referred to as plasticity). Artificial NNs developed as a method to mimic these properties, and terms relating to NN design (neurons, synaptic weights) are taken from the biological description of the brain function. However, it is generally the case that NNs in popular use by researchers use only the concepts of parallelism, non-linearity and plasticity within a mathematical framework, and do not attempt to copy exactly the functions of the brain (which are still not fully understood).

The most popular NN model is the multi-layer perceptron (MLP) since the formalisation of the backpropagation (BP) leaning algorithm in the early 1980s. The basic design of an MLP is shown in Figure 1.

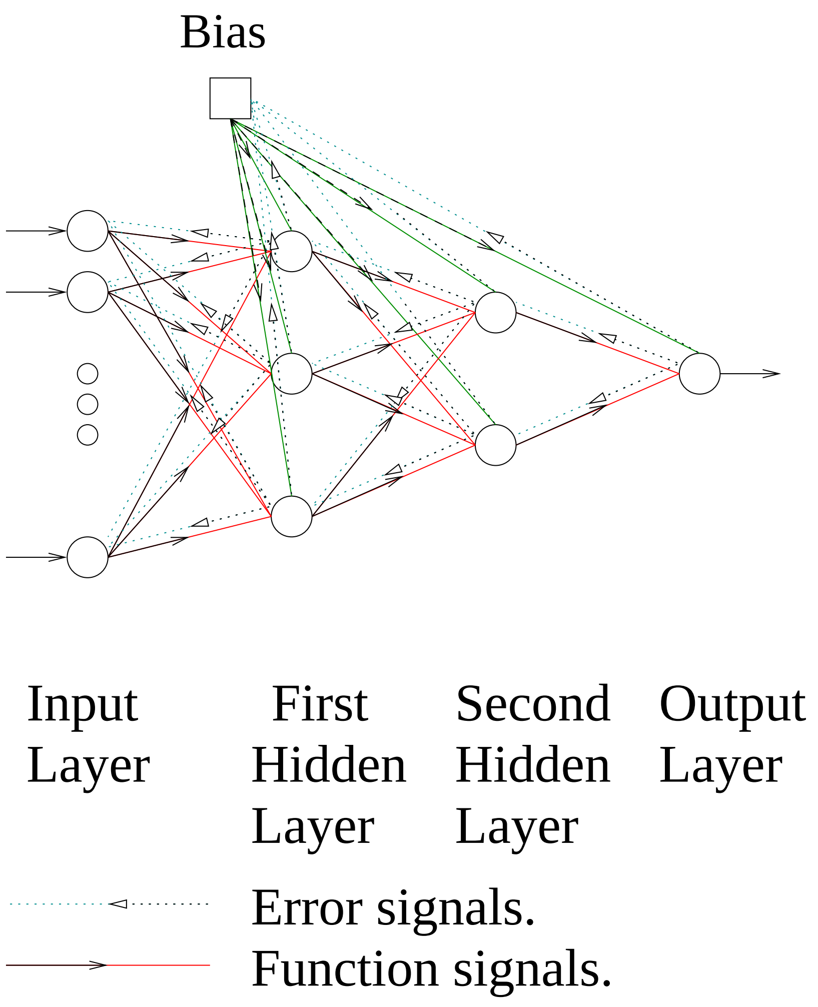


Figure 1: Generic multi-layer perceptron, showing the forward flow of the input signal (function signal) and the backward flow of the error signal.

The input signal of an MLP (or feature vector) is propagated through the network (neuron by neuron), and transformed during its passage by the combination of the synaptic weights and mathematical properties of the neurons, until on the final layer an output signal is generated. In the example shown in Figure 1 the network is defined as being fully connected, each neuron (or node) being connected to each other neuron in the layers directly preceding and proceeding it, and having a I:3:2:1 topological design. That is it has I input nodes, followed by two *hidden* layers, the first containing 3 nodes and the second 2 nodes, with a single output node. The two middle layers are referred to as *hidden* due the fact that the user does not commonly observe the inputs or outputs from these nodes (unlike the input layer where the feature vector is known and the output layer where the output is observed). The most common transfer function used in the MLP is the sigmoid function $\varphi()$. For the jth hidden node of a network with a vector of $\mathbf{z}$ inputs its logistic form is defined as:

$$\varphi(\mathbf{z}) = \frac{1}{1 + e^{-\left(B_j + \sum_{i=1}^{|\mathbf{z}|} w_{i,j} z_i\right)}} \tag{1}$$

where $w_{i,j}$ is the ith input weight between node j and the previous layer, z_i is the output of the ith node in the layer preceding node j and B_j is the weight of the bias input the jth node. The bias is similar to the, intercept term using in linear regression and has a fixed value for all patterns.

The adjustment of the synaptic weight parameter variables within an MLP are most commonly performed in a *supervised learning* manner using the fast backpropagation algorithm. Sequences of input and resultant outputs are collected from an undefined functional process $f(\mathbf{a}) \mapsto \mathbf{b}$. This set of *patterns* are then presented to the MLP in order for it to emulate the unknown function. The kth input pattern $\mathbf{a}(k)$ is fed through the network generating an output $\hat{\mathbf{b}}(k)$, an approximation of the desired output $\mathbf{b}(k)$ (illustrated with the arrows pointing to the right in Figure 1). The difference between the desired output $\mathbf{b}(k)$ and the actual output $\hat{\mathbf{b}}(k)$ is calculated (usually as the Euclidean distance between the vectors), and this error term, E, is then propagated back through the network, proportional to the partial derivative of the error at that node (illustrated with the dashed arrows pointing to the left in Figure 1). An in-depth discussion of the history and derivation of the back-propagation algorithm, though the calculus chain, can be found in Bishop (1998) and Haykin (1999), however as the main focus of this thesis is not on extending gradient descent methods themselves, it is sufficient to directly describe the *delta rule* which is at the heart of the backpropagation algorithm. In relation to weight between ith and the jth node it is defined as:

$$\Delta w_{i,j}(k) = -\eta \frac{\partial E(k)}{\partial w_{i,j}(k)} \tag{2}$$

where η is known as the learning rate of the system (used to dampen the effect of each pattern so that the change in network weights from one pattern to the next is not too drastic). By using a partial derivative of the sigmoidal transfer function, Equation 2 can be expressed as:

$$\Delta w_{i,j}(k) = \eta e_j(k)\varphi'(h_j(k))z_i(k) \tag{3}$$

where $e_j(k)$ is the differentiation of $E(k)$ with respect to the error on the jth node, $\varphi'(h_j(k))$ is the derivative of the activation function on the jth node and z_i is the output of the ith node (the local gradient). Each pattern in turn is presented to the MLP, with its weights adjusted using the delta rule at each iteration. The passing of an entire pattern set through the MLP is called a training epoch. MLPs are usually trained, epoch by epoch, until the observed average error of the function approximation reaches a plateau, or the error on a secondary data set whose patterns are not used during training (known as a validation set) begins to rise. The generalisation ability of the approximated function is then assessed on another set of collected data which the NN has not been trained on. The common use of a validation set is in order to prevent so called *over-fitting.* Because of the high function complexity that NNs can emulate, there is always a risk that the NN will simply map the input and output vectors directly without recourse to creating an internal representation of their generation process. Selecting an appropriate network size therefore becomes an important task (for which a potential solution is discussed in Chapter 11).

2.2 Basic principles of evolutionary algorithms

In order to facilitate multi-objective optimisation in the NN domain the techniques of EC (also known as evolutionary algorithms (EAs)) will be employed. The field of MOEAs is already well developed (Beale & Cook, 1978; Borges & Hansen, 1998; Coello, 1999; Deb, 1999; Deb *et al.* , 2000; Deb *et al.* , 2001; Ehrgott & Gandibleux, 2000; Everson *et al.* , 2002; Fieldsend & Singh, 2002b; Fieldsend *et al.* , 2003; Fonseca & Fleming, 1995; Fonseca & Fleming, 1993; Hajela & Lin, 1992; Hanne, 2000; Hanne, 1999; Horn *et al.* , 1994; Knowles & Corne, 2000; Knowles & Corne, 1999; Kupinski & Anastasio, 1999; Laumanns *et al.* , 2000; Laumanns *et al.* , 2001; Zitzler *et al.* , 2001; Mostaghim *et al.* , 2001; Mostaghim *et al.* , 2002; Murata & Ishibuchi, 1995; Parks & Miller, 1998; Ray & Liew, 2002; Schaffer, 1985; Srinivas & Deb, 1995; Veldhuizen & Lamont, 2000a; Veldhuizen &

Lamont, 2000b; Zitzler *et al.* , 2000; Zitzler & Thiele, 1999; Zitzler, 1999). The first practical study in the area was Schaffer (1985) and is discussed in Part I of this Thesis. Most recently developed algorithms in this field are capable multi-objective optimisers, however they are all based on EC techniques developed on uni-objective problems, the principle two of these EC methods will now be defined.

2.2.1 Evolution strategy (evolutionary programming)

In evolution strategy (ES), also known as evolutionary programming (EP), a problem is represented by a floating point string with n adjustable parameters, $\mathbf{x} = (x_1, x_2, \dots, x_n)$. These *solutions* or *decision vectors* are perturbed at each iteration (have their component values adjusted) before their *fitness* is evaluated by calculating the effect of the new parameters on the process being modelled. These decision vectors are iteratively adjusted (perturbed) in order to improve their performance, with each iteration known as a *generation*. The perturbation usually takes the form,

$$x_i := x_i + \gamma_i \cdot \Theta \tag{4}$$

where x_i is the i^{th} decision parameter of a vector, Θ is a random value drawn from some (pre-determined) distribution and γ is some multiplier. The process of an evolution strategy is designed in part to emulate the biological evolutionary process of adjustment and selection. Potential solutions are subtly altered, and the fitter of these evaluated solutions are more likely to pass into subsequent generations

A (μ, λ)-ES process is one in which μ decision vectors are available at the start of a generation (called parents), which are then perturbed to generate λ variants of themselves (called children or offspring). This set of λ children is then truncated/replicated to provide the μ parents of the following iteration. The process of selection for which children should form the set of parents in the next iteration is usually dependent on their evaluated fitness (the fitter being more probable to 'survive'). A $(\mu + \lambda)$-ES process denotes one where the parents compete with the children in the selection process for the formation of the next generation parent set.

2.2.2 Genetic algorithms

In genetic algorithms (GAs), the parameters of a model are converted into a representative 'chromosome', usually in the form of a binary string representation, with a number of bits (called genes) used to represent each parameter (a GA can also use the floating point representation used in ES, or indeed a mixture of bit and floating point representation). The number of bits define the resolution (or granularity O'Neil (1992)) of the resultant individual. In GAs a population of decision vectors is again maintained, and the population is adjusted from generation to generation using the three basic GA operators: reproduction (selection), crossover and mutation. These are defined below.

- Reproduction (as defined in Chen & O'Connell (1997)) is where a copy of an individual is chosen for direct insertion into the next generation, usually determined by some probability or ranking given the individual's (chromosome's) fitness.

Table 1: GA reproduction (elitist selection).

Current Generation	$\longrightarrow$	Next Generation
0110011	$\longrightarrow$	0110011
0011110	$\longrightarrow$	0011110

- Crossover takes place when two individuals are recombined to create one or more offspring by the cutting and exchanging of certain (random) alleles (gene sequences). The parents themselves are usually determined according to some fitness measure

Table 2: GA crossover (one point example).

Current Generation	$\longrightarrow$	Next Generation
$0110\underline{011}$	$\longrightarrow$	$0110\underline{110}$
$0011\underline{110}$	$\longrightarrow$	$0011\underline{011}$

- Mutation is chromosome alteration caused by the stochastic random bit flipping of an individual's genes

Table 3: GA mutation.

Current Generation	$\longrightarrow$	Next Generation
$01100\underline{1}1$	$\longrightarrow$	$01100\underline{0}1$
$0\underline{0}11110$	$\longrightarrow$	$0\underline{1}11110$

Again, as in ES, exact selection methods vary from study to study, however two of the most popular approaches to select individuals to form the next generation's search population are roulette wheel selection (i.e. (Huang & Huang, 1997)) and binary tournament selection (Zitzler *et al.* , 2000).

In roulette wheel selection the fitter members of the population are given a larger probability to be selected for direct insertion into the next generation or crossover than those that are less fit (usually proportional to their fitness). However as all individuals are given a probability, no one individual is denied the chance of reproduction no matter how unfit. As such this process can be represented as a roulette wheel, with different size segments determined by each individuals fitness.

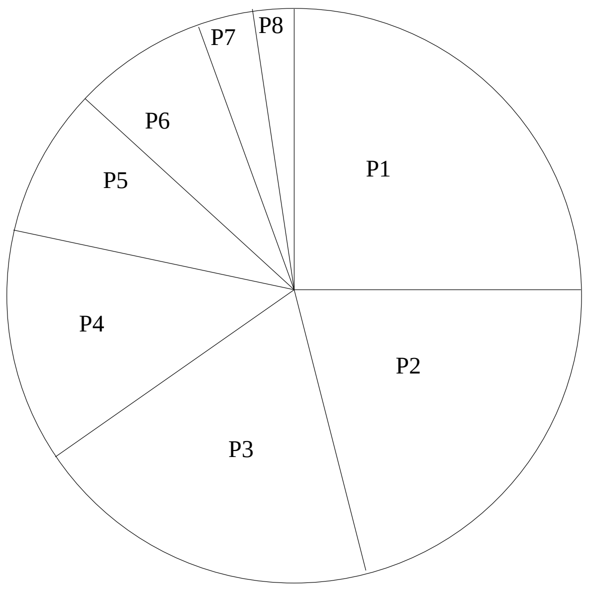


Figure 2: Roulette wheel selection.

This is illustrated in Figure 2, where a population of 8 individuals is shown. Chromosome 1 has $P1$ probability of being selected for a particular operator (with chromosome 8 having a far smaller probability of $P8$).

In binary tournament selection individual pairs are selected at random from the current population, with the fitter of the two selected.

The aims of the thesis have been broadly outlined in the introductory chapter, and in this chapter the reader has been provided with a basic overview of the main CI techniques in that will be employed in the remainder of the thesis. The following chapter will proceed with a more formal discussion of multi-objective optimisation, and introduces the first novel aspect of the thesis, new data structures to facilitate more efficient storage and maintenance of multi-objective solutions.

Part I

Improving Multi-Objective Optimisation

In this first part of the thesis, the general area of MOEAs is described, which has gained widespread use, both in academic and industrial circles, since the first major study in the area by Schaffer (1985). The area of active elite sets, which have drastically improved the performance of these algorithms, is highlighted as the most recent major advance in the area. It is also shown that this advance has highlighted a new problem in the domain, that of the efficient storage, query and update of this archive. When the cost of evaluating the fitness function of a solution is suitably high (as can be the case in NN evaluation, and many industrial tasks), this becomes of considerable importance.

It is shown in Part I that a consequence of restricting the number of solutions in the elite front can be shrinking, Zitzler *et al.* (2001), and oscillating/retreating estimated Pareto fronts (Hanne, 1999; Laumanns *et al.* , 2001; Everson *et al.* , 2002). These problems also occur in the strength Pareto evolutionary algorithm (Zitzler *et al.* , 2000; Zitzler & Thiele, 1999; Zitzler, 1999), Pareto archived evolutionary strategy, Knowles & Corne (2000; 1999), and other existing MOEAs which use a truncated elite archive. A remedy to this situation is simply to retain all non-dominated solutions found (as an active input to the continuing search process), as for example used by Parks & Miller (1998); however, this approach can be very time consuming (as any individual inserted into the elite archive must first be compared to every individual already present in the archive). It is important to note that in many studies an elite offline store of solutions is maintained which is unbounded, even when truncation takes place in the active population, and therefore the linear time update costs are still incurred (in addition to the time cost of truncation).

New data structures called *dominated* and *non-dominated trees* are introduced in the following chapters that permit faster searching of the elite archive, allowing even very large active elite sets to become feasible. It is shown that the use of the dominated and non-dominated trees in basic ES and GA MOEAs on a number of test sets leads to faster computation in comparison to maintaining an unconstrained archive as a linear list. Also, the estimated Pareto front discovered is significantly better than that found with a truncated elite archive.

Chapter 3

Evolutionary Multi-Objective Search

In this chapter the general area of MOEAs is described, some of the major theoretical advances in the area over the last 18 years highlighted, culminating in the unified model introduced by Laumanns *et al* .(2000) and the use of active elite archives of solutions. Problems with archive truncation are then illustrated, that show the common problem of maintaining unrestricted archives.

3.1 Multi-objective evolutionary algorithms: a brief overview

Frequently a number of competing objectives have to be traded against one another whilst seeking a viable solution to a given problem, often without any *a priori* knowledge of exactly how the objectives interact with each other. For instance, in product design a firm may wish to maximise the performance of an appliance whilst also trying to minimise its production cost. These two objectives cannot typically be met by a single solution, therefore, by adjusting the various design parameters, the firm may seek to discover what possible combinations of these two objectives are available given a set of constraints (for instance legal requirements and size limits of the product).

In Fonseca & Fleming (1993) for example, multi-objective optimisation is applied to four performance measures of a gas turbine. Similarly in Hajela & Lin (1992) different loads in trusses are the competing objectives to be minimised and in Parks & Miller (1998) different properties of a pressurised water reactor load pattern are optimised.

The curve (for two objectives) or surface (more than two objectives) that describes the optimal trade-off possibilities between objectives is known as the Pareto front (Pareto, 1927). A feasible solution lying on the Pareto front cannot improve any objective without degrading at least one of the others, and, given the constraints of the model, no solutions exist beyond the true Pareto front. The goal, therefore, of multi-objective algorithms is to locate the Pareto front of these *non-dominated* solutions.

Multi-objective evolutionary algorithms (MOEAs) represent a popular approach to solving these types of problem by using evolutionary search techniques. MOEAs have been in use for a considerable length of time now. Beale & Cook (1978) used a random search technique in an attempt to simultaneously minimise a number of objectives in an aircraft simulator. However, it was the work of Schaffer (1985), which recognised the need to return a set of solutions that has been widely quoted as the first MOEA study (Fonseca & Fleming, 1995; Veldhuizen & Lamont, 2000a; Zitzler *et al.* , 2000). The use of EAs as the tool of choice is due to such problems being typically complex, with both a large number of parameters to be adjusted and several objectives to be optimised. In addition, EAs which maintain a population of solutions are able to explore several parts of the Pareto front simultaneously.

3.1.1 Important issues in multi-objective optimisation

Pareto optimality

Most recent work on multi-objective algorithms (MOAs), EC based or otherwise, is formulated in terms of non-dominance and Pareto optimality, which is now briefly described.

The multi-objective optimisation problem seeks to simultaneously extremise D objectives:

$$y_i = f_i(\mathbf{x}), \qquad i = 1, \ldots, D \tag{5}$$

where each objective y_i depends upon a vector $\mathbf{x}$ of P parameters or decision variables. The parameters may also be subject to J constraints:

$$e_j(\mathbf{x}) \geq 0, \qquad j = 1, \ldots, J. \tag{6}$$

Without loss of generality it is assumed that the objectives are to be minimised so that the multi-objective optimisation problem may be expressed as:

$$\text{Minimise } \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_D(\mathbf{x})) \tag{7}$$

$$\text{subject to } \mathbf{e}(\mathbf{x}) = (e_1(\mathbf{x}), \ldots, e_J(\mathbf{x})) \geq 0 \tag{8}$$

where $\mathbf{x} = (x_1, \ldots, x_P)$ and $\mathbf{y} = (y_1, \ldots, y_D)$.

When faced with only a single objective, an optimal solution is the one that minimises the objective given the model constraints. However, when there is more than one objective to be minimised, solutions may exist for which performance on one objective cannot be improved without sacrificing performance on at least one other. Such solutions are said to be *Pareto optimal*, (Veldhuizen & Lamont, 2000a), named after the 19th century Engineer, Economist and Sociologist Vilfredo Pareto, whose work on the distribution of wealth led to the development of these trade-off surfaces (Pareto, 1927). The set of all Pareto optimal solutions are said to form the true Pareto front, which for the remainder of this thesis is denoted by $\mathcal{P}$.

The notion of *dominance* may be used to make Pareto optimality clearer. A decision vector $\mathbf{u}$ is said to *strictly dominate* another $\mathbf{v}$ (denoted $\mathbf{u} \prec \mathbf{v}$) iff

$$\begin{aligned} f_i(\mathbf{u}) &\leq f_i(\mathbf{v}) \quad \forall i = 1, \ldots, D \quad \text{and} \\ f_i(\mathbf{u}) &< f_i(\mathbf{v}) \quad \text{for at least one } i. \end{aligned} \tag{9}$$

Less stringently, $\mathbf{u}$ *weakly dominates* $\mathbf{v}$ (denoted $\mathbf{u} \preceq \mathbf{v}$) iff

$$f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \forall i = 1, \ldots, D. \tag{10}$$

A set of M decision vectors $W = \{W_1, W_2, \ldots, W_M\}$ is said to be a *non-dominated set* (an estimated Pareto front $\mathcal{E}$) if no member of the set is dominated by any other member:

$$W_i \not\prec W_j \quad \forall i, j = 1, \ldots, M. \tag{11}$$

Extent, resolution and density of the estimated Pareto set

There are a number of requirements of estimated Pareto fronts that researchers wish their algorithms to produce. These can be broadly described as high accuracy, representative extent, minimum resolution and and equal density.

The first concept, accuracy, is based on the difference between the true and estimated Pareto fronts (simply that the estimated solutions should be as close as possible to the true Pareto front). In Figure 3, the estimated front of Algorithm A is clearly more accurate than that of Algorithm B, however the comparison of A and C is more difficult to quantify, as in some members of A dominate members of C, but also the reverse is true.

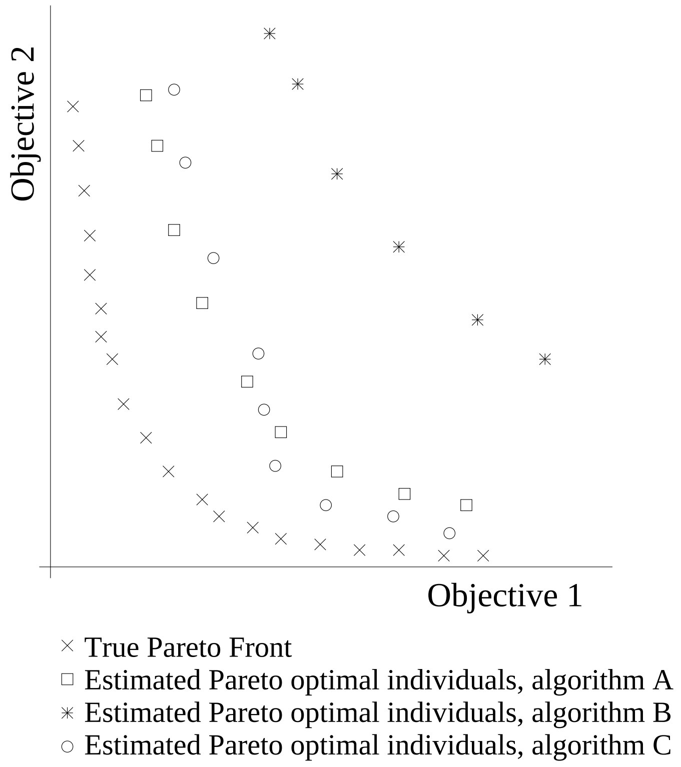


Figure 3: Illustration of the true Pareto front, and three estimates of it. The estimate of algorithm A being clearly more accurate than B, but the comparison of A and C is not as easy to quantify.

Ideally the Pareto solutions returned (or estimates of them) should lie across the entire surface of the true Pareto front, and not simply be concerned with a small subsection of it. Minimum resolution is a common requirement as in many applications the end user may wish that the difference between potential solutions is no bigger than a fixed value (of course, in discontinuous Pareto problems this requirement is not entirely realistic).

Much emphasis has been placed by researchers on the non-dominated solutions returned by the search algorithm being equally distributed (of even density), for instance Deb *et al.* (2001), however, it is arguable that this should be of concern only if the generating process results in evenly distributed solutions. In an actual application it may well be the case that the generating process produces an unbalanced Pareto front. This information itself may be very pertinent to the decision maker - by forcing MOEAs to misrepresent this fact by penalising any representation other than equal density they are providing a misleading representation of the generating process which may have negative repercussions for the final user of the information.

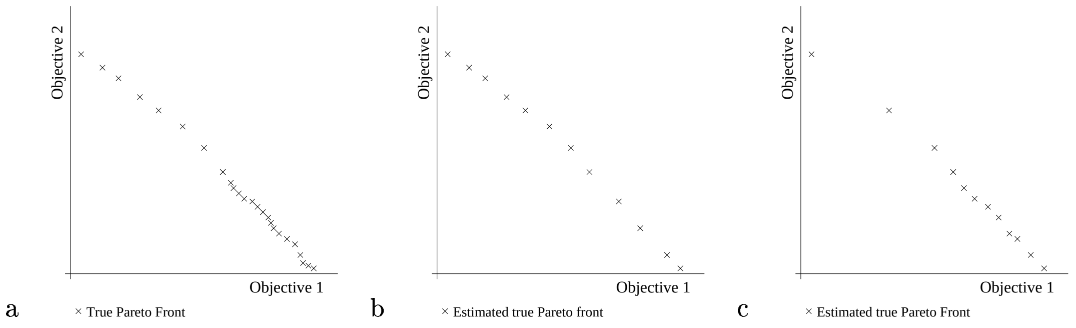


Figure 4: Comparing the density of estimated Pareto fronts. Illustration of an underlying true Pareto front (a), and its approximation using an MOEA that is designed to return equal density along the front (b) and one that does not (c).

An illustration of this is provided in Figure 4. Figure 4a shows the true Pareto front, with Figures 4b and 4c illustrating the returned sets of two MOEAs; the first one shows uniform density and the second one does not. By forcing the returned Pareto front estimate to be uniform, Figure 4b does not indicate to the end user of the density of solutions to the lower left of the front.

Given the different requirements outlined in this section, two recent measures to compare estimated Pareto fronts will now be discussed.

The problem of scalability

Another problem that has become a focus of concern is that of scalability - that is whether an algorithm performs well irrespective of the dimensionality of the task being optimised (in objective or parameter space). Recent work by Khare et al. (2002) has shown that the ordering of search algorithms in terms of the best estimate of the true Pareto front found, is not scale independent of the number of objectives D. With algorithms that performed well on lower (2-3) dimensions performing less well on higher (up to 8) objective dimensions. 'Better' was defined in terms of diversity of solutions, distance from the true Pareto front and algorithm run time.

Comparing estimated Pareto fronts

In recent years academics have become concerned with exactly how to compare the results of two or more competing algorithms (Hansen & Jaszkiwicz, 1998), i.e. how can one non-dominated set be 'proved' to be better or worse than another? Comparison of Pareto front estimates is difficult as there are several ways in which a front can be inferior or superior to another (as discussed earlier in terms of front extent and resolution). Indeed it is unlikely that any one single measure will be sufficient to encompass all desired information when evaluating the output of an MOEA. In this part two performance measures that to encapsulate different properties of competing non-dominated sets are used.

First, an alternative to the popular $\mathcal{C}$ metric (Zitzler *et al.* , 2000; Zitzler, 1999; Zitzler & Thiele, 1999) is discussed.

The $\mathcal{C}$ metric is defined as:

$$\mathcal{C}(A,B) = \frac{|\{\mathbf{b} \in B : \exists \, \mathbf{a} \in A \, , \, \mathbf{a} \preceq \mathbf{b}\}|}{|B|}, \tag{12}$$

where A and B are two sets of decision vectors and $A, B \subseteq X$.

The $\mathcal{C}$ metric measures the fraction of members of B which are (strictly or weakly) dominated by members of A. As such it measures the quality of A with respect to B. When $\mathcal{C}(A,B) = 1$, then all of the individuals in B are dominated by solutions in A; $\mathcal{C}(A,B) = 0$ represents the situation in which none of the individuals in B are dominated by any member of A.

It should be noted that $\mathcal{C}(A,B)$ is not technically a metric since $\mathcal{C}(A,A) \neq 0$ and $\mathcal{C}(A,B)$ is not symmetrical in its arguments and it does not satisfy the triangle inequality. Furthermore, $\mathcal{C}$ has the following undesirable property: suppose that W is a non-dominating set and $A \subseteq W$ and $B \subseteq W$, then $\mathcal{C}(A, B)$ can take on any value in $[0, 1]$.

In this work the following modified version of the $\mathcal{C}$ measure is used:

$$\tilde{\mathcal{C}}\,(A, B) = \frac{|\{\mathbf{b} \in B : \exists\, \mathbf{a} \in A\,,\ \mathbf{a} \prec \mathbf{b}\}|}{|B|} \tag{13}$$

Now $\tilde{\mathcal{C}}(A, A) = 0$ and, in addition, it measures two mutually non-dominating sets as equivalent, i.e. if $A \subseteq W$ and $B \subseteq W$ are each subsets of a non-dominating set W, then $\tilde{\mathcal{C}}(A, B) = 0$.

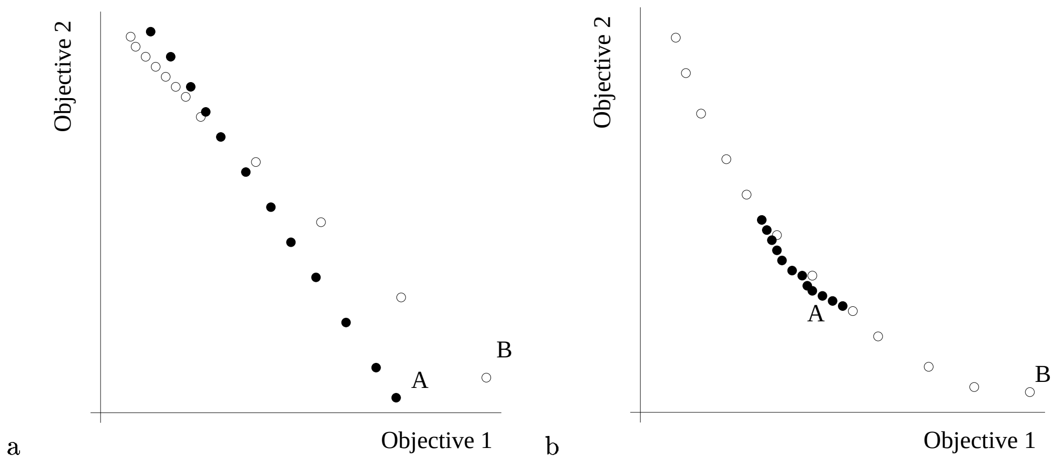


Figure 5: (a) Illustrates dense and evenly distributed estimated Pareto fronts. Front A dominates a larger extent of Front B, but both fronts dominate an equal number of each others' constituent members. (b) Illustrates estimated Pareto fronts of differing extents. Front B is of far greater extent than A, but will receive a lower $\mathcal{C}$ metric value.

Nevertheless, $\tilde{\mathcal{C}}$ and $\mathcal{C}$ fail to account for either the difference in the extent of the fronts being compared or the uniformity of the distribution of points along the front. For example, Figure 5a illustrates two fronts with a similar extent, but points describing A are uniformly distributed along the front, whereas those describing B are clustered in one region. However, $\mathcal{C}(A, B) = \mathcal{C}(B, A) = \tilde{\mathcal{C}}(A, B) = \tilde{\mathcal{C}}(B, A) = 4/12$, even though elements of A dominate elements of B along the majority of their extents. In Figure 5b although B has a much greater extent, A is ranked higher using $\mathcal{C}$ and $\tilde{\mathcal{C}}$: $\mathcal{C}(A, B) = \tilde{\mathcal{C}}(A, B) = 2/12$, whereas $\mathcal{C}(B, A) = \tilde{\mathcal{C}}(B, A) = 0/12$.

As some of the problems highlighted with the $\mathcal{C}$ and $\tilde{\mathcal{C}}$ measure demonstrate, there are a number of properties which are usually desired of estimated Pareto fronts. As discussed previously, these include that the 'distance' of the estimated Pareto front from the true Pareto front should be minimised, and that the extent of the estimated front should be maximised (i.e. a wide range of solutions in objective space should be returned). A measure which is designed to satisfy these requirements is now discussed.

The second measure used in this study is conceptually similar to the performance measure used in Laumanns *et al.* (2000), and it is a measure of the objective space volume that is dominated by one front but not the other. $\mathcal{V}(A, B)$ can be expressed as the fraction of the volume of the minimum hypercuboid containing both fronts that is strictly dominated by members of A but not dominated by members of B, and therefore it lies within the range $[0, 1]$. An illustration of this is provided in Figure 6 where two continuous fronts A and B have differing extents and also dominate each other in different regions of the objective space.

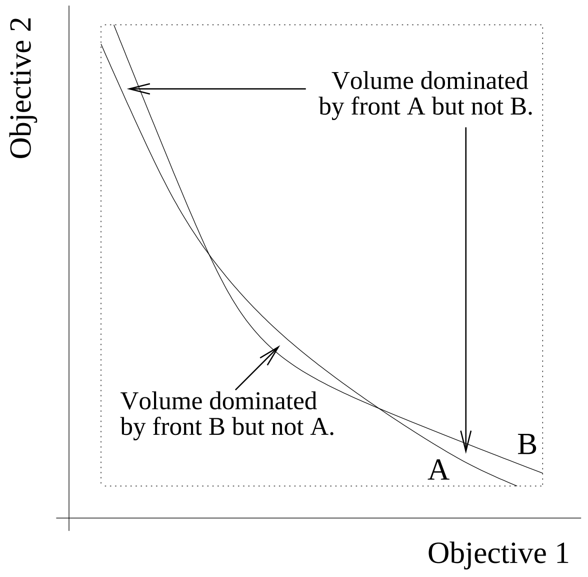

Objective 2
Volume dominated by front A but not B.
Volume dominated by front B but not A.
B
A
Objective 1

Figure 6: Two dimensional illustration of minimum surrounding hypercube volume dominated by two fronts (hypercuboid boundaries marked with dashed lines).

$\mathcal{V}(A, B)$ is defined in the following manner. For any set of D-dimensional vectors Y, let H_Y denote the smallest axis-parallel hypercuboid containing Y:

$$H_Y = \left\{ \mathbf{z} \in \mathbb{R}^D \ : \ a_i \leq z_i \leq b_i \text{ for some } \mathbf{a}, \mathbf{b} \in Y, \ i = 1, \ldots, D \right\} \tag{14}$$

Now $h_Y(\mathbf{y}) : H_Y \longmapsto [0,1]^D$ denotes the normalising scaling and translation that maps H_Y onto the unit hypercube. This transformation serves to remove the effects of scaling the objectives. Let

$$D_Y(A) = \left\{ \mathbf{z} \in [1,0]^D \ : \ \mathbf{z} \prec h_Y(\mathbf{a}) \text{ for some } \mathbf{a} \in A \right\} \tag{15}$$

be the set of points in the hypercube defined by Y which are dominated by the normalised front A. Then $\mathcal{V}(A,B)$ is defined as:

$$\mathcal{V}(A,B) = \lambda\left(D_{A \bigcup B}(A) \setminus D_{A \bigcup B}(B)\right) \tag{16}$$

where $\lambda(A)$ denotes the Lebesgue measure of A (Jones, 1993).

Despite this rather cumbersome description, $\mathcal{V}(A,B)$ and $\mathcal{V}(B,A)$ are easily calculated by Monte Carlo sampling of $H_{A \bigcup B}$ and counting the fraction of samples that are dominated exclusively by A or B. In this part of the thesis, 50000 samples are taken for Monte Carlo estimates. An example of the variance decrease of the estimate of $\mathcal{V}$ with increased Monte Carlo samples is shown in Figure 7. Here a Monte Carlo simulation of a hypercuboid surrounding the fronts defined by the standard strength Pareto evolutionary algorithm (SPEA) MOEA (Zitzler *et al.* (2000)) and the SPEA with an unconstrained archive on their second test function after 500 generations. 1 million random samples were generated, with difference in proportion of samples dominated by the unconstrained SPEA and standard SPEA shown in Figure 7a, and log variance of these differences shown in Figure 7b. The log variance smoothed by a finite impulse response filter is also shown.

The benefit of the volume measure $\mathcal{V}$ is that it will reward sets that are of greater extents when those extents are in front of the comparison set, but not when they are behind, it is not effected by the distribution of points across a front. In addition it also gives information regarding how far one set is (on average) in front of another.

Unfortunately the $\mathcal{V}$ measure, like the original $\mathcal{C}$ metric, has the property that, if W is a non-dominating set, and $A \subseteq W$ and $B \subseteq W$, both $\mathcal{V}(A,B)$ and $\mathcal{V}(B,A)$ may be positive.

3.1.2 MOEA: The major advances 1985-2000

Since the first study in the area by Schaffer (1985), which realised the need to return a set of solutions, there have been several MOEA models developed in the literature, and a number of very

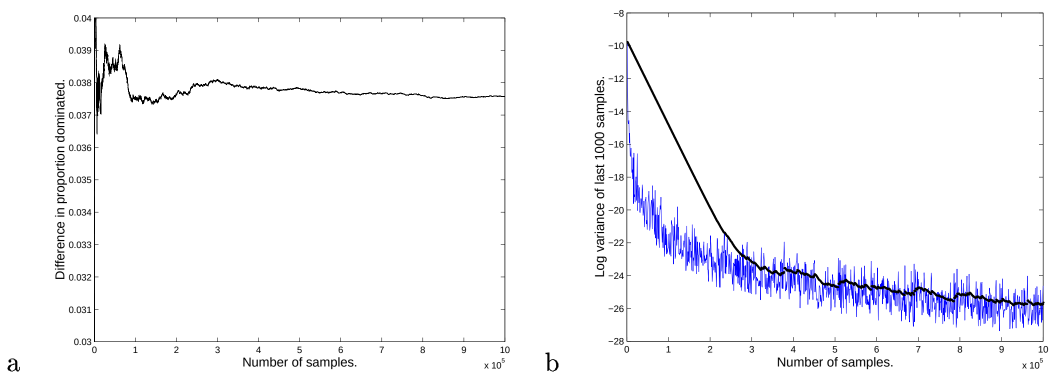


Figure 7: An example of the estimated $\mathcal{V}$ measure for two estimated Pareto fronts as the number of Monte Carlo samples increases.

good review studies that cover them (Coello, 1999; Fonseca & Fleming, 1995; Tan *et al.* , 2002; Veldhuizen & Lamont, 2000a). As such, this section does not intend to extensively describe them, but to list the important chronological advances in the field concluding with the unified model of Laumanns *et al.* (2000).

Returning set a of solutions (1985)

As stated earlier, the vector evaluated genetic algorithm (VEGA) by Schaffer (1985) is generally recognised as the first MOEA paper, as it recognised the need to return a *set* of solutions to a multi-objective problem, as opposed to a single individual. A single GA population was used with a fraction of the population chosen at each generation for recombination based on their fitness with respect to one of the evaluated objectives. The estimated Pareto solutions were removed at the end of the process. In future studies, however, it was noted that the approach tends to bias selection in favour of those individuals at the extreme (that solely minimise one objective) at the cost of solutions that lie on the trade-off front.

Adaptive Linear Weighting (1992)

Hajela and Lin's weighted-sum genetic algorithm (HLGA) resolved the problem of biased selection (Hajela & Lin, 1992). In their model an extra D parameters where maintained in a solution's decision vector. These parameters acted as the linear weighting coefficients described earlier in

Equation 5 and were perturbed at each generation (i.e. no two particular individuals were evaluated in the same manner), and solutions were discovered along the estimated Pareto front. However, this method cannot find solutions that lie on non-convex areas of Pareto fronts, and as such it is limited to problems that are strictly convex (which is commonly not known *a priori*).

Pareto Selection (1993)

It was in 1993 that studies began to seriously use Pareto dominance itself as a means for selection and comparison of individuals during the search process (e.g. Fonseca and Fleming's multi-objective EA (FFGA) (Fonseca & Fleming, 1993)). The use of this measure meant that solutions, both in convex and non-convex regions of estimated Pareto fronts, would be represented equally in the selection stages of these algorithms leading to better convergence.

Elite Individual Retention (1999/2000)

Zitzler et al. (2000) present a comparative study on six test functions introduced by Deb (1999) using a number of the most widely used MOEAs, including FFGA, the niched Pareto genetic algorithm (NPGA) of Horn *et al.* (1994), HLGA, VEGA and the non-dominated sorting genetic algorithm (NSGA) of Srinivas & Deb (1995). Their study suggests that their genetic algorithm (GA) based strength SPEA outperforms the other algorithms by consistently recording better results as measured by the $\mathcal{C}$ metric (Zitzler *et al.* , 2000; Zitzler, 1999; Zitzler & Thiele, 1999) on the test functions. In an earlier paper by Zitzler & Theile (1999), SPEA's superior performance was also demonstrated in comparison to four other MOEAs on a 0/1 knapsack problem. Another MOEA which has demonstrated significant results is the ES based Pareto archived evolution strategy (PAES) of Knowles and Corne (1999; 2000). The salient feature of these new methods is their active use of elitism in the multi-objective domain. Both SPEA and PAES retain an archive of the estimated Pareto optimal solutions that they have discovered in their search process so far, and reinsert copies of them back into the search population (in the case of SPEA), or use them as a source of parents for perturbation (in the case of PAES).

The unified model (2000)

As evinced by a number of comparative studies (Knowles & Corne, 1999; Knowles & Corne, 2000; Zitzler *et al.* , 2000; Zitzler, 1999; Zitzler & Thiele, 1999), both the SPEA and the PAES provide an effective methodology for multi-objective optimisation problems. Both algorithms can be seen as variants of the unified model for multi-objective evolutionary algorithms, UMMEA, introduced by Laumanns et al. (Laumanns *et al.* , 2000), and is outlined in Algorithm 1.

Algorithm 1 The sequential unified multi-objective evolutionary algorithm (Laumanns *et al.* , 2000). F_t denotes the elite archive, X_t the general (search) population and p_t^e the elitism intensity at generation t.

```
1: $t := 0$
2: $(F_0, X_0, p_0^e) := initialise\,()$
3: while $terminate\,(F_t, X_t, t) = false$
4:     $t := t + 1$
5:     $F_t' := update\,(F_{t-1}, X_{t-1})$
6:     $F_t := truncate\,(F_t')$
7:     $p_t^e := adapt\,(F_t, X_{t-1}, p_{t-1}^e)$
8:     $X_t := vary\,(sample\,(evaluate\,(F_t, X_{t-1}, p_t^e)))$
9: end
```

The algorithm summarises the UMMEA framework in terms of stochastic operators. The genetic search population X_0 is initialised, together with the elite archive F_0 (generally to an empty set). At each generation the elite archive is augmented to form F_t' by incorporating those solutions in X_{t-1} which are not dominated by any members of $F_{t-1} \bigcup X_{t-1}$; in addition any elements of F_{t-1}which are dominated by members of X_{t-1} are deleted from F_t'. For reasons of computational efficiency the new archive F_t' is then truncated to create the archive F_t, usually to some pre-stated size. In the SPEA this truncation is achieved by clustering, and in PAES a density based method is used. In Algorithm 1, crossover/mutation/perturbation are abstractly represented by the *vary* () operator. Individuals are selected using the *sample*() operator from F_t and X_{t-1} based on the fitness operator *evaluate*() for crossover/mutation/perturbation; the 'elitism intensity', p_t^e, controls the probability that an individual from the elite archive rather than the general population is selected. SPEA, PAES and the UMMEA all use an archive limited to a fixed maximum number of individuals, presumably to avoid the computational costs of maintaining a large archive.

3.2 Manifest problems of current approaches

For computational efficiency most MOEAs limit the size of their elite archive. In SPEA, for instance, this is achieved by clustering (in objective space) the individuals comprising F_t' and replacing clusters by the individual closest to the cluster centroid.

However, an artifact of truncation is that the Pareto front (as represented by the truncated archive) can shrink if individuals that define the boundaries of the estimated Pareto front[1] are removed (referred to here as the *extremal individuals).* If subsequent evolution fails to rediscover the extremal individuals, then repeated clustering will shrink the Pareto front and the final estimate of the Pareto set will lie across a narrow subset of the true frontier.

This effect is present even in the case of an offline 'dormant' estimated Pareto front (e.g., one that acts as a passive store for non-dominated individuals, and plays no part in the search process), as search will not be directed towards the extremal values. It is interesting to note that after the criticism by Zitzler *et al.* (2000), Coello (1999) and Srinivas & Deb (1995) of VEGA because of its bias toward extremal values, that its replacements should in turn be biased toward search in the centre of the front. (This is supported by the results presented in Zitzler and Thiele (1999) and Zitzler et al. (2000), where VEGA outperforms NPGA, FFGA and HLGA).

The shrinking front effect can be detrimental in two ways. The main consequence is the narrow extent of the estimated front (mis-representing the true Pareto front); secondly, extra search time is required in order to rediscover the extremes of the estimated Pareto front.

These problems are easily circumvented by removing the extremal individuals from the truncation process and passing them directly to F_t. In this thesis it is referred to as the *pinning* of extremal individuals, and has also been used in a recent extension to SPEA (Zitzler *et al.* , 2001).

3.2.1 Shrinking and oscillating estimated Pareto fronts

The elite archive is, in essence, a memory of where the algorithm has reached in previous generations in its estimation of the Pareto front, and should contain the 'best' estimate of the Pareto front at any stage. The estimated front should 'advance' in the sense that no individual in F_t should be dominated by any member of an earlier set, $F_0, \ldots, F_{t-1}$. Informally, it is said that an individual $\mathbf{x}$

[1]Containing what are referred to as component minima and maxima in (Laumanns *et al.* , 2000), in the case of the true Pareto set.

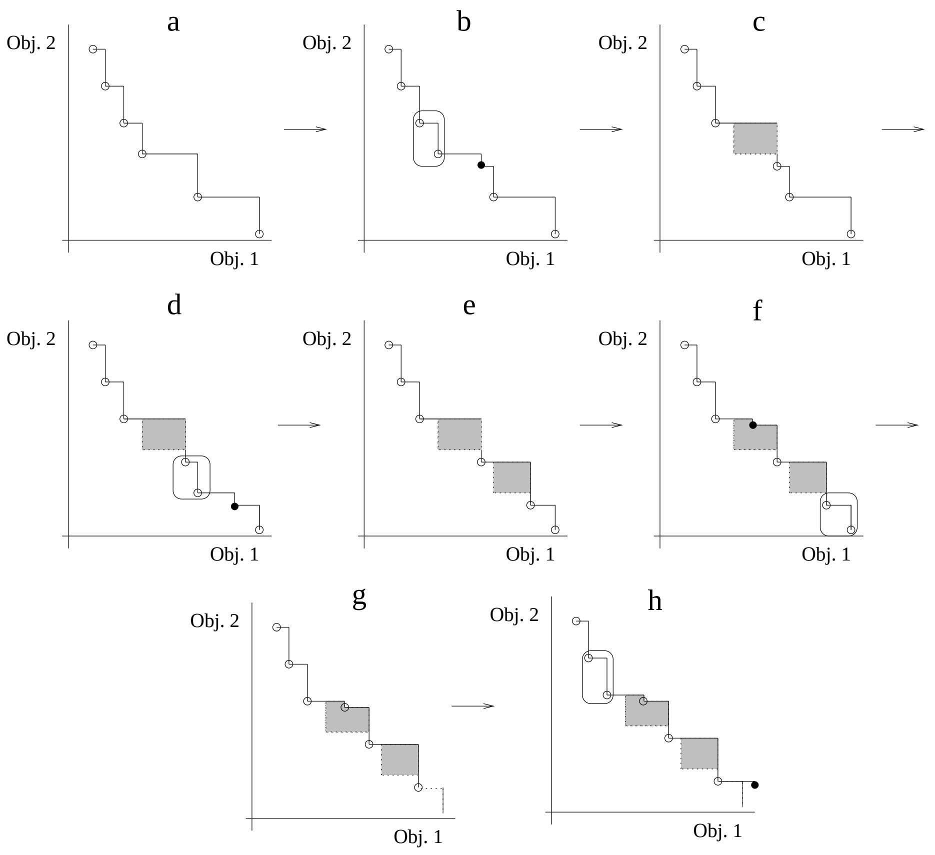


Figure 8: Example of a retreating estimated Pareto front, produced by truncation by clustering.

lies *behind* the front if a member of the elite set dominates $\mathbf{x}$. However, the requirement of several MOEAs, that F_t be limited to $\widehat{M}$ members, can produce 'retreating' or, more commonly, 'oscillating' estimates of the Pareto front. In these cases, members of F_t may lie behind the earlier estimates of the Pareto set. The nature of this behaviour is now described.

An illustration of a retreating front is shown in Figure 8. Figure 8a illustrates an archive, F_t, with a maximum of $\widehat{M} = 6$ members. In Figure 8b a new non-dominated member (drawn as a filled circle) has entered the set from the current population. Since there are now 7 elements in F_t', one member must be removed by clustering. The pair of solutions nearest each other form a cluster of two (circled) elements and one of them (chosen at random) is deleted, resulting in truncated archive F_t shown in Figure 8c. If at a subsequent generation a new element enters the full archive (Figure 8d), the clustering process will truncate the archive to the set as shown in Figure 8e. This results in an archive F_t (Figure 8g) containing an element that lies behind (i.e. is dominated by) elements of the original archive set (Figure 8a). Repeating this process can lead to the estimated front retreating or, more commonly, oscillating as the front advances in the MOEA search stage but retreats during truncation. This phenomenon was first noted by Hanne (1999) in different situation. In Hanne's MOEA (an applied example of which is available in Hanne (2000)), a $(\lambda + \mu)$-ES scheme was used with a child replacing the parent if the parent does not dominate the child. As such, the population that Hanne used was not a Pareto archive as it was not a non-dominated set (Equation 11); in fact, the population constructed in Hanne (1999) may only have one Pareto solution at any generation. The oscillation highlighted by Hanne is produced as a result of the search process as opposed to the truncation of the elite set. Laumanns et al. (2001), however, note its application to all elite archive truncation methods (but again without empirical examples) and used it as an impetus for their development of ϵ-dominance and ϵ-approximate Pareto sets.

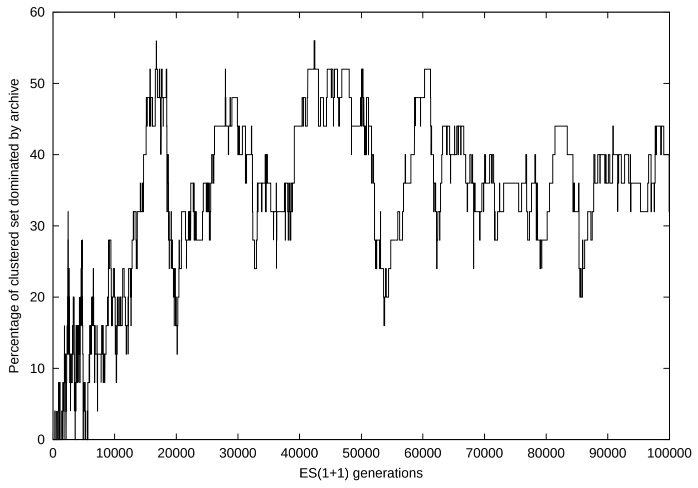


Figure 9: Percentage of individuals in active clustered archive dominated by members of the dormant unconstrained archive.

A simple empirical example of oscillation can be shown using a (1+1)-ES MOEA to optimise the ZDT1 function of Zitzler *et al.* (2000). The ZDT1 function was optimised using clustering to truncate the archive to 20 individuals. However, a dormant archive of unlimited size containing all non-dominated individuals was also maintained. Figure 9 shows the percentage of individuals in the truncated F_t that are dominated by members of the dormant unconstrained archive. As the number of generations increase the number of dominated individuals in the clustered archive increases and then oscillates around the 40% mark. Clearly the search process is being forced to rediscover portions of the front lost in the truncated archive but retained in the unconstrained archive (at generation 100,000 the dormant archive consisted of 95 individuals.) Since the extremal individuals were pinned, the effect seen is 'oscillation', rather than 'shrinking' or 'retreat'.

This artifact has two main consequences. First, search time is wasted 'rediscovering' individuals and regions that have been eliminated by truncation. Secondly, convergence to the true Pareto front is impaired. Numerical simulations show that this oscillation is particularly serious when the estimated front lies close to the true front. The oscillation can prevent convergence to the true front leading to poor estimates and difficulties in assessing convergence.

In the light of the artifacts discussed here, it is recommended that a secondary population of *all* currently non-dominated individuals found during the evolutionary search is used actively within the search process. In terms of the UMMEA (Algorithm 1), this thesis advocates dropping the *truncate* () operation. The set of all currently non-dominated individuals is referred to as the *frontal set* F_t. This approach has previously been adopted by, for example, Murata & Ishibuchi (1995) and Parks & Miller (1998). It should be noted that, while Laumanns *et al.* (Laumanns *et al.* , 2000) recommend the truncation of the external set, their results from this study are based on an MOEA which retains all non-dominated individuals. However, the issue of time cost of using an unconstrained elite archive has not been addressed, be it in the case of an active elite archive, or a dormant archive as recommended in Veldhuizen & Lamont (2000a).

As the number of objective dimensions increases, the issue of archive growth becomes yet more important. Beyond the argument of oscillating fronts, and the need to minimise fitness evaluations, the fact is that for every increase in the number of objective for a problem, there is, *ceteris paribus*, a doubling in probability of finding a solution that is mutually non-dominating with the elite archive.

A theoretical justification for this argument is illustrated in Figure 10. Here a single point is placed in the centre of a square covering the unit range and a cube covering the unit range. In the 2-D case this point dominates 1/4th of the area, is dominated by 1/4th of the area, and is mutually non-dominating with 1/2 of the area. In the 3-D case it dominates 1/8th of the volume, it is dominated by 1/8th of the volume and is mutually non-dominating with 3/4th of the volume. This process continues as D increases. More formally, taking a single point in the centre of D

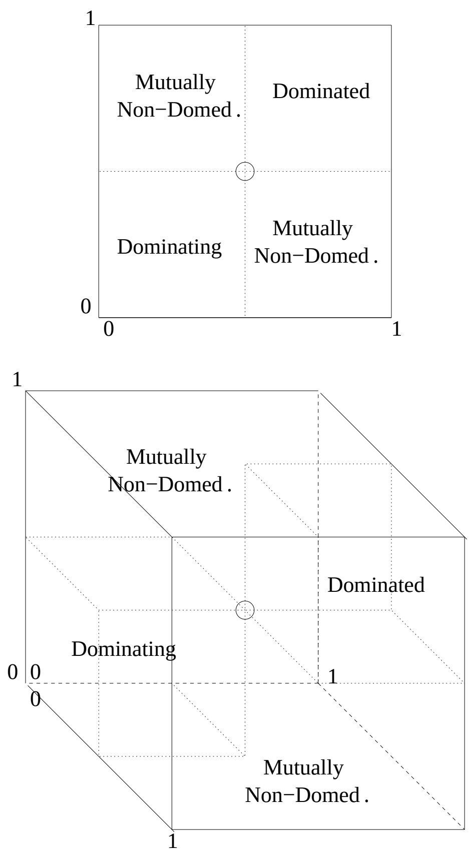


Figure 10: Non-dominated space around a point in 2 and 3 dimensions.

dimensional space, the amount of space that is mutually non-dominating with this point is $\frac{2^{D-1}-1}{2^{D-1}}$, giving proportions of $0, 0.5, 0.75, 0.875, 0.9375, \ldots, 0.998$ for dimensions 1 to 10.

In reality of course the front is moving forward in time as well as spreading out, so the growth will not tend to be quite as drastic as the formula would infer, however it can still be very large. An empirical example is provided in Figure 11a to 11i. Here objective vectors are generated at random from $\sim N(0,1)$ for objective vectors of length $D = 2, 3, \ldots, 10$ and stored in an unconstrained elite archive. After 100000 random individuals are generated, approximately 20 individuals reside in the elite archive at $D = 2$, as opposed to nearly 800 at $D = 5$ and 25000 when $D = 10$.

As the number of objectives increases the number of archive solutions grows, *ceteris paribus*. Also as the number of generations an MOEA runs for increases the the number of archive solutions also tend to grow (again highlighted in Figure 11). In application terms this means a larger and larger

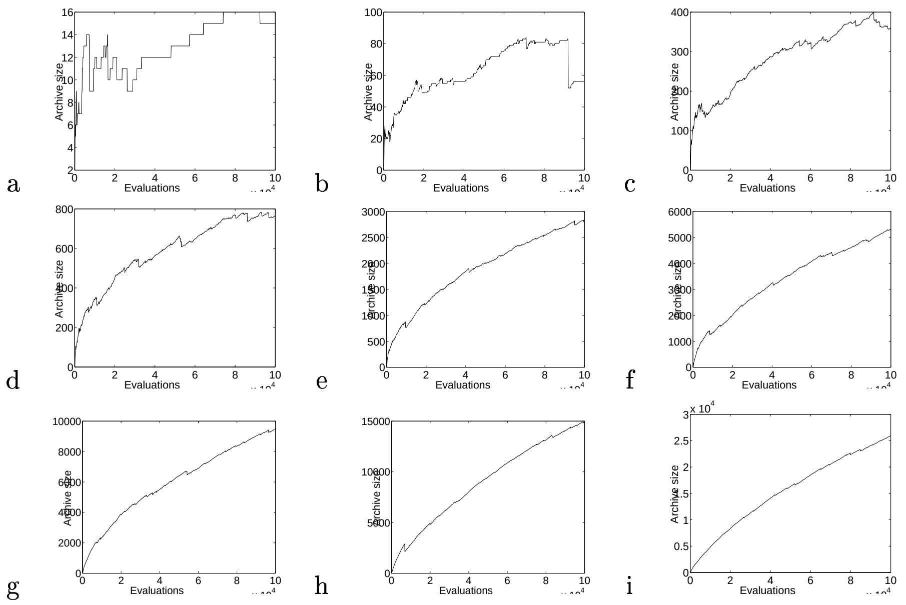


Figure 11: Archive growth in 2 to 10 dimensions, using random number generation.

proportion of an optimisers time is spent comparing solutions it finds to solutions it already has (in the archive), as opposed to searching for new solutions. The next chapter therefore introduces a new data structure to enable faster comparison of solutions to an archive.

Chapter 4

The Dominated and Non-Dominated Tree Data Structures

In this chapter, novel data structures are introduced to aid the efficient maintenance of estimated Pareto sets. These data structures maintain a quasi-linear order on a set of multi-objective non-dominating points. The properties of this representation allows searches with complexity which can be as low as that of binary search. Proofs of operation are derived, including the complexity of various different operations, and their algorithms are provided.

4.1 Using an unconstrained elite archive

In most MOEAs using an active archive, the archive must be searched at two distinct junctures:

1. When representative individuals are selected for binary tournament selection, roulette wheel selection, etc. (in the case of a GA) or for perturbation (in the case of an ES).

2. When individuals are compared with the archive F_t to determine whether they dominate or are dominated by members of F_t.

A data structure to facilitate searching F_t which is faster than a linear list is the *dominated tree.* Since the frontal set contains all of the currently non-dominated decision vectors found, it may become very large as the search progresses. In order for an MOEA search using an unconstrained archive to be computationally viable, efficient ways of selecting elements of F_t (for crossover, mutation and perturbation and of storing F_t to permit rapid query, insertion and deletion of its elements) must be found. These problems and their solutions are now discussed.

4.1.1 Selection

In a MOEA implementing an unconstrained elite archive many more individuals than necessary may be available for selection during its evolutionary processes. In this case the manner in which the *sample* () operator (Algorithm 1) selects individuals becomes important. Uniform random selection of individuals from F_t artificially concentrates the search on densely populated regions of the front. It is therefore helpful to select a number of representative individuals. An approach could be to use the SPEA clustering method to find representatives, however, this proves to be too time consuming for large frontal populations (due to the computational complexity of the process). Here the new method of partitioned quasi-random selection (PQRS) is introduced as a selection routine and used in conjunction with active unconstrained elite archives.

Suppose that N elite individuals are required for selection. In PQRS an arbitrary objective out of the D objective dimensions is partitioned into $(N - 1)$ bins of equal width and one individual is selected at random from each of these bins. This ensures that individuals are selected uniformly across the extent of the front in the chosen dimension. The objective dimension selected for partitioning rotates through the D dimensions with the generation t. The individual on the extreme of the objective dimension is always selected ensuring the concept of pinning discussed earlier.

An example of PQRS in a $D = 2$ objective problem is shown in Figure 12. Here $N = 5$ individuals are to be selected from a frontal population of $M = 24$ (where $M = |F_t|$ denotes the current size of the frontal set). Given the extremal individual is selected, four additional individuals must be selected from F_t. Having selected an objective coordinate, the frontal set is partitioned on that coordinate into $(N - 1)$ equally spaced bins. In Figure 12 this can be seen as four bins for the selected dimension, each spanning 1/4th of the range of the front on that objective dimension. An individual in each bin is selected by generating a random number uniformly distributed across the range of the bin, and selecting the closest individual (based on Euclidean distance). If a bin is empty (for instance due to a discontinuity in the Pareto front), the closest individual from the entire front is used. In addition, no individual is selected twice (unless $M < N$).

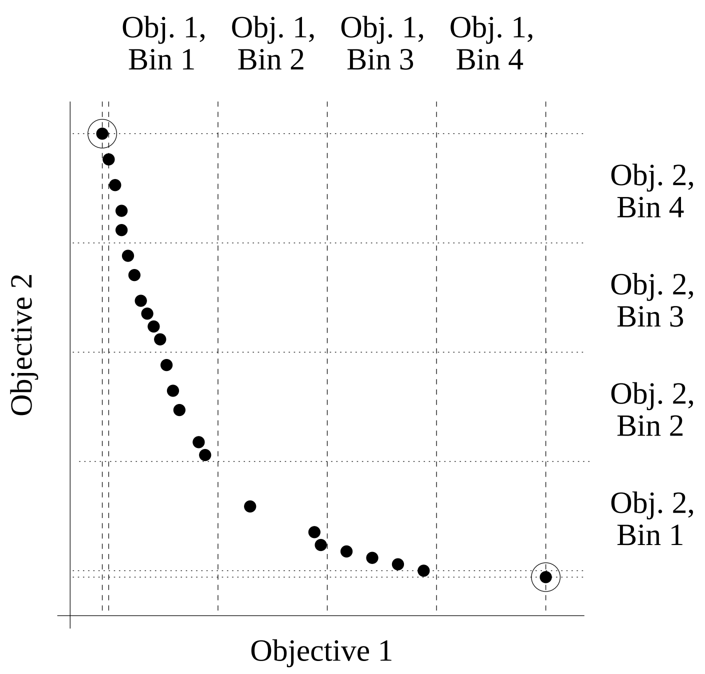


Figure 12: A two objective example of partitioned quasi-random selection, with the objective 1 and 2 dimensions partitioned in illustration (during selection only one dimension is partitioned at each generation). $N = 5$ representative individuals are required, so selection is from $N - 1 = 4$ bins (after automatic selection of the extremal value (circled)).

Note that in SPEA clustering is used to reduce the archive before individuals are selected for binary tournament selection; in contrast, PQRS does not reduce the frontal set population: it only selects individuals for breeding and does not remove them from F_t. This approach is similar to that used in PAES, when the objective space is partitioned into grids. However, as the partitioning is done with bins in PQRS, grid knowledge does not need maintenance and updating as in Knowles & Corne (2000).

Rapid selection from the frontal set is enabled by maintaining D binary trees, one for each objective dimension. This means that each selection takes $\mathcal{O}(2 \log_2 M)$ comparisons as opposed to $\mathcal{O}(M)$ for a linear search. Since the frontal set is constantly changing this can be conveniently implemented using self balancing trees (e.g. AVL or Red-Black trees (de Berg *et al.* , 1997)), or doubly linked lists.

If only one elite individual is desired (for instance a (1+1)-ES is being used), then the objective space is still separated into $(N-1)$ bins, with a particular bin (or extreme individual) selected with equal probability. A random number is generated from a uniform distribution across the range of the bin, and the individual closest to this number is selected. The larger the value of N, the less the selection process will be affected by dense areas of the archive, and the less the bias toward selecting the extremal individual.

4.1.2 Efficient storage of the frontal set

An important constraint on using a large active frontal set is the number of comparisons that must be made with individuals in the frontal set at each generation. When the archive is small in size, for instance $M = 20$ (as in (Zitzler *et al.* , 2000)), the time for a linear search of F_t is negligible. However, with no limits imposed on the size of the frontal set in an MOEA, the linear search of a large number of individuals (1000 for instance) before assigning an individual as non-dominated, may simply be too costly to make the method practical. Hence intelligent storage is needed before the unconstrained frontal set approach is viable.

4.2 Dominated and non-dominated trees

To determine whether an individual, $\mathbf{y} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_D(\mathbf{x}))$, should become a member of the frontal set, F,[1] it must be first checked that $\mathbf{y}$ is not dominated by any other element of F. At the same time any elements of F that are dominated by $\mathbf{y}$ should be deleted from F. When the frontal set is small, a simple linear search is sufficiently cheap to perform these checks. However, as the size of the frontal set grows the cost of querying the frontal set becomes prohibitive. In this section two data-structures are described— *dominated trees* and *non-dominated trees* — for efficient storing and rapid querying, and updating of the frontal set.

For this discussion it is convenient to denote members of the frontal set and individual(s) from the search population as points $\mathbf{y}$ in D-dimensional space. Geometrically, finding individuals in F that dominate $\mathbf{y}$ amounts to finding frontal individuals that lie to the 'south-west' or 'left and below' $\mathbf{y}$. Mathematically, the set of dominating individuals is:

$$\{\mathbf{z} \in F \,:\, \mathbf{z}_i \leq \mathbf{y}_i \text{ for all } 1 \leq i \leq D \text{ and } \mathbf{z}_j < \mathbf{y}_j \text{ for at least one } 1 \leq j \leq D\} \tag{17}$$

[1]Since the genetic generation plays no role here, the subscript t has been dropped.

It would be possible to use kd-trees (Bentley, 1975; de Berg *et al.* , 1997) or range trees (Bentley & Friedman, 1979; de Berg *et al.* , 1997), however these are both suited to querying F for elements which lie in *bounded* (hyper-) rectangles. Priority trees, developed by McCreight (1985), are suited to rectangular queries in which the rectangle is unbounded on a *single* side. Sun and Steuer (Sun & Steuer, 1996) describe an alternative data structure adapted for answering queries about domination and non-domination which has been extended recently by Mostaghim et al. (Mostaghim *et al.* , 2002).

The 'dominates' relation imposes a partial ordering on individuals. However, since the elements of F are mutually non-dominating, this relation cannot be used directly to construct, for example, a binary tree to enable fast searching. In the dominated tree, *composite points* are constructed from original individuals such that the composite points are ordered by the dominates relation so that binary search may be used in the new structure.

The dominated tree consists of a list of $L = \lceil M/D \rceil$ *composite points*, $\mathbf{c}_j$, ordered by the weakly-dominates relation, $\preceq$:

$$\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_2 \preceq \mathbf{c}_1\} \tag{18}$$

Usually, the stronger condition, $\mathbf{c}_i \prec \mathbf{c}_j$ iff $i > j$, will hold. The coordinates of each composite point are defined by up to D elements of F, the *constituent points* of a composite point. An example of a dominated tree in two dimensions is shown in Figure 13.

Denote by Y_i the constituent points of $\mathbf{c}_i$, namely the D-tuple defining the coordinates of $\mathbf{c}_i$; so that if

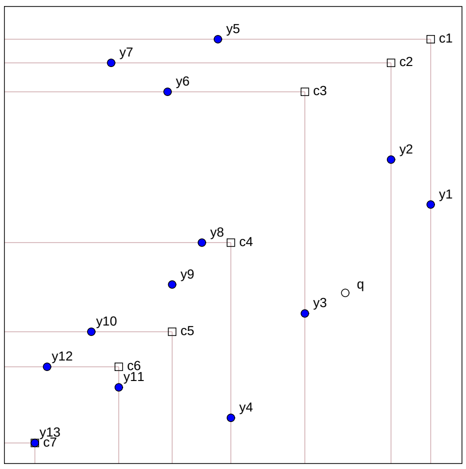
$$Y_i = \langle \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(d)}, \ldots, \mathbf{y}^{(D)} \rangle \tag{19}$$

then the dth coordinate of the composite point is the dth coordinate of $\mathbf{y}^{(d)}$; thus $c_{i,d} = y_d^{(d)}$. Dominated trees are constructed to have the property that the constituent points of $\mathbf{c}_i$ (at least) weakly dominate $\mathbf{c}_j$ and at least one of the constituent points of $\mathbf{c}_i$ strictly dominates $\mathbf{c}_j$:

$$\text{If } \mathbf{c}_i \preceq \mathbf{c}_j \text{ then } \mathbf{y}^{(d)} \preceq \mathbf{c}_j \ \forall \mathbf{y}^{(d)} \in Y_i \text{ and } \exists\, \mathbf{y}^{(d)} \in Y_i \text{ such that } \mathbf{y}^{(d)} \prec \mathbf{c}_j \tag{20}$$

For example, in Figure 13 the constituent points of $\mathbf{c}_4, \mathbf{c}_5$ and $\mathbf{c}_6$ dominate $\mathbf{c}_3$. However, note that they do not necessarily dominate the constituent points of $\mathbf{c}_3$, namely $\mathbf{y}_3$ and $\mathbf{y}_6$.

It follows from (20) that if $\mathbf{c}_i \preceq \mathbf{c}_j$ then all the constituent points of $\mathbf{c}_i$ (at least) weakly dominate



Composite node	Constituent points
$\mathbf{c}_1$	$\langle \mathbf{y}_1, \mathbf{y}_5 \rangle$
$\mathbf{c}_2$	$\langle \mathbf{y}_2, \mathbf{y}_7 \rangle$
$\mathbf{c}_3$	$\langle \mathbf{y}_3, \mathbf{y}_6 \rangle$
$\mathbf{c}_4$	$\langle \mathbf{y}_4, \mathbf{y}_8 \rangle$
$\mathbf{c}_5$	$\langle \mathbf{y}_9, \mathbf{y}_{10} \rangle$
$\mathbf{c}_6$	$\langle \mathbf{y}_{11}, \mathbf{y}_{12} \rangle$
$\mathbf{c}_7$	$\langle \mathbf{y}_{13}, \mathbf{y}_{13} \rangle$

Figure 13: Top: 13 points $\{\mathbf{y}_1, \ldots, \mathbf{y}_{13}\}$ in two dimensions and the composite points $\{\mathbf{c}_1, \ldots, \mathbf{c}_7\}$ (squares) forming a dominated tree. The open circle, $\mathbf{q}$ represents a query point. Bottom: Composite nodes listed as ordered by $\prec$.

all points that are dominated by $\mathbf{c}_j$:

$$\text{If} \quad \mathbf{c}_i \preceq \mathbf{c}_j \preceq \mathbf{y} \quad \text{then} \quad \mathbf{y}^{(d)} \preceq \mathbf{y} \quad \forall \mathbf{y}^{(d)} \in Y_i \tag{21}$$

It should be emphasised that dominated and non-dominated trees do not require that the constituent elements form a non-dominated set (as seen in Figure 13). The dominated tree is illustrated with a general set of two-dimensional points, because non-dominated sets of two-dimensional elements have the property that listing the points in order of increasing first coordinate (objective), y_1, is equivalent to listing them in order of decreasing second coordinate, y_2. This is seen easily by considering two mutually non-dominating points, say, $\mathbf{u} \in \mathbb{R}^2$ and $\mathbf{v} \in \mathbb{R}^2$. Without loss of generality, suppose that $u_1 < v_1$. If $u_2 \leq v_2$ then $\mathbf{u} \prec \mathbf{v}$, contradicting the fact that they are mutually non-dominating, and thus it may be concluded that $u_2 \geq v_2$. This ordering property is special to two dimensions, and it can be misleading if one tries to generalise to higher dimensions from a visual representation in two dimensions. The points and the tree illustrated in Figure 13 are more akin to the general ($D > 2$) case. Dominated trees also have applications to general sets (i.e., sets whose elements are not mutually non-dominating), such as answering queries about enclosing rectangles (McCreight, 1985; McCreight, 1980).

4.2.1 Construction

Algorithm 2 Construction of a dominated tree.

```
Inputs:
      $F = \{\mathbf{y}_1, \ldots, \mathbf{y}_m, \ldots, \mathbf{y}_M\}$          Vectors to form the dominated tree

 1:  $\mathcal{T} := \emptyset$
 2:  $L = \lceil M/D \rceil$
 3:  for $i := 1, \ldots, L$
 4:      for $d := 1, \ldots, D$
 5:          $\mathbf{y} := \max_{\mathbf{y}_m \in F}(y_{m,d})$
 6:          $c_{i,d} := \mathbf{y}_d$                    Keep pointers from $\mathbf{y}_m$ to and from $\mathbf{c}_i$
 7:          if $|F| \neq 1$
 8:              $F := F \setminus \mathbf{y}$            Delete $\mathbf{y}$ from $F$
 9:          end
10:      end
11:      $\mathcal{T} := \{\mathbf{c}_i \preceq \mathbf{c}_{i-1} \preceq \ldots \preceq \mathbf{c}_1\}$    Append $\mathbf{c}_i$ to $\mathcal{T}$
12:  end
```

Construction of a dominated tree from M points $F = \{\mathbf{y}_1, \ldots, \mathbf{y}_m, \ldots, \mathbf{y}_M, \}$ is described in Algorithm 2 and proceeds as follows. The first composite point $\mathbf{c}_1$ is constructed by finding the individual $\mathbf{y}_m$ with maximum first coordinate; this value forms the first coordinate of the composite point:

$$c_{1,1} = \max_{\mathbf{y}_m \in F} (y_{m,1}) \tag{22}$$

Thus, for example, in Figure 13, $c_{1,1} = y_{1,1}$. This individual $\mathbf{y}_m$ is now associated with $\mathbf{c}_1$ and removed from F. Likewise, the second coordinate of $\mathbf{c}_1$ is given by the maximum second coordinate of the points remaining in F: $c_{1,2} = \max_{\mathbf{y}_m \in F \setminus \mathcal{T}} (y_{m,2})$; in Figure 13 $c_{1,2} = y_{5,2}$. This procedure is repeated to construct $\mathbf{c}_2$ and subsequent composite points until all M elements of F are associated with the tree. In general the dth coordinate of the ith composite point is given by:

$$c_{i,d} = \max_{\mathbf{y}_m \in F \setminus \mathcal{T}} y_{m,d} \tag{23}$$

Note that in the construction of the final composite point (that is, the composite point that dominates all other composite points), the M elements of F may have been used before all of the D coordinates of the final composite point $\mathbf{c}_L$ have been defined. As illustrated by composite point $\mathbf{c}_7$ in Figure 13, the last remaining point in F ($\mathbf{y}_{13}$ in Figure 13) is reused to define the remaining coordinates.

It is clear from the construction of $\mathcal{T}$ that it possesses properties (18) and (20). Since D elements of F are used in the construction of each composite point (except possibly for the dominating composite point), the number of composite points in $\mathcal{T}$ is $L = \lceil M/D \rceil$.

If a na ive search for the maximum in line 5 of Algorithm 2 is used, construction of $\mathcal{T}$ from M points takes $\mathcal{O}(M^2)$ comparisons (because each of the M points $\mathbf{y}_m$ is inserted after searching F of length $M - m$). However, if D PQRS trees have been constructed (at cost $\mathcal{O}(DM \log_2 M)$), they can be used to efficiently find the constituents of a new composite point. Each point in the PQRS trees is marked as 'used' trees as it is inserted into $\mathcal{T}$; the dth coordinate of the composite point under construction is then the maximum unused point in the dth PQRS tree. By maintaining D pointers to the maximum unused element in the PQRS trees the dominated tree may be constructed in $\mathcal{O}(MD)$ operations.

4.2.2 Query

Given a test point $\mathbf{q}$, the properties of dominated trees can be used as follows to discover which points in F dominate $\mathbf{q}$. Although the dominated tree is implemented efficiently as a binary tree, the query procedure is most easily described in terms of an ordered list of L composite points. First, the list is searched to find the indices h and l of composite points $\mathbf{c}_h$ and $\mathbf{c}_l$ that dominate and are dominated by $\mathbf{q}$ respectively:

$$h = \begin{cases} L + 1 & \text{if } \mathbf{q} \prec \mathbf{c}_L \\ \min \{ i \,:\, \mathbf{c}_i \prec \mathbf{q} \} & \text{otherwise} \end{cases} \tag{24}$$

and

$$l = \begin{cases} 0 & \text{if } \mathbf{c}_1 \prec \mathbf{q} \\ \max\{i : \mathbf{q} \prec \mathbf{c}_i\} & \text{otherwise} \end{cases} \tag{25}$$

Also $\mathbf{c}_H$ denotes the 'least' composite point that strictly dominates $\mathbf{c}_h$:

$$H = \min\{i : \mathbf{c}_i \prec \mathbf{c}_h\} \tag{26}$$

For the query point illustrated in Figure 13, $l = 2$, $h = 5$, and $H = 6$. (Note that it is not necessarily true that $H = h + 1$.) Since $\mathbf{c}_h \prec \mathbf{q}$ it is clear from (20) that all the constituent points of the composite points $\mathbf{c}_i$ that dominate $\mathbf{c}_h$, $H \leq i \leq L$, also dominate $\mathbf{q}$. (Note that the constituent points of $\mathbf{c}_h$, and indeed $\mathbf{c}_i$, that only weakly dominate $\mathbf{c}_h$, need not dominate $\mathbf{q}$; e.g. in Figure 13 $\mathbf{c}_5 \prec \mathbf{q}$, but $\mathbf{y}_9 \not\prec \mathbf{q}$.) Also, since $\mathbf{q} \prec \mathbf{c}_l$ and the constituent points of $\mathbf{c}_l$ have at least one coordinate equal to a coordinate of $\mathbf{c}_l$, it may be concluded that $\mathbf{q}$ is not dominated by any of the constituent points of $\mathbf{c}_1, \ldots, \mathbf{c}_l$. Each constituent point of $\mathbf{c}_i$ $l < i < H$ must be checked individually to determine whether it dominates $\mathbf{q}$; in Figure 13 the points $\mathbf{y}_3, \mathbf{y}_6, \mathbf{y}_4, \mathbf{y}_8, \mathbf{y}_9$, and $\mathbf{y}_{10}$ must be individually checked.

If the dominated tree is used to maintain a mutually non-dominating set, when determining whether $\mathbf{q}$ is to be included in F it can be rejected immediately if $h < L$ because it is certainly dominated by at least one of the constituents of $\mathbf{c}_L$.

Since the composite points are arranged as a sorted list, determination of l and h takes $\mathcal{O}(\log_2(M/D))$ domination comparisons each. Hence the total number of domination comparisons required to enumerate the elements in F that dominate a query point is $\mathcal{O}(2\log_2(M/D) + DK)$, where K is the number of points that have to be checked individually. Clearly, certain configurations of F and $\mathbf{q}$ can result in all elements of F being checked — in linear time. However, such arrangements are seldom encountered in practice. This rapid checking permits very large archive sets to be maintained efficiently.

Although, as already mentioned, dominated trees may be used to search general sets, their primary use is for maintaining an archive F of mutually non-dominating points. In this case a query is often made to discover whether $\mathbf{q}$ is dominated by any element of F; if not, $\mathbf{q}$ is inserted into F. If it is determined that $\mathbf{q}$ is to be included in F then those elements of F that are dominated by $\mathbf{q}$ must be identified and deleted from F. Queries about which elements of F are dominated by $\mathbf{q}$ can be answered using the dominated tree, however, it may be inefficient. This is because even though $\mathbf{q} \prec \mathbf{c}_i$ for $1 \leq i \leq l$, $\mathbf{q}$ need not dominate the constituent points of these $\mathbf{c}_i$ and the constituent points must therefore be checked individually. Hence, in Figure 13 for example, $\mathbf{q} \prec \mathbf{c}_2 \prec \mathbf{c}_1$, but $\mathbf{y}_5$ and $\mathbf{y}_7$ are not dominated by $\mathbf{q}$. The *non-dominated tree* is a data structure that permits this sort of query to be answered efficiently.

4.2.3 Non-dominated trees

Non-dominated trees are analogous to dominated trees, but they facilitate searching for the subset of F that is dominated by $\mathbf{q}$; that is, points that are to the 'north-east' or 'right and above' $\mathbf{q}$ may be located efficiently. A non-dominated tree consists of an ordered list of composite points:

$$\mathcal{T} = \{\mathbf{c}_1 \preceq \mathbf{c}_2 \preceq \ldots \preceq \mathbf{c}_L\} \tag{27}$$

Analogous to property (20), non-dominated trees are constructed with the property that if $\mathbf{c}_i \prec \mathbf{c}_j$, then all of the constituent points of $\mathbf{c}_j$ are (at least) weakly dominated by $\mathbf{c}_i$:

$$\text{If } \mathbf{c}_i \preceq \mathbf{c}_j \text{ then } \mathbf{c}_i \preceq \mathbf{y}^{(d)}\ \forall \mathbf{y}^{(d)} \in Y_j \text{ and } \exists\ \mathbf{y}^{(d)} \in Y_i \text{ such that } \mathbf{c}_j \prec \mathbf{y}^{(d)} \tag{28}$$

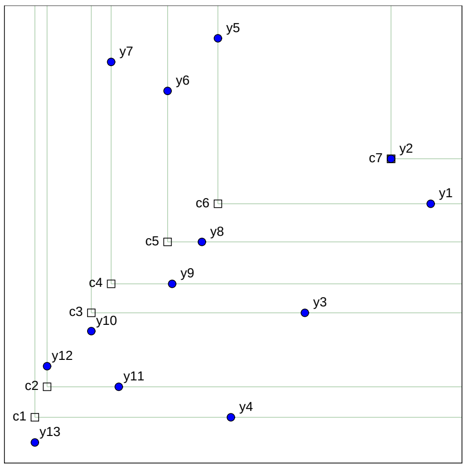


Figure 14: The non-dominated tree representing the 13 points $\{\mathbf{y}_1, \ldots, \mathbf{y}_{13}\}$ illustrated in figure 13.

An example of a non-dominated tree is shown in Figure 14 (again for a general set). Operations on non-dominated trees are analogous to those on dominated trees. Construction proceeds by starting with the *minimum* coordinate of each objective and successively inserting points with the minimum coordinate remaining in each dimension. Likewise, queries are analogous to queries in dominated trees.

If one needs to find out both if a point dominates members of the stored set and if it is dominated by set members, then both data structures (dominated and non-dominated trees) need to be maintained.

4.2.4 Insertion and deletion

Elements are added and deleted continually from the frontal set during the course of an optimisation. It is therefore important that the data structure used to support F can be modified dynamically. Online insertion of a new point $\mathbf{y}$ is accomplished in a simple manner as outlined in Algorithm 3.

Algorithm 3 Insertion into a dominated tree.

```
Inputs:
    $\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$      Dominated tree of $L$ composite points
    $\mathbf{y}$                                  Point to be inserted into $\mathcal{T}$

 1: if $\mathbf{y} \preceq \mathbf{c}_L$
 2:   for $k = 1, \ldots, D$
 3:     $c'_k := y_k$
 4:   end
 5:   $\mathcal{T} := \{\mathbf{c}' \preceq \mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$
 6: else
 7:   $j := \arg\min(\mathbf{c}_i \succ \mathbf{y})$          Binary search
 8:   $\mathbf{c}' = \mathbf{c}_{j+1}$                   New composite point
 9:   for $k = 1, \ldots, D$
10:     if $y_k > c_{j+1,k}$
11:       $c'_k := y_k$
12:       break
13:     end
14:   end
15:   for $k = L, \ldots, j+1$
16:     $\mathbf{c}_{k+1} := \mathbf{c}_k$                 Relabel
17:   end
18:   $\mathbf{c}_{j+1} := \mathbf{c}'$                 $\mathcal{T} := \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_{j+1} \preceq \mathbf{c}' \preceq \mathbf{c}_j \preceq \ldots \preceq \mathbf{c}_1\}$
19: end
```

If the new point dominates $\mathbf{c}_L$ (the composite point that dominates all others) then a new composite point, $\mathbf{c}'$, is created with all of its coordinates defined by $\mathbf{y}$ (steps 1-4 in Algorithm 3).

It is clear that $\mathbf{c}' \preceq \mathbf{c}_L$, and therefore $\mathbf{c}'$ may be appended to $\mathcal{T}$ as $\mathbf{c}_{L+1}$, preserving the ordering property (18).

If $\mathbf{y}$ does not dominate $\mathbf{c}_L$ a bisection (binary) search is used to locate composite points, such that $\mathbf{c}_{j+1} \not\preceq \mathbf{y}$ and $\mathbf{y} \preceq \mathbf{c}_j$. A new composite point $\mathbf{c}'$ is then constructed with coordinates equal to $\mathbf{c}_{j+1}$ except in the first coordinate for which $y_k > c_{j+1,k}$ as accomplished by lines 8-14.

By construction, $\mathbf{c}_{j+1} \preceq \mathbf{c}' \preceq \mathbf{c}_j$, so $\mathbf{c}'$ may be inserted in $\mathcal{T}$ between $\mathbf{c}_{j+1}$ and $\mathbf{c}_j$ while preserving the ordering property (18). In Algorithm 3 this properly and the necessary relabelling is accomplished by lines 15-19. Most practical implementations of ordered lists (such as binary trees or linked lists) are not index based, so this relabelling is unnecessary. The binary search of $\mathcal{T}$ takes $\mathcal{O}(\log_2(M/D))$ domination comparisons and the calculation of the coordinates of $\mathbf{c}'$ is equivalent to another domination comparison (D comparisons), therefore the cost of insertion is $\mathcal{O}(\log_2(M/D))$ domination comparisons.

Figure 15 shows the dominated tree resulting from the insertion of a new point $\mathbf{y}_{14}$ in the tree shown in Figure 13. Note that the tree resulting from insertion is less than optimal in the sense that a point (e.g., $\mathbf{y}_6$) may contribute to more than one composite point.

Deletion of a point $\mathbf{y}_m$ from the tree is slightly more complicated, because the composite point to which $\mathbf{y}_m$ contributed, say $\mathbf{c}_j$, must remain (at least) weakly dominated by $\mathbf{c}_{j+1}$ after the deletion. Assume that in the construction of $\mathcal{T}$, pointers are kept from each element $\mathbf{y}_m$ to the composite point $\mathbf{c}_j$ of which $\mathbf{y}_m$ is a constituent (illustrated in Figure 17).

Algorithm 4 Deletion from a dominated tree.

```
Inputs:
    $\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$    Dominated tree of $L$ composite points
    $\mathbf{y}_m$ with $c_{j,k} = y_{m,k}$    Point to be deleted from $\mathcal{T}$
                                              $\mathbf{y}_m$ defines the $k$th coordinate of $\mathbf{c}_j$

1:  if $j < L$
2:      $\mathbf{c}_{j,k} := \mathbf{c}_{j+1,k}$    Use coordinate from dominating composite point
3:  else
4:      for $\mathbf{u} \in Y_j \setminus \mathbf{y}_m$    Check the remaining constituents of $\mathbf{c}_j$
5:          if $u_k > c_{j,k}$
6:              $c_{j,k} := u_k$
7:          end
8:      end
9:  end
```

If $\mathbf{y}_m$ defines the kth coordinate of $\mathbf{c}_j$, then, as shown in Algorithm 4, upon deletion of $\mathbf{y}_m$, the deleted coordinate of $\mathbf{c}_j$ is replaced by the kth coordinate of $\mathbf{c}_{j+1}$. This assignment ensures that $c_{j,k} \leq c_{j+1,k}$, so that $\mathbf{c}_{j+1} \preceq \mathbf{c}_j$, as required by (18). Also note that the constituent points of $\mathbf{c}_j$ after the deletion are either the constituent points of $\mathbf{c}_j$ before the deletion or the constituent points of $\mathbf{c}_{j+1}$. Therefore, if before the deletion $\mathbf{c}_j \preceq \mathbf{y}$, then after the deletion the constituent points of $\mathbf{c}_j$ continue to weakly dominate $\mathbf{y}$, showing that properties (20) and (21) are preserved.

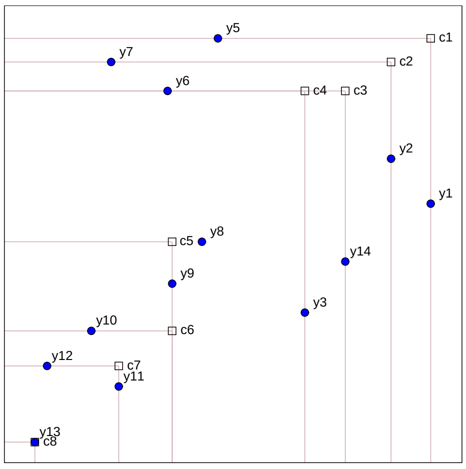


Figure 15: Dominated tree resulting from the tree shown in figure 13 after the insertion of $\mathbf{y}_{14}$ and deletion of $\mathbf{y}_4$.

Deletion of $\mathbf{y}_4$ from the tree shown in Figure 13 is illustrated in Figure 15. Note that if $\mathbf{y}_m$ contributes to more than one composite point, then each of the composite points to which it contributes must be dealt with in turn, beginning with the one that dominates all of the others. On the other hand, if $\mathbf{y}_m$ is the sole constituent point of $\mathbf{c}_j$ (e.g., $\mathbf{c}_{13}$ in Figure 13) then $\mathbf{c}_j$ is deleted from $\mathcal{T}$ when $\mathbf{y}_m$ is deleted. Provided that pointers are kept from each $\mathbf{y}_m$ to the composite points to which it contributes, then deletion of a point contributing to k composite nodes is achieved with $\mathcal{O}(kD)$ comparisons or, equivalently, $\mathcal{O}(k)$ domination comparisons.

Cleaning

Insertion and deletion operations sometimes result in points contributing to more than one composite point. The dominated and non-dominated trees therefore contain more composite nodes than the optimum $\lceil M/D \rceil$ and hence increased time is needed to search them. This can be alleviated by

Algorithm 5 Cleaning a dominated tree.

```
Inputs:
    $\mathcal{T} = \{\mathbf{c}_L \preceq \ldots \preceq \mathbf{c}_1\}$        Dominated tree of $L$ composite points

1:  $j := 1$
2:  while $j < |\mathcal{T}|$
3:      for $\mathbf{y} \in Y_j$
4:          $L := |\mathcal{T}|$
5:          $\mathcal{T} := append(delete(\mathbf{y}, \mathcal{T}_{j+1,L}), \mathcal{T}_{1,j})$
6:      end
7:      $j := j + 1$
8:  end
```

periodically 'cleaning' the tree in the following manner. Let $\mathcal{T}_{j,k}$ denote the sub-tree of $\mathcal{T}$ composed of the composite points $j, \ldots, k$:

$$\mathcal{T}_{j,k} = \{\mathbf{c}_k \preceq \ldots \preceq \mathbf{c}_j\} \tag{29}$$

and let the deletion operation (Algorithm 4) be denoted by $delete(\mathbf{y}, \mathcal{T})$. As described in Algorithm 5 cleaning is achieved by successively deleting constituent nodes from all composite nodes except for the least-dominating node to which they contribute. This ensures that every point $\mathbf{y}_m$ contributes to exactly one composite point. A cleaned tree is shown in Figure 16.

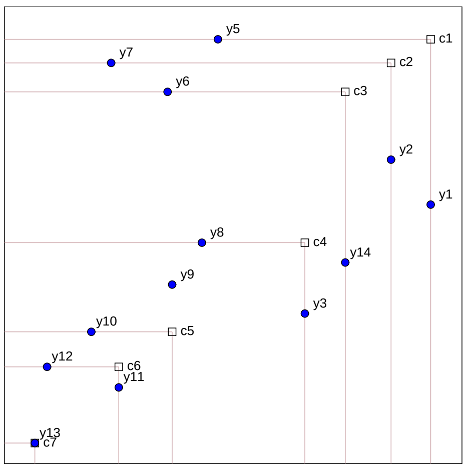


Figure 16: Dominated tree resulting from 'cleaning' of the tree shown in figure 15.

The number of composite nodes in an ideal tree is $\lceil M/D \rceil$, so in a dirty tree of L composite nodes the number of points that contribute to more than one composite node is $LD - M$. Since deletion of a single point takes $\mathcal{O}(1)$ domination comparisons, the time taken to clean a tree is estimated as $\mathcal{O}(LD - M)$. In practice it is not efficient to clean the tree following every insertion and deletion, but occasional cleaning may be triggered when the number of composite points exceeds a threshold, say $2M/D$. Alternatively, if PQRS trees have been constructed then they may be used to efficiently rebuild the dominated tree 'from scratch'.

4.3 Application of data structures

In the following empirical chapter the data structures are implemented using C++ with doubly linked lists, as illustrated in Figure 17.

As is shown, in this implementation each archived individual is contained in one or more composite points in the dominated and non-dominated trees, and also has 'knowledge' of which composite points it is in (through pointer references to doubly linked list nodes). In the illustration point $\mathbf{y}_m$ contributes to two composite points of the dominated tree ($\mathbf{c}_2$ and $\mathbf{c}_3$) and one composite point of the non-dominated tree ($\mathbf{c}_8$). However, as discussed earlier, the dominated and non-dominated trees described are formulated to be independent of the base tree format. Since the work has been made available electronically by the author from his publications website,[2] a number of different implementations have been developed including doubly linked lists, binary trees and vectors, in a number of different research institutions.

4.3.1 Use in MOEAs

In practical application to MOEAs the author represents the archive frontal set F by D PQRS trees together with one dominated and one non-dominated tree. The dominated tree permits rapid searching of F to discover whether a member $\mathbf{q}$ of the internal population should be included in F; if it is, the non-dominated tree is used to rapidly locate and remove those elements of F that are dominated by $\mathbf{q}$. The PQRS trees are used for ensuring that elite individuals (from F) selected for breeding are selected uniformly along F and for maintenance of the dominated and non-dominated trees.

[2]http://www.dcs.ex.ac.uk/people/jefields/seminars.html

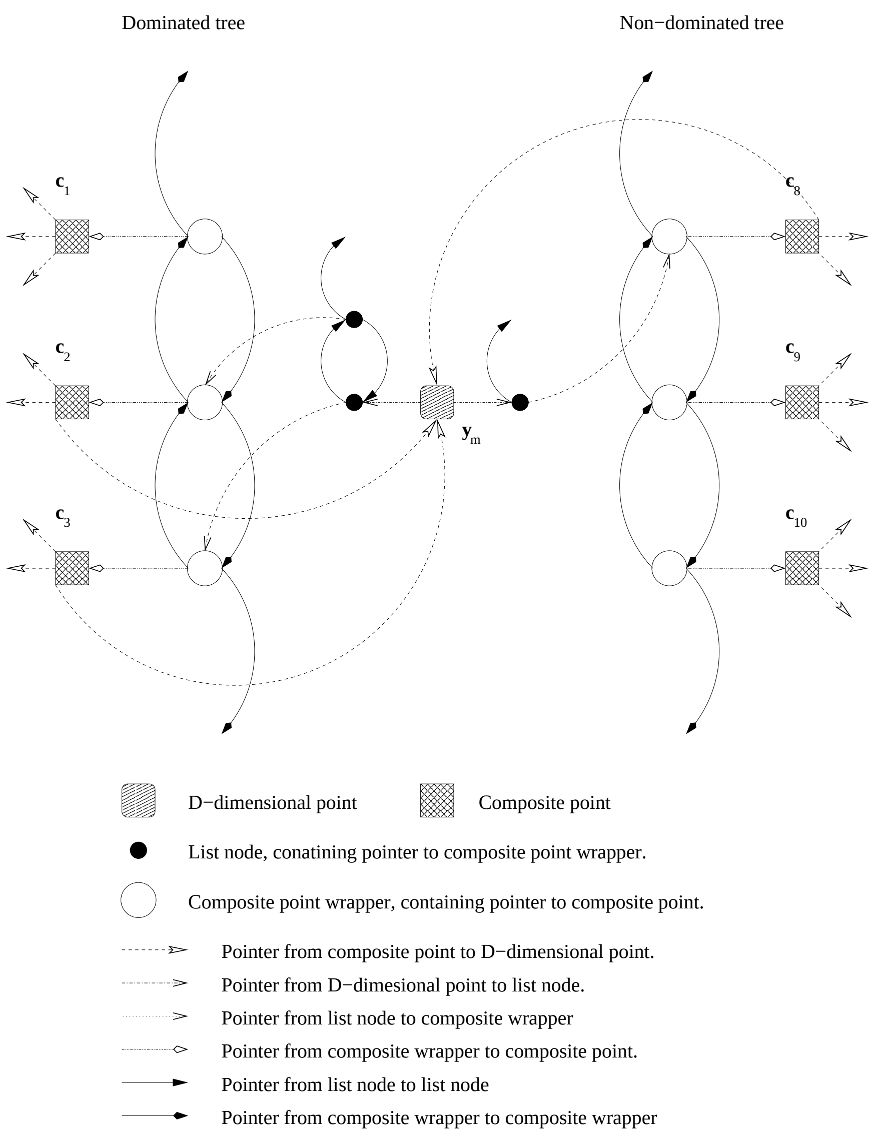

Dominated tree
Non−dominated tree
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathbf{c}_3$
$\mathbf{c}_8$
$\mathbf{c}_9$
$\mathbf{c}_{10}$
$\mathbf{y}_m$
D−dimensional point
Composite point
List node, conatining pointer to composite point wrapper.
Composite point wrapper, containing pointer to composite point.
Pointer from composite point to D−dimensional point.
Pointer from D−dimesional point to list node.
Pointer from list node to composite wrapper
Pointer from composite wrapper to composite point.
Pointer from list node to list node
Pointer from composite wrapper to composite wrapper

Figure 17: Illustration of dominated tree represented in a doubly-linked list.

Chapter 5

An Empirical Validation of the New Data Structures

In this chapter the novel data structures are demonstrated to significantly improve the performance of MOEAs.

5.1 Experimental design

In order to evaluate the efficiency of the new data structures, results are presented of a comparison with the GA and ES MOEAs using (a) full elite archives using PQRS and dominated trees, (b) full elite archives with a standard linear search, and (c) truncated archives with clustering. This allows for the comparison of efficiency in terms of time, between linear lists and the new data structures when using unconstrained archives, and in terms of archive quality, between the algorithms using unconstrained archives and clustered archives. In addition to the standard ZDT test functions in Zitzler *et al.* (2000), new multi-objective test problems are also introduced that have the characteristic of large archive growth, which this work wishes to investigate. However, first a brief critique of the ZDT multi-objective test suite is given.

The ZDT test functions involve two objectives and have the following structure:

$$\begin{array}{lll} \text{Minimise} & T(\mathbf{x}) & = (f_1(x_1), f_2(\mathbf{x})), \\ \text{where} & f_2(\mathbf{x}) & = g(x_2, \ldots, x_P) \cdot h(f_1(x_1), g(x_2, \ldots, x_p)), \\ \text{and} & \mathbf{x} & = (x_1, \ldots, x_P). \end{array}$$

Although the six ZDT functions represent many different features and levels of difficulty, the first objective (f_1) is always a function solely of the first decision parameter (in fact, for the first four test functions, $f_1(x_1) = x_1$). This simple form of f_1 means that when genes describing x_1 are initialised as uniformly distributed random numbers, the initial estimate of the Pareto front extends over the full range of f_1. Consequently optimisation chiefly consists of minimising f_2, rather than the combined minimisation of both objectives.

This structure also appears to produce the artifact that the growth of the frontal set is quite limited, as shown in Figures 18 and 19. As opposed to the experience of other applied studies that have experienced much larger archive growth (e.g. (Parks & Miller, 1998)). In addition, all six test functions are only two objective problems.

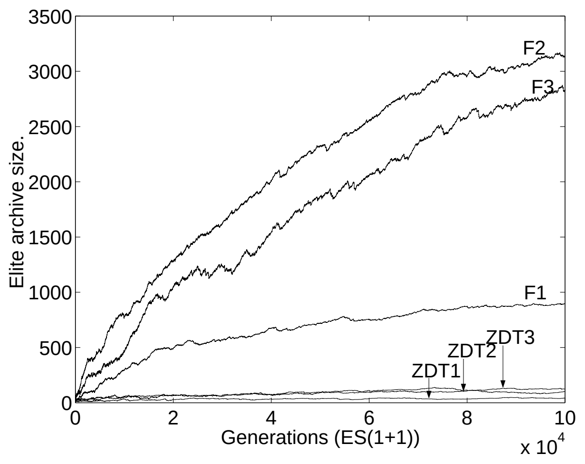


Figure 18: Growth of archive size using ZDT1, ZDT2 and ZDT3 (Zitzler *et al.* , 2000) and the test functions F1, F2 and F3 introduced in this work, using a simple (1+1)-ES based MOEA.

For the purposes of this work, where the focus is on maintaining large archives, the ZDT functions unfortunately are arguably of limited use (note however that Deb *et al.* (2001) have recently

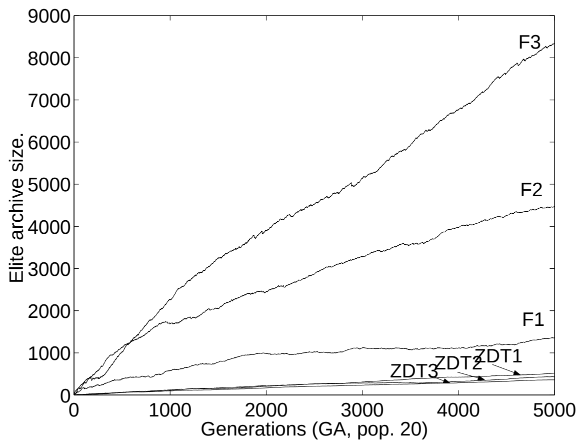


Figure 19: Growth of archive sizes using ZDT1, ZDT2 and ZDT3 (Zitzler *et al.* , 2000) and the test functions F1, F2 and F3 introduced in this work, using a simple GA based MOEA.

introduced additional higher-dimension test functions). The first three ZDT functions are still employed in this chapter as they are well known to the community at large. Their formulation is shown in Table 4

Table 4: First three Test functions from (Zitzler *et al.* , 2000).

#	Function	$\mathbf{x}$
ZDT1	$f_1(x_1) = x_1,$ $g(x_2, \dots, x_N) = 1 + 9\left(\sum_{n=2}^{N} x_n\right)/(n-1),$ $h(f_1, g) = 1 - \sqrt{f_1/g}.$	$N = 30,$ $x_i \in [0, 1].$
ZDT2	$f_1(x_1) = x_1,$ $g(x_2, \dots, x_N) = 1 + 9\left(\sum_{n=2}^{N} x_n\right)/(n-1),$ $h(f_1, g) = 1 - (f_1/g)^2.$	$N = 30,$ $x_i \in [0, 1].$
ZDT3	$f_1(x_1) = x_1,$ $g(x_2, \dots, x_N) = 1 + 9\left(\sum_{n=2}^{N} x_n\right)/(n-1),$ $h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1).$	$N = 30,$ $x_i \in [0, 1].$

In addition three new test functions are introduced, in which all objectives are dependent on all decision parameters. These functions do exhibit large archive growth. All three are combinations of the following five base functions, $B_i(\mathbf{x})$.

- Base functions

$$B_1(\mathbf{x}) = \sum_{i=1}^{m} \left| x_i - \frac{1}{3}\exp\left((i/m)^2\right) \right|^{\frac{1}{2}} \tag{30}$$

$$B_2(\mathbf{x}) = \sum_{i=1}^{m} \left(x_i - \frac{1}{2}\left(\cos\left(10\pi\left(i/m\right)\right) + 1\right) \right)^2 \tag{31}$$

$$B_3(\mathbf{x}) = \sum_{i=1}^{m} \left| x_i - \sin^2(i-1)\cos^2(i-1) \right|^{\frac{1}{2}} \tag{32}$$

$$B_4(\mathbf{x}) = \sum_{i=1}^{m} \left| x_i - \frac{1}{4}\left(\cos(i-1)\cos\left(2\left(i-1\right)\right) + 2\right) \right|^{\frac{1}{2}} \tag{33}$$

$$B_5(\mathbf{x}) = \sum_{i=1}^{m} \left(x_i - \frac{1}{2}\left(\sin\left(1000\pi\left(i/m\right)\right) + 1\right) \right)^2 \tag{34}$$

- Multi-objective functions

$$F_1(\mathbf{x}) = \left(B_1(\mathbf{x}), B_2(\mathbf{x})\right) \tag{35}$$

$$F_2(\mathbf{x}) = \left(B_2(\mathbf{x}), B_3(\mathbf{x}), B_4(\mathbf{x})\right) \tag{36}$$

$$F_3(\mathbf{x}) = \left(B_1(\mathbf{x}), B_3(\mathbf{x}), B_4(\mathbf{x}), B_5(\mathbf{x})\right) \tag{37}$$

In all cases m=30 and $x_i \in [0, 1]$.

Parallel coordinated graphs shown in Figure 20 of these test functions, which illustrate the function bounds and the trade-off between objectives (with the lines crossing), on the true Pareto front.

5.1.1 Algorithm implementation

The implementation of both the ES and GA models use floating-point representation of parameters in the individual chromosomes. In order to compare the linear search and the dominated tree method the two versions of each MOEA were each executed 50 times on each test problem, with the cumulative time saved at each generation (the time spent in the methods dealing with comparison to the archive and selection from the archive). In order to compare the unconstrained and truncated approach, each was executed 50 times on each test problem (with different starting points), and the resultant non-dominated solutions saved at the end of each run. The runs were repeated twice;

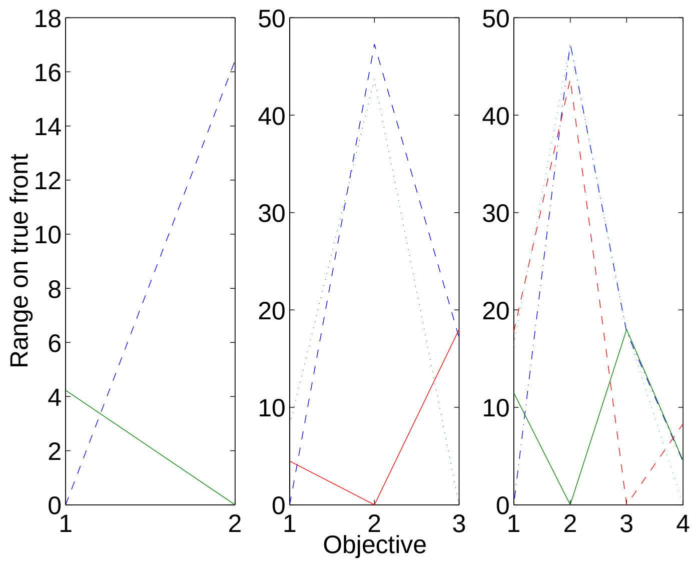


Figure 20: Parallel coordinated graphs of the objective bounds of the test functions (each line relating to an individual which minimises one of the objectives)

once for the same number of generations (function evaluations) and once for the same empirical run time. In the case of the clustering algorithm no off-line store was maintained, as maintaining this store would incur an additional time cost above the standard (unclustered) approach, for which truncation is designed to alleviate. Clustering was performed using the method from SPEA (Zitzler (2000)) Each simulation was performed using the parameters shown below:

GA

Number of generations : 5000

Search population size : $|B_t| = 20$

Max. archive population size (when clustered) : $\widehat{M} = 50$

Crossover rate : 0.8

Mutation rate : 0.05

(1+1)-ES

Number of generations : 100000

Max. archive population size (when clustered) : $\widehat{M} = 50$

Mutation rate : 0.2

Single point crossover was used and the mutator variable was drawn from a Gaussian distribution with 0.1 variance and zero mean. In each of the 50 different runs the MOEAs were initialised from identical decision vector populations of size 100, with the non-dominated individuals residing in these populations forming the initial elite archive. Initialisation of decision vectors was from $\mathcal{U}(0,1)$. The random number generators were identically seeded for each of the 50 runs, so that the fronts found by the two unconstrained methods were identical. This means that time comparisons were made on strictly identical update/selection problems. Elitism intensity was 1.0 and the dominated and non-dominated trees were cleaned when they exceeded $1.2M/D$ composite points. In all algorithms selection of elite individuals from the archive used PQRS.

5.1.2 Results

Timing results

When comparing the speed of the three algorithm types the sum of clock ticks counted when operating in the relevant methods of each algorithm (those that dealt with the comparing of a new individual to the archive (and updating), and the selection of an individual from the archive as a parent) was maintained, with these value transformed into seconds[1].

As shown in Tables 5 and 6, the simple GA using the dominated trees was significantly faster than the standard linear search GA for all six test problems. The ES dominated tree algorithms were significantly faster than the standard linear approach for the three new test problems, however there was no significant difference in speed on the three ZDT functions. The clustering algorithm was faster than both on the 3 and 4-objective problem using the simple ES, and faster than linear or dominated tree searches on all the test functions except F_1 using the GA. This is a reflection of the size of the archive set – see Figures 18 and 19. Table 6 shows the generations at which one algorithm

[1]The machine used in these experiments was a 1.4GHz AMD Athlon processor with 256Mbytes of DDRAM.

Table 5: Mean execution time (in seconds) of the three archive methods, standard deviation in parentheses. Results in bold signify significantly faster results as suggested by the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail) compared to the other two algorithms. Results in italics signify that the dominated tree algorithm is significantly faster than linear search.

		Linear	Dominated tree	Clustered
ES	ZDT1	0.61 (0.06)	0.60 (0.06)	1.04 (0.08)
	ZDT2	0.44 (0.06)	0.45 (0.05)	0.48 (0.07)
	ZDT3	0.59 (0.05)	0.60 (0.06)	1.00 (0.07)
	F1	7.29 (0.18)	***5.62*** (0.21)	48.39 (0.50)
	F2	134.00 (9.56)	*111.80* (7.63)	**77.26** (0.69)
	F3	191.82 (19.21)	*151.25* (14.71)	**78.79** (1.36)
GA	ZDT1	3.11 (0.20)	*1.79* (0.12)	**1.03** (0.08)
	ZDT2	2.44 (0.18)	*1.53* (0.11)	**0.54** (0.07)
	ZDT3	2.36 (0.12)	*1.27* (0.08)	**1.07** (0.08)
	F1	10.38 (0.82)	***7.87*** (0.86)	29.65 (0.65)
	F2	105.07 (12.80)	*81.86* (9.61)	**62.94** (0.96)
	F3	341.54 (38.25)	*245.61* (27.604)	**74.64** (1.961)

tested using the non-parametric Wilcoxon signed rank test, from Wilcoxon & Wilcox (1964), at the 0.05 level (0.025 in each tail)[2] was found to be significantly faster than another. As it can be seen, both the unconstrained algorithms were initially faster than the clustered approach, however, as their archives grew their time cost per generation (the gradient of the cumulative time curves shown in Figure 21), surpassed that of the clustering approach.

It can also be seen that in a number of cases standard linear approach is initially faster than the dominated tree method. This is due to the time cost of maintaining the data structures outweighing the search cost reduction compared to the linear list, when the archive population is small. A number of additional experiments were therefore run where the time taken to reach different archive sizes was assessed. Again 50 separate runs of the two unconstrained approaches (for both the ES and GA MOEA) were made, with the time taken to reach archive sizes in 50 member increments recorded for each of the test functions. As can be seen in Table 7, for the ZDT functions it is generally the case that after the archive has exceeded 50-100 members the dominated tree method is significantly faster than the linear list. A higher threshold is experienced by the new functions introduced in this work, with the dominated tree not being significantly faster until the archive reaches 150-250

[2]t-tests cannot be used as the samples cannot reasonably be assumed to be drawn from normal distributions, neither are the samples independent (each pair being correlated). The Wilcoxon signed rank test does not assume normality, and its independence assumption is only that each pair is independent of all other pairs.

Table 6: Generations (multiples of 1000) for which algorithm A is significantly faster than algorithm B. $t(A, B)$, calculated using the Wilcoxon non-parametric signed ranks test at the 0.05 significance level with 0.025 in each tail. Where L denotes the standard unconstrained archive with linear search, D is the dominated tree archive and C is the clustered archive.

		$t(L,D)$	$t(L,C)$	$t(D,L)$	$t(D,C)$	$t(C,L)$	$t(C,D)$
ES	ZDT1	1-48	1-100	-	1-100	-	-
	ZDT2	1-64	1-100	-	1-100	-	-
	ZDT3	1-64	1-100	-	1-100	-	-
	F1	1-5	1-100	10-100	1-100	-	-
	F2	-	1-53	2-100	1-61	57-100	66-100
	F3	-	1-44	3-100	1-50	48-100	54-100
GA	ZDT1	1-16	-	22-100	-	12-100	1-100
	ZDT2	2, 4-10	-	22-100	-	4-100	2-100
	ZDT3	1-15	15-19	22-100	20-57	30-100	1-11, 71-100
	F1	1-3	1-100	8-100	1-100	-	-
	F2	-	1-56	2-100	1-69	62-100	76-100
	F3	-	1-26	3-100	1-31	29-100	35-100

members. This is thought to be because the test function surfaces are searched in all objective directions (unlike the ZDT functions for which optimisation principally consists of 'pushing down' the second objective), and as such have a higher proportion of individuals discovered at the edges of the frontal set at each generation (meaning the number of composite points to be searched linearly is on average larger).

Table 7: Table showing archive size beyond which the dominated tree is significantly faster than the linear search for the various test problems for the total (absolute) time cost of the method up to that archive size, the second column pair are for the incremental cost (the difference between the time taken in reaching one range and the next). Significance calculated using a Wilcoxon signed ranks test (0.025 in each tail).

	Trees Sig. Faster			
	Cumulative		Incremental	
	ES	GA	ES	GA
ZDT1	150	150	100	100
ZDT2	150	150	50	50
ZDT3	250	150	100	50
F1	450	300	250	150
F2	550	400	200	150
F3	400	350	200	150

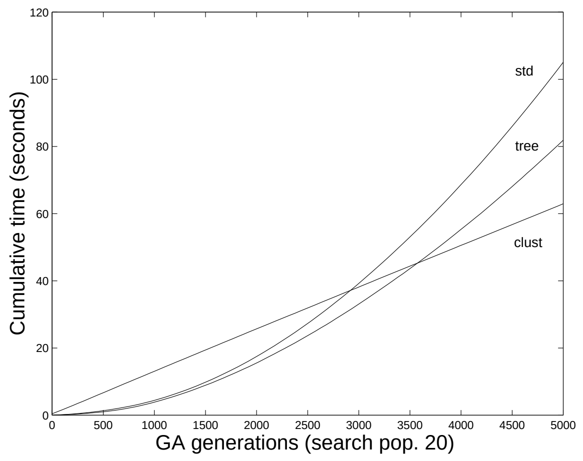


Figure 21: The average time (in seconds) spent checking an individual against the elite archive and updating the archive in the three different archive methods (unconstrained standard, unconstrained with the new data structures and clustered), using the GA up to 5000 generations (with the three objective problem F_2).

Performance results

Table 8 shows the $\tilde{\mathcal{C}}$ performance comparison of the fronts found by the unconstrained and truncated algorithms after 100000/5000 ES/GA generations. On all six test functions the unconstrained approach performed significantly better than the constrained approach using the GA over the 50 runs, both in terms of equal generation length and equal computation time. The ES unconstrained method performed significantly better than the clustered method using this measure on test functions F_1, F_2, and F_3 on equal generation length, the clustered method being significantly better on ZDT2 and ZDT3. However, when the clustered algorithm was run for the same amount of computation time as the different dominated tree runs, the unconstrained method was significantly better than the constrained method for all the test functions except ZDT2 (where there was no observed significant difference between the two methods over the 50 runs). This result however is simply due to the slower archive increase experienced for this test problem; meaning that although the marginal cost of each extra generation was more for the standard linear list storage method at generation 5000, the algorithms where not run for long enough for this to transfer to a significant difference on the total time run (due to the relative efficiency of the linear list at low archive sizes).

These results are further supported by those using the $\mathcal{V}$ measure (Table 9). Fronts found by

Table 8: Comparison between end-of-run fronts from the unconstrained and clustered elite ES and GA archive models, using the $\tilde{\mathcal{C}}$ measure. $\tilde{\mathcal{C}}(U, C)$ is the mean proportion of the members of the clustered generated front dominated by members of the unconstrained generated front. Means are over 50 runs, with standard deviation in parentheses. The first two columns relate to the results after an equal number of generations. The third and fourth columns relate to the results after the clustered algorithms have run for an equal time as the dominated tree based unconstrained algorithm. Results in bold signify significantly better results under the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail).

		Equal generations		Equal time	
		$\tilde{\mathcal{C}}(U, C)$	$\tilde{\mathcal{C}}(C, U)$	$\tilde{\mathcal{C}}(U, C)$	$\tilde{\mathcal{C}}(C, U)$
ES	ZDT1	0.318 (0.142)	0.408 (0.172)	**0.636** (0.173)	0.163 (0.161)
	ZDT2	0.337 (0.156)	**0.465** (0.180)	0.383 (0.175)	0.413 (0.179)
	ZDT3	0.244 (0.114)	**0.382** (0.135)	**0.417** (0.152)	0.245 (0.152)
	F1	**0.899** (0.030)	0.005 (0.001)	**0.999** (0.003)	0.000 (0.001)
	F2	**0.894** (0.035)	0.001 (0.001)	**0.878** (0.039)	0.002 (0.001)
	F3	**0.816** (0.048)	0.003 (0.004)	**0.787** (0.055)	0.005 (0006)
GA	ZDT1	**0.981** (0.012)	0.000 (0.000)	**0.982** (0.011)	0.000 (0.000)
	ZDT2	**0.982** (0.010)	0.000 (0.000)	**0.981** (0.011)	0.000 (0.000)
	ZDT3	**0.984**(0.011)	0.000 (0.000)	**0.984** (0.011)	0.000 (0.000)
	F1	**0.995** (0.010)	0.001 (0.002)	**0.996** (0.010)	0.000 (0.001)
	F2	**0.947** (0.025)	0.000 (0.001)	**0.935** (0.030)	0.000 (0.001)
	F3	**0.962** (0.029)	0.000 (0.001)	**0.921** (0.032)	0.000 (0.001)

the unconstrained full archive are significantly ahead of the clustering truncated method using the GA, for both equal generations and equal computation time. Using the ES scheme the fronts found by the dominated tree method are significantly better than those found by clustering for all test functions except ZDT2, where there was no significant difference between the two methods.

5.2 Key results

New data structures and methods – PQRS and dominated trees – have been introduced to facilitate faster performance when maintaining an unconstrained archive of non-dominated solutions. Although dominated trees are usually used for maintaining a set of mutually non-dominating elements, it is to be emphasised that there is no restriction to using them only with mutually non-dominating sets. Unconstrained archives themselves are necessary to prevent oscillations and retreat of the frontal set, problems which beset MOEAs in which the elite archive is truncated. The problem of oscillating fronts was demonstrated empirically in this thesis for the first time.

Table 9: Comparison between end-of-run fronts from the unconstrained and clustered elite ES and GA archive models, using the $\mathcal{V}$ measure. Where $\mathcal{V}(U,C)$ is the mean proportion of the volume of the minimum hypercube containing both estimated fronts, which is dominated by members of the unconstrained generated front but not by members of the constrained generated front. Means are over 50 runs, standard deviation in parentheses, value as a percentage. The first two columns relate to the results after an equal number of generations. The third and fourth columns relate to the results after the clustered algorithms have run for and equal amount of time as the data structure based unconstrained algorithm. Results highlighted in bold signify significantly better results under the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail).

		Equal generations		Equal time	
		$\mathcal{V}(U,C)$	$\mathcal{V}(C,U)$	$\mathcal{V}(U,C)$	$\mathcal{V}(C,U)$
ES	ZDT1	**0.760**% (0.360)	1.260% (4.810)	**1.650**% (0.530)	0.810% (4.550)
	ZDT2	0.802% (0.447)	1.021% (0.618)	0.957% (0.512)	0.836% (0.549)
	ZDT3	**0.581**% (0.320)	1.151% (5.530)	**0.912**% (0.466)	0.944% (5.374)
	F1	**9.287**% (0.708)	0.014% (0.031)	**4.601**% (0.288)	0.000% (0.000)
	F2	**9.466**% (0.669)	0.019% (0.011)	**5.614**% (0.413)	0.025% (0.018)
	F3	**11.215**% (0.748)	0.224% (0.115)	**8.517**% (0.765)	0.300% (0.161)
GA	ZDT1	**5.829**% (0.374)	0.000 (0.000)	**5.423**% (0.413)	0.000 (0.000)
	ZDT2	**8.882**% (0.762)	0.000 (0.000)	**7.617**% (0.666)	0.000 (0.000)
	ZDT3	**3.612**% (0.359)	0.000 (0.000)	**3.713**% (0.403)	0.000 (0.000)
	F1	**5.146**% (0.439)	0.000% (0.000)	**5.553**% (0.400)	0.000 (0.000)
	F2	**8.651**% (0.492)	0.004% (0.013)	**8.649**% (0.563)	0.006% (0.009)
	F3	**13.770**% (0.962)	0.001% (0.002)	**12.514**% (0.978)	0.046% (0.095)

The dominated trees reduce the time taken using an unconstrained archive, in both simple ES and GA MOEAs, although when the archive is relatively small (below approximately 50 members) the cost of maintaining these structures negates the decreased search time. For the test sets used in this thesis even when running the (typically) quicker clustering algorithm for the same time as a simple MOEA with an unconstrained archive and dominated trees, the unconstrained algorithm still produces superior results. The precise impact of dominated trees on the overall run time of an algorithm will, of course, depend upon the complexity of test function itself and the complexity of the fitness assignment/selection procedure used.

The exact complexity of the search of a dominated tree is dependent on the number of individual composite points to be checked linearly; as such the data structure described is affected more by individuals that lie on the extremes of the estimated Pareto front. This is shown empirically in Table 7, with a larger archive size needed for the test functions introduced in this thesis than the ZDT functions before the data structures are found to be significantly faster than the linear search.

In extreme circumstances it still may be the case that the maintenance of all frontal solutions may be infeasible (simply in memory requirements); in such cases the data structures introduced here could be used in tandem with an unconstrained ϵ-approximate Pareto set.

In Laumanns *et al.* (2001) the problem of oscillating fronts is addressed by the development of the ϵ-dominance concept and ϵ-Pareto sets. In this approach the objective values of a solution are calculated as usual, but an additional D ϵ-Pareto objective values are also calculated, which are used in the comparison of individuals. If the objectives are to be minimised these are

$$y_i^\epsilon = f_i(\mathbf{x}) \cdot (1 - \epsilon)\,, \qquad i = 1, \ldots, D \tag{38}$$

where ϵ is pre-set by the user, and determines how much a new solution has to be better than a previous archived solution in order to be archived. An illustration of this is shown in Figure 22.

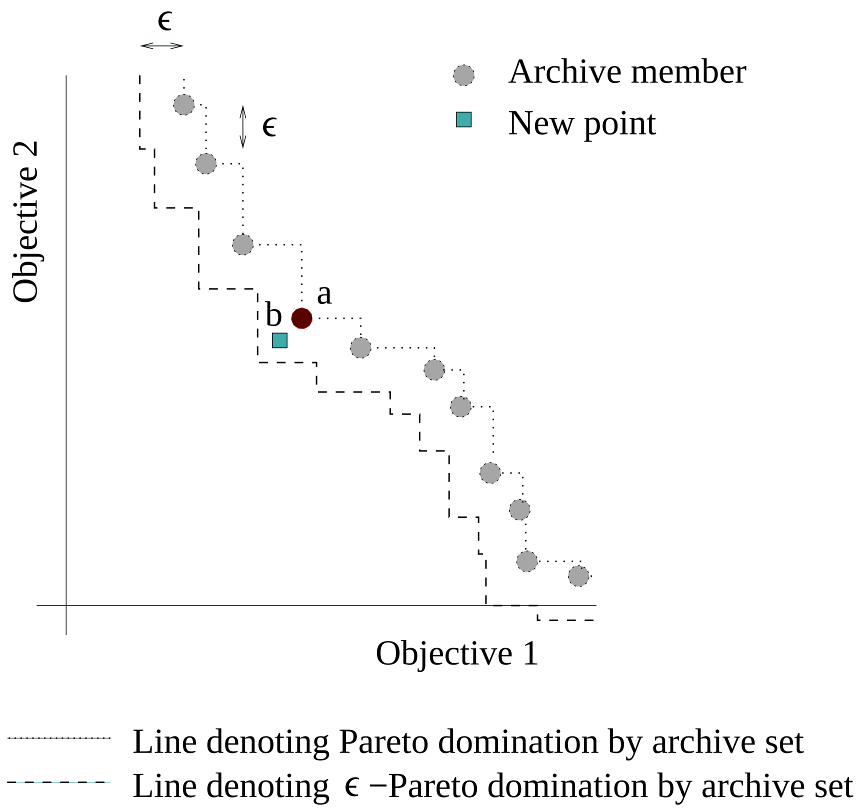

ϵ
ϵ
Archive member
New point
Objective 2
a
b
Objective 1
Line denoting Pareto domination by archive set
Line denoting ϵ −Pareto domination by archive set

Figure 22: Illustration of ϵ-dominance, although the new point b dominates the archive member it does not ϵ dominate it, and therefore is not stored.

On initialisation the objective values of the seed solutions are calculated, and the non-dominated individuals are stored in terms of their ϵ-objectives (an initial ϵ-Pareto set). On the generation of new individuals for insertion into the ϵ-Pareto set, comparison is based on the new solution's objective $\mathbf{y}$ and the stored solution's $\mathbf{y}^{\epsilon}$. Only if the new solution is not ϵ-dominated by any element of the set will it be inserted into the archive, its ϵ-objectives are then stored and any ϵ-dominated individuals removed. This is illustrated in Figure 22, although new point b dominates the archived point a, it does not ϵ-dominate it, and therefore it is discarded. Laumanns *et al.* (2001) show that a result of this is an ϵ dependent bound on the size of the ϵ-Pareto set. However this is at a cost; the final estimated front may lie behind the true Pareto front in each dimension by a factor of $(1+\epsilon)$. In addition, although overt oscillation of the Pareto set (in its ϵ-Pareto form) is removed, oscillation (and decreased search efficiency) will still occur. This is because solutions that dominate, but do not ϵ-dominate the ϵ-frontal set, are not added to the ϵ-Pareto set and therefore have to be rediscovered. The empirical impact of this oscillation and that of restricting forward movements of the ϵ-Pareto set to steps of greater than ϵf_i has not been reported. If the ϵ-dominance approach were to be adopted, a potentially time consuming search of the ϵ-Pareto set would still be necessary when inserting a new solution, and therefore the data structures introduced here would be useful.

5.3 Further implications: robust stopping criteria

Robust stopping criteria are largely missing from the MOEA literature. Beale & Cook (1978) include a fitness-based stopping criterion in their study, in which the algorithm is terminated if the fittest individual has remained unchanged for 1000 consecutive generations. As Coello (1999) points out, MOEAs since Schaffer (1985), which carry a set of non-dominated individuals, are usually terminated after a fixed number of generations, or the population is monitored at intervals and a decision made on a visual basis.

The use of an elite archive which can only 'advance' however allows a number of robust stopping methodologies to be introduced. Examples of these are:

- When no individual that dominates a member of F is discovered after a given number of consecutive generations.

This is similar to the approach taken by Beale & Cook (1978), however in a MOEA there is a set of elite solutions instead of a single elite individual. It is to be emphasised that this sort of criterion can fail with an truncated archive which is prone to oscillation. Note that new individuals may be found that are non-dominated by the frontal set - and therefore are inserted into F. However, these individuals may only fill in the front (increasing its resolution), rather than pushing it forward.

- When there has been no change in the extremal values for a given number of consecutive generations.

The previous stopping criterion, when taken by itself, may lead to sub-optimal stopping in that, although the front may have ceased moving forwards, it may still be moving outwards toward the extremes. This second criterion takes this form of search into account and can usefully be combined with the previous criterion.

- When the average distance in objective space between neighbouring individuals in F reaches a specific threshold.

The practitioner may wish the front to be defined to a particular resolution, therefore they may not solely wish to stop the search process after the front has finished moving, but prefer to wait until this resolution is reached. With two objectives, the resolution can be estimated by calculating the maximum over F of the nearest neighbour distances[3]. With more than two objectives a similar termination criterion can be defined based on the maximum area of any triangle of a Delauney tessellation (de Berg *et al.* , 1997) of F.

In practice the third criterion alone may lead to stopping before a good estimate of the true Pareto front has been found (if the resolution is set too coarse), however in conjunction with the first two criteria a good estimate of the true Pareto front to any desired resolution can be achieved.

If a practitioner prefers a small number of evenly distributed individuals to be returned after algorithm termination, the final elite archive may be clustered using the method employed in standard SPEA (the computational cost is not too high as it only needs to be performed once).

The stopping criteria defined above are not readily applied to methods which do not use an unconstrained active elite archive because, even if a dormant offline store is used, these methods are susceptible to oscillating active estimates of the Pareto front, which may cause spurious early

[3]This can fail in the pathological case that the true front contains isolated points in objective space.

termination. (An alternative discussion on stopping techniques for MOEAs, through the use of multiple run-time error metrics, can be found in Deb & Jain (2002)).

In the previous three chapters generally applicable data structures for multi-objective optimisation have been developed. The focus of the thesis shall now turn to the development of novel multi-objective optimisation algorithms that are dependent specifically on the properties of these data structures to impove not just their empirical speed, but their *search* processes too.

Part II

Dominated Tree based Multi-Objective Particle Swarm Optimisation

In this second part, the particle swarm optimisation heuristic is described, which has recently found popular use as a NN optimiser. A novel extension to the general model which facilitates the optimisation of multiple objectives is introduced. This new approach relies on the new data structures developed in Part I of the thesis. The new method is demonstrated to be consistently superior to another multi-objective particle swarm optimisation method also published in 2002 (Coello & Lechunga, 2002) and a recent MOEA (Knowles & Corne, 1999; Knowles & Corne, 2000) on a number of well known test functions.

Additional experiments show that the introduction of a stochastic 'turbulence' term within MOPSO improves performance across models. Sixteen different methods of MOPSO are then compared, some included from the literature and others developed in the thesis, and from this empirical evaluation it is shown that a hierarchy of models exists. In addition the benefits of this new method in comparison to MOEA methods is further underlined. The types of problem where the new method excels and where it has limitations are highlighted, with a discussion as to why this may be the case.

Chapter 6

Particle Swarm Optimisation

In this chapter particle swarm optimisation is introduced, some of its common variants described, and its recent applications to NN training highlighted.

6.1 The heuristic and standard algorithm

The Particle Swarm Optimisation heuristic (PSO) was first proposed by Kennedy and Eberhart (1995) for the optimisation of continuous non-linear functions. This in turn has been the catalyst for several papers in the last few years, (Carlisle & Dozier, 2001; Conradie *et al.* , 2002b; Conradie *et al.* , 2002a; Duch & Korczak, 1999; Duch, 1999; Engelbrecht & Ismail, 1999; Ismail & Engelbrecht, 1999; Ismail & Engelbrecht, 2000; Kennedy & Spears, 1998; Lovbjerg *et al.* , 2001; Ozcan & Mohan, 1999; Parsopoulos & Vrahatis, 2001; Parsopoulos *et al.* , 2001a; Parsopoulos *et al.* , 2001b; Parunak *et al.* , 2001; Shi & Eberhart, 1998; van den Bergh, 1999; van den Bergh & Engelbrecht, 2000; van den Bergh & Engelbrecht, 2001a; van den Bergh & Engelbrecht, 2001b; Zhang & Shoa, 2001), which have covered both the theoretical expansion of the subject area, and the empirical/theoretic comparison to existing EC techniques and specific applications. PSO is of specific interest to this thesis since: (a) given its representational similarities to EC approaches, studies have highlighted its relative speed and effectiveness compared to EC approaches in the uni-objective domain, the development of a multi-objective variant, though not trivial, does strike as a 'natural' progression and (b) one of the most popular applications of the method has been NN optimisation (Conradie *et al.* , 2002b; Conradie *et al.* , 2002a; Duch, 1999; Duch & Korczak, 1999; Engelbrecht & Ismail, 1999; Ismail & Engelbrecht, 1999; Ismail & Engelbrecht, 2000; van den Bergh, 1999; van den Bergh & Engelbrecht, 2000; van den Bergh & Engelbrecht, 2001b; Zhang & Shoa, 2001). With this in mind, the PSO method in its general and advanced forms will now be described.

In PSO a solution is represented by a vector of floating point numbers (in a similar fashion to ES), such that in n-dimensional decision space the ith 'particle' is represented as $X_i = \mathbf{x} = (x_1, x_2, \ldots, x_n)$, which resides in a population (called swarm) of m individuals $X = \{X_1, X_2, \ldots, X_m\}$. These decision vectors/particles are kept in a population of fixed size. However this is where the similarity to traditional evolutionary motivated algorithms ends. PSO was inspired by the flocking behaviour of birds and the swarming behaviour of insects; they have an innate ability to search as a group (a so called *social* element, Ozcan & Mohan (1999)) but also as an individual (a so called *cognition* element Ozcan & Mohan (1999)). Each particle in PSO has its parameters altered at each generation in an attempt to 'fly' them toward better solutions, weighting this flight by some random variable in order to promote search through the process of over/under shooting. The swarm interactions at the social and cognition levels are aimed at creating a balance between staying close to the optimum, whilst searching areas of local interest in the search space. In attempting to emulate this behaviour the individual solutions in PSO have a number of additional properties. They maintain a copy of the decision parameters of the fittest 'position' that they have recorded in previous iterations $P_i = \mathbf{p} = (p_1, p_2, \ldots, p_n)$ - therefore encapsulating basic cognition, a 'memory'. They also maintain a copy of their previous velocity in different directions of decision space $V_i = \mathbf{v} = (v_1, v_2, \ldots, v_n)$ - that is the difference between their current n-dimensional position, and that at the previous generation. Using notation consistent with that used in Part I, the general algorithm for updating these velocities (for 'flying' the swarm from one generation to the next) is:

$$V_i := wV_i + c_1\mathbf{r}_1(P_i - X_i) + c_2\mathbf{r}_2(P_g - X_i) \tag{39}$$

$$X_i := X_i + \chi V_i \tag{40}$$

Where $w, c_1, c_2, \chi \geq 0$. w is the inertia of a particle (how much its previous trajectory affects its new velocity), c_1 and c_2 are constraints on the velocity toward global best (*gbest*), indexed by g, and previous best (*pbest*) of a particle. These are most commonly fixed at 1 or 2 (Coello & Lechunga, 2002; Kennedy & Eberhart, 1995), although more explicit guidelines are presented in a recent study (Shi & Eberhart, 1998). The smaller the inertia the more the algorithm will tend toward convergence as the velocity will have a proportionally higher weighting toward the current global and personal best particle positions. The high inertia values promote a greater degree of search as a particle has a greater probability of searching around less fit areas of decision space it may find itself in through the previous positions encapsulated in V). As such the inertia value is sometimes adjusted as the search progresses, starting at a high level and then decreasing over time. χ (the construction coefficient, Lovbjerg *et al.* (2001)) is a constraint on the overall shift in position and $\mathbf{r}_1, \mathbf{r}_2 \sim U(0,1)$. In the original study by Kennedy and Eberhart (1995) both the inertia and construction coefficient were fixed at 1. The velocity at the start of the swarming process is typically either random, or set at zero.

6.2 Common extensions

A large amount of work has consolidated the base PSO model since its inception, considering its relative youth as a technique. This may well be due to its intuitive appeal and ease of coding. Some of the most popular variants will now be described.

6.2.1 Multi-swarm PSO

In multi-swarm PSO, a number of sub-swarms are used that are concerned with optimising particular parts of the decision space, van den Bergh & Engelbrecht (2001a; 2001b), (called cooperative particle swarm optimisation, CPSO), or swarming separately with sporadic information transfer, Lovbjerg *et al.* (2001), which is synonymous to the island models from the GA literature. Indeed, the subpopulation method in Lovbjerg *et al.* (2001) uses GA crossover as this source of information transfer between its swarms. As indicated, CPSO, separates the decision vector into a number of sub-vectors, each represented by an individual swarm, with solutions generated by the concatenation of these vectors. Another version combines both of these approaches, using both standard PSO and CPSO in parallel, Bergh & Engelbrecht (2001a), with the global best information shared between both algorithms at the end of each generation.

6.2.2 Local best PSO

Instead of moving a particle in part toward a global best solution (in terms of the best individual found entire population so far), a local best solution is used (termed *lbest*), such that a particle it is trajected toward the fittest of its nearest swarm neighbours (Hu & Eberhart, 2002). Therefore in this variant Equation 39 is replaced with

$$V_i := wV_i + c_1\mathbf{r}_1(P_i - X_i) + c_2\mathbf{r}_2(X_{(i)} - X_i) \tag{41}$$

where the parenthesis around i in $X_{(i)}$ indicate the the particle that is nearest (in objective space) to X_i.

6.2.3 Hybrid PSO

The final popular extension is the fusion of PSO with an existing EA technique. For instance in Conradie *et al.* (2002b) two hybrid PSO models are developed, one combined with GA techniques, and one using sub-populations (as discussed in Section 6.2.1). These were then compared to standard PSO and GA on a number of test functions. In the hybrid PSO/GA model, after 'flying' the particles, a number of the swarm are crossed-over (with a certain probability) using for instance the arithmetic crossover function (Lovbjerg *et al.* , 2001). Although not found to be in general better than the comparative models in Conradie *et al.* (2002b), the hybrids were significantly better on the multi-modal test problems encountered, highlighting some problems that PSO can tend to have with this type of problem (which will be discussed further in Chapters 7 and 8, in the context of multi-objective PSO search).

6.3 Previous neural network applications

As stated earlier in this chapter, PSO has been used by a number of studies in the training of NNs. The approaches used fall into three broad categories. These are as follows:

6.3.1 Partial weight training

In partial weight training, PSO is used in conjunction with another NN optimisation method, either in tandem or for the initial/end training. For instance in van den Bergh (1999) PSO is used to find the best initial training weights for a network, whose training is then competed with gradient descent methods. In Conradie *et al.* (2002b) on the other hand PSO is used in tandem with an EA, whereas in Conradie *et al.* (2002a) although the NNs are initially trained with traditional gradient descent, PSO is then used for the online adaptation of weights (as the underlying process being modelled changes over time, in the case of the study a non-linear bioreactor).

6.3.2 Full weight training

In the second approach PSO is solely used for determining the NN weight parameters. PSO has already found a niche in this area for product unit MLPs (Engelbrecht & Ismail, 1999; Ismail & Engelbrecht, 1999; Ismail & Engelbrecht, 2000) where the the fitness surface is prone to be very disjunct. In the studies mentioned above it has been found to be better than gradient descent and random search approaches (in terms of function evaluations needed to reach the fittest solutions), and of a similar standard to GAs. In van den Bergh & Engelbrecht (2000; 2001b) PSO and CPSO where compared with respect to their evaluated performance, with training product unit NNs. CPSO proved to be consistently superior with an average of one swarm per five parameters proving optimal, although in van den Bergh & Engelbrecht (van den Bergh & Engelbrecht, 2001a) (a later study by the same authors) it was recommended that a combination of both the subpopulation methods used in van den Bergh & Engelbrecht (van den Bergh & Engelbrecht, 2000) should be used, as opposed to purely CPSO by itself.

6.3.3 Architecture optimisation

In Zhang and Shoa (2001), two separate PSO swarming techniques are used. The first PSO swarm is concerned simply with varying the network architectures of a population of heterogeneous NNs. After each generation of this manipulation however, each particle of this swarm has another swarm seeded from it with homogeneous architectures (inherited from its ‘parent’ particle). These swarms’ sole concern is to optimise the weights of that particular architecture. The best particle of these sub populations after training is then inserted back into the original population and the process is repeated.

After this general overview of PSO, the next chapter will introduce a multi-objective variant, using additional properties of the novel data structures from Part I to enable the selection of a global best individual from the archive for each member of X.

Chapter 7

Directed Multi-Objective Particle Swarm Optimisation

In this chapter a novel method for multi-objective optimisation using PSO is introduced. This new method is dependent on the data structures introduced in Part I, and is shown to be significantly better than the current best performing multi-objective PSO model in the literature, Coello & Lechunga (2002), which in turn has outperformed a number of recent MOEAs (Deb *et al.* , 2000; Knowles & Corne, 1999; Knowles & Corne, 2000).

7.1 MOPSO

Until recently PSO had only been applied to single objective problems. However as discussed in depth earlier several problems are multi-objective. PSO has, in its short history, a number of studies which highlight its use in NN optimisation. Therefore the development of this technique, firstly to effectively handle general multiple objectives, and then to apply this to NN optimisation seems a natural progression, alongside the parallel development of GA and ES techniques.

In the year 2002 the first part of this progression seemed to occur, with a number of different studies published on multi-objective particle swarm optimisation (MOPSO) (Coello & Lechunga, 2002; Hu & Eberhart, 2002; Parsopoulos & Vrahatis, 2002)[1]. However, although most of these

[1]In addition a study by the author, Fieldsend & Singh (2002a), was also published in 2002, based upon the work

studies were generated in parallel with the same overall aim, each implements MOPSO in a different fashion. Given the wealth of MOEAs in the literature this may not seem particularly surprising, however the PSO heuristic puts a number of constraints on MOPSO that MOEAs are not subject to. In PSO itself the swarm population is fixed in size, and its members cannot be replaced, but only adjusted by their *pbest* and the *gbest*, which are themselves easy to define. However, in order to facilitate a multi-objective approach to PSO a set of non-dominated solutions (the best individuals found so far using the search process) must replace the single global best individual in the standard uni-objective PSO case. In addition, there may be no single previous best individual for each member of the swarm. Interestingly the conceptual distinction between *gbest* and *lbest* tends to get blurred in the multi-objective application of PSO. A local individual may be selected for each swarm member, however these *lbest* individuals may all also be non-dominated (representing local areas of the estimated Pareto front maintained by the swarm), also making them all *gbest*. Choosing which *gbest*, *lbest* and *pbest* to direct a swarm member's flight therefore is not trivial in MOPSO. The principle divergence within (Coello & Lechunga, 2002; Hu & Eberhart, 2002; Parsopoulos & Vrahatis, 2002) has therefore been on how these are selected, with a separate divergence on whether an elite archive is maintained.

7.2 Previous studies

In this section brief descriptions and critiques of preceding works in this area are provided.

7.2.1 Hu and Eberhart

A considerable degree of *a priori* knowledge in terms of test function properties is used in the implementation of the $D = 2$ MOPSO in Hu & Eberhart (2002). Instead of a single *gbest*, a local *lbest* is found for each swarm member selected from the 'closest' two swarm members. The concept of closeness is calculated in terms of only one of the evaluated objective dimensions, with the selection of the local optima from the two based upon the other objective. The selection of which objective to fix (used to find the 'closest') and which to optimise is based on the knowledge of the test function design – the relatively simple objective function being fixed. This is shown in Figure 23 with the nearest particles to b highlighted (in terms of the 'simpler' objective 2), meaning that the *lbest* for b

in this chapter.

is c (the fitter of the two neighbours in terms of objective 1). A single *pbest* is maintained for each swarm member, which is only replaced when a new solution is found which dominates it (identical to the 'conservative' preservation of efficiency selection rule described by Hanne (1999), although not referenced in Hu & Eberhart (2002)). This is demonstrated in Figure 23 with particle a moving to a fitter position at generation $t+1$ (one that dominates its previous position). This new position is mutually non dominating with the *pbest* of a, however, as the multi-objective evaluation of the new particle does not lie in the lower quadrant of the *pbest* (represented in Figure 23 with a square), $\mathbf{P}_a$ remains unchanged.

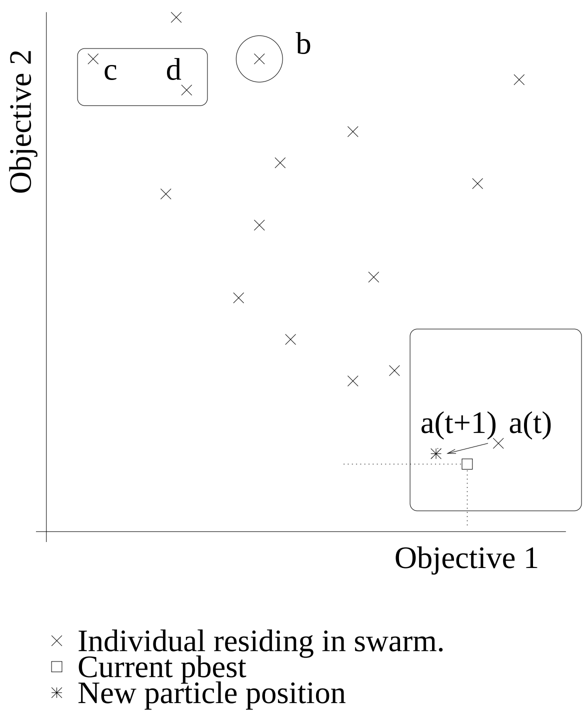


Figure 23: The multi-objective particle swarm optimisation method of Hu & Eberhart (2002).

The performance of the MOPSO was demonstrated on a number of test functions from the literature (including the ZDT test functions from Zitzler *et al.* (2000)), however no comparison was made with any other models, or the true Pareto fronts for the problems.

7.2.2 Parsopoulos and Vrahatis

Parsopoulos & Vrahatis (2002) introduce two methods that use a weighted aggregate approach and another that is loosely based on the VEGA MOEA by Schaffer (1985). These were compared on a number of two dimensional problems. In the first two approaches the weighted aggregate algorithms needed to be run K times to produce K estimated Pareto optimal points (meaning that each run had a single global best). Although Parsopoulos & Vrahatis (2002) state that this approach has a low computational cost, the need for a separate run for each solution found does not necessarily support this. Their final method - the vector evaluated particle swarm optimiser (VEPSO), uses one swarm for each objective (as illustrated in Figure 24, where the two swarms are shown pushing toward the opposite axis). The model is inspired by the original 1985 VEGA model developed by Schaffer (1985) and discussed in Chapter 3. The best particle of the second swarm was used to determine the velocities of the first swarm (act as its global best), and vice-versa.

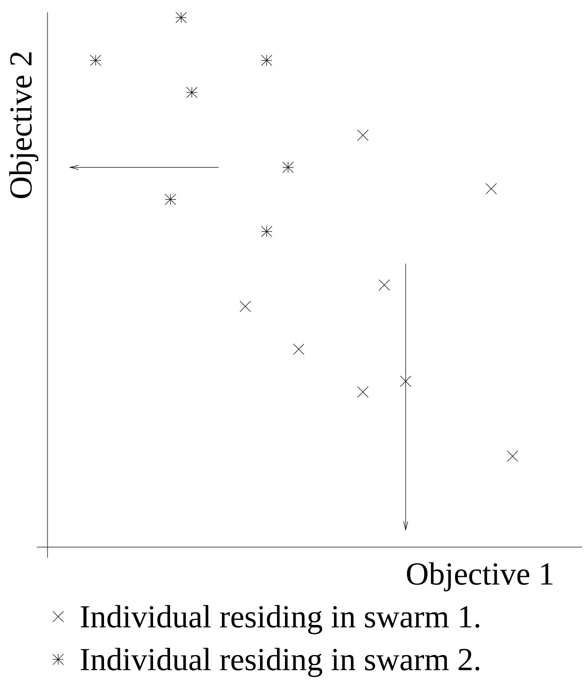

Objective 2
Objective 1
× Individual residing in swarm 1.
✳ Individual residing in swarm 2.

Figure 24: The multi-objective particle swarm optimisation model of (Parsopoulos & Vrahatis, 2002).

Comparison between the algorithms was qualitative (based on visual inspection of the found fronts), with no comparison made to recent competitive methods in the MOEA domain. In addition the current VEPSO model is only designed for $D = 2$ problems.

7.2.3 Coello and Lechunga

In the previous two studies the maximum number of estimated Pareto points returned at the end of the search process equalled the swarm size, meaning large swarms were typically used. In comparison Coello & Lechunga (2002) propose a method which is inspired by more recent developments in the MOEA literature. Two repositories are maintained in addition to the search population. The first is an archive of the global best individuals found so far by the search process, F, and the second containing a single local best for each member of the swarm. A truncated archive is used to store the (global) elite individuals. This archive uses the method of Knowles & Corne (2000) to separate the objective function space into a number of hypercubes (an adaptive grid), with the most densely populated hypercubes truncated if the archive exceeds its membership threshold. The archive also facilitates the selection of a global best for any particular individual in Coello & Lechunga (2002). A *fitness* value is given to each hypercube that contains archive members, equal to dividing 10 by the number of resident particles. Thus a more densely populated hypercube is given a lower score. This is illustrated in Figure 25.

Selection of a global best for a particle is then based on roulette wheel selection of a hypercube first (according to its score), and then uniformly choosing a member of that hypercube. This method therefore biases selection toward under-represented areas of the estimated Pareto front (unlike the original method developed in Knowles & Corne (2000)). Only one local best solution is maintained for each swarm member. However, if a particle X_i is evaluated and found to be mutually non-dominating with P_i, then one of the two is randomly selected to be the new P_i.

An illustration of the swarm is shown in Figure 26, where once again particle a is highlighted in its generational move. However in this model, unlike Hu & Eberhart (2002), $a(t+1)$ has a 50% probability of becoming the new *pbest* of a.

The MOPSO method in Coello & Lechunga (2002) was compared with two highly regarded MOEAs, the PAES of Knowles & Corne (2000) and the non-dominated sorting genetic algorithm II of Deb *et al.* (Deb *et al.* , 2000), with promising results. On the two dimensional test functions used, the MOPSO either outperforms or is not significantly different to the competing algorithms

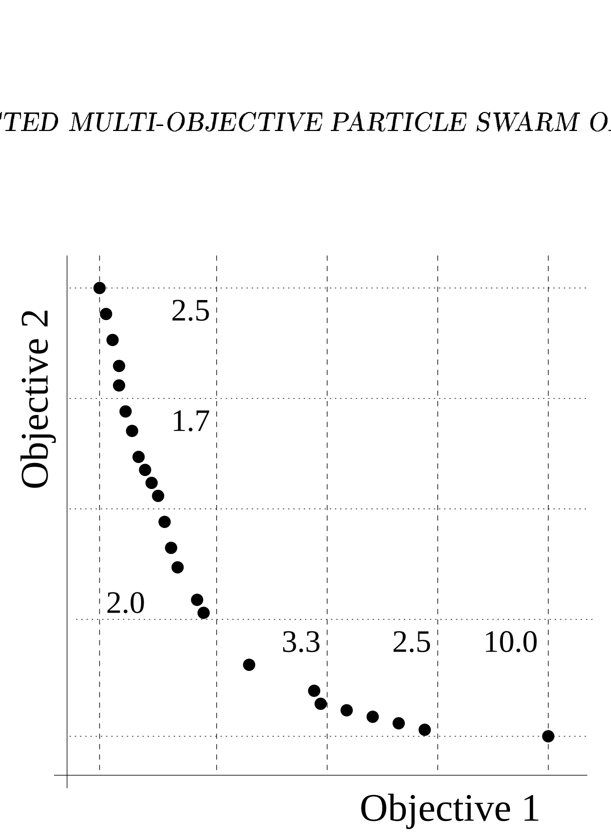


Figure 25: 2D Illustration of grid based selection scheme used in (Coello & Lechuga, 2002), with the 'fitness' of populated hypercubes highlighted.

(using the M_1^* measure (Zitzler *et al.* , 2000))[2].

7.3 The new MOPSO model: local-global optimisation using dominated trees.

The main problems with studies including Hu & Eberhart (2002) and Parsopoulos & Vrahatis (2002) are their formulation purely for 2-dimension problems, and that, by taking their inspiration from early work in the MOEA domain, they themselves are susceptible to the problems that beset these early models (which research in the 1990s highlighted, and in a large part rectified). For instance,

[2]The work also compares the empirical run time of the two approaches - showing their MOPSO to be quite significantly faster than PAES. This is actually quite a concerning result. It is relatively easy to see that Coello and Lechunga's MOPSO actually has a higher complexity than PAES - they both have identical storing methods, therefore the difference is on perturbation. PAES adds a single random value to each parameter of a perturbed solution, however Coello and Lechunga's MOPSO adds four values to each solution/particle and has the additional overhead of maintaining the local best solutions and velocity for each particle. If the PAES algorithm consistently discovered significantly more non-dominating solutions than the other method (which it would have to maintain) then it could explain the observed result. However both methods truncate this set, so it is unlikely to be the cause. The only other explanation the author can hypothesise is that the two methods had separate implementations - with the MOPSO being written in either a faster programming language - or an optimised form of code that the PAES wasn't. As such any claims of significant speed up would be disingenuous as they would caused by different modes of implementation as opposed to algorithmic complexities.

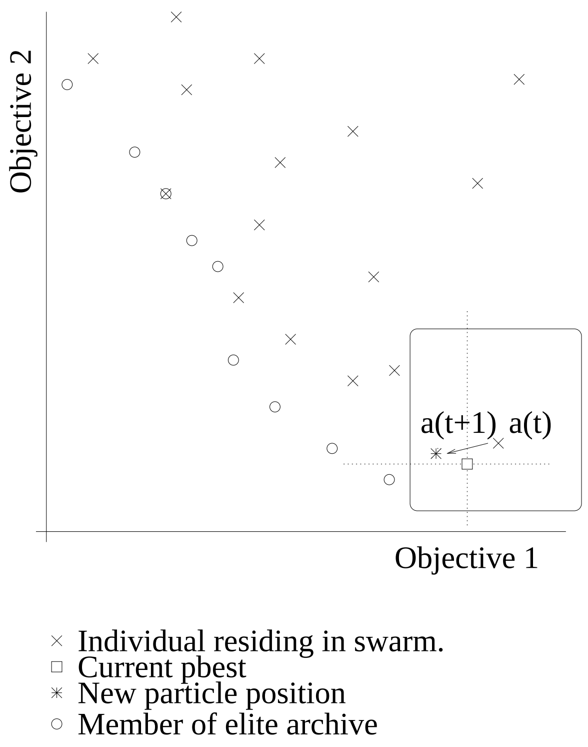


Figure 26: The multi-objective particle swarm optimisation model of (Coello & Lechuga, 2002).

by using a swarm for each objective the VEPSO model of Parsopoulos & Vrahatis (2002) will tend to suffer from the same problem of biasing its search toward the optimisation of the individual objectives as Schaffer's VEGA does. The degree of prior knowledge needed by the MOPSO of Hu & Eberhart (2002) severely restricts its application, and inspection of the plots provided in their paper show that the model experiences problems discovering solutions over the full extent of the front. The constraints of needing m swarm members to have the potential of m estimated Pareto optimal solutions at the end of the search process is also very restrictive; given the fact that typically only a small proportion of solutions in the search population at the end of the process will be estimated Pareto optimal. This in turn necessitates large swarm sizes and therefore increased number of fitness evaluations, which may be costly in most applications (let alone inefficient). In addition the oscillating phenomena described in earlier chapters is exacerbated when the Pareto front 'memory' is solely contained in X and P.

The MOPSO model of Coello & Lechunga (Coello & Lechunga, 2002) is by far the more robust, due to its use of an elite archive. Given the fact that the authors felt confident enough to compare their model against state-of-the-art MOEAs, unlike the other two studies, supports this almost as much as their positive results. However here it is argued that even this model can be improved further, as it is not a full transference of the PSO heuristic to the multi-objective domain. In uni-objective PSO the swarm is concerned with improving the fitness of each of its members with respect to a single objective. The formulation of Coello and Lechunga's model transfers this across by having each particle concerned with improving on *all* objectives. That is, the selection of the *gbest* from the archive takes no consideration of the particles position in the fitness landscape; in randomly choosing a *gbest,* a particle may be pushed towards an area in decision space whose fitness evaluation may be fitter with respect to one objective than the particle's present position, but worse on one or more of the other objectives. In contrast an approach closer to the original model would try and push a particle toward an area in decision space which was evaluated as dominating its current position, better on at least one objective and no worse on any other objective. This can be taken even further by attempting to push the particle toward the member of the archive that not only, at a minimum, weakly dominates it, but that it is closest to it in the objective space. In this interpretation of MOPSO, each swarm member is therefore concerned with improving a particular region of $\mathcal{E}$, however, only a single swarm is used. A shift in the relative position of a particle is not problematic as 'memory' of its search in a particular multi-dimension objective area is retained in the archive should any other particle become concerned with it (or indeed if it moves back in subsequent generations).

Although this focused or *directed* form of MOPSO seems an appealing transference of PSO to the multi-objective domain, the costs of implementation are prohibitive with existing methods. To find the closest archive member to a swarm individual X_i takes $\mathcal{O}(D \cdot |F|)$ objective comparisons, meaning $\mathcal{O}(m \cdot D \cdot |F|)$ objective comparisons each generation! However by using the unique ordering of individuals represented by the composite point data structure discussed in Chapter 4 this approach is now viable. For any member of the swarm, X_i, the first non-dominated composite point, $\mathbf{c}_j$, of the global non-dominated set is sought (i.e. where $\mathbf{c}_j \not\prec s \prec \mathbf{c}_{j-1}$), this takes $\mathcal{O}(\log_2(M+1))$ domination comparisons to find (where M is the number of composite points). The global best for an individual X_i is that archive member of the composite point $\mathbf{c}_j$ contributing the vertex which is less than or equal to the corresponding objective in X_i. An illustration of this is provided in Figure

27.

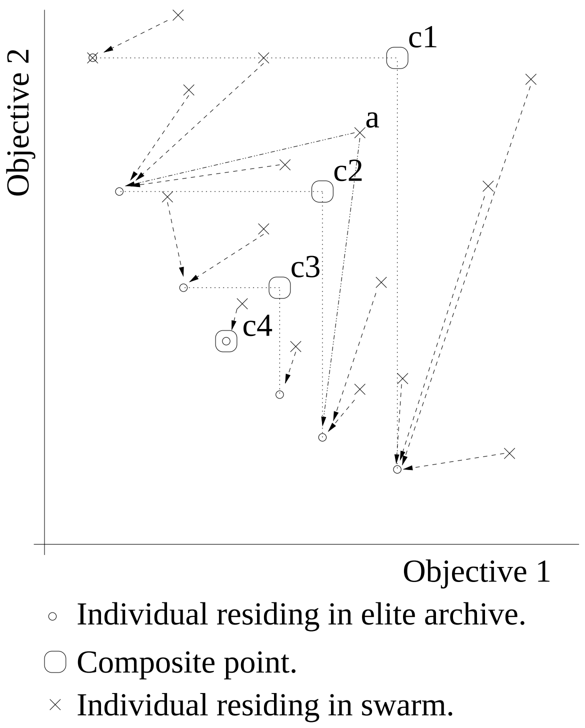

Objective 2
Objective 1
c1
a
c2
c3
c4
○ Individual residing in elite archive.
▢ Composite point.
× Individual residing in swarm.

Figure 27: Selection of local *gbest* for each swarm member.

In the case of a composite point $\mathbf{c}_j$ with more than one vertex less than or equal to the corresponding objectives of an individual X_i (as is illustrated in Figure 27 between composite point $\mathbf{c}_2$ and individual a), one of the vertices that meets the condition is selected at random to provide the global best ($F_{(i)}$) for the swarm individual X_i.

7.4 Empirical comparison of two MOPSO models

The experiments in this chapter are designed to evaluate the new algorithm developed in this chapter to existing models on a number of test functions previously described in the literature. In addition they are also designed to evaluate the utility of introducing a turbulence term to MOPSO models.

7.4.1 The comparative models

The first comparative model is based on the (1+1)-ES PAES model of Knowles & Corne (2000) with an unlimited archive. However, instead of grid based selection, the PQRS method introduced in Part I is used. Both methods attempt to provide unbiased selection from the estimated front. PQRS is simply preferred in this case as grid knowledge need not be maintained and the method is easily integrated into the dominated and non-dominated tree framework used to store the archives. At each generation in PQRS one objective dimension is selected and partitioned into $Q-1$ bins of equal width (with an extra bin containing the best individual in that dimension). To select a representative from the archive, first one bin (or the best solution) is selected uniformly to ensure that there is no bias towards the dense areas of the front, and then an individual is uniformly selected from the bin. This is easily implemented by maintaining D balanced binary trees of the archive individuals in each objective dimension. Selection then follows randomly generating a number that lies in the chosen bin's range and selecting the nearest tree member. A more detailed description can be found in Chapter 4. The second model is based on Coello & Lechunga's (2002) MOPSO with an unlimited archive and selection from PQRS. This second model uses the biased roulette wheel selection from Coello & Lechunga (2002) for bin selection.

7.4.2 The introduction of turbulence

During the early development of PSO (Kennedy & Eberhart, 1995), a stochastic variable called 'craziness' was used, such that in this early algorithm Equation 39 read;

$$V_i := V_i^{Y_i} + \mathbf{r}_3, \tag{42}$$

where $V_i^{Y_i}$ is the velocity of the nearest neighbour to X_i, Y_i, and where r_3 is a vector of random variables of length n. As this early PSO developed and changed into the more familiar algorithms described at the start of Chapter 6, this craziness parameter was dropped. In this study however the author shall empirically validate the re-introduction of an extra stochastic variable within PSO (specifically in the multi-objective domain). In keeping with the overall design of the PSO, this term is referred to as 'turbulence' (equivalent to perturbation in ES), as it reflects the change in a particle's flight which is out of its control. Where turbulence is used Equation 39 is changed to:

$$V_i := wV_i + c_1\mathbf{r}_1(P_i - X_i) + c_2\mathbf{r}_2(P_g - X_i) + \mathbf{r}_3. \tag{43}$$

7.4.3 Comparative MOAs and test functions

The comparative MOEA is based on the (1+1)-ES PAES model of Knowles & Corne (2000) with an unlimited archive and PQRS selection of parent (Fieldsend *et al.* , 2003).

Four test functions are used in this Chapter to compare the MOAs, drawn from the $D = 2$ ZDT test functions introduced in Zitzler *et al.* (2000). These functions take the form:

$$\begin{array}{lll} \text{Minimise} & T(\mathbf{x}) & = (f_1(x_1), f_2(\mathbf{x})), \\ \text{where} & f_2(\mathbf{x}) & = g(x_2, \ldots, x_N) \cdot h(f_1(x_1), g(x_2, \ldots, x_N)) \\ \text{and} & \mathbf{x} & = (x_1, \ldots, x_N). \end{array}$$

A description of all four test functions used can be found in Table 10. ZDT1 is a 2D convex test problem, ZDT2 a 2D concave test problem, ZDT3 a 2D noncontiguous convex test problem and ZDT4 a 2D multi-modal test problem.

Table 10: Test functions from (Zitzler *et al.* , 2000) used in this chapter.

#	Function	$\mathbf{x}$
ZDT1	$f_1(x_1) = x_1$, $g(x_2,\ldots,x_N) = 1 + 9\left(\sum_{n=2}^{N} x_n\right)/(n-1)$, $h(f_1,g) = 1 - \sqrt{f_1/g}$.	$N = 30$, $x_i \in [0,1]$.
ZDT2	$f_1(x_1) = x_1$, $g(x_2,\ldots,x_N) = 1 + 9\left(\sum_{n=2}^{N} x_n\right)/(n-1)$, $h(f_1,g) = 1 - (f_1/g)^2$.	$N = 30$, $x_i \in [0,1]$.
ZDT3	$f_1(x_1) = x_1$, $g(x_2,\ldots,x_N) = 1 + 9\left(\sum_{n=2}^{N} x_n\right)/(n-1)$, $h(f_1,g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$.	$N = 30$, $x_i \in [0,1]$.
ZDT4	$f_1(x_1) = x_1$, $g(x_2,\ldots,x_N) = 1 + 10(n-1) + \sum_{n=2}^{N}(x_n^2 - 10\cos(4\pi x_n))$, $h(f_1,g) = 1 - \sqrt{f_1/g}$	$N = 10$, $x_1 \in [0,1]$, $x_2, \ldots x_N \in [-5,5]$.

A visual representation of the Pareto optimal fronts for these test functions is provided in Figure 28. Also plotted for each test function are 100,000 randomly generated individuals (whose parameters are selected uniformly from the appropriate range for each test function). This gives a visual indication of the density of potential solutions for each problem.

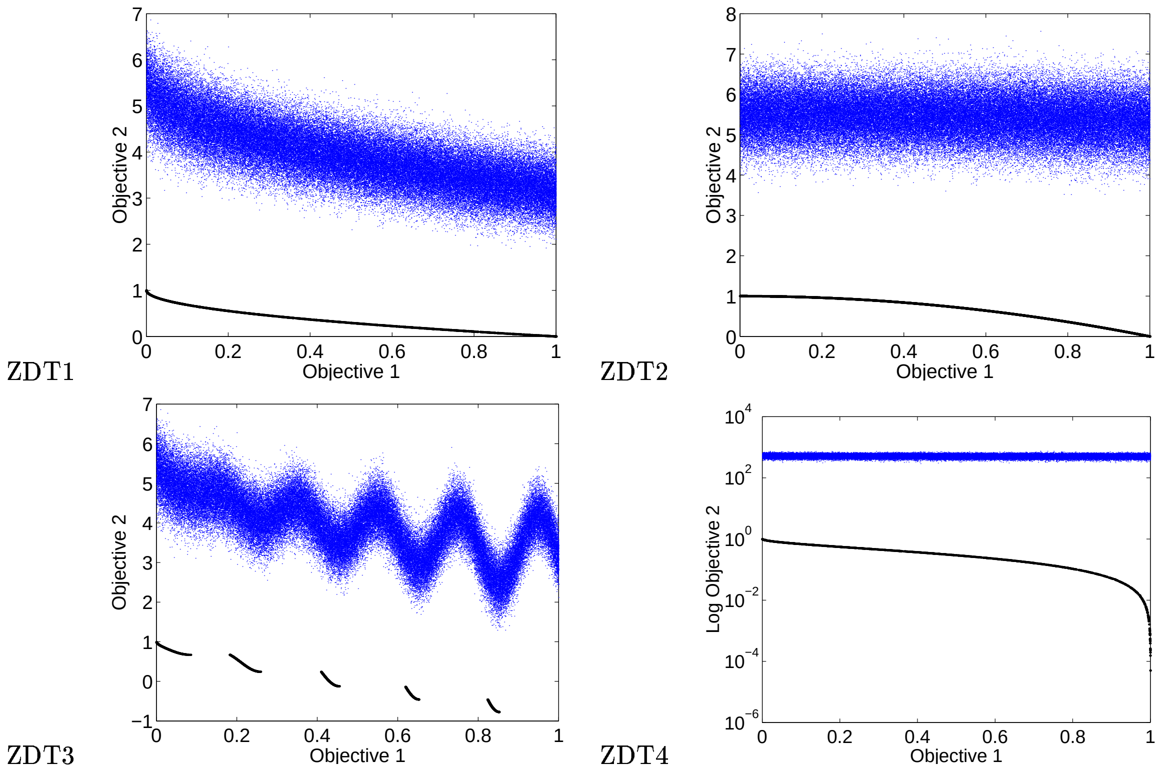


Figure 28: The Pareto optimal fronts for the ZDT functions 1-4. Optimal Pareto fronts plotted, along with 100,000 randomly generated individuals.

7.4.4 Comparative measure

Results are compared using the $\mathcal{V}^{\mathcal{P}}$ measure proposed by Fieldsend & Singh (2002a), which is similar to the $\mathcal{V}$ measure used in Part I. However, as the true Pareto front is known for all of the test functions used here, its calculation is slightly different. $\mathcal{V}^{\mathcal{P}}$ is a measure of the multi-objective error volume that is dominated by the true Pareto front but not the estimated Pareto front. Loosely $\mathcal{V}^{\mathcal{P}}$ is the fraction of the volume of a hypercuboid containing $\mathcal{P}$ $(H_{\mathcal{P}})$ that is strictly dominated by $\mathcal{P}$ but is not dominated by members of $\mathcal{E}$, the estimated Pareto set. The $\mathcal{V}^{\mathcal{P}}$ measure is easily estimated by Monte Carlo sampling of $H_{\mathcal{P}}$ and counting the fraction of samples that are dominated exclusively by $\mathcal{P}$ and dividing by the number of samples dominated by $\mathcal{P}$. Therefore the lower the $\mathcal{V}^{\mathcal{P}}$, the lower the hypercuboid volume exclusively dominated by the true Pareto front, and the nearer the estimated front is to the true front. This measure therefore alleviates the problem of pairwise comparisons needed by $\mathcal{V}$, as $\mathcal{P}$ acts as the baseline to which all competing models are compared.

The hypercuboid bounds are determined by the $\mathcal{P}$ range of f_1 and the $\mathcal{P}$ range of $f_2 + 3.0$ (the cuboid used in Zitzler *et al.* (2000) when visually comparing their MOEA results). This allows direct comparison of the $\mathcal{V}^{\mathcal{P}}$ measure across all models. A total of 250000 samples were taken for Monte Carlo estimates, and $\mathcal{P}$ was represented by 250 randomly drawn members of $\mathcal{P}$.

7.4.5 Algorithm implementation

The implementation of all the models use floating point representation of parameters in the individual chromosomes. In order to compare the new MOPSO technique, each MOA was executed 25 times on each test problem, and the resultant non-dominated solutions saved at the end of each run. Each simulation was performed using the parameters shown below:

- ES. Number of generations = 4000. Mutation rate =0.2.
- PSO (all variants). Number of generations 200. Search population size = 20.

The turbulence (perturbation) variable for all models was $\sim N(0, 0.1R)$, where R is the absolute range of the model parameter[3]. In each of the 25 different runs, the MOAs were initialised from identical decision vector populations of size 20, with the non-dominated individuals residing in these populations forming the initial elite archives. Initialisation of decision vectors was from Uniform distributions, over the range of the chromosome parameters for the particular test function. The experiments were repeated with and without turbulence and w was set at 0.4 (as used in Coello & Lechunga (2002)) as well as 0.8. Turbulence probability was fixed at 0.2. In PQRS, for all algorithms, $Q = 20$. As in (Coello & Lechunga, 2002), $c_1, c_2, \chi = 1$.

[3]The choice of a Gaussian distribution is just one of many possible distributions. Other equally as valid choices may be those with thicker tails like the Laplacian or the Chaucy distribution, which has been demonstrated to significantly improve performance of single objective evolutionary algorithms (Yao *et al.* (1999))

7.4.6 Results

Table 11 shows the results of these experiments. The use of turbulence is seen to significantly increase the performance of both the MOPSO algorithms across the test functions, as does the use of a higher w value. In addition, the new dominated tree MOPSO algorithm can be seen to be significantly better than the ES method and the competing MOPSO method when using turbulence and a high w on the first three test functions. The new MOPSO also tends to be better than the competing MOPSO even when no turbulence is used and with lower w values (sub-optimal parameter settings). When considering why this may be the case one must view the effect of turbulence in the context of the general PSO arithmetic form. In the general case, a particle X_i is pushed/pulled towards its *pbest* and *gbest* operating points, as well as along its current velocity. This means that a hypercuboid is generated in decision/particle space containing these four points, the bounds of which are defined by the sum of the absolute distances from X_i to the three other points, each distance multiplied by its relevant constraint from Equation 39. X_i can therefore effectively move to any point within this hypercuboid, but not *outside* it. An illustration of this is shown in Figure 29 where the decision space is comprised of two variables. The highlighted area in Figure 29 shows the bounding hypercuboid where particle X_i can move to given its previous best P_i, global best P_g and velocity V_i. Therefore, as illustrated, it is feasible for X_i to be moved to 'a', but impossible for it to shift to 'b'.

This restriction on a particles movement means that local optima within this bound may be found, but any global optima outside will not be found at that generation, and search may easily be stuck at a local minima. Turbulence has the effect of increasing the volume of this bounded region, indeed, if the turbulence is drawn from distributions that extend beyond the range of the variables then in effect there is no bound on the search process at all. The turbulence term can therefore be seen to operate as a stochastic process shaking the particles out of local optima they may become stuck in.

The fourth test function however, with multi-modality (21^9 local Pareto-optimal fronts, Zitzler *et al.* (2000)), causes great problems for both MOPSO algorithms, with none of the estimated fronts from these models anywhere near the true Pareto front.[4] This is due to the function design, where to pass through local Pareto fronts, a swarm member may have to fly in a direction opposite to its

[4]It is interesting to note that, although not discussed in the original study, during its conference presentation this problem was also noted by Coello on results not reported in Coello & Lechunga (2002).

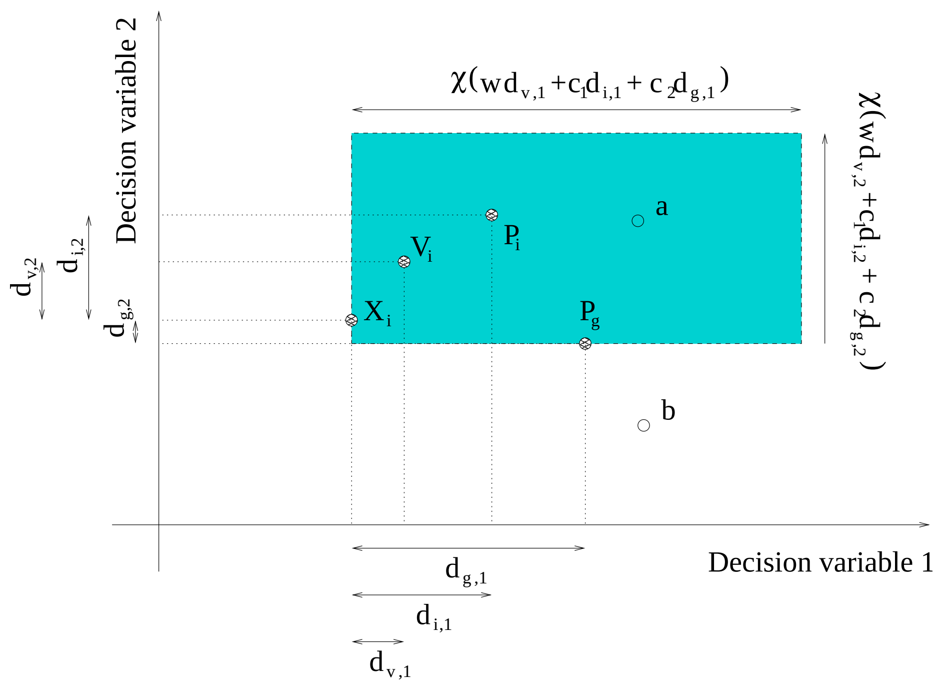


Figure 29: Illustration of the PSO search process, and the volume in which a particle X_i can move at each iteration. X_i can feasibly be moved to point 'a', but not to point 'b'.

local and global best (in parameter space).

To put these results in context, Zitzler *et al.* (2000) compares six MOEAs on these test functions: FFGA, the NPGA, HLGA, the VEGA, the NSGA and their own elitist SPEA. This comparison was performed with search populations four two five times larger than here and for 250 generations. Even after 16,000-20,000 evaluation functions (as opposed to the 4,000 used in this chapter) many algorithms still did not converge to the true Pareto fronts as well as the MOPSO methods described here.

Table 11: MOA comparison results with various parameter settings. ES refers to the (1+1)-ES MOEA, P′ refers to the MOPSO model based on (Coello & Lechunga, 2002) and P″ refers to the MOPSO method developed in this Chapter. Means highlighted in bold are significantly better than both competing models (using the Wilcoxon Signed Ranks Test at the 0.05 level, 0.025 in each tail). Means in italics are significantly better than one other competing model. # is the test function number, w the inertia and T refers to whether turbulence is present.

		$\mathcal{V}^{\mathcal{P}}$ (%)		
#	w , T	ES	P′	P″
	0.4	**15.0**	55.0	*32.3*
	No	(1.3)	(6.7)	(8.0)
	0.8	*15.0*	20.1	**3.7**
1	No	(1.3)	(7.6)	(3.8)
	0.4	15.0	**3.0**	*3.6*
	Yes	(1.3)	(0.5)	(0.7)
	0.8	15.0	*1.2*	**0.7**
	Yes	(1.3)	(0.3)	(0.1)
	0.4	**12.0**	65.6	*60.9*
	No	(1.2)	(7.8)	(8.5)
	0.8	**12.0**	27.8	31.2
2	No	(1.2)	(5.1)	(10.2)
	0.4	12.0	*6.7*	**5.8**
	Yes	(1.2)	(0.8)	(0.9)
	0.8	12.0	*4.4*	**1.6**
	Yes	(1.2)	(1.1)	(0.5)
	0.4	**8.2**	32.2	*12.1*
	No	(1.0)	(3.3)	(2.2)
	0.8	*8.2*	17.3	**3.1**
3	No	(1.0)	(4.9)	(2.6)
	0.4	8.2	*2.8*	*3.1*
	Yes	(1.0)	(0.9)	(0.6)
	0.8	8.2	*1.3*	**0.7**
	Yes	(1.0)	(0.5)	(0.2)
	0.4	**68.1**	100	100
	No	(16.5)	(0.0)	(0.0)
	0.8	**68.1**	100	100
4	No	(16.5)	(0.0)	(0.0)
	0.4	**68.1**	100	100
	Yes	(16.5)	(0.0)	(0.0)
	0.8	**68.1**	100	100
	Yes	(16.5)	(0.0)	(0.0)

Chapter 8

Empirical Comparison of MOPSO Models

Following the success in the previous chapter of the new dominated tree MOPSO model, in this chapter sixteen MOSPSO models are derived based on different *pbest, gbest* and *lbest* selection methods (of which the models of Coello & Lechunga (2002) and Fieldsend & Singh (2002a) are variants), and compared. A hierarchy of models is shown to exist, with the new method introduced in chapter 7 vying for top position with another novel MOPSO method introduced in this chapter.

8.1 The derived models

Following the significant improvement of the new MOSPSO introduced in the previous section with regards to the MOPSO of Coello & Lechunga (2002), sixteen different MOPSO models are now compared that all maintain an unconstrained elite archive of *gbest* solutions and a fixed population of search particles. Four different methods of selection of the *gbest*/*lbest* individual for a particle are compared and four different methods of *pbest* selection are compared (as described below). A general algorithm for the implementation of these MOPSOs is described in Algorithm 6.

In this set of experiments, as in those of the previous chapter and the work by Coello & Lechunga (2002), $c_1 = c_2 = \chi = 1$. The different methods of selecting from the *gbest* archive F and the set/hyperset of *pbest* individuals $\mathbf{L}$ in Algorithm 6, as used in this chapter, will now be described.

Algorithm 6 General MOPSO algorithm.

```
1: Generation counter $t := 0$. Initialise the swarm population $X(t)$, and update the
   non-dominated population $F(t)$ with non-dominated members of $X(t)$.
2: Initialise the local non-dominated set/hyperset $\mathbf{L}(t)$ with members of $P(t)$, $\mathbf{L}_1(t) := P(t)$.
3: Initialise the velocity set $V(t)$, $V_i(t) := 0 \ \forall i = 1, \dots, |X|$.
4: $t := t + 1$.
5: Calculate new velocity of each particle. $V_i(t) := wV_i(t-1) + c_1\mathbf{r}_1\left(\mathbf{L}_{(i)}(t-1) - P_i(t-1)\right)$
   $+c_2\mathbf{r}_2\left(F_{(i)}(t-1) - P_i(t-1)\right) \ \forall i = 1, \dots, |X|$, where $\mathbf{r}_{1,j}, \mathbf{r}_{2,j} \sim \mathcal{U}(0,1) \ \forall j = 1, \dots, n$.
6: Accelerate the swarm members along their new trajectories, $P(t) := P(t-1) + \chi V(t)$.
7: Update non-dominated global store $F(t)$, and local set/hyperset $\mathbf{L}(t)$.
8: If user defined termination conditions are not met, go to 4.
9: end
```

8.1.1 Selection of *gbest* or *lbest*

Four different types of *gbest*/*lbest* selection are compared in this chapter. The first three are based on the random selection of an individual from F to act as a *gbest* for a swarm member. The first method, $\mathcal{M}_{1,i}$,[1] is simple uniform selection of an instance from F. The primary benefit of this approach is that this selection is rapid, $\mathcal{O}(1)$, however it biases selection to already densely represented areas of the estimated Pareto front.

The second method, $\mathcal{M}_{2,i}$, is based on unbiased selection of the front. Here the front is partitioned, with selection first uniformly of a partition, and then uniformly from that partition. The method used here is PQRS (as introduced in Part I and used in the previous chapter.) This selection makes $\mathcal{O}(\lg(|F|))$ objective comparisons. As before, PQRS is simply preferred in this case to the grid schemes used in Coello & Lechunga (2002) and Knowles & Corne (2000) as grid knowledge need not be maintained and the method is easily integrated into the dominated and non-dominated tree framework. At each generation in PQRS one objective dimension is selected and partitioned into $Q - 1$ bins of equal width (with an extra bin containing the best individual in that dimension). To select a representative from the archive, first one bin (or the best solution) is selected uniformly to ensure that there is no bias toward dense areas of the front, and then an individual is uniformly selected from the bin. This is easily implemented by maintaining D balanced binary trees of the archive individuals in each objective dimension. Selection then follows randomly generating a number that lies in a chosen bin's range and selecting the nearest tree member.

The third, $\mathcal{M}_{3,i}$, method also uses PQRS partitioning, but biases selection toward those partitions

[1] i indicates the various *pbest* selection methods used, which are discussed in the next section

which have fewer members by using roulette wheel selection of partitions (as used by the MOPSO in Coello & Lechunga (2002) and in the previous Chapter).

The final, $\mathcal{M}_{4,i}$, method is the local-global method previously introduced, where a *gbest* individual from an elite archive is selected locally to each swarm member. As stated previously, this makes $\mathcal{O}\left(\log_2\left(M+1\right)\right)$ domination comparisons, as opposed to $\mathcal{O}\left(D \cdot |F|\right)$ objective comparisons if a linear comparison between particle and archive was used to find the closest.

8.1.2 Selection and maintenance of *pbest*

Four different *pbest* maintenance and selection strategies are compared. The first two methods only maintain a single *pbest* for each individual. The first method, $\mathcal{M}_{i,1}$,[2] comes from the MOPSO in Coello & Lechunga (2002) and the second, $\mathcal{M}_{i,2}$, from the MOPSO in Hu & Eberhart (2002) (as described previously).

In the third and fourth methods, a set of local best individuals found is maintained for each swarm member. In the third method, $\mathcal{M}_{i,3}$, the selection of a local best for an individual from the hyperset $\mathbf{L}$ (the set all of the local best sets) is uniform (as used in the previous chapter). In the fourth method, $\mathcal{M}_{i,4}$, however the *pbest* selection is based on local closeness. As the size of the particle estimated Pareto fronts are typically small these are maintained in linear lists and closeness is determined by Euclidean distance. This distance is normalised by mapping the minimum axis-parallel hypercube that contains the particle's $\mathcal{E}$ (local estimated Pareto front) to unit range. The Euclidean distance is then calculated on this transformed representation (this mapping is of course not permanent - only a temporary measure when calculating local distances).

8.2 Comparative measure

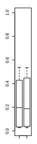
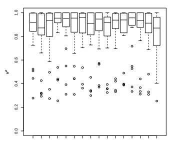
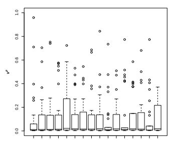
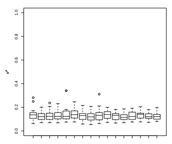
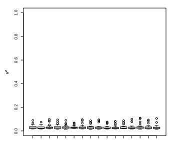
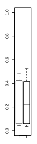
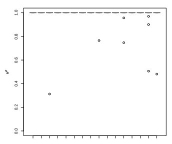
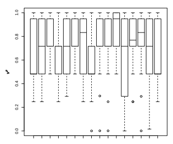
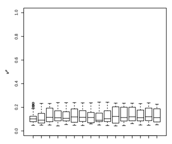
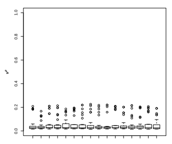
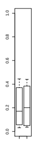
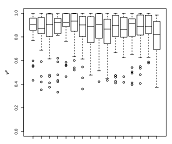
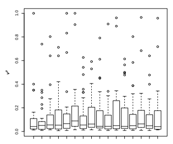
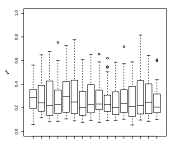
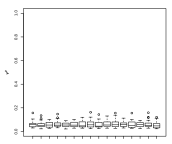
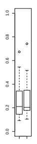
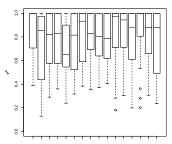
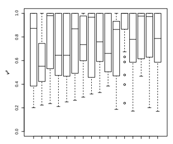
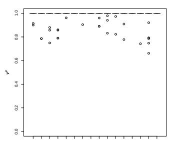
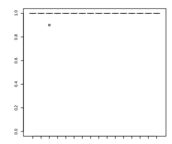
The test functions used here and other algorithm parameters are identical to the previous MOPSO experiments, however each algorithm was run for 30 times in these experiments. Again the $\mathcal{V}^{\mathcal{P}}$ measure is used, however the hypercube bounds for the ZDT test functions are defined by the minimum axis parallel hypercube that contains $\mathcal{P}$ and the reference front created by adding $0.1 \times [\max\{(f_2)\} - \min\{(f_2)\}]$ to the f_2 values of $\mathcal{P}$. A total of 250000 samples were taken for Monte Carlo estimates, and $\mathcal{P}$ was represented by 250 randomly selected members of $\mathcal{P}$.

[2] i indicates the various *gbest* selection methods used, which were discussed in the previous section

8.3 Results

Boxplots of the empirical results can be found in Table 12. The influence of the various parameters of the models are now discussed.

Table 12: Boxplots of results using $\mathcal{V}^{\mathcal{P}}$ measure. The 16 MOPSO models are ordered in four groups of four, the groups ordered by the *gbest* selection type and the order within a group determined the *pbest* selection type. The first box therefore denotes the $\mathcal{M}_{1,1}$ MOPSO, the fourth box denotes the $\mathcal{M}_{1,4}$ MOPSO and the sixteenth box denotes the $\mathcal{M}_{4,4}$ MOPSO. The column titled ES contains the boxplots of the (1+1)-ES MOEA (first) and the hybrid MOESPSO (second).

		Inertia, Turbulence			
#	ES	0.4, No	0.8, No	0.4,Yes	0.8,Yes
ZDT1					
ZDT2					
ZDT3					
ZDT4					

8.3.1 Inertia

Apart from test function ZDT4, the higher inertia weight clearly increases convergence to the true Pareto front (and indeed it is consistently statistically significant using the Wilcoxon signed ranks test at the 0.05 significance level with 0.025 in each tail). This is shown in Table 12 with the boxplots across the models using an inertia $w = 0.8$, both with and without turbulence, closer to the x-axis than the identical models with $w = 0.4$. This is interesting, as too high a level of inertia in the uni-objective application of PSO is thought to lead to premature convergence, however possibly due to the maintenance of a population of solutions in MOPSO, this problem is not so pronounced in the multi-objective domain.

8.3.2 Turbulence

Again, apart from ZDT4, all the MOPSO models tend to perform significantly better (at both inertia levels) when the turbulence term is introduced. With the high inertia, high turbulence model variants performing best of all, significantly outperforming the competing MOEA, and hybrid MOESPSO model on the first three test functions. However the obvious reversal of this situation occurs in ZDT4, where the use of turbulence has a negative effect. However even when turbulence is not used PSO still relatively under-performs on this test function, leading to the assumption that the properties of ZDT4 itself may be having an effect, which will be discussed more fully in Section 8.4.

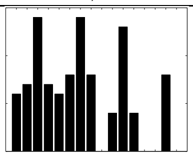
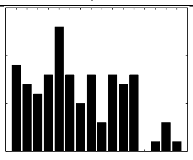
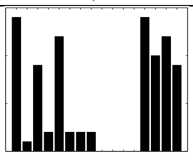
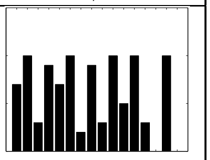
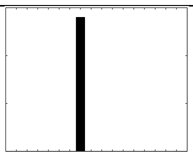
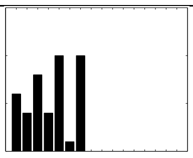
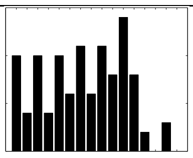
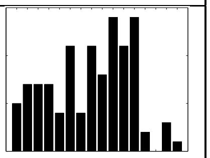
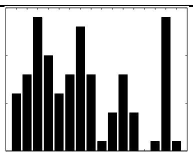
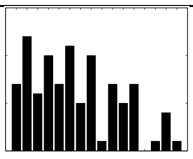
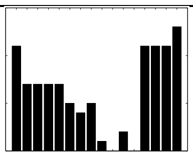
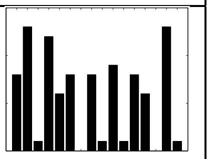
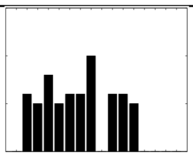
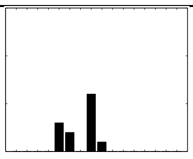
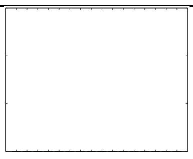
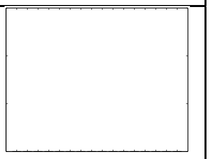
8.3.3 *pbest* and *gbest* selection

As highlighted in Table 12, the impact of the *pbest* and *gbest* selection methods is less important to overall MOPSO performance than the use of turbulence or the inertia weight. In addition, their effect does not seem to follow any discernible pattern, i.e. no one model clearly stands out as significantly better than the others. Table 13 shows a number of competing MOPSO models that a given model significantly outperforms with a bar chart.

The average performance of the MOPSOs across test functions without turbulence and with turbulence are shown in Figures 30 and 31.

A general hierarchy of models, and the interactive affect of turbulence is easier to see in these figures. Models using the composite tree based *gbest* selection method, $\mathcal{M}_{4,i}$s, are seen to perform relatively poorly when turbulence is not included. However with high turbulence and high inertia this *gbest* selection method, in combination with the third *pbest* method (which maintains a population of *pbest* solutions for each particle, the new model introduced in the previous chapter, $\mathcal{M}_{4,3}$), is seen to perform on average joint best. The other best performing model when turbulence is used is the simplest model, $\mathcal{M}_{1,1}$, that uses random selection and a single *pbest* solution randomly is

Table 13: Bar charts showing significant results using $\mathcal{V}^{\mathcal{P}}$ measure. The 16 MOPSO models are ordered in the same fashion as in Table 12. The bar chart plots range between 0 and 15, and shows the number of competing MOPSO models that a particular MOPSO is significantly better than (using the Wilcoxon Signed Ranks Test, at the 0.05 significance level.)

	Inertia, Turbulence			
#	0.4, No	0.8, No	0.4,Yes	0.8,Yes
ZDT1				
ZDT2				
ZDT3				
ZDT4				

replaced when pairwise non-dominated. Although as stated, $\mathcal{M}_{1,1}$ is the easiest form of MOPSO to implement - none of the previous studies reviewed had actually done so. The best performing MOPSO model when turbulence is not used, unbiased *gbest* selection and uniform *pbest* selection from a hyperset, $\mathcal{M}_{2,3}$, is also the second worst performing model with turbulence.

8.4 Key results

In the first set of experiments in Chapter 6 a new method for selecting the best global and local individuals for MOPSO swarm members, facilitated by the data structures introduced in Part I was compared to an existing well validated MOPSO and an MOEA. The new dominated tree MOPSO is based on a concept of closeness to members of the global set, and maintains a set of local best solutions for each swarm member. It has been shown to be significantly better than both comparative

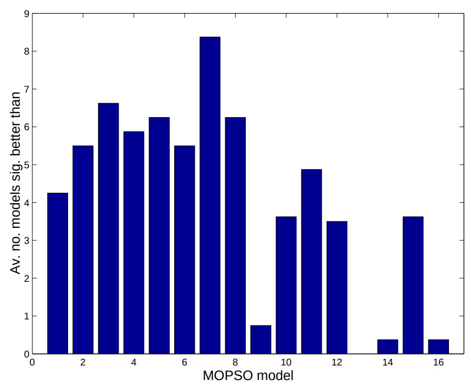


Figure 30: Average number of competing models that the MOPSOs are significantly better than, when no turbulence is used. Model ordering as Table 12.

MOAs. It has also been demonstrated that the use of a stochastic turbulence variable can be a significant aid to general MOPSO. However this approach does has some deficiencies. Clearly if there is little or no relationship between 'closeness' in objective space and 'closeness' in parameter space, MOPSO methods (and PSO methods in general) may experience problems (for instance in ZDT4).

In the set of experiments in this chapter a large number of MOPSO models have been compared, all derived from the general PSO framework, the majority of which have not been previously evaluated. Two of these models, the random *gbest* selection and a single *pbest* solution model, and the more advanced global *lbest* selection model with uniform *pbest* solution selection from a set (the dominated tree MOPSO), have been shown, on average, to produce the best results. However it must be noted that the test functions used have evenly distributed solutions. The detrimental bias of random selection in the simpler model may be more problematic on less well distributed solutions. This should merit further investigation when test functions exhibiting these properties are fully developed. Nevertheless it is recommended to implement one or both of these methods as comparative MOPSOs in future studies of MOPSO and MOEA.

In addition the use of a stochastic turbulence variable within MOPSO has been shown to have significant impact. Its implementation consistently increases the performance of all MOPSO models on three of the four test functions. High inertia weights have also been shown to be significantly beneficial to all MOPSOs on the first three test functions.

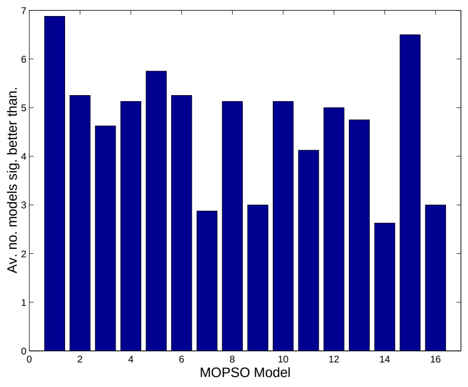


Figure 31: Average number of competing models that the MOPSOs are significantly better than, when turbulence is used. Model ordering as Table 12.

The author also notes that MOPSOs compared here showed some deficiencies. The multi-modal test problem ZDT4 reversed all the trends of the earlier results. Turbulence and higher inertia actually decreased performance, with the MOEA significantly outperforming the MOPSOs on all parameter settings. Although the hybrid MOESPSO did not succumb to the problems on ZDT4 that the MOPSOs did, it performed no better than the ES MOEA on the other test functions.

It may be argued that due to its focus on *gbest* and *pbest* to drive its search process, PSO will be more susceptible to getting trapped in the local fitness peaks that multi-modal test functions encapsulate. If a particle's global best and previous best are both on one of these peaks, then there is nothing to direct it away. In theory, the stochastic turbulence term should then help this process by pushing an individual out of this local optimum - however as the results show, this is not the case (at least - not at the generation length compared).

Until a suitable approach for minimising the detrimental impact of multi-modality on MOPSO is developed, it is recommended that, if the problem is known *a priori* to be multi-modal, an MOEA and not an MOPSO should be the optimiser of choice.

Parts I and II of this thesis have dealt with the general improvement of the multi-objective search process through the development of efficient data structures to enable faster optimisation and the generation of new MOEA methods (in the domain of MOPSO). In the following final part of the thesis these advances shall be applied to the domain of multi-objective neural network training, where until this point the vast majority of approaches have been founded on the suboptimal technique of propagating a linear sum of errors through gradient descent methods. A general framework for multi-objective evolutionary neural networks is developed and validated on a large number of real world and artificial data sets.

Part III

Multi-Objective Evolutionary Neural Network Framework

In Part III the advances made in the previous two Parts with regards to general multi-objective search, are applied to the training of NNs. A brief review of NN time-series forecasting is provided, and a general framework for the use of MOEAs to train NN is formulated. Empirical research is undertaken which demonstrates the validity of this approach even in noisy function approximation, using test data from recent approaches in the literature to gradient descent multi-objective optimisation (Lo & Bassu, 2002a; Lo & Bassu, 2002b). After the application to function approximation with systematic noise, and the development of novel generalisation techniques, the new approach is applied to noisy real-world forecasting problems from the finance domain, underlining the significant trade-off between fixing a Euclidean minimisation model and one that maximises potential return. This approach is then refined and applied to the prediction of 37 different international stock indices, producing estimated Pareto optimal models describing the trade-off between expected return and prediction volatility through trading on neural network forecasts.

Chapter 9

Neural Networks for Time Series Forecasting

In this chapter a review of NN time series forecasting is provided, and salient areas of data selection/processing are discussed.

9.1 Neural network time series forecasting & function approximation

The use of NNs in the time series forecasting domain is now well-established. There are review papers on this matter (for example, that by Adya and Collopy (1998)), as well as methodology studies (Moody, 1998; Refenes *et al.* , 1997). The main attribute which separates NN time series modelling from traditional econometric methods, and the reason practitioners most often cite their use, is their ability to generate non-linear relationships between a vector of time series input variables and a dependent series, with little or no *a priori* knowledge of the form that this non-linearity should take. This is opposed to the rigid structural form of most econometric time series forecasting methods (e.g. linear Auto-Regression (AR) models, Exponential Smoothing models, (Generalised) Auto-Regressive Conditional Heteroskedasticity models (G)ARCH, and Auto-Regressive (Integrated) Moving Average (AR(I)MA) models) (Bera & Higgins, 1993; Fieldsend, 1999; Gujarati, 1992). Apart from this important difference, the underlying approach to time series forecasting itself has remained relatively unchanged during its progression from explicit regression modelling to the non-linear generalisation approach of NNs. Both of these approaches are typically based on the concept that the most accurate forecast, if not the actual realised (target) value, is the one with the smallest Euclidean distance from the actual. An assumption of all forecasting models is that there is an underlying functional (causal) relationship between the inputs to a model and the output to be forecast. In the following section, the methods that aid models detect these relationships are described.

9.2 Data processing and effective experiment design in the Neural Network literature

There have been hundreds if not thousands of studies using NNs for modelling of various systems, with a number of journals and conferences that focus specifically on NN technique and advances. Although a number of researchers have differing views on exact implementational techniques, one consensus at least has been reached: "the quality of the model is greatly dependent on the quality of the data given to it." As such the first part of this chapter introduces various techniques in the literature for data pre-processing.

Data is typically sampled in NN training, with a base pool of explanatory and dependent variable vectors separated into a *training* set, a *validation* set, and a *test* set, all of which do not *overlap.* The NN model weights are adjusted such that the model exhibits minimum error on the training set. This error measure is pre-defined by the user, and the minimisation property itself is local if a gradient descent based optimiser is used, or asymptotic to the global if an EA approach is used. If a validation set is used, then this is usually conditional on some error property with relation to this set, so that weight adjustment may terminate before the minimum is reached on the training set (in this case, typically NN weights are adjusted with respect to the training set and adjustment is terminated when the model error with respect to the validation set starts to rise). The generalisation error is typically calculated with respect to the test set.

There are however a number of different approaches in the time series forecasting literature on how to construct these sets.A common approach to data partitioning for neural network training is that of consecutive partitioning. This involves selecting the first temporal portion of the available data to train/fit the model, and the remainder to test the generalisation ability of the model (Refenes *et al.* , 1997). Data randomisation is also widely used, here data vectors are randomised before the training, validation, and test sets are constructed, e.g. Lajbcygier *et al.* (1995). In its simplest form a single column array of random values can be associated with the matrix of input/output vectors, and by sorting the random data array in ascending (or descending) order the data matrix can be effectively shuffled. In this approach the input pattern of the system (which explanatory variables to use, and which specific lags) must be decided before the randomisation takes place. This means that a certain degree of data tracking must take place for such regimes as sensitivity analysis to be implemented (Moody, 1998). LeBaron & Weigend (1998) demonstrate that variation in network performance is affected by the data sample used in training to a greater extent than the random initialisation condition of the NN. They use what is known as *bootstrapping,* where input-output vector pairs are randomly extracted from the data (with or without replacement) for the generation of the training, validation and test sets. This differs from randomisation due to the 'replacement' factor, meaning that the union of the training, validation and test sets may be in fact larger than the original data set, as there may be duplication of pairs.

Many different forms of time series pre-processing are also commonly used in the time series forecasting domain including standard transformations such as principal components analysis (PCA) and Fourier analysis, and other application dependent approaches. Data pre-processing is typically used to alter the statistical properties of the data, reduce noise and detect trends, in order to make the forecasting task itself simpler.

Often in time series modelling there can be a large number of explanatory time series available, some of which may be largely redundant due to them being different measures of the same process. For example in economics there are three different measures of money supply, m1, m2 and m3. The first includes cash and bank deposits, the second includes all those in the first measure plus short term investments (funds etc) and the third includes the previous measures and institutional money market funds and inter-bank agreements. Other time series available may actually be spurious (by being unrelated to the dependent series). In this section the common 'data driven' methods for data reduction in NN time series forecasting are introduced.

At a bare minimum data is usually scaled before being presented to a NN (for example Kaastra & Boyd (1996)), with respect to the upper and lower bounds of the transformation functions. This is typically between 0 and 1, although also on the ranges [0.05,0.95] and [0.1,0.9] (to enable NN functionality slightly beyond the range of the training data.) A number of data transforms are performed in the literature that are application specific. That is, prior knowledge is used in order to transform the data into new series that are thought to express a relationship with the dependent series which is easier to model (e.g. Saad *et al.* (1998) and LeBaron & Weigend (1998)). Adjusting the data with respect to periodic seasonal variation (seasonality) is a technique regularly used in the econometric literature, which has gained use in NN time series forecasting recently. Nelson *et al.* (1999) for instance report that by deseasonalising their data prior to NN modelling, the forecast accuracy of the model increases. Atiya *et al.* (1999) also seasonally adjust their time series data. Typically, deseasonalisation is accomplished by subtracting the seasonal averages from the time series at the relevant points. The theoretical justification itself is relatively straightforward, as is the case with (hopefully) all data transforms. The problem itself is made simpler as the NN model no longer has to represent the seasonal changes.

A technique known as *differencing* is used extensively in NN time series modelling for ensuring data stationarity. For example in Hann & Steurer (1996) 13 inputs are used in a multivariate time series model, 10 are differences and one is a double difference. The aim of differencing is to remove linear trends from the input data (known as non-stationarity) where the series mean is time dependent. Without removal it is apparent that out-of-sample data will present the network with inputs far in excess of the range of data used in its training.

The level of differencing, I, is denoted as

$$\zeta_t \sim I\left(d\right) \tag{44}$$

where d is the order of difference (Virili & Freisleben, 2000), and ζ_t is the transformed series. The calculation of $\zeta_t \sim I\left(1\right)$ is

$$\zeta_t = \frac{b_t}{b_{t-1}} \tag{45}$$

where b is the original non-stationary time series. The calculation of $\zeta_t \sim I\left(d\right)$ is

$$\zeta_t := \frac{b_t}{\zeta_t \sim I\left(d-1\right)} \tag{46}$$

Simple moving averages for converting non-stationary data into a stationary equivalent for NNs are discussed in Kaastra & Boyd (1996). The moving average transform ξ_t of non-stationary series b_t is calculated as

$$\xi_t = \frac{n \cdot b_t}{(\sum_{i=1}^{n} b_{t-i})} \tag{47}$$

where the moving average window length n is user defined.

Several of the studies discussed in this chapter have had their performance evaluated with respect to a number of different error measures (objectives), but have been fitted with respect to only single error term (or a weighted linear sum of more than one). In the next section a discussion is provided on why the linear weighting approach to multi-error neural networks is inadequate, and examples are given where multi-error training is needed.

9.3 Traditional multi-error training

When committed to forecasting (or classification) tasks, NNs are typically trained with respect to Euclidean distance minimisation. This is commonly irrespective of any other end user preferences. In a number of situations, for instance time series forecasting, users may have other objectives in addition to Euclidean distance minimisation. Users may, for example, desire model predictions to be consistent in a relative error measure, to be accurate in their directional change predictions, or may prefer a number of application dependent error measures. Recent studies have confronted the problem of multi-objective training of NNs by back-propagating a linear sum of errors (Moya & Hush, 1996; Wang & Wahl, 1997; Wen & Lee, 1998; Yao & Tan, 2000). However this approach implicitly assumes *a priori* knowledge of the error surface defined by the problem, which typically is not the case.

9.3.1 Single model example

In forecasting financial, economic and several other time series, it is often important to correctly predict the directional change of the series (Armstrong & Collopy, 1992). Consider predicting the consumer demand for a particular good. Just as important as the actual level of demand predicted in the next time step, is whether the demand at the next time step is higher or lower than the present level. The same can also be true of many physical and physiological systems. For example, in the medical field changes in direction of time series (heart rate, blood pressure) can be as important as the actual level (see Figure 32). If it is accepted that any model forecast will contain some degree of error, the practitioner may be willing to sacrifice some of the Euclidean accuracy for a greater degree of confidence associated with the predicted directional change of the model forecast.

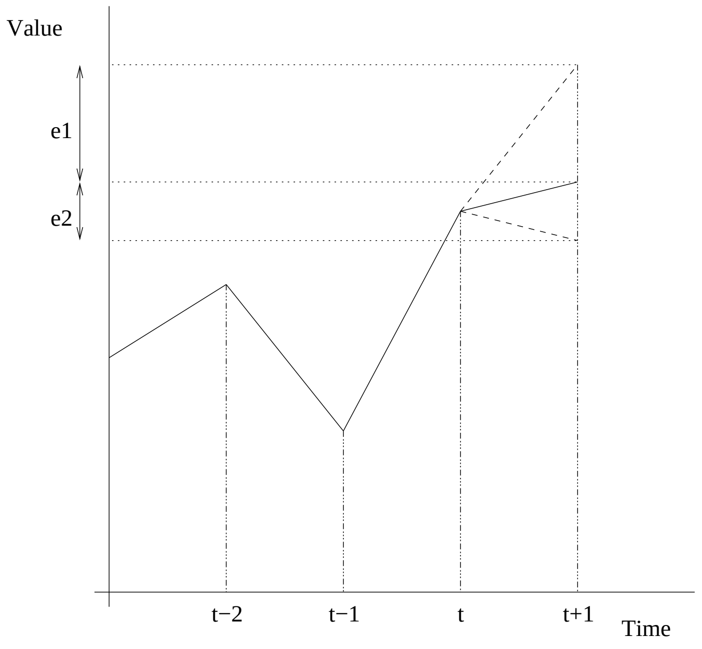


Figure 32: Correct direction change prediction versus Euclidean minimised model.

In Figure 32, even though the Euclidean error of model B ($e2$) is smaller than that of model A ($e1$), model A correctly predicts the directional movement of the series (with respect to the realised previous time step), and as such may be the user-preferred model.

In addition, the end user might not be solely concerned with the accuracy of the forecast, but also with the properties of the residual error. They may desire the model's error to be consistent (being drawn from a Platykurtic distribution), as opposed to one that has both periods of very low error and also occasional periods of very high forecast error (drawn from a Leptokurtic distribution). End users may often prefer a model that has an overall higher average Euclidean error than another, provided it is less likely to produce instances of very large prediction error. This preference may be derived from the costs associated with large forecast errors. Take for example the situation of forecasting a manufacturing firm's inventory level. A certain degree of forecast error may be absorbed by the company, but sporadic large differences between the predicted and actual could leave the firm with a large surplus of material (and associated storage costs) - or worse, lacking the materials it needs to satisfy demand.

9.3.2 Set of models example

An illustration of the interaction between multiple objectives in a problem, where a set of models is desired for collective use (as opposed to comparison), can be shown by analogy with the capital asset pricing model (CAPM) from finance (Brealey & Myers, 1996). The CAPM describes the relationship between risk and return in an optimum portfolio of stocks, where risk is to be minimised and return maximised.

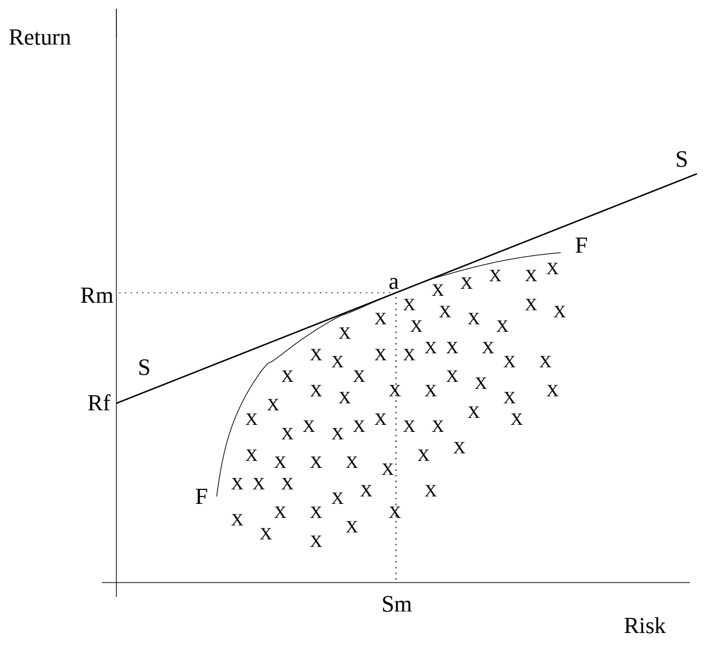


Figure 33: The CAPM model. Pareto front defining trade-off between profit and risk in a portfolio of stocks, and also in relation to a prediction model genus with various model parameters.

In Figure 33 the front FF represents the Pareto optimal portfolios (called *efficient* portfolios in CAPM), with examples of other sub-optimal portfolios lying beneath FF also marked. Line SS is the capital market line, with point Rf, where the capital market line intersects the y-axis, representing the level of 'risk free' return available in the market place to the individual (i.e. through treasury bonds). The capital market line is tangential to the efficient portfolio front, the point where it touches the front at a being the optimal *market* portfolio. In the simple illustration shown in Figure 33, by investing in the market portfolio at point a, and lending or borrowing at the risk free rate Rf, it is possible to operate on the capital market line, gaining a higher rate of return for any level of risk than that possible by investing in an efficient portfolio of stocks. More complex interactions can also be modelled within the CAPM framework. For example in those cases where there are two different zero-risk rates in the market (that available to the user when borrowing, and that available from government bonds) the situation illustrated in Figure 34 occurs.[1]

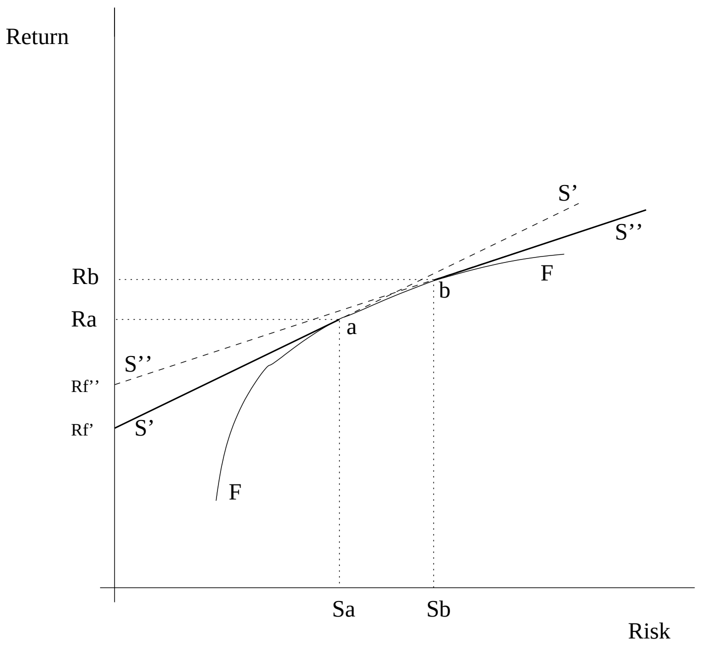


Figure 34: Two risk free rates of interest in the CAPM model (and forecast model analogy).

Here the rate of return demanded by lenders is Rf'', whereas the 'risk free' rate of return for investors in bonds is lower at Rf'. The two tangential lines generated are $S'S'$ and $S''S''$, with the kinked capital market line itself a combination of the two (represented as a solid line). The central section of this line is described by the efficient portfolio front between portfolios a and b. In

[1]The divergence of risk levels occurs in actuality as the Bank needs to make some profit, or at least cover its costs, even if it thinks the borrower is zero risk. In the case where the bank does incorporate an addition risk premium to cover then the divergence between the two rates, the proportion of the efficient frontier consisting of efficient portfolio points will be even higher.

this situation the user therefore desires to know the portfolios described by points a and b, and all those in between on the efficient portfolio frontier. The rates of risk and return described by the capital market line to the left of a can be accessed by distributing the individuals wealth between government bonds (and gaining Rf' return at zero risk) and portfolio a (and potentially gaining Ra return at a risk of Sa). The risk and return levels described by the capital market line to the right of b can be accessed by the individual borrowing from the market at the rate Rf'' and investing this, and all their other wealth in portfolio b.

An analogy can be drawn with the prediction of stock market prices. The Euclidean error of a model can be seen as a proxy for a forecast model's risk, and a trading strategy (based around the direction success error for instance) as a measurement of the expected return of a model. In this situation, front FF represents the Pareto optimal set of regression models, and models a to b are the final models desired by the practitioner (to enable operation on the capital market line). In addition, given that different individuals may experience differing Rfs (due to differing costs of borrowing and lending available to different individuals and institutions in the economy), points a and b will vary across individuals.

9.4 Problems with current approach

An illustration of the problems associated with the current approach to multi-objective training in NN regression is provided in Figures 35 and 36.

Consider the situation where a number of errors measures are used that lie in the range [0,1]. Given that the practitioner wishes to minimise these errors, the typical approach in linear sum back-propagation is to minimise the composite error ε_C. In the D error case (where there are D errors to be minimised) this is:

$$\varepsilon_C = w_1\varepsilon_1 + w_2\varepsilon_2 + \ldots + w_D\varepsilon_D \, , \, \sum_{i=1}^{D} w_i = 1 \, , \, \forall i \; 0 < w_i < 1 \tag{48}$$

In the $D = 2$ dimensional case illustrated in Figures 35 and 36, where the practitioner gives equal weighting to both errors, and both errors lie upon the same range, this is calculated as:

$$\varepsilon_C = 0.5\varepsilon_1 + 0.5\varepsilon_2. \tag{49}$$

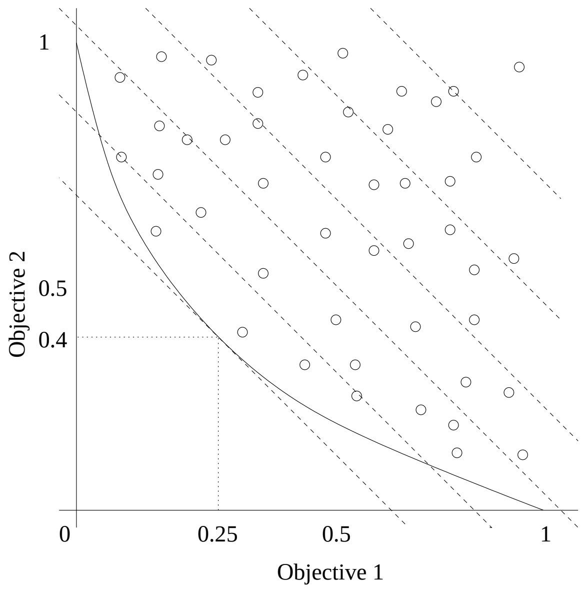


Figure 35: Two dimensional error surface 1. Suboptimal models denoted by circles. The optimal model returned by equal weighting of the errors highlighted at the tangent point.

This approach implicitly assumes that the interaction between the two error terms is symmetric. Consider Figures 35 and 36. Figure 35 illustrates the situation described, where the minimum error surface defined by the problem is shown, with suboptimal models lying behind it denoted by circles. On its extremes it can be seen that the error combinations (0.0, 1.0) and (1.0, 0.0) are possible, which define the axial symmetric hyper-boundaries of the front. In applying Equation 49, each dashed line shown represents a set of objective combinations that are ranked as equivalent (the lines gradient reflecting the prior weightings). As can clearly be seen, if the Pareto front is reached by the training process, then the model returned is one tangential to one of these parallel lines. In the case of Figure 35 this model can be seen to have the error properties (0.25, 0.4). Figure 36 illustrates the same situation, with identical hyper-boundaries but a slightly different degree of convexity of the front. In this case the model returned is defined by the error properties (0.3, 0.5). The two models are significantly different, and in both cases, due to the shape of the Pareto error fronts (and contrary to the desires of the user), the error properties of the models returned are not equal. Although the feasible range of both error measures are the same, the interaction of the errors, as demonstrated by the shape of their true Pareto fronts, results in the return of models, that though Pareto optimal in themselves, do not represent the preferences of the practitioner. An even worse situation arises

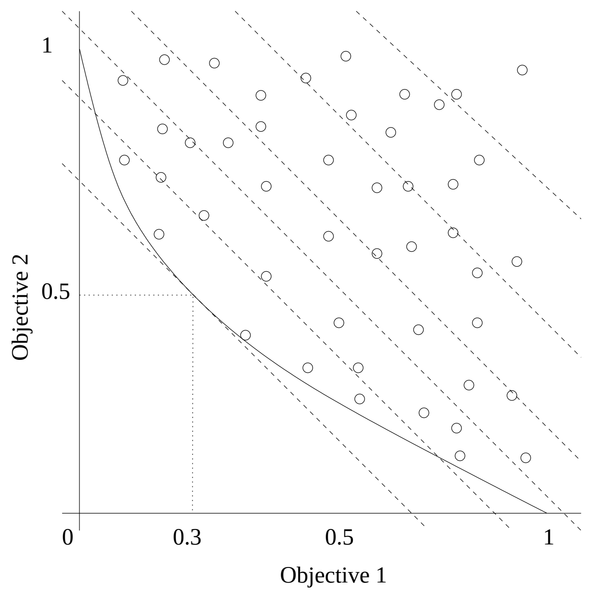


Figure 36: Two dimensional error surface 2. Suboptimal models denoted by circles. The optimal model returned by equal weighting of the errors highlighted at the tangent point.

if the true Pareto front is non-convex. In this case composite error training (if the Pareto front is reached) will only return those models that are on the extremes of the Pareto front, as illustrated in Figure 37.

This is irrespective of the values used for w_1 and w_2. The model returned will always be the one that strictly minimises one of the objectives (errors). This problem with the linear weighting approach has been know for a number of years in the MOEA literature, however it has not been addressed by those using linear weighting to propagate multiple objectives in NN training.

Given the observed need for multi-error training methods for NNs - and the problems highlighted with propagating a linear sum of errors through gradient descent training techniques, using the methods described in the previous Parts of the thesis to generate a general multi-error/objective NN training framework is an obvious progression.[2] The next chapter outlines this framework, in relation to the existing evolutionary neural network literature (for uni-objective problems), and highlights the unique generalisation problems that Pareto multi-objective NN training faces.

[2]Indeed a general model could be envisaged in this context for *any* parameterised model for signal forecasting/function mapping where errors are present

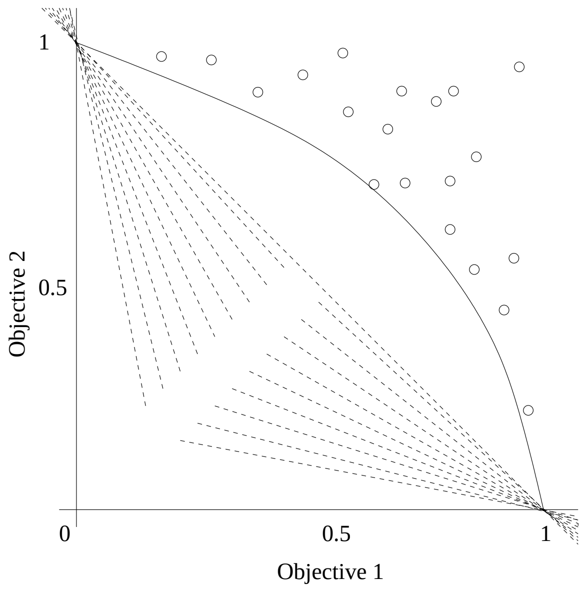


Figure 37: Example the effect of composite weighting when the front is convex with respect to the origin. Irrespective of weights given to the respective errors, the optimal model returned will only be one of the extreme optimal solutions.

Chapter 10

A Pareto Neural Network Training Model

In this chapter an overview of a novel general Pareto NN training model is provided, based on the novel contributions made in the previous chapters. Some ENNs from the literature are described and existing techniques to aid generalisation from these uni-objective models are shown to be inadequate for application to the multi-objective domain. New generalisation techniques are therefore derived and compared. The final model will be empirically validated in Chapter 11.

10.1 Pareto optimal multi-objective evolutionary neural networks

Selecting the correct inputs for a NN for its given task is important. If the NN is not given enough relevant inputs, then there will be still be error present in the model which is systematic (can be explained). If spurious inputs are put into the model additional error may be generated as some of the relationships modelled by the weights fitted during training will themselves be spurious, causing systematic bias and extraneous noise. In a linear model inputs are selected prior to model fitting using various correlation measures (e.g. linear, Kendall's tau, Spearman's rank, partial auto correlation functions), or by removal after fitting using statistical significance tests (t-tests) on the effect of the removal of an input on the model's residual error. In non-linear modelling there is no single robust and widely accepted method for model selection.

A number of algorithms are in use that prune (remove) extraneous nodes and/or weights from a network. Moody (1998) mentions a number of model based methods, including the constructive algorithm (SNC) which adds nodes, sensitivity based pruning algorithms like optimal brain surgery (OBS) and optimal brain damage (OBD) and pruning algorithms based on principal components, PCP. The reason often cited for pruning is the need for bias/variance trade-off (also referred to as the sensitivity/specificity trade-off); too large a network will over-generalise, and too small a network will not form a suitably diverse representation. Smoothing regularisers also do a similar job through weight decay Bishop (1998). Finally much work has been produced in the last decade on the use of EAs for simultaneously optimising NN architecture and weights (through exclusively EA means or hybrid EA/gradient descent methods). This is of significant relevance to the thesis as, just as it is accepted that no single NN topology is optimal for any given problem, it may be assumed that no single NN topography is optimal for a given set of non-commensurable error combinations in relation to any single problem. Just as the set of NN models is needed to represent the Pareto error surface defined by a problem need to be heterogeneous in their weight vector, so they may need to be heterogeneous in their features and architectures.

10.2 Uni-objective evolutionary neural networks

Since an extensive review on this matter is already available in the literature by Yao (1999), this section will restrict itself to a basic overview of how common EC techniques have been applied to ENN training, before Section 10.3 which develops a general model for MOENN training. The section commences with brief discussions of techniques which maintain a single network that is evolved from one iteration to the next (Sections 10.2.1-10.2.2), and then introduces techniques which maintain populations of diverse networks which are evolved as a group at each generation (Sections 10.2.3-10.2.4).

10.2.1 Simulated annealing

Simulated annealing is a generalised Monte Carlo technique that uses a monotonically decreasing variance controlled by a temperature annealing schedule, and was used to adapt the weights of an

NN in Porto *et al.* (1995). The process can be visualised as a ball rolling down a steadily changing non-linear surface in side a container, with the container being shook repeatedly to enable the ball to jump out of local minima, but with the amount of shaking (temperature) decreasing over time.

Given that E_x is the total error over all patterns with the n-dimensional network weight vector $\mathbf{x}$, the simulated annealing algorithm used in Porto *et al.* (1995) is described in Algorithm 7.

Algorithm 7 Simulated annealing of NN weight space.

```
 1: Let $\mathbf{x}_0$ be an arbitrary starting neural network weight vector (either specified or selected at
    random).
 2: Set initial temperature $t$.
 3: Calculate $E(\mathbf{x}_0)$.
 4: If $E(\mathbf{x}_0) < \varepsilon$ halt.
 5: Generate $m$ independent standard normal variates $Y_1, \ldots, Y_m$ and compute the components
    of $\mathbf{U} : U_i = Y_1 / \left(Y_1^2 + \ldots + Y_m^2\right)^{0.5}, i = 1, \ldots, m$.
 6: Set $\mathbf{x}^* := \mathbf{x}_0 + (\Delta r)\,\mathbf{U}$.
 7: Calculate $E(\mathbf{x}^*)$.
 8: If $E(\mathbf{x}^*) \leq E(\mathbf{x}_0), \mathbf{x}_0 := \mathbf{x}^*$, if $E(\mathbf{x}^*) < \varepsilon$, halt, otherwise, go to 4.
 9: If $E(\mathbf{x}^*) > E(\mathbf{x}_0), p := \exp\{-(E(\mathbf{x}^*) - E(\mathbf{x}_0))/t\}$. Generate a uniform [0,1] variate $V$
    (N.B. in (Porto et al. , 1995) a Cauchy distribution was used).
10:     a) If $V \geq p$, go to 5.
11:     b) If $V < p$, $\mathbf{x}_0 := \mathbf{x}^*$, go to 5.
12: end
```

Δr and t are determined empirically. ε is the error level desired of the NN. [1] If the temperature t is held constant, then the algorithm reverts to a pure Monte Carlo method, and if $t = 0$ it approximates a gradient decent algorithm. In general the temperature is calculated as a function of time (in (Porto *et al.* , 1995) it was set inversely proportional to the number of iterations).

10.2.2 Single-agent stochastic search

Random search techniques were traditionally based around single agent stochastic search strategies, which have also been used to train recurrent neural networks, as in McDonnell & Waagen (1994). In this procedure a search point (network weight vector) is perturbed by a uniform random variable. A more advanced variant uses an adaptive bias vector to add momentum to the search and a function

[1]This highlights a general problem that can be seen in number of ENN studies - that of arbitrary algorithm termination methods. Unless there is a reasonable value for ε known *a priori*, which by definition must be application dependent, then these methods can be susceptible to a degree of selection bias (i.e. running the algorithm a number of times with different ε and seeing which value gives the best generalisation error). This differs significantly from gradient descent training - when a validation set can be used to determine algorithm termination for example, which is a general (data independent) method.

evaluation to drive the search in a particular direction (gradient). The variance of the uniform perturbation ξ is then controlled by the repetition of successes, *scnt*, and of failures, *fcnt*, of the search to reach a higher performing point (as is described in Algorithm 8).[2]

Algorithm 8 Single-Agent Stochastic Search (algorithm taken from (McDonnell & Waagen, 1994)).

```
 1: Initialise the NN weight vector $x_0$ and bias vector $b_0 = 0$.
 2: Set maximum number of generations, $N$.
 3: Set number of repeated successes $scnt = 0$.
 4: Set number of repeated failures $fcnt = 0$.
 5: Fix expansion constant, $ex$.
 6: Fix contraction constant, $ct$.
 7: Fix $Scnt$, the number of repeated successes that trigger a change in perturbation variance.
 8: Fix $Fcnt$, the number of repeated failures that trigger a change in perturbation variance.
 9: Fix upper and lower bounds of perturbation variance $\sigma_{ub}$, $\sigma_{lb}$ and initialise variance $\sigma_0 = 1$.
10: Set generation counter $t := 0$.
11: Set $\sigma_{t+1} := \begin{cases} ex \cdot \sigma_t & \text{if } scnt > Scnt \\ ct \cdot \sigma_t & \text{if } fcnt > Fcnt \\ \sigma_{ub} & \text{if } \sigma_t < \sigma_{lb} \\ \sigma_t & \text{otherwise} \end{cases}$
12: Generate a multivariate Gaussian random vector $\xi_t \sim N(b_t, \sigma_{t+1})$.
13:     If $E(x_t + \xi_t) < E(x_t)$, then $x_{t+1} = x_t + \xi_t$ and $b_{k+1} = 0.4\xi_k + 0.2b_k$, $scnt := scnt + 1$,
        $fcny := 0$. Go to 16.
14:     If $E(x_t - \xi_t) < E(x_t) < E(x_t + \xi_t)$, then $x_{t+1} = x_t - \xi_t$ and $b_{k+1} = b_k - 0.4\xi_k$,
        $scnt := scnt + 1$, $fcny := 0$. Go to 16.
15:     $x_{t+1} = x_t$ and $b_{t+1} = 0.5b_t$. $fcnt := fcnt + 1$, $scnt := 0$.
16: If $t = N$, stop, else $t := t + 1$, go to 11.
17: end
```

The contraction constant *ct* and expansion constant *ex*, which define the rate of reduction or increase of the variance are user defined, as well as the upper and lower bounds of the perturbation variance. *Scnt* and *Fcnt* (limits to number of successes/failures before perturbation variance is altered) are also user defined.

10.2.3 Evolution strategy

There are a number of different variants of ES used in the literature with relation to ENNs, however in general they take the following form in relation to weight determination:

$$w_{i,k+1}^n = w_{i,k}^n + \gamma_i^n \cdot \Theta \tag{50}$$

[2]Interestingly a similar approach is recommended in some Monte Carlo methods, when adjusting parameter perturbation variances during ‘burning-in’ to ensure a certain acceptance rate (Denison *et al.* , 2002)

where $w_{i,k}^{n}$ is the i^{th} weight of the n^{th} network in the population at the k^{th} epoch of training, $\Theta \sim N(0, \omega)$ and γ is some multiplier. There does not seem to be any uniformity with regards to the selection of these variables in the literature, and optimal values tend to be problem specific.

In Greenwood (1997) and Saravanan & Fogel (1998) ω is set as 1 and in Belfore & Arkadan (1997) 0.1. In Fogel *et al.* (1995) and Porto *et al.* (1995) ω is set as the parent's mean square error. In Berlanga *et al.* (1999) however ω is adaptive, where it is determined by an associated value to each weight ω_0, where $\omega_i = \omega_0 \cdot \exp\{\sim N(0, \Delta\omega)\}$.

In Belfore & Arkadan (1997), Berlanga *et al.* (1999) and Fogel *et al.* (1995) γ^n is a fixed value for all networks (at 1), and in Fogel *et al.* (1997), Greenwood (1997; 1997), and Saravanan & Fogel (1998) γ^n is variable.

In Fogel *et al.* (1997), Greenwood (1997), and Saravanan & Fogel (1998) γ^n is adaptive such that $\gamma_{k+1}^{n} = \gamma_k^n \cdot \exp\{\tau' \cdot \Theta + \tau \cdot \Theta\}$, where $\tau' = (2I)^{-0.5}$ and $\tau = \left(2I^{0.5}\right)^{-0.5}$ and I is the number of weights in the ENN. In Greenwood (1997) a lower limit is placed on γ^n, with its value being determined by the mean value of the γs of its two parents (in this situation the offspring is the copy of a population member that is ranked worthy to have an offspring, and its γ is determined by this parent and a randomly chosen member of the population). By contrast in Fang & Xi (1997) and Yao *et al.* (1996) $\gamma^n = \lambda_i \cdot (\phi^n / \Phi_{sum})$, where ϕ^n is the fitness of the n^{th} member of the population (network), Φ_{sum} is the sum fitness of the entire population and λ_i is an evolving weight (between the range [0,1]). The determination of λ_i is undefined in Yao *et al.* (1996). In Fang & Xi (1997) λ_i is user defined (set at 1.2 and 1.1) and ϕ^n and Φ_{sum} are the respective error terms, and not fitness (the distinction being errors are minimised whereas fitness is maximised).

In most studies a given ENN's ability to reproduce into the next epoch's (generation's) population is determined by its relative average error, either in relation to the training set, or a validation set. The population of networks is ranked at each generation (epoch), with the fittest networks having a higher probability of offspring in the next generation.

The exact selection methods vary from study to study. In Porto *et al.* (1995) members of the ENN population are compared to 'c' randomly selected other populations and they are assigned a rank dependent upon how many of the c individuals it is fitter than (i.e. every individual will be given a rank between 0 and c). The proportion of the population with the highest ranks is then selected as parents for the next generation.

Greenwood (1997) used a (15, 105)-ES, that is, of a population of 105, the 15 fittest (as ranked by the error term used) each have 7 offspring, which generates the next generation.

In Fang & Xi (1997) the top 50% are transferred to the next generations population, one copy without perturbation and one with perturbation. Whereas in Belfore & Arkadan (1997) out of a population of 100, 10 offspring in the new generation are created from the fittest individual (one of them without perturbation), the next 80 being generated from the next 20 ranked individual and the final 10 being randomly chosen and mutated from the entire population. A general algorithm for evolutionary neural network training with ES is provided in Algorithm 9.

Algorithm 9 General algorithm for evolution strategy neural network training used in the literature.

```
 1: Initialise the $\mu$ network weight/topography vector(s) $\mathbf{x}$.
 2: Set maximum number of generations, $N$.
 3: Initialise generation counter $t := 1$
 4: Set probabilities for mutation and/or node deletion/addition.
 5: Set perturbation multiplier $\gamma$.
 6: Fix perturbation distribution $\Theta$.
 7: Generate $\lambda$ offspring from $\mu$ parents.
 8: Evaluate offspring and select new $\mu$ population.
 9: $t := t + 1$.
10: If $t = N$ terminate algorithm, otherwise go to 7.
11: end
```

Typically the offspring generation (Algorithm 9, line 7) will entail either simple perturbation of the network weights and topography, or consist of perturbation followed by gradient descent training (e.g. Yao & Liu (1997)). As discussed previously, the parameter γ may also be adjusted during the iteration process.

10.2.4 Genetic algorithms

Binary string representation is the common mode of representation of weights in GA approaches to ENN training. For instance, in the work of Janson & Frenzel (1993), each weight is represented by 32 bits, with a network containing 37 adjustable weights, resulting in a chromosome 1,184 bits in length. Maricic (1991) used 8 bits per weight, resulting in a weight resolution/granularity of 0.0078 over the interval [-1.0, 1.0] (that is, a weight can only be adjusted by increments of 0.0078). Genetic algorithms and punctuated equilibria (GAPE) neural network learning, Brill *et al.* (1992), is implemented in the same fashion as a standard GA, except that there are a number of separate populations that only interact after a specified number of generations (this approach is also referred to as island populations). Population based incremental learning (PBIL), used in Baluja (1996), is a variation of the GA architecture where the population is represented in terms of probability vectors. Again the weight vectors and/or node information is transformed into a binary representation (usually constrained by a pre-specified range). In PBIL, however, as the population search proceeds, a probability is given to each bit position with respect to its association with a high fitness solution. Initially these values are set to 0.5 (from the range [0,1]). This is then subject to an update rule as shown in Equation 51

$$P_{i,t+1} = (P_{i,t} \cdot (1.0 - LR)) + (LR \cdot s_i) \tag{51}$$

where $P_{i,t}$ is the probability of generating a 1 in bit position i at time step t and s_i is the value of the ith element of the highest evaluated vector (neural network) in the current generation and LR is the learning rate (user defined). The probability vector is also moved toward the complement vector of the worst solution in the population (on those bit positions where it differs from the fittest solution). Mutation is applied like a standard GA, but with a smaller rate than that of LR. Crossover however was not implemented in PBIL.

Using these works as a foundation, a general model for the multi-objective optimisation of ENNs will now be introduced.

10.3 The general model

This section provides a synthesis of work from the ENN literature and the MOEA literature, and outlines general framework for Pareto MOENN training. In this framework a set of estimated Pareto optimal ENNs is maintained in tandem with the training process through the dominated and non-dominated trees introduced in Part I of this thesis. Four evolutionary operators are recommended within the evolutionary algorithm training process itself - however their individual use is dependent on the optimisation process used (ES, GA or PSO). Specific methods from the evolutionary neural network literature are not directly applicable to the multi-objective domain, and some techniques are impossible to transfer. These issues will be discussed towards the end of this chapter, highlighting that, even in comparison with Parts I and II, the extension of ENN to the multi-objective domain is by no means trivial.

In the general MOENN model presented here, as in previous studies, ENNs are stored within chromosomes, with the representation of the *direct encoding* form (Yao, 1999), that will now be described.

Given a maximum size for a four layer feed-forward MLP MOENN of I input units (features), H_1 and H_2 hidden units in the first and second hidden layers, and O output units, the chromosome length used to represent this network within an MOEA is of size S, where

$$S = (I+1) \cdot H_1 + (H_1+1) \cdot H_2 + (H_2+1) \cdot O + I + H_1 + H_2 + D \tag{52}$$

The first $(I+1) \cdot H_1 + (H_1+1) \cdot H_2 + (H_2+1) \cdot O$ genes are floating point and store the weight parameters (including biases) of the NN, the next $I + H_1 + H_2$ are bit represented genes, whose value (0 or 1) denotes the presence of a unit or otherwise (in the two hidden layers and input layer). The next D genes are again floating point and these are used to hold the D error values associated with the network on the training data.

(1) Topology/feature selection through node addition/deletion

GAs have been the main evolutionary search process used in the population based MOEA literature over the last 15 years (Coello, 1999; Fonseca & Fleming, 1993; Fonseca & Fleming, 1995; Hajela & Lin, 1992; Horn *et al.* , 1994; Schaffer, 1985; Veldhuizen & Lamont, 2000a; Veldhuizen & Lamont, 2000b; Zitzler, 1999; Zitzler *et al.* , 2000; Zitzler & Thiele, 1999). Other EA approaches have also been in addition. One of the earliest works in the area by Beale and Cook (1978), and a number of other more recent models also use ES (Hanne, 2000; Knowles & Corne, 1999; Knowles & Corne, 2000).

Topography and input feature selection is implemented within this multi-objective evolutionary neural network model by bit mutation of the section of the chromosome representing the network architecture. This is facilitated by first determining a superset of input features and maximum hidden layer sizes. Once this is determined, any chromosome has a fixed maximum representation capability. Manipulation of structure is stochastic. By randomly bit flipping members of the first I genes of the binary section of the chromosome the set of input features used by the network is adjusted. This is illustrated in Figure 38, where network A is mutated into network B by flipping

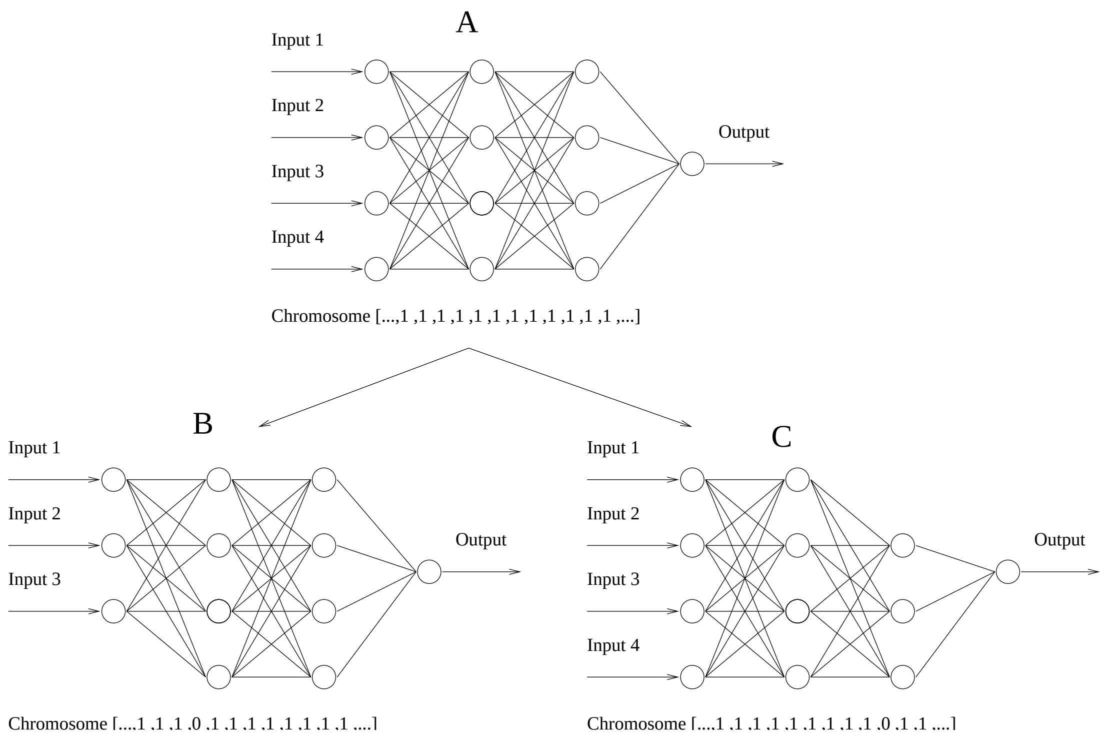

A
Input 1
Input 2
Input 3
Input 4
Output
Chromosome [...,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,...]
B
Input 1
Input 2
Input 3
Output
Chromosome [....1 ,1 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ....]
C
Input 1
Input 2
Input 3
Input 4
Output
Chromosome [....1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,0 ,1 ,1 ....]

Figure 38: Illustration of network topology and feature adaptation through genetic bit mutation.

the 4th gene in the binary segment of the chromosome. Switching the value of genes in the rest of this chromosome section affects the hidden topography of the network, again illustrated in Figure 38 as the mutation of network A into network C by flipping the 9th binary gene.

(2) Weights adjustment

In ES, the weight space of a network is perturbed by a set of values drawn at each epoch (generation) from a known distribution (Gaussian, Laplacian, etc), as shown in Equation 50 in Section 10.2.3.

Direct adjustment of weight values is implemented in this ES fashion. The implementation of weight perturbation of member gene of a chromosome has a set probability, an illustration of this is shown in Figure 39, where the probability of perturbation is set at 0.20.

In the empirical section of this part the variance and probability of perturbation is fixed - however this is not determined within the general model (the practitioner can of course use whatever mutation method they prefer).

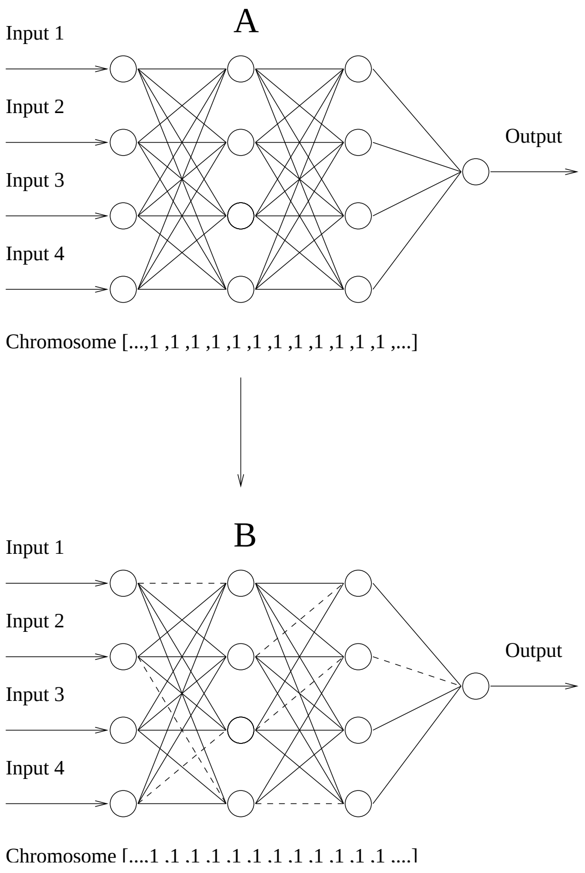

A
Input 1
Input 2
Input 3
Input 4
Output
Chromosome [...,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,...]
B
Input 1
Input 2
Input 3
Input 4
Output
Chromosome [....1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 ....]

Figure 39: Illustration of probabilistic weight perturbation. Dashed connection in network 'B' representing original weight value in network 'A' plus perturbation of $\gamma \cdot \Theta$, where Θ represents values generated by a distribution selected by the user.

(3) Weight addition/deletion

A GA type mutator can also be used to effect the connectivity of individual synaptic weights. An illustration of this is shown in Figure 40, where the probability of weight deletion is 0.05.

Figure 40 illustrates the transformation of a copy of neural network 'A' into 'B' through stochastic weight deletion. Input node 2 can be seen to be only partially connected to the first hidden layer in network 'B', as is the third node of the first hidden layer.

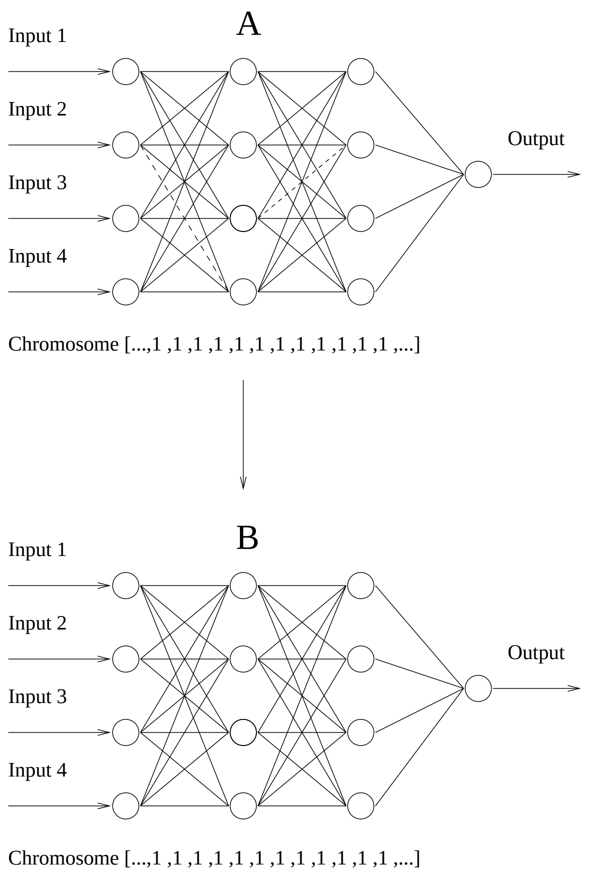

A
Input 1
Input 2
Input 3
Input 4
Output
Chromosome [...,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,...]
B
Input 1
Input 2
Input 3
Input 4
Output
Chromosome [...,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,...]

Figure 40: Illustration of probabilistic weight deletion. Network 'B' represents the original network 'A' after weight deletion of probability 0.05 (the link between Input 2 and the fourth hidden node removed, along with the link between the 3rd and 6th hidden unit.)

(4) Topology/feature selection and weight addition/deletion through crossover

In addition to previous three methods, a GA implementation of the general MOENN framework allows the alteration of both the high level topology of the ENN (the nodes) and the low level architecture (the connectivity) through *breeding* existing ENNs using crossover. Crossover takes place only across the weight space of the network chromosomes.

Figure 41 illustrates two potential offspring, C and D, from the single point crossover of two ENNs, A and B. Child C is constructed with the crossover operator cutting the parent chromosomes at the first weight of the fourth input feature. The weights up to the cutting point are provided by ENN A and those after by ENN B. The ENN C's topology is inherited at random (in the illustration

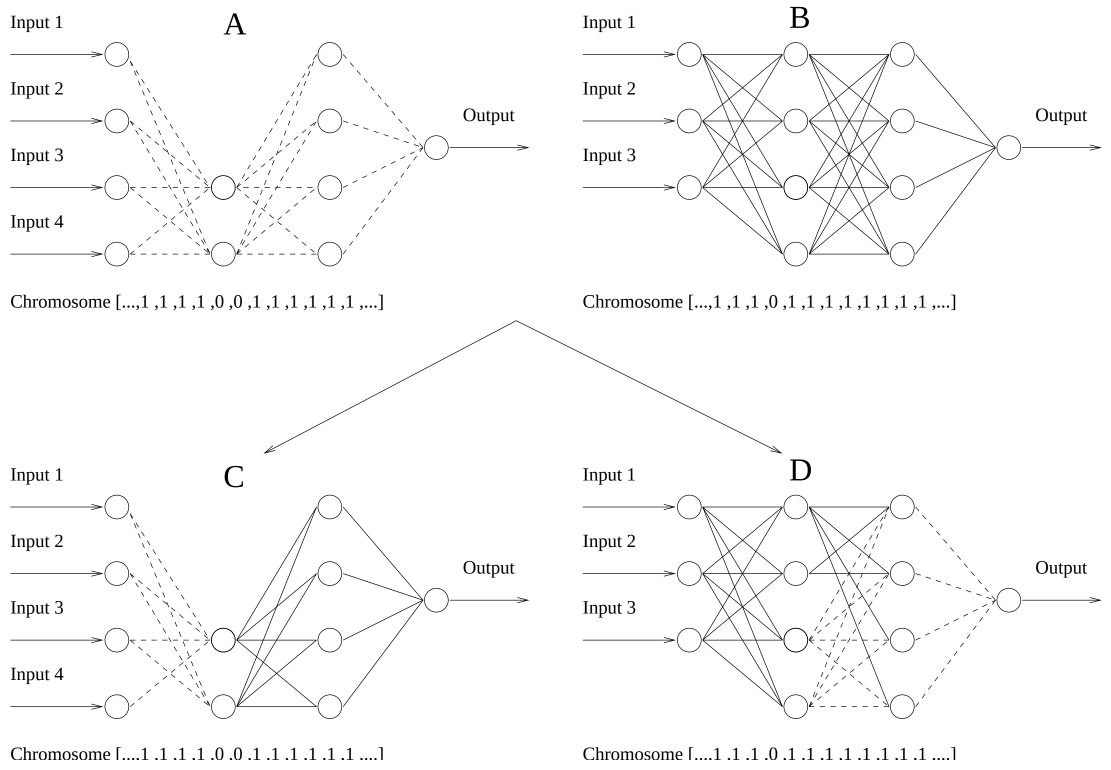

Input 1
Input 2
Input 3
Input 4
A
Output
Chromosome [...,1 ,1 ,1 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,...]
Input 1
Input 2
Input 3
B
Output
Chromosome [...,1 ,1 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,...]
Input 1
Input 2
Input 3
Input 4
C
Output
Chromosome [...,1 ,1 ,1 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,...]
Input 1
Input 2
Input 3
D
Output
Chromosome [...,1 ,1 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,...]

Figure 41: Illustration of network connectivity adaptation through genetic crossover. Dashed connections in offspring 'C' and 'D' represent weights inherited from parent network 'A'. Solid connections represent weights inherited from parent 'B'.

this is from A). As input 4 is not active within network B, the illustration shows that the fourth input node of the child C is only partially connected with its first hidden layer. A similar effect is illustrated in the second example child D. In this instance the crossover operator cuts the parent chromosomes at the second weight of the second node of the second hidden layer. The first section of the weight vector is inherited from B, and the second from A, with B's topology is inherited. The consequence of this is that the second node of the second hidden layer is only partially connected. Further breeding in later generations can lead to networks with more partially connected nodes, though connectivity is only affected in this implementation when ENNs with different topologies are crossed over at a point of node-dissimilarity.

An algorithmic description of the general MOENN framework is shown in Algorithm 10 and flow diagram is shown in Figure 42.

Algorithm 10 Implementation of MOEA in the NN domain (standard training approach), $\mathcal{M}_S$.

```
Input:   $M$, size of initial random population of solutions. Each solution chromosome $\mathbf{x}$
         representing the weights and topology of a NN model.
Output:  A non-dominated set of NN models that are an estimate of the true Pareto front
         defined by the data generation process (represented by the training data), and the NN
         genus.
1:       **Initialisation:** Generate random NN population of size $M$, such that each parameter
         (weight) of the ENNs $\sim N(0,\alpha)$, and the binary part of the chromosome is
         either initialised at 1 or $\sim U(0,1)$. Generate the empty frontal (non-dominated) set
         $F_0 = \emptyset$. Update $F_0$ with the non-dominated solutions from the random population,
         with respect to the chosen error terms.
         Initialise generation counter $t := 0$.
2:       **Frontal Representatives**: Use Partitioned Quasi-Random Selection (PQRS) to
         select network representative(s) from $F_t$, $F_t^R$. Create replica network(s) of $F_t^R$, $H_t$.
3:       **Population Selection**: Use preferred selection method (roulette wheel,
         binary tournament selection, etc) from $X_t \bigcup H_t$ to generate intermediate search
         population $X_t^*$.
4:       **Genetic Recombination**: Adjust weights, topology, connectivity and inputs of NN
         individual(s) $X_t^*$ using EC techniques, dependent on the EA
         process used (e.g. ES, GA or PSO) and user preferred methods.
5:       **Fitness Assignment**: Evaluate the ENN(s) $X_t^*$ with respect to the user
         determined error measures on the training data presented. If $F_t \not\preceq X_t^*$ go to 5
         otherwise go to 6.
6:       **Update Archive:**
         a) Insert ENN chromosome(s) $X_t^*$ into $F_t$ if it is not dominated by individuals in $F_t$.
         b) Remove ENN chromosome(s) from $F_t$ which are dominated by the $X_t^*$.
7:       **Looping**: Iterate epoch count, $t := t + 1$. If stopping criteria have not been met then
         go to 2, else terminate algorithm and save members of $F_t$ for evaluation on test data.
8:       end
```

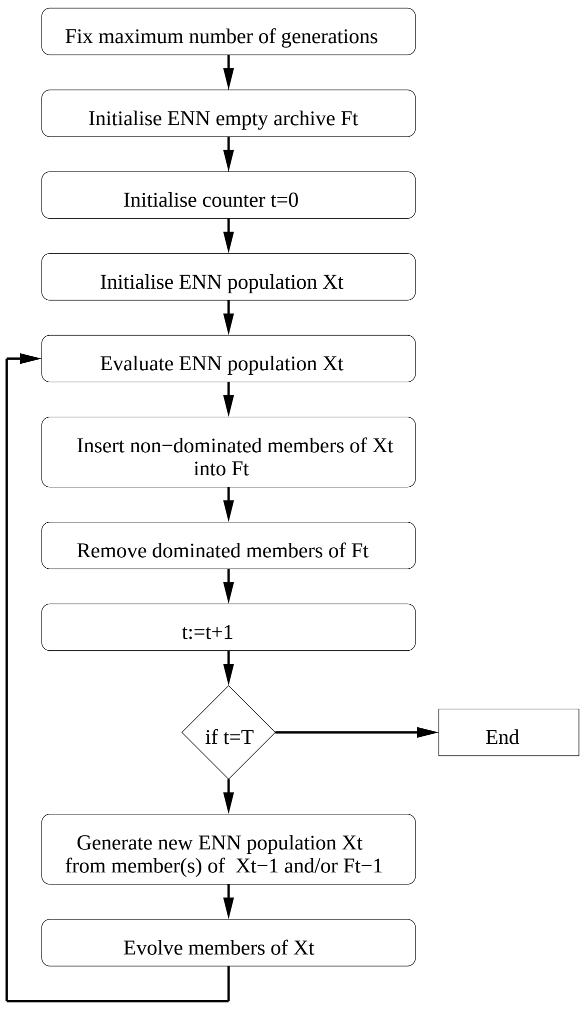

Fix maximum number of generations
Initialise ENN empty archive Ft
Initialise counter t=0
Initialise ENN population Xt
Evaluate ENN population Xt
Insert non−dominated members of Xt into Ft
Remove dominated members of Ft
t:=t+1
if t=T
End
Generate new ENN population Xt from member(s) of Xt−1 and/or Ft−1
Evolve members of Xt

Figure 42: Flow diagram of the general MOENN framework.

10.4 Novel generalisation techniques

As described in Chapter 1 of this thesis, a single error term cannot be meaningfully propagated through a network in a multi-objective application. The hybrid methods used for example in (Yao & Liu, 1997) and highlighted in (Yao, 1999), where individual networks are trained at each generation using a gradient descent technique in addition to their evolutionary manipulation, are infeasible. As such perhaps a greater focus needs to be paid to the perturbation methods used in MOENN as they provide the only method of synaptic adjustment (it also means that the MOENN training process is necessarily slower, again highlighting the importance of the data structures introduced in Part I). Another important issue is the problem of generalisation in the MOENN domain. In order to aid generalisation a commonly used technique is to separate time series data into consecutive training, validation and test sets. The forecast model is then trained on the first set until the measured error beings to increase on the second set (from an optimal minimum), with the final generalisation error calculated on the third set. By stopping training when the observed validation error begins to rise the practitioner aims to prevent *overfitting* of the model on the training data. However the use of validation sets to prevent overfitting of NNs on training data is problematic in the multi-error case where a set of models is returned. For instance a case may arise where some members of the set exhibit falling error values both on training and validation data, whilst others exhibit rising validation error values (or indeed some validation errors rising and some others falling). As such two novel techniques to aid generalisation are now presented, which will be compared in the following chapter.

10.4.1 The use of a 'validation' set

The first new method to improve generalisation is inspired by the traditional validation set approach. As stated previously, this approach cannot be transferred directly to the MOENN domain, however the main thrust of the approach can be recreated. As in the traditional approach, the data set is partitioned so that a portion of the data is separated as an 'unseen' test set on which the generalisation ability of the model(s) will be evaluated. The remaining data is split again to provide a training set and a validation set. A potential solution (ENN) is evaluated with regard to the training set. If this solution is found to be non-dominated by the current archive a copy is created and saved in the archive F(line 5 of Algorithm 11). A second archive however is also maintained in the validation method, which maintains the non-dominated set with respect to the validation data. However, potential solutions are only considered for insertion to this validation archive if (and only if) it has initially been found to be non-dominating with regard to the training data at that epoch/generation. This process continues until algorithm termination, and the set of models returned are those residing in the validation archive, not the training archive.

By only comparing solutions to the validation archive (if they have been accepted previously to the training archive) this method attempts to prevent overfitting to the validation archive. As solutions that are not non-dominating with respect to the validation set are not selected (even if there are non-dominating with the training set), this method also aims to prevent overfitting on the training data. A description this training approach is shown in Algorithm 11.

Algorithm 11 Implementation of MOEA in the NN domain (validation training approach), $\mathcal{M}_V$.

```
Input:   As in Algorithm 10.
Output:  A non-dominated set of NN models that are an estimate of the true Pareto front
         defined by the data generation process (represented by the training and validation
         data), and the NN genus.
1:       **Initialisation:** Generate random NN population of size $M$, such that each parameter
         (weight) of the ENNs $\sim N(0,\alpha)$, and the binary part of the chromosome is
         either initialised at 1 or $\sim U(0,1)$. Generate the empty frontal (non-dominated) set
         $F_0 = \emptyset$, and the empty validation archive $F_0^v = \emptyset$.
         Update $F_0$ and $F_0^v$ with the non-dominated solutions from the random population,
         with respect to the chosen error terms.
         Initialise generation counter $t := 0$.
2:       **Frontal Representatives**: As in $\mathcal{M}_S$.
3:       **Population Selection**: As in $\mathcal{M}_S$.
4:       **Genetic Recombination**: As in $\mathcal{M}_S$.
5:       **Fitness Assignment**: As in $\mathcal{M}_S$.
6:       **Update of Training Archive:**
         a) Insert ENN chromosome(s) $X_t^*$ into $F_t$ if it is not dominated by individuals in $F_t$.
         b) Remove ENN chromosome(s) from $F_t$ which are dominated by the $X_t^*$.
7:       **Update of Validation Archive:**
         a) Insert ENN chromosome(s) $X_t^*$ into $F_t^v$ if it is not dominated by individuals in $F_t \bigwedge F_t^v$.
         b) Remove ENN chromosome(s) from $F_t^v$ which are dominated by the $X_t^*$.
8:       **Looping**: Iterate epoch count, $t := t + 1$. If stopping criteria have not been met then
         go to 2, else terminate algorithm and save members of $F_t^v$ for evaluation on test data.
9:       end
```

10.4.2 Bootstrap generalisation

The second new method to improve generalisation is based on bootstrap techniques. The data is partitioned as in $\mathcal{M}_S$ into a training and test set. The training set is then bootstrap sampled to create n data subsets, on which the neural networks are evaluated during the training process. Potential solution networks produced by the MOEA are initially evaluated with respect to all of the bootstrap sets. Initially this will lead to nD fitnesses associated with a solution (the number of bootstraps multiplied by the number of error terms to be optimised). The final D fitness values attached to a decision vector for the archiving processes are the worst D objective values recorded over the n bootstrap sets.[3] This training method is designed to prevent overfitting on a particular subset of the training data, and also to prevent general overfitting to the training data itself. A description of this training approach is shown in Algorithm 12.

Algorithm 12 Implementation of MOEA in the NN domain (bootstrap training approach),$\mathcal{M}_B$.

```
Input:   $M$, size of initial random population of solutions. Each solution chromosome $\mathbf{x}$
         representing the weights and topology of a NN model.
         $n$, the number of bootstrap subsets generated from the original training sets.
         $s$, the size of the bootstrap subsets.
Output:  As in Algorithm 10.
1:       **Initialisation:** Generate $n$ bootstrap subsets of the training data of size $s$.
         Generate random NN population of size $M$, such that each parameter
         (weight) of the ENNs $\sim N(0,\alpha)$, and the binary part of the chromosome is
         either initialised at 1 or $\sim U(0,1)$. Generate the empty frontal (non-dominated) set
         $F_0 = \emptyset$. Update $F_0$ with the non-dominated solutions from the random population,
         with respect to the chosen error terms (using a solution's worst $D$ terms over the
         $n$ subsets).
         Initialise generation counter $t := 0$.
2:       **Frontal Representatives**: As in $\mathcal{M}_S$.
3:       **Population Selection**: As in $\mathcal{M}_S$.
4:       **Genetic Recombination**: As in $\mathcal{M}_S$.
5:       **Fitness Assignment**: Evaluate the ENN(s) $X_t^*$ with respect to the user
         determined error measures on the training subsets presented. If $F_t \not\preceq X_t^*$ go to 5
         otherwise go to 6.
6:       **Update of Archive:**
         a) Insert ENN chromosome(s) $X_t^*$ into $F_t$ if it is not dominated by individuals in $F_t$.
         b) Remove ENN chromosome(s) from $F_t$ which are dominated by the $X_t^*$.
7:       **Looping**: Iterate epoch count, $t := t + 1$. If stopping criteria have not been met then
         go to 2, else terminate algorithm and save members of $F_t$ for evaluation on test data.
8:       end
```

[3] If bootstrapping is performed without replacement, it is evident that the size of these sets must be smaller than the original training set, otherwise the approach mimics the standard training method.

Models derived from the general MOENN framework using the new generalisation methods will now be compared.

10.5 Empirical comparison of the new generalisation techniques

This section investigates problem of 'noisy' Pareto estimates when approximating functional relationships. The comparison of the new generalisation techniques introduced in the previous section is presented using the data that has been previously used in recent gradient descent approaches to MONN training by Lo & Bassu (2002a). As above, the standard training method is referred to as $\mathcal{M}_S$, and the new validation and bootstrap training methods as $\mathcal{M}_V$ and $\mathcal{M}_B$ respectively.

10.5.1 Application, data and error measures

In the case of signal processing in the presence of an environmental parameter that is impossible or difficult to adapt to, minimising the maximum error of a function approximation becomes important, as highlighted in Lo & Bassu (2002a), leading to the so called *robust* approximation. The generic formulation of robust programming is $b = f(\mathbf{a},c) + \epsilon$ where $\mathbf{a}$ is known, b is observed and c is an uncertain environmental parameter that changes too fast for adaptation. [4] ϵ is some environmental error which cannot be modelled - but for which the distributional properties may be known (an example may be the error inherent in the measuring device which collects the process data). In their recent papers Lo & Bassu, (2002a; 2002b), develop a risk-averting (RA) training criterion that attempts to trade-off Euclidean error minimisation on one extreme with the minimax error criterion on the other, leading to the (λ, p) RA training criterion. In the situation where A and B are respectively sets of observed input-output pairs of the process, their robust error term is formulated as

$$E_{\text{Robust}}(\mathbf{x}) = \exp\left(\lambda \left|y - \hat{y}\right|^p\right) \tag{53}$$

where $\hat{f}\left(\mathbf{A}_k, \mathbf{x}\right)$is the output of the function approximation model with parameter vector $\mathbf{x}$. p is fixed at 2. When $\lambda = 1$ the criterion is identical to Euclidean minimisation and as $\lambda \to \infty$ it approaches

[4]Similarities can of course be made to other forecasting domains. In forecasting economic series (i.e. demand for a good), many explanatory macroeconomic variables may be used, money supply, average level of debt, balance of payments etc. However the modelling of microeconomic events in the economy is usually too difficult, an advertising campaign in a different area of the country, changes in tastes, simple individual choice decisions.

the minimax criterion (Lo, 2002). These errors are then propagated during training, with either a fixed value of λ, or with it changing over time, depending on the NN evaluation.

A number of test functions are developed in (Lo & Bassu, 2002a) in order to demonstrate the effectiveness of their approach, and the trade-off of between Euclidean (maximum-likelihood) fitting and robust fitting. The first of these test function is defined below

$$f(x) = \chi_{[0.1,0.4]}(x) + c\chi_{[0.6,0.9]}(x) \tag{54}$$

where $x \in [0,1]$ and the environment parameter c has an 80% probability of taking that value of 1, and a 20% probability of being zero. The indicator function χ has the property $(\chi_G(x) = 1|x \in G) \wedge (\chi_G(x) = 1|x \notin G)$. A graphical representation of this function, with 6000 randomly (Uniformly) generated points, is shown in Figure 43.

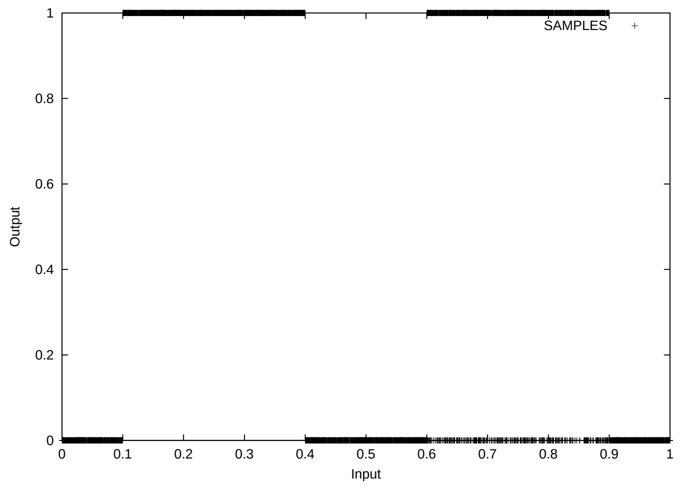


Figure 43: Lo and Bassu's first noisy function (Lo & Bassu, 2002a). (Test function 1).

Their second test function is a noisy sinusoid, with a random phase shift. It is defined as:

$$f(a) = \sin(a + c) + \epsilon \tag{55}$$

where $x \in [0, 2\pi]$, c has a 75% probability of being 0, and a 25% probability of being equal to 2. $\epsilon \sim N\left(0, 0.0706^2\right)$. A plot of 6000 input-output pair generated from this function by Uniformly sampling across the range of a is provided in Figure 44.

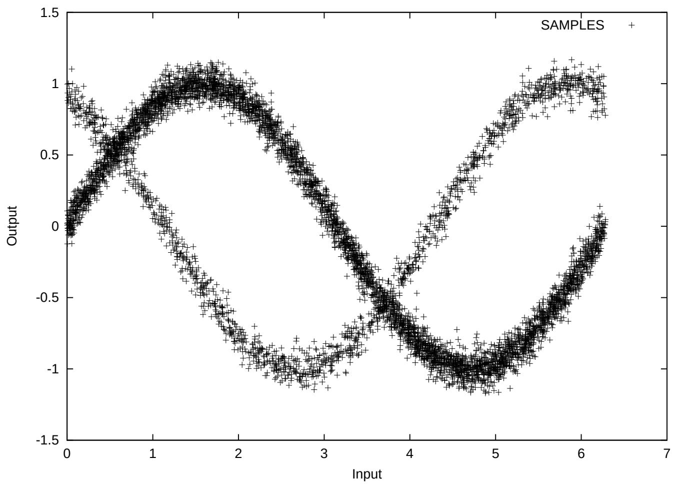


Figure 44: Lo and Bassu's second noisy function (Lo & Bassu, 2002a). (Test function 2).

10.5.2 Methods

A (1+1)-ES MOEA based on the PAES (Knowles & Corne, 1999; Knowles & Corne, 2000) is used in this section to train the Pareto MOENN models (identical to the MOEA used described in Chapter 7). The archive is initialised with 1000 randomly initialised networks for each test function, whose maximum topological representation is identical to those used in Lo & Bassu (2002a) (1:10:1 and 1:5:1). The algorithm parameters are:

- Probability of weight perturbation = 0.2.
- Probability of node deletion/addition = 0.02.
- Probability of weight deletion = 0.02.
- Perturbation $\sim N(0,1) \times 0.1$.
- Initial weights $\sim N(0,1)$.
- Training set size ($|T|$) = 600 patterns (or 400/200 with train/validation) for test functions 1, 300 patterns (or 200/100 with train/validation) for test function 2.

- Bootstrap size = 150.
- Number of Bootstraps = 10.
- Test set size 3000 patterns.
- Training Generations = 1000000 for test functions 1, 250000 for test function 2.

Each algorithm was run 30 times with different initialisation vectors and different random seeds, the two training errors to be minimised are the Euclidean distance and the RA criterion described in Equation 53 (λ set at 20). The evaluated errors of the trained models are the average Euclidean error and maximum Euclidean error. The training size is kept deliberately small and the number of generations used large to help quantify the effect of overfitting when attempting to estimate the Pareto error generating process. The three validation methods described in the previous Chapter are compared, that is $\mathcal{M}_S$ (Kupinski & Anastasio, 1999; Fieldsend & Singh, 2002b), $\mathcal{M}_V$ and $\mathcal{M}_B$.

Two measures are used to compare the different model sets returned by the MOENNs on the test data.

The first of these measures is the $\widetilde{C}$ measure, which has been used previously in this thesis, and counts the proportion in points in one set that are dominated by points in another (Fieldsend *et al.* , 2003). In this section it is used both to estimate the internal consistency of a single set of models evaluated on test data, $\widetilde{C}(\mathcal{M}_S, \mathcal{M}_S)$, and to compare two sets of ENNs returned by two different training methods, $\widetilde{C}(\mathcal{M}_S, \mathcal{M}_V)$. In the first instance, the higher the value returned the greater number of models that are dominated by others of the same set, and therefore the more inconsistent the set is (the further away it is from being a Pareto set with regards to the test data). In the second instance, $\widetilde{C}(\mathcal{M}_S, \mathcal{M}_V)$, it measures how accurate one front is in comparison to another.

The second measure is the volume measure $\mathcal{V}$, which was also introduced in Part I, and again compares the accuracy of two sets of models on the test data.

10.5.3 Results

As the results in Tables 14 - 19 clearly show, the proposed bootstrap training approach, $\mathcal{M}_B$, is markedly superior to both of the other approaches, producing model sets which are more consistent than the other two training methods as well as its test set evaluations being significantly *in front* of the competing approaches using the $\mathcal{V}$ measures.

Table 14: Test function 1. Proportion of archive set fitted on the training data that is non-dominated on test data ($\widetilde{C}(\widehat{F},\widehat{F})$). Means highlighted in bold signify significantly better results under the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail.) Standard deviations in parenthesis.

	$\widetilde{C}(\widehat{F},\widehat{F})$
$\mathcal{M}_S$	0.2421 (0.0240)
$\mathcal{M}_V$	0.3289 (0.0674)
$\mathcal{M}_B$	**0.2207** (0.0166)

Table 14 for instance shows that for test function 1 the set of ENNs returned by the $\mathcal{M}_S$ method contained on average 24.4% dominated ENNs on the test data (i.e. 24.4% suboptimal in comparison to other ENNs of the same set). For the $\mathcal{M}_V$ measure this was 32.9% whereas the $\mathcal{M}_B$ experienced 22.1% self dominated points. For test function 2 this was 10.2%, 12.7% and 6.9% respectively (Table 17). Table 15 shows that ENNs from the $\mathcal{M}_B$ dominated 75.4% of those produced by $\mathcal{M}_S$ on the test function 1 testing data, and 85.4% of those of $\mathcal{M}_V$. The left table in Table 16 shows that the ENN models produced by $\mathcal{M}_B$ lay on average 2.8% ahead on those produced by the other two methods in objective space.

Figures 45 - 46 show the estimated Pareto fronts of a single run by the three methods on the two test functions.

Table 15: Test function 1. Comparison between estimated Pareto fronts on test data from the standard, validation, and bootstrapping training models, using the $\tilde{\mathcal{C}}$ measure. $\tilde{\mathcal{C}}(a,b)$ is the mean proportion of the members of the estimated front produced by the training method 'b' dominated by members of the estimated front produced by the training method 'a'. Means are over 30 runs, with standard deviation in parentheses.

$\tilde{\mathcal{C}}(\mathcal{M}_S, \mathcal{M}_V)$	**0.6134** (0.3075)
$\tilde{\mathcal{C}}(\mathcal{M}_S, \mathcal{M}_B)$	0.1403 (0.2392)
$\tilde{\mathcal{C}}(\mathcal{M}_V, \mathcal{M}_S)$	0.2898 (0.2299)
$\tilde{\mathcal{C}}(\mathcal{M}_V, \mathcal{M}_B)$	0.0970 (0.1211)
$\tilde{\mathcal{C}}(\mathcal{M}_B, \mathcal{M}_S)$	**0.7536** (0.3058)
$\tilde{\mathcal{C}}(\mathcal{M}_B, \mathcal{M}_V)$	**0.8540** (0.2134)

Table 16: Test Function 1. Comparison between estimated Pareto fronts on test data from the standard, validation, and bootstrapping training models, using the $\mathcal{V}$ measure. Where $\mathcal{V}(a)$ is the mean proportion of the volume of the minimum hypercube containing all estimated fronts, which is dominated by members of the estimated front produced by method 'a' . Means are over 30 runs, standard deviation in parentheses, value as a percentage. Results highlighted in bold signify significantly better results under the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail.)

	Total	Exclusively
$\mathcal{V}(S)$	0.7527 (0.0369)	0.0041 (0.0086)
$\mathcal{V}(V)$	0.7456 (0.0277)	0.0030 (0.0068)
$\mathcal{V}(B)$	**0.7917** (0.0173)	**0.0284** (0.0231)

Table 17: Test function 2. Proportion of archive set fitted on the training data that is non-dominated on test data ($\widetilde{C}(\widehat{F},\widehat{F})$). Means highlighted in bold signify significantly better results under the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail.) Standard deviations in parenthesis.

	$\widetilde{C}(\widehat{F},\widehat{F})$
$\mathcal{M}_S$	0.1019 (0.0709)
$\mathcal{M}_V$	0.1569 (0.0696)
$\mathcal{M}_B$	**0.0695** (0.0533)

Table 18: Test function 2. Comparison between estimated Pareto fronts on test data from the standard, validation, and bootstrapping training models, using the $\tilde{C}$ measure. $\tilde{C}(a,b)$ is the mean proportion of the members of the estimated front produced by the training method 'b' dominated by members of the estimated front produced by the training method 'a'. Means are over 30 runs, with standard deviation in parentheses.

$\widetilde{C}(\mathcal{M}_S,\mathcal{M}_V)$	0.1740 0.1551
$\widetilde{C}(\mathcal{M}_S,\mathcal{M}_B)$	0.1545 0.1090
$\widetilde{C}(\mathcal{M}_V,\mathcal{M}_S)$	**0.2789** (0.2077)
$\widetilde{C}(\mathcal{M}_V,\mathcal{M}_B)$	0.2015 (0.1507)
$\widetilde{C}(\mathcal{M}_B,\mathcal{M}_S)$	**0.2979** (0.2482)
$\widetilde{C}(\mathcal{M}_B,\mathcal{M}_V)$	0.2999 (0.2844)

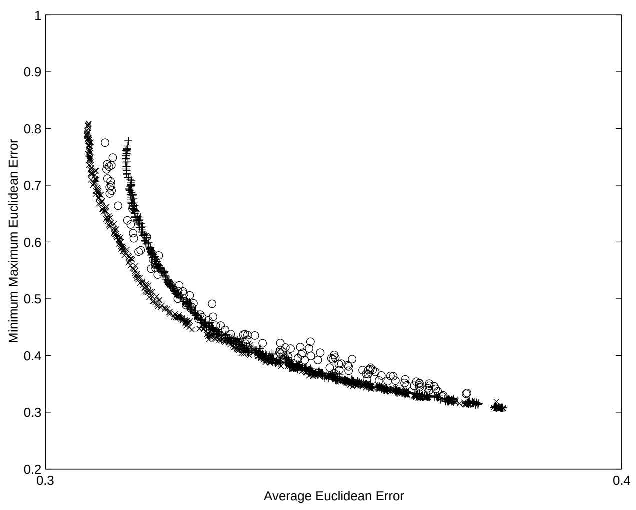


Figure 45: Test function 1, estimated Pareto fronts, points from the standard training method denoted by '+', points from the validation training method denoted by 'o' and points from the bootstrap training method denoted by 'x'.

Table 19: Test Function 2. Comparison between estimated Pareto fronts on test data from the standard, validation, and bootstrapping training models, using the $\mathcal{V}$ measure. Where $\mathcal{V}(a)$ is the mean proportion of the volume of the minimum hypercube containing all estimated fronts, which is dominated by members of the estimated front produced by method 'a' . Means are over 30 runs, standard deviation in parentheses, value as a percentage. Results highlighted in bold signify significantly better results under the Wilcoxon non-parametric signed ranks test (2 tailed, 0.025 in each tail.)

	Total	Exclusively
$\mathcal{V}(S)$	0.8321	0.0021
	0.0270	0.0029
$\mathcal{V}(V)$	0.8310	0.0021
	0.0289	0.0041
$\mathcal{V}(B)$	**0.8561**	**0.0173**
	0.0263	0.0107

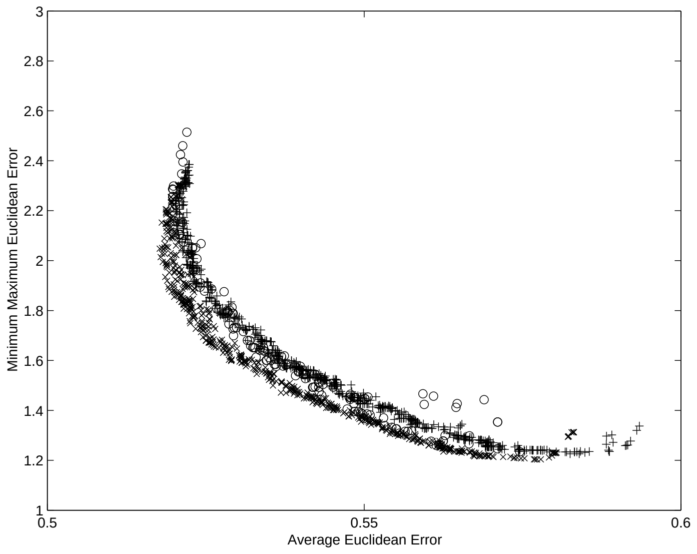


Figure 46: Test function 2, estimated Pareto fronts, points from the standard training method denoted by '+', points from the validation training method denoted by 'o' and points from the bootstrap training method denoted by 'x'.

10.5.4 Comments

These results clearly show that the generalisation from MOENN training can be improved beyond the approach currently taken in the literature (Kupinski & Anastasio, 1999; Fieldsend & Singh, 2002b), by using the bootstrap training method. If this new generalisation technique is used, then the set of ENNs evaluated on the test data can be observed to be significantly more consistent on the two test functions used (i.e. contain significantly less self dominated points). However the improvement is not just manifest in the internal consistency of the fronts, but also in terms of the actual optimality of the Pareto approximation. The estimated Pareto fronts generated by the bootstrap generalisation lie significantly *in front* of those generated by the standard method and the new validation method for both test functions (as measured by the $\mathcal{V}$ measure and $\widetilde{C}$ measure).

In the following section models derived from the general MOENN framework introduced in this chapter will be empirically validated on a number of real world data sets from the finance domain.

Chapter 11

Empirical Application of Pareto Neural Networks

In this chapter a number of experiments are performed to assess the performance of models derived from the general model described in Chapter 10.

- The trade-off between models that maximise profit and minimise Euclidean error is shown in the forecasting of the Dow Jones Industrial Average stock index (Section 11.1).[1]
- Section 11.2 provides an extended set of experiments based on those from Section 11.1, with 37 international index series from financial markets. These experiments use topologically heterogeneous ENNs, maximising return and minimising return variance.

11.1 Proof of concept; initial application of MOENN training to real world data

The first empirical section of this chapter is concerned with applying a model, derived from the general MOENN framework, to a financial forecasting problem. The ENN models are trained in order to forecast an international stock index. The concern in this process is the optimisation of two measures, Euclidean error (minimised) and *return* (maximised). Therefore the final set of achieved

[1]The results from this Section were published in (Fieldsend & Singh, 2002b).

Pareto optimal members, F, should provide an estimate of the trade-off of the accuracy/return defined by the generating process and trading strategy. As this is the first application of MOEA trained ENNs to real world data (the only previous work using an MOEA to train ENNs used a simple synthetic data set (Kupinski & Anastasio, 1999)), it is designed to provide a simple initial model evaluation. The MOEA used is a GA based model, the SPEA of Zitzler (1999), and the topology of the trained ENNs is fixed.

Financial forecasting (modelling the generating process of a financial time series, or process) is a popular application of both NNs and other nonlinear models as discussed in Chapter 9. However in many applications misleading claims are made (or inferred) with regards to the actual efficiency of the models presented. Typically the accuracy of a model will be described for some data set (usually in terms of Euclidean error), and an estimate of the profit generated by using the model forecasts and a trading strategy is provided. However, often the the cost of trading (transaction costs) are not factored into this calculation. These addition costs, typically trading commission plus any taxation that may be relevant (e.g. stamp duty in the UK) can have a significant affect on realised profits. For instance, take a hypothetical model which is found to experience an average profit over a year of test data of 16.2% (excluding transaction costs), which involves actively trading (buying or selling) 25% of the time. If the market return of the financial series over the same period was 6%, then one would assume that there were significant opportunities for realising excess profits by trading using the forecast model. However, by including a reasonable transaction cost level the analysis changes completely. The yearly return of the forecast model infers a daily return of 0.06% compounded over the period (or approximately 0.30% compounded per trade)[2]. With transaction costs of 0.2% per trade the actual profit of the model drops to slightly over 5.1% per annum (0.1% per trade or the equivalent of 0.02% per day) - actually worse than simply investing at the start of the year and waiting until the end. A transaction cost of more than 0.3% would lead the model which initially boasted a yearly return of over 16% to actually realise a loss. As such the approach used in this and later sections is to include transaction costs in the training and final evaluation.

[2]financial markets typically experience 250 trading days per year

11.1.1 Trading strategy and error measures

Due to the difficulty in predicting raw market time series for profitable day trading when transaction costs are taken into consideration, this section combines a number of time series in a novel transformation in order to make the forecasting task easier. The data transformation used is an application specific one (determined by the trading strategy to be used), but has the additional benefit of creating a stationary time series for forecasting. (Other studies have also used data specific transformations, e.g. Saad *et al.* (1998) and LeBaron & Weigend (1998)).

The dependent time series that will be modelled in this section, y_t, is a transformation of the one day return between the open price of a market or individual stock, and the next day realised high, as shown in Equation 56.

$$y_t = \left(\frac{b_t^h}{0.993 b_{t-1}^o}\right) \tag{56}$$

where b_t^o is the open level of a market/stock at day t and b_t^h is the market/stock high at day t.

The trading strategy is dependent on the market/stock level falling during the day by at least 0.7% before buying (trading-in) - as described in Algorithm 13. The purchase cost is therefore 99.3% of the open price. The return measure is calculated using a simple trading strategy based upon transaction costs calculated at 0.1% of price (defined as a reasonable level in Schittenkopf *et al.* (2000)), and therefore a minimum increase in price from buy to sell of 0.2% is needed before any profits can be realised.

In addition, the trading strategy is designed such that a trade into the market will only take place if the estimated profits beyond transaction costs of selling the next day equal approximately 1.5%. This regime is used in order to eliminate trades where the price fluctuation is small – and where a small over estimate of the dependent value will lead to losses. The forecast of the transformed series y_t, $\widehat{y}_t$, therefore needs to be greater than 1.017. The return objective measure is formally described in Algorithm 13.

The measure shows that if $\hat{y}_{t+1}$>1.017 (predicting that tomorrows high is 1.7% higher than the trade-in price on day t), and during the day the price falls to (or below) a level of 99.3% of the open price, trading will occur (Algorithm 13). If this happens, and the realised value of value of y_t is greater than 1.017, then when the market level reaches the point of being 1.7% above the price paid on entrance (when the next days price is 1.0% higher than todays open), the assets will be sold and profits realised (after costs incurred). If however the market level does not reach a level 1.7% above

Algorithm 13 Trading strategy (return objective).

```
1:  $t$, current time step (day).
2:  $\widehat{y}_t$, the model forecast at day $t$.
3:  $q_t^c = \frac{b_t^c}{0.993 b_{t-1}^o}$, where $b_t^c$ is the market close on day $t$.
4:  $\varepsilon_t^{Rt}$, Return value at time $t$ (as a percentage of capital at $t-1$).
5:  Set $t := 1$, first trading day of train (or test) set instance.
6:  If $(\widehat{y}_{t+1} \geq 1.017) \bigwedge \left(x_{t-1}^l / x_{t-1}^o \leq 0.993\right)$ shift capital from risk free deposit into market at
    the point where the market price falls to 99.3% of open (incurring transaction costs), go to
    7, otherwise go to 8.
7:  $t := t+1$, Calculate profit/loss.
8:      If $(y_t \geq 1.017)$, sell when market reaches the level 101.7% of that when entered,
    $\varepsilon_t^{Rt} = 1.5$, go to 2. Else:
9:      If $(y_t < 1.017)$, sell at the end of day, $\varepsilon_t^{Rt} = (q_t^c - 1) - (0.1 + 0.1 q_t^c)$, go to 6.
10: Calculate nominal risk free interest accrued on assets, $\varepsilon_t^{Rt} = 0.0016$ (compound equivalent
    to 4% p.a.) , $t := t+1$, go to 6.
10: Halt process when end of train (or test) set is reached.
12: end
```

the price paid on entrance then the assets are disposed of at the end of $t+1$, with the potential for either profit or loss. If $b_{t-1}^l / b_{t-1}^o > 0.993$, or $\widehat{y}_t < 1.017$ then no trade will occur and the capital will lie in a bank deposit accruing the equivalent of 4% interest p.a. (0.016% a day compounded over 250 trading days).

The second training objective is to minimise the RMSE of the model prediction of y_t - (a direct measure of the standard deviation of the forecast of the model prediction from the actual), equivalent to minimising the Euclidean error. It should be noted that a measure more in keeping with the CAPM framework would be the standard deviation of the model return. This is used in later experimentation (Section 11.2), however, by using RMSE the results from this Section can be used to demonstrate the competition between Euclidean error minimisation and profit maximisation in financial forecasting applications.

11.1.2 ENN model

In order to use the trading strategy introduced, a model is needed to produce a prediction of the time series y_t, $\widehat{y}_t$. To do this, a set of recurrent ENN time series forecast models will be trained (describing the trade-off between the Euclidean prediction error and return objective).

Fifteen explanatory variables were used in the ENNs, and are defined as follows:

$$v_t^{1,\ldots,10} = y_{t-2}, \ldots, y_{t-11} \tag{57}$$

$$v_t^{11,\ldots,15} = \widehat{y}_{t-1}, \ldots, \widehat{y}_{t-5} \tag{58}$$

variables 1 to 10 contain the last 10 lagged realised values of y_t (2 weeks of trading), of course y_{t-1} cannot be used as it incorporates information that will not be available at the start of day at $t-1$. Variables 11 to 15 are recurrent variables. Figure 47 illustrates the network design.

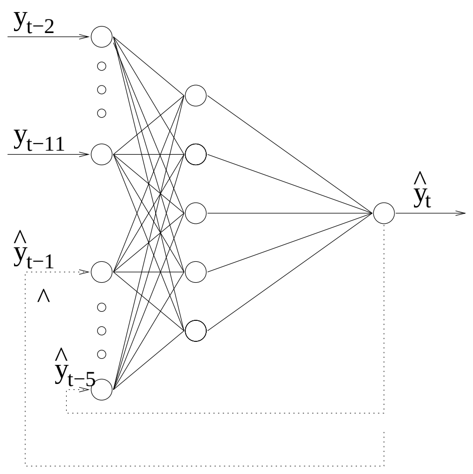


Figure 47: Network design, illustrating topography and recurrence.

11.1.3 Data

The data used in the model is the open, high, low and close of the Dow Jones Industrial Average (DJIA) over the 2500 trading day period from 28/2/1986 to 3/1/2000 (the open and high are needed in the generation of dependent time series y, and the low and close are needed for the profit calculation of the trading strategy. The low is needed to know whether a trade is triggered, and the close is needed for returns where the actual level does not rise as much as predicted). In Section 11.1.4 a sliding window is used to contain the training and test sets which are generated by first creating the relevant explanatory vector and dependent value pairs (embedded matrix), and then passing a window with the first 1000 samples as training data and the next 100 samples as test data across the series, sliding the window forward by 100 samples 25 times. As illustrated in Figure 48 below, this means that the 25 test sets contain a total of 2500 trading days (approximately 10 years) from 12/2/1990 to 3/1/2000.

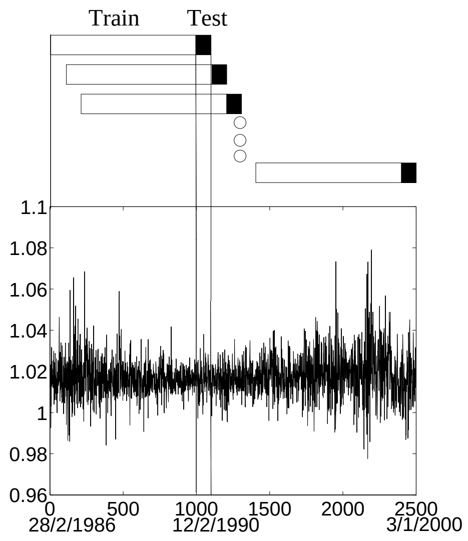


Figure 48: An illustration of the test and training sets (top) in relation to the transformed data y_t (bottom).

11.1.4 Experiments and results

The experiments in this section are designed to demonstrate the feasibility of MOENN, through an application in the time series forecasting domain. It is also designed to highlight the benefit of producing a population of models which lie on an estimate of the Pareto front of the generating process. As stated, this allows the practitioner to choose a model from a viable set that describes their error trade-off preferences after training and therefore knowledge of the training error interactions (instead of the approach of summation, where only one model is returned and where the practitioner must have *a priori* knowledge of the error surface). However, if the error properties do not hold true on the test data, this approach is of no use in the financial domain.

To test this, three preferences of three general practitioners are defined: risk averse (model C), profit maximiser (model A) and middle-way (model B). The relevant models for each of these type of investor are selected at each of the training windows, model C is the ENN set member which has the highest return on the training data, model B is the ENN at the middle of the set (when the set is ordered by one of the two objectives) and model A is the ENN set member which has the lowest RMSE on the training data. The performance of each of these models is then evaluated on the test data. An illustration of where these individuals reside in terms of the Pareto front is shown in Figure 49.

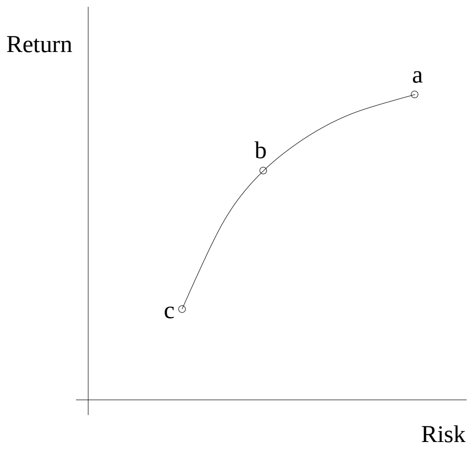


Figure 49: An illustration of the three investor types compared, the profit-maximiser operating at point ‘a’, the middle-way practitioner operating at point ‘b’ and the risk-averse individual operating at point ‘c’.

A multi-objective GA was used, the SPEA of (Zitzler *et al.* , 2000; Zitzler & Thiele, 1999; Zitzler, 1999), updated to incorporate an unconstrained archive and the data structures described in Part I. It was implemented using single-point crossover and the mutator variable was drawn from a zero-mean, symmetric, Leptokurtic distribution (kurtosis ≈ 10) generated by the product of two uniform distributions covering the range [0,1], and a Gaussian distribution with a variance of 0.1 and zero mean. The probability of mutation was 0.1 and the probability of crossover was 0.8. Each population of networks was trained for 2000 generations, with the search population in each instance generated with the search population at the end of the previous training window. For the first DJIA training window the search population ENN parameters where randomly generated from $\sim N(0,1)$. The search population itself was implemented as in (Zitzler *et al.* , 2000; Zitzler & Thiele, 1999; Zitzler, 1999), consisting of 80 individuals updated at each generation using binary tournament selection with (upto) 20 individuals from the archive F.

The average Euclidean error and return for the three practitioners described earlier (selected as those NN models operating at the relevant points on the training set Pareto front), the market return, and the performance of the random-walk forecast of y_t for the 25 test sets are shown in Table 20. (Again, as y_t is not known at the start of day t, due to its use of the daily high of the market, the random walk model takes the form $\widehat{y}_t = y_{t-2}$).

Table 20: Mean risk and return over the 25 test sets for the extreme and mid-way models, the random walk model and the market return (standard deviations in parenthesis).

	Train		Test	
	RMSE	% Ret	RMSE	% Ret
Risk Averse	0.00903 (0.00181)	0.1391 (0.0306)	0.00923 (0.00316)	0.0907 (0.0742)
Middle Way	0.00908 (0.00182)	0.2299 (0.0569)	0.00923 (0.00308)	0.1714 (0.1317)
Profit Maximiser	0.00927 (0.00184)	0.2904 (0.0797)	0.00978 (0.00302)	0.2233 (0.1780)
Market	- -	0.0508 (0.0208)	- -	0.0619 (0.0717)
Random Walk	0.01348 (0.00312)	0.1293 (0.0364)	0.01295 (0.00461)	0.1175 (0.0968)
Risk Free	0	0.0016	0	0.0016

As it can be clearly seen, the model attributes of the different points on the estimated Pareto front on the training data are consistent over the test data also, although with a degree of noise. An example of this is illustrated in Figure 50, with the training Pareto front and estimated test Pareto front plotted for the first training and test window. The mean RMSE of the model B, although above that of the risk averse models on the test sets, is not significantly so. However the central models' mean return is significantly higher, as are the profit maximisers models' return significantly higher than both the central models' return and minimal risk models return. (Calculated using the non-parametric Wilcoxon Signed Ranks Test (Wilcoxon & Wilcox, 1964) at the 2% level (1% in each tail)).

The tabulated results are further supported in a visual fashion by the profit plots over the 10 year period for the various models, which are shown in Figure 51. It is of interest to note that all three ENN model types outperform the market return, however the risk averse models (RMSE minimiser) display a lower return over the period than the simple random walk model on the transformed data,

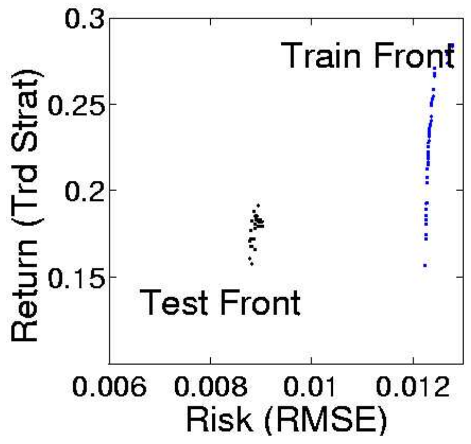


Figure 50: Estimated Pareto error surface on training set and the noisy estimated Pareto error surface on the test set (first window).

once more underlining the fact that models should be trained with respect to the error preferences of the user (models trained strictly to minimise RMSE will not necessarily generate excess profits).

11.1.5 Comments

In this section a novel approach to the construction of financial time series models has been formed by analogy with the CAPM from portfolio theory. Approximate Pareto frontiers have been generated for the DJIA index based on NN model risk and return, showing that the MOENN framework introduced in the previous section is a feasible method for the modelling of these types of problem. As a result of this it has also been demonstrated that risk and return are competing in model parameter specification, and that this generalises to test data.

However there are still further areas of research in this field. Both Kupinski & Anastasio (1999) and the work presented in this section do not fully confront the problem of generalisation/validation in the domain of Pareto population training. The MOEA developed in the literature address 'clean' process domains. In noisy domains such as financial forecasting, where the generating process itself is being modelled, the divergence between the estimated Pareto surface from the training data, and the actual surface defined by the process itself merits much further investigation. This is confronted in the next set of experiments in this chapter, which use a number of 'noisy' test functions used in

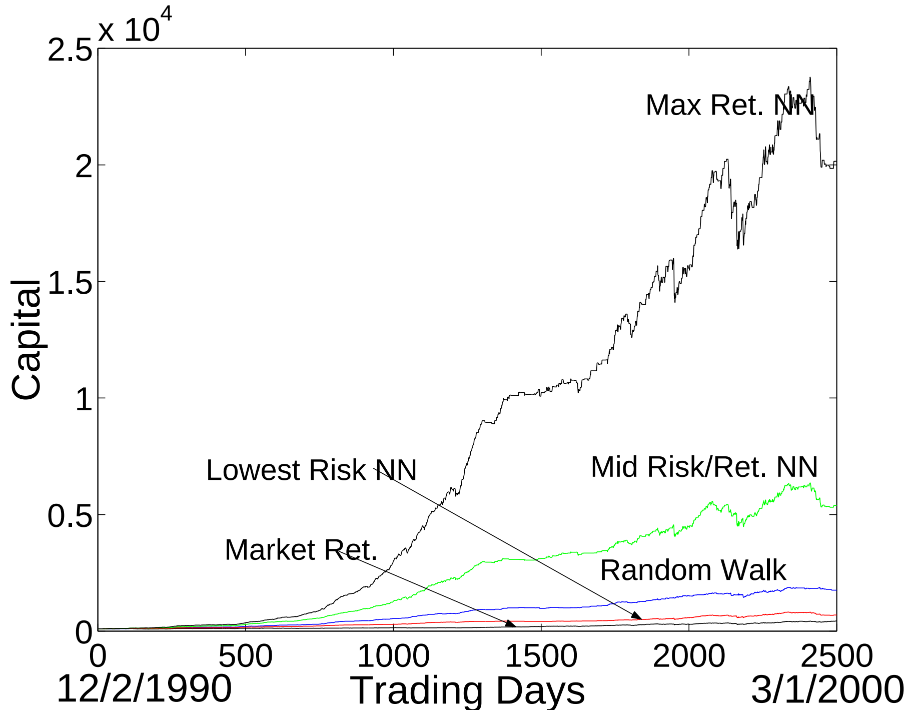


Figure 51: Profit plots for the 10 year test period for the extreme and mid models on the training Pareto front, the random walk model and the market return (capital initialised at 100).

the linear weighting MONN literature.

11.2 Extensive application of MOENNs to regression problems.

The final empirical analysis of this thesis is based on the prediction of thirty-seven international indices. The data itself varies in length across samples, the longest being 4845 data series (over 19 years) and the shortest 416 data samples (under 2 years). All series are daily, containing open, high, low and close, and run until 7th February 2003, and were obtained from `http://uk.finance.yahoo.com/`. A description of these series is provided in Table 21.

The two error measures to be optimised, based on forecasting transforms of these series and using a trading strategy, are again risk (minimised) and return (maximised). The transform and trading strategy used are similar to that introduced in Section 11.1, however the risk term, instead of being the RMSE of the model, has a firmer grounding in Finance theory - being the variance of the realised returns.

The return error measure is almost identical to that introduced in Algorithm 13, but instead of trading dependent on a fall of 0.7%, a fall of 0.5% is used. The strategy is therefore to trade into the market on day t when the level drops by 0.5% of the open value and sell the following day when the level rises 1.7% above the purchase level (1.2% above the open of the previous day), accruing 1.5% profit (including transaction costs of 0.1% each way). If the level does not fall by at least 0.5% then the initial trade does not occur and the capital is invested overnight as a 'risk-free' asset (i.e. a bank deposit earning 0.016% - equivalent to 4% per annum). If the initial trade has occurred and the level does not rise 1.7% above the purchase level the following day, trade out of the market occurs at the market close of day $t+1$ and profit/loss (including transaction costs) occurs. The predicted series ς_t is a composite series based upon this trading strategy, and is described below:

$$\varsigma_t = \left(\frac{b_t^h}{0.995 b_{t-1}^o}\right) \tag{59}$$

$$\text{if}\, b_{t-1}^l > 0.995 b_{t-1}^o,\ \varsigma_t = 1.00016 \tag{60}$$

$$\text{else if}\, \varsigma_t \geq 1.017,\ \varsigma_t = 1.017 \tag{61}$$

$$\text{else}\, \varsigma_t = \left(\frac{b_t^c}{0.995 b_{t-1}^o}\right) \tag{62}$$

where b_t^o is the open level of the market at day t, b_t^h is the market high at day t, b_t^l is the market low at day t, and b_t^c is the market close at day t. Thus ς_t exactly encapsulates the profit/loss of this trading strategy (excluding the transaction costs).

Table 21: Stock index descriptions

Country	Index	From	Until	Samples
Americas				
Argentina	MerVol	24-10-1996	07-02-2003	1538
Brazil	Bovespa	13-05-1993	07-02-2003	2408
Canada	S&P TSX Composite	22-08-1984	07-02-2003	4647
Mexico	IPC	29-05-2001	07-02-2003	416
Peru	Lima General	15-05-1998	07-02-2003	1169
USA	S&P 500	25-11-1983	07-02-2003	4845
Asia & Pacific				
Australia	All Ordinaries	28-08-1984	07-02-2003	4659
China	Shanghai Composite	21-07-1997	07-02-2003	1329
Hong Kong	Hang Seng	18-12-1990	07-02-2003	3000
India	BSE 30	17-07-1997	07-02-2003	1368
Indonesia	Jakarta Composite	18-07-1997	07-02-2003	1336
Japan	Nikkei 225	19-09-1986	07-02-2003	4035
Malaysia	KLSE Composite	21-12-1993	07-02-2003	2247
New Zealand	NZSE 40	06-10-1992	07-02-2003	2590
Pakistan	Karachi 100	23-07-1997	07-02-2003	1322
Philippines	PSE Composite	18-07-1997	07-02-2003	1376
Singapore	Straits Times	10-07-1997	07-02-2003	2651
South Korea	Seoul Composite	18-07-1997	07-02-2003	1357
Sri Lanka	All Share	15-10-1998	07-02-2003	1032
Thailand	SET	18-07-1997	07-02-2003	1362
Taiwan	Taiwan Weighted	18-07-1997	07-02-2003	1351
Europe				
Austria	ATX	27-11-1992	07-02-2003	2517
Belgium	BEL-20	01-07-1992	07-02-2003	2587
Czech Republic	PX50	16-07-1999	07-02-2003	875
Denmark	KFX	11-02-1993	07-02-2003	2499
France	CAC 40	19-03-1990	07-02-2003	3231
Germany	DAX	12-12-1990	07-02-2003	3048
Greece	General Share	13-05-1998	07-02-2003	1182
Italy	MIBTel	04-08-1993	07-02-2003	2391
Netherlands	AEX General	28-10-1992	07-02-2003	2604
Russia	Moscow Times	15-05-1998	07-02-2003	1125
Spain	Madrid General	27-05-1999	07-02-2003	857
Sweden	Stockholm General	17-01-2001	07-02-2003	505
Switzerland	Swiss Market	27-11-1990	07-02-2003	3031
Turkey	ISE National 100	18-07-1997	07-02-2003	1321
U.K.	FTSE 100	18-04-1984	07-02-2003	4750
Africa and Middle East				
Israel	TA-100	25-05-1998	07-02-2003	931

A visual example of this is given in Figure 52, which shows the open level of the Japanese Nikkei 225 index over the past 17 years, and its corresponding ς_t transformation.

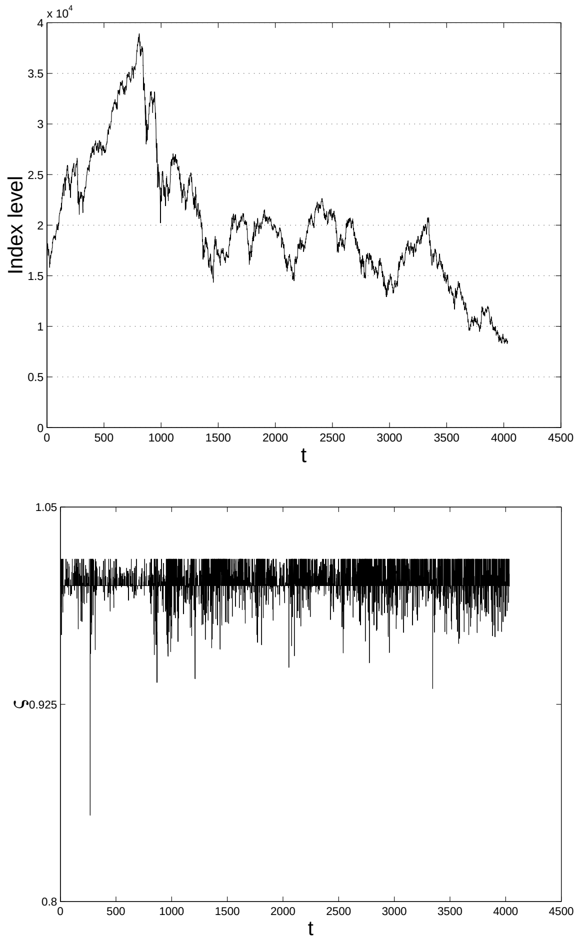


Figure 52: **Top**: The Nikkei 225 index (open level). **Bottom**: The ς_t transformation of the Nikkei 225, as described by Equations 59-62.

A completely adaptive topology is used in this section, allowing heterogeneous topologies to be maintained by the estimated Pareto set of NNs. The model inputs are defined as follows:

$$v_t^{1,\ldots,10} = \varsigma_{t-2}, \ldots, \varsigma_{t-11} \tag{63}$$

variables 1 to 10 contain the last 10 lagged realised values of y_t (2 weeks of trading). Again ς_{t-1}

cannot be used as it incorporates information that will not be available at the start of the day at $t-1$.

A sliding window is used to contain the training and test sets, which were generated by first creating the relevant explanatory vector and dependent value pairs (embedded matrix). The first 80% of the data was initially used for training, and the next 20% for testing. The bootstrap based MOENN model of the previous Sections was used as the training algorithm.

Each run was initialised with 200 random networks. The algorithm parameters were:

- Probability of weight perturbation = 0.2
- Probability of individual weight elimination = 0.02
- Probability of individual node elimination = 0.02
- Perturbation $\sim N(0, 0.1)$
- Initial weights $\sim N(0, 0.1)$
- Generations = 25,000

11.2.1 Results

Graphical examples of estimated Pareto fronts defined by the archive of ENNs on the test data are provided in Figures 53 - 55. Figure 53 gives the estimated Pareto front generated for a market from the Americas group (the S&P 500 index), Figure 54 gives the estimated Pareto front generated for a market from the Asia/Pacific group (the Nikkei 225 index), and Figure 55 gives the estimated Pareto front generated for a market from the European group (the FTSE 100 index).

These figures show that although there is some degree of noise present, the general shape and properties of ENN models fitted on the training data is consistent with their performance on the training data. ENNs with high return and high volatility on the training data also maintain these properties on testing data, and likewise for models with low volatility and low return.

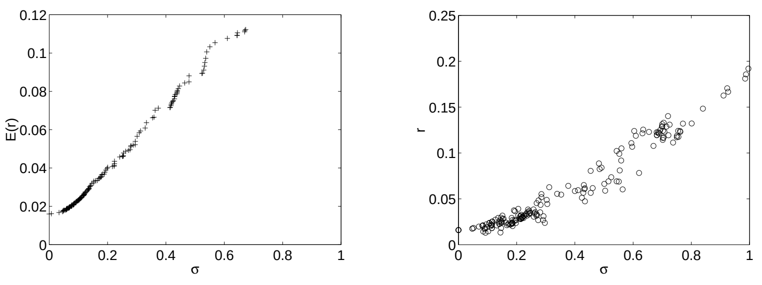


Figure 53: Risk and return on the S&P 500 index (80% train, 20% test). **Left:** Training Pareto front. **Right:** Evaluation of set on test data.

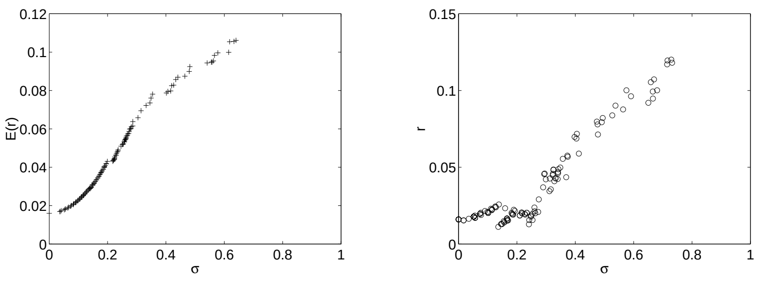


Figure 54: Risk and return on the Nikkei 225 index (80% train, 20% test). **Left:** Training Pareto front. **Right:** Evaluation of set on test data.

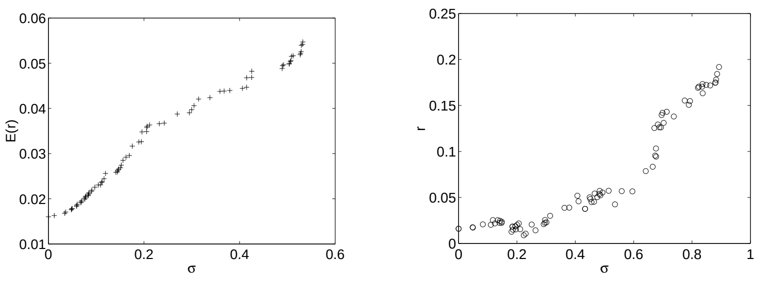


Figure 55: Risk and return on the FTSE 100 index (80% train, 20% test). **Left:** Training Pareto front. **Right:** Evaluation of set on test data.

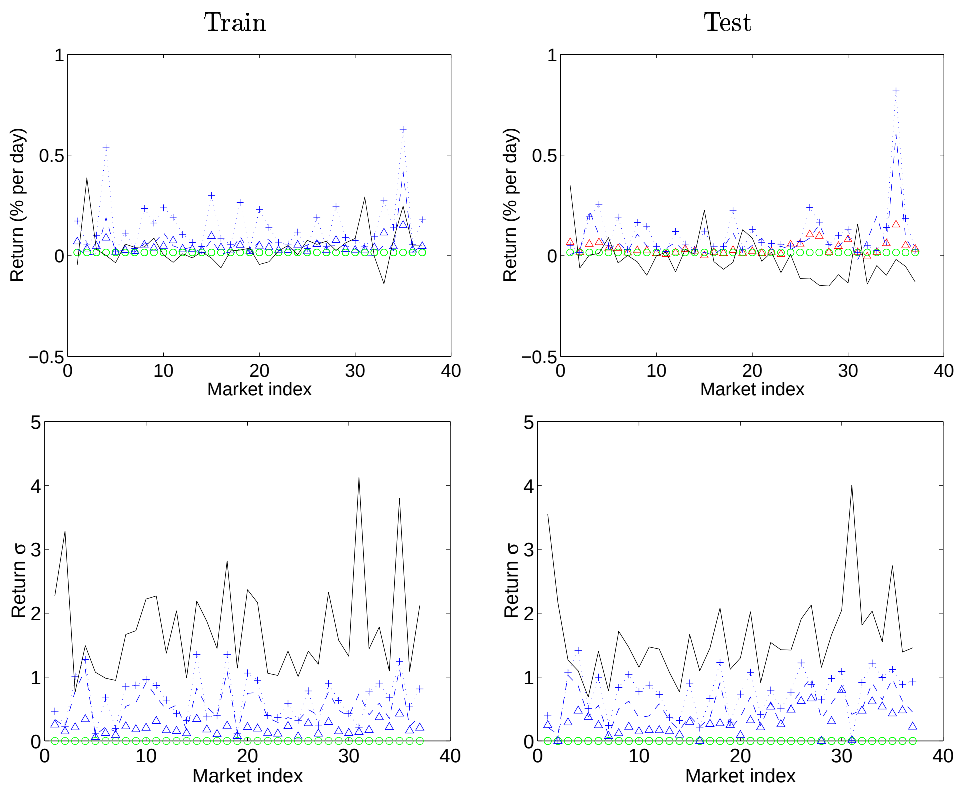


Figure 56: Risk and return for 3 different exemplar members of the archived ENNs across the 37 international indices (indices ordered as in Table 21). Market performance and performance of the random walk model using the same trading strategy are also shown.

Figure 56 illustrates the risk and return experienced by different operating points (individual ENNs) of the Pareto front of MOENNs, and their corresponding performance on the test data. Three investor types are again taken as exemplars from the archives, those that lie on either extreme of the front, and the mid set member, for each of the 37 different stock indices (points ‘A’, ‘B’ and ‘C’, as shown previously in Figure 49). The performance of the profit maximising extreme ENN, ‘A’, is denoted by plus signs in Figure 56, joined by dots. The performance of the mid set members, ‘B’, are denoted by triangles and the performance of the risk minimising ENNs, ‘C’, by circles. The market performance is shown with a solid line, and the random walk model performance is shown with a dashed line (in this case the random walk model uses the trading strategy based on its prediction that $\varsigma_{t+1} = \varsigma_{t-1})^3$. As can be seen the profit maximiser model consistently produces

[3]The random walk model of $\varsigma_{t+1} = \varsigma_t$ cannot be used as ς_t is unknown at the start of day t and can only be

higher rate of return than the other models across both training data and test data, meaning that the returned capital on investment in this model was consistently higher than the two other ENN operating points (implying the consistency of the ENN set) and higher than the returns produced by trading on random walk predictions, or investing the capital in the market at the start and removing it at the end of the time period. The market rate however experiences the highest volatility of return across the data, implying that it is the least safe of the five models for investment purposes. The relative performance of the five different models can be seen more clearly in Figure 57.

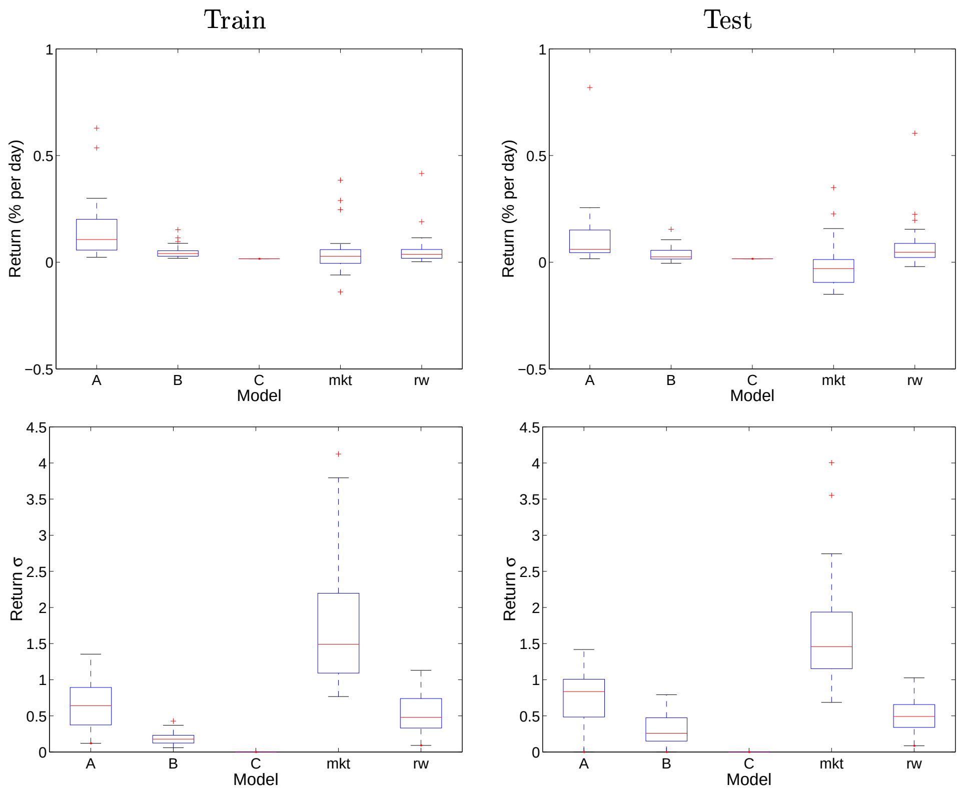


Figure 57: Boxplots of the realised risk and return for 3 different exemplar members of the archived ENNs across the 37 international indices. Market performance and performance of the random walk model using the same trading strategy are also shown.

Here boxplots of the results of the five models over training and test sets for the two objective measures are shown. Using the Wilcoxon signed ranks test (two tailed at the 0.05 significance level), the higher return rates of model ‘A’ are found to be significantly better than all the other models,

calculated at the close of trade on day t.

with the next best model being the random walk, followed by model 'B'. Over the training data the market return is higher than the MOENN operating point 'C', however on the test data, 'C' (effectively keeping the capital in the bank and not trading) is significantly better than investing in the markets (which isn't surprising given recent market performance).

The volatility of the market return was significantly higher than all the other models, followed by model 'A', the random walk model and then model 'B'. Firstly these results show the three different operating points of the MOENN results were consist across the test data, with 'A' having higher return followed by 'B' and then 'C', and 'C' having the lowest volatility, followed by 'B' and 'A'. Model 'A' was also the best performing model (in terms of return) out of the five, and produced consistent positive returns on all markets (as shown by the box-plots), even when some of the markets were experiencing significant downward trends. Mean values and standard deviations are shown in Table 22. As can be seen, when the extreme profit maximisation ENN model 'A' (the ENN which has the maximum return of the Pareto set of ENN models on the training data) is used, an average daily return of 0.1142% on the test data is realised. This relates to an average annual return of 33.2%, compared to the random walk annual return of 20.1% and the market annual return of -4.3%.

Table 22: Mean risk and return over the 37 international indices. Results shown for the archived ENN exemplar models '*A*', '*B*' and '*C*', the market performance and from the random walk model (standard deviations in parenthesis).

	Train		Test	
	Risk	% Ret	Risk	% Ret
Model *C* (archive ENN)	0.0000 (0.0000)	0.0160 (0.000)	0.0000 (0.0000)	0.0160 (0.000)
Model *B* (archive ENN)	0.1923 (0.0882)	0.0475 (0.0285)	0.2949 (0.2025)	0.0366 (0.0330)
Model *A* (archive ENN)	0.6570 (0.3526)	**0.1494** (0.1317)	0.7573 (0.3551)	**0.1142** (0.1375)
Market	**1.7533** (0.7917)	0.0416 (0.0941)	**1.6207** (0.6864)	-0.0176 (0.1061)
Random Walk	0.5142 (0.2782)	0.0544 (0.0711)	0.5137 (0.2634)	0.0733 (0.1037)

The fact that consistent profits are observed using the random walk approach is interesting. This may be due to the trading strategy itself restricting the random walk model to trade where the market is volatile and the potential for large returns is higher (which is its aim), it must be remembered that the random walk is operating on the transformed trading series not the raw indices. However it is conceivable that the transaction costs modelled may be too low for certain markets. If this second case is true, it should be noted that even with transactions cost raised in order to make the random walk model unprofitable, model 'A' will still experience significant positive returns beyond both the market and bank returns.

Figure 58 illustrates the wide range of ENN architectures apparent in the non-dominated set of models returned by the Pareto-ENN process. Figure 58a shows a Hinton plot of the weights of the 84 ENNs residing in the archive of the Nikkei 225 Pareto-ENN, each column representing a different ENN (each white square denoting an active weight, and each empty square denoting a disconnected weight). The weights are ordered such that $w_1, \dots, w11$ represent the weights from the 10 inputs nodes and the bias to the first hidden node, $w_{12}, \dots, w23$ represent the same for the second hidden node and so on. $w_{110}, \dots, w121$ represent the weights between the hidden layer and the output node, and the output bias. As it can be seen, a wide range of weights and degrees of connectivity are used, from 75 active weights (the 81st ENN) to 121 active weights (the 83rd ENN). Figure 58b shows which inputs were used by each of the ENNs (with some ENNs using all the available inputs, and others using as few as 6) and Figure 58c shows which hidden units when used by each of the ENN set members (with between 7 and 10 hidden units used).

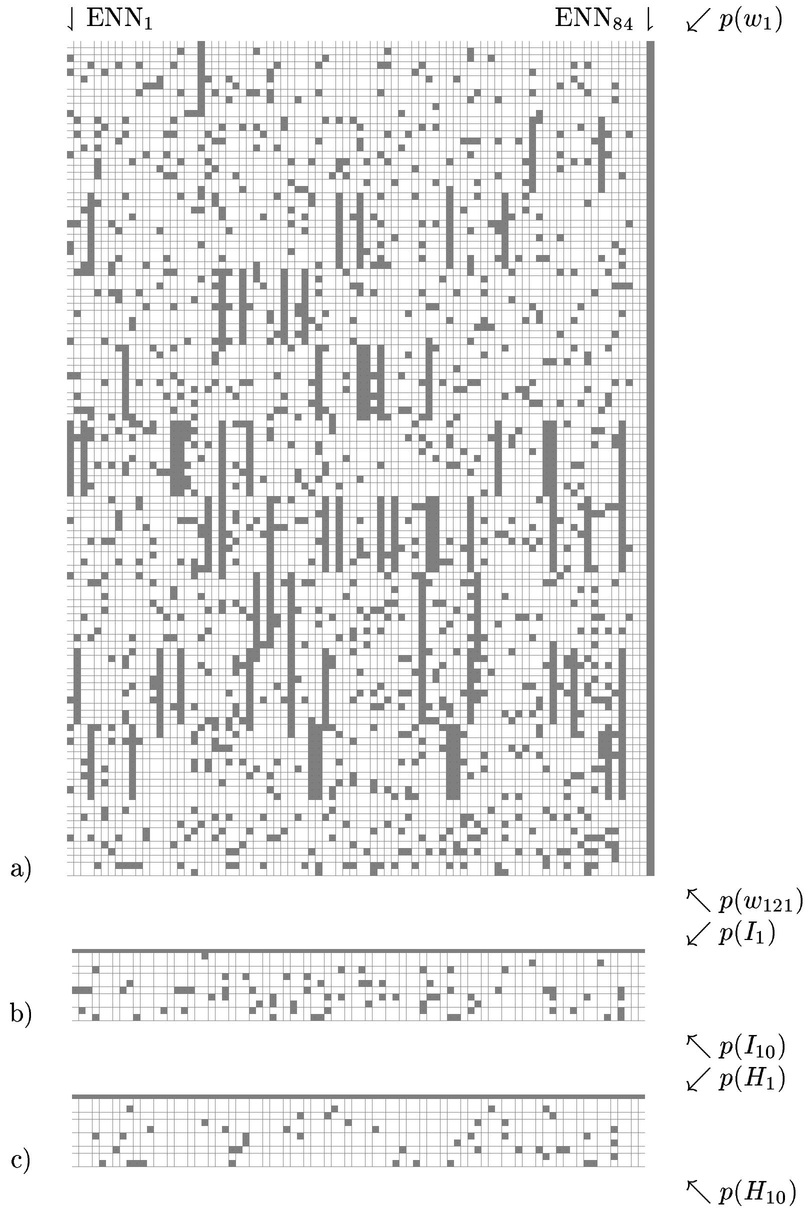


Figure 58: Example of the range of ENN topographies on an estimated Pareto front. Hinton plots are shown for the ENN weights (a), input topography (b) and hidden topography (c) of the 84 ENNs lying on the estimated Pareto error surface for the Nikkei 225 data.

11.2.2 Further efficiency tests: Comparison with single objective ES runs

As described earlier, another potential solution to this problem is to run a single objective optimiser many times on a problem keeping the other objective(s) 'fixed' (i.e. constraining acceptable solutions to have the other objective(s) equal to or below/above a pre-defined value). It has been argued previously that this is inefficient in time resources, however this statement will be empirically investigated here. To facilitate this a uni- objective optimiser for the previous finance problem was formulated. Here the optimiser (again a (1+1)-ES) is solely concerned with optimising return, for a maximum acceptable risk level. The perturbation techniques and probabilities are identical to those used in the MOENN previously, and again the bootstrap generalisation method is used to prevent over-fitting. For each of the 37 finance problems five single objective ENNs were trained, with the constraint of risk being no higher than 0.2, 0.4, 0.6, 0.8 and 1.0, for a maximum of 25000 generations (meaning 185 separate optimiser runs). The optimiser is described in Algorithm 14.

Algorithm 14 Implementation of the uni-objective ES NN optimiser (bootstrap training approach), for the finance problem.

```
Input:   $M$, size of initial random population of solutions. Each solution chromosome $\mathbf{x}$
         representing the weights and topology of a NN model.
         $n$, the number of bootstrap subsets generated from the original training sets.
         $s$, the size of the bootstrap subsets.
         $r$, the maximum risk allowed by a model.
Output:  A single NN which estimates the maximum return possible given $r$.
1:       Initialisation: Generate $n$ bootstrap subsets of the training data of size $s$.
         Generate random NN population of size $M$, such that each parameter
         (weight) of the ENNs $\sim N(0,\alpha)$, and the binary part of the chromosome is
         either initialised at 1 or $\sim U(0,1)$. Generate the empty elite individual
         $E_0 = \emptyset$. Update $E_0$ with the fittest solution from the random population,
         with respect to the chosen error term (using a solution's worst term over the
         $n$ subsets), constrained that the highest risk on the subsets is lower than $r$.
         Initialise generation counter $t := 0$.
2:       $X_t^* = E_t$.
3:       Genetic Recombination: As in $\mathcal{M}_B$.
4:       Fitness Assignment: Evaluate the ENN(s) $X_t^*$ with respect to return
         on the training subsets presented. If the return is greater than that of $E_t$,
         and risk is less than $r$, go to 5, otherwise go to 6.
5:       $E_{t+1} := X_t^*$
6:       Looping: Iterate epoch count, $t := t + 1$. If stopping criteria have not been met
         then go to 2, else terminate algorithm and save $E_t$ for evaluation on test data.
7:       end
```

Each run was initialised with 200 random networks. The algorithm parameters were:

- Probability of weight perturbation = 0.2
- Probability of individual weight elimination = 0.02
- Probability of individual node elimination = 0.02
- Perturbation $\sim N(0, 0.1)$
- Initial weights $\sim N(0, 0.1)$
- Generations = 25,000

The progress of the elite ENN in objective space, E_t in Algorithm 14, was recorded every 250 generations and compared to the set of ENNs discovered by the MOENN after 25000 generations, the generation at which the single objective ENN is no longer dominated by member(s) of the saved front F being marked. For each test problem, if the average number of generations needed for the single objective optimiser to train an ENN that is not *behind* F is less than $25000/|F|$ then there is no efficiency benefit to using the MOENN training regime. If however the average number of generations needed is greater than $25000/|F|$ there are tangible efficiency gains, to which a value can even be assigned.

Results

Results from the different runs are shown in Table 23

On all bar two of the 37 test problems the MOES is shown to be more efficient than the uni-objective optimiser - performing 22 times better on average (i.e. for the uni-objective optimiser to find the same points in $|F|$ it would need to perform 22 times more function evaluations). This indeed may even be an underestimate - as nearly 40% of the 185 uni-objective optimiser runs did not reach the front found by the MOES within 25000 generations (an example of this is shown in Figure 59).

The implication of this is that the formulation of nominally uni-objective problem as a multi-objective problem can actually improve the search process - as indeed has been previously postulated (e.g. Abbass & Deb (2003)). This is most likely due to the synergies present in multi-objective search. The evolution of a point in one area of objective space, through its decision space parameters, may lead to its shifting to a different area of objective space. In multi-objective optimisation this solution

Table 23: Results comparing uni-objective and multi-objective optimiser. G^U is the average number of generations taken by the uni-objective to reach the estimated Pareto front found by the MOES. G^M is the number of generations per non-dominated point on the estimated Pareto front found by the MOES. The ratio of the two values, G^U/G^M, gives an approximation as to how much more efficient the MOES is at finding estimated Pareto solutions from a given set than the uni-objective ES.

Financial Index	G^U	G^M	G^U/G^M
MerVol	10650	892.9	11.9
Bovespa	1250	781.3	1.6
S&P TSX Composite	4200	675.7	6.2
IPC	16850	263.2	64.0
Lima General	25000	1388.9	18.0
S&P 500	1900	342.5	5.5
All Ordinaries	25000	925.9	27.0
Shanghai Composite	11700	384.6	30.4
Hang Seng	3200	471.7	6.7
BSE 30	15950	396.8	40.2
Jakarta Composite	14350	609.8	23.5
Nikkei 225	1950	625.0	3.1
KLSE Composite	2950	581.3	5.0
NZSE 40	500	925.9	0.5
Karachi 100	13300	431.0	30.8
PSE Composite	7950	568.1	14.0
Straits Times	24350	1086.9	22.4
Seoul Composite	1150	595.2	1.9
All Share	25000	1388.9	18.0
SET	17100	500.0	34.2
Taiwan Weighted	10950	675.7	16.2
ATX	13600	757.6	18.0
BEL-20	25000	581.4	43.0
PX50	20100	1086.9	18.5
KFX	25000	806.5	31.0
CAC 40	16750	294.1	57.0
DAX	25000	531.9	47.0
General Share	10600	609.8	17.4
MIBTel	18050	657.9	27.4
AEX General	15000	333.4	45.0
Moscow Times	750	1562.5	0.5
Madrid General	11150	735.3	15.2
Stockholm General	7150	609.8	11.7
Swiss Market	20550	362.3	36.7
ISE National 100	9700	347.2	27.9
FTSE 100	2350	925.9	2.5
TA-100	11700	454.5	25.7

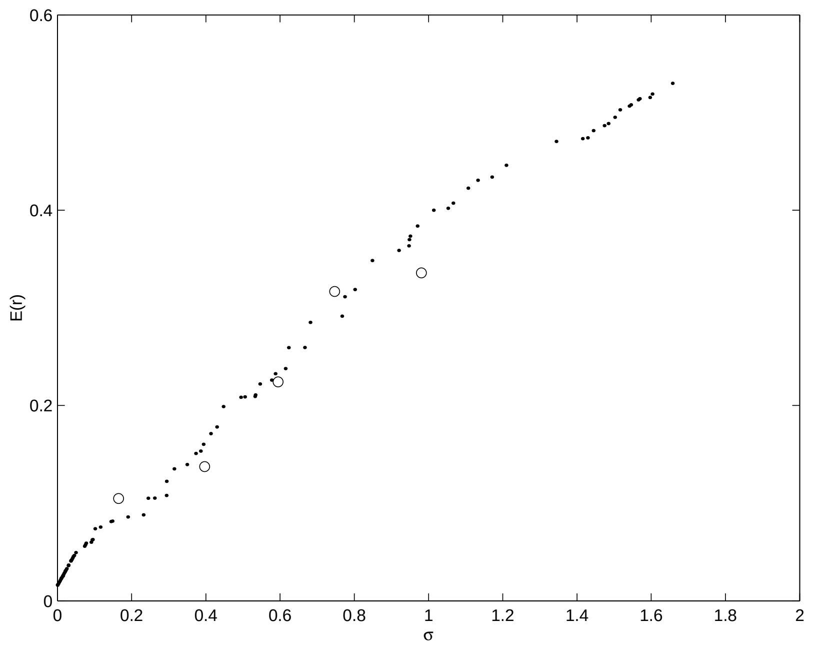


Figure 59: Risk and return on the Lima General index training data, the models found by the MOES after 25000 generations plotted as points, the models found by the uni-objective ES with 5 different risk maximums plotted as circles, again after 25000 generations.

is stored if it is non-dominating with other solutions in the area it has moved to. In the case of uni-objective optimisation these kind or fortuitous movements are not easily sustained as a movement in decision space is purely defined as better or worse - therefore the uni-objective formulation may be more likely to be caught in local optima.

11.3 General comments

The MOENN framework has been shown to be a robust approach to multi-objective training in noisy domains, with experimentation on 38 real-world data sets showing that a Pareto set of ENNs training using the methodology can perform in a consistent manner on unseen test data. This has been manifest in both the visual inspection of the trade-off fronts produced, and by statistical comparison of different operating points on training data, and their relative position on test data. The technique is also shown to generate significant results when compared to other models in the financial domain, and to produce significant returns even when considering transaction costs. New methods to improve generalisation in MOENNs have also been introduced and compared on test sets from the multi-objective NN literature, with an approach based on the bootstrapping of training data found to be significantly better than the comparative models.

The advances of this body of work as a whole will now be discussed in the concluding chapter of the thesis.

Chapter 12

Conclusion

This thesis provides a synthesis of a number of methods, from a number of disparate application domains, in the construction of a novel methodology for implementing multi-objective optimisation within the NN domain. In addition a number of significant new techniques have been developed to improve the general area of multi-objective optimisation.

In Part I:

- The theoretical development of two novel data structures (called dominated and non-dominated trees) was presented for the storing of multi-dimensional points.
- Details of the operation, including computational complexity of these data structures was also derived.
- These new structures are shown to perform faster than linear lists in the active archiving of *non-dominating* sets for their use by multi-objective optimisation methods,
- In addition their use in the storing of general sets of multi-dimensional points is also highlighted.

In Part II:

- The generation of a new multi-objective variant of the recent particle swarm optimisation algorithm based on the properties of the new data structures is presented.

- It was demonstrated that this new algorithm more closely transfers the particle swarm heuristic to the multi-objective domain than its peers which have been developed by other researchers during the course of this thesis.
- It was also empirically demonstrated to be superior to the best performing of these competing multi-objective particle swarm optimisers on a number of well known test problems.
- A general overview of the PSO and MOPSO algorithm has been provided, indicating the strengths and weakness of the method and placing it in the context of other EC techniques.

In Part III:

- The formulation of a general MOENN architecture is provided, where the ENNs are adaptive in both parameters and topologies.
- Novel generalisation techniques are derived from this general framework and a new method based on bootstrap sampling of training data is shown to produce significantly better and consistent results on test data.
- Results are provided on extensive real world financial forecasting problems, highlighting the benefit of multi-objective ENN training.

12.1 Further issues

12.1.1 Bias/variance tradeoff in uni-objective problems

As discussed previously, a common problem in the uni-objective function modelling domain is the calculation of appropriate network size. A number of different methods are used in the literature on this matter which tend to consist consist of iteratively adding and removing nodes (OBS, OBD) or penalising complex representation (i.e. weight decay regularisation). However, using the methods introduced in this part, it is possible to estimate the optimum network complexity for a uni-objective modelling task during the learning process and in a single run. Abbass (2001) has previously taken this approach, trading off ENN accuracy and ENN size, however the method can be extended further by encapsulating the complexity of a ENN in terms of the sum of its squared weights - making the problem continuous in both dimensions as opposed to continuous in one and discrete in the other.

12.1.2 Multi-objective simulated annealing and Markov chain Monte Carlo

Recent work has attempted to create simulated annealing multi-objective optimisation models (Czyzak & Jaskiewicz, 1998; Hapke *et al.* , 2000; Jaskiewcz, 2001; Matos & Melo, 1999; Nam & Park, 2000), although these generally have been of the form of linear sum methods. The work presented in this thesis is equally applicable to these methods, both in terms of the use of the data structures and the general ENN training procedures. A rigorous extension of simulated annealing process to the multi-objective domain would however open up a whole new area of multi-objective optimisation, as it would open the door for the development of Markov Chain Monte Carlo (Denison *et al.* , 2002; Liu, 2001) multi-objective methods.

References

Abbass, H.A. 2001. A Memetic Pareto Evolutionary Approach to Artificial Neural Networks. *Pages 1–12 of: The australian joint conference on artificial intelligence*. Springer.

Abbass, H.A., & Deb, K. 2003. Searching under multi-evolutionary pressures. *Pages 391–404 of: Proceedings of the 2003 evolutionary multiobjective optimization conference (emo03)*. Springer-Verlag.

Adya, M., & Collopy, F. 1998. How Effective are Neural Networks at Forecasting and Prediction? A Review and Evalution. *International Journal of Forecasting*, **17**, 481–495.

Alba, E., Aldana, J.F., & Troyla, J.M. 1993. Full Automatic ANN Design: A Genetic Approach. *Lecture Notes in Computer Science*, **686**, 399–404.

Angeline, P., Saunders, G., & Pollack, J. 1994. An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, **5**(1), 54–65.

Armstrong, J.S., & Collopy, F. 1992. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, **8**(1), 69–80.

Atiya, A.F., El Shoura, S.M., Shaheen, S.I., & El Sherif, M.S. 1999. A Comparison Between Neural-Network Forecasting Techniques–Case Study: River Flow Forecasting. *IEEE Transactions on Neural Networks*, **10**(2), 402–409.

Baluja, S. 1996. Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller. *IEEE Transactions on Systems Man and Cybernetics - Part B: Cybernetics*, **26**(3), 450–463.

Beale, G.O., & Cook, G. 1978. Optimal Digital Simulation of Aircraft via Random Search Techniques. *AIAA Journal of Guidance and Control*, **1**(4), 237–241.

Belfore, L., & Arkadan, A. 1997. Modeling Faulted Switched Reluctance Motors Using Evolutionary Neural Networks. *IEEE Transactions on Industrial Electronics*, **44**(2), 226–233.

Bentley, J.L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**(9), 507–517.

Bentley, J.L., & Friedman, J.H. 1979. Data Structures for Range Searching. *Computing Surveys*, **111**(4), 398–409.

Bera, T., & Higgins, L. 1993. ARCH Models: Properties, Estimation and Testing. *Journal of Economic Surveys*, **7**(4), 305–362.

Berlanga, A., Molina, J.M., Sanchis, A., & Isasi, P. 1999. Applying Evolutionary Strategies to Neural Networks Robot Controller. *Lecture Notes in Computer Science*, **1607**, 516–525.

Bishop, C.M. 1998. *Neural Networks for Pattern Recognition*. Oxford University Press.

Borges, P.C., & Hansen, M.P. 1998. *A basis for future successes in multiobjective combinatorial optimization*. Tech. rept. Technical University of Denmark. IMM-REP-1998-8.

Brealey, R.A., & Myers, S.C. 1996. *Principles of Corporate Finance*. 5th edn. McGraw-Hill.

Brill, F., Brown, D., & Martin, W. 1992. Fast Genetic Selection of Features for Neural Network Classifiers. *IEEE Transactions on Neural Networks*, **3**(2), 324–328.

Carlisle, A., & Dozier, G. 2001 (April). *Tracking Changing Extrema with Particle Swarm Optimizer*. Tech. rept. CSSE01-08. Auburn University, Computer Science and Software Engineering Department.

Chen, Q., & Weigand, W.A. 1992. Neural Net Model of Batch Process Optimization Based on An Extended Genetic Algorithm. *Pages 519–524 of: IJCNN'92 IEEE/INNS Baltimore*, vol. IV.

Chen, Y., & O'Connell, R. 1997. Active Power Line Conditioner with a Neural Network Control. *IEEE Transactions on Industrial Applications*, **33**(4), 1131–1136.

Coello, C.A.C. 1999. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, **1**(3), 269–308.

Coello, C.A.C., & Lechunga, M.S. 2002. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. *Pages 1051–1056 of: Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Coit, D.W., & Smith, A.E. 1996. Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. *Computers and operations research*, **23**(6), 515–526.

Conradie, A.V.E., Miikkulainen, R., & Aldrich, C. 2002a. Adaptive Control utilising Neural Swarming. *Pages 60–67 of: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA.*

Conradie, A.V.E., Miikkulainen, R., & Aldrich, C. 2002b. Intelligent Process Control utilising Symbiotic Memetic Neuro-Evolution. *Pages 623–628 of: Proceedings of the 2002 IEEE Congress on Evoluionary Computation (CEC-2002).*

Czyzak, P., & Jaskiewicz, A. 1998. Pareto simulated annealing - a metaheuristic technique for multi-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, **7**, 34–47.

de Berg, M., van Kreveld, M., Overmars, M., & Schwarzkopf, O. 1997. *Computational geometry : algorithms and applications.* Berlin: Springer.

de Garis, H. 1991. GenNETS : Genetically Programmed Neural Networks. *Pages 1391–1396 of: Proceedings of IJCNN'91 Singapore IEEE/INNS*, vol. 2.

Deb, K. 1999. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary computation*, **7**(3), 205–230.

Deb, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms.* Wiley.

Deb, K., & Jain, S. 2002. *Running Performance Metrics for Evolutionary Multi-objective Optimization.* Tech. rept. 2002004. Indian Insitute of Technology Kanpur.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. *Pages 849–858 of: Proceedings of Parallel Problem Solving from Nature - PPSN vi.* Springer.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. 2001 (August). *Scalable Test Problems for Evolutionary Multi-Objective Optimization.* Tech. rept. 2001001. Kanpur Genetic Algorithms Laboratory (KanGAL).

Denison, D.G.T, Holmes, C.C., Mallick, B.K., & Smith, A.F.M. 2002. *Bayesian Methods for Nonlinear Classification and Regression.* Wiley.

Dominic, S., Das, R., Whitley, D., & Anderson, C. 1992. Genetic Reinforcement Learning for Neural Networks. *Pages 71–76 of: Proceedings of IJCNN'92 Seattle IEEE/INNS*, vol. II.

Dracopoulos, D.C., & Jones, A.J. 1995. Neural Networks and Genetic Algorithms for the Attitude Control Problem. *Lecture Notes in Computer Science*, **930**, 315–321.

Duch, W. 1999. Alternatives to gradient-based neural training. *Pages 59–64 of: Fourth Conference on Neural Networks and Their Applications, Zakopane, Poland.*

Duch, W., & Korczak, J. 1999. *Optimization and global minimization methods suitable for neural networks.* Tech. rept. KMK UMK 1/99. Nicolaus Copernicus University, Faculty of Physics, Astronomy and Informatics.

Ehrgott, M., & Gandibleux, X. 2000. An Annotated Bibliography of Multi-objective Combinatorial Optimization. *OR Spektrum*, **22**(4), 425–460.

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

Everson, R.M., Fieldsend, J.E., & Singh, S. 2002. Full Elite Sets for Multi-Objective Optimisation. *Pages 343–354 of:* Parmee, I.C. (ed), *Adaptive Computing in Design and Manufacture V.* Springer.

Fang, J., & Xi, Y. 1997. Neural Network design based on evolutionary programming. *Artificial Intelligence in Engineering*, **11**, 155–161.

Fieldsend, J.E. 1999. *Non-linear ARCH Volatility Estimation Using Neural Networks.* MSc Thesis, University of Plymouth.

Fieldsend, J.E., & Singh, S. 2002a. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. *Pages 37–44 of: Proceedings of UK Workshop on Computational Intelligence (UKCI'02).*

Fieldsend, J.E., & Singh, S. 2002b. Pareto Multi-Objective Non-Linear Regression Modelling to Aid CAPM Analogous Forecasting. *Pages 388–393 of: Proceedings of the 2002 IEEE International Joint Conference on Neural Networks, part of the 2002 IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Fieldsend, J.E., Everson, R.M., & Singh, S. 2003. Using Unconstrained Elite Archives for Multi-Objective Optimisation. *IEEE Transactions on Evolutionary Computation*, **7**(3), 305–323.

Fogel, D.B., Wasson, E.C., & Boughton, E.M. 1995. Evolving neural networks for detecting breast cancer. *Cancer Letters*, **96**, 49–53.

Fogel, D.B., Wasson, E.C., Boughton, E.M., & Porto, V.W. 1997. A step toward computer-assisted mammography using evolutionary programming and neural networks. *Cancer Letters*, 93–97.

Fonseca, C.M., & Fleming, P.J. 1993. Genetic Algorithms for Multiobjective Optimizationi: Formulation, Discussion and Generalization. *Pages 416–423 of: Proceedings of the Fifth International Conference on Genetic Algorithms.* San Mateo, California: Morgan Kaufmann.

Fonseca, C.M., & Fleming, P.J. 1995. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, **3**(1), 1–16.

Greenwood, G.W. 1997. Training Partially Recurrent Neural Networks Using Evolutionary Strategies. *IEEE Transactions on Speech and Audio Processing*, **5**(2), 192–194.

Greenwoood, G.W. 1997. Training Multiple-Layer Perceptrons to Recognise Attractors. *IEEE Transactions on Evolutionary Computation*, **1**(4), 244–248.

Gujarati, D. 1992. *Essentials of Econometrics.* McGraw-Hill.

Hajela, P., & Lin, C-Y. 1992. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, **4**, 99–107.

Hann, T., & Steurer, E. 1996. Much ado about nothing? Exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data. *Neurocomputing*, **10**(4), 323–340.

Hanne, T. 1999. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, **117**, 553–564.

Hanne, T. 2000. Global Multiobjective Optimization Using Evolutionary Algorithms. *Journal of Heuristics*, **6**, 347–360.

Hansen, J.V., & Meservy, R.D. 1996. Learning experiments with genetic optimization of a generalized regression neural network. *Decision Support Systems*, **18**(3), 317–325.

Hansen, M.P., & Jaszkiwicz, A. 1998. *Evaluating the quality of approximations to the non-dominated set.* Tech. rept. IMM-REP-1998-7. Institute of Mathematical Modelling, Technical University of Denmark.

Hapke, M., Jaskiewicz, A., & Slowinski, R. 2000. Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. *Journal of Heuristics*, **6**(3), 329–345.

Haykin, S. 1999. *Neural Networks A Comprehensive Foundation.* 2 edn. Prentice Hall.

Horn, J., Nafpliotis, N., & Goldberg, D.E. 1994. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. *Pages 82–87 of: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1. Piscataway, New Jersey: IEEE Service Center.

Hu, X., & Eberhart, R. 2002. Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. *In: Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Huang, S., & Huang, C. 1997. Application of Genetic-Based Neural Networks to Thermal Unit Commitment. *IEEE Transactions on Power Systems*, **12**(3), 654–660.

Hung, S., & Adeli, H. 1994. A Parallel Genetic/Neural Network Learning Algorithm for MIMD Shared Memory Machines. *IEEE Transactions on Neural Networks*, **5**(6), 900–909.

Ishigami, H., Fukuda, T., Shibata, T., & Fumihito, A. 1995. Structural optimization of fuzzy neural network by genetic algorithm. *Fuzzy Sets and Systems*, **71**(3), 257–264.

Ismail, A., & Engelbrecht, A. 1999. Training Product Units in Feedforward Neural Networks using Particle Swarm Optimization. *Pages 36–40 of: Proceedings of the International Conference on Articial Intelligence, Durban, South Africa.*

Ismail, A., & Engelbrecht, A. 2000. Global optimization algorithms for training product unit neural networks. *Pages 132–137 of: International Joint Conference on Neural Networks IJCNN'2000*, vol. I. IEEE Computer Society, LosAlamitos, CA.

Janson, D.J., & Frenzel, J.F. 1992. Application of Genetic Algorithms to the training of Higher Order Neural Networks. *Journal of Systems Engineering*, **2**(4), 272–276.

Janson, D.J., & Frenzel, J.F. 1993. Training Product Unit Neural Networks with Genetic Algorithms. *IEEE Expert*, **8**(5), 26–33.

Jaskiewcz, A. 2001. *Multiple objective metaheuristic algorithms for combinatorial optimization.* Ph.D. thesis, Poznan University of Technology.

Jones, F. 1993. *Lebesgue Integration on Euclidean Space.* Boston: Jones and Bartlett.

Kaastra, I., & Boyd, M. 1996. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, **10**(3), 215–236.

Kennedy, J., & Eberhart, R. 1995. Particle Swarm Optimization. *Pages 1942–1948 of: Proceedings of the Fourth IEEE International Conference on Neural Networks.* Perth, Australia: IEEE Service Center.

Kennedy, J., & Spears, W. 1998. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. *Pages 74–77 of: Proceedings 1998 IEEE World Congress on Computational Intelligence, Anchorage: Alaska, May 1998.*

Khare, V., Yao, X., & Deb, K. 2002. *Performance Scaling of Multi-Objective Evolutionary Algorithms.* Tech. rept. 2002009. Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology Kanpur.

Knowles, J., & Corne, D. 1999. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. *Pages 98–105 of: Proceedings of the 1999 Congress on Evolutionary Computation.* Piscataway, NJ: IEEE Service Center.

Knowles, J.D., & Corne, D. 2000. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, **8**(2), 149–172.

Koza, J.R., & Rice, J.P. 1992. Genetic generation of both the weights and architecture for a neural network. *Pages 397–404 of: Proceedings of IJCNN'92 Seattle IEEE/INNS*, vol. II.

Kupinski, M.A., & Anastasio, M.A. 1999. Multiobjective Genetic Optimization of Diagnostic Classifiers with Implications for Generating Receiver Operating Characterisitic Curves. *Ieee Transactions on Medical Imaging*, **18**(8), 675–685.

Lajbcygier, P., Boek, C., Palaniswami, M., & Flitman, A. 1995. *Neural Network Pricing of all Ordinaries SPI Options on Futures.* Pages 64–77.

Laumanns, M., Zitzler, E., & Thiele, L. 2000. A Unified Model for Multi-Objective Evolutionary Algorithms with Elitism. *Pages 46–53 of: Proceedings of the 2000 Congress on Evolutionary Computation.* Piscataway, NJ: IEEE Service Center.

Laumanns, M., Thiele, L., Deb, K., & Zitzler, E. 2001 (June). *On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms.* Tech. rept. TIK-Report 108. Swiss Federal Institute of Technology Zurich (ETH).

LeBaron, B., & Weigend, A. 1998. A bootstrap Evaluation of the Effect of Data Splitting on Financial Time Series. *IEEE Transactions on Neural Networks*, **9**(1), 213–220.

Lee, S-K., & Jang, D. 1996. Translation, rotation and scale invariant pattern recognition using spectral analysis and hybrid genetic-neural-fuzzy networks. *Computers and Industrial Engineering*, **30**(3), 511–522.

Liu, J.S. 2001. *Monte Carlo Strategies in Scientific Computing.* Springer.

Liu, Y., & Yao, X. 1998. Towards designing neural network ensembles by evolution. *Lecture Notes in Computer Science*, **1498**, 623–632.

Lo, J.T. 2002. Minimization through Convexitization in Training Neural Networks. *Pages 1889–1894 of: Proceedings of the IEEE International Joint Conference on Neural Networks, part of the IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Lo, J.T., & Bassu, D. 2002a. Robust Approximation of Uncertain Functions where Adaptation is Impossible. *Pages 1956–1961 of: Proceedings of the IEEE International Joint Conference on Neural Networks, part of the IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Lo, J.T., & Bassu, D. 2002b. Robust Identification of Uncertain Dynamical Systems where Adaptation is Impossible. *Pages 1558–1563 of: Proceedings of the IEEE International Joint Conference on Neural Networks, part of the IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Lopez, P., Vilarino, D.L., & Cabello, D. 1999. Genetic Algorithm Based Training for Multilayer Discrete-Time Cellular Neural Networks. *Lecture Notes in Computer Science*, **1607**, 467–476.

Lovbjerg, M., Rasmussen, T.K., & Krink, T. 2001. Hybrid Particle Swarm Optimiser with Breeding and Subpopulations. *Pages 469–476 of:* Spector, L., Goodman, E.D., Wu, A., Langdon, W.B., Voigt, H-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., & Burke, E. (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001).* San Francisco, California, USA: Morgan Kaufmann.

Maniezzo, V. 1994. Genetic Evolution of the Topology and Weight Distribution of Neural Networks. *IEEE Transactions on Neural Networks*, **5**(1), 39–53.

Maricic, B. 1991. Genetically Programmed Neural Network for Solving Pole-Balancing Problem. *Pages 1273–1277 of: Artificial Neural Networks: Proceedings of the International Conference on Neural Networks.* North-Holland.

Marin, F.J., & Sandoval, F. 1993. Genetic Synthesis of Discrete-Time Recurrent Neural Network. *Lecture Notes in Computer Science*, **686**, 179–184.

Marti, L. 1992. Genetically Generated Neural Networks I: Representational Effects. *Pages 537–542 of: Proceedings of IJCNN'92 Baltimore IEEE/INNS*, vol. IV.

Matos, M.A., & Melo, P. 1999. Multiobjective Reconfiguration for Loss Reduction and Service Restoration Using Simulated Annealing. *In: Proceedings of the IEEE Budapest Power Tech'99 Conference.* I EEE.

McCreight, E.M. 1980. *Efficient algorithms for enumerating intersecting intervals and rectangles.* Tech. rept. CSL-80-9. Xerox Palo Alto Research Center, Palo Alto, CA, USA.

McCreight, E.M. 1985. Priority search trees. *SIAM Journal on Computing*, **14**, 257–276.

McDonnell, J., & Waagen, D. 1994. Evolving Recurrent Perceptrons for Time-Series Modeling. *IEEE Transactions on Neural Networks*, **5**(1), 24–38.

Merelo, J.J., Paton, M., Canas, A., Prieto, A., & Moran, F. 1993. Optimization of a competitive learning neural network by Genetic Algorithms. *Lecture Notes In Computer Science*, **686**, 185–192.

Moody, J. 1998. Forecasting the Economy with Neural Nets: A survey of Challenges and Solutions. *Pages 347–371 of:* Orr, G.B., & Mueller, K-R (eds), *Neural Networks: Tricks of the Trade.* Berlin: Springer.

Mostaghim, S., Teich, J., & Tyagi, A. 2001 (December). *Comparison of Data Structures for Storing Pareto Sets in MOEA.* Tech. rept. 02/01. Computer Engineering Lab., University of Paderborn.

Mostaghim, S., Teich, J., & Tyagi, A. 2002. Comparison of Data Structures for Storing Pareto-sets in MOEAs. *In: Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence.* Hawaii, May 12-17: IEEE Press.

Moya, M.M., & Hush, D.R. 1996. Network Constraints and Multi-ojective Optimization for One-class Classification. *Neural Networks*, **9**(3), 463–474.

Murata, T., & Ishibuchi, H. 1995. MOGA: Multi-objective genetic algorithms. *Pages 289–294 of: Proceedings of the 1995 IEEE International Conference on Evolutionary Computation.* Perth, November: IEEE Press.

Nam, D., & Park, C.H. 2000. Multiobjective Simulated Annealing - A Comparative Study to Evolutionary Algorithms. *International Journal of Fuzzy Systems*, **2**(2), 87–97.

Nelson, M., Hill, T., Remus, W., & O'Connor, M. 1999. Time Series Forecasting Using Neural Networks: Should the Data be Deseaonalized First? *Journal of Forecasting*, **18**, 359–367.

Olmez, T. 1997. Classification of ECG waveforms using RCE neural network and genetic algorithms. *Electronics Letters*, **33**(18), 1561–1562.

O'Neil, A. 1992. Genetic Based Training of Two-Layer, Optoelectronic Neural Network. *Electronics Letters*, **28**(1), 47–48.

Ozcan, E., & Mohan, C.K. 1999. Particle Swarm Optimization: Surfing the Waves. *Pages 1939–1944 of:* Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., & Zalzala, A. (eds), *Proceedings of the Congress on Evolutionary Computation*, vol. 3. Mayflower Hotel, Washington D.C., USA: IEEE Press.

Pareto, V. 1927. *Manuel D'Économie Politique*. 2nd edn. Paris: Marcel Giard.

Park, S ., Park, L-J., & Park, C. 1995. A Neuro-Genetic Controller for Nonminimum Phase Systems. *IEEE Transactions on Neural Networks*, **6**(5), 1297–1300.

Parks, G. T., & Miller, I. 1998. Selective Breeding in a Multiobjective Genetic Algorithm. *Lecture Notes in Computer Science*, **1498**, 250–259.

Parsopoulos, K., & Vrahatis, M. 2001. Particle Swarm Optimizer in Noisy and Continuously Changing Environments. *Pages 289–294 of:* Hamza, M.H. (ed), *Artificial Intelligence and Soft Computing*. IASTED/ACTA press.

Parsopoulos, K.E., & Vrahatis, M.N. 2002. Particle Swarm Optimization Method in Multiobjective Problems. *Pages 603–607 of: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002).*

Parsopoulos, K.E., Laskari, E.C., & Vrahatis, M.N. 2001a. Solving L1 Norm Errors-In-Variables Problems. *Pages 185–190 of:* Hamza, M.H. (ed), *Artificial Intelligence and Applications*. IASTED/ACTA Press.

Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., & Vrahatis, M.N. 2001b. Stretching technique for obtaining global minimizers through Particle Swarm Optimization. *Pages 22–29 of: Proceedings of the Particle Swarm Optimization Workshop, Indianapolis, USA.*

Parunak, H. Van Dyke, Brueckner, S., Sauter, J., & Matthews, R.S. 2001. Distinguishing Environmental and Agent Dynamics: A Case Study in Abstraction and Alternate Modeling Technologies. *Lecture Notes in Computer Science*, **1972**, 1–14.

Porto, V.W., Fogel, D.B., & Fogel, L.J. 1995. Alternative Neural Network Training Methods. *IEEE Expert*, **10**(3), 16–22.

Prados, D. 1992. New Learning Algorithm for Training Multilayered Neural Networks that uses Genetic-Algorithm Techniques. *Electronics Letters*, **28**(16), 1560–1561.

Ray, T., & Liew, K.M. 2002. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization*, **34**(2), 141–153.

Refenes, A-P.N., Burgess, A.N., & Bentz, Y. 1997. Neural Networks in Financial Engineering: A Study in Methodology. *IEEE Transactions on Neural Networks*, **8**(6), 1222–1267.

Saad, E.W., Prokhorov, D.V., & Wunsch, D.C. 1998. Comparitive Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks. *IEEE Transactions on Neural Networks*, **9**(6), 1456–1470.

Saravanan, N., & Fogel, D.B. 1998. Evolving Neural Control Systems. *IEEE Expert*, **10**(3), 23–27.

Schaffer, J., Caruana, R., & Eshelman, L. 1990. Using Genetic Search to Exploit the Emergent Behavior of Neural Networks. *Physica D*, **42**, 244–248.

Schaffer, J.D. 1985. Multiple objective optimization with vector evaluated genetic algorithms. *Pages 99–100 of: Proceedings of the First International Conference on Genetic Algorithms.*

Schittenkopf, C., Tino, P., & Dorffner, G. 2000. The profitability of trading volatility using real-valued and symbolic models. *Pages 8–11 of: IEEE/IAFE/INFORMS 2000 Conference on Compuational Intelligence for Financial Engineering (CIFEr).*

Shi, Y., & Eberhart, R. 1998. Parameter Selection in Particle Swarm Optimization. *Pages 591–601 of: Proceedings of the Seventh Annual Conference on Evolutionary Programming.*

Spofford, J., & Hintz, K. 1991. Evolving Sequential Machines in Amorphous Neural Networks. *Pages 973–978 of: Artificial Neural Networks: Proceedings of the International Conference on Neural Networks.* North-Holland.

Srinivas, M., & Patnaik, L.M. 1991. Learning Neural Network Weights using Genetic Algorithms - Improving Performance By Search-Space Reduction. *Pages 2331–2336 of: Proceedings of IJCNN'91 Singapore IEEE/INNS*, vol. 3.

Srinivas, N., & Deb, K. 1995. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, **2**(3), 221–248.

Sun, M., & Steuer, R.E. 1996. InterQuadi: An ineteractive quad tree based procedure for solving the discrete alternative multiple criteria problem. *European Journal of Operational Research*, 462–476.

Tan, K.C., Lee, T.H., & Khor, E.F. 2002. Evolutionary Algorithms for Multiobjective Optimization: Performance Assessments and Comparisons. *Artificial Intelligence Review*, 253–260.

Topchy, A.P., Lebedko, O.A., Miagkikh, V.V., & Kasabov, N.K. 1997. An Approach to Radial Basis Function Networks Training based on Cooperative Evolution and Evolutionary Programming. *In: Proceedings of the International Conference on Neural Information Processing, ICONIP'97.*

van den Bergh, F. 1999. Particle Swarm Weight Initialization in Multi-layer Perceptron Artificial Neural Networks. *Pages 41–45 of: Proceedings of the International Conference on Articial Intelligence, Durban, South Africa.*

van den Bergh, F., & Engelbrecht, A. 2000. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal*, 84–90.

van den Bergh, F., & Engelbrecht, A. P. 2001a. Effects of Swarm Size on Cooperative Particle Swarm Optimisers. *Pages 892–899 of:* Spector, Lee, Goodman, Erik D., Wu, Annie, Langdon, W. B., Voigt, Hans-Michael, Gen, Mitsuo, Sen, Sandip, Dorigo, Marco, Pezeshk, Shahram, Garzon, Max H., & Burke, Edmund (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001).* San Francisco, California, USA: Morgan Kaufmann.

van den Bergh, F., & Engelbrecht, A.P. 2001b. Training Product Unit Networks using Cooperative Particle Swarm Optimisers. *In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), Washington DC, USA.*

Veldhuizen, D. Van, & Lamont, G. 2000a. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, **8**(2), 125–147.

Veldhuizen, D. Van, & Lamont, G. 2000b. On Measuring Multiobjective Evolutionary Algorithm Performance. *Pages 204–211 of: Proceedings of the 2000 Congress on Evolutionary Computation.*

Vico, F.J., & Sandoval, F. 1991. Use of Genetic Algorithms in Neural Networks Definition. *Lecture Notes in Computer Science*, **540**, 196–203.

Virili, F., & Freisleben, B. 2000. Nonstationarity and Data Preprocessing for Neural Network Predictions of an Economic Time Series. *Pages 5129–5136 of: Proceedings of IJCNN 2000, Como IEEE/INNS.*

Wang, Y., & Wahl, F.M. 1997. Multiobjective neural network for image reconstruction. *IEE Proceedings - Vision, Image and Signal Processing*, **144**(4), 233–236.

Wen, C-G., & Lee, C-S. 1998. A neural network approach to multiobjective optimization for water quality management in a river basin. *Water Resources Research*, **34**(3), 427–436.

White, D. 1993. GANNet: A Genetic Algorithm for Optimizing Topology and Weights in Neural Network Design. *Lecture Notes in Computer Science*, **686**, 322–327.

Whitehead, B.A., & Choate, T.D. 1996. Cooperative - Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction. *IEEE Transactions on Neural Networks*, **7**(4), 869–880.

Wieland, A.P. 1992. Evolving Neural Network Controllers for Unstable Systems. *Pages 667–673 of: Proceedings of IJCNN'92 Seattle IEEE/INNS*, vol. II.

Wilcoxon, F., & Wilcox, R.A. 1964. *Some Rapid Approximate Statistical Procedures.* New York: Lederle Labs.

Yao, J., & Tan, C.L. 2000. Time dependant Directional Profit Model for Financial Time Series Forecasting. *In: IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks.*

Yao, S., wei, C., & He, Z. 1996. Evolving wavelet neural networks for function approximation. *Electronics Letters*, **32**(4), 360–361.

Yao, X. 1999. Evolving Artificial Neural Networks. *Proceedings of the IEEE*, **87**(9), 1423–1447.

Yao, X., & Liu, Y. 1997. A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, **8**(3), 694–713.

Yao, X., & Liu, Y. 1998. Making Use of Population Information in Evolutionary Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, **28**(3), 417–425.

Yao, X., Liu, Y., & Lin, G. 1999. Evolutionary Programming Made Faster. *Ieee transactions on evolutionary computation*, **3**(2), 82–102.

Zaus, M., & Roland, M. 1991. Fusion-Technology and the design of Evolutionary Machines for Neural Networks. *Pages 1165–1168 of: Artificial Neural Networks: Proceedings of the International Conference on Neural Networks.* North-Holland.

Zhang, B-T., & Veenker, G. 1991. Neural Networks that Teach Themselves through Genetic Discovery of Novel Examples. *Pages 690–695 of: Proceedings of IJCNN'91 Singapore IEEE/INNS*, vol. 1.

Zhang, C., & Shoa, H. 2001. An ANN's Evolved by a New Evolutionary System and its Application. *Pages 3562–3563 of: Proceedings of the 39th IEEE conference on Decision and Control*, vol. 4.

Zitar, R., & Hassoun, M. 1995. Neurocontrollers Trained with Rules Extracted by a Genetic Assisted Reinforcement Learning System. *IEEE Transactions on Neural Networks*, **6**(4), 859–876.

Zitzler, E. 1999. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* Ph.D. thesis, Swiss Federal Institute of Technology Zurich (ETH). Diss ETH No. 13398.

Zitzler, E., & Thiele, L. 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, **3**(4), 257–271.

Zitzler, E., Deb, K., & Thiele, L. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, **8**(2), 173–195.

Zitzler, E., Laumanns, M., & Thiele, L. 2001 (May). *SPEA2: Improving the Strength Pareto Evolutionary Algorithm.* Tech. rept. TIK-Report 103. Swiss Federal Institute of Technology Zurich (ETH).