CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimisation
ANN	Artificial Neural Network
BOA	Bayesian Optimisation Algorithm
cGA	Compact Genetic Algorithm
CGA	Classical Genetic Algorithm
CCA	Cooperative Co-evolutionary Architecture
cHM-EDA	Continuous Hierarchical Model Estimation of Distribution Algorithm
CMA-EA	Covariance Matrix Adaptation Evolution Strategy
CNGM	Computational Neuro-genetic Modelling
CoEvo	Co-evolutionary Strategy
D/E	Differential Evolution
EA	Evolutionary Algorithm
ECoS	Evolving Connectionist System
EDA	Estimation of Distribution Algorithm
ES	Evolution Strategy
eSNN	Evolving Spiking Neural Network
FeaSANNT	Feature Selection and Artificial Neural Network Training
FES	Fitness Evaluations
GA	Genetic Algorithm
hHM-EDA	Heterogeneous Hierarchical Model Estimation of Distribution Algorithm
LIF	Leaky Integrate-and-Fire
LSM	Liquid State Machine
MBOA	Mixed Bayesian Optimisation Algorithm
MLP	Multilayer Perceptron
MVG-EDA	Multi-variate Gaussian Model Estimation of Distribution Algorithm
NBC	Naïve Bayesian Classifier
PBIL	Probabilistic Incremental Learning

PBIL_c	Continuous Probabilistic Incremental Learning
QEA	Quantum-inspired Evolutionary Algorithm
QiSNN	Quantum-inspired Spiking Neural Network
sGA	Standard Genetic Algorithm
SNN	Spiking Neural Network
SRM	Spike Response Model
STDP	Spike-timing Dependent Plasticity
UMDA	Uni-variate Marginal Distribution Algorithm
vQEA	Versatile Quantum-inspired Evolutionary Algorithm

ACKNOWLEDGEMENTS

I have the pleasure to acknowledge some of the many people who have inspired, supported, and educated me over the past three years. First and foremost, I am very grateful to my primary supervisor *Prof. Nikola Kasabov* who gave me the chance to start a PhD programme in the beautiful country of New Zealand and become a member of a high-profile, well known, successful and highly international research group. Скъпи Проф. Касабов, благодаря за всичко, което направихте за мен, за доверието, напътствията и неоценимата помощ, които допринесоха толкова много за моето професионално израстване.

I wish to express my sincere appreciation to my secondary supervisor *Dr. Michaël Defoin Platel* who has been a permanent source of stimulation and encouragement throughout my research. His enthusiasm, patience and intellectual rigour were all invaluable contributions to the personal and professional growth of an inexperienced and often very stubborn young student. Michaël, merci beaucoup pour tes précieux conseils académiques, ils m'ont beaucoup aidé!

Many thanks to *Dr. Sue Worner* from the Centre for Advanced Bio-protection Technologies at Lincoln University, New Zealand, for many useful insights into modelling predictive insect distributions and for giving me the opportunity to present my work to her research group at Lincoln University in Christchurch. I also would like to acknowledge my colleague *Dr. Lubica Benuskova* who always provided me with some very valuable feedback on my research and gave detailed insights into the functioning of the academic world.

I am very indebted to *Joyce D'Mello*, the manager and the soul of the Knowledge Engineering and Discovery Research Institute. Her first-rate administrative work, her enthusiasm for solving the everyday problems of students and the many encouraging words were extremely helpful to me and are highly appreciated.

I would like to thank the past and present members of KEDRI at the Auckland University of Technology, who provided a vibrant and intercultural research environment. I am especially grateful to *Simei Wysoski* and *Snjezana Soltic* with whom I had many productive discussions about spiking neural networks; to *Raphael Hu* for many discussions over a cup of coffee and the interesting cultural exchange; to *Peter Hwang* for his professional technical support; to *John Graves* for kindly reading the

drafts of this thesis and providing many insights into the use of the English language; to *Marin Karaivanov* and *Kshitij Dhoble* for kindly helping me out with some translations; to *Harya Widiputra*, *Haza Nuzly* and *Garry Chen* for many scientific and non-scientific discussions.

Last but not least, I am delighted to thank my *parents* and my partner *Kerstin Günther* for their tremendous support during my entire education. Their insight and wisdom prevented me from making some blundering decisions and their emotional support was invaluable for me to overcome the sometimes frustrating periods of a PhD programme.

My research has been carried out with the financial support of the AUT Vice Chancellor's PhD Scholarship and the KEDRI PhD Scholarship.

Chapter 1

INTRODUCTION

The human brain is a highly complex and dense network consisting of approximately 100 billion (10^{11}) interconnected elementary processing units called neurons. These neurons can communicate with each other through the exchange of short electrical pulses, which are also referred to as spikes. Motivated by the desire to better understand the truly remarkable information processing capabilities of the brain, numerous biologically plausible mathematical models have been developed in recent decades. Traditional artificial neural networks (ANN) assume that the neural code used for an information exchange between neurons is based on their average rate of spike emission. This is modelled as a propagation of continuous variables from one processing unit to the next. Increasing evidence from recent neuro-biological experiments suggests that the exact timing of spikes plays a key role in the neural information processing, *cf. e.g.* the early seminal work by Wiersma (1951) and by Segundo, Moore, Stensaas, and Bullock (1963), but also the more recent discussion about spike and rate codes in Gerstner (1999).

Due to the decreasing costs of computational resources, more complex and biologically plausible connectionist models have been developed, namely spiking neural networks (SNN), *cf. e.g.* (Maass, 1999) for an introduction and (Gerstner & Kistler, 2002b) for a comprehensive standard text on the subject. These models use trains of spikes as internal information representation rather than continuous variables. Maass argues that SNNs have at least similar computational power to the traditional ANN, such as the multi-layer perceptron (MLP) derivatives developed by Rumelhart, Hinton, and Williams (1986). Nowadays many studies attempt to use spiking neural networks for practical applications, some of them demonstrating very promising results in solving complex real world problems. Substantial progress has been made in areas such as speech recognition (Verstraeten, Schrauwen, & Stroobandt, 2005), learning rules (Bohte, Kok, & Poutré, 2002), associative memory (Knoblauch, 2005), and function approximation (Iannella & Kindermann, 2005), to name just a few.

An *evolving* spiking neural network (eSNN) architecture was proposed in (Wysoski, Benuskova, & Kasabov, 2006a). The eSNN belongs to the family of Evolving Connectionist Systems (ECoS), which was first introduced in (Kasabov, 1998a) and (Kasabov, 1998b). ECoS based methods represent a class of constructive ANN algorithms that modify both the structure and connection weights of the network as part of the training process. Due to the evolving nature of the network and the employed fast one-pass learning algorithm, the method is able to accumulate information as it becomes available, without the requirement of retraining the network with previously presented training data. ECoS methods have a long history and numerous variants and applications were developed, including fuzzy neural networks (Kasabov, 1998c), self-organising maps (Deng & Kasabov, 2000) and dynamically evolving fuzzy systems (Kasabov & Song, 2002). Additional information about ECoS can be found in the comprehensive text book by Kasabov (2007). The review presented in (Watts, 2009) summarises the latest developments in the ECoS related research areas.

The eSNN proposed in (Wysoski et al., 2006a) was initially designed as a visual pattern recognition system. The classification method is built upon a simplified integrate-and-fire neural model first proposed in (Thorpe, 1997), which was developed to mimic the information processing of the human eye. Applied to a face recognition task, eSNN was reported to have a competitive performance when compared to a number of common pattern recognition methods in the field (Wysoski, Benuskova, & Kasabov, 2006b).

Numerous other studies have investigated the eSNN classifier recently and the method is well established in the scientific community. The generic nature of the eSNN allows its application to a variety of classification problems. In (Wysoski, Benuskova, & Kasabov, 2007), eSNN was applied to a text-independent speaker authentication problem and tested on the speech part of the VidTIMIT audio-visual database obtained from (Sanderson & Paliwal, 2003). The visual part of the same database was later used to study the characteristics of another extension of eSNN. In (Wysoski, Benuskova, & Kasabov, 2008b) a fast and adaptive multi-view pattern recognition system was proposed, where training samples are presented in an on-line fashion to an eSNN, which in turn is trained to learn different views of the presented object. Using a voting mechanism, the system accumulates information over several views of the test object for the recognition of the test samples. The classification result corresponds to the class label that has received the most votes.

Recently an audio-visual pattern recognition system was proposed in (Wysoski, Benuskova, & Kasabov, 2008a), in which the auditory and visual system developed in (Wysoski et al., 2007) and (Wysoski et al., 2008b) respectively were combined

into an integrated architecture that provides a reliable person authentication based on a short video clip. Other applications were presented in (Soltic, Wysoski, & Kasabov, 2008) where eSNN was used to classify data consisting of water and wine samples collected and presented in (de Sousa & Riul Jr., 2002) and (Riul et al., 2004). A comprehensive discussion of most results on previous eSNN related work can be found in the two PhD dissertations of (Wysoski, 2008) and (Soltic, 2009), respectively. These studies also pointed to the need of optimisation algorithms for the identification of adequate feature subsets and eSNN related parameters.

1.1 AIM OF THE STUDY

In order to further improve upon the classification accuracy of eSNN, this thesis proposes a novel framework that allows the application of eSNN to feature selection problems. The extension follows the well known wrapper approach first introduced in (Kohavi & Sommerfield, 1995) and comprehensively discussed in (Kohavi & John, 1997). The wrapper approach combines a classification method with a generic optimisation algorithm, for which Evolutionary Algorithms (EA) are commonly used. The optimisation task for the EA consists in the identification of an optimal feature subset, which maximises the classification accuracy determined by the classifier.

In all of the previous studies on eSNN the neural and learning parameters of the method were manually fine-tuned in order to achieve satisfying classification results. The method involves numerous parameters and finding an appropriate configuration can quickly become a challenging task. Hence, an *integrated* wrapper approach is proposed here, in which an appropriate feature subset is evolved during an evolutionary process, while *simultaneously* the neural and learning-related parameters of eSNN are optimised.

We note that the connection weights of the neural network are *not* subject to the evolutionary optimisation. Instead, the weights are obtained through the use of an efficient one-pass learning algorithm that was developed as part of the eSNN architecture.

Self-adapting parameters promote the straight forward application of eSNN to other problem domains, since only limited expert knowledge is required to configure the method for a specific task. Furthermore, an improvement of the classification performance is expected, since the method can rely on an optimised set of parameters. More specifically, the framework is able to effectively avoid poor classification results that are often the consequence of the choice of inappropriate parameter con-

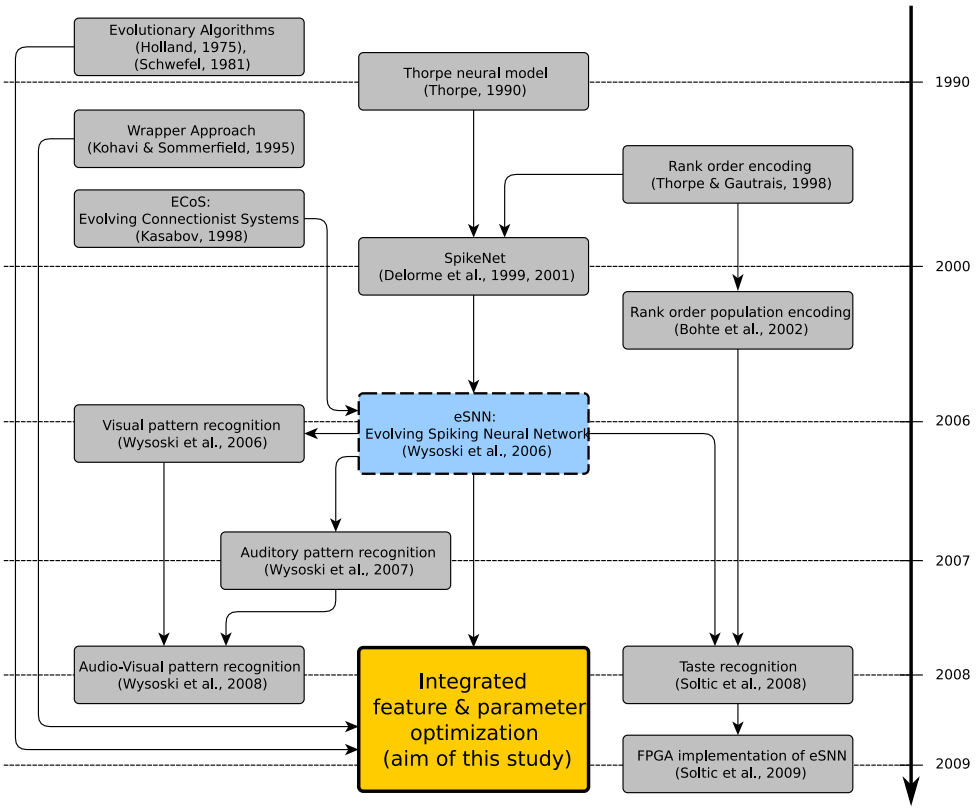


Figure 1.1: Schematic illustration of the historical evolution of the evolving spiking neural network (eSNN) architecture. Based on eSNN a number of applications were developed, especially in the context of visual, auditory and taste recognition problems. This thesis proposes an integrated feature and parameter optimisation method following the wrapper approach with eSNN in its core.

figurations by the experimenter. This characteristic becomes particularly handy, if the method is used for the purpose of data mining and *knowledge discovery* in an area that is not related to SNN. In the context of an increasing amount of interdisciplinary research, self-adaptation is a highly desired property of any method.

The integration of the proposed extension of eSNN in the context of current research in this area, along with the historical evolution of eSNN based systems, is outlined in Figure 1.1. As described above, the development of eSNN was motivated and influenced by a number of previous studies in the area of spiking neural models, neural encoding and evolving connectionist systems.

For the proposed extension of eSNN the need for state-of-the-art optimisation methods arises. The simultaneous exploration of two different search spaces is required: While the feature search space is represented by a string of concatenated bits, where each bit encodes the presence/absence of the corresponding feature, the parameter space of eSNN is a continuous one. The situation is illustrated in Figure 1.2.

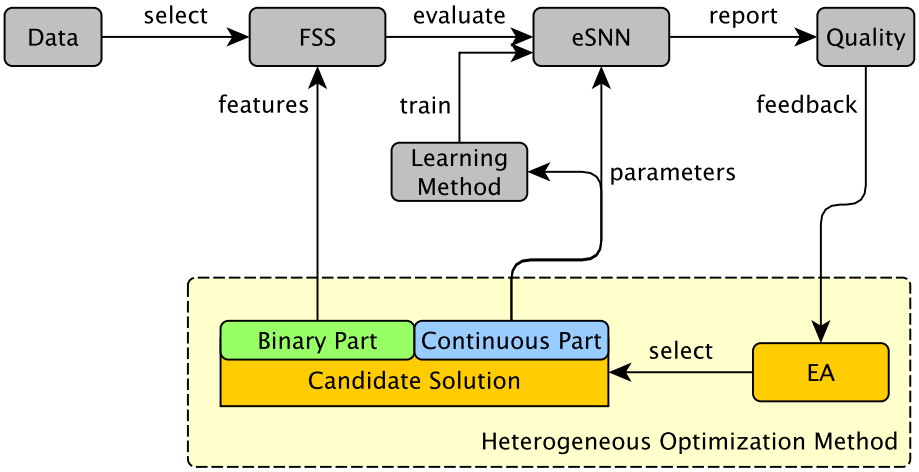


Figure 1.2: Structure of the proposed extension of the eSNN classification method. A specialised evolutionary algorithm evolves a combined solution consisting of a binary and a real-valued sub-component, which represent a feature subset (FSS) for a data sample and a parameter configuration for an eSNN classifier respectively. The quality of this combined solution is evaluated by determining the classification accuracy of eSNN on a set of test samples. The study develops a heterogeneous evolutionary optimisation method (*cf.* the dashed rectangle in the figure).

Given a specific data set, samples are selected and for each of them a feature subset is extracted using a bit mask in which each bit represents a single feature. The quality of this feature subset is then evaluated by the eSNN classification method, which is configured using a specific parameter set, *i.e.* a vector of real values. The quality measure for both the bit mask and the parameter configuration is used as the fitness criterion for an evolutionary algorithm, which in turn proposes a new candidate solution. This solution consists of a binary and a continuous sub-component, that represent a bit mask and a parameter set respectively. The process iterates until a termination criterion is met.

Thus, the aim of this study is to develop an extension of eSNN for the domain of feature selection by means of a heterogeneous optimisation method. The optimisation algorithm has to be developed and studied with the specific focus on a state-of-the-art *performance* in terms of convergence speed and solution quality, a competitive *robustness* in noisy search spaces, and a *small set of parameters* in order to promote its straight forward application to a given problem. Furthermore, the proposed algorithm should not rely on any eSNN specific characteristics that may prevent its application to more general optimisation problems. This property allows an efficient

mechanism for replacing the classification method with more advanced techniques that may be developed in future.

An additional goal of the thesis is the *integration* of the developed method into the context of current research on evolutionary computation and a comprehensive experimental elaboration on its similarities and differences when compared to similar algorithms in the field.

Finally, the thesis aims to *compare* the extended eSNN based feature selection method to some already established algorithms in this research area. In order to emphasise on the *knowledge discovery* aspect of the method, a real-world case study on an ecological modelling problem is undertaken. Dr. Sue Worner from the Centre for Bioprotection at Lincoln University, Christchurch, New Zealand, was invited as an advisor and scientific expert in this research area.

1.2 RESEARCH OBJECTIVES

Considering the fact that an original and *generic* optimisation method is required for the proposed feature selection framework, the task is split into the separate development of a novel *binary* and a novel *continuous optimisation algorithm*. Both methods may be applied independently to either combinatorial or numerical optimisation problems. The appropriate combination of the two optimisers results in a hybrid algorithm, which is finally employed in the desired extension of eSNN to the feature selection domain.

Based on the above considerations, the following list of research objectives is derived.

1. Development of a binary and a continuous optimisation method that can be hybridised to form a heterogeneous optimisation algorithm. This would allow the extension of an eSNN classifier towards an integrated feature and parameter optimisation framework following the wrapper approach.

2. Comprehensive experimental analysis of the optimisation methods in terms of their suitability for real-world applications with explicit focus on robustness, performance and scalability.

3. Integration of the proposed methods into the corresponding research community through experimental comparison to related algorithms in the field.

4. Comparison of the developed eSNN based feature selection framework to other feature selection methods.

5. Demonstration of the applicability of the developed framework for real-world problems through a case study from ecological modelling.

1.3 THESIS STRUCTURE

The structure of the thesis follows the research objectives presented in the previous section and is outlined below.

CHAPTER 2 This chapter reviews current developments in the area of spiking neural networks by providing background information on biological neurons and their mathematical models, along with neural encoding strategies, learning algorithms and applications. A specific emphasis is put on the functioning of the eSNN classification method and the principles of various eSNN based applications. Furthermore, the chapter addresses open problems of eSNN and provides an overview of previously proposed heterogeneous optimisation algorithms.

CHAPTER 3 In line with the presented research objectives, a novel probabilistic binary optimisation method is developed. It improves upon an earlier proposed quantum-inspired evolutionary algorithm (QEA) introduced in (Han & Kim, 2002). Due to its significantly different behaviour the method is introduced as the Versatile QEA (vQEA). The method is compared to the original QEA and a traditional genetic algorithm on a variety of benchmark problems.

CHAPTER 4 This chapter integrates vQEA proposed in chapter 3 into the research field of evolutionary computation by systematically establishing vQEA as an original algorithm belonging to the family of Estimation of Distribution Algorithms (EDA). The characteristics and specifics of vQEA are highlighted and the method is compared to a number of similar EDA using several benchmark problems. This chapter also addresses the question *why* vQEA performs well.

CHAPTER 5 The capability of an optimisation method to handle noisy or inaccurate information obtained from the fitness criterion is generally regarded as a very important pre-condition for a successful application of the method to real world studies. Thus, the robustness of vQEA to noise in comparison to several different EDAs is extensively investigated in this chapter.

CHAPTER 6 A continuous version of vQEA is proposed and investigated on a state-of-the-art benchmark suite and compared to five contemporary, highly competitive numerical optimisers. Specific characteristics and the robustness of the method are studied. Furthermore, guidelines for the configuration of parameters are derived.

CHAPTER 7 A hybrid version of the two previous algorithms is presented. The suitability of this heterogeneous optimiser is demonstrated on a benchmark problem and compared to a variety of related evolutionary algorithms. Guidelines for the configuration of parameters are derived. Similarities and differences to co-evolutionary methods are discussed.

CHAPTER 8 Using the novel heterogeneous optimiser, the eSNN architecture is extended towards the domain of feature selection and parameter optimisation following the wrapper approach. An experimental comparison between the proposed and traditional methods is undertaken. Key principles of the eSNN based feature selection are discussed.

CHAPTER 9 As a demonstration of the inherent suitability of the extended eSNN architecture, the method is applied on a ecological modelling problem. The experimental results are validated by Dr. Sue Worner, who is an ecological expert from the Centre for Bioprotection at Lincoln University, Christchurch, New Zealand.

CHAPTER 10 Conclusions are drawn and future directions for research are given.

1.4 PUBLICATIONS

The material presented in this thesis was partially published in a number of peer-reviewed international conference and journal articles:

- M. Defoin Platel, S. Schliebs, N. Kasabov, A versatile quantum-inspired evolutionary algorithm, CEC 2007, IEEE Congress on Evolutionary Computation, pp.423-430, 25-28 Sept. 2007

- M. Defoin Platel, S. Schliebs, N. Kasabov, Quantum-Inspired Evolutionary Algorithm: A Multi-model EDA, IEEE Transactions on Evolutionary Computation, vol 13 , issue 6, pp.1218 - 1232, 2009

- S. Schliebs, M. Defoin Platel, N. Kasabov, Integrated Feature and Parameter Optimisation for an Evolving Spiking Neural Network, in: M. Koeppen, N. Kasabova and G. Goghill (eds) Advances in neural information processing, Proc. of ICONIP 2008, Auckland, Springer LNCS, pp.1229-1236, 2009

- S. Schliebs, M. Defoin Platel, S. Worner, N. Kasabov, Integrated Feature and Parameter Optimisation for Evolving Spiking Neural Networks: Exploring Heterogeneous Probabilistic Models, Neural Networks, vol 22, issues 5-6, pp.623-632, 2009

- S. Schliebs, M. Defoin Platel, S. Worner, N. Kasabov, Quantum-inspired Feature and Parameter Optimisation of Evolving Spiking Neural Networks with a Case Study from Ecological Modelling, Proc. of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, pp.2833-2840, 2009

- S. Schliebs, Heterogeneous Probabilistic Models for Optimisation and Modelling of Evolving Spiking Neural Networks, Proceedings of the 8th New Zealand Computer Science Research Student Conference, Wellington, New Zealand, 2010

- S. Schliebs, M. Defoin Platel, N. Kasabov, Integrated Feature and Parameter Optimisation based on Evolving Spiking Neural Networks, International Journal of Neural Systems, in print, 2010

- S. Schliebs, M. Defoin Platel, N. Kasabov, Integrated Feature and Parameter Optimization for an Evolving Spiking Neural Network: A Parameter Analysis, Proc. of International Joint Conference on Neural Networks, Barcelona, Spain, in print, 2010

Chapter 2

SPIKING NEURAL NETWORKS – A REVIEW

The next few sections review recent developments in the area of spiking neurons and summarise the main contributions to the research field. First some background information about the functioning of biological neurons is given. Then the most important mathematical neural models are discussed, along with neural encoding techniques, learning algorithms and applications of spiking neurons. The functioning of the eSNN classification method is presented in detail and the principles of numerous eSNN based applications are highlighted and discussed. Furthermore, the chapter addresses a number of open problems of the eSNN method. Finally, an overview of previously proposed heterogeneous optimisation algorithms is provided.

2.1 BIOLOGICAL NEURONS, ELEMENTARY NOTIONS AND CONCEPTS

The brain is arguably the most complex organ of the human body. It contains approximately 10^{11} neurons, which are the elementary processing units of the brain. These neurons are interconnected and form a complex and very dense neural network. On average one cm^3 of brain matter contains 10^4 cell bodies and several kilometres of "wire", *i.e.* connections between neurons in the form of branching cell extensions.

Like most cells in the human body, neurons maintain a certain ion concentration across their cell membrane. Therefore the membrane contains ion pumps which actively transport sodium ions from the intra-cellular to the extra-cellular liquid. Potassium ions are pumped in the opposite direction from the outside to the inside of the cell. Additional to the ion pumps, a number of specialised proteins, so called ion channels, are embedded in the membrane. They allow a slow inward flow of sodium ions into the cell, while potassium ions leak outwards into the extra-cellular liquid. Thus, the ion streams at the channels have opposite directions to the ion pumps. Furthermore, since both ion streams differ in their strengths, an electrical potential exists

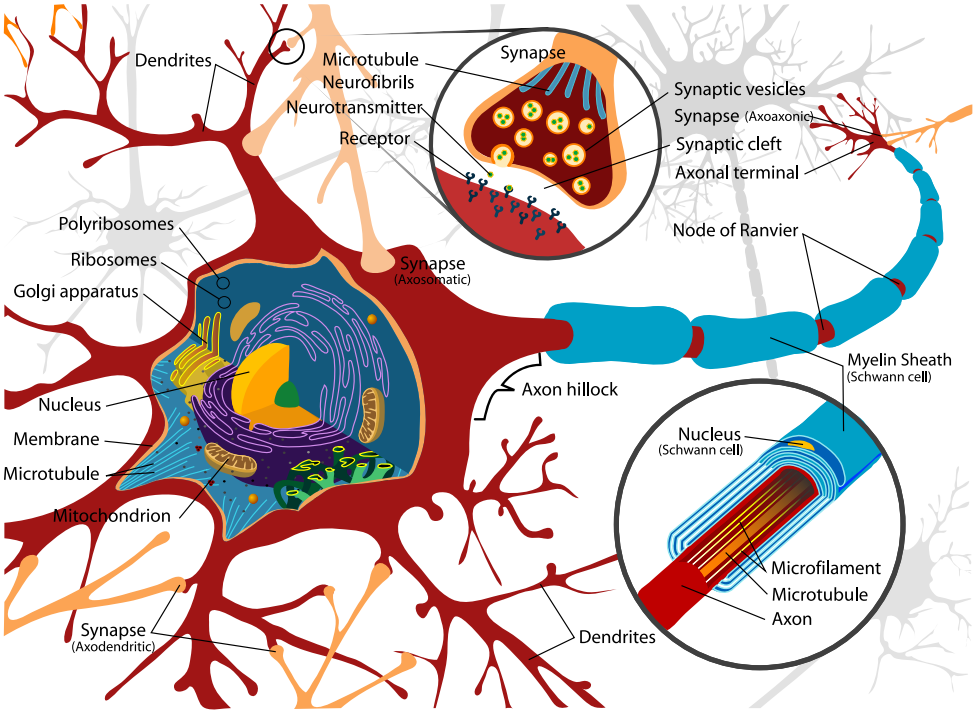


Figure 2.1: Schematic illustration of a typical neuron in the human brain. The main part of the neuron is the soma containing the genetic information, the dendrites and the axon, which are responsible for the reception and emission of electrical signals. Signal transmission occurs at the synapse between two neurons, see text for detailed explanations. The figure is in the public domain and available at http://wikipedia.org.

across the cell membrane. The inside of the cell is negatively charged in relation the extra-cellular liquid. The membrane is polarised which is the resting condition of the neuron.

A large variety of neural shapes and sizes exist in the brain. A typical neuron is illustrated in Figure 2.1. The central part of the neuron is called the soma, in which the nucleus is located. It contains the genetic information of the cell, *i.e.* the DNA, from which genes are expressed and proteins constructed that are important for the functioning of the cell. The cell body has a number of cellular branch-like extensions known as dendrites. Dendrites are specialised for *receiving* electrical signals from other neurons that are connected to them. These signals are short pulses of electrical activity, also known as spikes or action potentials. If a neuron is stimulated by the spike activity of surrounding neurons and the excitation is strong enough, the cell triggers a spike. The spike is propagated via the axon, a long thin wire-like extension of the cell body, to the axonal terminals. These terminals in turn are connected to the dendrites of surrounding neurons and allow the transfer of information from one

neuron to the other. Thus an axon is responsible for *sending* information to other neurons connected to it. An axon may be covered by myelin sheaths that allow a faster propagation of electrical signals. These sheaths act as insulators and prevent the dissipation of the depolarisation wave caused by an electrical spike triggered in the soma.

Information exchange between two neurons occurs at a synapse which is a specialised structure that links two neurons together. A synapse is illustrated in the upper middle part of Figure 2.1. The sending neuron is called pre-synaptic neuron, while the neuron receiving the signal is called post-synaptic. Sending information involves the generation of an action potential in the soma of the pre-synaptic cell. As described above, this potential is propagated through the axon of the neuron to the axonal terminals. These terminals contain the synapses in which neurotransmitter chemicals are stored. Whenever a spike is propagated through the axon, a portion of these neurotransmitters is released into a small gap between the two neurons also known as the synaptic cleft. The neurotransmitter diffuses into the cleft and interacts with specialised receptor proteins of the post-synaptic neuron. The activation of these receptors causes the sodium ion channels to open, which in turn results in the flow of sodium ions from the extra-cellular liquid into the post-synaptic cell. The ionic concentration across the membrane equalises rapidly and the membrane depolarises. Immediately after the depolarisation the potassium channels open. As a consequence potassium ions stream outside the cell, which causes the re-polarisation of the membrane. The process of de- and re-polarisation, *i.e.* the action potential, lasts only around 2ms, which explains the name spike or pulse.

A synaptic transmission can be either excitatory or inhibitory depending on the type of the transmitting synapse. Different neurotransmitters and receptors are involved in excitatory and inhibitory synaptic transmissions respectively. Excitatory synapses release a transmitter called L-glutamate and increase the likelihood of the post-synaptic neuron triggering an action potential following stimulation. On the other hand, inhibitory synapses on the other hand, release a neurotransmitter called GABA and decrease the likelihood of a post-synaptic potential.

The efficacy of a synapse, *i.e.* the strength of the post-synaptic response due to the neurotransmitter release in the synapse, is not fixed. The increase or decrease of the efficacy of a synapse is called *synaptic plasticity* and it enables the brain to learn and to memorise. Several different possibilities exist to accomplish synaptic plasticity. One way is to change the time period of receptor activity in the post-synaptic neuron. Longer periods of receptor activity cause the ion channels to remain open for a longer time, which in turn results in a larger amount of ions flowing into the post-synaptic

cell. Thus, the post-synaptic response increases. Short periods of receptor activity have the opposite effect.

Another way to change the synaptic efficacy is to increase or decrease the number of receptors, which would have a direct impact on the number of opened ion channels and as a consequence on the post-synaptic potential. The third possibility is a change of the amount of neurotransmitter chemicals released into the synaptic cleft. Here larger/smaller amounts would increase/decrease the synaptic efficacy.

Comprehensive information and details about the structure, functions, chemistry and physiology of neurons can be found in the standard text book on the matter by Kandel (2000).

2.2 MODELS OF SPIKING NEURONS

The remarkable information processing capabilities of the brain have inspired numerous mathematical abstractions of biological neurons. Spiking neurons represent the third generation of neural models, incorporating the concepts of time, neural and synaptic state explicitly into the model (Maass, 1997). Earlier artificial neural networks were described in terms of mean firing rates and used continuous signals for transmitting information between neurons. Real neurons, however, communicate by short pulses of electrical activity. In order to simulate and describe biologically plausible neurons in a mathematical and formal way, several different models have been proposed in the recent past. Figure 2.2 illustrates schematically the mathematical abstraction of a biological neuron.

Neural modelling can be described on several levels of abstraction. On the microscopic level, the neuron model is described by the flow of ions through the channels of the membrane. This flow may, among other things, depend on the presence or absence of various chemical messenger molecules. Models at this level of abstraction include the Hodgkin-Huxley model (Hodgkin & Huxley, 1952) and the compartment models that describe separate segments of a neuron by a set of ionic equations.

On the other hand, the macroscopic level treats a neuron as a homogeneous unit, receiving and emitting spikes according to some defined internal dynamics. The underlying principles of how a spike is generated and carried through the synapse, dendrite and cell body is not relevant. These models are typically known under the term integrate-and-fire models.

In the next sections the major neural models are discussed and their functions are explained. Since the macroscopic neuronal models are more relevant for this thesis,

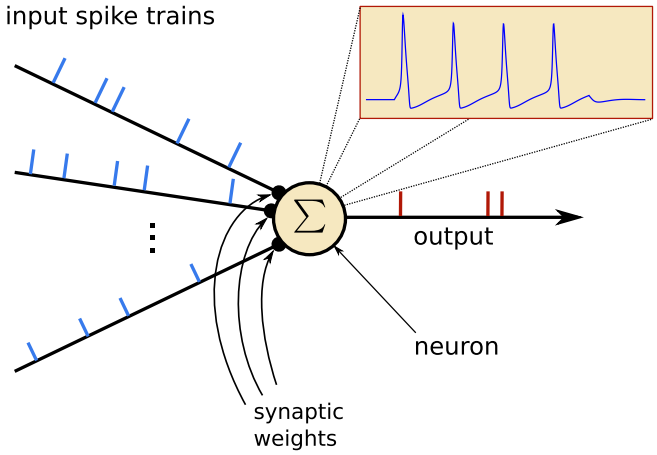


Figure 2.2: A schematic illustration of a mathematical neuronal model. The model receives electrical stimulation in form of spikes through a number of connected pre-synaptic neurons. The efficacy of a synapse is modelled in the form of synaptic weights. Most models focus on the dynamics of the post-synaptic potential only. Output spikes are propagated via the axon to connected post-synaptic neurons.

the focus of the survey is put on these models. The only microscopic model presented here is the Hodgkin-Huxley model, due to its high significance for the research area of neuroscience.

2.2.1 *Hodgkin-Huxley Model*

This model dates back to the work of Alan Lloyd Hodgkin and Andrew Huxley in 1952 where they performed experiments on the giant axon of a squid (Hodgkin & Huxley, 1952). Due to the significance of their contribution to neuroscience, both received the 1963 Nobel Price in Physiology and Medicine. The model is a detailed description of the influences of the conductance of ion channels on the spike activity of the axon. The diameter of the squid's giant axon is approximately 0.5mm and is visible to the naked eye. Since electrodes had to be inserted into the axon, its large size was a big advantage for biological analysis at that time.

Hodgkin and Huxley discovered three different ion currents in a neuron: a sodium, potassium and a leak current. Voltage-dependent ion channels control the flow of ions through the cell membrane. Due to an active transport mechanism, the ion concentration within the cell differs from that in the extra-cellular liquid, resulting in an electrical potential across the cell membrane. In the mathematical model such a membrane is described as an electrical circuit consisting of a capacitor, resistors and batteries that model the ion channels, *cf.* Figure 2.3. The current I at a time t splits

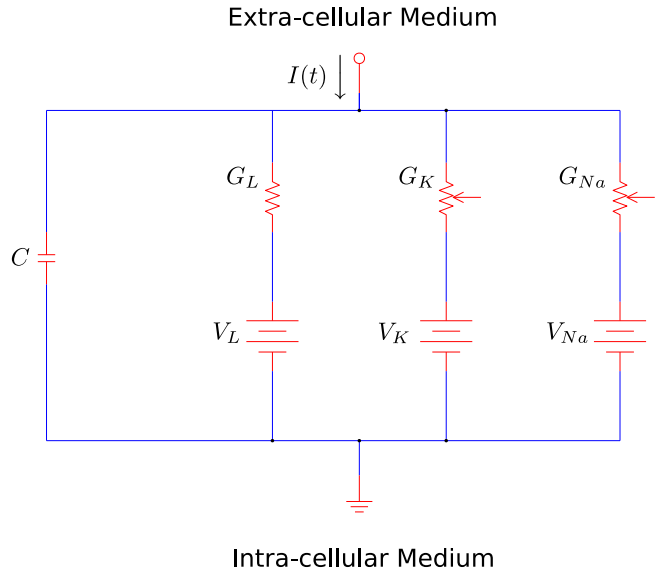


Figure 2.3: The schematic illustration of the Hodgkin-Huxley model in the form of an electrical circuit according to (Hodgkin & Huxley, 1952). The model represents the biophysical properties of the cell membrane of a neuron. The semipermeable cell membrane separates the interior of the cell from the extra-cellular liquid and thus acts as a capacitor. Ion movements through the cell membrane (in both directions) are modelled in the form of (constant and variable) resistors. In the diagram the conductance of the resistors $G_x = 1/R_x$ is shown. Three ionic currents exist: A sodium current (Na ions), potassium current (K ions) and a small leakage current (L) that is primarily carried by chloride ions.

into the current stored in the capacitor and the additional currents passing through each of the ion channels:

$$I(t) = I_{\text{cap}}(t) + \sum_k I_k(t) \tag{2.1}$$

where the sum runs over all ion channels.

Substituting $I_{\text{cap}}(t) = Cdu/dt$ by applying the definition of the capacitance $C = Q/u$, where Q is the charge and u the voltage across the capacitor leads to

$$C\frac{du}{dt} = -\sum_k I_k(t) + I(t) \tag{2.2}$$

As mentioned earlier, in the Hodgkin-Huxley model three ion channels are modelled: A sodium current, potassium current and a small leakage current that is primarily

x	V_x (in mV)	G_x (in mS/cm^2)
Na	115	120
K	-12	36
L	10.6	0.3

x	$\alpha_x(u)$	$\beta_x(u)$
n	$\frac{0.1-0.01u}{\exp(1-0.1u)-1}$	$0.125\exp(-\frac{u}{80})$
m	$\frac{2.5-0.1u}{\exp(2.5-0.1u)-1}$	$4\exp(-\frac{u}{18})$
h	$0.07\exp(-\frac{u}{20})$	$\frac{1}{\exp(3-0.1u)+1}$

Table 2.1: Parameters of the Hodgkin-Huxley model. The membrane capacitance is $C = \mu$F/cm^2. The voltage scale is shifted in order to have a resting potential of zero.

carried by chloride ions. Hence the sum in Equation 2.2 consists of three different components that are formulated as

$$\sum_k I_k(t) = G_{Na}m^3h(u - V_{Na}) + G_K n^4(u - V_K) + G_L(u - V_L) \tag{2.3}$$

where V_{Na}, V_K and V_L are constants called reverse potentials. Variables G_{Na} and G_K describe the maximum conductance of the sodium and potassium channel respectively, while the leakage channel is voltage-independent with a conductance of G_L. The variables m, n and h are gating variables whose dynamics are described by differential equations of the form

$$\frac{m}{dt} = \alpha_m(u)(1 - m) - \beta_m(u)m \tag{2.4}$$
$$\frac{n}{dt} = \alpha_n(u)(1 - n) - \beta_n(u)n \tag{2.5}$$
$$\frac{h}{dt} = \alpha_h(u)(1 - h) - \beta_h(u)h \tag{2.6}$$
$$\tag{2.7}$$

where m and h control the sodium channel and variable n the potassium channel. Functions α_x and β_x, where $x \in \{m, n, h\}$, represent empirical functions of the voltage across the capacitor u, that need to be adjusted in order to simulate a specific neuron. Using a well parametrised set of the above equations, Hodgkin and Huxley were able to describe a significant amount of data collected from experiments with the giant axon of a squid. The discovered parameters of the model are given in Table 2.1

The dynamics of the Hodgkin-Huxley model are presented in Figure 2.4. For the simulation, the parameter values from Table 2.1 are utilised. The membrane is stimulated by a constant input current $I_0 = 7\mu$A, switched on at time $t = 10$ms for a duration of 70ms. The current is switched off at time $t = 80$ms. For $t < 10$ms,

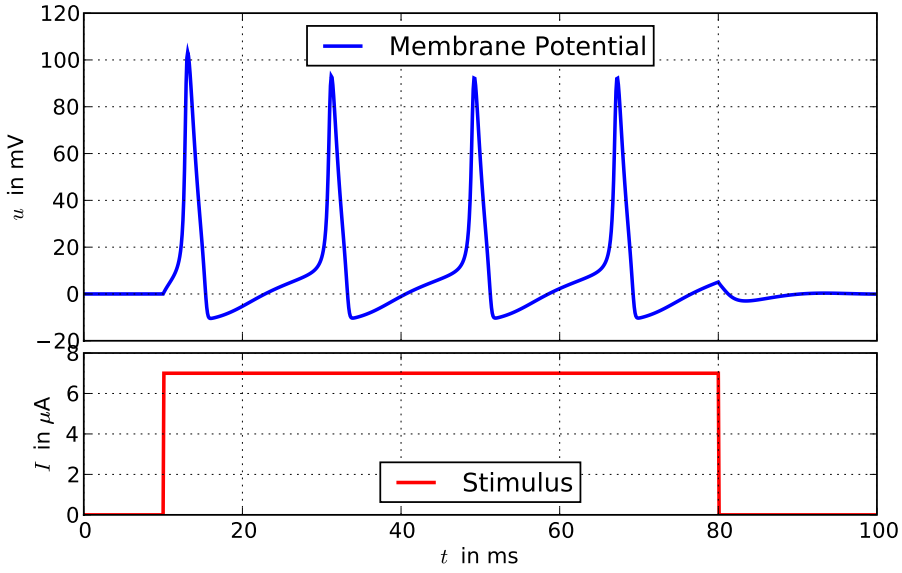


Figure 2.4: Evolution of the membrane potential u for a content input current I_0 using the Hodgkin-Huxley model. The current is switched on at time $t = 10$ms for a duration of 70ms, *cf.* lower diagram. The stimulus is strong enough to generate a spike train across the cell membrane (upper diagram). As soon as the input current vanishes ($I = 0$), the electrical potential returns to its resting potential ($u = 0$).

no input stimulus occurs and the potential across the membrane stays at the resting potential. For $10 \leq t \leq 80$ the current is strong enough to generate a sequence of spikes across the cell membrane. At time $t > 80$ms and input current $I = 0$, the electrical potential returns to its resting potential.

Additional reading on the Hodgkin-Huxley model can be found in the excellent review of Nelson and Rinzel (1995), which also summarises the historical developments of the model. A guideline for computer simulations of the model using the simulation platform GENESIS[1] can be found in (Bower & Beeman, 1995).

2.2.2 *Leaky Integrate-and-Fire Model (LIF)*

The Hodgkin-Huxley model can reproduce electrophysiological measurements very accurately. Nevertheless, the model is computationally costly and simpler, more phenomenological models are required for the simulation of larger networks of spiking neurons. The leaky integrate-and-fire neuron (LIF) may be the best known model for

1 Acronym for **GE**neral **NE**ural **SI**mulation **S**ystem

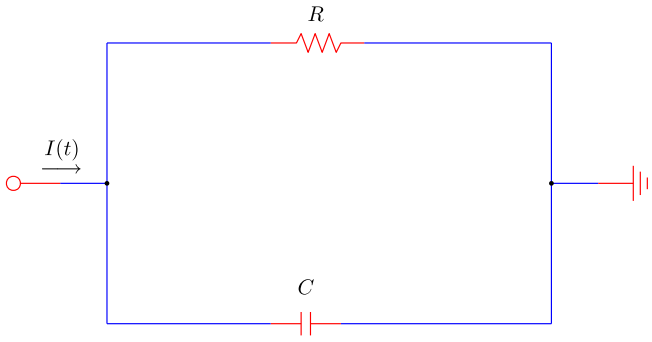


Figure 2.5: The schematic illustration of the leaky integrate and fire model in the form of an electrical circuit. The model consists of a capacitor C in parallel with a resistor R, driven by a current $I = I(R) + I_{\text{cap}}$.

simulating spiking networks efficiently. The model has a long history and was first proposed by Louis Lapicque in 1907, long before the actual mechanisms of action potential generation were known (Lapicque, 1907). Discussions of this work can be found in (Abbott, 1999) and in (Brunel & Rossum, 2007). However, it was Bruce Knight who introduced the term "Integrate-and-Fire" in (Knight, 1972). He called these models "forgetful", but the term "leaky" quickly became more popular.

Similar to the Hodgkin-Huxley model, the LIF model is based on the idea of an electrical circuit, *cf.* Figure 2.5. The circuit contains a capacitor with capacitance C and a resistor with a resistance R, where both C and R are assumed to be constant. The current $I(t)$ splits into two currents:

$$I(t) = I_R + I_{\text{cap}} \tag{2.8}$$

where I_{cap} charges the capacitor and I_R passes through the resistor. Substituting $I_{\text{cap}} = C\ du/dt$ using the definition for capacity, and $I_R = u/R$ using Ohm's law, where u is the voltage across the resistor, one obtains:

$$I(t) = \frac{u(t)}{R} + C\frac{du}{dt} \tag{2.9}$$

Replacing $\tau_m = RC$ yields the standard form of the model:

$$\tau_m \frac{du}{dt} = -u(t) + R\, I(t) \tag{2.10}$$

The constant τ_m is called the membrane time constant of the neuron. Whenever the membrane potential u reaches a threshold ϑ, the neuron fires a spike and its potential is reset to a resting potential u_r. It is noteworthy that the shape of the spike itself

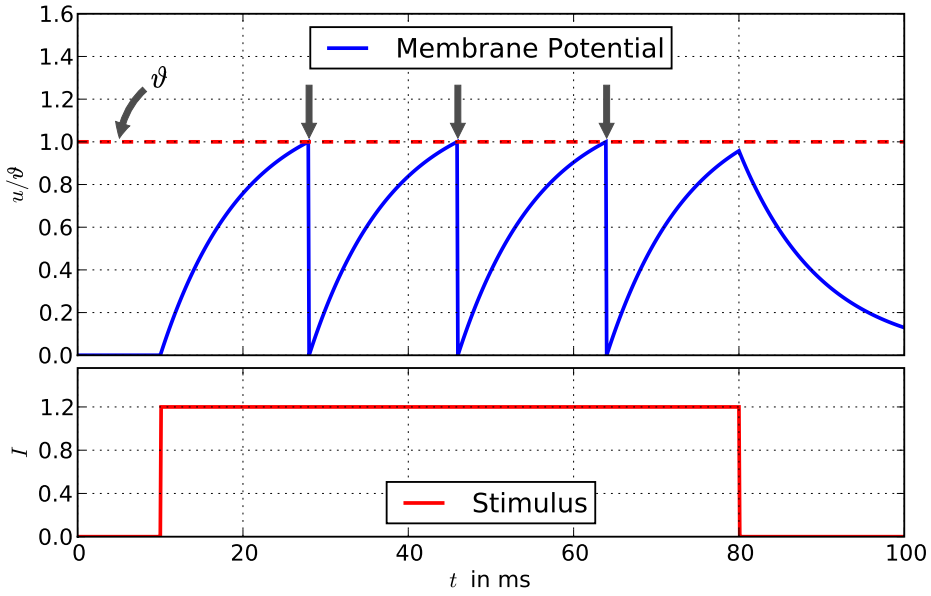


Figure 2.6: The figure shows the evolution of the potential u for a constant input current I_0 using the leaky integrate-and-fire model. The membrane potential u is given in units of the threshold ϑ. The current is switched on at time $t = 10$ms for a duration of 70ms, *cf.* lower diagram. The stimulus is strong enough to generate a sequence of spike trains, *cf.* straight dark arrows. As soon as the input current vanishes, the potential returns to its resting potential.

is not explicitly described in the traditional LIF model. Only the firing times are considered to be relevant. Nevertheless, it is possible to include the shape of spikes as well, *cf. e.g.* (Meffin, Burkitt, & Grayden, 2004).

A LIF neuron can be stimulated by either an external current I_{ext} or by the synaptic input current I_{syn} from pre-synaptic neurons. The external current $I(t) = I_{ext}(t)$ may be constant or represented by a function of time t. Figure 2.6 presents the dynamics of a LIF neuron stimulated by an input current $I_0 = 1.2$. The current is strong enough to increase the potential u until the threshold ϑ is reached. As a consequence, a spike is triggered and the potential resets to $u_r = 0$. After the reset, the integration process starts again. At $t = 80$ms, the current is switched off and the potential returns to its resting potential due to leakage.

If a LIF neuron is part of a network of neurons, it is usually stimulated by the activity of its pre-synaptic neurons. The resulting synaptic input current of a neuron

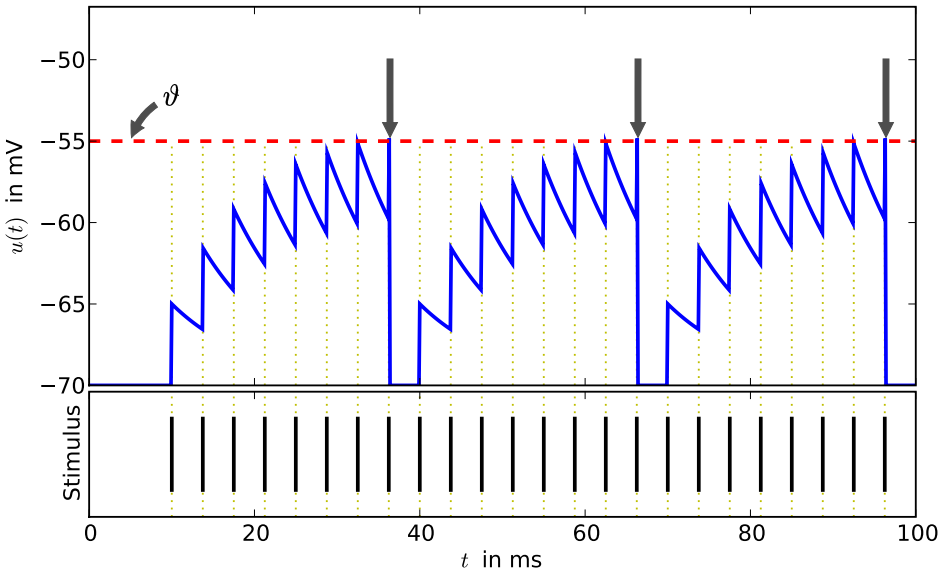


Figure 2.7: The dynamics of the leaky integrate-and-fire model. The potential u increases due to the effect of pre-synaptic input spikes. If the membrane potential crosses a threshold ϑ, a spike is triggered, *cf.* straight dark arrows. The shape of this action potential is not explicitly described by the model, only the time of the event is of relevance. The synapse may have either an inhibitory or an excitatory effect on the post-synaptic potential that is determined by the sign of the synaptic weights.

i is the weighted sum over all spikes generated by pre-synaptic neurons j with firing times $t_j^{(f)}$:

$$I(t) = I_{syn_i}(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)}) \tag{2.11}$$

The weights w_{ij} reflect the efficacy of the synapse from neuron j to neuron i. Negative weights correspond to inhibitory synapses, while positive weights correspond to excitatory synapses. The time course of the post-synaptic current $\alpha(\cdot)$ can be defined in various ways. In the simplest form it is modelled by Dirac pulse $\delta(x)$, which has a non-zero function value for $x = 0$ and zero for all others. Thus the input current caused by a pre-synaptic neuron decreases/increases the potential u in a step-wise manner. More realistic models often employ different functions usually in the form $x \exp(-x)$, which is typically referred to as an α function.

In Figure 2.7, a LIF neuron is stimulated by a spike train from a single pre-synaptic neuron. The post-synaptic current is modelled in the form of a Dirac pulse as described above. This results in a step-wise increase of the post-synaptic potential. If

the potential reaches the threshold ϑ, a spike is triggered and the potential resets. Due to its simplicity, many LIF neurons can be connected to form large networks, while still allowing an efficient simulation.

Extensive additional information about the LIF model can be found in the excellent text book by Gerstner and Kistler (2002b) and in the two recent reviews by Anthony N. Burkitt, (Burkitt, 2006a) and (Burkitt, 2006b).

2.2.3 *Izhikevich Model*

Another neural model was proposed by Izhikevich (2003). It is based on the theory of dynamical systems. The model claims to be as biologically plausible as the Hodgkin-Huxley model while offering the computational complexity of LIF models. Depending on its parameter configuration, the model reproduces different spiking and bursting behaviours of cortical neurons. Its dynamics are governed by two variables:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{2.12}$$

$$\frac{du}{dt} = a(bv - u) \tag{2.13}$$

where v represents the membrane potential of the neuron and u is a membrane recovery variable, which provides negative feedback for v. If the membrane potential reaches a threshold $\vartheta = 30$mV, a spike is triggered and a reset of v and u occurs:

$$\text{if } v \geq 30\text{mV, then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2.14}$$

Variables a, b, c, d are parameters of the model. Depending on their setting, a large variety of neural characteristics can be modelled. Each parameter has an associated interpretation: Parameter a represents the decay rate of the membrane potential, b is the sensitivity of the membrane recovery, and c and d reset the v and u respectively.

In Figure 2.8, the meaning of the parameters is graphically explained along with their effect on the dynamics of the model. For example, if we want to produce a regular spiking neuron, we would set $a = 0.02$, $b = 0.25$, $c = -65$ and $d = 8$. The figure was generated by a freely available simulation tool provided by Eugene M. Izhikevich on his website[2].

More information on this model can be found in the recently published textbook on dynamical systems in neuroscience (Izhikevich, 2006). There are also a number

2 http://www.izhikevich.com

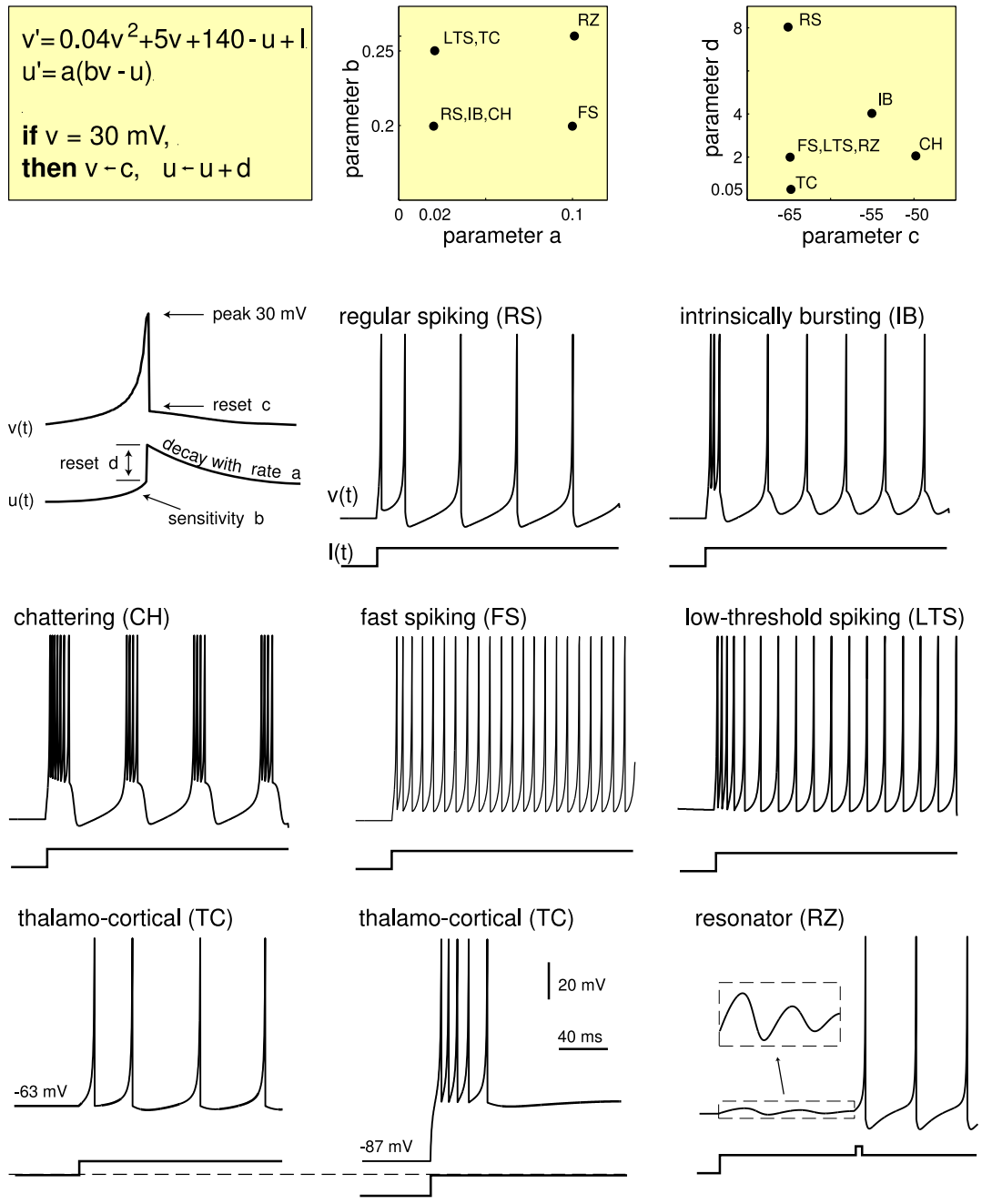


Figure 2.8: The dynamics of the Izhikevich model. Depending on the settings of the parameters a, b, c and d, different neuron characteristics are modelled. The electronic version of the figure and reproduction permissions are available at http://www.izhikevich.com.

of articles on the topic, *cf. e.g.* the work on the suitability of mathematical models for simulation of cortical neurons (Izhikevich, 2004), and the large-scale simulation of a mammalian thalamocortical system (Izhikevich & Edelman, 2008), which involves one million neurons and almost half a billion synapses.

2.2.4 *Spike Response Model (SRM)*

The Spike Response Model (SRM) is a generalisation of the LIF model and was introduced by Gerstner and Kistler (2002b). In this model, the state of a neuron is characterised by a single variable u. A number of different kernel functions describe the impact of pre-synaptic spikes and external stimulation on u, but also the shape of the actual spike and its after-potential. Whenever the state u reaches a threshold ϑ from below, *i.e.* $u(t) = \vartheta$ and $du(t)/dt > 0$, a spike is triggered. In contrast to the LIF model, the threshold ϑ in SRM is not required to be fixed, but may depend on the last firing time $\hat{t}_i$ of neuron i. For example, the threshold might be increased after the neuron has spiked (also known as the refractory period) to avoid triggering another spike during that time.

Let $u_i(t)$ be the state variable that describes neuron i at time t and $\hat{t}_i$ is the last time when the neuron emitted a spike, then the evolution of $u_i(t)$ can be formulated as:

$$\begin{aligned} u_i(t) &= \eta(t-\hat{t}_i) + \sum_j w_{ij} \sum_f \epsilon_{ij}(t-\hat{t}_i, t-t_j^{(f)}) \\ &\quad + \int_0^\infty \kappa(t-\hat{t}_i, s) I_{ext}(t-s) ds \end{aligned} \tag{2.15}$$

where $t_j^{(f)}$ are the firing times of pre-synaptic neurons j, while w_{ij} represents the synaptic efficacy between neuron j and i.

Functions η, ϵ and κ are response kernels. The first kernel, η, is the reset kernel. It describes the dynamics of an action potential and becomes non-zero each time a neuron fires. This kernel models the reset of the state u and its after-potential. A typical implementation is:

$$\begin{aligned} \eta(t-\hat{t}_i) &= \eta_0 \left(K_1 \exp\left(-\frac{t-\hat{t}_i}{\tau_m} \right) \right. \\ &\quad \left. - K_2 \left(\exp\left(-\frac{t-\hat{t}_i}{\tau_m} \right) - \exp\left(-\frac{t-\hat{t}_i}{\tau_s} \right) \right) \right) \Theta(t-\hat{t}_i) \end{aligned} \tag{2.16}$$

where $\eta_0 = \vartheta$ equals the firing threshold of the neuron. The first term in Equation 2.16 models the positive pulse with a decay rate τ_m and the second one is the negative spike after-potential with a decay rate τ_s, while K_1 and K_2 act as scaling factors. Function $\Theta(\cdot)$ is a step function known as the Heaviside function:

$$\Theta(s) = \begin{cases} 0 & \text{if } s < 0 \\ 1 & \text{if } s \geq 0 \end{cases} \tag{2.17}$$

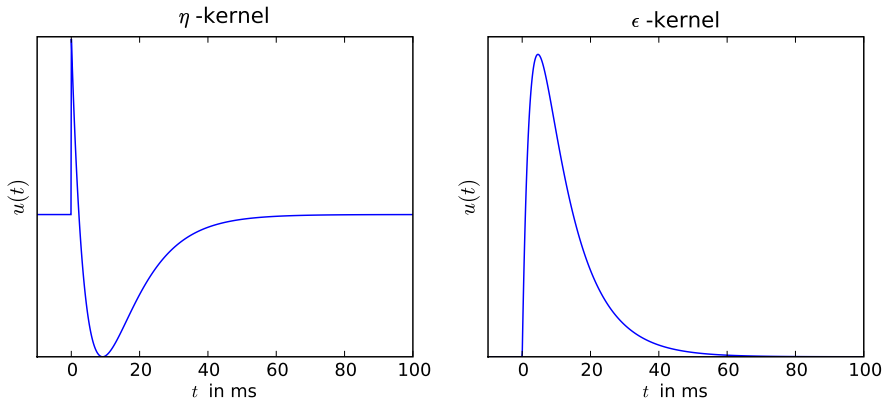


Figure 2.9: Shape of the response kernels η and ϵ. In the left diagram, a spike is triggered at time $t = t^{(f)} = 0$, which results in the activation of the η kernel. The shape of the spike and its after potential are modelled by this kernel function. In the right diagram, the neuron receives an input spike at time $t = 0$ which results in the activation of the ϵ kernel. If no further stimulus is received, the potential u returns to its resting potential.

which ensures that the effect of the η kernel is zero if the neuron has not emitted a spike, *i.e.* $t < \hat{t}$. The shape of this kernel is presented in the left diagram of Figure 2.9. For the figure, $K_1 = 1$, $K_2 = 5$, $\tau_s = 0.005$ and $\tau_m = 0.01$ were used.

The second kernel determines the time course of a post-synaptic potential whenever the neuron receives an input spike. The kernel depends on the last firing time of the neuron $t - \hat{t}$ and on the firing times $t - t_j^{(f)}$ of the pre-synaptic neurons j. Due to the first dependence the post-synaptic neuron may respond differently to input spikes received immediately after a post-synaptic spike. A typical implementation of this kernel is *e.g.*

$$\epsilon(t - \hat{t}, t - t_j^{(f)}) = \left(\exp\left(-\frac{t - t_j^{(f)}}{\tau_m} \right) - \exp\left(-\frac{t - t_j^{(f)}}{\tau_s} \right) \right) \Theta(t - t_j^{(f)}) \quad (2.18)$$

where $\Theta(\cdot)$ once more corresponds to the Heaviside function, the two exponential functions model a positive and a negative pulse with the corresponding decay rates, and $t_j^{(f)}$ is the spike time of a pre-synaptic neuron j. In Equation 2.18, the first dependency of ϵ is neglected, which corresponds to a special case of the model, namely the simplified SRM. This simplified version of SRM is discussed in the next section. The time course of the ϵ kernel of Equation 2.18 is presented in the right diagram of Figure 2.9. For the figure $\tau_s = 0.005$ and $\tau_m = 0.01$ were used. The implementations for the response kernels η and ϵ are adopted from the study on spike timing dependent plasticity in (Masquelier, Guyonneau, & Thorpe, 2008).

The third kernel function κ represents the linear response of the membrane to an input current I_{ext}. It depends on the last firing time of the neuron $t - \hat{t}$ and the time prior to t. It is used to model the time course of u due to external stimuli to the neuron.

A comprehensive discussion of the spike response model and its derivates can be found in the excellent textbook by Gerstner and Kistler (2002b) and also in (Maass & Bishop, 1999).

Simplified Model (SRM$_0$)

In a simplified version of SRM, the kernels ϵ and κ are replaced:

$$\epsilon_0(s) = \epsilon_{ij}(\infty, s) \tag{2.19}$$

$$\kappa_0(s) = \kappa_{ij}(\infty, s) \tag{2.20}$$

which makes the kernels independent of the index j of pre-synaptic neurons and also of the last firing time $\hat{t}_i$ of the post-synaptic neuron. Using simple implementations of these kernel functions reduces the computational cost significantly. Hence, this model has been used to analyse the computational power of spiking neurons (Maass, 1994, 1999), of network synchronisation (Gerstner, Hemmen, & Cowan, 1996) and collective phenomena of coupled networks (Kistler, Seitz, & Hemmen, 1998).

The dynamics of the SRM$_0$ model are presented in Figure 2.10. For the diagram, the ϵ and η kernels are defined by Equation 2.18 and 2.16, respectively. The neuron receives a pre-synaptic stimulus in the form of several spikes which impact the potential u according to the response kernel ϵ. Due to the pre-synaptic activity, an action potential is triggered at time $t = 77.9$ms which results in the activation of the η kernel and the modelling of the spike shape and the after-potential. The figure only presents excitatory synaptic activity.

2.2.5 *Thorpe Model*

A simplified LIF model was formally proposed in (Thorpe & Gautrais, 1998). However, the general idea of the model can be traced back to publications as early as 1990, *cf.* (Thorpe, 1990). This model lacks the post-synaptic potential leakage. The spike response of a neuron depends only on the arrival time of pre-synaptic spikes. The importance of early spikes is boosted and affects the post-synaptic potential more strongly than later spikes. This concept is very interesting due to the fact that the brain

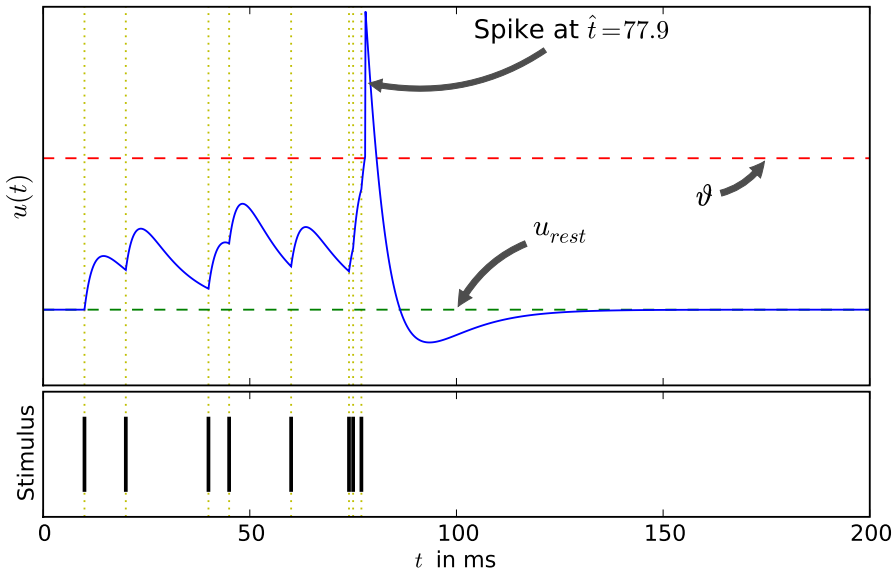


Figure 2.10: The dynamics of the spike response model (SRM). In the post-synaptic neuron, spikes change the membrane potential described by the kernel function ϵ. If the membrane potential crosses a threshold ϑ, a spike is triggered. The shape of this action potential is modelled by the function η.

is able to compute even complex tasks quickly and reliably. For example, the human brain requires for the processing of visual data only approximately 150ms (Thorpe, Fize, & Marlot, 1996), see also a similar study on rapid visual categorisation of natural and artificial objects (Van Rullen & Thorpe, 2001). Since it is known that this type of computation is partly sequential and several parts of the brain involving millions of neurons participate in the computation, it has been argued in (Thorpe & Gautrais, 1996) and (Thorpe, 1997) that each neuron has time and energy to emit only very few spikes that can actually contribute to the processing of the input. As a consequence, few spikes per neuron are biologically sufficient to solve a highly complex recognition task in real time.

Similar to other models, the dynamics of the Thorpe model are described by the evolution of the post-synaptic potential $u_i(t)$ of a neuron i:

$$u_i(t) = \begin{cases} 0 & \text{if fired} \\ \sum_{j|f(j)<t} w_{ji}\, m_i^{order(j)} & \text{else} \end{cases} \tag{2.21}$$

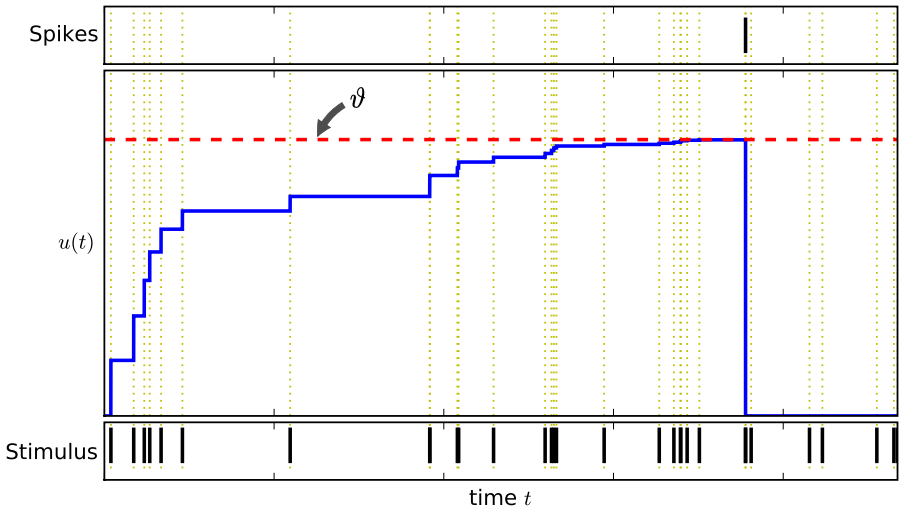


Figure 2.11: Evolution of the post-synaptic potential (PSP) of the Thorpe neuronal model for a given input stimulus. If the potential reaches threshold ϑ, a spike is triggered and the PSP is set to 0 for the rest of the simulation, even if the neuron is still stimulated by incoming spike trains.

where w_{ji} is the weight of a pre-synaptic neuron j, $f(j)$ is the firing time of j, and $0 < m_i < 1$ is a parameter of the model, namely the modulation factor. Function $order(j)$ represents the rank of the spike emitted by neuron j. For example, a rank $order(j) = 0$ would be assigned if neuron j is the first among all pre-synaptic neurons of i that emits a spike. In a similar fashion, the spikes of all pre-synaptic neurons are ranked and then used in the computation of u_i. A neuron i fires a spike when its potential reaches a certain threshold ϑ. After emitting a spike, the potential resets to $u_i = 0$. Each neuron is allowed to emit only a single spike at most. The threshold $\vartheta = c\, u_{max}$ is set to a fraction $0 < c < 1$ of the maximum potential u_{max} reachable for a neuron. Figure 2.11 presents the change of the post-synaptic potential for the Thorpe neural model if a series of input spikes stimulates the neuron through different synapses.

These simplifications allow a very fast real-time simulation of large networks. Due to its low computational costs this model was mainly used for studying image and speech recognition methods involving thousands of connected neurons (*cf. e.g.* (Delorme & Thorpe, 2003) and (Thorpe, Guyonneau, Guilbaud, Allegraud, & VanRullen, 2004)). Many studies have investigated the Thorpe model, *e.g.* for face recognition (Van Rullen, Gautrais, Delorme, & Thorpe, 1998) and (Delorme, Perrinet, & Thorpe, 2001). Additional studies utilising this model are presented in section 2.6,

where principles and applications of the evolving spiking neural network architecture are discussed.

2.3 NEURAL ENCODING

This section addresses a fundamental question in neuroscience: What is the code used by neurons to transmit information? Is it possible for an external observer to read and understand the message of neural activity? Traditionally, there are two main theories about neural encoding – pulse codes and rate codes. Both theories are discussed below.

2.3.1 *Rate codes*

The first theory assumes that the mean firing rate of a neuron carries the most, maybe even all the information of a transmission. These codes are referred to as rate codes and have inspired the classical perceptron approaches. The mean firing rate v is usually understood as the ratio of the average number of spikes n_{sp} observed over a specific time interval T, and T itself:

$$v = \frac{n_{sp}}{T} \tag{2.22}$$

This concept has been especially successful in the context of sensory or motor neural system, *cf. e.g.* the pioneering work by Adrian on the direct relationship between the firing rate of stretch receptor neurons and the applied force in the muscles of frog legs (Adrian, 1926). Nevertheless, the idea of a mean firing rate has been repeatedly criticised, *cf. e.g.* (Rieke, Warland, Steveninck, & Bialek, 1999). The main argument is the comparably slow transmission of information from one neuron to another, since each neuron has to integrate the spike activity of pre-synaptic neurons at least over a time T. Especially, the extremely short response times of the brain for certain stimuli, can not be explained by the temporal averaging of spikes. For example, in (Thorpe et al., 1996) it was shown that the human brain can recognise a visual stimulus in approximately 150ms. It is known that a moderate number of neural layers are involved in the processing of visual stimuli. If every layer had to wait a period T to receive the information from the previous layer, the recognition time would be much longer.

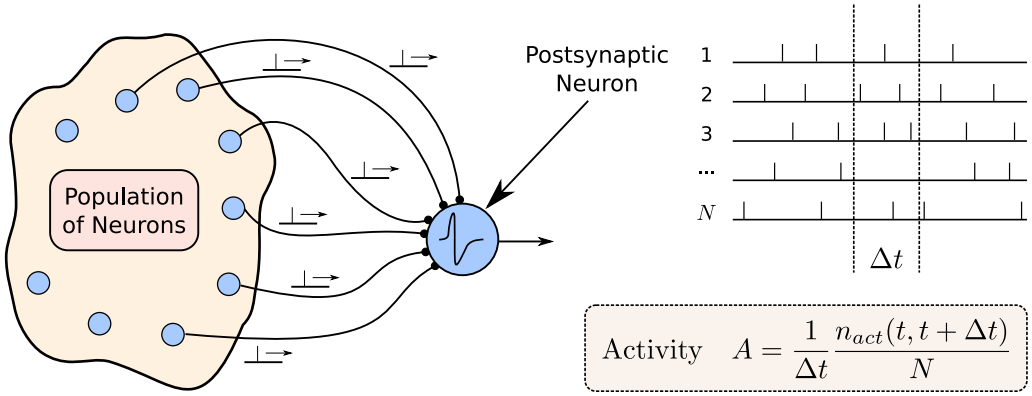


Figure 2.12: A neuron receives input spikes from a population of pre-synaptic neurons producing a certain activity A. The activity is defined as the fraction of neurons being active within a short interval $[t, t + \Delta t]$, divided by the population size N and the time period Δt. The figure was redrawn from a diagram presented in (Gerstner & Kistler, 2002b).

However, there is also another interpretation for the concept of the mean firing rate. It is defined as the average spike activity over a population of neurons. The principle of this interpretation is explained in Figure 2.12. A post-synaptic neuron receives stimulating inputs in the form of spikes emitted by a population of pre-synaptic neurons. This population produces a certain spike activity A which is defined as the fraction of neurons being active within a short interval $[t, t + \Delta t]$:

$$A = \frac{1}{\Delta t}\frac{n_{act}(t, t + \Delta t)}{N} \tag{2.23}$$

where $n_{act}(t, t + \Delta t)$ denotes the number of active neurons in interval $[t, t + \Delta t]$, and N is the total number neuron in the population. The activity of a population may vary rapidly and thus allow fast responses of the neurons to changing stimuli, *cf.* (Gerstner, 2000) and (Brunel, Chance, Fourcaud, & Abbott, 2001).

2.3.2 *Pulse codes*

The second type of neural encoding is referred to as a spike or pulse code. These codes assume the precise spike time as the carrier of information between neurons. Experimental evidence for temporal correlations between spikes was given through computer simulations, *cf. e.g.* the work in (Lestienne, 1995) where integrate-and-fire models are investigated, but also through biological experiments, *cf.* the electrophysiological recordings and staining procedures in (Nawrot, Schnepel, Aertsen, & Boucsein, 2009). See also the *in vivo* measurements described in (Villa, Tetko, Hyland, &

Najem, 1999) in which spatio-temporal patterns of neuronal activity are analysed in order to predict the behaviour responses of rats.

A pulse code based on the timing of the first spike after a reference signal, was discussed in (Thorpe et al., 1996). This encoding is called time-to-first-spike and is inspired by the visual processing of the human eye. It was argued that each neuron has time to emit only few spikes that can contribute to the overall processing of a stimulus. Indeed, it was also shown in (Tovee, Rolls, Treves, & Bellis, 1993) that a new stimulus is processed in the first 20 to 50ms after its onset. Thus, earlier spikes carry most information about the stimulus. A specific neural model, namely the Thorpe model that boosts the importance of early spikes, was discussed already in section 2.2.5.

Other pulse codes consider correlation and synchrony to be important. Neurons that represent a similar concept, object or label are "labeled" by firing synchronously (Malsburg, 1981). More generally, any precise spatio-temporal pulse pattern could be potentially meaningful and encode a particular information. Neurons that fire with a certain relative time delay may signify a certain stimulus.

As a practical example, the so-called rank order population encoding is presented in section 2.6.1. Additional information about neural encoding in general can be found in the book by (Rieke et al., 1999).

2.4 LEARNING IN SNN

This section presents some typical learning methods in the context of spiking neurons. A variety of problems impair the development of learning procedures for SNN. The explicit time dependence results in asynchronous information processing that commonly requires complex software and/or hardware implementations to simulate these neural networks. Additional difficulties are added by the fact that recurrent network topologies are commonly used in SNN and thus the formulation of a straightforward learning method, such as back-propagation for MLP, is not possible.

Similar to traditional neural networks, three different learning paradigms can be distinguished in SNN, which are referred to as unsupervised, reinforcement and supervised learning. Reinforcement learning in SNN is probably the least common among the three. Some algorithms have been successfully applied in robotic applications, *cf. e.g.* (Florian, 2005), but were also theoretically analysed in (Florian, 2007), (Seung, 2003) and (Xie & Seung, 2004). Unsupervised learning in the form of Hebbian learning is the most biologically realistic learning scenario. The so-called

spike-timing dependent plasticity (STDP) belongs to this category and is discussed in the next section. Supervised techniques impose a certain input-output mapping on the network which is essential for practical applications of SNN. Two methods are discussed in greater detail in the next sections. The learning algorithm employed in the eSNN architecture is discussed separately in section 2.6.2. An excellent comparison of supervised learning methods developed for SNN can be found in (Kasinski & Ponulak, 2006).

2.4.1 *STDP – Spike-timing dependent plasticity*

Spike-timing dependent plasticity is inspired by the experiments of Donald O. Hebb published in his famous book "The Organisation of Behaviour" (Hebb, 1949). His essential postulate is often referred to as Hebb's Law:

> When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

First experimental evidence that supports Hebb's postulate was given 20 years later in (Bliss & Lomo, 1973) and (Bliss & Gardner-Medwin, 1973). Today, it is known that the change of synaptic efficacy in the brain is correlated to the timing of pre- and post-synaptic activity of a neuron (Bell, Han, Sugawara, & Grant, 1997; Markram, Lubke, Frotscher, & Sakmann, 1997; Bi & Poo, 1998). Whenever the efficacy of a synapse is strengthened or weakened, we speak of long-term potentiation (LTP) or long-term depression (LTD), respectively. STDP is described by a function $W(t_{pre} - t_{post})$ that determines the fractional change of the synaptic weight in dependence of the difference between the arrival time t_{pre} of a pre-synaptic spike and the time t_{post} of an action potential emitted by the neuron. Function W is also known as the STDP window. Typical approximations of W are ***e.g.*** :

$$W(t_{pre} - t_{post}) = \begin{cases} A_+ \exp(\frac{t_{pre}-t_{post}}{\tau_+}) & \text{if } t_{pre} < t_{post} \\ A_- \exp(-\frac{t_{pre}-t_{post}}{\tau_+}) & \text{if } t_{pre} > t_{post} \end{cases} \tag{2.24}$$

where parameters τ_+ and τ_- determine the temporal range of the pre- and post-synaptic time interval, while A_+ and A_- denote the maximum fractions of synaptic modification, if $t_{pre} - t_{post}$ is close to zero. Figure 2.13 presents the STDP window W according to Equation 2.24.

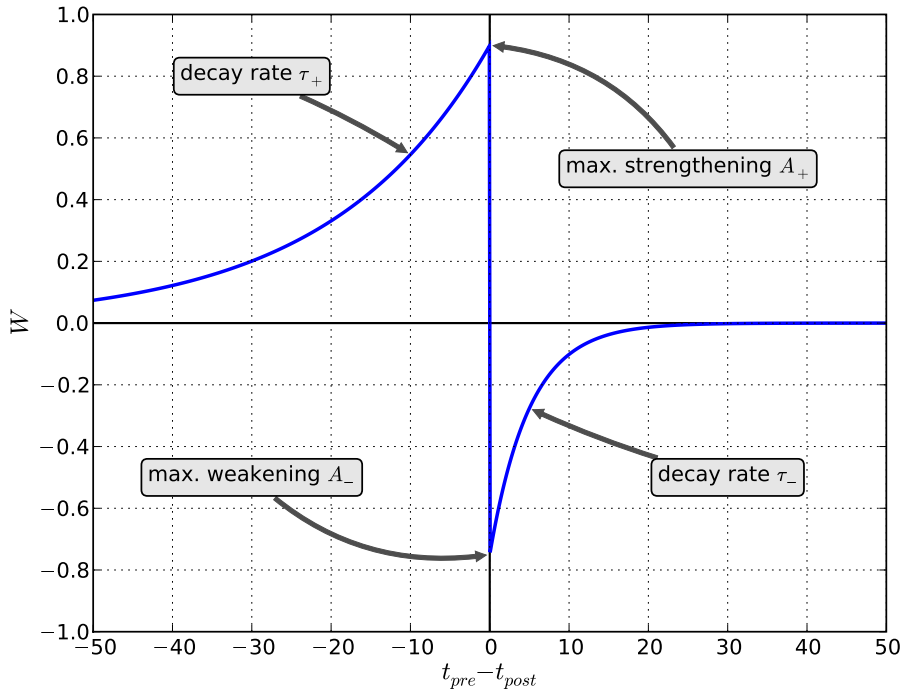


Figure 2.13: STDP learning window W as function of the time difference $t_{pre} - t_{post}$ of pre- and post-synaptic spike times. The presented function is based on Equation 2.24 using the following parameter setting: $A_+ = 0.9$, $A_- = -0.75$, $\tau_+ = 20$ and $\tau_- = 5$.

The parameters for A_+, A_-, τ_+ and τ_- are adjusted according to the particular neuron to be modelled. The window W is usually temporally asymmetric, *i.e.* $A_+ \neq A_-$ and $\tau_+ \neq \tau_-$. However, there are also some exceptions, *e.g.* synapses of layer 4 spiny stellate neurons in the rat barrel cortex appear to have a symmetric window (Egger, Feldmeyer, & Sakmann, 1999).

A study investigated the dynamics of synaptic pruning as a consequence of the STDP learning rule (Iglesias, Eriksson, Grize, Tomassini, & Villa, 2005). Synaptic pruning is a general feature of mammalian brain maturation and refines the embryonic nervous system by removing inappropriate synaptic connections between neurons, while preserving appropriate ones. Later studies extended this work by including apoptosis (genetically programmed cell death) into the analysis (Iglesias & Villa, 2006), and the identification of spatio-temporal patterns in the pruned network indicating the emergence of cell assemblies (Iglesias & Villa, 2007).

More information on STDP can be found in the excellent review on the matter by (Bi & Poo, 2001) and also (Kempter, Gerstner, & van Hemmen, 1999; Gerstner & Kistler, 2002a; Kistler, 2002).

2.4.2 *Spike-Prop*

Traditional neural networks, like the multi-layer perceptron, usually employ some form of gradient based descent, *i.e.* error back-propagation, to modify synaptic weights in order to impose a certain input-output mapping on the network. However, the topological recurrence of SNN and their explicit time dependence do not allow a straightforward evaluation of the gradient in the network. Special assumptions need to be made to develop a version of back-propagation appropriate for spiking neurons.

In (Bohte, Kok, & Poutré, 2000) and (Bohte et al., 2002) a back-propagation algorithm called Spike-Prop is proposed, which is suitable for training SNN. It is derived from the spike-response model discussed in section 2.2.4. The aim of the method is to learn a set of desired firing times t_j^d of all output neurons j for a given input pattern presented to the network. Spike-Prop minimises the error E defined as the squared difference between all network output times t_j^{out} and desired output times t_j^d:

$$E = \frac{1}{2} \sum_j \left(t_j^{out} - t_j^d \right)^2 \tag{2.25}$$

The error is minimised with respect to the weights w_{ij}^k of each synaptic input:

$$\Delta w_{ij}^k = -\eta \frac{dE}{dw_{ij}^k} \tag{2.26}$$

with η defining the learning rate of the update step.

A limitation of the algorithm is given by the requirement that each neuron is allowed to fire only once, which is similar to the limitations of the Thorpe neural model presented in section 2.2.5. This simplification allows the error function defined in Equation 2.25 to depend entirely on the difference between actual and desired spike time. Thus, only time-to-first-spike encoding is suitable in combination with Spike-Prop.

The algorithm was modified in a number of studies. In (Xin & Embrechts, 2001) a momentum term was included in the update of the weights, while (Schrauwen & van Campenhout, 2004) extended the method to learn additional neural parameters,

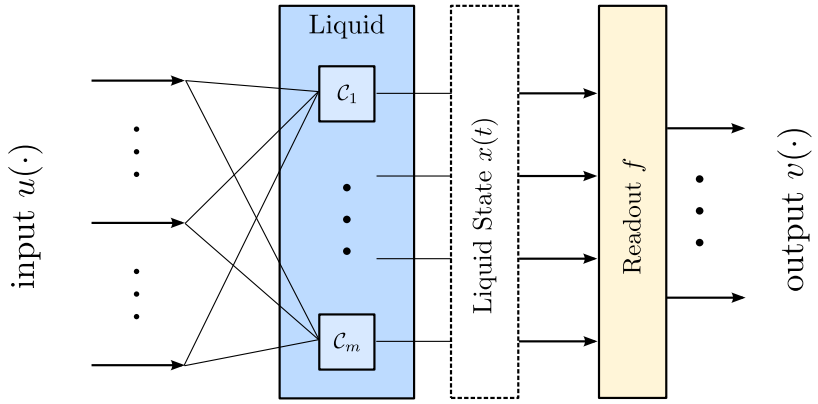


Figure 2.14: Principle of the Liquid State Machine (LSM). The liquid transforms inputs u into a liquid state x which in turn is mapped by a (linear) readout function f into the output v of the network. Figure redrawn from (Natschläger, Maass, & Markram, 2002).

such as synaptic delays, time constants and neuron thresholds. An extension towards recurrent network topologies was presented in (Tiňo & Mills, 2006).

2.4.3 *Liquid State Machine*

A very different approach to neural learning was proposed with the Liquid State Machine (LSM) introduced in (Maass, Natschläger, & Markram, 2002). The method is a specific form of reservoir computing (Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007), that constructs a recurrent network of spiking neurons, for which all parameters of the network, *i.e.* synaptic weights, connectivity, delays, neural parameters, are randomly chosen and fixed during simulation. Such a network is also referred to as a *liquid*. If excited by an input stimulus, the liquid exhibits very complex non-linear dynamics that are expected to reflect the inherent information of the presented stimulus. The response of the network can be interpreted by a learning algorithm.

Figure 2.14 illustrates the principle of the LSM approach. As a first step in the general implementation of the LSM a suitable liquid is chosen. This step determines for example, the employed neural model along with its parameter configuration, as well as the connectivity strategy of the neurons, network size and other network-related parameters. After creating the liquid, so-called liquid states $x(t)$ can be recorded at various time points in response to numerous different (training) inputs $u(t)$. Finally, a supervised learning algorithm is applied to a set of training examples of the form

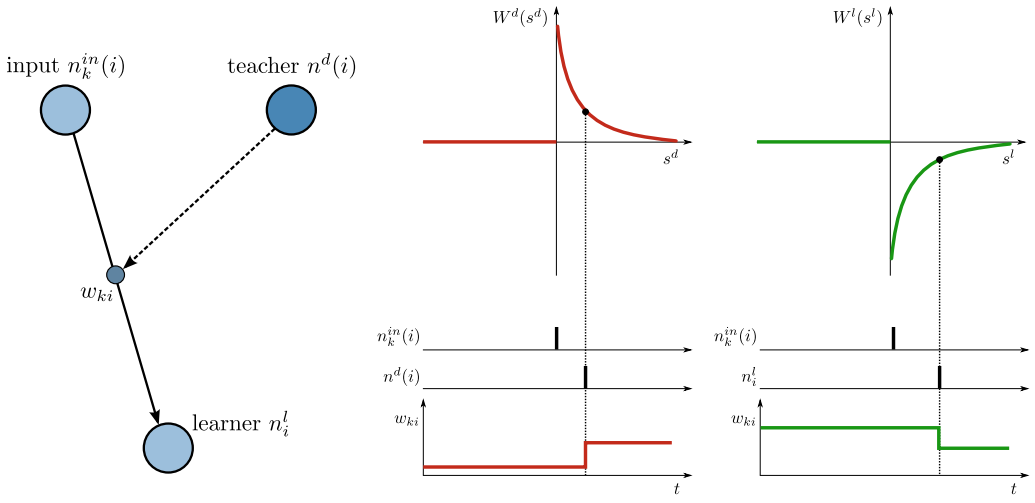


Figure 2.15: Schematic illustration of the Remote Supervised Method (ReSuMe). The synaptic change depends on the correlation of spike activities between input, teaching and learning neurons. Spikes emitted by neuron input $n_k^{in}(i)$ followed by a spike of the teacher neuron $n^d(i)$ leads to an increase of synaptic weight w_{ki}. The value of w_{ki} is decreased, if $n_k^{in}(i)$ spikes before the learning neuron n_i^l is activated. The amplitude of the synaptic change is determined by two functions $W^d(s^d)$ and $W^l(s^l)$, where s^d is the temporal difference between the spike times of teacher neuron and input neuron, while s^l describes the difference between the spike times of learning neuron and input neuron. Figure redrawn from (Ponulak, 2005).

$(x(t), v(t))$ to train a readout function f, such that the actual outputs $f(x(t))$ are close to $v(t)$.

It was argued in (Natschläger et al., 2002) that the LSM has universal computational power. A very appealing feature of the applied training method, *i.e.* the readout function, is its simplicity, since only a single layer of weights is actually modified, for which a linear training method is sufficient.

A specific implementation of the readout, the so-called Remote Supervised Method (ReSuMe) introduced in (Ponulak, 2005), is presented here. The goal of ReSuMe is to impose a desired input-output spike pattern on a SNN, *i.e.* produce target spike trains in response to a certain input stimulus. The method is based on the already presented STDP learning window, *cf.* section 2.4.1 for details, in which two opposite update rules for the synaptic weights are balanced. Additional teacher neurons are defined for each synapse which remotely supervise the evolution of its synaptic weight. The teacher neuron is not explicitly connected to the network, but generates a reference spike signal which is used to update the connection weight in a STDP-like fashion. The post-synaptic neuron, whose activity is influenced by the weight update, is called the learning neuron.

Figure 2.15 illustrates the principle of ReSuMe. Let n_i^l denote the learning neuron which receives spike sequences from pre-synaptic neuron $n_k^{in}(i)$, the corresponding synaptic weight being w_{ki} and neuron $n^d(i)$ being the teacher for weight w_{ki}. If input neuron $n_k^{in}(i)$ emits a spike which is followed by a spike of the teacher neuron $n^d(i)$, the synaptic weight w_{ki} is increased. On the other hand, if $n_k^{in}(i)$ spikes before the learning neuron n_i^l is activated, the synaptic weight is decreased. The amplitude of the synaptic change is determined by two functions $W^d(s^d)$ and $W^l(s^l)$, where s^d is the temporal difference between the spike times of teacher neuron and input neuron, while s^l describes the difference between the spike times of learning neuron and input neuron. Thus, the precise time difference of spiking activity defines the strength of the synaptic change.

A few studies on LSM can be found, *cf. e.g.* the overview paper in (Natschläger, Markram, & Maass, 2003) and the specific case study for isolated word recognition in (Verstraeten et al., 2005). More information on ReSuMe is available in (Kasinski & Ponulak, 2005; Ponulak & Kasinski, 2006; Ponulak, 2008).

2.5 APPLICATIONS OF SNN

Traditionally, SNN have been applied in the area of neuroscience to better understand brain functions and principles, the work by Hodgkin and Huxley (Hodgkin & Huxley, 1952) being among the pioneering studies in the field. A number of main directions for understanding the functioning of the nervous system are given in (Carnevale & Hines, 2006). Here it is argued that a comprehensive knowledge about the anatomy of individual neurons and classes of cells, pathways, nuclei and higher levels of organisation is very important, along with detailed information about the pharmacology of ion channels, transmitters, modulators and receptors. Furthermore, it is crucial to understand the biochemistry and molecular biology of enzymes, growth factors, and genes that participate in brain development and maintenance, perception and behaviour, learning and diseases. A range of software systems for analysing biologically plausible neural models exist, NEURON[3] and GENESIS[4] being the most prominent ones. Modelling and simulation are fundamental for the understanding of neural processes.

A number of large-scale studies have been recently undertaken to understand the complex behaviour of ensembles of spiking neurons, *cf. e.g.* (Glackin, McGinnity,

3 Available at http://www.neuron.yale.edu/neuron
4 Available at http://genesis-sim.org

Maguire, Wu, & Belatreche, 2005; Izhikevich & Edelman, 2008). The review presented in (Maguire et al., 2007) discusses challenges for implementations of spiking neural networks on FPGAs in the context of large-scale experiments.

SNN are also applied in many real-world applications. Notable progress has been made in areas like speech recognition (Verstraeten et al., 2005), learning rules (Bohte et al., 2002), associative memory (Knoblauch, 2005), and function approximation (Iannella & Kindermann, 2005). Other applications include biologically more realistic controllers for autonomous robots, *cf.* (Floreano & Mattiussi, 2001; Floreano, Epars, Zufferey, & Mattiussi, 2006) and also (Wang, Hou, Zou, Tan, & Cheng, 2008) for some interesting examples in this research area.

In the next section we focus on a few applications of the evolving spiking neural network architecture, which is the main focus of this thesis.

2.6 EVOLVING SPIKING NEURAL NETWORK ARCHITECTURE

Based on Kasabov (2006), an evolving spiking neural network architecture (eSNN) was proposed in (Wysoski et al., 2006a) which was initially designed as a visual pattern recognition system. Other studies have utilised eSNN as a general classification method, *e.g.* in the context of classifying water and wine samples (Soltic et al., 2008). The method is based on the already discussed Thorpe neural model, in which the importance of early spikes (after the onset of a certain stimulus) is boosted, *cf.* section 2.2.5. Synaptic plasticity is employed by a fast supervised one-pass learning algorithm that is explained as part of this section.

In order to classify real-valued data sets, each data sample, *i.e.* a vector of real-valued elements, is mapped into a sequence of spikes using a certain neural encoding technique. In the context of eSNN, the so-called rank order population encoding is employed, but other encoding may be suitable as well. The topology of eSNN is strictly feed-forward and organised in several layers. Weight modification only occurs on the connections between the neurons of the output layer and the neurons of either hidden layer or the input layer.

In the next section, the encoding principle used in eSNN is presented, followed by the description of the one-pass learning method and the overall functioning of the eSNN method. Finally, a variety of applications based on the eSNN architecture is reviewed and summarised. A number of open problems that have not been solved in the existing literature, have motivated this PhD study and are outlined at the end of this review.

2.6.1 *Rank order population encoding*

Rank order population encoding is an extension of the rank order encoding introduced in (Thorpe & Gautrais, 1998). It allows the mapping of vectors of real-valued elements into a sequence of spikes. An implementation based on arrays of receptive fields is firstly described in (Bohte et al., 2002). Receptive fields allow the encoding of continuous values by using a collection of neurons with overlapping sensitivity profiles. Each input variable is encoded independently by a group of M one-dimensional receptive fields. For a variable n an interval $[I^n_{min}, I^n_{max}]$ is defined. The Gaussian receptive field of neuron i is given by its centre μ_i

$$\mu_i = I^n_{min} + \frac{2i-3}{2} \cdot \frac{I^n_{max} - I^n_{min}}{M-2} \tag{2.27}$$

and width σ:

$$\sigma = \frac{1}{\beta} \cdot \frac{I^n_{max} - I^n_{min}}{M-2} \tag{2.28}$$

with $1 \leq \beta \leq 2$. Parameter β directly controls the width of each Gaussian receptive field. Figure 2.16 depicts an example encoding of a single variable. For the diagram, $\beta = 2$ was used, the input interval $[I^n_{min}, I^n_{max}]$ was set to $[-1.5, 1.5]$, and $M = 5$ receptive fields were used.

More information on rank order coding strategies can be found in (Perrinet, Delorme, Samuelides, & Thorpe, 2001) and the accompanying article (Delorme et al., 2001). Very interesting is also the review on rapid spike-based processing strategies in the context of image recognition presented in (Thorpe, Delorme, & Rullen, 2001), where most work on the Thorpe neural model and rank order coding is summarised. Rank order coding was also explored for speech recognition problems (Loiselle, Rouat, Pressnitzer, & Thorpe, 2005) and is a core part of the eSNN architecture.

2.6.2 *One-pass learning*

The aim of the learning method is to create output neurons, each of them labeled with a certain class label $l \in L$. The number and value of class labels depends on the classification problem to solve, *i.e.* L corresponds to the set of class labels of the given data set. After presenting a certain input sample to the network, the corresponding spike train is propagated through the SNN which may result in the firing of certain output neurons. It is also possible that no output neuron is activated and the

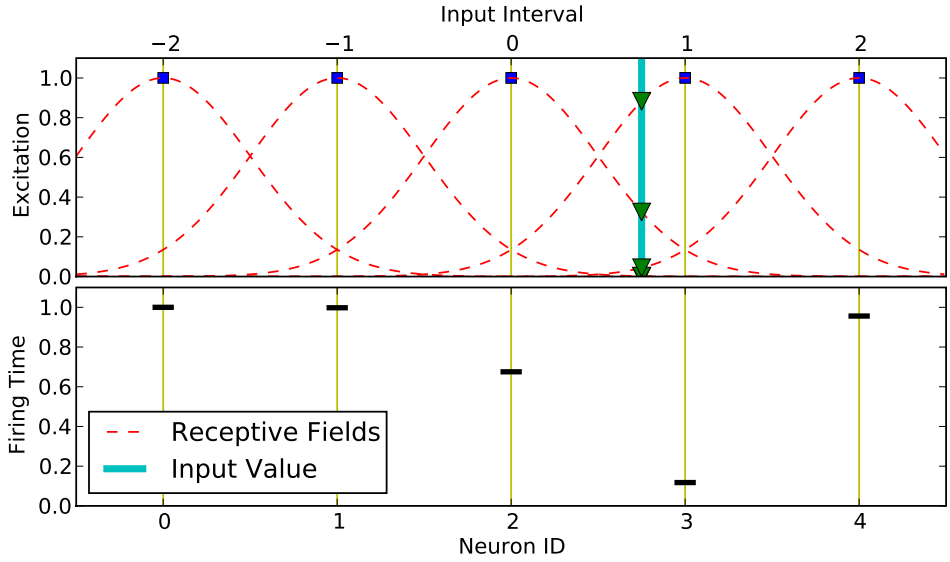


Figure 2.16: Population encoding based on Gaussian receptive fields. For an input value $v = 0.75$ (thick straight line in top figure) the intersection points with each Gaussian is computed (triangles), which are in turn translated into spike time delays (lower left figure).

Algorithm 1 Training an evolving spiking neural network (eSNN)

Require: m_l, s_l, c_l for a class label $l \in L$

initialise neuron repository $R_l = \{\}$
for all samples $X^{(i)}$ belonging to class l **do**
  $w_j^{(i)} \leftarrow (m_l)^{order(j)}, \quad \forall\, j \mid j$ pre-synaptic neuron of i
  $u_{max}^{(i)} \leftarrow \sum_j w_j^{(i)} (m_l)^{order(j)}$
  $\vartheta^{(i)} \leftarrow c_l u_{max}^{(i)}$
  if $\min(d(w^{(i)}, w^{(k)})) < s_l, \quad w^{(k)} \in R_l$ **then**
    $w^{(k)} \leftarrow$ merge $w^{(i)}$and $w^{(k)}$ according to Equation 2.32
    $\vartheta^{(k)} \leftarrow$ merge $\vartheta^{(i)}$and $\vartheta^{(k)}$ according to Equation 2.33
  else
    $R_l \leftarrow R_l \cup \{w^{(i)}\}$
  end if
end for

network remains silent. In this case, the classification result is undetermined. If one or more output neurons have emitted a spike, the neuron with the shortest response time among all activated output neurons is determined, *i.e.* the output neuron with the earliest spike time. The label of this neuron represents the classification result for the presented input sample.

The learning algorithm successively creates a repository of trained output neurons during the presentation of training samples. For each class label $l \in L$ an individual repository is evolved. The procedure is described in detail in Algorithm 1. For each training sample i with class label $l \in L$ a new output neuron is created and fully connected to the previous layer of neurons resulting in a real-valued weight vector $w^{(i)}$, with $w_j^{(i)} \in \mathbb{R}$ denoting the connection between the pre-synaptic neuron j and the created neuron i. In the next step, the input spikes are propagated through the network and the value of weight $w_j^{(i)}$ is computed according to the *order* of spike transmission through a synapse j, *cf.* line 3 in Algorithm 1:

$$w_j^{(i)} = (m_l)^{order(j)}, \quad \forall\, j \mid j \text{ pre-synaptic neuron of } i \tag{2.29}$$

Parameter m_l is the modulation factor of the Thorpe neural model. Differently labeled output neurons may have different modulation factors m_l. Function $order(j)$ represents the rank of the spike emitted by neuron j. For example, a rank $order(j) = 0$ would be assigned, if neuron j is the first among all pre-synaptic neurons of i that emits a spike. In a similar fashion the spikes of all pre-synaptic neurons are ranked and then used in the computation of the weights.

The firing threshold $\vartheta^{(i)}$ of the created neuron i is defined as the fraction $c_l \in \mathbb{R}$, $0 < c_l < 1$, of the maximal possible potential $u_{max}^{(i)}$, *cf.* lines 4 and 5 in Algorithm 1:

$$\vartheta^{(i)} = c_l u_{max}^{(i)} \tag{2.30}$$

$$u_{max}^{(i)} = \sum_j w_j^{(i)} (m_l)^{order(j)} \tag{2.31}$$

The fraction c_l is a parameter of the model and for each class label $l \in L$ a different fraction can be specified.

The weight vector of the trained neuron is then compared to the ones of neurons that are already stored neurons in the repository, *cf.* line 6 in Algorithm 1. If the minimal Euclidean distance between the weight vectors of the neuron i and an existing neuron k is smaller than a specified similarity threshold s_l, the two neurons are considered too "similar" and both the firing thresholds and the weight vectors are merged according to:

$$w_j^{(k)} \leftarrow \frac{w_j^{(i)} + N w_j^{(k)}}{1+N}, \quad \forall j \mid j \text{ pre-synaptic neuron of } i \tag{2.32}$$

$$\vartheta^{(k)} \leftarrow \frac{\vartheta^{(i)} + N \vartheta^{(k)}}{1+N} \tag{2.33}$$

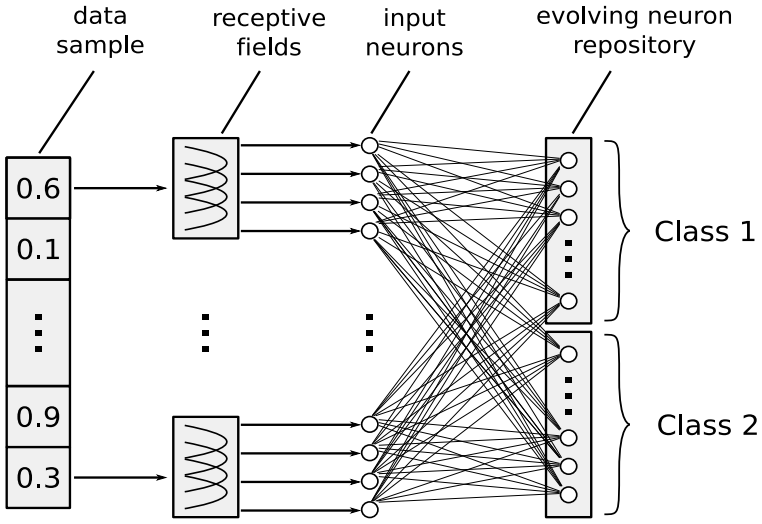


Figure 2.17: Schematic illustration of the evolving spiking neural network architecture (eSNN). Real-valued vector elements are mapped into the time domain using rank order population encoding based on Gaussian receptive fields. As a consequence of this transformation input neurons emit spikes at pre-defined firing times, invoking the one-pass learning algorithm of the eSNN. The learning iteratively creates repositories of output neurons, one repository for each class. Here a two-class problem is presented. Due to the evolving nature of the network, it is possible to accumulate knowledge as it becomes available, without the requirement of re-training with already learnt samples.

Integer N denotes the number of samples previously used to update neuron k. The merging is implemented as the (running) average of the connection weights, and the (running) average of the two firing thresholds. After the merging, the trained neuron i is discarded and the next sample processed. If no other neuron in the repository is similar to the trained neuron i, the neuron i is added to the repository as a new output neuron.

Figure 2.17 depicts the eSNN architecture. Due to the incremental evolution of output neurons, it is possible to accumulate knowledge as it becomes available. Hence, a trained network is able to learn new data without the need of re-training on already learnt samples. Real-world applications of the eSNN architecture are discussed in the next section.

2.6.3 *Applications*

The eSNN architecture is used in a variety of applications that are described and summarised here.

Visual pattern recognition

Among the earliest application of the eSNN is the visual pattern recognition system presented in (Wysoski et al., 2006a) which extends the work of (Delorme, Gautrais, VanRullen, & Thorpe, 1999; Delorme & Thorpe, 2001) by including the on-line learning technique described above. In (Wysoski et al., 2006a) and (Wysoski et al., 2006b) the method was studied on an image data set consisting of 400 faces of 40 different persons. The task here was to predict the class labels of presented images correctly. The system was trained on a subset of the data and then tested on the remaining samples of the data. Classification results were similar to (Delorme et al., 1999; Delorme & Thorpe, 2001) with the additional advantages of the novel on-line learning method.

In a later study another processing layer was added to the system which allows efficient multi-view visual pattern recognition (Wysoski et al., 2008b). The additional layer accumulates information over several different views of an image in order to reach a final decision about the associated class label of the frames. Thus, it is possible to perform an efficient on-line person authentication through the presentation of a short video clip to the system, although the audio information was ignored in this study.

The main principle of this image recognition method is briefly outlined here. The neural network is composed of four layers of Thorpe neurons, each of them grouping a set of neurons into several two-dimensional maps, so-called neural maps. Information in this network is propagated in a feed-forward manner, *i.e.* no recurrent connections exist. An input frame in form of a grey-scale image is fed into the first neural layer (L_1), each pixel of the image corresponding to one neuron in a neural map of L_1. Several neural maps may exist in this layer. The map consists of "On" and "Off" neurons that are responsible for the enhancement of the high contrast parts of the image. Each map is configured differently and thus is sensitive to different grey scales in the image. The output of this layer is transformed into the spike domain using rank order encoding as described in (Thorpe & Gautrais, 1998). As a consequence of this encoding, pixels with higher contrast are prioritised in the neural processing.

The second layer, denoted L_2, consists of orientation maps. Each map is selective for different directions, *e.g.* $0°, 45°, \ldots, 315°$, and is implemented by appropriately parametrised Gabor functions. It is noted that the first two layers are passive filters that are not subject to any learning process. In the third layer, L_3, the learning occurs using the one-pass learning method described in section 2.6.2. Here neural maps

are created and merged according to the rules of the learning algorithm. Finally, the fourth layer, L_4, consists of a single neuron for each output class, which accumulates opinions about the class label of a certain sequence of input frames. The weights between L_3 and L_4 are fixed to a constant value, usually 1, and are not subject to learning. The first L_4 neuron that is activated by the presented stimuli determines the classification result for the input. After the activation of an L_4 neuron the system stops.

Experimental evidence about the suitability of this pattern recognition system is provided in (Wysoski et al., 2008b) along with a comparison to other typical classification methods.

Auditory pattern recognition

A similar network, but in an entirely different context, was investigated in (Wysoski et al., 2007), where a text-independent speaker authentication system is presented. The classification task in this work consisted of the correct labelling of audio streams presented to the system.

Speech signals are split into temporal frames, each containing a signal segment over a short time period. The frames are first pre-processed using the Mel Frequency Cepstrum Coefficients (MFCC) (Rabiner & Juang, 1993) and then used to invoke the eSNN. The MFCC frame is transformed into the spike domain using rank order encoding (Thorpe & Gautrais, 1998) and the resulting stimulus is propagated to the first layer of neurons. This layer, denoted L_1, contains two neural ensembles representing the speaker and the background model, respectively. While the former model is trained on the voice of a certain speaker, the latter one is trained on the background noise of the audio stream. This system also collects opinions about the class label of the presented sequence of input frames, which is implemented by the second layer of the network. Layer L_2 consists of only two neurons, each of which accumulates information about whether a given frame corresponds to a certain speaker or to the background noise. Whenever an L_2 neuron is activated, the simulation of the network stops and the classification output is presented.

Audio-visual pattern recognition

The two recognition systems presented above were successfully combined, forming an audio-visual pattern recognition method. Both systems are trained individually, but their output is propagated to an additional supra-modal layer. The supra-modal layer integrates incoming sensory information from individual modalities and cross-

modal connections enable the influence of one modality upon the other. A detailed discussion of this system along with experimental evidence is given in (Wysoski et al., 2008a) and in the PhD dissertation of Simei Wysoski in (Wysoski, 2008).

Taste recognition

The last application of eSNN being discussed here investigates the use of a SNN for taste recognition in a gustatory model. The classification performance of eSNN was experimentally explored based on water and wine samples collected from (de Sousa & Riul Jr., 2002) and (Riul et al., 2004). The topology of the model consists of two layers. The first layer receives an input stimulus obtained from the mapping of a real-valued data sample into spike trains using a rank order population encoding, *cf.* section 2.6.1. The weights from the first neural layer are subject to training according to the already discussed one-pass learning method. Finally, the output of the second neural layer determines the class label of the presented input stimulus.

The method was investigated in a number of scenarios, where the size of the data sets and the number of class labels was varied. Generally, eSNN reported promising results on both large and small data sets, which has motivated an FPGA hardware implementation of the system (Zuppicich & Soltic, 2009).

2.6.4 *Open problems*

The eSNN architecture requires the appropriate setting of a number of neural and learning parameters in order to achieve satisfying classification results. Their configuration can quickly become a challenging task, since it usually requires comprehensive knowledge about the influence of each parameter. Some parameters might be linked to each other and should not be chosen independently. For example, modifying the modulation factor of the Thorpe neural model should also involve the careful choice of the firing threshold. A small modulation factor increases the sensitivity of the neuron to the input significantly, thus the threshold has to be adapted accordingly to prevent the neuron from becoming over-specialised for a certain input. The situation becomes even more complicated in the context of many class labels, since then the number of parameters increases linearly with the number of classes.

All the above presented applications require a careful manual tuning of the eSNN parameters. The problem of this approach was explicitly recognised and mentioned in a number of studies, *e.g.* in (Wysoski et al., 2008a; Wysoski, 2008; Soltic et al., 2008), suggesting an automatic optimisation of the involved parameters. Self-

adapting parameters require less expert knowledge in order to configure eSNN for a specific task, which promotes its straightforward application to other problem domains.

Another issue with the eSNN classification method is the lack of an explicit feature selection mechanism, although such an extension might not be crucial for the applications presented above. For example, the taste recognition system involves only a limited number of features, *i.e.* seven taste sensors, which are all assumed to be relevant for the given classification task. For the visual pattern recognition systems on the other hand, a certain feature selection was implemented through the contrast detection of the "on"/"off"-neurons and the orientation maps in the first and second neural layer of the network respectively. Although this principle works well in the context of a visual stimulus, it is not suitable for many other problem domains. Thus, a general feature selection component as part of the eSNN method might be beneficial when applying the method to many other problems.

2.7 SURVEYING HETEROGENEOUS OPTIMISATION METHODS

Both issues described above have motivated the work presented in this thesis, which involves the development of an integrated feature and parameter optimisation component for eSNN. As outlined in the introductory chapter, such an automatic optimisation requires the application of a generic heterogeneous optimisation method. A specialised EA evolves a combined solution consisting of a binary and a real-valued sub-component, which represent a feature subset and an eSNN parameter configuration respectively. In this section a survey of previously proposed heterogeneous algorithms is presented, with the aim to determine which EA is most suitable for the intended extension.

The concept of a simultaneous exploration of heterogeneous search spaces is not new. Numerous studies have discussed such schemes, especially in the context of the simultaneous evolution of the weight matrix and topology of neural networks. Among the earliest contributions to this area is the work by (Hintz & Spofford, 1990). A genetic algorithm is used that operates on a binary chromosome, which is semantically structured in sub-components in order to allow the encoding of connectivity and connection weights in a single bit string. Promising results have been reported on a 9×9 bit character recognition problem. A similar approach was investigated

in (Maniezzo, 1994), where an algorithm called ANNA ELEONORA[5] is presented. Here the presence or absence of a connection between two neurons is encoded by a connectivity bit, followed by a number of additional bits representing the corresponding connection weight. Due to the binary representation of the weights, a conversion from bit strings into real values is required. The granularity of the weights, *i.e.* the number of bits used for encoding a single weight, is adapted as part of the evolutionary process. Since the interpretation of the bits in the chromosome is not homogeneous, a set of complex crossover and mutation operators is defined. The method was later further developed in (Leung, Lam, Ling, & Tam, 2003), in which the binary chromosome was replaced by a continuous one. Although the real value representation seems appropriate for the evolution of connection weights, it is less suitable for the representation of the connectivity bit. Similar genetic approaches were discussed in (White & Ligomenides, 1993), (Alba, Montes, & Troya, 1993) and (Oliker, Furst, & Maimon, 1993).

All of the above studies employ evolutionary algorithms to explore a heterogeneous search space using either a binary or a continuous representation of the chromosome. As a consequence these methods are not optimally adapted for the exploration of either the continuous or the binary sub-component of a candidate solution. In (Valko, Marques, & Castelani, 2005) this issue was explicitly addressed by proposing a method called FeaSANNT (Feature Selection and Artificial Neural Network Training). FeaSANNT is a genetic algorithm using a binary representation for the evolution of appropriate feature subsets and a continuous representation for the optimisation of the weight matrix of the neural net. The method is discussed in greater detail in (Castellani & Marques, 2008). For each representation individual genetic operators are implemented. A standard two-point crossover along with bit-flip mutation is used for the binary landscape, while uniform random mutations and Lamarckian learning using back-propagation are applied to the variables of the continuous solution part. A practical application of FeaSANNT on a wood veneer classification problem can be found in (Castellani & Rowlands, 2009). Further efforts have been made to also evolve the network topology, *cf.* (Castellani, 2006), which required the definition of additional operators, such as node deletion and insertion according to some user-specified probability parameters.

Very recent studies follow similar trends. The chromosome in (Rivero, Dorado, Fernández-Blanco, & Pazos, 2009) consists of the concatenation of three parts: A connectivity bit encoding the presence or absence of a connection, a real-valued

5 Abbreviation for **A**rtificial **N**eural **N**etworks **A**daptation: **E**volutionary **Le**arning **o**f **N**eural **O**ptimal **R**unning **A**bilities

weight, and another bit representing the presence or absence of a particular hidden neuron. A genetic algorithm is used but the actual genetic operators are unfortunately not reported in the article. A modified version of a Particle Swarm Optimiser is proposed in (Garro, Sossa, & Vazquez, 2009) that also evolves the neural transfer function in addition to topology and connection weights.

The methods discussed above are all designed with the clear aim to apply them in the context of topology and weight optimisation of neural networks, which prevents their straightforward application in a different context. For example, most of them employ network specific genetic operators like back-propagation to drive the search, which may not be available if the context of the problem changes.

Only very few general mixed-variable algorithms exist. Arguably among the most promising algorithms on heterogeneous optimisation is the Mixed Bayesian Optimisation Algorithm (MBOA) introduced in (Ocenasek & Schwarz, 2002). In MBOA, a set of decision trees that are iteratively constructed and adapted during the evolutionary process, explore the search space in a probabilistic fashion. New solution candidates are sampled according to the current state of the trees. Although MBOA was not extensively investigated on heterogeneous problems, promising results have been obtained on binary benchmark problems. The continuous optimisation performance of MBOA, on the other hand, is less competitive as experimentally demonstrated in (Kern et al., 2004). Furthermore, the method involves a significant computational overhead, which has motivated a multi-threaded implementation on parallel hardware (Ocenasek, 2002).

Other directions have suggested the use of different EA variants. A heterogeneous version of an Ant Colony Optimisation (ACO) algorithm was proposed in (Socha, 2004). Due to the lack of comparison algorithms, the authors have experimentally investigated the performance of the method using a number of continuous benchmark functions. Thus, the suitability of this ACO on mixed-variable problems is less clear. However, it is interesting to note that the principle idea of ACO is also based on a probabilistic exploration of the search space, as shown in (Cordón, Fernández de Viana, Herrera, & Moreno, 2000) and (Monmarché, Ramat, Dromel, Slimane, & Venturini, 1999). This is very similar to the above-mentioned MBOA, despite the very different metaphor employed in ACO. While ACO assumes a population of “ants”, each of them iteratively constructing a solution according to discrete or continuous probability distributions, MBOA emphasises on an entirely mathematical description of its working.

A so-called Bell-Curve Genetic Algorithm (BCGA) is discussed in (Kincaid, Griffith, Sykes, & Sobieszczanski-Sobieski, 2004), which is an optimisation heuristic

similar to Evolutionary Strategies (ES) (Schwefel, 1981). Here the term bell-curve refers to the Gaussian probability density function, which is employed to sample new solution candidates. The analysis of BCGA on a very specialised problem domain, *i.e.* hub structures found in buildings, complicates a comparison of this approach to other methods and allows only speculations about its suitability in different problem scenarios.

2.7.1 *What EA to choose?*

When summarising the presented survey on mixed-variable optimisation methods, several conclusions can be made. First of all, the exploration of heterogeneous search spaces is feasible and was implemented in numerous algorithms. However, few of them are suitable for an application to general heterogeneous optimisation problems. Either the representation of the search space is non-optimal, *i.e.* binary-only or continuous-only representations, or the optimisation algorithm is too problem specific, *e.g.* its application aims explicitly towards topology and weight optimisation of neural networks. Furthermore, although some general purpose mixed-variable optimisers have been developed recently, none of them was studied thoroughly on heterogeneous problems. Thus, it is argued in this thesis, that a novel generic optimisation technique is required, that is applicable to general, domain-independent problems.

As mentioned earlier, the most promising mixed-variable algorithms employ a probabilistic model to explore the search space, although each method was proposed with an entirely different metaphor in mind. Taking this observation into account, a new optimisation method is proposed in the next chapter that is also based on the evolution of probabilistic models to identify promising areas in the solution space of a problem.

Chapter 3

OPTIMISING BINARY SEARCH SPACES – A VERSATILE QUANTUM-INSPIRED EVOLUTIONARY ALGORITHM

Quantum-Inspired Evolutionary Algorithms (QEA) apply Quantum Computing Principles to enhance classical Evolutionary Algorithms (EA). In the last ten years of QEA research, investigators demonstrated promising benefits compared to classical EA on solving complex benchmark problems in the fields of combinatorial (Han & Kim, 2002), numerical (Han & Kim, 2004; da Cruz, Vellasco, & Pacheco, 2006) and multi-objective optimisation (Talbi, Draa, & Batouche, 2006). Others addressed real world problems including disk allocation (Kima, Hwang, Han, Kim, & Park, 2003), face detection (Jang, Han, & Kim, 2004), rigid image registration (Draa, Batouche, & Talbi, 2004a), training of multi-layer perceptrons (Venayagamoorthy & Singhal, 2005), signal processing (F. Liu, Li, Liang, & Hu, 2006) and clustering of gene expression data (W. Zhou, Zhou, Huang, & Wang, 2005). However, despite this work, ambiguity in the definition of QEA hampered its understanding and integration into the theory of Evolutionary Computation.

Arguably, the most illustrative example of QEA is the algorithm first proposed by Han and Kim (2002), in which they used some major principles of Quantum Computing such as the quantum and collapsed bit, the linear superposition of states and the quantum rotation gate. This algorithm has been investigated both experimentally and theoretically in numerous studies. Classical optimisation benchmarks were considered by Han and Kim (2002), while Kima et al. (2003) and Jang et al. (2004) applied QEA to some real world problems. Han and Kim (2003) suggested some practical guidelines on configuring QEA, analysing the role and impact of the involved parameters on the functioning of the method. An alternative update operator, namely the H_ϵ gate, was considered in (Han & Kim, 2004). Han and Kim (2006) presented a proof of convergence towards the global optimum on a one dimensional One Max problem. Han (2003) comprehensively discussed and studied QEA, presenting most of the prior work.

Nevertheless, some specific characteristics of QEA remain unexamined. Section 3.1 briefly outlines some basic quantum principles that have inspired QEA, and formulates a revised description of its features. Exploring the dynamics of QEA reveals a clear trend in promoting the phenomenon of hitch-hiking. Section 3.2 introduces a novel algorithm called Versatile Quantum-inspired Evolutionary Algorithm (vQEA), in which a simple but critical mechanism is proposed to avoid problems encountered in the original QEA. With vQEA, the information about the search space collected during evolution is continuously renewed and shared among the whole population rather than being kept at the individual level. In section 3.3, vQEA is tested on different benchmark problems and compared to classical versions of EA, namely a genetic algorithm (GA) (Goldberg, 1989) and the original QEA. Finally, section 3.4 discusses vQEA in the light of Estimation of Distribution Algorithms (EDA).

3.1 PRINCIPLES OF QUANTUM-INSPIRED EVOLUTIONARY ALGORITHMS

A quantum bit or $\mathcal{Q}$bit (Hey, 1999) is the smallest unit of information in a quantum computer. A $\mathcal{Q}$bit is defined by its state $|\Psi\rangle$:

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.1}$$

where α and β are complex numbers defining probabilities at which the corresponding state is likely to appear when a $\mathcal{Q}$bit is *collapsed*, *i.e.* read or measured. Here the probability of a $\mathcal{Q}$bit to collapse to state "0" and "1" is $|\alpha|^2$ and $|\beta|^2$ respectively[1]. In a more geometrical aspect, a $\mathcal{Q}$bit state can also be defined by θ such that $\cos(\theta) = |\alpha|$ and $\sin(\theta) = |\beta|$.

In order to modify the probability amplitudes α and β, *quantum gates* can be applied. We note that several quantum gates have been proposed such as (controlled) NOT-gate, rotation gate and Hadamard gate, see (Hey, 1999) for details.

3.1.1 *Description of the QEA*

In this section we propose a revised description of the QEA, originally published in (Han & Kim, 2002). See (Han, 2003) for a comprehensive definition. QEA is a generational, population-based search method whose behaviour can be decomposed in three different and interacting levels, *cf.* Figure 3.1.

1 Normalisation of the states to unity guarantees $|\alpha|^2 + |\beta|^2 = 1$ at any time.

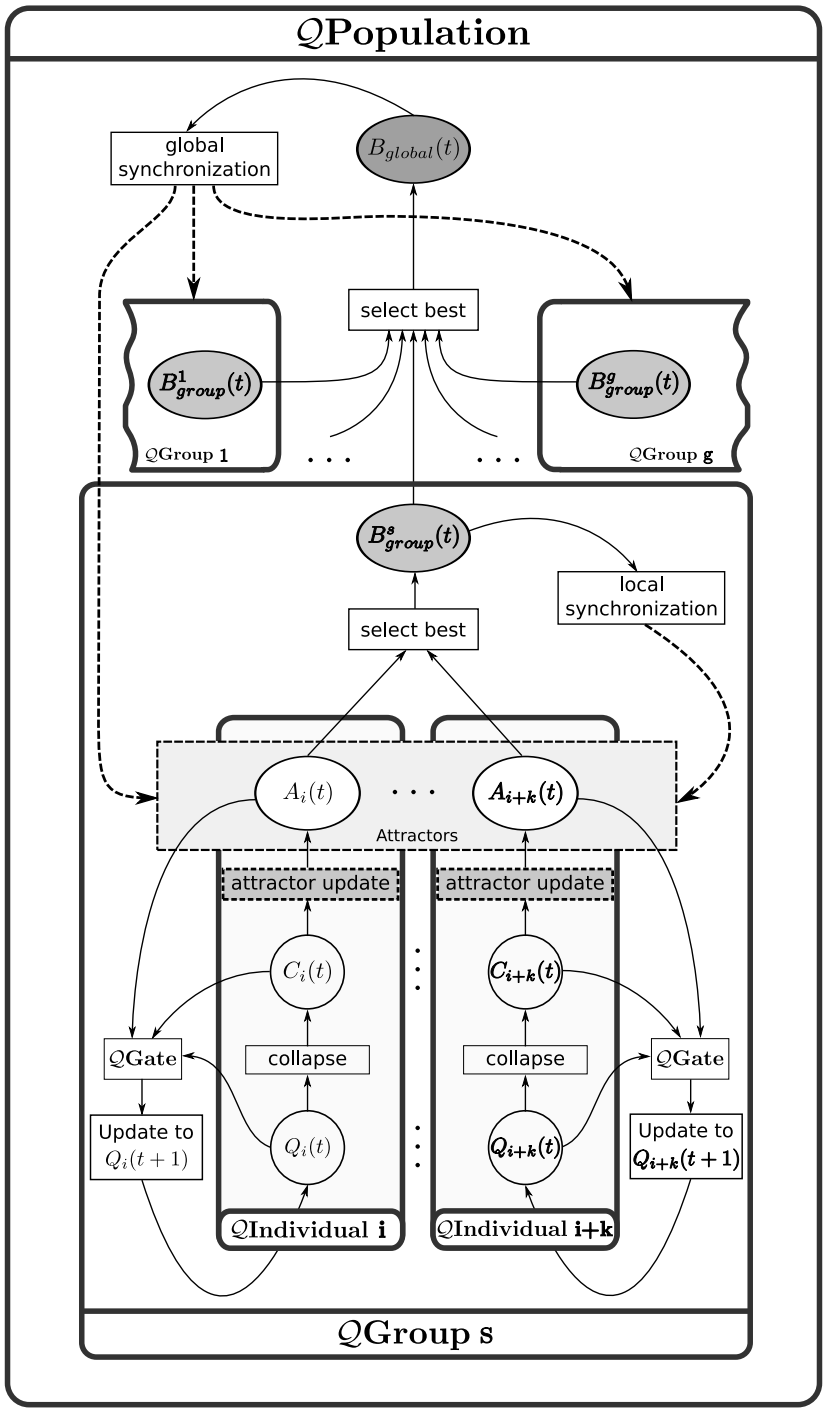


Figure 3.1: Description of QEA with three levels

QUANTUM INDIVIDUALS The lowest level corresponds to *quantum individuals*[2] . A $\mathcal{Q}$individual i at generation t contains a $\mathcal{Q}$bit string $Q_i(t)$ and two binary

2 Note that the original notation of Han and Kim has been slightly revised here. An individual here is composed of a $\mathcal{Q}$bit string and two binary strings rather than the $\mathcal{Q}$bit string only. The revised notation of QEA allows a more structured and compact description of the method.

strings $C_i(t)$ and $A_i(t)$. More precisely Q_i corresponds to a string of N concatenated $\mathcal{Q}$bits:

$$Q_i = Q_i^1 Q_i^2 \dots Q_i^{\mathrm{N}} = \begin{bmatrix} \alpha_i^1 & \alpha_i^2 & \dots & \alpha_i^{\mathrm{N}} \\ \beta_i^1 & \beta_i^2 & \dots & \beta_i^{\mathrm{N}} \end{bmatrix} \tag{3.2}$$

For the purpose of fitness evaluation each Q_i is first sampled (or collapsed) to form a bit string C_i. Each $\mathcal{Q}$bit in Q_i is sampled according to a probability defined by $|\beta_i^j|^2$, so that C_i represents a configuration in the search space and its quality can be classically determined using a fitness function f. In the sense of EA, Q_i is the genotype while C_i is the phenotype of a given individual. We will show later that in the sense of EDAs, Q_i defines a probabilistic model

$$\mathcal{P}_i = \left[|\beta_i^1|^2 \dots |\beta_i^{\mathrm{N}}|^2\right]$$

while C_i is a realisation of this model.

A solution A_i is attached to each individual i acting as an attractor for Q_i. Every generation, C_i and A_i are compared in terms of both fitness and bit values. If A_i is better than C_i (*i.e.* $f(A_i) > f(C_i)$ in a maximisation problem) and if their bit values differ, a quantum gate operator is applied on the corresponding $\mathcal{Q}$bits of Q_i. Thus the probabilistic model P_i defined by Q_i is moved slightly towards the attractor A_i. The attractor A_i is replaced by C_i whenever C_i is better in terms of fitness. More specifically, if $f(A_i) \leq f(C_i)$ (assuming a maximisation problem), no update of the probabilistic model occurs, but the attractor A_i is replaced by C_i.

In classical EA, variation operators like crossover or mutation operations are used to explore the search space. The quantum analogue for these operators is called a quantum gate. In this study, the rotation gate is used to modify the $\mathcal{Q}$bits. The j^{th} $\mathcal{Q}$bit at generation t of Q_i is updated as follows:

$$\begin{bmatrix} \alpha_i^j(t+1) \\ \beta_i^j(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \alpha_i^j(t) \\ \beta_i^j(t) \end{bmatrix} \tag{3.3}$$

where the constant $\Delta\theta$ is a rotation angle designed in compliance with the application problem (Han & Kim, 2003). We note that the sign of $\Delta\theta$ determines the direction of rotation (clockwise for negative values). In this study the application of the rotation gate operator is limited in order to keep θ in the range $[0, \pi/2]$.

QUANTUM GROUPS The second level corresponds to *quantum groups*. The population is divided into g $\mathcal{Q}$groups each containing k $\mathcal{Q}$individuals with the ability to synchronise their attractors. The best attractor (in terms of fitness) of a group, denoted B_{group}, is stored at every generation and is periodically distributed to the group attractors. The parameter S_{local} controls the frequency of local synchronisation events.

QUANTUM POPULATION The set of all $p = g \times k$ $\mathcal{Q}$individuals forms the *quantum population* and defines the topmost level of QEA. As for the $\mathcal{Q}$groups, the individuals of the $\mathcal{Q}$population can synchronise their attractors, too. The best attractor (in terms of fitness) among all $\mathcal{Q}$groups, noted B_{global}, is stored every generation and is periodically distributed to the group attractors. The frequency of global synchronisation events is controlled by a parameter S_{global}. We note that in the initial population all the $\mathcal{Q}$bits are fixed with $|\alpha|^2 = |\beta|^2 = 0.5$, such that the two states "0" and "1" are equi-probable in collapsed individuals.

3.1.2 *QEA on the One Max problem*

The One Max problem consists of maximising the number of ones of a bit string and the global optimum is denoted as 1^λ. In this section the behaviour of QEA on the One Max problem is studied for $\lambda = 100$. For that purpose new tools for monitoring the dynamics of both $\mathcal{Q}$individuals and $\mathcal{Q}$bits are used. The setting of the evolutionary parameters is similar to the settings proposed in (Han & Kim, 2002), with a population of 10 individuals, 5 groups, $\Delta\theta = 0.01\pi$, $S_{local} = 1$ and $S_{global} = 100$.

Figure 3.2 presents the evolution of the 100 $\mathcal{Q}$bits of $\mathcal{Q}$individual Q_4, on the One Max problem. Each point $Q_4^j(t)$ corresponds to a given $\mathcal{Q}$bit j and a given generation t. The colour indicates the value of the corresponding $|\beta|^2$: From black for $|\beta|^2 = 0$ to white for $|\beta|^2 = 1.0$. Thus, a $\mathcal{Q}$individual with all $\mathcal{Q}$bits being $|\beta|^2 \simeq 1$ (and as a consequence $|\alpha|^2 \simeq 0$) is likely to collapse into the global optimum 1^λ. We note that the evolutionary process starts by construction with initial values $|\beta|^2 = 0.5$. Most of the $\mathcal{Q}$bits evolve toward the optimum as the color changes to white. Nevertheless we can clearly see that some $\mathcal{Q}$bits are rotated towards the wrong direction as some very dark points appear. The vast majority of them, finally moves toward the expected value with $|\beta|^2$ close to 1, but one of them, namely Q_4^{75}, has converged to $|\beta|^2 \approx 0$. For this run, QEA was not able to find the global optimum in 500 generations.

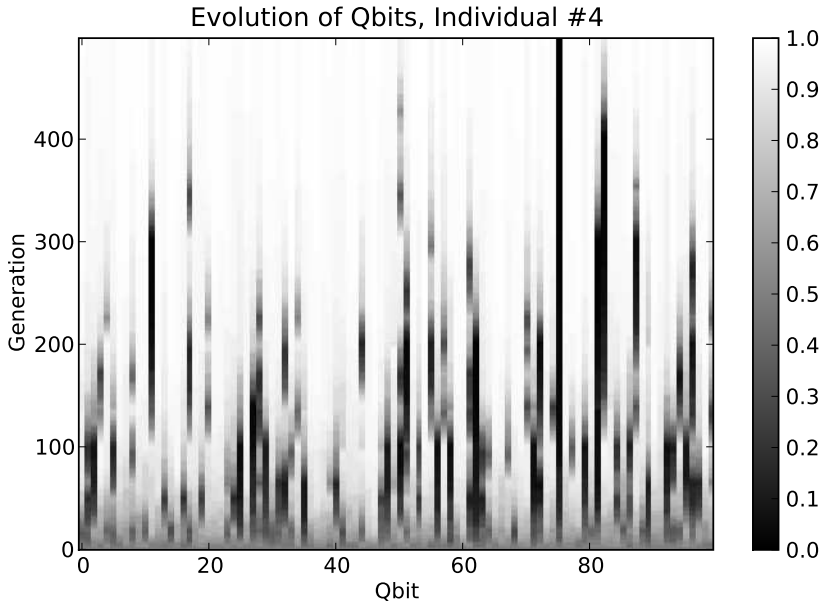


Figure 3.2: Typical evolution of a $\mathcal{Q}$bit string using QEA on the One Max problem. The shades indicate the value of $|\beta|^2$ for each of the 100 $\mathcal{Q}$bits at a given generation. The global optimum of the problem is a bit string in which all bits are "1". Thus, ideally all qbits should converge towards 1, but in this run qbit Q_4^{75} has converged early towards 0 instead.

To understand the inappropriate behaviour of Q_4^{75}, we have plotted its evolution, *i.e.* values of $|\alpha|^2$ (dotted line) and $|\beta|^2$ (solid line), as well as the states of the corresponding collapsed bit C_4^{75} and attractor bit A_4^{75}, *cf.* Figure 3.3. We see that $|\beta|^2$ converges towards 0 from the first generations driving the state of the collapsed bit to 0. We also note that the state of the attractor bit demonstrates very few variations and is nearly always 0, except for a very short period before generation 50. An attractor is always chosen according to its fitness. So the attractor A_4 is always better than the collapsed bit string C_4 even if the value of its 75^{th} bit is not well adapted.

3.1.3 *Hitch-hiking and the irreversible choice*

A quantum individual i explores a search space through sampling (collapsing) its $\mathcal{Q}$bit string Q_i. If the individual has identified a promising solution, it is chosen as an attractor and the exploration will concentrate on this new area. Generally, there are only two ways to update an attractor A_i. First, a better solution C_i is sampled, and as a consequence it replaces the current attractor or, alternatively, the attractor is replaced due to a synchronisation event. In this case, the new attractor was sampled

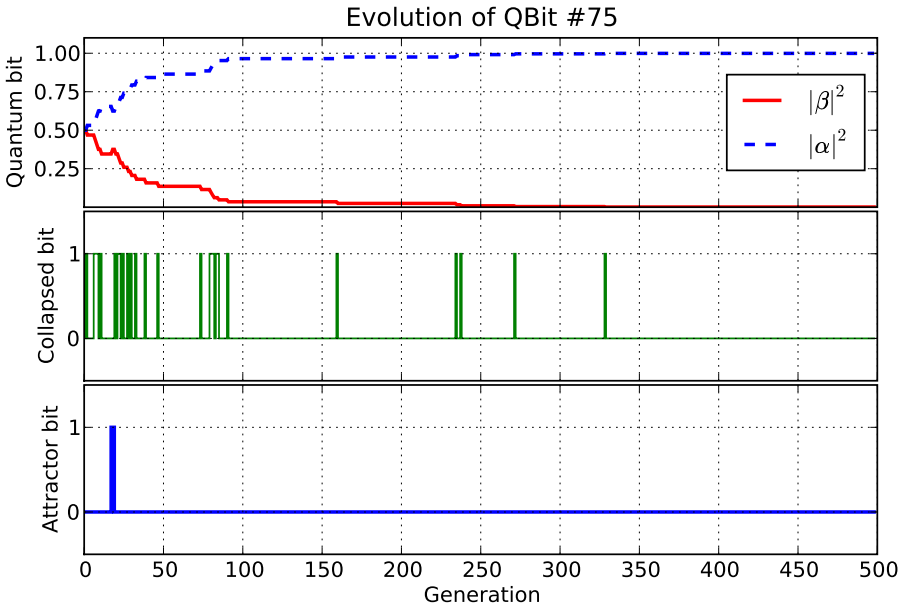


Figure 3.3: The evolution of the components of a problematic $\mathcal{Q}$individual with QEA on the One Max problem: $\mathcal{Q}$bit state (top), collapsed bit (middle) and attractor bit (bottom).

from the $\mathcal{Q}$bit string of a different individual. That means that in all generations in which no attractor update occurs, the $\mathcal{Q}$bits of Q_i are moved slightly towards A_i. As a consequence, Q_i may prematurely converge towards A_i, *i.e.* if there is no better solution found in time. Then the individual i is trapped, due to its inability to sample new solutions from Q_i. The only opportunity for the individual to escape from this attractor is a synchronisation phase which replaces its attractor with a different one produced elsewhere. Otherwise, it is possible that the choice of a very good but not optimal attractor is irreversible.

The issue of QEA described here is similar to a well-known problem occurring in classical genetic algorithms (CGA). The so called *hitch-hiking* phenomenon was first described as a serious bottleneck for CGA in (Forrest & Mitchell, 1992). Hitch-hiking corresponds to the increase in frequency of a “bad” allele at a given locus in the population due to the presence of nearby highly fit alleles on the same chromosomes (Futuyma, 1998). As a consequence, the potentially better alleles at the same locus (as the hitch-hiking allele) tend to disappear in the population and there is no way for the evolutionary process to retrieve them. In CGA, random mutation and uniform crossover are two known remedies against hitch-hiking. For QEA no such counter-measure exists.

3.2 VERSATILE QUANTUM-INSPIRED EVOLUTIONARY ALGORITHM

In this section an improved version of QEA is introduced called the Versatile Quantum-inspired Evolutionary Algorithm (vQEA). It aims to avoid the issues reported above. This algorithm was published in (Defoin-Platel, Schliebs, & Kasabov, 2007).

3.2.1 *Description of vQEA*

In order to prevent the case of irreversible choice and the hitch-hiking phenomenon, the strategy for updating attractors is modified. We introduce a new parameter that is used to modify the strategy: *Elitism*. In the original QEA, the update procedure (called "attractor update" in Figure 3.1) applies elitism: an attractor A_i is *only* replaced by C_i, if C_i is better. With vQEA this parameter is simply switched off. Therefore, the attractors are replaced at every generation without considering their fitness and thus attractors demonstrate a high degree of volatility. Moreover, to ensure the convergence of vQEA, the global synchronisation is also performed every generation in such a way that all the attractors are identical, *i.e.* the attractor at generation $t+1$ corresponds to the best solution found at generation t.

We note that, with such a setting, the group size n and local synchronisation parameters S_{local} do not affect the algorithm anymore. With vQEA, the information about the search space collected during evolution is not kept at the individual level, but continuously renewed and shared among the whole population[3]. Nevertheless, the concept of quantum groups, which is similar to demes in classical EA, is interesting and thus it is not intended to remove it. In this study, however, we avoid the tuning of n and S_{local} and concentrate on the effects of removing elitism from QEA. Thus the simplified sequential procedure of vQEA is detailed in Algorithm 2. Notice the non-elitist attractor update in line 9, which is the key change of vQEA over QEA.

3.2.2 *vQEA on the One Max problem*

Similar to QEA above, the behaviour of vQEA on the One Max problem is studied for $\lambda = 100$. The settings of the evolutionary parameters in vQEA are kept almost identical to QEA, in order to allow fair comparison between the two. A population of ten individuals is chosen, rotation angle set to $\Delta\theta = 0.01\pi$, of course no elitism

3 It is worth noting that an extra long-term memory mechanism has been added to store the best collapsed individual ever found, but this mechanism has no impact on the algorithm

Algorithm 2 Versatile Quantum-inspired Evolutionary Algorithm

```
1: t ⇐ 0
2: initialise all Q_i(t)
3: while not termination condition do
4:    for all individuals i do
5:       collapse the Qbits of Q_i(t) to form a solution C_i(t)
6:       evaluate solution C_i(t)
7:    end for
8:    B_global(t) ⇐ best solution among all C_i(t)
9:    global synchronisation:
      non-elitist replacement of attractors: A_i(t) ⇐ B_global(t),   ∀i
10:   update all Q_i(t) to Q_i(t + 1) using a QGate
11:   t ⇐ t + 1
12: end while
```

is used and global synchronisation occurs every generation, *i.e.* $S_{global} = 1$, rendering the parameter for local synchronisation S_{local} across the groups redundant, since $B_{local} = B_{global}$.

In Figure 3.4, the evolution of two illustrative $\mathcal{Q}$bits for QEA (dashed line) and vQEA (solid line) are plotted. The figure reports the value of $\theta(t)$ in the polar coordinates system. The radius is given by t and the angle corresponds to θ such that $cos(\theta) = |\alpha|$ and $sin(\theta) = |\beta|$. For both algorithms, a successful run is presented, since for both cases the angle θ finally reaches an expected value close to $90°$, *i.e.* β close to 1.0. It is clear that QEA and vQEA display very different behaviour. QEA tends to make strong decisions and when a certain attractor is chosen, it is followed for several generations. In fact, this constancy is related to the strategy adopted for updating the attractors based on elitism. Conversely, for vQEA the trajectory of $\theta(t)$ shows jitter during the first 200 generations. Nevertheless, the overall evolution is much smoother compared to the original QEA.

To illustrate this situation, we have also computed for both algorithms the average total number of different attractors used per individual during one run of 500 generations on the One Max problem. We found 25.5 for QEA and more than 372 for vQEA, meaning that the "life time" of an attractor is approximately 19.6 generations for QEA and only 1.34 generation for vQEA.

Figure 3.5 presents the typical evolution of the 100 $\mathcal{Q}$bits using vQEA on the One Max problem. We can see a phase of more than 100 generations where the states of the $\mathcal{Q}$bits remain undecided. Then all the $\mathcal{Q}$bits evolve slowly towards the optimum and the colours change to white. In contrast to the evolution of $\mathcal{Q}$bits in QEA, no hitch-hiking phenomenon is visible using vQEA on this problem. This fact is clearly

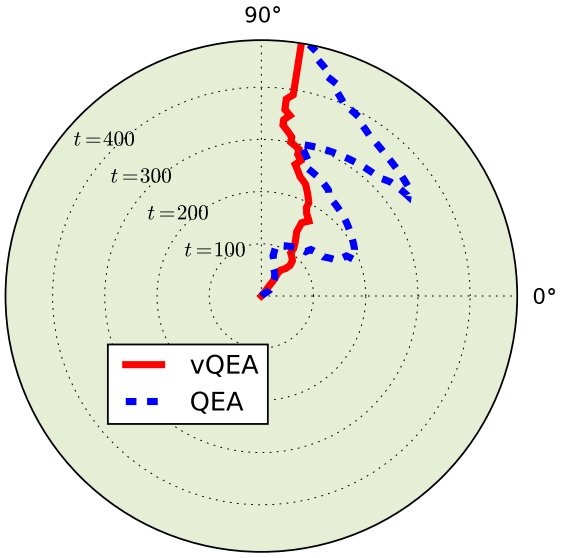


Figure 3.4: Typical evolution of a Quantum bit (value of $\theta(t)$) for QEA (dashed line) and vQEA (solid line) on the One Max problem. Due to elitism in QEA attractors are less frequently replaced compared to vQEA, and as a consequence a $\mathcal{Q}$bit is updated towards the same attractor for many generations. In vQEA on the other hand, an attractor is exchanged much more frequently, which results in a globally smoother exploration of the search space.

demonstrated by comparing the two Figures 3.2 and 3.5. More specifically, no dark colours appear in Figure 3.5. We note that for this run vQEA was able to find the global optimum in 340 generations.

To understand the characteristic evolution of $\mathcal{Q}$bits in vQEA better, the evolution of $\mathcal{Q}$bit Q_4^{23}, *i.e.* values of $|\alpha|^2$ (dotted line) and $|\beta|^2$ (solid line), as well as the states of the corresponding collapsed bit C_4^{23} and attractor bit A_4^{23} are presented in Figure 3.6. We see that $|\beta|^2$ moves slowly but continuously towards 1, which is the desired behaviour. In the early stage of evolution the attractor bit reports many changes of its state, while in later generations the frequency of change decreases and finally the attractor bit converges to 1.

3.3 EXPERIMENTS

In this section, vQEA is tested and compared to a classical genetic algorithm (CGA) (Goldberg, 1989) and to the original QEA on two benchmark problems. For both problems, the fitness of the average best solution found in 30 runs is presented. We use a sta-

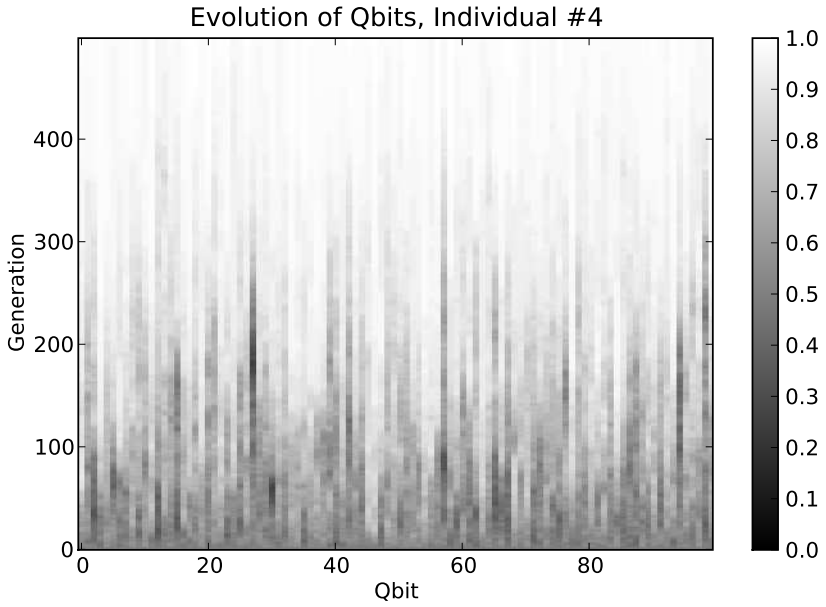


Figure 3.5: Typical evolution of a $\mathcal{Q}$bit string using vQEA on the One Max problem. The shades indicate the value of $|\beta|^2$ of each of the $\lambda = 100$ $\mathcal{Q}$bits at a given generation. All $\mathcal{Q}$bits gradually evolve towards the global optimum 1^λ of the problem, *i.e.* no hitch-hiking phenomenon occurs.

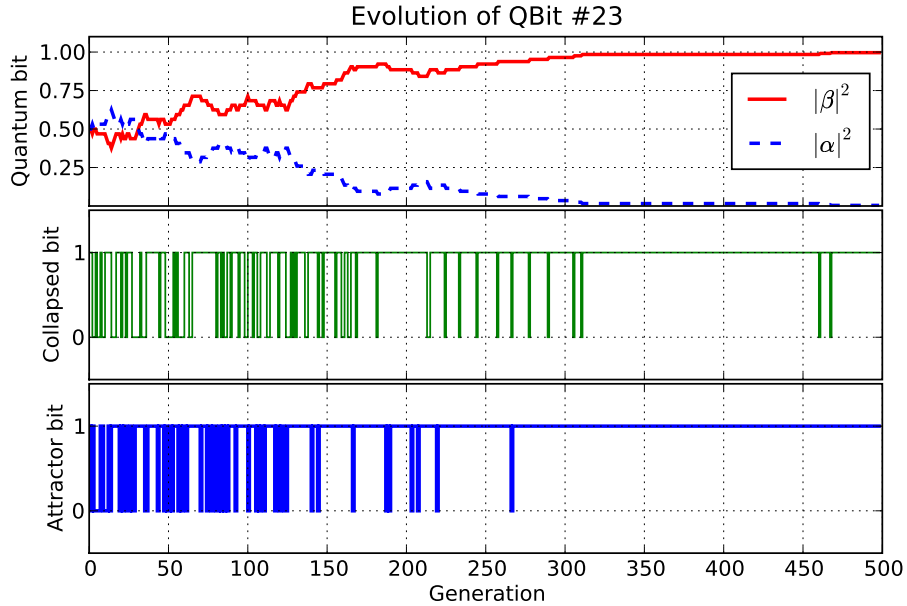


Figure 3.6: Typical evolution of a Quantum bit, Collapsed bit and Attractor bit with vQEA on the One Max problem

tistical unpaired, two-tailed t-test with 95% confidence to determine if results are significantly different.

$\lambda = N = 500$, 1000 generations		
CGA	QEA	vQEA
2963.5 (19.7)	3013.5 (18.9)	3058.0 (15.9)

Table 3.1: Average profit of the best solution found on the 01-knapsack problem for all tested algorithms. Each profit was obtained in 30 independent runs; brackets indicate the standard deviation.

3.3.1 *Optimisation of a* 01*-knapsack problem*

The 01-knapsack problem is a classical NP-hard benchmark problem in which the most valuable subset among N items that have different profits and volumes needs to be identified. This subset also must fit in a knapsack of limited capacity. Han and Kim (2003) evaluated CGA and QEA already on a 01-knapsack problem. Hence, we adopt here exactly the same settings for the evolutionary parameters of CGA and QEA, respectively. We note that these settings were obtained from a comprehensive parameter study and were shown to be suitable for the problem at hand. Additionally, we use the same 01-knapsack problem definition as described in (Han & Kim, 2003). It is worth mentioning that the population size in CGA is equal to 100 and only 10 in both QEA and vQEA. For vQEA elitism was switched off and S_{global} set to one. All three algorithms were allowed to evolve over 1000 generations and for each algorithm 30 independent runs were performed and then averaged.

The results are reported in Table 3.1 for N=500 items. Our implementation of CGA and QEA found solutions comparable to (Han & Kim, 2003). QEA significantly outperforms CGA, but the best results are reported by vQEA. The improvement of vQEA over QEA is very similar to the improvement of QEA over CGA.

In Figure 3.7, the evolution of the average best profit is presented for the three tested algorithms. We see that during the first generations CGA reports the best profit, but is outperformed by vQEA after 306 generations and later also by QEA after generation 454. As indicated by the error bars in the figure, all algorithms achieve significantly different final profits at the end of the optimisation process, also *cf.* Table 3.1. It is noteworthy, that CGA was allowed to use 10 times more fitness evaluations, due to its larger population size. More specifically, in each generation 100 individuals were evaluated in the case of CGA, compared to only 10 individuals/evaluations for each of the QEA methods. Hence, both versions of QEA require less computational resources than CGA, and simultaneously deliver better optimisation results.

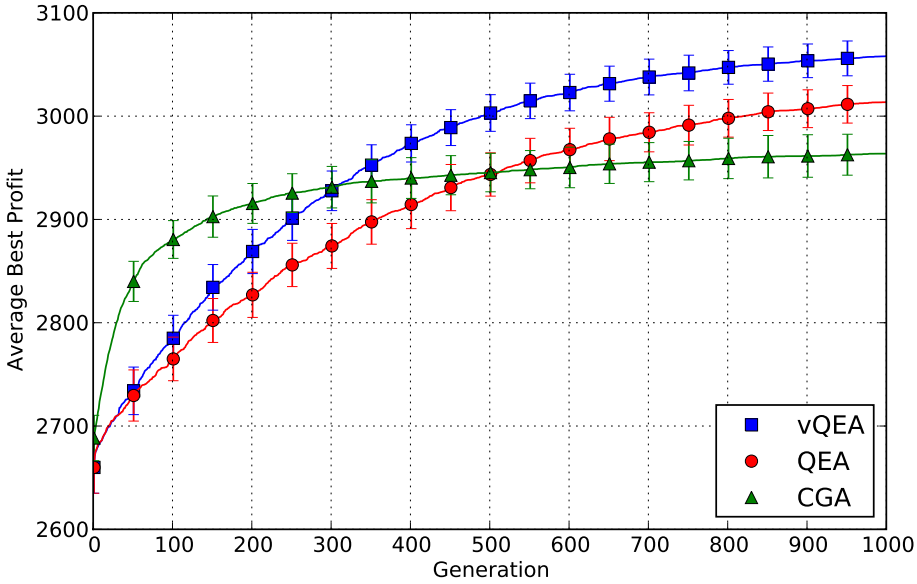


Figure 3.7: Average profit of the best solution found on a 01-knapsack problem of size N=500. Both QEA methods demonstrate superior optimisation performance compared to CGA, and vQEA being significantly faster than QEA.

3.3.2 *Optimisation of NK-landscapes problems*

In (Kauffman, 1993) a synthetic benchmark problem was developed, namely the NK-landscapes, which allows the explicit modelling of linkage between variables. Here the quality of a solution depends not only on the state of its N variables, but also on the K interactions between them. The problem requires the setting of two parameters: N determines the size of the search space and K controls the number of variable links. With increasing K, the number of local optima also increases. Setting $K = 0$ results in a single global optimum, while $\frac{2^N}{N+1}$ local optima exist for $K = N - 1$. NK-landscapes have been used in theoretical biology, *e.g.* to study gene networks, the evolution of proteins or immune systems. The NK-landscapes define also a family of combinatorial optimisation problems that are now widely used as benchmarks for EA. According to (Weinberger, 1996), the model allows the generation of a "tunable rugged" fitness landscape.

In this study, the K interactions between the N variables are chosen randomly and the corresponding problem has been proved to be NP-complete for $K \geq 1$ (Weinberger, 1996). The performances of the three algorithms are studied for problems of increasing size with N =256, N =512, N =1024, N =2048 and N =4096 and of increasing difficulty with K varying from 0 to 8. To allow a statistical analysis of the

results, each algorithm was evaluated in 30 independent runs. Each run corresponds to $10,000$ generations.

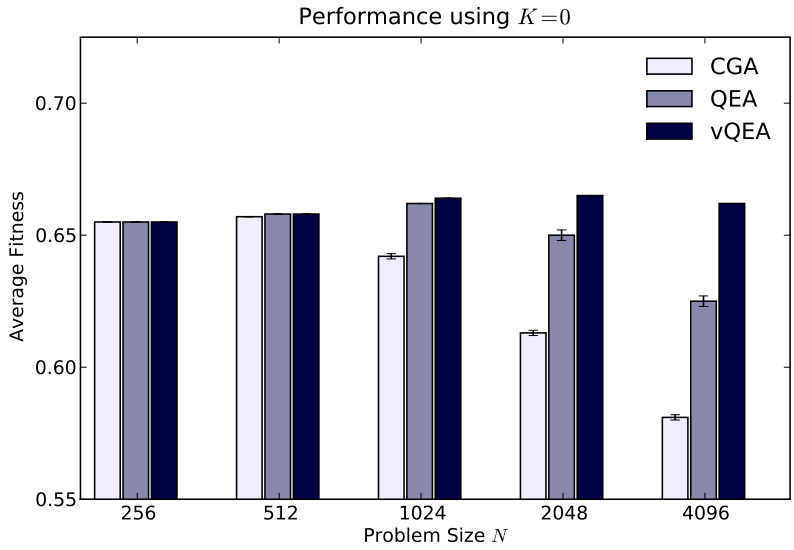
Han and Kim (2004) introduced a modified rotation gate operator, namely the H_ϵ gate, which prevents the convergence of the probability amplitudes $|\alpha|^2$ and $|\beta|^2$ of a $\mathcal{Q}$bit towards 0 or 1. Instead, the authors suggested to use a minimum and maximum value for the amplitudes, ϵ and $1-\epsilon$, respectively, where $\epsilon \in \mathbb{R}$ is a parameter of the operator. An experimental analysis of the H_ϵ gate revealed a superior optimisation performance especially on multi-modal problems.

Preliminary experiments using vQEA on the NK-landscapes benchmark confirmed the advantages of the H_ϵ gate as reported in (Han & Kim, 2004). Hence, we introduce it here as the default operator for vQEA. Parameter ϵ is set to $\sin^2(2 \times \Delta\theta)$ which stops the amplitude update two rotation steps before their convergence towards 0 or 1. A default learning rate $\Delta\theta = 0.01\pi$ is assumed.

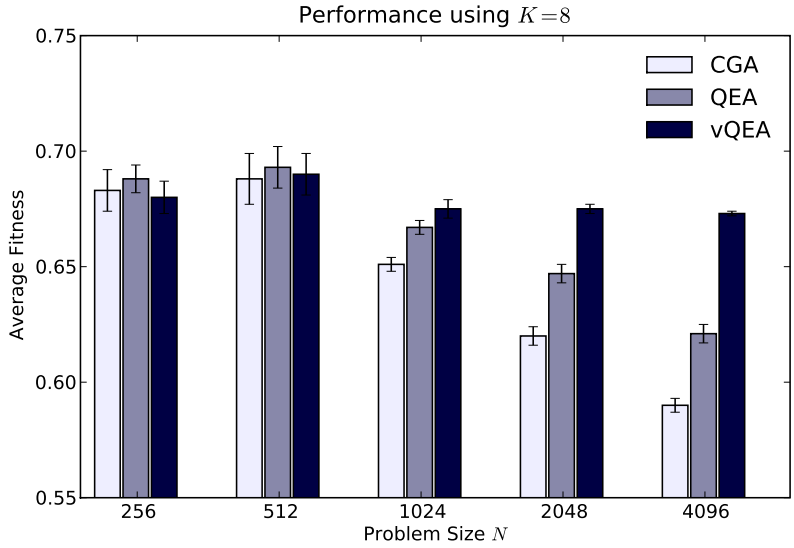
The average fitness of the best solutions found with $K = 0$ and 8 are plotted in Figure 3.8 (error bars indicate standard deviation). For $K = 0$ and $N = 256$ the problem is very easy and can be solved by all three algorithms. With the increase of N, the performance of CGA and QEA decreases and both methods are significantly outperformed by vQEA. Moreover, the average fitness of the solutions found with vQEA is almost unaffected by N. For $K = 8$, all three algorithms perform equally well for $N = 256$ and $N = 512$, but for larger N a similar trend as observed for $K = 0$ is reported. From these results it is claimed that vQEA is a highly scalable algorithm even for difficult problems.

In Figure 3.9 the final fitness of all tested methods is presented relative to the obtained fitness of CGA, *i.e.* the performance of CGA is used as a reference. In this diagram the problem size is $N = 4096$ and the fitness for different values of K is shown. It is clear vQEA outperforms CGA by 14% regardless of the difficulty of the problem. Conversely, for QEA, this ratio varies from 8% for $K = 0$ to 5% for $K = 8$. We note also that the standard deviation reported in Table 3.1 for vQEA is the smallest, indicating that most of the 30 runs have found similarly good solutions. For further inter-comparisons, the overall numerical results obtained with CGA, QEA and vQEA are reported in Tables 3.2 and 3.3.

Figure 3.10 presents two isofitness clouds. In these clouds, each point (gen_A, gen_B) corresponds to the average number of generations needed by two different algorithms A and B, in order to reach the same fitness value. For example: The point $(100, 200)$ would indicate, that algorithm A required 100 generation to achieve the same fitness level as algorithm B in generation 200.



(a) $K = 0$



(b) $K = 8$

Figure 3.8: Average fitness of the best solution found on NK-landscapes with $K = 0$ and $K = 8$. On larger sized problems ($N \geq 1024$) vQEA demonstrates a significant improvement over CGA and the original QEA.

We introduce this kind of representation to allow practical comparisons of computational resources required by algorithms reporting different best fitness values and different convergence speeds. In our case, the isofitness clouds have been computed from all the experiments on the NK-landscapes reported above. The underlying assumption is that the resources needed for computing one generation are the same for all tested algorithms, which is partly false. Indeed, when CGA and QEA (or CGA

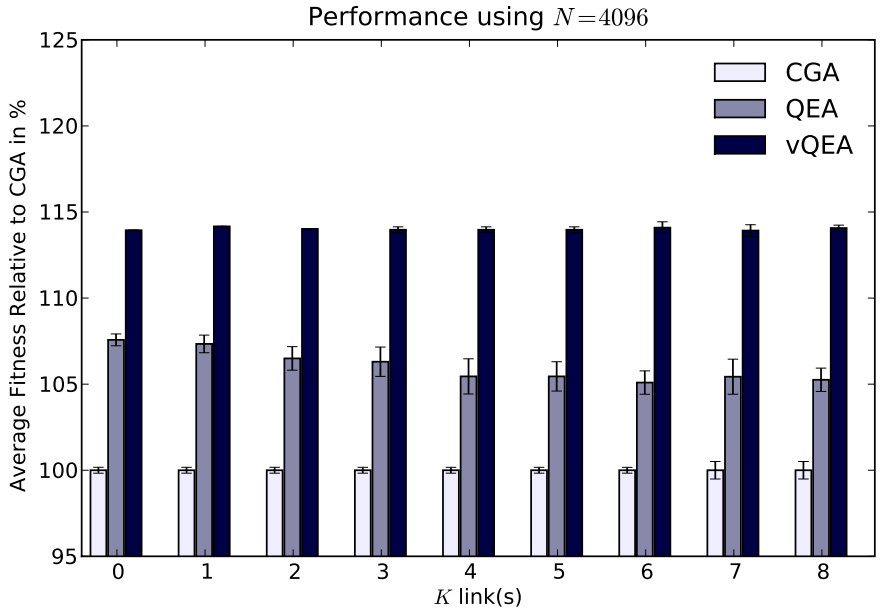


Figure 3.9: Relative fitness of the best solution found on NK-landscapes with N=4096. Here the performance of CGA is used as a reference and the bars show the relative improvement of the methods over CGA. Error bars indicate standard deviation. On average QEA reports a 8% to 5% advantage in terms of fitness compared to CGA. For vQEA this improvement is around 14%.

and vQEA) are compared, the size of the populations are significantly different, respectively 100 and 10 individuals and so a generation is processed faster with QEA or vQEA than with CGA.

Figure 3.10a compares CGA and QEA. Notice that most of the points fall below the line $y = x$ showing that QEA was faster than CGA. The biggest difference in convergence speed is reported for points at the bottom right corner of the figure meaning that CGA required $10,000$ generations to discover solutions of similar quality as those found by QEA at generation $\approx 1,000$. However, we note that for the early generations, *i.e.* before 1000, some points indicate that CGA was the first to reach a given fitness level. After studying the data, we have found that those points correspond to the easiest problems with small values of N (256 and 512) and $K = 0$. Figure 3.10b displays the isofitness cloud obtained for QEA *vs* vQEA. It is clearly demonstrated that vQEA is almost always faster than QEA independent of the size and the difficulty of the problem. We also see that after generation 4,000, the "slope" of the cloud is nearly equal to 0. This means that the QEA needs a very high number (asymptotically an infinite number) of generations to find solutions as good as the solutions found by vQEA in less than 2,000 generations.

K	CGA	QEA	vQEA
	$N = 256$		
0	$0.655_{(\sigma=0.000)}$	$0.655_{(\sigma=0.000)}$	$0.655_{(\sigma=0.000)}$
1	$0.677_{(\sigma=0.000)}$	$0.677_{(\sigma=0.000)}$	$0.677_{(\sigma=0.000)}$
2	$0.679_{(\sigma=0.003)}$	$0.680_{(\sigma=0.001)}$	$0.680_{(\sigma=0.000)}$
3	$0.682_{(\sigma=0.010)}$	$0.679_{(\sigma=0.009)}$	$0.674_{(\sigma=0.005)}$
4	$0.690_{(\sigma=0.007)}$	$0.694_{(\sigma=0.004)}$	$0.695_{(\sigma=0.002)}$
5	$0.689_{(\sigma=0.007)}$	$0.691_{(\sigma=0.008)}$	$0.683_{(\sigma=0.006)}$
6	$0.686_{(\sigma=0.009)}$	$0.692_{(\sigma=0.003)}$	$0.691_{(\sigma=0.004)}$
7	$0.690_{(\sigma=0.011)}$	$0.691_{(\sigma=0.007)}$	$0.695_{(\sigma=0.009)}$
8	$0.683_{(\sigma=0.009)}$	$0.688_{(\sigma=0.006)}$	$0.680_{(\sigma=0.007)}$
K	$N = 512$		
0	$0.657_{(\sigma=0.000)}$	$0.658_{(\sigma=0.000)}$	$0.658_{(\sigma=0.000)}$
1	$0.681_{(\sigma=0.000)}$	$0.682_{(\sigma=0.000)}$	$0.682_{(\sigma=0.000)}$
2	$0.671_{(\sigma=0.002)}$	$0.673_{(\sigma=0.001)}$	$0.673_{(\sigma=0.000)}$
3	$0.673_{(\sigma=0.005)}$	$0.676_{(\sigma=0.003)}$	$0.678_{(\sigma=0.000)}$
4	$0.681_{(\sigma=0.003)}$	$0.683_{(\sigma=0.000)}$	$0.683_{(\sigma=0.000)}$
5	$0.679_{(\sigma=0.006)}$	$0.684_{(\sigma=0.001)}$	$0.685_{(\sigma=0.000)}$
6	$0.687_{(\sigma=0.011)}$	$0.692_{(\sigma=0.006)}$	$0.687_{(\sigma=0.006)}$
7	$0.678_{(\sigma=0.003)}$	$0.680_{(\sigma=0.003)}$	$0.680_{(\sigma=0.004)}$
8	$0.688_{(\sigma=0.011)}$	$0.693_{(\sigma=0.009)}$	$0.690_{(\sigma=0.009)}$
K	$N = 1024$		
0	$0.642_{(\sigma=0.001)}$	$0.662_{(\sigma=0.000)}$	$0.664_{(\sigma=0.000)}$
1	$0.648_{(\sigma=0.001)}$	$0.665_{(\sigma=0.002)}$	$0.669_{(\sigma=0.002)}$
2	$0.643_{(\sigma=0.001)}$	$0.660_{(\sigma=0.001)}$	$0.665_{(\sigma=0.000)}$
3	$0.649_{(\sigma=0.002)}$	$0.667_{(\sigma=0.002)}$	$0.672_{(\sigma=0.002)}$
4	$0.653_{(\sigma=0.003)}$	$0.673_{(\sigma=0.003)}$	$0.679_{(\sigma=0.000)}$
5	$0.658_{(\sigma=0.003)}$	$0.675_{(\sigma=0.002)}$	$0.681_{(\sigma=0.001)}$
6	$0.653_{(\sigma=0.002)}$	$0.667_{(\sigma=0.003)}$	$0.674_{(\sigma=0.003)}$
7	$0.654_{(\sigma=0.004)}$	$0.670_{(\sigma=0.003)}$	$0.676_{(\sigma=0.003)}$
8	$0.651_{(\sigma=0.003)}$	$0.667_{(\sigma=0.003)}$	$0.675_{(\sigma=0.004)}$

Table 3.2: Average profit of the best solution found on the NK-landscapes problem after $10,000$ generations for N=256, 512, and 1024. In brackets the standard deviation is shown.

3.4 DISCUSSION

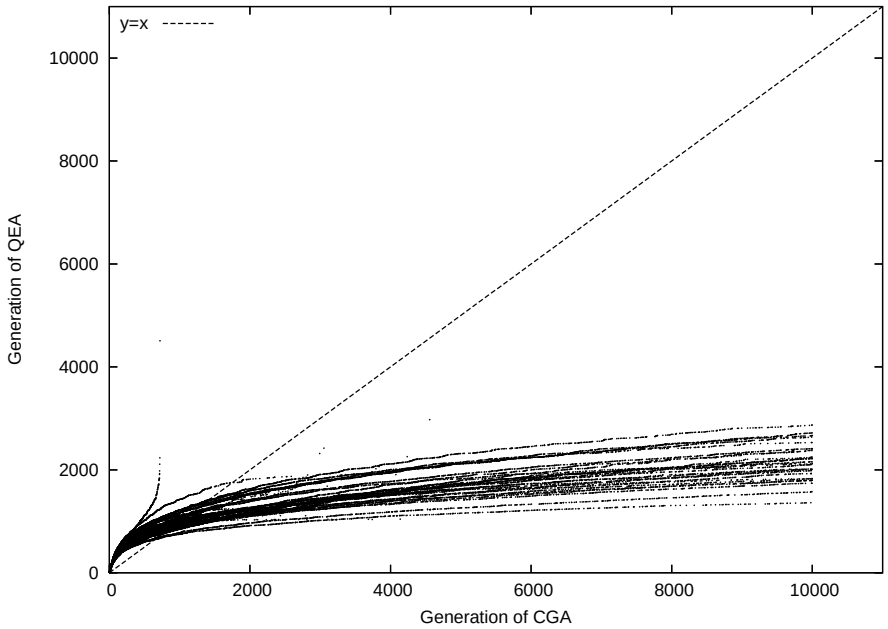
According to (Mühlenbein & Paass, 1996), algorithms using a probabilistic model to explore a search space are called Estimation of Distribution Algorithms (EDA). Thus, it is argued here that in QEA the $\mathcal{Q}$individuals act as probabilistic models and so, as has already been claimed in (S. Zhou & Sun, 2005b) and (Han & Kim, 2006),

	CGA	QEA	vQEA
K	$N = 2048$		
0	$0.613_{(\sigma=0.001)}$	$0.650_{(\sigma=0.002)}$	$0.665_{(\sigma=0.000)}$
1	$0.612_{(\sigma=0.001)}$	$0.645_{(\sigma=0.004)}$	$0.665_{(\sigma=0.000)}$
2	$0.617_{(\sigma=0.001)}$	$0.649_{(\sigma=0.004)}$	$0.671_{(\sigma=0.001)}$
3	$0.617_{(\sigma=0.002)}$	$0.650_{(\sigma=0.004)}$	$0.673_{(\sigma=0.000)}$
4	$0.623_{(\sigma=0.001)}$	$0.655_{(\sigma=0.004)}$	$0.678_{(\sigma=0.000)}$
5	$0.617_{(\sigma=0.002)}$	$0.647_{(\sigma=0.004)}$	$0.671_{(\sigma=0.000)}$
6	$0.624_{(\sigma=0.002)}$	$0.653_{(\sigma=0.005)}$	$0.678_{(\sigma=0.001)}$
7	$0.623_{(\sigma=0.004)}$	$0.653_{(\sigma=0.004)}$	$0.678_{(\sigma=0.001)}$
8	$0.620_{(\sigma=0.004)}$	$0.647_{(\sigma=0.004)}$	$0.675_{(\sigma=0.002)}$
K	$N = 4096$		
0	$0.581_{(\sigma=0.001)}$	$0.625_{(\sigma=0.002)}$	$0.662_{(\sigma=0.000)}$
1	$0.586_{(\sigma=0.001)}$	$0.629_{(\sigma=0.003)}$	$0.669_{(\sigma=0.000)}$
2	$0.585_{(\sigma=0.001)}$	$0.623_{(\sigma=0.004)}$	$0.667_{(\sigma=0.000)}$
3	$0.587_{(\sigma=0.001)}$	$0.624_{(\sigma=0.005)}$	$0.669_{(\sigma=0.001)}$
4	$0.587_{(\sigma=0.001)}$	$0.619_{(\sigma=0.006)}$	$0.669_{(\sigma=0.001)}$
5	$0.587_{(\sigma=0.001)}$	$0.619_{(\sigma=0.005)}$	$0.669_{(\sigma=0.001)}$
6	$0.589_{(\sigma=0.001)}$	$0.619_{(\sigma=0.004)}$	$0.672_{(\sigma=0.002)}$
7	$0.589_{(\sigma=0.003)}$	$0.621_{(\sigma=0.006)}$	$0.671_{(\sigma=0.002)}$
8	$0.590_{(\sigma=0.003)}$	$0.621_{(\sigma=0.004)}$	$0.673_{(\sigma=0.001)}$

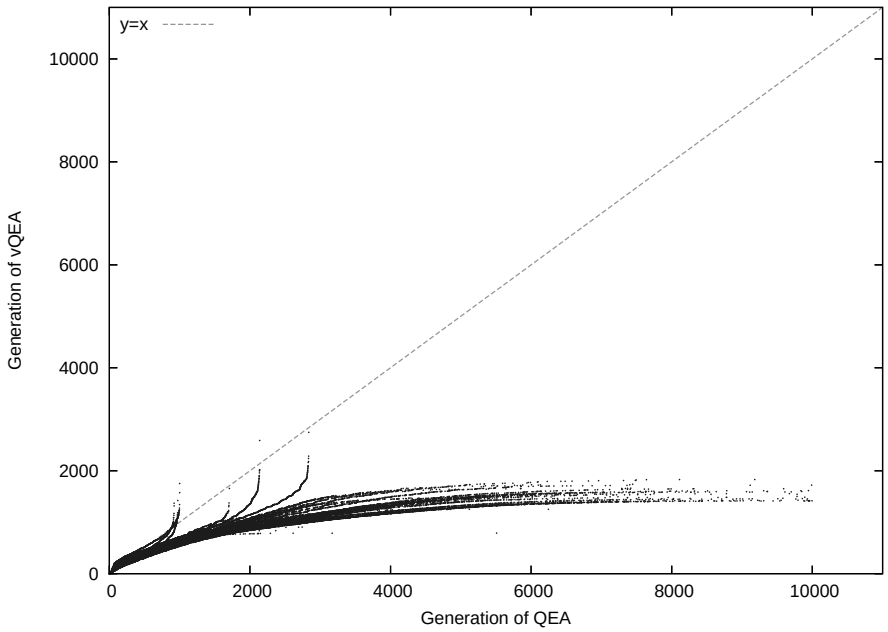
Table 3.3: Average profit of the best solution found on the NK-landscapes problem after $10,000$ generations for N=2048 and 4096. In brackets the standard deviation is shown.

QEA is a new EDA approach. In this section, the role of elitism in EDA is briefly discussed.

In CGA, elitism has been introduced as a protection mechanism to counteract the disruptive effects of genetic operators such as the uniform crossover. In some EDA, the probabilistic models can undergo perturbations to explore the search space, but these perturbations do not have strong consequences and elitism is not necessary. Moreover, with some other EDA, the probabilistic models are completely reconstructed every generation and elitism is not used. Nevertheless, Ahn, Kim, and Ramakrishna (2003a) report an interesting counter example where an EDA is presented and better results are reported with elitism but in that case also a uniform crossover is applied to the bit strings. Thus, so long as no disruptive operators are employed, there is no need for an EDA or a quantum inspired algorithm to have recourse to elitism.



(a) CGA *vs* QEA



(b) QEA *vs* vQEA

Figure 3.10: Isofitness clouds comparing the required computational costs between the algorithms. A point in these diagrams corresponds to the generations needed for an algorithm A to achieve the same fitness level of an algorithm B. CGA uses more computational resources than QEA, while QEA requires more resources than vQEA.

3.5 CONCLUSION

The Quantum-Inspired Evolutionary Algorithm (QEA) introduced in (Han & Kim, 2002) and studied here is elitist. The exploration of the search space is driven by

attractors corresponding to the best solution found so far at either the individual, local or global level. If a non-optimal solution is propagated to the global level, this solution starts to attract the entire population. In that case, to avoid being trapped, the algorithm has to discover a better solution before converging to this global attractor. Hence, the choice of a sub-optimal attractor may become irreversible.

To counteract this issue, the Versatile Quantum-Inspired Evolutionary Algorithm (vQEA) is proposed. In vQEA elitism is removed and the search at time $t + 1$ is driven by the best solution found at time t. Simply removing elitism has strong consequences. With vQEA, the information about the search space collected during evolution is not kept at the individual level but continuously renewed and shared among the whole population. In terms of both speed and accuracy vQEA performs better than QEA on different benchmark problems.

The dynamics of QEA and vQEA are very distinct. The short-term behaviour of QEA is almost always constant because preferential search directions are chosen and followed during several generations. Conversely, the short-term behaviour in vQEA is much more unsettled and the search directions are reevaluated every generation. Thus the eventual decision errors do not have long-term consequences. vQEA is continuously adapting the search according to local information while the quantum individuals act as memory buffers to keep track of the search history. This leads to a much smoother and more efficient long-term exploration of the search space.

In this study, since all the attractors are synchronised at every generation, the local level with the $\mathcal{Q}$groups are redundant. Nevertheless, the concept of groups is very interesting, since it is similar to demes in classical EA. Further studies may address the setting of both local and global synchronisation.

Chapter 4

QUANTUM-INSPIRED EVOLUTIONARY ALGORITHM: A MULTI-MODEL EDA

Numerous natural and physical real world processes have recently inspired researchers in various domains of Artificial Intelligence, such as neuro-computing, Artificial Evolution, Ant Colony Optimisation or Simulated Annealing, to name a few. The use of metaphoric comparisons is a clear trend for search and optimisation algorithms. Nevertheless, metaphors can not last long without strong theoretical justification.

Quantum physics and quantum computing principles have also been widely seen as a source of inspiration, for example in Neural Networks (Menneer & Narayanan, 1995), Genetic Algorithms (Narayanan & Moore, 1996), Differential Evolution (Draa, Batouche, & Talbi, 2004b), Artificial Immune Systems (Li & Jiao, 2005) and Particle Swarm Optimisation (J. Liu, Sun, & Xu, 2006). In the field of Evolutionary Computation, the introduction of the Quantum-Inspired Evolutionary Algorithms (QEA) by Han and Kim might be the most successful application of the quantum metaphor (Han & Kim, 2002, 2003; Han, 2003). It has been earlier alluded that QEA is related to Estimation of Distribution Algorithms (EDA) (S. Zhou & Sun, 2005b; Han & Kim, 2006). The first aim of this chapter is to integrate QEA in a systematic way into the class of EDA as an original algorithm.

EDA have shown their ability to avoid the disruptive effects of genetic operators in Evolutionary Algorithms (EA), namely crossover and mutation, by iteratively evolving a probabilistic model to explore the search space. Three different classes of EDA have been proposed to categorise these algorithms according to the modelling of interaction between variables of optimisation problems (Pelikan, Goldberg, & Lobo, 1999). See also (Larrañaga, Etxeberria, Lozano, & Peña, 1999) for an overview of proposed EDA for each class.

Early EDA assume independent relationships between parameters for a given problem and thus the probability distribution of solutions can be factored as a product of

independent uni-variate probabilities. This class of EDA includes the well-known Probabilistic Incremental Learning (PBIL) (Baluja, 1994), the compact Genetic Algorithm (cGA) (Harik, Lobo, & Goldberg, 1999) and the Uni-variate Marginal Distribution Algorithm (UMDA) (Mühlenbein & Paass, 1996), to name a few.

Recent developments in the field of EDA take possible interactions between variables explicitly into account. Modelling bi-variate dependencies represents the second class of EDA and is implemented by *e.g.* the Mutual Information Maximisation for Input Clustering (MIMIC) algorithm (Bonet, Isbell, & Viola, 1997), the COMIT algorithm (Combining Optimisers with Mutual Information Trees) (Baluja & Davies, 1997, 1998) and the Bi-variate Marginal Distribution Algorithm (BMDA) (Pelikan & Mühlenbein, 1999).

The third class of EDA can model multivariate variable interactions. Examples of algorithms of this class are the Factorised Distribution Algorithm (FDA) (Mühlenbein, Mahnig, & Rodriguez, 1999), the Extended Compact Genetic Algorithm (EcGA) (Harik, 1999) and the Bayesian Optimisation Algorithm (BOA) (Pelikan, Goldberg, & Cantú-paz, 2000).

It is worth noting that the second and third classes of EDA require complex learning algorithms and significant additional computational resources in order to handle variable interactions. It has been pointed out, *e.g.* in (Johnson & Shapiro, 2001), that under certain conditions the benefit of this overhead might still be unclear. As a consequence the first class of EDA, although being simple, should not be discredited *a priori*. In this chapter, the common points and specifics of QEA compared to other EDA are highlighted. In a similar way, other methods have also been shown to belong to EDA. For example, Cordón et al. (2000) and Monmarché et al. (1999) show how EDA and the Ant Colony Optimisation (ACO) algorithm (Dorigo, Maniezzo, & Colorni, 1996) are actually very similar and differ mainly in the way their probabilistic model is updated.

The use of a probabilistic model is the key concept of any EDA. The QEA follows the same strategy to guide its search in a given space of solutions. Moreover, in QEA multiple probabilistic models are created and incrementally modified. The idea of using multiple interacting models in EDA is not new. Probably initiated in (Zhang, Sun, Tsang, & Ford, 2002), this idea is now very popular (Ahn, Kim, & Ramakrishna, 2003b; Ahn, Goldberg, & Ramakrishna, 2003; delaOssa, Gámez, & Puerta, 2006; Madera, Alba, & Ochoa, 2006; S. Zhou & Sun, 2005a).

We can identify at least two reasons why the multi-model approach might be useful for optimisation problems. First, simple EDA such as UMDA and PBIL cannot solve complicated problems as shown in (González, Lozano, & Larrañaga, 2000)

and (Zhang, 2004). Second, even advanced EDA using a complex – but still single – probabilistic model may not work well in practise (Zhang, Sun, & Tsang, 2005). In QEA, the interaction of the probabilistic models is unique. It is this interaction that provides the search with an adaptive learning speed and a buffer against potential decision errors. An explicit aim of this chapter is to confirm that several models together perform better than only one and then to explain why. This study was published in (Defoin-Platel, Schliebs, & Kasabov, 2009).

We start this analysis by investigating the key components of QEA in the light of EDA. Therefore the probabilistic model, selection and sampling procedures, learning strategies and population structure used in a QEA are compared to some classical EDA. In an extensive experimental study the behaviour and performance of QEA in terms of fitness, scalability, diversity loss and robustness against noise is investigated. In the final part the role of multiple probabilistic models is discussed and some potential advantages are highlighted.

4.1 VQEA IS AN EDA

According to (Mühlenbein & Paass, 1996), the algorithms that use a probabilistic model of promising solutions to guide further exploration of the search space are called Estimation of Distribution Algorithms (EDAs). We have seen in chapter 3 that each $\mathcal{Q}$individual defines a probability vector and so, as it has already been claimed in (S. Zhou & Sun, 2005b) and (Han & Kim, 2006), vQEA is a new algorithm belonging to the class of EDAs. A generic description of EDAs is proposed in Algorithm 3.

Algorithm 3 Estimation of Distribution Algorithm (EDAs)

1: $t \Leftarrow 0$
2: initialise the probabilistic model $\mathcal{P}(t)$
3: **while** not termination condition **do**
4: sample M new solutions from $\mathcal{P}(t)$ into $D(t)$
5: evaluate the elements of $D(t)$
6: select $L \leq M$ solutions from $D(t)$ into $D_s(t)$ using a selection method
7: learn the probabilistic model $\mathcal{P}(t+1)$ from $D_s(t)$ and eventually from $\mathcal{P}(t)$
8: $t \Leftarrow t+1$
9: **end while**

In this section an extensive study of the features of vQEA is proposed. The common points and specifics of vQEA compared to other EDAs are highlighted.

4.1.1 *Probabilistic model*

The complexity of the probabilistic model, denoted $\mathcal{P}$ in Algorithm 3, varies largely among EDAs. In (Pelikan et al., 1999) a survey on EDAs reports three different classes based on the level of interactions between the variables that their models can represent. In the version of vQEA discussed in this study, binary states are superposed and the eventual interactions between variables are not explicitly taken into account. At the $\mathcal{Q}$individual level the probabilistic model

$$\mathcal{P}_i = \left[|\beta_i^1|^2 \dots |\beta_i^{\mathrm{N}}|^2\right] \tag{4.1}$$

is a vector of probabilities, since each $|\beta_i^j|^2$ value is used independently for sampling. Therefore, vQEA belongs to the first family of EDAs that assumes independent variables and for which the probabilistic model is a vector of probabilities, such as population-based incremental learning (PBIL) (Baluja, 1994), compact GA (cGA) (Harik et al., 1999) and uni-variate marginal distribution algorithm (UMDA) (Mühlenbein & Paass, 1996). All these algorithms are described in detail in Appendix A. This family of EDA – although simple – should not be discredited *a priori* since the benefit of searching complex variable interactions could be still unclear under particular circumstances (Johnson & Shapiro, 2001). We will see in section 4.3 how the p individuals of the $\mathcal{Q}$population interact to form a multi-model EDA, with $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_p\}$.

In EDAs, the probabilistic model $\mathcal{P}$ is iteratively updated to account for the fitness of the last L solutions selected in D_s. Nevertheless, the state space on which PBIL, UMDA, cGA and vQEA act, is different. In PBIL an element $\mathcal{P}^j$ of the probability vector has an arbitrary precision $\Delta\mathcal{P}$ and so the number of possible values for $\mathcal{P}^j$ is infinite. Conversely, in cGA this number is finite and the precision $\Delta\mathcal{P}$ is constant. The so-called *virtual population size* parameter n determines the accuracy of the model since the update steps have a constant size $\Delta\mathcal{P} = 1/n$. With UMDA the accuracy of $\mathcal{P}^j$ depends directly on the number L of solutions selected to compute the next probability. However, the update steps are not constant and depend on the variance of the empirical frequency at locus j.

For vQEA, the situation is even more complex. At the level of a $\mathcal{Q}$bit Q_i^j, the application of the rotation gate operator according to $\Delta\theta$ can only produce a finite number $\frac{\pi}{2} \times \frac{1}{\Delta\theta}$ of positions for the angle $\theta_i^j \in [0, \pi/2]$ and so for the probability $\mathcal{P}_i^j =$

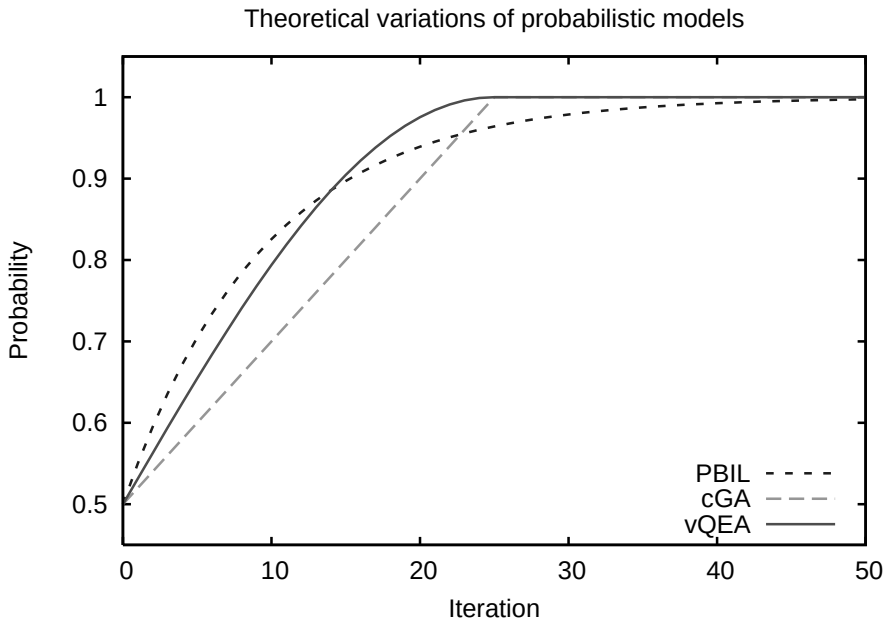


Figure 4.1: Theoretical variations of the probabilistic model in PBIL, cGA and vQEA

$|\beta_i^j|^2 = sin^2(\theta_i^j)$. The size of the update steps is constant in angle but subsequently varies for $\mathcal{P}_i^j$. More formally we have:

$$\begin{aligned} \Delta\mathcal{P}(\theta_i^j) &= sin^2(\theta_i^j + \Delta\theta) - sin^2(\theta_i^j) \\ &= 2cos(\theta_i^j)sin(\theta_i^j) \times \Delta\theta \end{aligned} \tag{4.2}$$

It is worth noticing that, according to equation (4.2), the more a $\mathcal{Q}$bit is converged (with $\theta_i^j \to \frac{\pi}{2}$ or $\theta_i^j \to 0$), the smaller the update step. This phenomenon can be seen as a form of deceleration of the algorithm before convergence.

We can see in Figure 4.1 how an element of the probability vector is affected by several successive applications of the update operators for PBIL, cGA and vQEA. We note that this diagram does not reflect the real behaviour of the algorithms. This is a theoretical situation where the conditional aspects of the update are not taken into account and hence all models are updated at every generation. The initial probability is set to 0.5 and the update direction is toward '1' for each operator. The learning rate of PBIL is fixed to $R_l = 0.1$, the virtual population size of cGA to $n = 50$ and for vQEA the parameter $\Delta\theta$ is equal to $\frac{1}{50}\frac{\pi}{2}$ and only one $\mathcal{Q}$individual is used. With such a setting, both cGA and vQEA require 25 update steps to converge.

When considering the population level of vQEA, a set of p probability vectors interact in a complex way (*cf.* section 4.3). The accuracy of the overall model $\mathcal{P} =$

$\{\mathcal{P}_1, \ldots, \mathcal{P}_p\}$ can be investigated by looking at the variations of the mean model at locus j, noted $\overline{\mathcal{P}^j}$, such that

$$\overline{\mathcal{P}^j} = \frac{1}{p} \sum_{i=1}^{p} |\beta_i^j|^2 \tag{4.3}$$

In vQEA the update of each θ_i^j and subsequently of each $|\beta_i^j|^2$ is conditional and is performed independently among the population. Therefore the number of positions for the average angle $\overline{\theta^j} \in [0, \pi/2]$ is $\frac{\pi}{2} \times \frac{1}{\Delta\theta} \times \frac{1}{p}$.

4.1.2 *Sampling and selection*

The classical EDAs are distinguished also by the number of solutions M (*cf.* line 4 in Algorithm 3) sampled at every generation to form the set D. Both PBIL and UMDA require a comparably large number of samples in order to explore the search space effectively. For example, in (Shapiro, 2005), the author claimed that M should be large compared to the square root of the problem size N for UMDA to find the optimum on a One Max problem. Conversely, cGA works with only $M = 2$ bit strings produced per generation. In vQEA all the $\mathcal{Q}$individuals collapse during one generation and so for each Q_i this phase corresponds to the sampling of only one solution from the corresponding model $\mathcal{P}_i$.

After sampling and evaluation of D, the next step in EDAs consists in selecting L solutions into D_s. This subset will be further used during the learning phase. Again various selection schemes exist in EDAs. For example, PBIL selects only the best (and sometimes together with the worst) element of D [1]. In cGA a tournament determines a winner and a loser solution whereas in UMDA a truncation selection is often employed (Mühlenbein, 1997) where the γ best solutions are selected (typically $\gamma = 50\%$). We note that other models can be used as well, such as proportional or tournament selection (Zhang & Mühlenbein, 2004).

At first glance, the selection process of vQEA may appear not so distinctive: as in a tournament each attractor $A_i(t)$ is basically compared in terms of fitness to the last collapsed string $C_i(t)$. Nevertheless these tournaments are not symmetric. A learning phase occurs only if an attractor wins a tournament, otherwise no solution is selected and there is no learning step in this generation.

1 A similar approach has been explored in the Best-Worst Ant System algorithm (Cordón et al., 2000), which also belongs to the class of EDAs.

It is noteworthy that $C_i(t+1)$ is sampled from $\mathcal{P}_i(t+1)$ and $A_i(t+1)$ from $\mathcal{P}_i(t)$. If the fitness of $A_i(t)$ is not strictly better than the fitness of $C_i(t)$ then the probabilistic model $\mathcal{P}_i(t)$ stays unchanged, *i.e.* $\mathcal{P}_i(t+1) = \mathcal{P}_i(t)$. In this case, $C_i(t+1)$ and $A_i(t+1)$ are sampled from the same probabilistic model. Therefore, from an evolutionary point of view, we can consider that both belong to the same generation. On the other hand, if the fitness of $A_i(t)$ is strictly better than the one of $C_i(t)$, $\mathcal{P}_i(t)$ is updated and $\mathcal{P}_i(t+1) \neq \mathcal{P}_i(t)$. In this case, the selection process involves $C_i(t+1)$ and $A_i(t+1)$ that are issued from generations $t+1$ and t, respectively. In other words, vQEA is a form of *steady-state* EDA where "parents" and "offspring" may compete against each other. This feature of vQEA is an important specific since most of the other EDAs are "generational". However, notable exceptions where elitism is implemented in EDAs exist, for example (Ahn, Kim, & Ramakrishna, 2003b) and (Ahn & Ramakrishna, 2003) where inter-generational competition exists. Larrañaga and Lozano (2002) also apply some steady-state EDAs in the continuous field.

4.1.3 *Learning and replacement*

Step 7 in Algorithm 3 is a learning phase where the probabilistic model $\mathcal{P}(t+1)$ is built to account for the solutions previously selected in $D_s(t)$. With UMDA, $\mathcal{P}(t+1)$ is fully determined using only the set $D_s(t)$ whereas with PBIL and cGA, both $D_s(t)$ and $\mathcal{P}(t)$ are involved and the learning is incremental. In cGA, the learning is also conditional since the update of the model occurs only at the positions where the winner and the loser bit strings differ. In the original version of PBIL the learning is unconditional but we note that some extensions of the basic algorithm have been proposed where the bits of the best and worst solutions are also compared to determine the update (Baluja, 1994).

Besides the update operator itself (*i.e.* the rotation gate) the learning process in vQEA is exactly the same as the one employed in cGA. If an attractor A_i wins a tournament then the binary strings C_i and A_i are systematically compared and the model $\mathcal{P}_i$ is updated toward A_i only where C_i and A_i differ.

Figure 4.2 shows how an element of the probability vector is affected by the learning process for PBIL, cGA and vQEA when solving a one bit One Max problem. We note that UMDA is not studied here because it instantaneously convergences after the first iteration on this problem. Contrary to Figure 4.1, we can see the real algorithms working here with the action of the conditional learning for cGA and both the asymmetric selection and conditional learning for vQEA. The curves correspond to

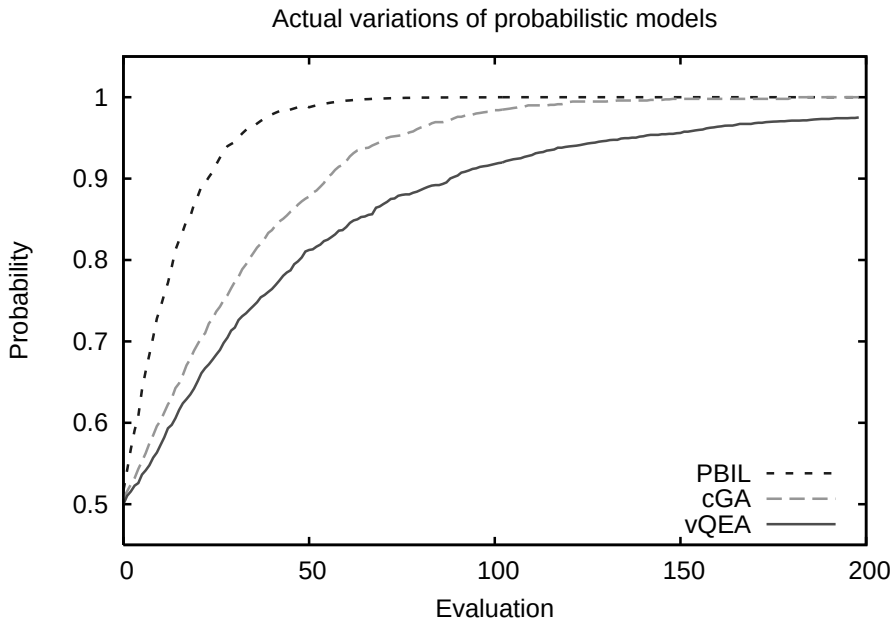


Figure 4.2: Actual variations of the probabilistic model in PBIL, cGA and vQEA

the evolution of one probability averaged among 30 independent runs of 200 generations. The learning rate of PBIL is fixed to $R_l = 0.1$ and $M = 2$ solutions are sampled from the model, the virtual population size of cGA is $n = 50$ and for vQEA the parameter $\Delta\theta = \frac{1}{50}\frac{\pi}{2}$ and only one $\mathcal{Q}$individual is used. With such a setting, the convergence of PBIL is the fastest primarily because the learning is unconditional. The actual shape is not so different from the theoretical shape depicted in Figure 4.1. In fact, with only two samples per generation according to this setting of PBIL, the probability of learning a '0' is not null (*e.g.* 0.25 at the beginning of the run) so the model is sometimes updated toward the wrong direction, slightly slowing down the actual convergence speed. When solving a one bit One Max, conditional learning prevents the models of cGA and vQEA from moving toward the wrong direction and also significantly decrease their convergence speed. In addition, the asymmetric selection makes vQEA slower than cGA. Indeed, the probability of updating the single dimension model $\mathcal{P}$ on this particular problem is $2\mathcal{P}(1-\mathcal{P})$ for cGA and $\mathcal{P}(1-\mathcal{P})$ for vQEA.

Most of the time in vQEA, at generation $t+1$ each $\mathcal{Q}$individual attractor $A_i(t+1)$ corresponds to the last sampled solution $C_i(t)$. Nevertheless, according to the structure of the $\mathcal{Q}$population and the local and global synchronisation periods, several $\mathcal{Q}$individuals can also share a common attractor during one generation.

4.1.4 *Population structure*

Because of the numerous aforementioned specifics of vQEA compared to other EDAs, it is clear that even when considering only a single $\mathcal{Q}$individual, vQEA is an original EDA. Nevertheless, what makes vQEA unique is that it was designed as a coarse-grained algorithm with a complex structured population of $\mathcal{Q}$individuals. The situation can be easily compared to multiple demes in EA where sub-populations are artificially separated to promote speciation and where migration allows to share information between demes. We note several interesting attempts of multi-population EDAs (Ahn, Kim, & Ramakrishna, 2003b; Ahn, Goldberg, & Ramakrishna, 2003; delaOssa et al., 2006; Madera et al., 2006).

In vQEA, the structure of the population is fully determined by the number g and the size k of the $\mathcal{Q}$groups together with the so-called local and global synchronisation periods, denoted S_{local} and S_{global} respectively[2]. Actually there is not a single fixed topology but rather three superimposed levels of organisations appearing iteratively according to the synchronisation periods. As an example, when a global synchronisation occurs at time t, the best attractor among the $\mathcal{Q}$population is selected and then used at time $t+1$ by the $p = g \times k$ $\mathcal{Q}$individuals. Therefore, at that particular time, the group structure of the $\mathcal{Q}$groups does not matter. The situation is the same for the $\mathcal{Q}$individuals in a $\mathcal{Q}$group that are to some extent connected but only during a local synchronisation event.

In this study we are interested in three different structures: a $\mathcal{Q}$population containing only one single $\mathcal{Q}$individual, a $\mathcal{Q}$population containing a single $\mathcal{Q}$group consisting of several $\mathcal{Q}$individuals and finally the most complex one, a $\mathcal{Q}$population containing several $\mathcal{Q}$groups of several $\mathcal{Q}$individuals each.

4.2 EXPERIMENTS

In this section, PBIL, cGA, UMDA and vQEA are experimentally compared to each other. Besides the fitness performance comparison, we are also interested in the diversity loss, the scalability and the robustness of each algorithm. However, the performance and the overall behaviour of PBIL, cGA and UMDA strongly depend on the setting of their parameters and the optimal setting varies as a function of the problem to solve. It is not the purpose of this study to find the most appropriate setting for each algorithm and then to state that one algorithm is better than another.

2 The reader is referred to chapter 3 for a detailed discussion of all parameters used in vQEA.

Algorithm	Setting	Name
sGA	$M = 100$, uniform crossover $P_{cross} = 1$, $P_{mut} = 0.01$	sGA
PBIL	$M = 10$, $R_l = 0.1$, $R_m = 0.02$, $R_s = 0.05$	PBIL
cGA	$n = \frac{\sqrt{\pi}}{2}\sqrt{\mathrm{N}}\log \mathrm{N}$	cGA
UMDA	$M = 500$, truncation $\gamma = 50\%$	UMDA
vQEA	$g = 1$, $k = 1$, $\Delta\theta = \pi/100$ $g = 1$, $k = 10$, $\Delta\theta = \pi/100$ $S_{global} = 1$ $g = 5$, $k = 2$, $\Delta\theta = \pi/100$ $S_{local} = 1$, $S_{global} = 100$	$\mathtt{vQEA}_{1,1}$ $\mathtt{vQEA}_{1,10}$ $\mathtt{vQEA}_{5,2}$

Table 4.1: Parameters settings for all tested algorithms

4.2.1 *Experimental setting*

We adopted different policies to set the parameters and this is shown in Table 4.1. For vQEA, three settings are investigated: a single $\mathcal{Q}$individual ($\mathtt{vQEA}_{1,1}$), one group of 10 fully synchronised $\mathcal{Q}$individuals ($\mathtt{vQEA}_{1,10}$) and 5 groups of 2 $\mathcal{Q}$individuals synchronised every 100 generation ($\mathtt{vQEA}_{5,2}$). The default H_ϵ gate as described in chapter 3 is used. For PBIL, we decided to fix M to 10 in such a way that the number of solutions sampled and evaluated in one generation is equivalent to both $\mathtt{vQEA}_{1,10}$ and $\mathtt{vQEA}_{5,2}$. Actually, according to (Shapiro, 2005), this setting is suitable for low-dimensional problems (N ~ 100). For cGA, the virtual population size is adapted according to the problem size N following the recommendation reported in (Sastry, Goldberg, & Llora, 2007) whereas for UMDA a fixed setting suitable for high-dimensional problems is used.

The experimental results presented hereafter are obtained by averaging 30 independent runs consisting of 10^5 fitness evaluations for each algorithm and problem tested. We use a statistical unpaired, two-tailed t-test with 95% confidence to determine if results are significantly different.

4.2.2 *Diversity loss*

The drift phenomenon in EA refers to the loss of genetic diversity due to finite population sampling. In (Shapiro, 2005), the loss of diversity is studied in the context of EDAs: It is shown that without selection, *i.e.* on a flat landscape, the variance of the probabilistic model iteratively decays to zero and consequently the model converges

towards a fixed configuration. Most EDAs do not compensate for this and the lost diversity cannot be restored. Moreover, it is also shown that, for a non flat problem, the random drift may counteract the effects of selection. Therefore, the parameters of the algorithms have to be tuned properly so that selection is the main force driving the search.

In this section, an empirical comparison of the loss of diversity of cGA, PBIL, UMDA and vQEA using the settings reported in Table 4.1 is performed on different benchmark problems. Following (Shapiro, 2005), to estimate the diversity of the bit strings sampled by an EDA at generation t, we compute the variance v as:

$$v(t) = \sum_{j}^{\mathrm{N}} \mathcal{P}^j(t)(1 - \mathcal{P}^j(t)) \tag{4.4}$$

where $\mathcal{P}^j(t)$ is the j^{th} element of the probabilistic model $\mathcal{P}$ at generation t. In the case of vQEA, the average model $\overline{\mathcal{P}}(t)$ over the p $\mathcal{Q}$individuals (*cf.* Eq. 4.3) is used instead. The maximum diversity corresponds to $v_0 = N/4$ and $v(t) = 0$ indicates that the models have converged.

We have seen in section 4.1.2 that the selection process determining the learning phase of vQEA is asymmetric since the update of a model $\mathcal{P}_i$ occurs only if $f(A_i) > f(C_i)$. On a flat landscape this situation is impossible, therefore $\mathcal{P}_i$ can not vary and vQEA can not loose diversity, *i.e.* $v(t) = v_0$. We note that this is the optimal behaviour for an EDA since if no information is provided each solution of the search space keeps an equal probability of being sampled. The previous remark also stands for the so-called needle-in-the-haystack problem. Nevertheless, the drift phenomenon exists in vQEA as well and can be monitored if we add noise to the flat landscape. We assume a random noise such that the fitness is either 0 or 1 with an equal probability.

Here we analyse the average empirical variance $v(t)$ as a function of the number of fitness evaluations for a noisy flat landscape; two NK-landscapes [3] with K equals 0 and 8; and a One Max problem. The size of these four problems is fixed to $N = 2048$ variables.

In the noisy flat landscape problem only random drift can cause the convergence of an algorithm, *cf.* Figure 4.3. `PBIL` is the algorithm that is the most prone to loose diversity, with $v(t) = v_0/2$ after only 720 evaluations, probably because the setting of the learning rate is not suitable for high dimensional problems. We see

3 According to (Weinberger, 1996), the K interactions between the N parts of the systems are chosen randomly and the corresponding problem has been proved to be NP-complete for $K \geq 1$.

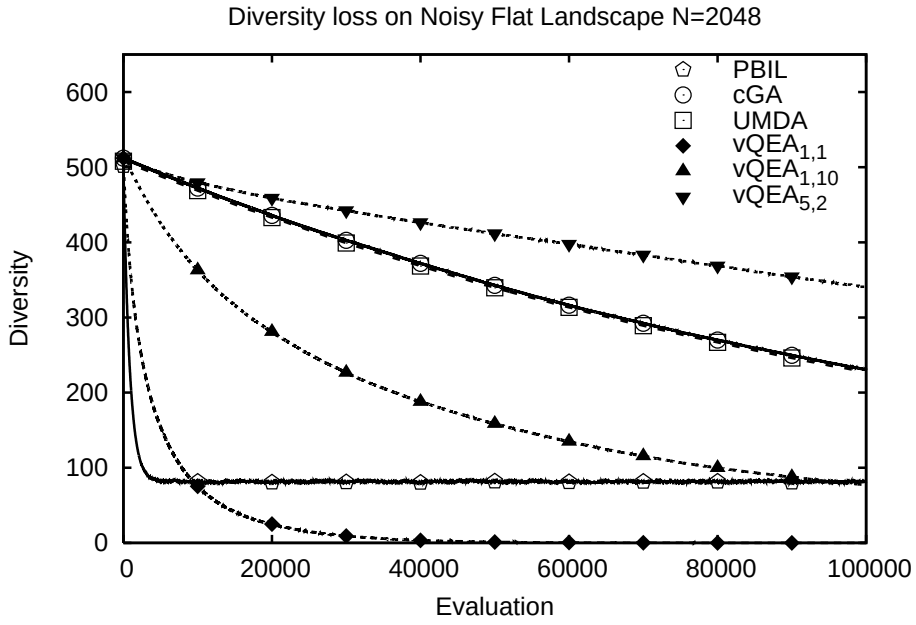


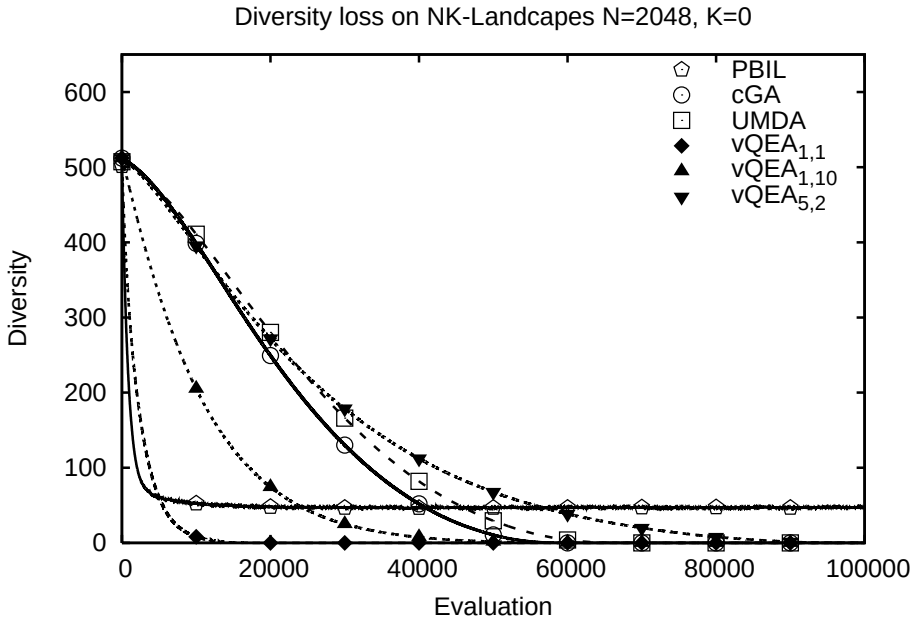
Figure 4.3: Loss of diversity on noisy flat landscape

also the effect of the mutation operator of PBIL that perturbs the probabilistic model and guarantees a residual level of diversity around $1/2\,\mathrm{N}\,R_m\,R_s(1-R_s)$, (*cf.* Appendix A for a detailed description of the mutation operator in PBIL). With L=250 for UMDA and n=305 for cGA, the average loss of diversity of both algorithms appears surprisingly almost identical and is also very slow compared to PBIL with $v(t) = v_0/2$ after around 85,000 evaluations. When comparing the loss of diversity of vQEA$_{1,10}$ and vQEA$_{1,1}$, we found that the shapes of the two curves are identical and that only their convergence speeds differ. Actually, the loss is exactly ten times faster for vQEA$_{1,1}$ than for vQEA$_{1,10}$, with $v(t) = v_0/2$ after $2,400$ and $24,000$ for vQEA$_{1,1}$ and vQEA$_{1,10}$ respectively. vQEA$_{5,2}$ reports the smallest loss of diversity since after 10^5 fitness evaluations we still have $v(t) > v_0/2$.

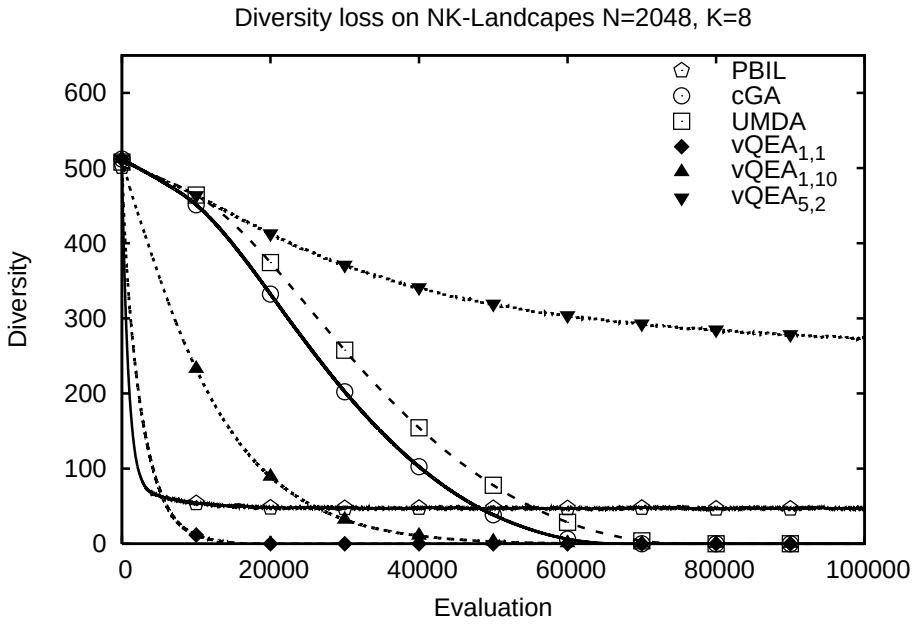
From (Shapiro, 2006), we know that the mathematical expression of the loss of diversity of UMDA on a flat landscape is :

$$v_{\mathrm{UMDA}}(t) = \frac{\mathrm{N}}{4}(1-1/L)^t \tag{4.5}$$

We claim that this expression stands also for the noisy flat landscape as defined above. An attempt was made to fit the variance $v(t)$ of vQEA by varying L in Equation 4.5 for N=2048. It was clear that the loss of diversity of vQEA does not follow the same model as UMDA. Nevertheless, the most appropriate values found for L were 65, 160 and 350, for vQEA$_{1,1}$, vQEA$_{1,10}$ and vQEA$_{5,2}$ respectively.



(a) $K = 0$



(b) $K = 8$

Figure 4.4: Loss of diversity on NK-landscapes with N=2048

On NK-landscapes, the loss of diversity is due to selection only and the global optimum is unique. For $K = 0$, the N variables can be optimised independently so this problem is considered easy to solve. Figure 4.4b shows that the loss of diversity is faster than on the noisy flat landscape for each algorithm tested. The convergence of the probabilistic models towards the global optimum is responsible for this loss

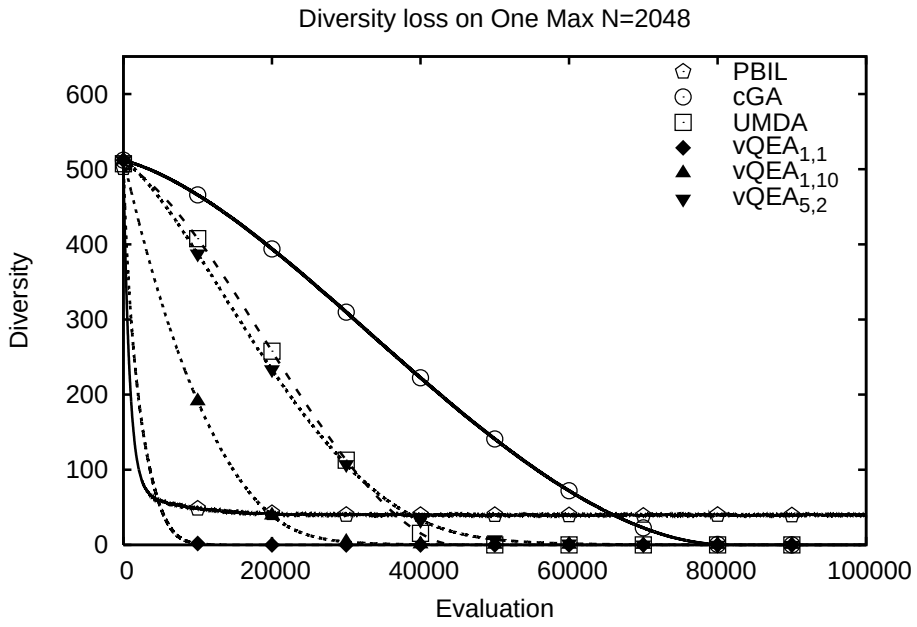


Figure 4.5: Loss of diversity on One Max

and, except for PBIL, the variance $v(t)$ falls down to zero within the 10^5 evaluations. Apart from vQEA$_{5,2}$, the introduction of interactions between the variables (with $K = 8$) does not seem to affect the way the algorithms converge. Although, we will see in section 4.3.3 that the ten probability vectors of vQEA$_{5,2}$ are all almost converged as well.

When we rank the algorithms according to the number of evaluations t at which $v(t) = v_0/2$, the noisy flat landscape and NK-landscapes have identical ranking. This is no longer the case on the One Max, in particular for cGA , *cf.* Figure 4.5. This problem has no local optima but a single global optimum. Additionally, some neutral dimensions exist, since different solutions may have equal fitness values. Hence, both selection and random drift are responsible for the loss of diversity here. As a consequence, the convergence speed of the algorithms is higher on the One Max than on the previously studied problems. Nevertheless, the diversity loss for cGA is slower than on NK-landscapes with K=0. Thus, we can reasonably assume that the neutrality of the problem is responsible for this behaviour and for the poor performance of cGA reported in the next section.

4.2.3 *Scalability*

In this section, we investigate the impact of the problem size on the number of fitness evaluations required to find a global optimal solution. For this experiment, we choose the One Max problem as the global optimum is unique and known in advance. Each of the algorithms is applied on the One Max problem with N varying from 50 to $2,000$ bits. The parameter settings reported in Table 4.1 were kept unchanged for all the algorithms.

Figure 4.6 shows the number of fitness evaluations as a function of the problem size N on One Max. For each algorithm, the filled symbol indicates that the global optimum was found in every single run being performed. If only some of the runs were successful, an empty symbol is used instead, and if none, the symbol is not plotted.

For small problem sizes, all of the algorithms except of cGA were always able to find the best solution. It is noted that, for almost every problem size, cGA was unable to find the global optimum in all of the runs. The number of evaluations grows exponentially for PBIL when facing a problem size of $N > 700$. It has to be noted that we have chosen $M = 10$ individuals for PBIL to give an equal number of evaluations per generation compared to $\text{vQEA}_{1,10}$. For small problem sizes, this setting seems to be very suitable, *e.g.* for $N < 600$ the average fitness evaluations required are the lowest among all other algorithms. We tried other settings for PBIL, but none of them scaled well. For example, using $M = 25$ individuals PBIL performed poorly for small problem sizes, but for $N = 1000$ all 30 runs converged to the global optimum, which required on average $30,138$ ($\sigma = 4139.4$) evaluations. In a similar way, the performance of UMDA clearly depends on the problem size. Setting the population size M to 500 is known to be suitable for high dimensional problems and the results vary as expected; for small size problems the number of fitness evaluations required is nearly double the other algorithms but for $N = 2000$, only $\text{vQEA}_{1,10}$ outperforms UMDA. $\text{vQEA}_{1,10}$ shows an almost linear increase of fitness evaluations while consistently finding the optimal solution up to a problem size of 1600. At least in this study, $\text{vQEA}_{1,10}$ demonstrates a high scalability. Furthermore, this single parameter setting appeared to be suitable for a large variety of problem sizes. No comparably robust setting was found for any of the other presented algorithms.

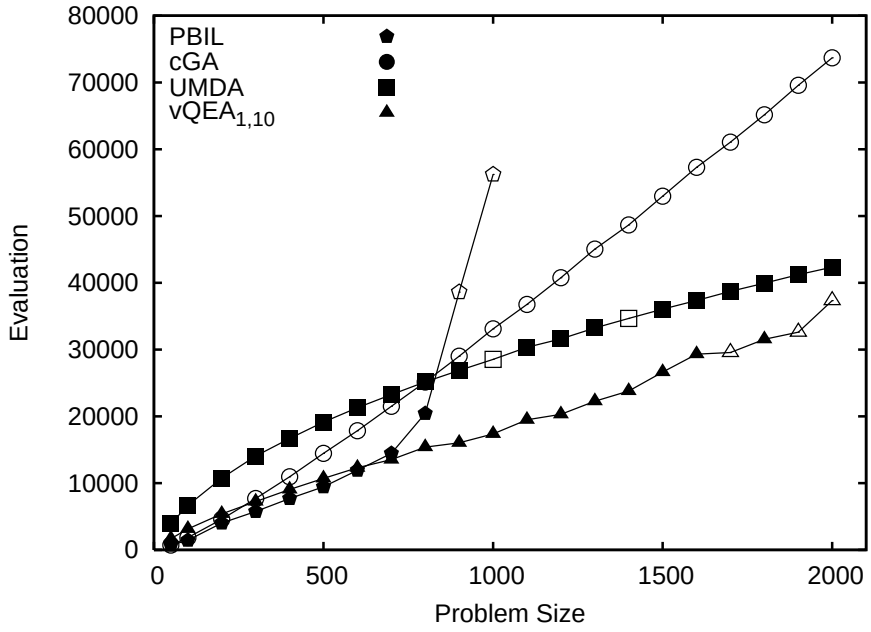


Figure 4.6: Number of evaluations as a function of N on the One Max problem

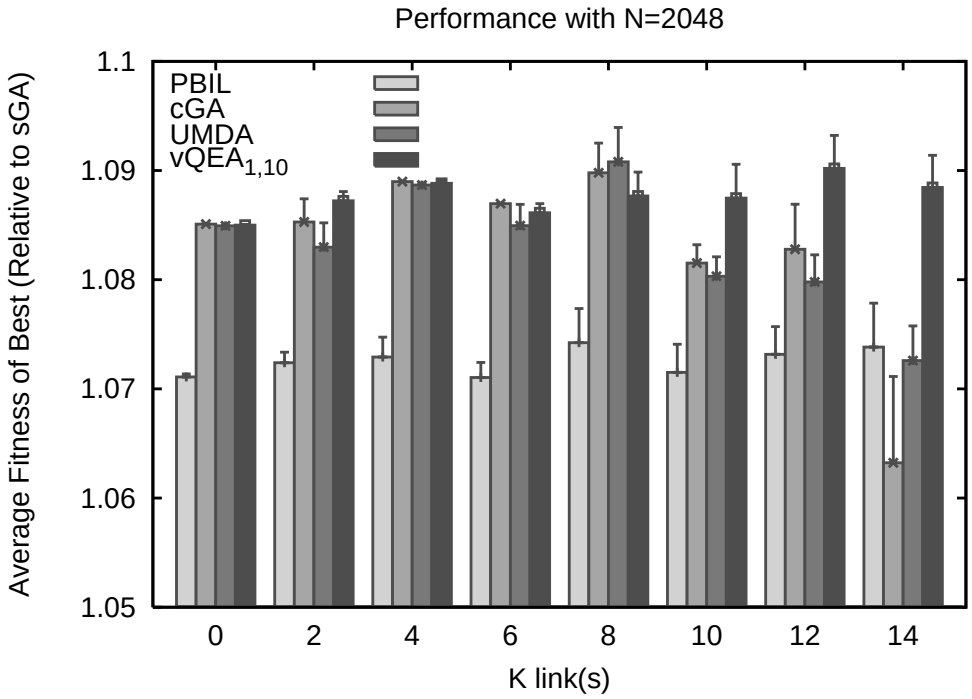


Figure 4.7: Performance on NK-landscapes N=2048

4.2.4 *Fitness*

In chapter 3, vQEA was already compared to a standard Genetic Algorithm (sGA) on NK-landscapes and was shown to be superior in terms of both speed and the

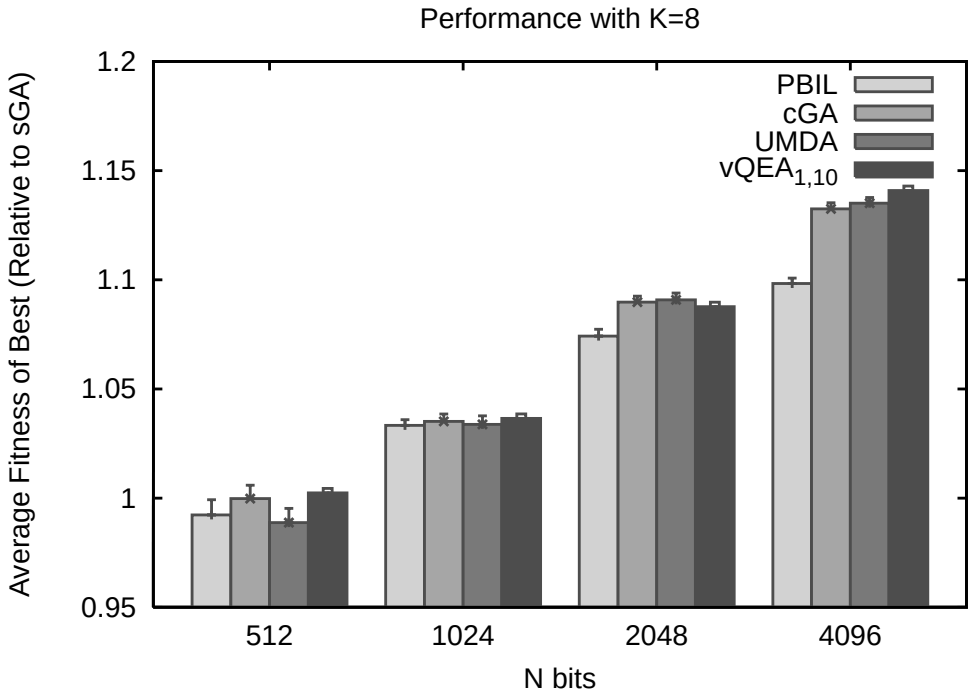


Figure 4.8: Performance on NK-landscapes K=8

quality of the solution found. In this section, we want to investigate the performance of PBIL, cGA and UMDA on the same optimisation problem. The quality of the results is presented in relation to a sGA. More precisely, the average fitness of the best solutions reached by an algorithm $\mathcal{A}$ is noted $f^*_{\mathcal{A}}$ and the relative performance of $\mathcal{A}$ is defined as the ratio $f^*_{\mathcal{A}}/f^*_{sGA}$.

In Figure 4.7, the relative performance is presented for $N = 2048$. It is clearly shown that each EDA outperforms sGA significantly. For small K (and therefore no or low level of interaction between the N variables), `PBIL` falls behind `UMDA`, `cGA` and $\texttt{vQEA}_{1,10}$ while the latter three do not show significant differences compared to each other. Nevertheless, it has to be noted that $\texttt{vQEA}_{1,10}$ shows the lowest variance in the quality of the best solution found among all the other algorithms. With $K \geq 10$, the performance of `UMDA` and `cGA` drops significantly due to the impact of the higher number of local optima in the fitness landscape. On the other hand, $\texttt{vQEA}_{1,10}$ stays rather unaffected by the problem difficulty, consistently reporting between 8 to 9% higher fitness than a sGA.

Figure 4.8 shows the average best relative fitness of several problem sizes N for fixed $K = 8$. For larger problem sizes ($N \geq 1024$) each algorithm performs significantly better than a `sGA` . Again, $\texttt{vQEA}_{1,10}$ shows the lowest variation in the final

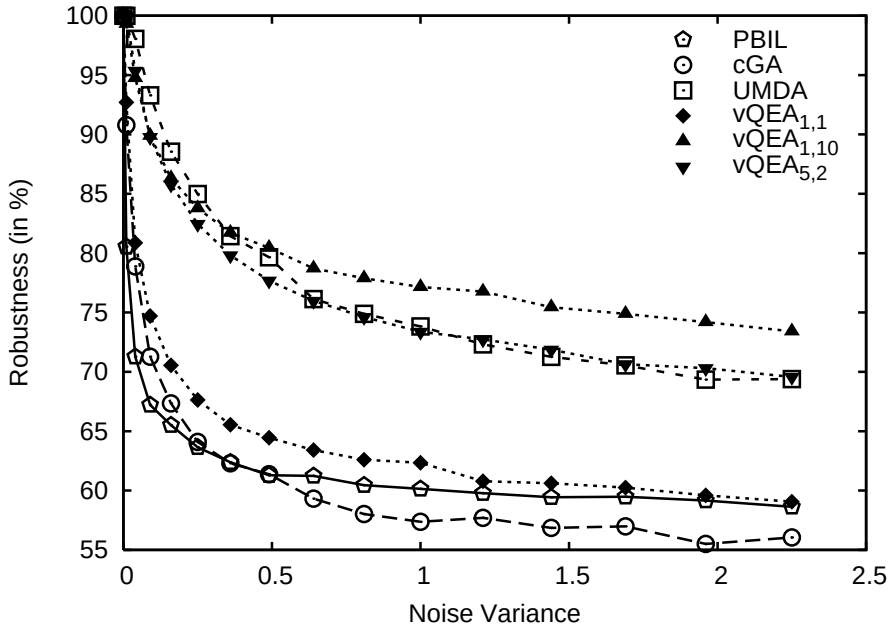


Figure 4.9: Robustness as a function of the noise rate on One Max, N=256

fitness. There are no significant fitness differences for the problem sizes $N = 512$ and $N = 1024$. For $N \geq 2048$, PBIL falls behind and, for $N = 4096$, $\text{vQEA}_{1,10}$ delivers the highest solution quality, performing slightly better than each of the other tested algorithms.

4.2.5 *Robustness*

Noise is known to be an important factor that influences Evolutionary Algorithms. Thus, the convergence robustness against fitness noise of PBIL, cGA, UMDA and vQEA is studied here. We assume a multiplicative Gaussian noise and we define the noisy fitness function F as :

$$F(x) = f(x) \cdot \mathcal{N}(1, \sigma^2) \tag{4.6}$$

with x an element of the solution space and σ^2 the noise variance. We also define the robustness $\mathcal{R}(\sigma^2)$ of an algorithm as the ratio between the average best fitness found when noise is applied ($\sigma^2 > 0$) and the average best fitness found without noise ($\sigma^2 = 0$). Experiments were performed on One Max with $N = 256$ and $\sigma \in [0, 1.5]$ and the results are presented in Figure 4.9.

For all algorithms, we know by construction that $\mathcal{R}(0) = 1.0$ and we see clearly that this robustness is strongly impacted by the increase of the noise variance. Nevertheless, we distinguish two groups of algorithms. The first includes PBIL, cGA and vQEA$_{1,1}$ and the second group is made of UMDA, vQEA$_{1,10}$ and vQEA$_{5,2}$. In the first group, as noise is introduced, the robustness decreases quickly even for small noise variance. For larger values of noise, the robustness is close to 55% which is comparable to the performance of a random search on a One Max problem. In the second group, the robustness decreases comparatively slowly and is still around 70% for $\sigma^2 = 2.25$, where vQEA$_{1,10}$ outperforms all the other algorithms tested with $\mathcal{R}(2.25) = 74\%$.

We note that cGA and vQEA$_{1,1}$ sample respectively two and one solutions per generation to update the probability vectors while with PBIL only the best among ten solutions is used. In the presence of noise, this low number of samples processed leads to decision errors. Indeed, a classical remedy known to counteract the effect of noise in EA is to perform multiple evaluations of the fitness. With UMDA $M = 500$, solutions are analysed before a learning phase occurs. This large number of evaluations before a model update is probably responsible for the convergence towards an average good solution. Population based search algorithms are also known to be robust because of their self-averaging nature. We claim that in vQEA the $\mathcal{Q}$population acts as a buffer against decision errors because $\mathcal{Q}$individuals are able to share information about the search space. Since in vQEA$_{1,10}$, all $\mathcal{Q}$individuals are embedded in the same $\mathcal{Q}$group and thus follow the same attractor ($S_{local} = 1$), the information share is maximised and therefore vQEA$_{1,10}$ is the most robust of the algorithms tested here. Moreover, we know from (Goldberg, 2002) that the interactions between variables may be seen as a form of fitness noise by the algorithms which could also explain the good results reported in the previous section for vQEA$_{1,10}$ on NK-landscapes for high degrees of apostasies ($K \geq 10$).

4.3 ROLE OF MULTIPLE MODELS

In this section, we concentrate our investigation on the role of the multiple probabilistic model $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_p\}$ in vQEA.

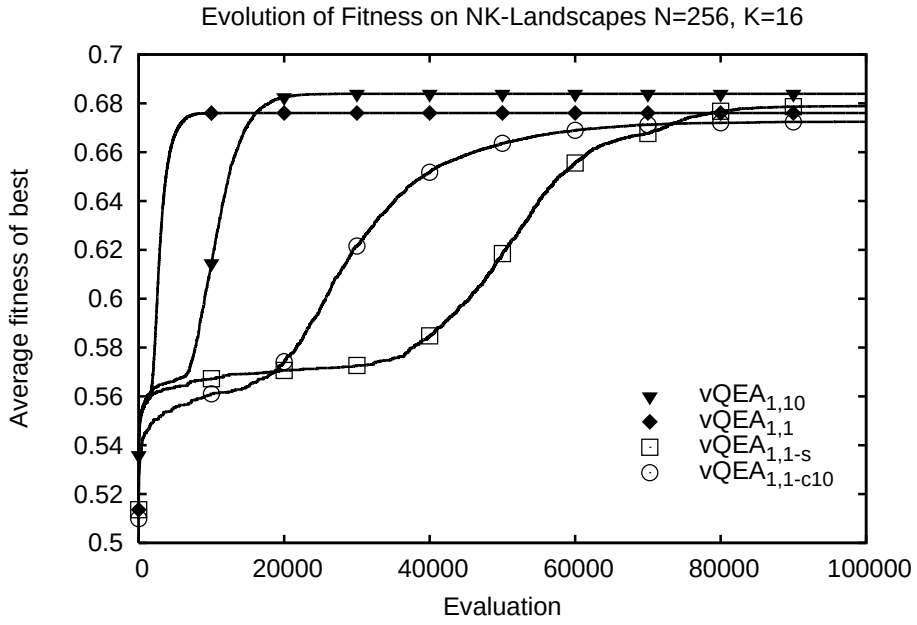


Figure 4.10: Fitness evolution of single and multiple models vQEA on NK-landscapes

4.3.1 *Do multiple models perform better than a single one?*

vQEA has been originally introduced as a coarse-grained evolutionary algorithm with several interacting $\mathcal{Q}$individuals. Nevertheless, we are not aware of any serious demonstration of the superiority of using a $\mathcal{Q}$population compared to using only a single $\mathcal{Q}$individual. A fair comparison between $\mathsf{vQEA}_{1,1}$ and $\mathsf{vQEA}_{1,10}$ (*i.e.* not based on an equivalent number of generations but on an equivalent number of fitness evaluations) is performed on One Max and NK-landscapes problems. For all the experiments carried out, the fitness of the best solution found with $\mathsf{vQEA}_{1,10}$ is better or equal to the best solution produced with $\mathsf{vQEA}_{1,1}$. As an illustration, in Figure 4.10, the average evolution of the best fitness found on NK-landscapes with $N = 256$ and $K = 16$ is plotted for $\mathsf{vQEA}_{1,1}$ and $\mathsf{vQEA}_{1,10}$ as a function of the number of evaluations. The setting of the parameters is given in Table 4.1. For both settings, the fitness improves quickly after few evaluations, while $\mathsf{vQEA}_{1,1}$ keeps a similar trend until it prematurely convergences. $\mathsf{vQEA}_{1,10}$ reports a more step-wise increase and finally reaches a higher fitness level.

In $\mathsf{vQEA}_{1,10}$, ten $\mathcal{Q}$individuals synchronise their attractors at every generation t using the best solution sampled at generation $t-1$. This setting implies that the ten corresponding probability vectors $\mathcal{P}_1, \ldots, \mathcal{P}_{10}$ are all following a unique attractor and therefore the same direction in the search space. If we assume that each model $\mathcal{P}_i(t)$

is not so different from the mean model $\overline{\mathcal{P}}(t)$, having several models instead of only one may appear redundant *a priori*. Nevertheless, the benefit of using multiple models is clear when demonstrated experimentally. Hence, we investigate two different hypothesis to explain the better results obtained with $\mathsf{vQEA}_{1,10}$.

In the first hypothesis, we assume that $\mathsf{vQEA}_{1,10}$ benefits from the fact that the search direction is chosen after sampling ten solutions, *i.e.* one per model. For that reason, we propose to produce ten solutions from the single probabilistic model and then to use the best among them as the next attractor. This algorithm is denoted $\mathsf{vQEA}_{1,1\text{-c10}}$ in Figure 4.10. We see that $\mathsf{vQEA}_{1,1\text{-c10}}$ is outperformed by $\mathsf{vQEA}_{1,1}$ and $\mathsf{vQEA}_{1,10}$ in terms of speed and average fitness of the best solution found.

In the second hypothesis, we assume that $\mathsf{vQEA}_{1,10}$ benefits from a slower convergence speed. We note that in $\mathsf{vQEA}_{1,10}$, it may happen that only one vector $\mathcal{P}_i$ out of any ten is updated during one generation t. In that case, the average model $\overline{\mathcal{P}}(t)$ moves very slowly towards the attractor and the update steps correspond to $\Delta\theta/10$. Therefore, we propose to evaluate the performance of a single $\mathcal{Q}$individual vQEA with a ten times smaller update step $\Delta\theta = 1/10 \times \pi/100$. As expected with this setting, the algorithm denoted $\mathsf{vQEA}_{1,1\text{-s}}$ in Figure 4.10 reports a slower convergence speed and outperforms $\mathsf{vQEA}_{1,1}$ in terms of fitness. The fitness increases slowly in a step-wise manner similar to $\mathsf{vQEA}_{1,10}$ but finally reaches a significantly smaller fitness value.

We have gone to great effort to reproduce results similar to $\mathsf{vQEA}_{1,10}$ using one probabilistic model only but have not been successful. Therefore, we claim that, even when they are fully synchronised (and so almost equal), the multiple probabilistic models perform better.

4.3.2 *Adaptive learning speed*

The interplay of the fully synchronised multiple models may lead to an adaptive learning speed. To illustrate this we plot the evolution of the mean model $\overline{\mathcal{P}}(t)$ when solving a one bit One Max problem for $\mathsf{vQEA}_{1,1}$, $\mathsf{vQEA}_{1,1\text{-s}}$ and $\mathsf{vQEA}_{1,10}$, see Figure 4.11. For that specific experiment the initial probability is set to $\sin^2(\Delta\theta) \sim 0$. The only difference between $\mathsf{vQEA}_{1,1}$ and $\mathsf{vQEA}_{1,1\text{-s}}$ is the setting of $\Delta\theta$ and consequently their convergence speeds. We see that for these two settings, the evolution of the probability looks like an $\arctan$ function. In particular, the shape of the two curves is identical when the probability leaves 0 or when it reaches 1. On the contrary, for $\mathsf{vQEA}_{1,10}$, an asymmetric behaviour is observed: the average probability leaves 0

much more quickly than it reaches 1. More precisely, the average probability evolves in a way similar to vQEA$_{1,1}$ when moving away from 0 and then similar to vQEA$_{1,1\text{-s}}$ when approaching 1. This is a very desired behaviour as we expect that the algorithm dedicates more effort to exploring the promising areas of the search space.

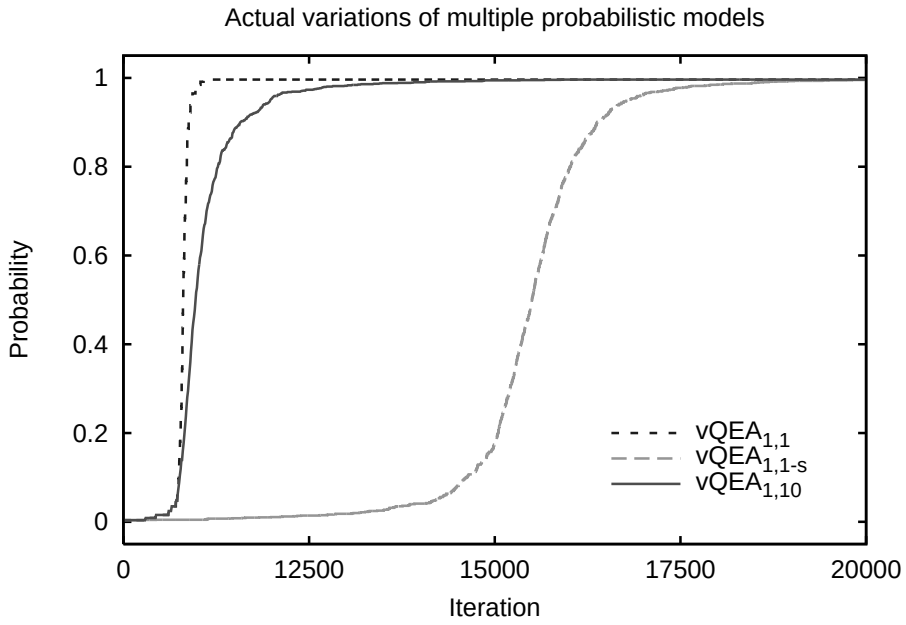


Figure 4.11: Actual variations of the mean probabilistic model observed in different configurations of vQEA

This phenomenon can easily be explained when considering the set of ten vectors $\{\mathcal{P}_1, \ldots, \mathcal{P}_{10}\}$. At the beginning of this experiment, almost every produced solution C_i is '0'. When by chance a '1' is sampled, it becomes the next attractor for the $\mathcal{Q}$population and so there is a high probability that the ten models are updated at the same time. Therefore, the learning speed of $\overline{\mathcal{P}}(t)$ can be high, *i.e.* depending on $\Delta\theta$. Afterwards, the number of models updated during one generation starts to decrease. The extreme case is when only one model is updated. This results in a much more slower learning speed for $\overline{\mathcal{P}}(t)$, *i.e.* corresponding to $1/10 \times \Delta\theta$. The situation can be seen as a voting mechanism controlling the overall learning speed. When the $\mathcal{Q}$individuals all agree that a certain direction in the search space is not appropriate their models all move away and subsequently the average model moves quickly. Conversely, when they disagree, the mean model moves very slowly, giving more accuracy and therefore more time to the algorithm to seek the right decision. This adaptive learning speed might also be responsible for the quality of the results reported for vQEA in terms of robustness to fitness noise (*cf.* section 4.2.5).

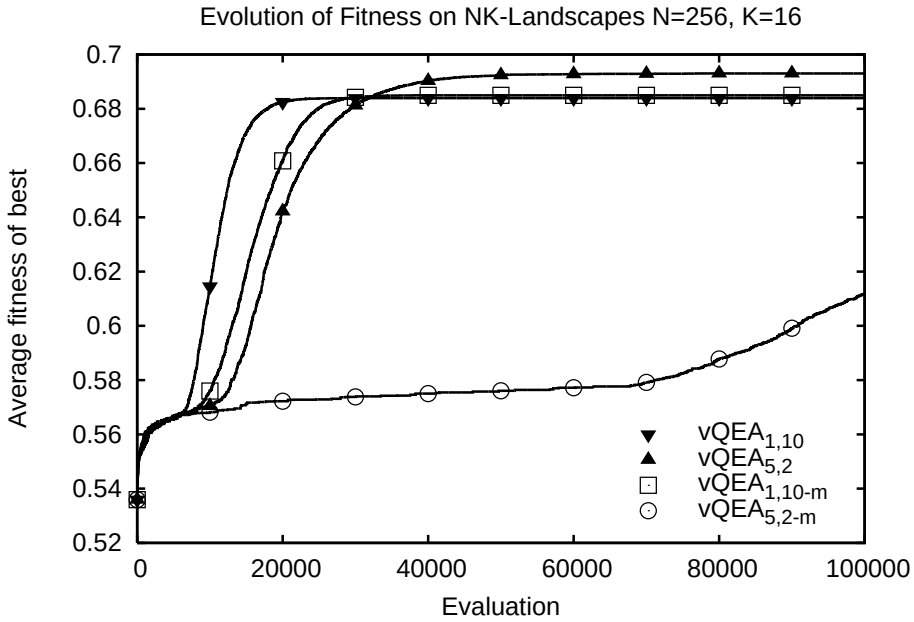


Figure 4.12: Fitness evolution of mean and multiple models vQEA on NK-landscapes

4.3.3 *Do multiple models perform better than a mean model?*

In the two previous sections, it was assumed that in $\mathsf{vQEA}_{1,10}$ the models $\mathcal{P}_1, \ldots, \mathcal{P}_{10}$ are almost identical at time t and therefore equivalent to the mean model $\overline{\mathcal{P}}(t)$. Subsequently, it was assumed that the distribution of solutions in the set $\{C_1, \ldots, C_{10}\}$ sampled from the ten models at time t was to some extent equivalent to the distribution obtained when sampling ten solutions from $\overline{\mathcal{P}}(t)$. We now evaluate the validity of this assumption for $\mathsf{vQEA}_{1,10}$ but also for $\mathsf{vQEA}_{5,2}$.

For that purpose, we introduce two variants, denoted $\mathsf{vQEA}_{1,10\text{-m}}$ and $\mathsf{vQEA}_{5,2\text{-m}}$, where the mean model is used for sampling. More precisely, the overall structure and settings of the algorithm are kept unchanged except that a mean model $\overline{\mathcal{P}}(t)$ is computed every generation and then used to produce the individual solutions $C_i(t)$. In particular, it is noteworthy that the adaptive learning speed described earlier works for these two variants as well. Therefore, any noticeable variation in the performance of the algorithm is only caused by the use of the mean model. In Figure 4.12, the average evolution of the best fitness found on NK-landscapes with N=256 and K=16 is plotted as a function of the number of evaluations.

We note that the two curves obtained for $\mathsf{vQEA}_{1,10}$ and $\mathsf{vQEA}_{1,10\text{-m}}$ are very similar and their average final fitness values are statistically identical. Notwithstanding the slightly faster convergence of $\mathsf{vQEA}_{1,10}$ compared to $\mathsf{vQEA}_{1,10\text{-m}}$, the assump-

tion made in the previous section seems to hold on this problem: sampling ten fully synchronised models $\mathcal{P}_i$ is indeed comparable to sampling the corresponding mean model $\overline{\mathcal{P}}$. The situation is clearly not the same for $\mathsf{vQEA}_{5,2}$. While $\mathsf{vQEA}_{5,2}$ is the best setting of vQEA tested on this problem, $\mathsf{vQEA}_{5,2\text{-m}}$ reports extremely poor results. Therefore, it is claimed that when they are not fully synchronised the multiple probabilistic models can perform better than the mean model.

In $\mathsf{vQEA}_{5,2}$, five $\mathcal{Q}$groups each containing two fully synchronised $\mathcal{Q}$individuals are evolved and the best attractor among the groups is shared, according to the parameter S_{global}, every 100$^{\text{th}}$ generation. However, in vQEA the attractors are systematically replaced at every generation, so that a given synchronisation can affect the evolution of the $\mathcal{Q}$groups during a single generation only. As a consequence, the groups can evolve separately towards different regions of the search space. With $K = 16$ epistatic links in the problem, the interactions between the 256 variables are important and the problem is not easy to solve. With $\mathsf{vQEA}_{5,2}$ each $\mathcal{Q}$group can specialise on different patterns of bits and the multiple models of vQEA allow sampling a more complex distribution of solutions than with a single probability vector.

4.3.4 *Measuring diversity*

In order to measure the diversity of the solutions sampled by the multiple models in vQEA, the variance $v(t)$, as defined in Equation 4.4, is not necessarily adapted. Actually, in section 4.2.2, the variance was computed using the mean model $\overline{\mathcal{P}}(t)$ but clearly this procedure does not consider the conditional probabilities and is not sufficient to represent interactions among variables. Thus, the more the vectors $\mathcal{P}_1, \ldots, \mathcal{P}_p$ differ at time t the less the variance $v(t)$ is suitable. Hence, we propose another approach where two metrics are used to represent the diversity of the solutions produced at time t: the convergence of the $\mathcal{Q}$population denoted $Conv(t)$, and the pairwise distance between the $\mathcal{Q}$individuals denoted $Dist(t)$.

The convergence of a $\mathcal{Q}$population reflects how the N $\mathcal{Q}$bits have converged in the whole population. We define $Conv^j$, the $\mathcal{Q}$bit convergence at locus j,

$$Conv^j = \frac{2}{p} \sum_{i}^{p} \left| \mathcal{P}_i^j - \frac{1}{2} \right| \tag{4.7}$$

and so the convergence of the $\mathcal{Q}$population corresponds to the mean $\mathcal{Q}$bit convergence over N $\mathcal{Q}$bits such that :

$$Conv = \frac{1}{N}\sum_{j}^{N} Conv^{j} \tag{4.8}$$

The pairwise distance between the $\mathcal{Q}$individuals reflects how their probabilistic models differ. To represent the distance $Dist_{i,k}$ between two probability vectors $\mathcal{P}_i$ and $\mathcal{P}_k$, we propose to simply compute:

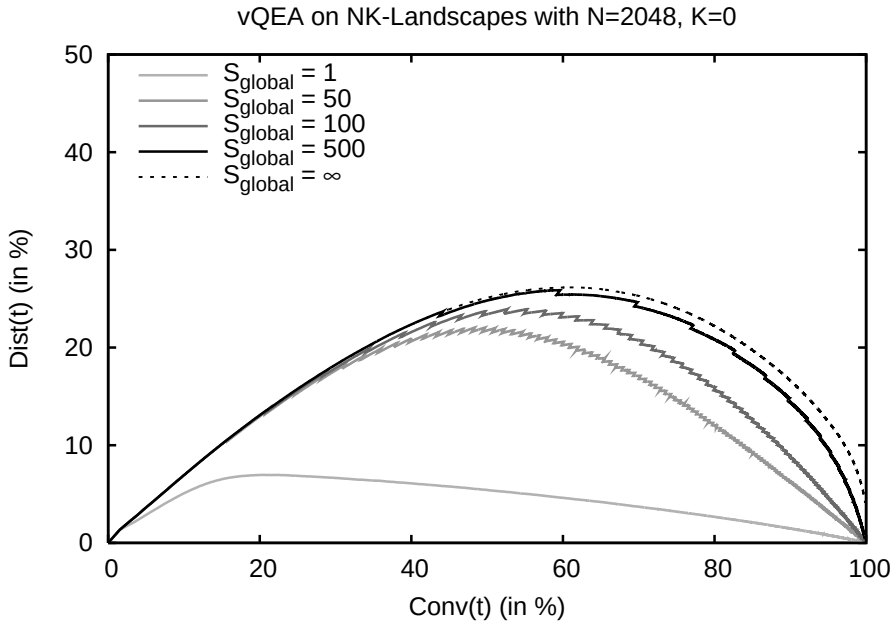
$$Dist_{i,k} = \frac{1}{N}\sum_{j}^{N} \left|\mathcal{P}_i^j - \mathcal{P}_k^j\right| \tag{4.9}$$

Following (Wineberg & Oppacher, 2003), this metric can be easily interpreted as the proportion of mutational changes required to transform a set of solutions sampled from $\mathcal{P}_i$ to a set of solutions sampled form $\mathcal{P}_k$. Hence, the pairwise distance between p $\mathcal{Q}$individuals corresponds to:

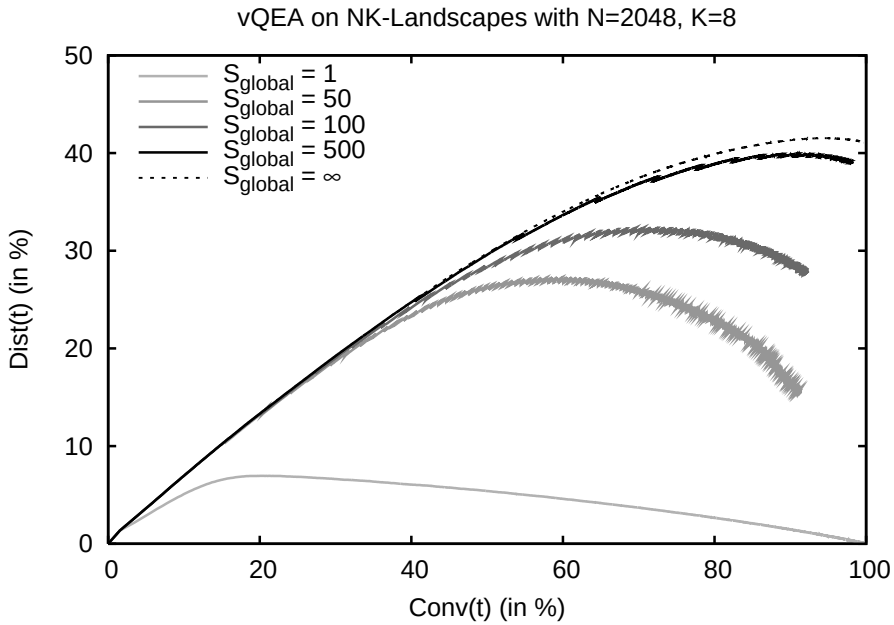
$$Dist = \frac{2}{p(p-1)}\sum_{i=1}^{p}\sum_{k=i+1}^{p} Dist_{i,k} \tag{4.10}$$

The evolution of $Conv(t)$ together with $Dist(t)$ on NK-landscapes with N=2048 for K=0 and K=8 was computed. The setting of $\mathsf{vQEA}_{5,2}$ was described in Table 4.1, *i.e.* five $\mathcal{Q}$groups of two synchronised $\mathcal{Q}$individuals are evolved and the best attractor is shared according to S_{global}. Furthermore, the influence of the global synchronisation period S_{global} was also investigated. The results averaged over 30 independent runs of 10^5 evaluations are plotted in Figure 4.13.

For every curve, a common trend is reported. After the initialisation phase, each $\mathcal{Q}$individual defines a probability vector $\mathcal{P}_i$ whose elements are all set to $1/2$ and therefore $Conv(0)$ and $Dist(0)$ are both equal to 0. At that particular time, the diversity of the solutions sampled is maximum. Then, under the effects of selection (together with drift on neutral problems), the $\mathcal{Q}$population starts to converge with $Conv(t) > 0$ and the probabilistic models become more and more different until $Dist(t)$ reaches a maximum. Finally, the pairwise distance decreases while the $\mathcal{Q}$population keeps converging continuously. As expected, S_{global} determines the amplitude of the peak of maximum distance between the multiple models. With S_{global}=1, the models are fully synchronised (as in $\mathsf{vQEA}_{1,10}$). For both $K = 0$ and $K = 8$, the maximum value for $Dist(t)$ with $S_{global} = 1$, is approximately 7%. This



(a) NK-landscapes using $N = 2048$, $K = 0$



(b) NK-landscapes using $N = 2048$, $K = 8$

Figure 4.13: Pairwise Distance between $\mathcal{Q}$individuals *vs* Convergence of the $\mathcal{Q}$population on NK-landscapes . Results are shown from a single typical run.

very low value means that the multiple models represent subspaces (hypercubes) that differ by 7% of their bits. With higher values for S_{global}, the $\mathcal{Q}$groups are more likely to evolve towards different regions of the search space and the maximum value for $Dist(t)$ increases.

When the multiple models are not fully synchronised, *i.e.* $S_{global} > 1$, we note major discrepancies between the case $K = 0$ and $K = 8$, *cf.* Figures 4.13a and 4.13b respectively: For $K = 0$, the maximum value for $Dist(t)$ is around 25% when the attractors are never synchronised ($S_{global} = \infty$), *i.e.* when the five $\mathcal{Q}$groups evolve separately. After 10^5 evaluations, we have $Conv(t) = 1$ and $Dist(t) = 0$, even for $S_{global} = \infty$. In this situation, the five $\mathcal{Q}$groups have converged towards the same solution of the search space. We observe also a saw-tooth shape of the curves where each tooth corresponds to an episode of synchronisation of the attractors. Given that $K = 0$, there is no local optima, so the information carried by the attractors is not contradictory and is therefore exchanged smoothly between the $\mathcal{Q}$groups.

For $K = 8$, the maximum value for $Dist(t)$ is around 40% for $S_{global} = \infty$. After 10^5 evaluations, $Dist(t)$ is not equal to zero and for $S_{global} = 50$, $S_{global} = 100$ and $S_{global} = 500$ the $\mathcal{Q}$population has not converged. The saw-tooth shape disappears and instead the curves are very rugged, in particular with $S_{global} = 50$. With $K = 8$, the information carried by attractors can be contradictory and therefore not easily exchanged between the $\mathcal{Q}$groups, tending to slow down the convergence speed of the $\mathcal{Q}$population. Nevertheless, as long as the best performance in terms of fitness is obtained for $S_{global} = 100$, some information is exchanged through the synchronisation process. Thus, multiple models in vQEA allow a more diverse exploration of the search space than with only a single model.

4.4 CONCLUSION

Behind the quantum metaphor, vQEA is an original approach that belongs to the class of EDAs. It clearly shares some common features with several simple EDAs such as PBIL and cGA, but its performance is more similar to UMDA, particularly with regard to the loss of diversity, the scalability and the robustness to fitness noise. Therefore, vQEA should benefit from prior work on simple EDAs where interactions between variables are not taken into account.

The main differentiating feature of the vQEA is the multi-model approach. In this chapter, the advantages of manipulating several probability vectors instead of only one were empirically demonstrated. First, vQEA is an effective algorithm that works with fairly generic settings of the control parameters for a collection of benchmark problems of various sizes, with different levels of interactions between variables and numbers of neutral dimensions. Note that no particular efforts have been dedicated to finding the best possible settings of vQEA, but rather a setting directly borrowed

from previous work on QEA was used, even though the behaviour of QEA is quite dissimilar to vQEA, and it was investigated on a very different test problem.

Second, the $\mathcal{Q}$population buffers against a finite number of decision errors making vQEA robust against fitness noise.

Finally, we have shown that vQEA can perform better than other simple EDAs when links are introduced between variables. These interesting results about the multi-model approach in vQEA can be explained by the adaptive learning speed and a more diverse sampling of the search space compared to other EDAs with a single probability vector. Future work might compare the mechanisms of existing multi-model EDAs approaches (Zhang et al., 2002; Ahn, Kim, & Ramakrishna, 2003b; Ahn, Goldberg, & Ramakrishna, 2003; delaOssa et al., 2006; Madera et al., 2006; S. Zhou & Sun, 2005a) to the one used in vQEA and evaluate their relative performance empirically.

The way the $\mathcal{Q}$population is structured, *i.e.* number and size of the $\mathcal{Q}$groups together with the local and global synchronisation periods, directly controls the adaptive learning speed and the diversity of the solutions sampled by vQEA. To properly choose this structure, we suggest the following approach. First of all, the $\mathcal{Q}$individuals should be fully synchronised in a $\mathcal{Q}$group (with $S_{local} = 1$) of size k in such a way that k determines the variation of the learning speed from $\Delta\theta/k$ to $\Delta\theta$. Second, several $\mathcal{Q}$groups should be introduced as long as the problem is known to report a significant number of local optima or similarly a significant level of dependency between the variables. Then the global synchronisation period controlling the diversity of the sampled solutions can be set inversely to the size of the problem.

Despite the scalability of vQEA, the generic setting proposed in this study is probably not optimal and therefore a general expression should be proposed. In particular, the optimal setting of $\Delta\theta$ according the size of the problem is still unknown and inasmuch as $\Delta\theta$ gives the fastest learning speed, its setting should be investigated, at least empirically, for example on a simple One Max problem.

We have seen that one of the strengths of vQEA comes from the specialisation of $\mathcal{Q}$groups on diverse sub-spaces. Actually, only the stochastic behaviour of the $\mathcal{Q}$individuals driven by fitness selection makes the $\mathcal{Q}$groups converge towards different regions of the search space. So far, even with a very low synchronisation frequency, we can not guarantee the diversity of the sampling for every problem. This question should be explored so that extra mechanisms for ensuring specialisation can eventually be added.

The impact of the synchronisation events on the probabilistic models has been shown to be rather limited. Nevertheless, the synchronisation of attractors defini-

tively helps the multiple $\mathcal{Q}$individuals to optimise non-decomposable problems. So far, the extent to which these problems can be solved using vQEA remains unclear. From the experiments presented in this study, one may conclude that the performance of vQEA is somewhere in between the one reported by the simple and the complex EDAs. Therefore, the efficiency of vQEA in terms of exchange of information and mixing building blocks should be addressed in future work, for example using a flexible benchmark such as the Random Additively Decomposable Problems (Pelikan, Sastry, Butz, & Goldberg, 2006), in which the variable interactions can be explicitly controlled and also the global optimum is known.

Chapter 5

EXPLORING NOISY SEARCH SPACES WITH VQEA

Noise is a typical property of most real world problems. Thus, the capability of an optimisation method to handle noisy or inaccurate information obtained from the fitness criterion is generally regarded as a very important pre-condition for a successful application of the method for real world applications. In chapter 4, vQEA has demonstrated promising results especially on problems with higher epistasis. The relationship between epistasis and fitness noise has been claimed many times in literature and epistasis may be interpreted as a certain form of noise (Goldberg, 2002). Due to this connection, we discuss the robustness of vQEA in this chapter.

The analysis of EAs optimising noisy fitness functions is the focus of many current research papers. The main effects of fitness noise are described as the decrease of convergence velocity and a residual location error of the optimum in the search space (Beyer, 2000). Noise can also introduce false optima in the fitness function, a phenomenon first described as noise-induced multi-modality in (Sendhoff, Beyer, & Olhofer, 2002) and studied comprehensively in (Beyer & Sendhoff, 2006). An excellent survey of recent developments in the field of noise-related optimisation can be found in (Jin & Branke, 2005).

One known remedy against fitness noise is the explicit and implicit averaging (Fitzpatrick & Grefenstette, 1988). The general idea of explicit averaging is to estimate the quality of a given solution by explicitly computing the average of several (noisy) fitness evaluations. Early studies have proposed the use of an adaptive scheme, *e.g.* in (Aizawa & Wah, 1993) and (Aizawa & Wah, 1994) for the genetic algorithm, in which the sample number increases in later generations of the evolutionary process. The implicit averaging suggests an increase of the population size of the used EA. The effects of noise on the evaluation of a certain solution are then likely to be compensated through the evaluation of a similar solution. It was shown in (Miller & Goldberg, 1996) that noise has no effect on proportional selection if the population size is infinite. Another possibility to cope with fitness noise is to modify of the selec-

tion process. In (Markon, Arnold, Back, Beielstein, & Beyer, 2001), a threshold for comparing the quality of two solutions in an Evolutionary Strategy was introduced where an offspring individual has to demonstrate a considerably better fitness in order to replace its parents. An optimal normalised threshold was found on the noisy sphere problem. In this study we want to investigate whether the multiple probabilistic model used in vQEA is beneficial in the context of noisy search spaces.

Many studies address additive noise of constant strength, *i.e.* all solutions in the search space are equally impacted by noise. For real world problems the assumption of constant noise levels is not necessarily true and indeed one can imagine many applications in which the noise is directly related to an area in the search space and thus not constant. More specifically, some solutions may suffer more due to noise than others. One example is the measurement error of physical devices metering properties like temperature or light intensity. This error is generally given as a proportion of the actual measurement. An error of $\pm 5\%$ results in a low absolute noise level for low-valued measurements and a higher one for high-valued measurements. A theoretical analysis on proportional noise was undertaken by (Arnold & Beyer, 2003) to compare Evolutionary Strategies to direct search methods. In their model, the noise was considered to be proportional to the fitness function. In (Di Pietro, While, & Barone, 2004), the use of *noise landscapes* was suggested. These define the noise level as dependent on the given fitness landscape. Different noise levels can be associated with any solution in the search domain. In this chapter, a general model based on noise landscapes is proposed and used to experimentally analyse the impact of noise on vQEA and to compare its performance to three classical EDA, namely UMDA, cGA and PBIL, as previously discussed in chapter 4.

The rest of the chapter is organised as follows: In section 5.1 we present the noise model used, followed by the experimental analysis in section 5.2. The benchmark problems along with their motivation and suitability are discussed and the obtained results are presented, followed by the discussion and conclusion of this chapter.

5.1 NOISE MODEL

Similar to the noise landscapes suggested in (Di Pietro et al., 2004) and the proportional noise in (Arnold & Beyer, 2003), a general noise model is presented, allowing the application of any noise level to any solution in the search domain.

Let x be a solution in the search space and $f(x)$ a fitness function measuring the quality of x. The following noise model is defined:

$$F(x) := f(x) + \sigma_m \, \delta(x) \, \mathcal{N}(0,1) \tag{5.1}$$

where $\delta(x) : x \mapsto [0,1]$ is a function describing the proportion of the maximum noise strength σ_m at each point in the search domain so that the product $\sigma_m \, \delta(x)$ defines the noise variance associated with a given solution x. $\mathcal{N}(0,1)$ is a normal distributed random variable. We note that, in this model, the noise depends on the location in search space. Therefore the level of noise may be different at any point in the search space. Nevertheless, it is also possible to model constant additive Gaussian noise by setting $\delta(x) = c$, $c \in [0,1]$.

In this study several functions $\delta(x)$ are investigated:

1. Constant noise – The noise level is constant at every point in the fitness landscape:

$$\delta(x) = c, \quad c \in [0,1] \tag{5.2}$$

2. Linear Noise – The noise level depends on the fitness landscape, *i.e.* the noise increases/decreases linearly in areas of higher fitness levels:

$$\delta(x) = af(x) + b \tag{5.3}$$

where $a, b \in \mathbb{R}$. It is worth noting that this noise type is also called multiplicative noise.

3. Cosine Noise – The noise level varies periodically over the fitness landscape:

$$\delta(x) = 0.5 \cos(\omega \, \pi \, f(x)) + 0.5 + b \tag{5.4}$$

where $\omega > 0$ controls the periodicity and b the minimum noise strength.

5.2 EXPERIMENTS

In this study we experimentally compare the behaviour of cGA (Harik et al., 1999), PBIL (Baluja, 1994) and UMDA (Mühlenbein & Paass, 1996) to several configurations of vQEA. The experiments are performed on three benchmark problems each

having different characteristics. The first is the simple bit counting problem (One Max) consisting in maximising the number of ones of a bit string. Here the fitness is normalised to be in the interval $[0, 1]$ by dividing the fitness by the problem size N. More formally, the problem is described as finding a bit vector $x = \{x_1, x_2, \ldots, x_N\}$, with $x_i \in \{0, 1\}$, that maximises the equation $f_{\text{om}}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i$. One Max has only a single optimum but also some neutral dimensions, since two different bit strings may have the same fitness value. The phenomenon of genetic drift is more likely to impact the search causing a faster loss of diversity.

The other two problems belong to the family of NK-landscapes first introduced in (Kauffman, 1993). NK-landscapes are stochastically generated fitness functions parametrised by N indicating the number of variables (problem size), and K which defines the number of interactions between these variables. NK-landscapes do not have any neutral dimensions, so all solutions in the search space have a unique fitness value being in the interval $[0, 1]$. For $K \geq 1$ the optimisation problem is NP-complete as shown in (Weinberger, 1996). For $K = 0$ this fitness function has only a single optimum. Increasing K results in an increasingly rugged fitness landscape with many local optima (*i.e.* $\frac{2^N}{N+1}$ optima for K=$N - 1$). We chose $K = 0$ and $K = 4$ with $N = 256$ as two representatives for this study.

5.2.1 *Settings*

Four different noise landscapes are considered. As the fitness functions for all of the problems are normalised to the same interval, we can use the same noise model parameters for each problem.

- *Constant Noise* – This noise model assumes the same noise strength σ_m for any solution in the search space. Therefore we chose according to equation (5.2) $c = 1$.

- *Positive Linear Noise* assumes a linearly increasing noise strength for better solutions, more precisely the better the solution the higher the noise. It is noteworthy that this type of noise is also referred to as multiplicative noise. We chose $a = 1$ and $b = 0$ as parameters for equation (5.3).

- *Negative Linear Noise* assumes a linearly decreasing noise strength for better solutions, more precisely the better the solution the lower the noise. We chose $a = -2$ and $b = 2$ as parameters for equation (5.3).

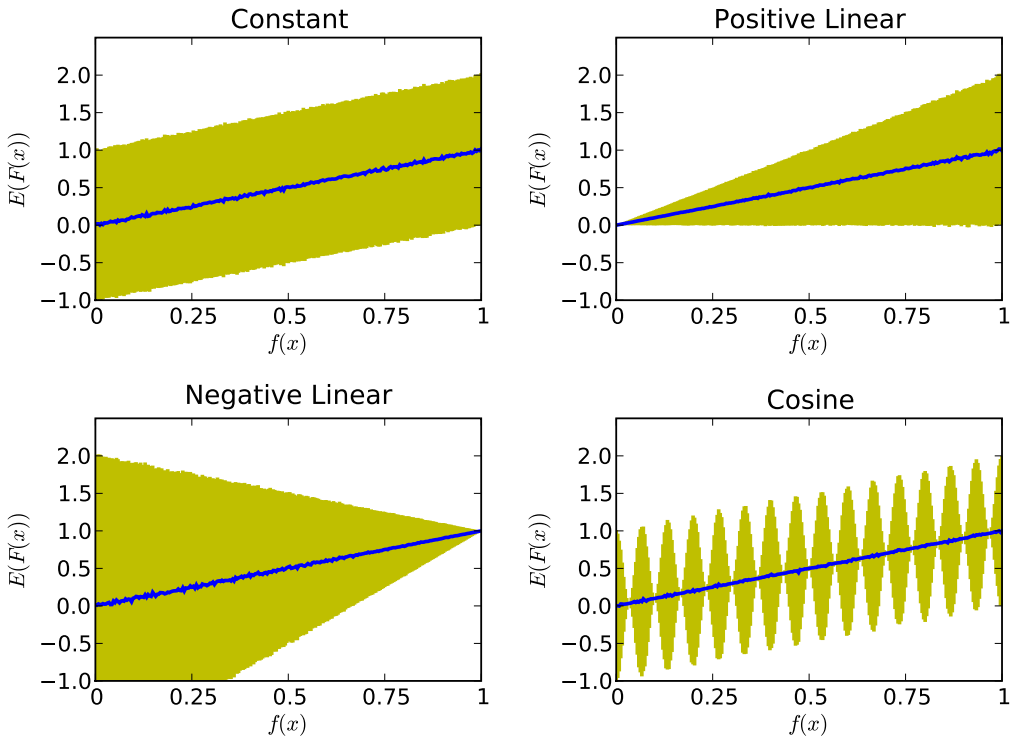


Figure 5.1: Expected value of $F(x)$ as a function of the (noise free) fitness function $f(x)$ used in the One Max problem of size $N = 256$. The error bars (in light col or) represent the standard deviation of 10,000 samples drawn from $F(x)$ for each of the possible 256 fitness values $f(x)$.

- *Cosine Noise* – In this model the noise strength changes with a certain frequency ω. We parametrised equation (5.4) with $\omega = 30$ with minimum noise strength $b = 0$.

In Figure 5.1, the expected values $E(F(x))$ were computed for distinct fitness values $f(x)$ of a normalised One Max problem of size $N = 256$. Each $E(F(x))$ is obtained by averaging 10,000 samples drawn from $F(x)$ for each of the possible 256 fitness values $f(x)$. The impact of the different noise types on the fitness landscape is clearly visible in the figure.

All experiments were performed using varying maximum noise, *i.e.* $\sigma_m = 0, 0.2, 0.4, \ldots, 2$. For each experiment, 50 runs were performed and the results averaged.

In each run, the best solution (in terms of $F(x)$) of the last generation is chosen as a representative for the best solution found in an algorithm. It is noteworthy that the global best solution found during the entire run (in terms of $F(x)$) usually can not be used as a representative since in some noise landscapes the noise strength is the

highest at an early stage of the run. Therefore, the global best solution would most likely represent the solution with the highest noise level although the algorithm might have converged to a different area in the search space.

It is well known that search algorithms should be carefully tuned according to the level of noise of the problem to solve (Goldberg, Deb, & Clark, 1991). Therefore, a comprehensive parameter study was undertaken in which different configurations of the algorithms were tested, in order to identify the best settings for each noise landscape on all three benchmark problems. For PBIL, 60 different combinations of population size and learning rate were considered. The virtual population size in cGA was explored in 28 different settings. As UMDA requires only the proper adaption of the population size, ten different sizes were investigated here. vQEA is a coarse-grained algorithm allowing a complex structure for the population of $\mathcal{Q}$individuals. Four structural settings were investigated: a single $\mathcal{Q}$individual (vQEA$_{1,1}$), one group of ten fully synchronised $\mathcal{Q}$individuals (vQEA$_{1,10}$), five groups of two $\mathcal{Q}$individuals (vQEA$_{5,2}$) and ten groups of one $\mathcal{Q}$individual (vQEA$_{10,1}$). All structures employ the default H_ϵ gate as described in chapter 3. For each structure, the rotation angle $\Delta\theta$ and the global synchronisation period S_{global} were explored, totalling 84 different configurations. All algorithms were allowed to perform 10^5 fitness evaluations. The complete parameter study is presented in Appendix C. As a result of this analysis, all tested algorithms are optimally configured for the presented problems. The settings for all methods are summarised in Table 5.1.

5.2.2 *Results*

Figure 5.2 shows the results for the different noise landscapes for the NK-landscapes and $K = 0$. As expected, the performance of all algorithms is impacted by increasing levels of noise leading to an asymptotic convergence of the fitness towards a minimum comparable to the performance of a random search.

In the case of constant noise, all of the algorithms have similar performance, with the exception of PBIL. For a positive linear noise landscape, vQEA$_{1,10}$ is significantly more robust than any other tested algorithm, followed by vQEA$_{5,2}$, vQEA$_{1,1}$ and UMDA, which deliver both almost identical results. PBIL demonstrates an interesting behaviour for noise levels above $\sigma_m > 1$ as the robustness decreases significantly more slowly than in other methods. This effect is caused by the mutation operator resulting in the inability of the probability vector to converge to some solution. In the positive linear noise landscape, the level of noise is correlated to the

OneMax				
	constant	linear	inverse linear	cosine
PBIL	$M = 10$ $R_l = R_s = 0.05$	$M = 100$ $R_l = R_s = 0.05$	$M = 10$ $R_l = R_s = 0.25$	$M = 10$ $R_l = R_s = 0.01$
cGA	$n = 250$	$n = 250$	$n = 200$	$n = 190$
UMDA	$M = 500$	$M = 500$	$M = 500$	$M = 500$
vQEA$_{1,10}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.03\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{1,1}$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{10,1}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 10$	$\Delta\theta = 0.005\pi$ $S_{glob} = 5$	$\Delta\theta = 0.02\pi$ $S_{glob} = 25$	$\Delta\theta = 0.01\pi$ $S_{glob} = 25$
vQEA$_{5,2}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 25$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 50$	$\Delta\theta = 0.005\pi$ $S_{glob} = 25$
NK-landscapes, $K = 0$				
	constant	linear	inverse linear	cosine
PBIL	$M = 10$ $R_l = R_s = 0.01$	$M = 100$ $R_l = R_s = 0.05$	$M = 50$ $R_l = R_s = 0.25$	$M = 10$ $R_l = R_s = 0.25$
cGA	$n = 190$	$n = 190$	$n = 190$	$n = 190$
UMDA	$M = 500$	$M = 500$	$M = 400$	$M = 500$
vQEA$_{1,10}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.01\pi$ $S_{glob} = 1$	$\Delta\theta = 0.01\pi$ $S_{glob} = 1$
vQEA$_{1,1}$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{10,1}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 100$	$\Delta\theta = 0.005\pi$ $S_{glob} = 300$
vQEA$_{5,2}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 75$	$\Delta\theta = 0.03\pi$ $S_{glob} = 300$
NK-landscapes, $K = 4$				
	constant	linear	inverse linear	cosine
PBIL	$M = 50$ $R_l = R_s = 0.05$	$M = 70$ $R_l = R_s = 0.05$	$M = 10$ $R_l = R_s = 0.01$	$M = 10$ $R_l = R_s = 0.01$
cGA	$n = 220$	$n = 180$	$n = 160$	$n = 180$
UMDA	$M = 300$	$M = 500$	$M = 500$	$M = 500$
vQEA$_{1,10}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 1$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.02\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{1,1}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{10,1}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 50$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.03\pi$ $S_{glob} = 50$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 10$
vQEA$_{5,2}$	$\Delta\theta = 0.03\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.02\pi$ $S_{glob} = 50$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 10$

Table 5.1: Parameter settings

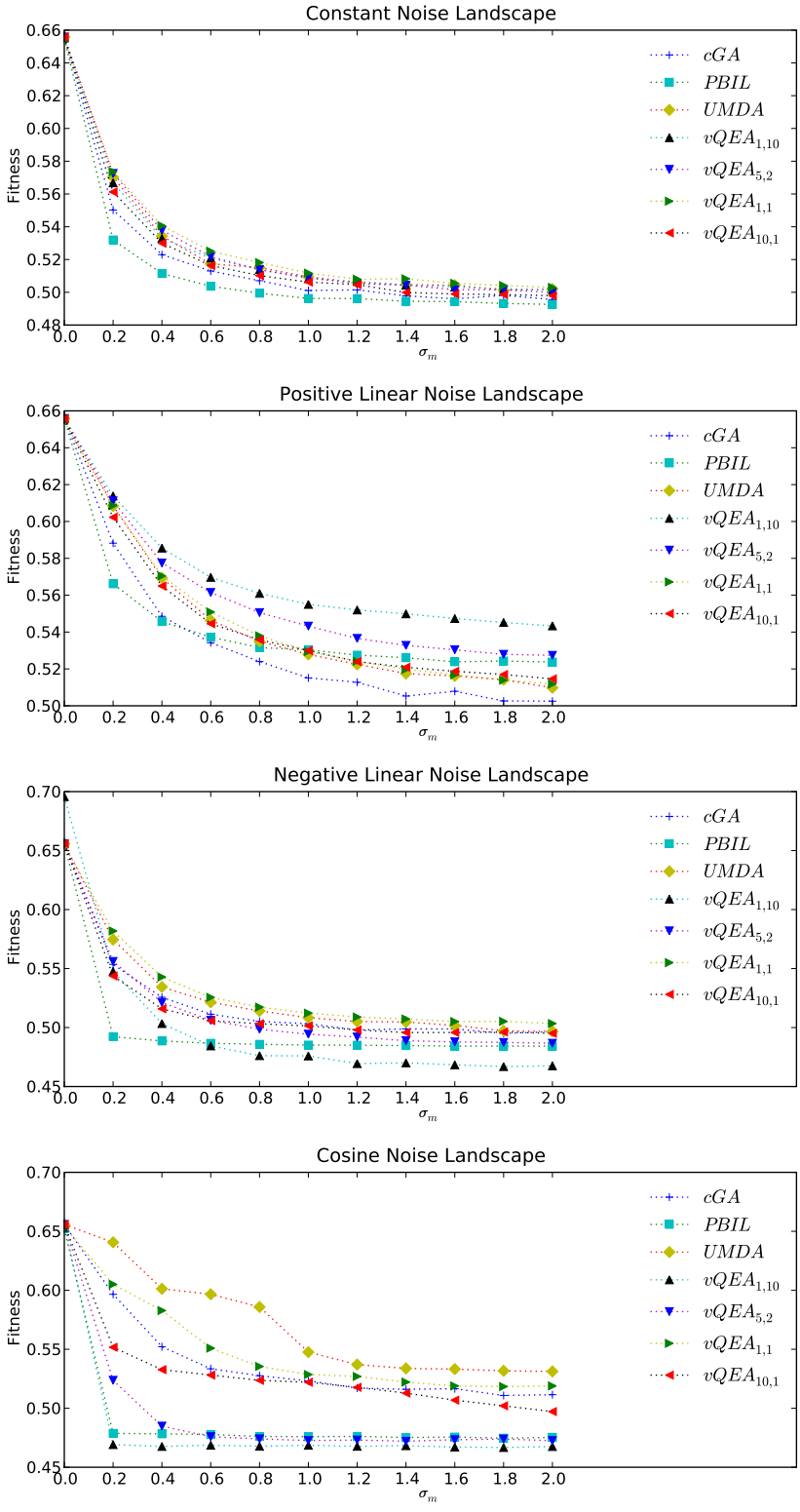


Figure 5.2: Robustness on NK-landscapes problem, K=0

fitness landscape, since an increased real fitness corresponds at the same time to an increased noise level. Being guided by a solution with the highest noisy fitness $F(x)$ in every generation, PBIL is strongly attracted by areas with high noise levels. $\text{vQEA}_{1,10}$ behaves in a very similar way and climbs the noise landscape efficiently. In the case of the positive linear noise landscape this strategy results in an advantage although following the highest noise in the search space might not appear advantageous in general.

This assumption is supported by analysing the results for the negative linear noise landscape. Climbing the noise landscape results here in a decrease of fitness and is hence misleading. Indeed, the previously best $\text{vQEA}_{1,10}$ delivers the worst performance on this problem. The algorithm efficiently maximises $F(x)$ and hence minimises $f(x)$ at the same time. The best performance is reported by $\text{vQEA}_{1,1}$ followed by UMDA and $\text{vQEA}_{10,1}$.

The cosine shaped noise landscape belongs to the category of misleading noise as well. Here the noise level is periodically changing with an increasing fitness level resulting in the maximum noise strength for many different solutions in the search space. These locally highest noise levels are known as noise optima. Several noise optima exist. Here the averaging strategy implemented by UMDA delivers a superior result, followed by $\text{vQEA}_{1,1}$, cGA and $\text{vQEA}_{10,1}$. Once more PBIL and $\text{vQEA}_{1,10}$ are quickly trapped on noise induced maxima and are unable to escape. Both deliver the worst robustness in this experiment. For UMDA, the performance decreases in a step-wise manner which is due to the convergence of the probability vector to two different adjacent noise optima.

When comparing the results obtained on NK-landscapes to the ones obtained on the One Max problem, one can identify some small differences between the two, *cf.* Figure 5.3. Nevertheless, the rank of each algorithm remains generally the same. Again $\text{vQEA}_{1,10}$ appears to be the most robust method on the positive linear noise landscape. The difference of $\text{vQEA}_{1,10}$ to all other algorithms is even greater compared to the NK-landscapes. Also, all other versions of vQEA demonstrate good performance among all tested methods. $\text{vQEA}_{1,1}$ and UMDA report almost identical fitness evolutions on this problem. The performance difference of the algorithms on the negative linear landscape is more pronounced in One Max compared to the NK-landscapes. UMDA demonstrates a superior behaviour over all considered algorithms. Being the best version among vQEA, $\text{vQEA}_{1,1}$ also significantly outperforms cGA. Similar to the results obtained on the NK-landscapes, PBIL and $\text{vQEA}_{1,10}$ represent the least robust algorithms on this noise landscape.

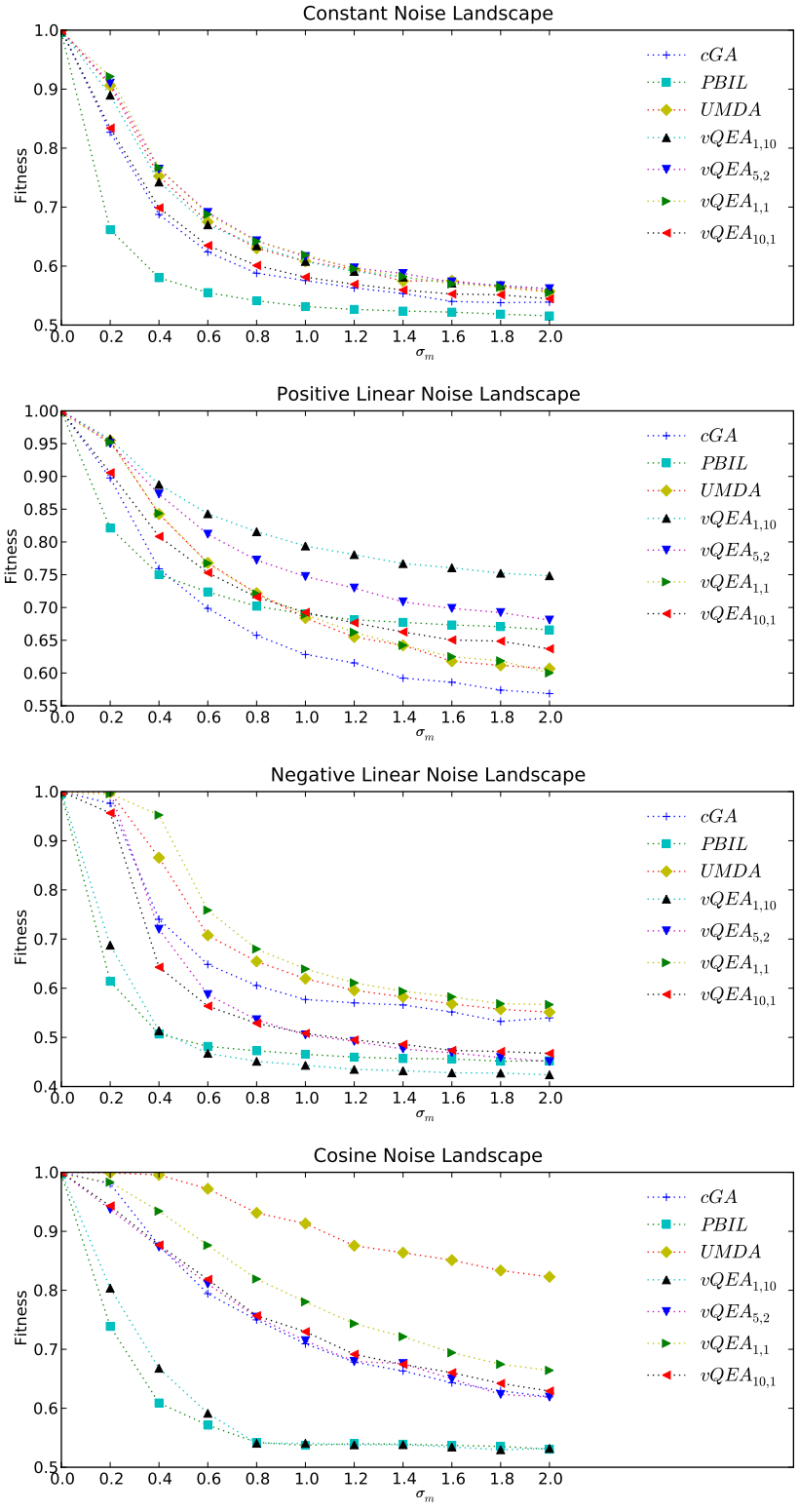


Figure 5.3: Robustness on One Max problem

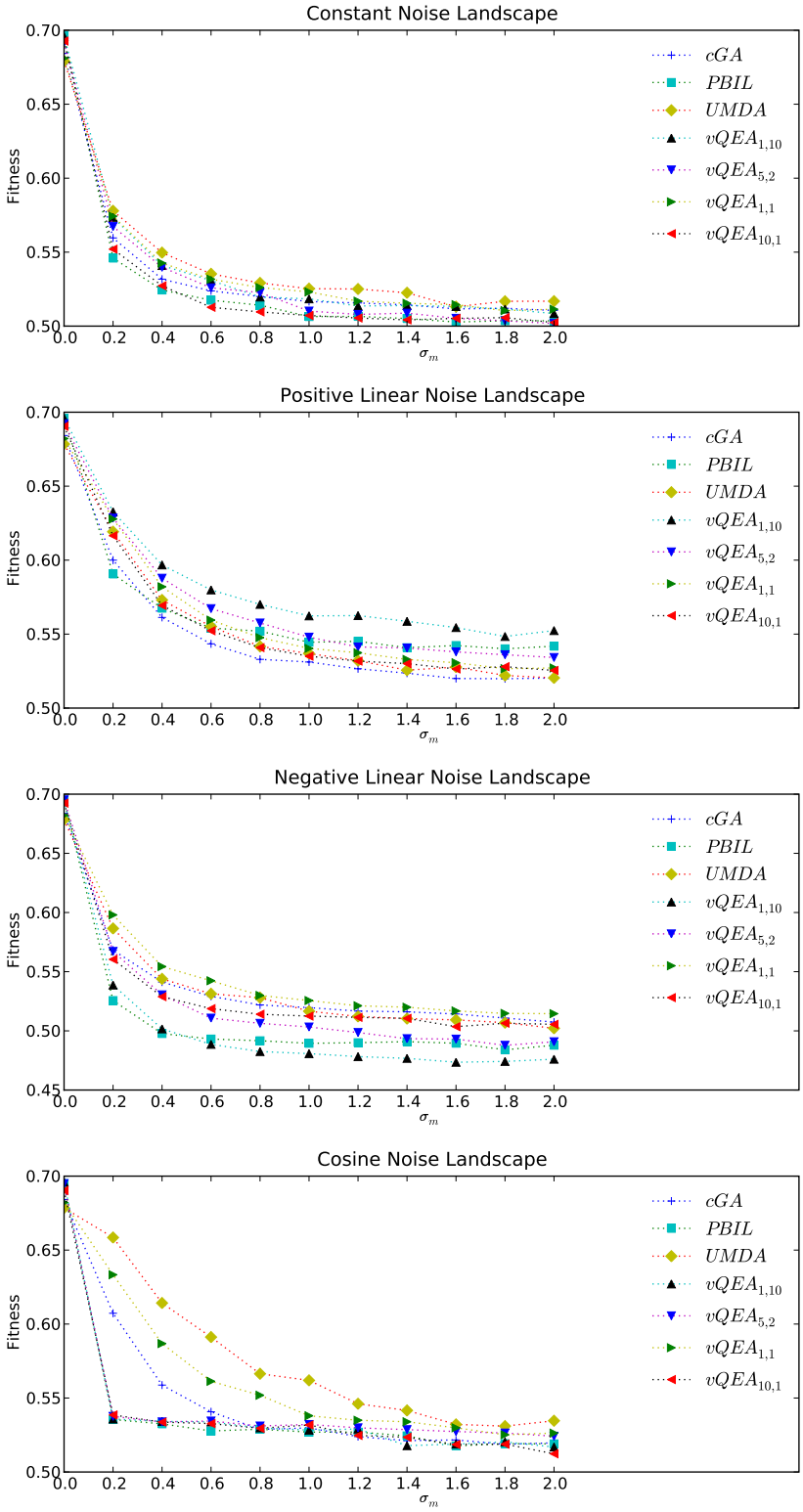


Figure 5.4: Robustness on NK-landscapes problem, K=4

Adding linkage to the NK-landscapes with $K = 4$ impacts the robustness of the algorithms significantly, *cf.* Figure 5.4. In general, the variability of the results is much higher than on the easier problems. On the constant noise landscape, all methods are more or less indistinguishable in their performance. Similar to the previous results $\mathsf{vQEA}_{1,10}$ performs very well in case of the positive linear noise compared to the other algorithms. On the negative linear landscape, $\mathsf{vQEA}_{1,1}$ is the highest ranked method, closely followed by UMDA and $\mathsf{vQEA}_{10,1}$. UMDA is still the most appropriate method for the cosine shaped landscape. Nevertheless, its performance is clearly strongly impacted by the increased difficulty of the problem. For higher noise levels, no significant difference to the other methods is observed.

5.3 DISCUSSION

In this section we investigate why some of the algorithms perform better than others and what the differences between the vQEA configurations are. We focus on classical aspects like the selective pressure and the way the fitness landscape is explored.

5.3.1 *Robustness and selective pressure*

At time t, search algorithms first collect information about the problem by sampling solutions x_t in the search space, then they select promising solutions s_t and move towards them. In the presence of misleading noise, the two main factors determining the performance of algorithms are: i) the way solutions s_t are selected, *i.e.* the selective pressure ii) the way the solutions s_t are utilised to further explore the search space. Therefore, we compare the selective pressure of the algorithms tested in this study. The selection intensity I is computed by a very informative metric of the selective pressure (Mühlenbein, 1997), as follows:

$$I = \frac{\overline{f}(S) - \overline{f}(X)}{\sigma(X)} \tag{5.5}$$

with X being a set of 500 solutions sampled by a given algorithm during a run[1], S is the set of the solutions selected from X, $\overline{f}(X)$ and $\overline{f}(S)$ are the average fitness of respectively X and S, and $\sigma(X)$ is the standard deviation of the fitness of the sampled solutions X. We note that a high value of I corresponds to a high selective pressure.

1 From each algorithm, 500 succeeding solutions were taken after the evolution of 10,000 fitness evaluations.

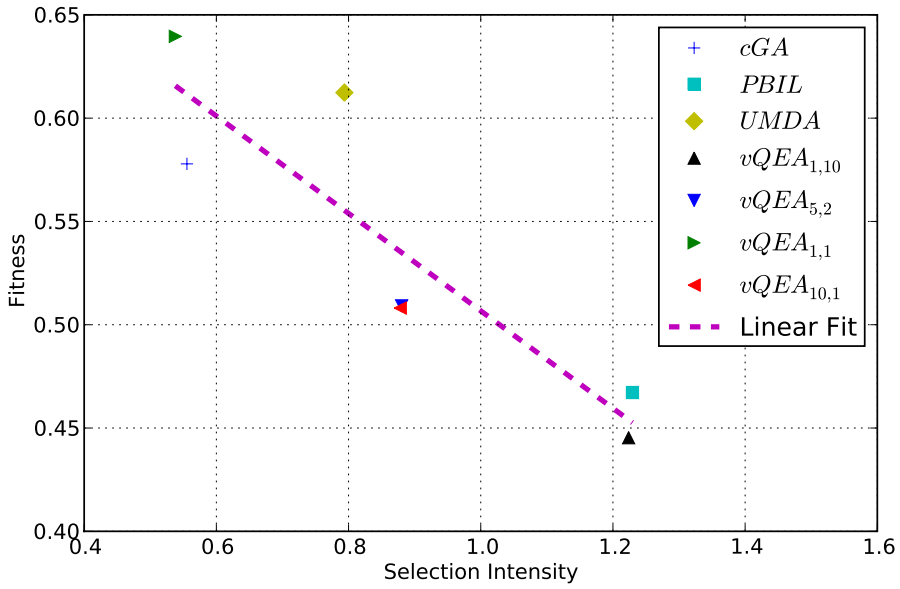


Figure 5.5: The average final fitness of all algorithms relative to the selection intensity. Each point in this diagram represents the average selection intensity obtained in 30 independent runs, and the corresponding average fitness achieved on the One Max problem using the deceptive negative linear noise landscape with noise strength $\sigma_m = 1$. It is demonstrated that for higher selective pressures the performance decreases on deceptive landscapes. All points were fitted using linear regression (dashed line) in order to indicate the trend.

In Figure 5.5, the average final fitness of each algorithm is presented as a function of the average selection intensity I on the One Max problem. The average fitness values are obtained from the negative linear noise landscape with $\sigma_m = 1$, while the values for the selection intensity are obtained by averaging the intensities of 30 independent runs on the One Max for each algorithm. On this noise landscape, the noise is deceptive and a low selective pressure is beneficial for an algorithm. We clearly see that the algorithms reporting the highest selective pressure, *i.e.* PBIL and $\text{vQEA}_{1,10}$ also report the lowest average fitness on this problem. Among all methods they are biased to follow the misleading information on this problem the most. On the other hand, $\text{vQEA}_{1,1}$ has the lowest selective pressure and reports the best performance. For the other algorithms, the situation is less clear. Some of them have a very similar selective pressure while reporting very different performance. In particular, according to the intensity selection, cGA selects solutions in a similar way compared to $\text{vQEA}_{1,1}$, but seems to be less able to exploit these solutions in a beneficial way.

UMDA has a comparably high selective pressure, but can exploit the selected solutions much better than for example cGA.

5.3.2 *Exploration of the noise landscape*

To better understand the behaviour of the algorithms in the presence of noise, we investigate the way a noisy search space is explored. Therefore, the cosine noise landscape on the One Max problem is considered here, because the achieved average fitness for noise strengths $\sigma_m = 1$ shows a large difference between the tested algorithms.

In Figure 5.6, the darker grey points of coordinates $(f(s), F(s))$ correspond to the (real and noisy) fitness of solutions $s \in s_t$ that have been selected and then used by the EDA to update their probabilistic models during a run. The number of solutions in s_t is different among the algorithms tested here. To follow the dynamics of the exploration, we grouped 250 succeeding selected solutions s together in sets S and plotted the averaged pair $(\bar{f}(S), \bar{F}(S))$. Those points define the trajectory of an algorithm through the noise landscape, *cf.* black points in Figure 5.6.

To properly explore a noise landscape, an algorithm has to estimate the real fitness f of a given solution x. One way of achieving this is first to measure the noisy fitness more than once, *e.g.* $F_1(x), \ldots, F_n(x)$, and then to integrate the information collected, for example by averaging them $\frac{1}{n}\sum_i F_i(x) \sim f(x)$, *cf.* (Fitzpatrick & Grefenstette, 1988).

The performance of the different algorithms can be partly explained from the position of the selected solutions s_t in the noise landscape. The exploration realised by PBIL is clearly too biased towards large values of F. Thus, the estimation of f is very poor and the algorithm is stuck in the first encountered noise-induced optimum. The situation is very similar for $\text{vQEA}_{1,10}$. We note that for these two algorithms s_t correspond only to the fittest solution (according to F) sampled at time t. As seen previously, this selection scheme is responsible for a high selection pressure and is an inappropriate strategy for exploring a misleading noise landscape. In the case of $\text{vQEA}_{1,1}$ and cGA, the selected solutions s_t cover a wider range of F values around the real fitness f and consequently the corresponding estimation of f is better. As reported by the trajectories in the noise landscape, both algorithms avoid being trapped in the first noise optimum and to some extent they are able to follow the real fitness f. We note that, for these two algorithms, s_t corresponds to the winner, according to F, of a tournament of size two.

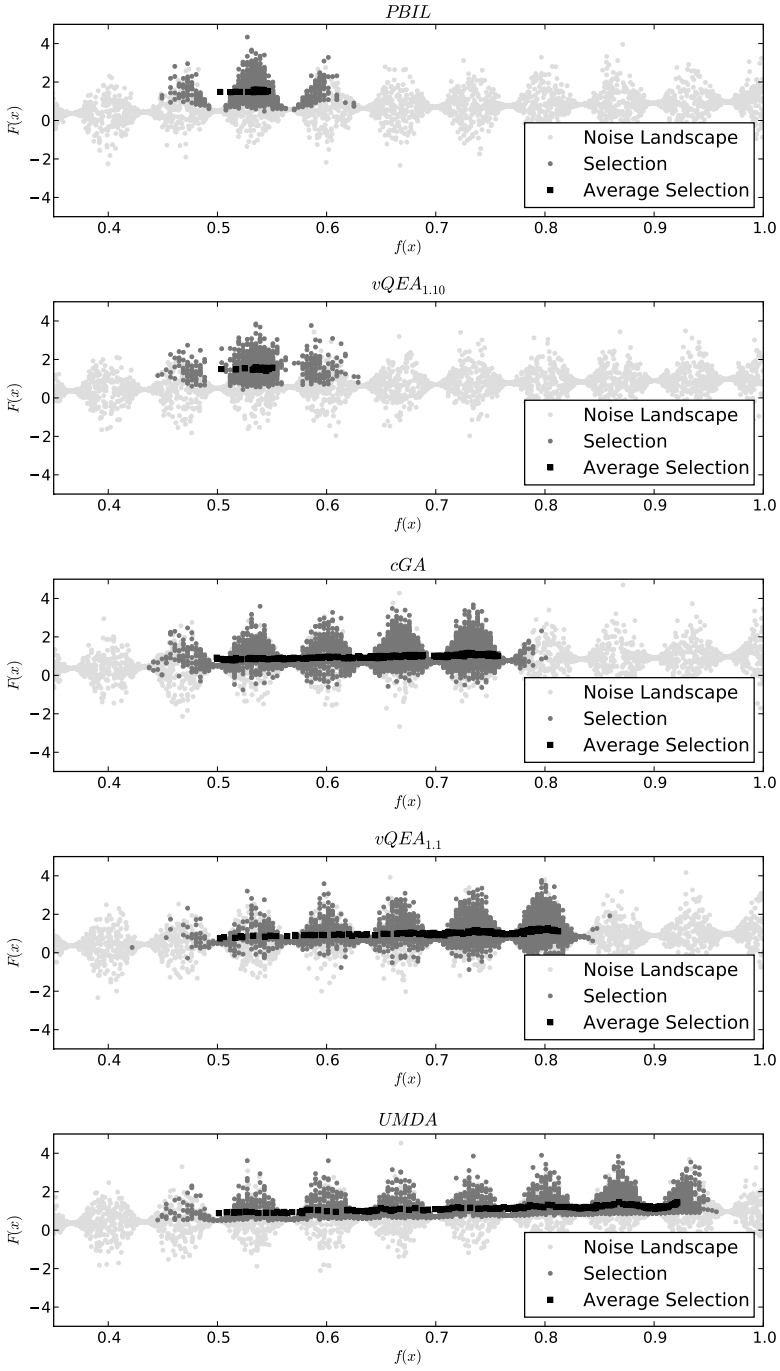


Figure 5.6: Exploration of the cosine noise landscape on the One Max problem. The choice of a too aggressive selection strategy (PBIL, $\text{vQEA}_{1,10}$) leads to an effective climbing of the noise landscape only. $\text{vQEA}_{1,1}$, cGA and UMDA are more successful due to a lower selective pressure in the selection scheme. The true fitness landscape can be explored effectively by averaging the fitness of a larger number of noisy solutions.

With regard to UMDA, the selected solutions s_t cover only the upper part of the noise landscape. This strategy should lead to a poor estimation of the real fitness f. However, the trajectory reveals a very good approximation of f. In UMDA, half of the solutions x_t sampled at time t are selected in s_t and then averaged to compute the probability model at $t+1$. In this study, UMDA benefits from the fact that the noise is normally distributed. Thus, the distribution of the $F(s_t)$ values is a truncated normal distribution for which the mean $\overline{F}(s_t)$ is close to the real fitness $f(s_t)$. This would not be the case with a uniform distribution. We note that this strategy is also not optimal when interactions between the variables exist as reported for NK-landscapes with $K = 4$, since averaging the selected solutions erases the patterns of interactions.

5.4 CONCLUSION

In this chapter the behaviour and the robustness of vQEA on several benchmark problems using different noise landscapes was analysed. At least on the tested benchmarks the results demonstrate a significant benefit for vQEA, especially when facing noise that is positively correlated to the fitness landscape, also called multiplicative noise. Multiplicative noise is often found in real world problems. It was shown that the selective pressure during the evolutionary process can be controlled by varying the population structure in vQEA. Small population sizes in combination with few global synchronisation events decrease the selective pressure, while a fully synchronised population structure increases it. This knowledge is very important for fine-tuning parameters for the algorithm on noisy problems. For different noise types, different strategies are necessary and the population structure needs to be adapted to the actual problem to solve. This requires *a priori* knowledge about the kind of noise in the search domain, which is sometimes unavailable. Hence an evolving selection scheme would be preferable and might result in an all-round version of vQEA. This concept could be implemented using a heterogeneous population structure. This would allow vQEA to switch between the strategies and choose the most appropriate configuration automatically.

Chapter 6

OPTIMISING CONTINUOUS SEARCH SPACES – A CONTINUOUS HIERARCHICAL MODEL EDA

Many real world problems require the optimisation of continuous search spaces. Although binary optimisation methods can be applied to this task, the use of a binary representation for a real-valued search space is not satisfactory since it introduces some critical issues into the optimisation process. Among the earliest studies pointing out the advantage of continuous over binary representations in a GA was given in (Janikow & Michalewicz, 1991). Each element of a continuous solution needs to be encoded by a number of bits. For the mapping of bit strings into a real value, additional computational overhead is necessary. Furthermore, a granularity is introduced into a continuous search space. Since a single continuous variable is represented by many bits, a binary optimisation method has to operate on more variables than a continuous optimiser. In other words, the one-gene-one-variable correspondence is lost in a binary representation. Thus, scaling problems can be expected, especially in the context of high-dimensional problems or whenever a need for a very precise optimisation of real-valued search variables arises. See (Janikow & Michalewicz, 1991) for an experimental comparison of the time performance of binary and real-coded GA. Furthermore, neighbouring solutions in the continuous domain might not be neighbours in their binary representation, a phenomenon known as Hamming cliffs (Goldberg, 1990). Exploring the local neighbourhood of a solution may require the optimiser to flip many bits at the same time that will encourage premature convergence and promote the phenomenon of hitch-hiking.

In the previous chapters, vQEA has demonstrated interesting characteristics and experimental results. The multiple probabilistic model and the hierarchical population structure allow an implicit adaptive learning rate which makes the method robust to its parameter configuration. Furthermore, the multi-model approach allows a finite number of decision errors resulting in competitive robustness against fitness noise. It was demonstrated that vQEA performs better in terms of speed and solution quality

than other first-level EDA, especially when links are introduced between variables (epistasis). Using several probabilistic models also allows a more diverse exploration of the search space than just using a single one.

In the following sections, the binary vQEA is extended towards the area of numerical optimisation. The $\mathcal{Q}$bits used in vQEA are replaced by a continuous probabilistic model and as a result the quantum metaphor is no longer suitable. Thus, the novel numerical optimiser is introduced as the *continuous hierarchical model EDA* (cHM-EDA). Since all key characteristics of vQEA are still present in cHM-EDA, similar advantages of this method in comparison to other continuous evolutionary methods are expected.

The chapter is organised in the following way. First the components of cHM-EDA are described and its functioning explained. Then its performance is investigated on a recently introduced state-of-the-art benchmark suite (Suganthan et al., 2005), which allows a direct comparison of results to other numerical optimisation methods in the field. The effects of the multiple probabilistic model on scalability and learning rate are experimentally demonstrated and discussed in separate sections. We also highlight briefly the robustness of cHM-EDA in the context of noisy fitness optimisation.

6.1 CONTINUOUS HIERARCHICAL MODEL EDA

The probabilistic model in vQEA is based on a Bernoulli random variable for each bit which is referred to as a $\mathcal{Q}$bit according to the quantum computing metaphor. Sampling from such a string of $\mathcal{Q}$bits results in the creation of a bit string which in turn can be evaluated by the corresponding fitness function. Since we want to consider continuous search spaces now, we have to replace the Bernoulli distribution by a continuous one such that it becomes possible to sample real values instead of discrete ones. A number of approaches to employ and model such distributions have been studied in literature about continuous EDA. The majority of approaches favour Gaussian distributions as the probabilistic model, some notable exceptions being *e.g.* (Servet, Travé-Massuyès, & Stern, 1998), where an interval representation for the PBIL was proposed, and (Yuan & Gallagher, 2003), where the authors present an improvement of the Gaussian based continuous PBIL, introduced as PBIL$_c$ in (Sebag & Ducoulombier, 1998), by implementing a histogram model.

We consider a continuous EDA based on Gaussian distributions here. For each dimension j of the continuous search space and for each probabilistic model i, a random variable following a Gaussian distribution is evolved. Therefore, the distribution

is fully described by two parameters: the mean $\mu_i^{(j)}$ and the standard deviation $\sigma_i^{(j)}$. In each generation, samples are drawn forming real-valued vectors whose quality can be evaluated by the fitness measure. An update rule is then applied to update $\mu_i^{(j)}$ and $\sigma_i^{(j)}$ to concentrate the search in promising areas of the search space, making higher quality solutions more likely to be sampled in the next generation. We will first describe the basic structure of the algorithm in detail, followed by the presentation of the chosen update rule.

6.1.1 *Model and population structure*

The overall structure of the proposed cHM-EDA is similar to vQEA. Like vQEA, the continuous version is also a population-based search method. Its behaviour can be decomposed in three different interacting levels as depicted in Figure 6.1.

INDIVIDUALS The lowest (inner) level corresponds to *individuals*. An individual i at generation t contains a probabilistic model $P_i(t)$ and two real-valued strings $R_i(t)$ and $A_i(t)$. More precisely, P_i corresponds to a string of N pairs of values $(\mu_i^{(j)}, \sigma_i^{(j)})$:

$$P_i = P_i^1 \ldots P_i^N = \begin{bmatrix} \mu_i^{(1)} & \ldots & \mu_i^{(N)} \\ \sigma_i^{(1)} & \ldots & \sigma_i^{(N)} \end{bmatrix} \tag{6.1}$$

The pair $(\mu_i^{(j)}, \sigma_i^{(j)})$ corresponds to the parameters of the distribution of the j^{th} variable of the i^{th} probabilistic model. Each variable in P_i is sampled according to $\mu_i^{(j)}$ and $\sigma_i^{(j)}$, so that R_i represents a configuration in the search space whose quality can be determined using a fitness function f. In most continuous optimisation problems, the variables have a specific domain of definition. Without loss of generality, we assume each $r_i^{(j)} \in R_i$ to be defined in to the interval $[-1, 1]$. As a consequence, each $r_i^{(j)} \in R_i$ follows a *truncated normal distribution* in the range $[-1, 1]$. Truncated normals can be sampled using a simple numerical procedure and the technique is widely adopted in pseudo-random number generation, see *e.g.* (Geweke, 1991) for an efficient implementation.

To each individual i a solution A_i is attached acting as an attractor for P_i. Every generation, R_i and A_i are compared in terms of their fitness. If A_i is better than R_i (*i.e.* $f(A_i) > f(R_i)$ assuming a maximisation problem), an update operation is applied on the corresponding model P_i. The update will move the mean values of the probabilistic model P_i slightly towards the attractor A_i. The choice of a suitable

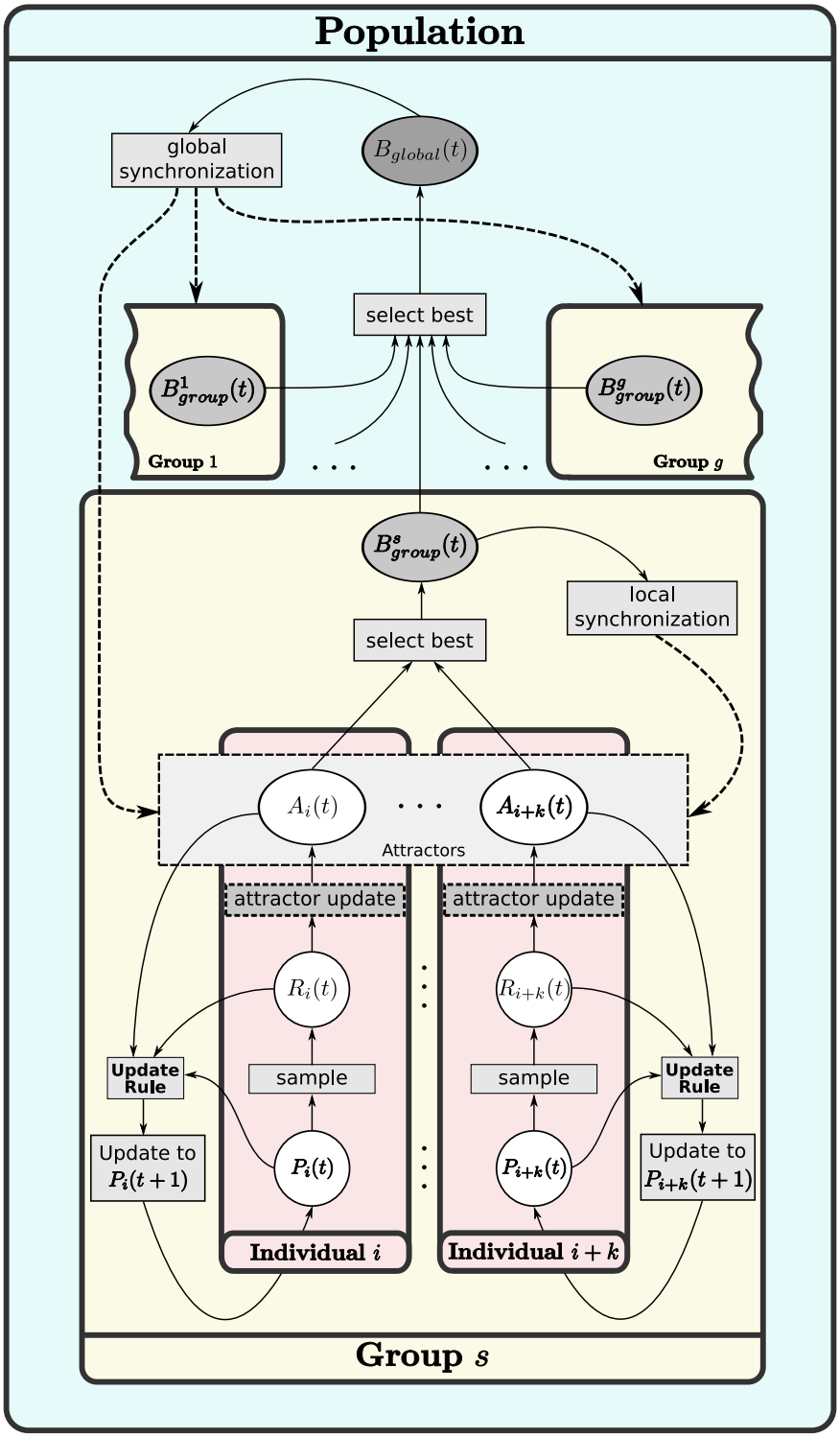


Figure 6.1: Three interacting levels can be distinguished in the continuous multi-model EDA: The individual, group and population level.

model update operation is critical for the working of the algorithm. We will elaborate the details of the probabilistic model update in section 6.1.2.

The update policy of an attractor A_i can follow two distinctive strategies. In the original QEA (Han & Kim, 2002) an *elitist* update strategy was used, in which the attractor A_i is replaced by R_i only if R_i is better than A_i in terms of fitness. Due to the *non-elitist* update strategy used in vQEA A_i is replaced at every generation. The choice of the update policy has great consequences for the algorithm and changes its behaviour completely. Since no experimental condition could be identified that favoured the elitist attractor update policy, we concentrate on the non-elitist version during the course of this chapter.

GROUPS The second (middle) level corresponds to *groups*. The population is divided into g groups each containing k individuals having the ability of synchronising their attractors. For that purpose, the best attractor (in terms of fitness) of a group, denoted B_{group}, is stored at every generation and is periodically distributed to the group attractors. This phase of local synchronisation is controlled by the parameter S_{local}.

POPULATION The set of all $p = g \times k$ individuals forms the *population* and defines the topmost (outer) level of the multi-model approach. As for the groups, the individuals of the population can synchronise their attractors, too. For that purpose, the best attractor (in terms of fitness) among all groups denoted B_{global}, is stored every generation and is periodically distributed to the group attractors. This phase of global synchronisation is controlled by the parameter S_{global}.

6.1.2 *Model Update*

The update of the probabilistic model is particularly interesting, since it governs how the search space is explored by the algorithm. Among the first continuous EDA proposed in literature is the continuous version of PBIL (PBIL$_c$) (Sebag & Ducoulombier, 1998), which uses independent Gaussian distributions for each variable of the problem. Several variants for updating the mean and standard deviation of each Gaussian distribution were presented and tested on a number of benchmark problems. The study in (Yuan & Gallagher, 2003) revealed a number of problems of the method and as a result an entirely different probabilistic model was proposed. In (Gallagher & Frean, 2005), a new update rule is investigated and compared to the generalised mean shift clustering framework. A general framework on continuous EDA, namely Iterated Density Estimation Evolutionary Algorithm (IDEA), was proposed in (Bosman

& Thierens, 2000), and the similarity to the EDA by Mühlenbein et al. (1999) was noted.

The common principle of all these continuous EDA is based on the sampling of a larger population. According to the fitness of the sampled individuals, the probabilistic model is updated. High quality solutions have a stronger impact on the update that drives the model towards promising areas in the search space. In cHM-EDA, the situation is very different, since only a *single* solution (for each probabilistic model) is sampled in every iteration. Hence, the model update cannot rely on the density of a population, but instead has to use a single attractor to perform the desired update. A very interesting continuous extension of the compact Genetic Algorithm (cGA) was developed in (Mininno, Cupertino, & Naso, 2008), which samples only two solutions in each iteration. Depending on the fitness, a winner and a loser solution is determined and the model is then shifted towards the winner solution. In (Mininno et al., 2008), the performance of this real-coded cGA was investigated by carrying out some very small-scale experiments. Comparisons with the standard genetic algorithm and the binary cGA did not show a significant advantage of this method. Nevertheless, it is very interesting to note that a probabilistic update based on only a single attractor (or winner solution) is feasible and is used in some methods.

We formulate here an appropriate update rule for the probabilistic models. Updating the mean $\mu^{(j)}$ in the Gaussian variable j is straightforward. We adopt a mean shift towards the value of the current attractor $a^{(j)}$ at location j, which is quite similar to the mean update used in some methods mentioned above. Depending on the distance $d^{(j)}(t) = a^{(j)}(t) - \mu^{(j)}(t)$, a shift $\Delta\mu^{(j)}(t)$ at generation t is defined as a sigmoid function:

$$\Delta\mu^{(j)}(t) = \frac{2}{1 + e^{-5d^{(j)}(t)}} - 1 \tag{6.2}$$

which is then used to perform the update:

$$\mu^{(j)}(t+1) = \mu^{(j)}(t) + \theta_\mu \Delta\mu^{(j)}(t) \tag{6.3}$$

In Equation (6.3) a parameter θ_μ is introduced, which we will refer to as the learning rate of the mean. We note that θ_μ corresponds to the maximum mean shift in a single generation.

Figure 6.2 visualises the effect of the update operation. In the diagram, the mean value was initialised to $\mu^{(j)}(0) = -1$ and then updated 100 times towards the attractor $a^{(j)}(t) = 1, \forall t < 100$. We note the deceleration in the update of $\mu^{(j)}$ when

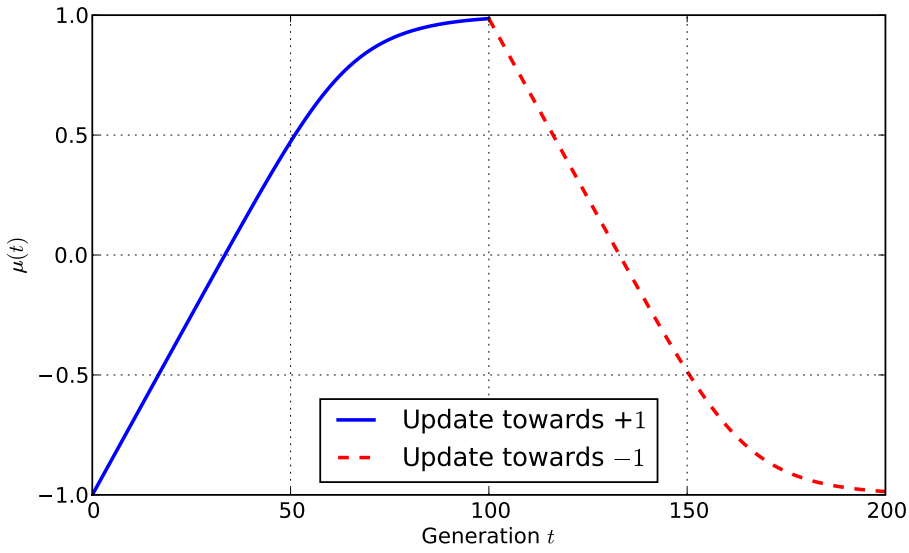


Figure 6.2: Theoretical variations of the mean value $\mu^{(j)}$ obtained through the successive application of the update operator used in cHM-EDA. The mean was initialised to $\mu^{(j)}(0) = -1$ and then updated for 100 generations towards attractor $a^{(j)}(t) = 1, \forall t < 100$. Then the attractor was exchanged to become $a^{(j)}(t) = -1, \forall t \geq 100$ and the mean was updated for further 100 generations. The update is asymmetrical since an attractor is approached slowly, but left quickly.

approaching the attractor. After generation $t \geq 100$, the attractor is set to $a^{(j)}(t) = -1, \forall t \geq 100$, which results in the update of $\mu^{(j)}$ in the opposite direction. We see that the update is not symmetrical: $\mu^{(j)}$ converges slowly towards an attractor but departs from it quickly.

Updating the standard deviation $\sigma^{(j)}$ is more difficult. If $\sigma^{(j)}$ is decreasing too quickly, the algorithm is prone to converge prematurely, while too slow a decrease might cause its non-convergence. Furthermore, an uncontrolled increase of $\sigma^{(j)}$ may also result in divergence. In that case, the resulting Gaussian distribution increasingly resembles a uniform one (in the interval $[-1, 1]$) and the algorithm performs a random walk in the search space. A small standard deviation allows a local search on a more specific area in the fitness landscape while a large deviation allows a stronger exploration of the search space. It is not trivial to answer at which time during the evolutionary run the algorithm should stop the exploration and start a localised search. Furthermore, this decision has to be made once more on the basis of a single attracting solution only.

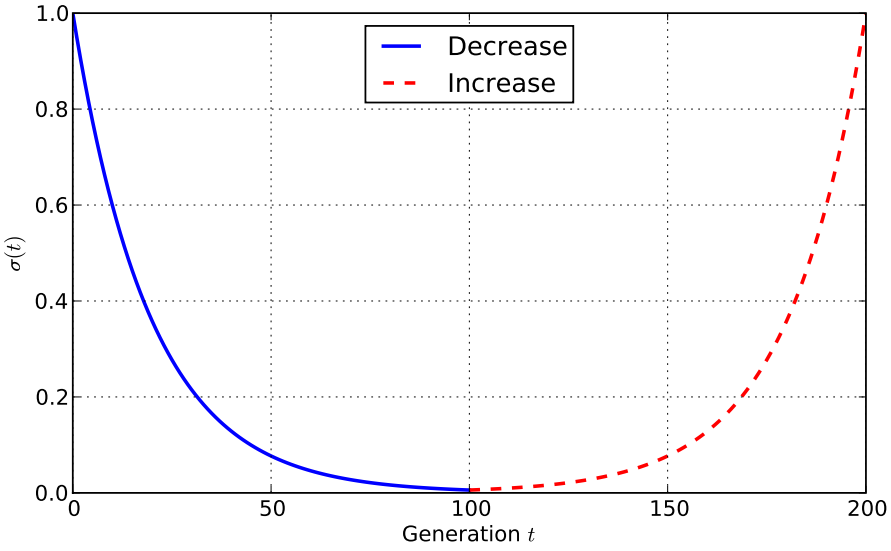


Figure 6.3: Theoretical variations of the standard deviation $\sigma^{(j)}$ obtained through the successive application of the update operator used in cHM-EDA. The standard deviation was initialised to $\sigma^{(j)}(0) = 1$ and then decreased for 100 generation using Eq. 6.4. After 100 generation the standards deviation was increased for another 100 generations. The update operator is symmetrical.

For the update of the standard deviation $\sigma^{(j)}(t)$, the hypothesis is that $\sigma^{(j)}(t)$ should decrease whenever $\mu^{(j)}(t)$ represents a "promising" area in the fitness landscape. We assume $\mu^{(j)}(t)$ to be "fit" when $|d^{(j)}(t)| < \sigma^{(j)}(t)$ at generation t. Thus, if the attractor $a^{(j)}(t)$ is close to $\mu^{(j)}(t)$ (within the boundaries defined by $\sigma^{(j)}(t)$), the standard deviation $\sigma^{(j)}(t)$ decreases. It is noteworthy that solutions fulfilling this condition are more likely to be sampled than other solutions, which means that on average $\sigma^{(j)}(t)$ will decrease. Attractors that are more distant to $\mu^{(j)}(t)$ and thus $|d^{(j)}(t)| \geq \sigma^{(j)}(t)$, will cause an increase of $\sigma^{(j)}(t)$, since it can be assumed that $\mu^{(j)}(t)$ does not represent a promising area in the landscape.

We define the standard deviation update at generation t as:

$$\sigma^{(j)}(t+1) := \begin{cases} \sigma^{(j)}(t) \times (1-\theta_\sigma) & \text{if } |d^{(j)}(t)| < \sigma^{(j)}(t) \\ \sigma^{(j)}(t) \times (1-\theta_\sigma)^{-1} & \text{otherwise} \end{cases} \tag{6.4}$$

In Equation (6.4) a parameter θ_σ is introduced, which we will refer to as the learning rate of the standard deviation. In order to avoid divergent behaviour of the algorithm, *i.e.* $\sigma^{(j)}(t)$ increases indefinitely, the domain of $\sigma^{(j)}(t)$ is restricted by defining upper and lower bounds, such that $\sigma_{\min} \leq \sigma^{(j)}(t) \leq \sigma_{\max}$.

Figure 6.3 visualises the effect of the update operation. In the diagram, the standard deviation was initialised to $\sigma^{(j)}(0) = 1$ and decreased during 100 update steps using Eq. 6.4. Again, we note the deceleration in the update steps when approaching $\sigma^{(j)}(t) = 0$ and thus convergence. After generation $t \geq 100$, the standard deviation $\sigma^{(j)}(t)$ increases for another 100 update steps. We see that this update operator is symmetrical.

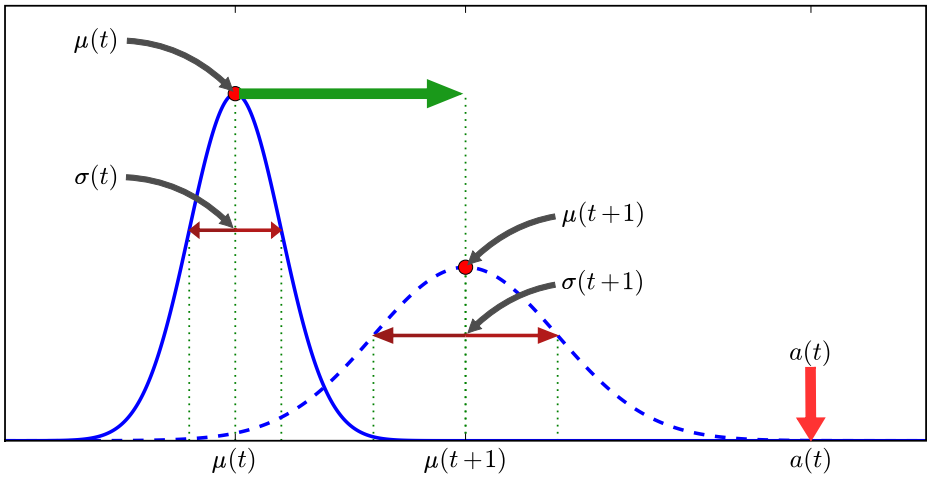
In Figure 6.4, the principle of the two defined update operators is summarised. Distant attractors (relative to the current mean of the Gaussian distribution) result in a large mean shift while at the same time the standard deviation increases, *cf.* Figure 6.4a. For close attractors, the mean shift is small and the standard deviation decreases, *cf.* Figure 6.4b.

It is important to note that the probabilistic update operator described above is similar to the rotation gate used in vQEA. As shown in chapter 4, the size of an update step using the rotation gate depends on the convergence of the probabilistic model. This phenomenon was described as a form of deceleration of the algorithm before convergence. As seen above, the shape of the theoretical variations of the update operations demonstrate a similar strategy in cHM-EDA, since here also the size of the update steps decreases with increasing convergence of the algorithm.

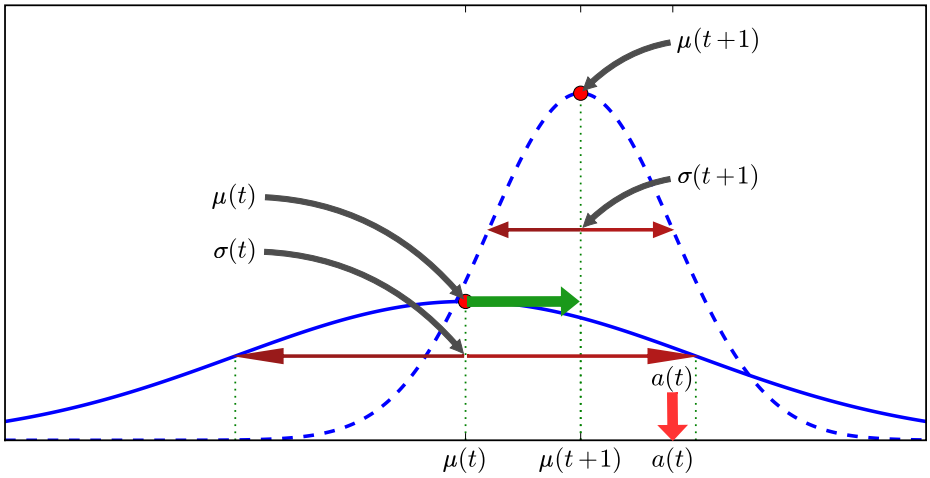
6.2 PERFORMANCE ANALYSIS

Among the most interesting aspects of a new algorithm is its optimisation performance compared against other algorithms in the field. In this section, the performance of the cHM-EDA is experimentally evaluated and compared to some state-of-the-art evolutionary methods.

The experimental methodology of evaluating evolutionary algorithms was repeatedly criticised in numerous publications, *cf. e.g.* (Hooker, 1995), (Whitley, Rana, Dzubera, & Mathias, 1996), (Eiben & Jelasity, 2002) and also (Gent et al., 1997). The criticisms aimed mainly at the “random” selection of benchmark functions and the lack of a clear motivation for this choice. Another issue for inter-comparisons arose due to the use of different measures to evaluate the performance of an algorithm (Eiben & Jelasity, 2002). Commonly used performance measures are i) the average success rate (SR), *i.e.* the average number of trials in which an algorithm was able to successfully solve the given problem, and ii) the mean best fitness (MBF) value obtained after the termination of the algorithm. Both measures are meaningful but can report very different outcomes. For instance, it is possible to have a low



(a) Update operation for distant attractors



(b) Update operation for close attractors

Figure 6.4: Update operation as used in cHM-EDA for a single Gaussian distribution. For each update step, the distance $d(t) = a(t) - \mu(t)$ between the attractor $a(t)$ and the mean $\mu(t)$ of the Gaussian distribution is computed at generation t. **(a)** If $d(t) \geq \sigma(t)$, the attractor is considered distant. It is assumed that $\mu(t)$ does not represent a promising area in the search space. In this case the mean $\mu(t)$ is strongly shifted towards the attractor (thick/green horizontal arrow) while at the same time the standard deviation $\sigma(t)$ is increased to broaden the search. **(b)** On the other hand, if the attractor is inside the boundaries defined by $\sigma(t)$, *i.e.* $|d(t)| < \sigma(t)$, then $\mu(t)$ is already in a promising area of the search space. The algorithm starts to localise the search by shifting $\mu(t)$ only slightly towards the direction of the attractor, while decreasing $\sigma(t)$ at the same time.

SR and at the same time a high MBF[1] and vice versa[2]. Different measures allow

1 The algorithm gets consistently close to the optimum, but rarely converges at the actual optimum.
2 Most trials are successful, but a few report a very poor final fitness.

more detailed insights into the tested algorithm and so multiple measures should be included in the performance analysis.

Most of these issues are explicitly addressed by contemporary benchmark suites. For the special session at the Congress on Evolutionary Computation (CEC) in 2005, an annual major event for the research field, a novel benchmark suite was proposed (Suganthan et al., 2005). The suite consists of 25 benchmark functions covering a variety of different problem characteristics. The functions range from simple separable uni-modal problems, over non-separable, non-linear, non-symmetrical, rotated and scalable functions, to complex hybrid composition functions in which several different function properties are mixed together. Furthermore, some noisy benchmarks are considered.

The suite was proposed as part of a competition on real-parameter optimisation at the CEC'05. An explicit design goal of the suite was the possibility of inter-comparisons between different methods. Thus, guidelines for statistical analysis and presentation of results are given as part of the benchmark specification. Very interesting is the fact that, as a result of the competition many algorithms have been compared on the same benchmark functions. Hence future algorithms can be easily compared to many existing methods and their performance evaluated.

Before studying cHM-EDA on the CEC'05 benchmark suite, the impact of the parameters on the optimisation performance is highlighted. Afterwards cHM-EDA is applied on the 25 test functions of the suite and compared to some state-of-the-art methods in the field of evolutionary computation.

6.2.1 *Guidelines for configuring cHM-EDA*

In order to get a better understanding of the meaning of the two learning rates θ_μ and θ_σ, the impact of these parameters on the performance of cHM-EDA is studied in this section. This information becomes very important when configuring the algorithm for a specific problem. The explicit goal is to derive some practical rules of thumb for a proper configuration of cHM-EDA, especially when some properties about the given search problem are known *a priori*.

Setup

In this analysis, a cHM-EDA with ten individuals that are fully synchronised in every generation is chosen as a specific population structure. This structure is directly adopted from previous experiments on vQEA, but might not be necessarily optimal

for cHM-EDA. Nevertheless, we restrict the analysis here to this simple configuration only and leave the analysis of more complex population hierarchies for future research.

A series of experiments is executed which show the achieved final fitness of the algorithm as a function of the two learning rates θ_μ and θ_σ. Since each problem has its own specific characteristics three different benchmark problems are chosen as representatives for potential optimisation problems. These three functions are taken from the CEC'05 test suite which is also used in the comprehensive benchmark study in the next section.

Among the most studied numerical benchmark functions is certainly the sphere function. Because of its uni-modal nature and its separability (*i.e.* no epistasis), it is supposed to be easy for any optimisation method. The function is named F1 in the test suite and a shifted version is used. The second function investigated here is the shifted Rosenbrock function (F6 in the test suite). It is multi-modal and non-separable and thus much more difficult to solve. The global optimum is inside a narrow flat valley. Finding the valley is comparatively easy, but converging towards the actual optimum is difficult. Finally a composed function is considered, namely F17, which is likely to reflect real-world scenarios the best. The number of local optima is large and different function properties are mixed together. Additionally, some additive Gaussian noise is involved in the function evaluation, which further complicates the problem.

The values for the mean shift θ_μ are varied in the range $[0.01, 0.4]$, while for the standard deviation update θ_σ values in the range $[0.001, 0.2]$ are considered. For each configuration, ten runs are performed and the obtained final fitness is averaged. All configurations are applied to the three problems using three different problem sizes $N = 10$, $N = 30$, and $N = 50$. Consistent with the guidelines of the CEC'05 test suite, the algorithm was allowed to perform $10 \times N$ fitness evaluations.

Results

Figure 6.5 presents the results obtained on the sphere benchmark problem F1. The diagram shows a contour plot in which the two axes represent the two learning rates θ_μ and θ_σ respectively while the colour reflects the solution quality obtained after the optimisation process. The darker the colour, the closer the algorithm has converged towards the global optimum of the function. Note the logarithmic scale of the colour axis.

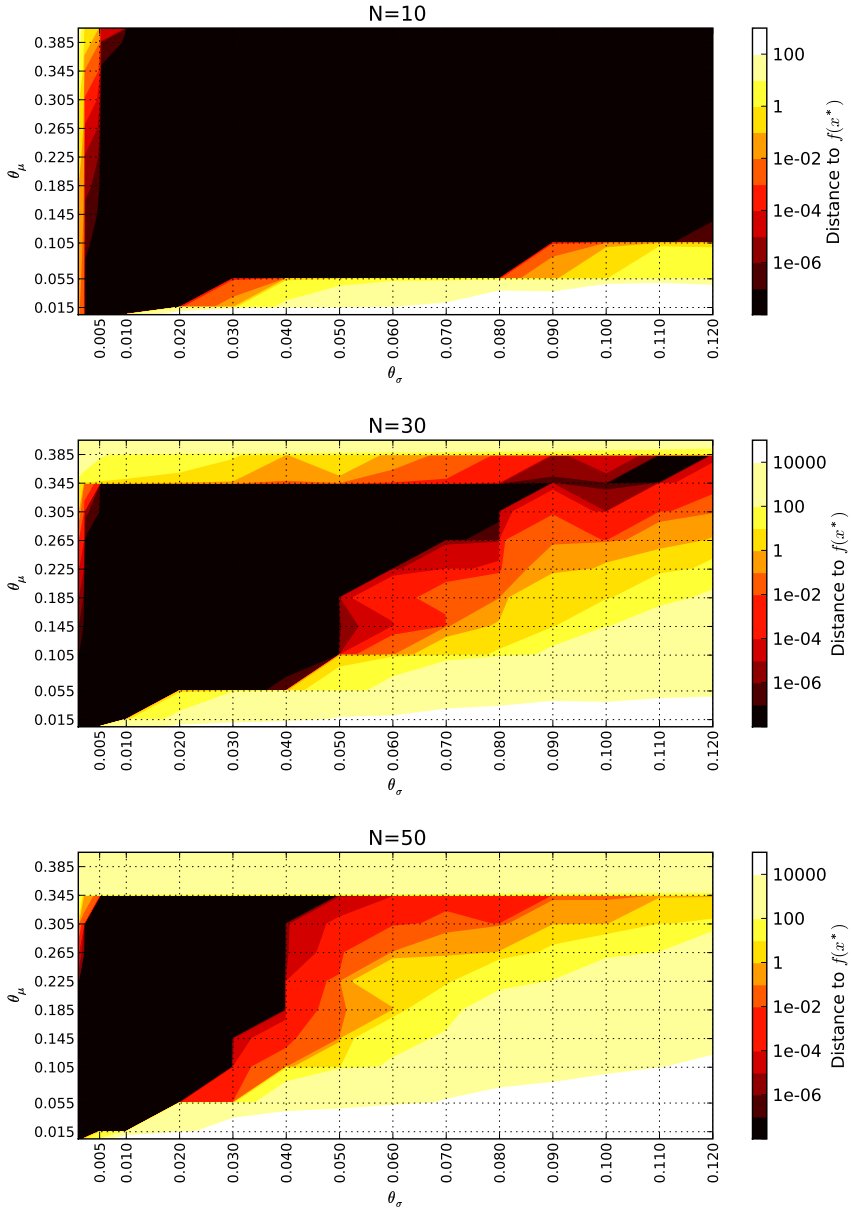


Figure 6.5: Impact of different parameter configurations on the performance of benchmark function F1

We clearly see that for this simple problem many configurations are suitable, especially for small problem sizes. We also note immediately that the setting of the mean shift θ_μ is almost irrelevant to solving the problem. A larger rate becomes more beneficial when the problem size increases which can be explained by the uni-modality of the function, *i.e.* since there is only a single optimum with a clear gradient pointing towards it, the "pace" of the algorithm can be fast without the risk to "skip" some optima. The only critical choice to make in this problem is the proper configura-

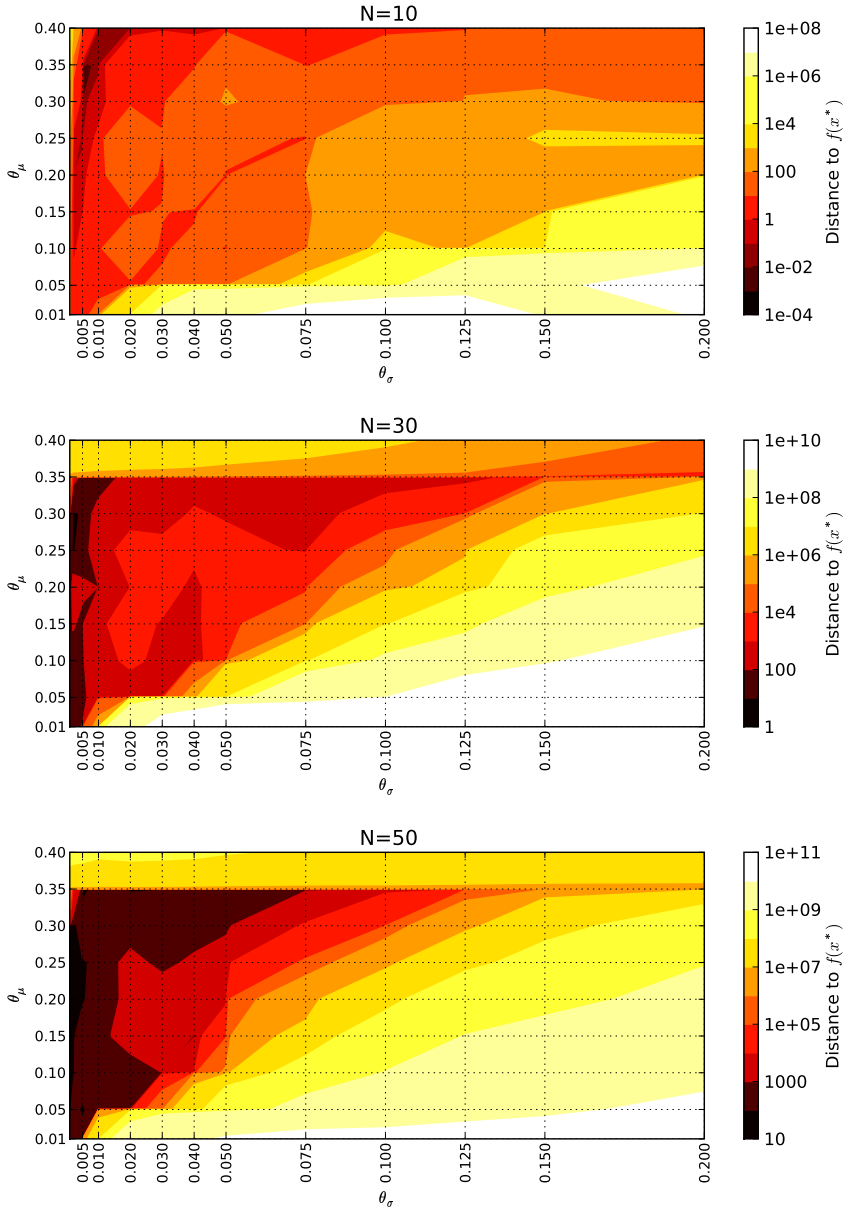


Figure 6.6: Impact of different parameter configurations on the performance of benchmark function F6

tion of the standard deviation shift θ_σ. For higher dimensions, the rate needs to be small enough to avoid premature convergence of the algorithm towards non-optimal solutions.

Increasing the difficulty of the problem increases the importance of the standard deviation shift θ_σ even more. In Figure 6.6, the results for the Rosenbrock function are presented. The meaning of the colours and axes in the diagram are the same as before. Once more we note the comparatively low influence of the mean shift θ_μ.

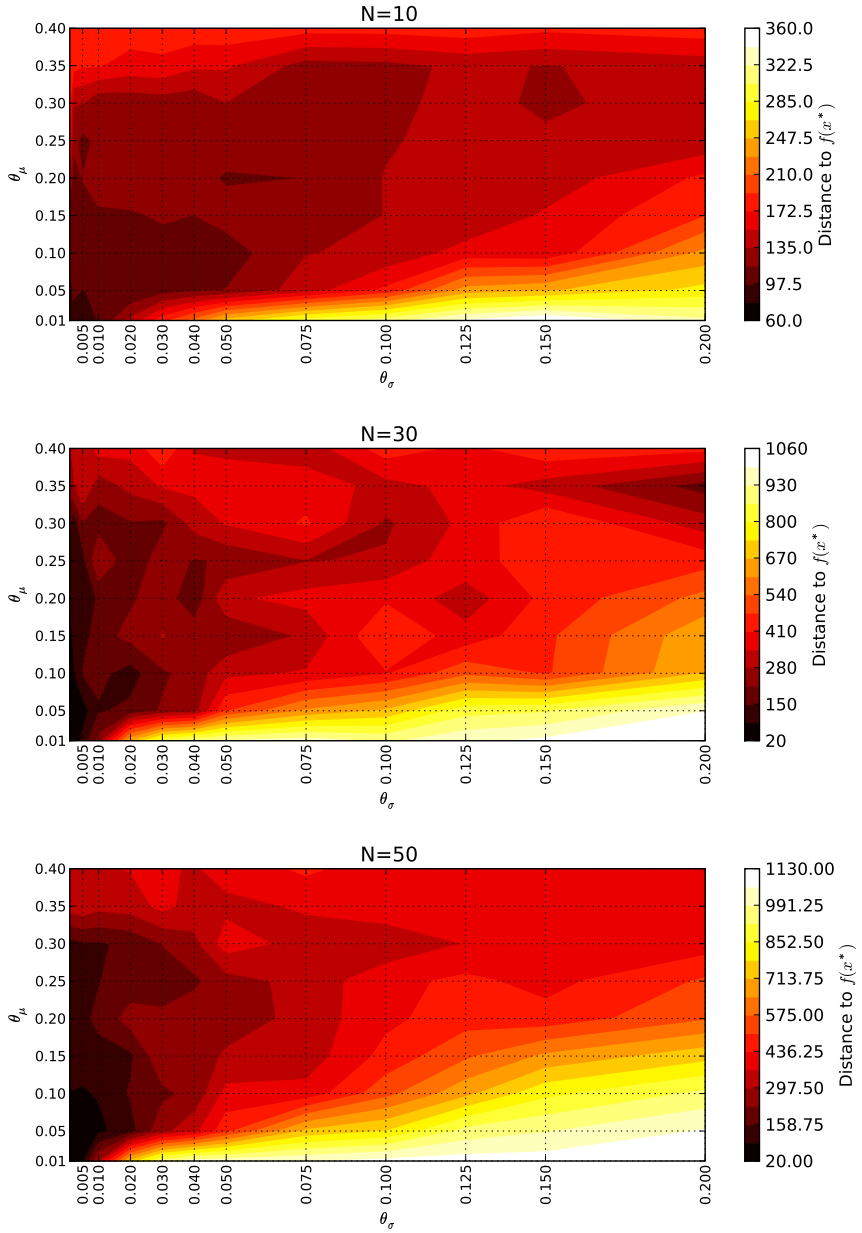


Figure 6.7: Impact of different parameter configurations on the performance of benchmark function F17

We also note that maintaining diversity is very important on this problem and this is expressed by the very small learning rates for θ_σ. This observation might be due to the flat areas in the landscape. On flat landscapes the optimisation progress is slow and the algorithm requires more iterations to follow the gradient. Furthermore, the mean shift rate θ_μ should be large in order to compensate for the slow optimisation progress.

Nevertheless, a large θ_μ rate is not always helpful as demonstrated by the results on function F17 presented in Figure 6.7. This function has a large number of local optima. Here smaller update steps (*i.e.* smaller mean shifts θ_μ) allow better compensation for decision errors. If the algorithm has identified a promising area in the search space and adjusted the μ values of the corresponding Gaussian distributions, the means are less prone to be shifted far away during single update steps due to a sudden attraction by a very distant attractor. Again, we notice the strong dependence of the performance on θ_σ.

Rules of thumb

A general observation of the presented results above is the comparably low importance of the mean shift rate θ_μ. If no information about the search space is given an appropriate default value is $\hat{\theta}_\mu = 0.05$. This setting allows the optimisation of simple uni-modal functions but also more complex problems within the given limit of fitness evaluations. It is explicitly noted that a larger learning rate may significantly reduce the number of required evaluations on easy problems. The mean shift should be larger when the landscape is known to be flat. Smaller rates are preferred on functions having many local optima.

The standard deviation rate θ_σ is generally the critical parameter in cHM-EDA. It should always be adjusted according to the dimensionality N of the problem. A reasonable choice for a default value is $\hat{\theta}_\sigma = \frac{1}{10\times N}$. On flat problems and problems with many optima this value should be decreased.

As a summary the following guidelines are presented:

I. If nothing is known about the fitness landscape, chose the default values for cHM-EDA: $\theta_\mu = \hat{\theta}_\mu = 0.05$ and $\theta_\sigma = \hat{\theta}_\sigma = \frac{1}{10N}$, where N is the problem size

II. If the landscape is known to be flat, increase the mean shift up to $\theta_\mu = 5 \times \hat{\theta}_\mu = 0.25$ and decrease the standard deviation rate to $\theta_\sigma = \frac{1}{2\times\hat{\theta}_\sigma} = \frac{1}{20\times N}$

III. If the landscape has a large number of local optima, decrease the mean shift to $\theta_\mu = \frac{1}{5}\hat{\theta}_\mu = 0.01$ and decrease the standard deviation rate to $\theta_\sigma = \frac{1}{2\times\hat{\theta}_\sigma} = \frac{1}{20\times N}$

The standard deviation rate θ_σ as a function of the problem size N is presented in Figure 6.8 for guideline I and guidelines II and III respectively.

It is not claimed that the above guidelines represent the most suitable configuration for all possible problems, but they can serve as a basis for further fine-tuning of the settings for a particular problem. The above findings are used to configure cHM-EDA to optimise the functions of the CEC'05 test suite for cHM-EDA.

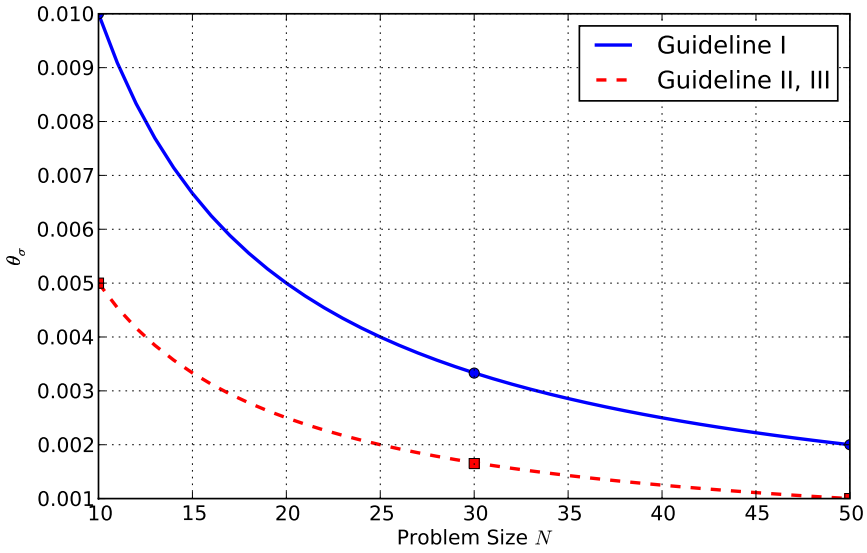


Figure 6.8: The standard deviation rate θ_σ as a function of the problem size N.

6.2.2 *Benchmark analysis*

In this section cHM-EDA is applied on the 25 functions of the CEC'05 benchmark suite. First, the experimental setup is explained, followed by the presentation of results and a comparison to some state-of-the-art methods in the field of evolutionary computation.

Setup

Three guidelines for configuring cHM-EDA have been experimentally derived in the previous section. cHM-EDA can be properly configured using these findings for the functions of the CEC'05 test suite. In Table 6.1, all 25 functions of the suite are associated with one of the three rules. The characteristics of each function are well known and described in the specification of the test suite.

Guideline	Function
I	F1, F2, F4, F7, F11 – F14
II	F3, F5, F6, F8
III	F9, F10, F15 – F25

Table 6.1: Configuration of cHM-EDA for the functions of the CEC'05 test suite

Algorithm	Description and Reference
CMA-ES	The quasi parameter free *Local Restart Covariance Matrix Adaptation Evolution Strategy* as described in (Auger & Hansen, 2005)
MVG-EDA	The *Multi-Variate Gaussian Model EDA* introduced in (Yuan & Gallagher, 2005) uses covariance sampling similar to the CMA-ES
D/E	The "classical" *Differential Evolution* algorithm using a DE/rand/1/bin scheme as described in (Storn & Price, 1997). Results obtained from (Rönkkönen, Kukkonen, & Price, 2005).
CoEvo	*Co-evolutionary Strategy* that co-evolves a population of solutions and a population of mutation steps used for exploration of the search space (Posik, 2005)
Hybrid GA	*Hybrid Real-coded Genetic Algorithm* with female and male differentiation that combines a local and a global search strategy (García-Martínez & Lozano, 2005)

Table 6.2: Algorithms used for comparison to cHM-EDA on the CEC'05 benchmark suite

Additional attempts have been made to further improve the performance on the uni-modal functions F1 to F5, resulting in the following exception for F2: the θ_μ rate was set to 0.1, while the θ_σ rate was kept unchanged according to guideline I.

Functions F7 and F25 represent problems without boundaries for the search space. Here the algorithm starts with an initial population specified in a certain range, but the optimum is outside of this initialisation range. For these functions, cHM-EDA uses (non-truncated) Gaussian distributions which allows the method to explore an unbounded search space.

All benchmark requirements and related settings are strictly adopted from the benchmark specification. We refer to (Suganthan et al., 2005) for a comprehensive list of all details.

In order to allow some performance comparison, the mean best fitness errors of recently published algorithms are taken into consideration. As discussed earlier, this single performance measure alone is not enough for a comprehensive comparison of methods. Nevertheless, it should provide the reader with a general overview of the obtained results. The considered algorithms are briefly described in Table 6.2. All methods have participated at the CEC'05 competition. They have demonstrated a highly competitive performance on the benchmark functions and are among the leading algorithms in the field of numerical optimisation. The benchmark results are directly available from the references given in the table.

Results

The test suite proposes guidelines for presentation of results and recommends a specific formatting of retrieved statistical information. The complete outcome of the results in the required format is presented in Appendix B of the thesis. To give a general overview of the obtained results some summarising figures are presented here.

Each function is tested using three different problem sizes $N = 10$, $N = 30$, $N = 50$. For $N = 10$ the mean best fitness error, *i.e.* the difference between the obtained final fitness to the globally optimal fitness value, is presented in Figure 6.9. The diagram shows the results of cHM-EDA and the comparison algorithms given in Table 6.2. For most problems, all methods report similar performance, function F3 being a notable exception. F3 is a uni-modal, shifted and rotated, high conditioned elliptic function that could not be properly solved by cHM-EDA. Algorithms like the hybrid GA, MVG-EDA and to some extent also D/E report similar difficulties here, but demonstrate better results compared to cHM-EDA. On functions F10 and F11, cHM-EDA is very competitive, F11 being solved consistently in most of the runs. More specifically, the success rate is 92% on F11, meaning that 23 out of 25 runs solved the problem with the required accuracy[3]. On the noisy function F4, the CMA-ES is outperformed by a surprisingly clear margin by all other methods. In (Auger & Hansen, 2005), this observation is explained by an initial step-size that is too small and by the failure of the method to enlarge it due to the effects of strong noise.

The difference between the tested methods becomes more obvious when the problem size increases to $N = 30$, *cf.* Figure 6.10. On the uni-modal functions F1 to F5, the Co-evolutionary Strategy (CoEvo) is significantly less competitive than other methods. Also, the performance of D/E is clearly affected by the dimensionality increase. It is known that function F7 becomes easier to solve with increasing problem sizes. All methods except CoEvo demonstrate an improved performance compared to $N = 10$ on this function. On the more complex problems F12 to F25 all methods report very similar performance with cHM-EDA having slight advantages on functions F16 and F17. Similar to $N = 10$, cHM-EDA shows better performance on F10 and F11, compared to all other methods.

A figure was also prepared for problem size $N = 50$. Since the CEC'05 competition included only problem sizes up to $N = 30$, not all algorithms have been tested on this problem size. Only the results of the CMA-ES are available for a comparison. Both methods are impacted by the higher dimensional search space. While

3 This is an example for a high success rate, but a comparatively low mean fitness due to very few suboptimal runs.

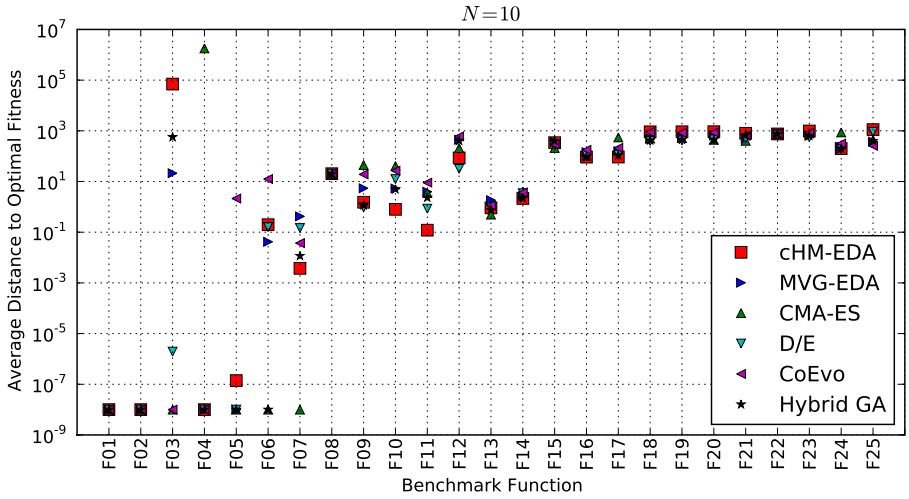


Figure 6.9: Comparison of the mean fitness error of contemporary state-of-art methods on the CEC'05 benchmark functions of problem size $N = 10$.

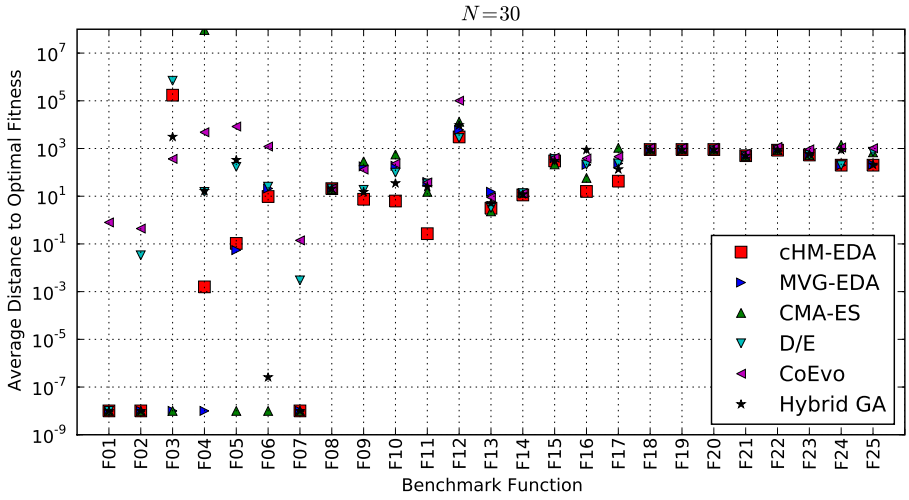


Figure 6.10: Comparison of the mean fitness error of contemporary state-of-art methods on the CEC'05 benchmark functions of problem size $N = 30$.

CMA-ES is consistently better on the uni-modal problems F1 to F3 and the multi-modal Rosenbrock function (F6), cHM-EDA reports superior results on functions F9 to F12, F16 and F17. The success rate on F11 is still 76% compared to 0% in the case of CMA-ES.

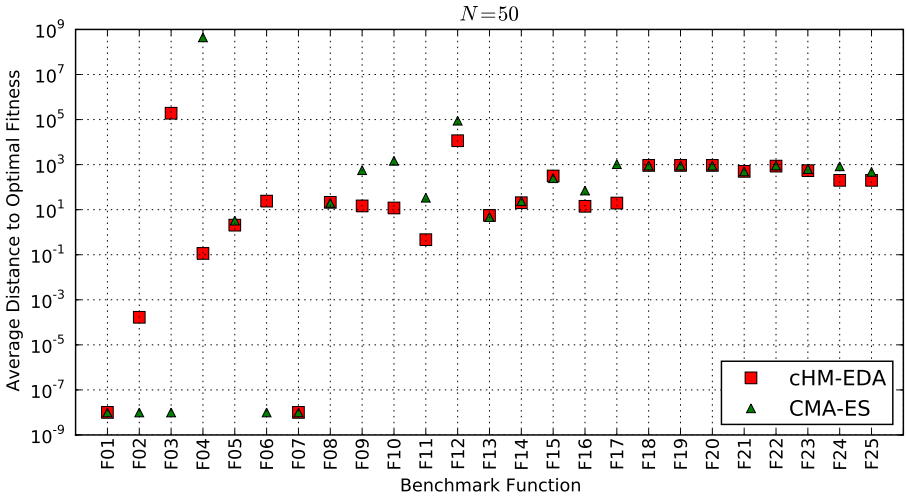


Figure 6.11: Comparison of the mean fitness error of contemporary state-of-art methods on the CEC'05 benchmark functions of problem size $N = 50$.

Conclusions

In this section, some interesting characteristics of cHM-EDA were demonstrated. The standard deviation rate θ_σ is the most critical parameter in the method and needs to be adjusted in dependence with the problem size while the mean shift rate θ_μ can be left constant for most problems. Three guidelines for configuring the algorithm were derived from the experimental observations and were demonstrated to work well in practise. Only function F2 benefits from a slightly different setting.

The overall performance of cHM-EDA is very competitive on most test functions. It becomes highly competitive on difficult multi-modal problems such as F9 to F12, F16 and F17, especially when the problem size increases. Clearly some problems arise with functions containing flat areas, or functions that require different learning rates depending on the stage of the evolutionary process. This observation was made on the functions F3 and F6, but also F2, which could only be solved after carefully fine-tuning the learning parameters. Here, an adaptive learning rate for the critical θ_σ parameter might be beneficial for cHM-EDA, since excellent results are reported by CMA-ES which employs a mechanism to adapt the learning rate during the search. On the complex hybrid problems, only some small differences between the tested methods could be observed.

6.3 ANALYSIS OF THE MULTI-MODEL

Similar to vQEA, the most important property of cHM-EDA is its multiple probabilistic model. In this section some potential advantages of this multi-model approach are highlighted, especially the adequate estimation of multi-modal fitness landscapes, the scalability of the method and its robustness against fitness noise.

6.3.1 *Estimating multi-modal fitness landscapes*

Single model EDAs, such as *e.g.* the PBIL$_c$, explore the search space using a single Gaussian distribution per variable. Hence the density estimation of promising areas in a multi-modal fitness landscape is limited. Only a single area in the search space can be explored at a certain generation since the Gaussian probability density function is uni-modal. Due to the use of more than one Gaussian distribution, cHM-EDA is supposed to be able to concentrate the search on many promising areas simultaneously. The models can independently explore several local optima in the landscape and "communicate" their findings about the corresponding solution quality during synchronisation events.

The ability of cHM-EDA to estimate non-trivial, complex fitness landscapes is experimentally demonstrated here. A simple multi-modal fitness function is defined and explored by a number of probabilistic models (individuals). The state of each model is then investigated after the evolution of a number of predefined generations. Two strategies are considered: one way of studying the state of all models is by sampling the corresponding distribution of each model and generating histograms of the samples. The second possibility is to explicitly compute the accumulated Gaussian probability density functions of all models. The accumulated Gaussian probability density function is given by

$$\phi^{acc}(x) = \frac{1}{N}\sum_{i=1}^{N}\phi_i^{trunc}(x) \tag{6.5}$$

with $f_i(x)$ being the probability density function of the truncation normal distribution with bounds $a = -1$ and $b = 1$ of individual i:

$$\phi_i^{trunc}(x) = \frac{\frac{1}{\sigma_i}\phi(\frac{x-\mu_i}{\sigma_i})}{\Phi(\frac{b-\mu_i}{\sigma_i})\Phi(\frac{a-\mu_i}{\sigma_i})} \tag{6.6}$$

where $\phi(\cdot)$ is the probability density function of the standard normal distribution, $\Phi(\cdot)$ its cumulative distribution function. It is noteworthy that both approaches should report equivalent results, as long the number of samples generated for the histograms is large and the histogram bins have high enough resolution.

For the sake of simplicity, a one-dimensional bi-modal fitness function is used:

$$f(x) = -x^4 + x^2 + \frac{x}{4} + \frac{1}{4} \tag{6.7}$$

where $x \in \mathbb{R}$ and $-1 \leq x \leq 1$. The function f has two local optima of different quality and is plotted as an overlay in Figure 6.12. As a result of the one-dimensional search space, all μ_i and σ_i in Eq. 6.6 are scalars.

The landscape of this fitness function is explored using 20 models, each of them organised in its own group. The 20 groups are synchronised every 50 generations and in total 1000 generations are allowed. Every 50^{th} generation 5000 samples are drawn from each model and a histogram of all 20×5000 samples is computed. Additionally, the accumulated Gaussian density functions are computed using Eq. 6.5. Small learning rates were used in this experiment to allow a very slow convergence.

A number of runs with varying initial random seeds were performed and a typical run is presented in Figure 6.12. After initialisation (generation 0) the mixture of Gaussian distributions resembles a uniform distribution. We see that after 50 generations many models have shifted their mean values towards the global optimum, but also the local optimum is represented as seen in generation 100 to 250. In later generations most models have converged towards the global optimum.

Conclusions

The experimental results suggest that cHM-EDA allows the estimation of multi-modal fitness landscapes. Due to their independent evolution, the probabilistic models can indeed simultaneously explore different areas in the search space. Individual models can be attracted by different local optima and, in case of a small enough global synchronisation rate, also converge towards them. This property of cHM-EDA is a key difference to the single model EDA.

6.3.2 *Scalability*

In this section, the scalability of cHM-EDA is investigated. For this analysis we compare four different population structures on the shifted sphere function, which

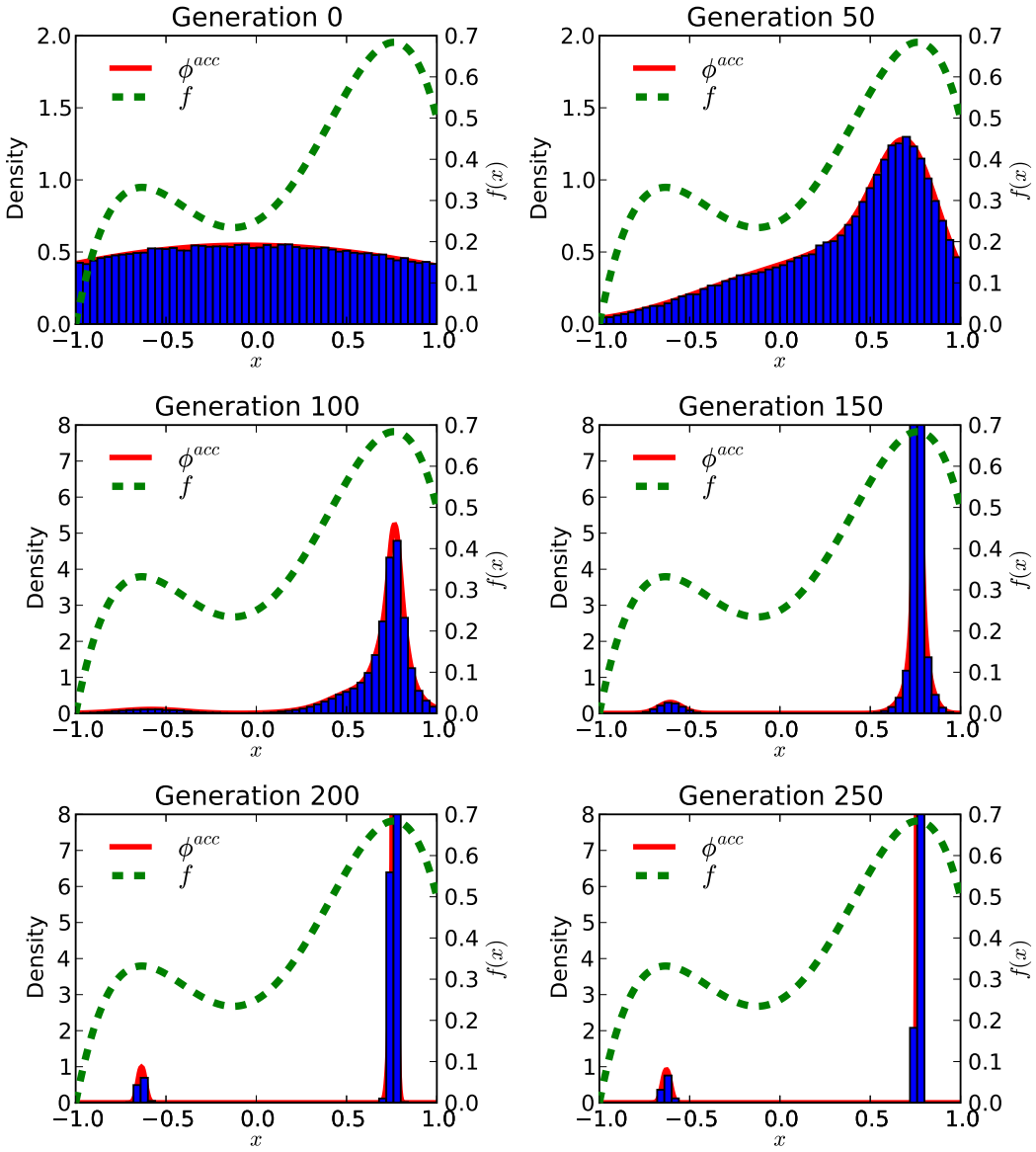


Figure 6.12: Histogram of sampled solution of all participating probabilistic models. The fitness function is overlaid (dashed line) in order to allow a direct comparison of the solution quality obtained by the model.

is function $F1$ of the CEC'05 benchmark suite. The following four structures were chosen: *1-1* (one group having a single individual), *1-5* (one group having five individuals), *1-10* and *1-20* (one group having ten and 20 individuals, respectively). Each of these configurations was applied on the sphere function with varying problem dimensions $N \in \{5, 10, \ldots, 100\}$. For each dimension and each of the four configurations 25 independent runs were performed and the average number of fitness evaluations required to solve the problem was determined. The problem was

Population Structure	Experiment I	Experiment II
1-1	$\theta_\mu = 0.1$ $\theta_\sigma = 0.00425$	$\theta_\mu = 0.1$ $\theta_\sigma = 0.003$
1-5	$\theta_\mu = 0.1$ $\theta_\sigma = 0.015$	$\theta_\mu = 0.1$ $\theta_\sigma = 0.0135$
1-10	$\theta_\mu = 0.1$ $\theta_\sigma = 0.0275$	$\theta_\mu = 0.1$ $\theta_\sigma = 0.02$
1-20	$\theta_\mu = 0.1$ $\theta_\sigma = 0.05$	$\theta_\mu = 0.1$ $\theta_\sigma = 0.025$

Table 6.3: Experimental setup of cHM-EDA for the two experiments on scalability

considered to be solved if the difference between global optimum and achieved fitness value was below an error threshold $\epsilon = 10^{-8}$. Each configuration was allowed to perform a maximum of 10^5 fitness evaluations (FES).

For each structure, the learning rates θ_μ and θ_σ have to be adjusted. As pointed out earlier, θ_μ is less significant on this benchmark function. Hence, it was fixed to $\theta_\mu = 0.1$ for all four hierarchies. The learning rate θ_σ is more critical and was determined for each population structure individually.

Two different experiments are conducted here. In the first experiment, the rates θ_σ of all four structures are configured to solve the problem of size $N = 30$ with the given accuracy using a *similar* amount of FES. Hence, none of the four settings has a performance advantage over the others on $N = 30$. It is then investigated how well these configurations perform on higher-dimensional problems, *i.e.* which configuration scales the best.

For the second experiment, the *most scalable* setting for the structures on all problem sizes is determined, *i.e.* all four settings can consistently solve the problem on all/most sizes. Then the settings are compared according to the required number of FES to find the optimum. More specifically, for each population structure the largest θ_σ is identified, such that the sphere problem of size $N = 90$ was solved by all 25 runs. That means the application of such a configuration on problem sizes $N > 90$ results in the failure of at least some runs to converge towards the global optimum. On the other hand, all four population structures can solve problem sizes $N \leq 90$. Hence, the chosen θ_σ is the most scalable setting that allows the successful optimisation of all/most problem sizes. The parameter settings for both experiments are presented in Table 6.3.

Figure 6.13 presents the outcome of the first experiment. In the diagram a filled symbol indicates that all 25 runs have identified the global optimum. If some runs

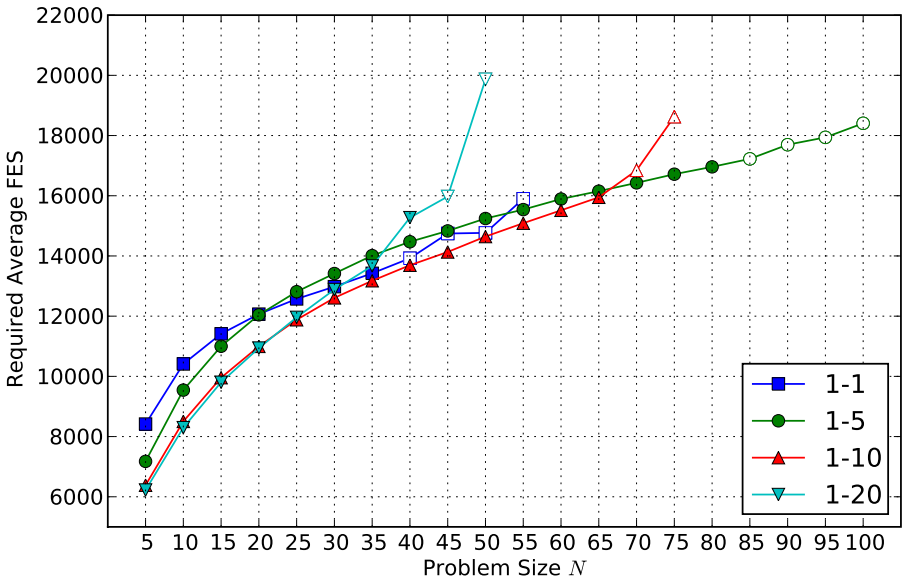


Figure 6.13: The required average number of fitness evaluations (obtained in 25 runs) as a function of the problem size of the sphere benchmark according to scalability experiment I. All four population structures use a single setting for all problem sizes. In the diagram, a filled symbol indicates that all 25 runs have identified the global optimum while an empty symbol is used when some runs were unsuccessful and the symbol is not shown if no run has found the optimum. Structure *1-5* is in this experiment the most scalable setting, followed by *1-10*.

were unsuccessful, an empty symbol is used instead and the symbol is not shown if no run has found the optimum. According to the experimental setup all four settings achieve a similar performance on problem size $N = 30$. On smaller sizes the results are also very similar to each other. With increasing problem sizes all configurations become increasingly unsuitable to solve the problem. At $N = 40$, population structure *1-1* is the first that does not solve the problem in all runs (note the empty symbol) and can not solve it at all for $N > 55$. Structure *1-20* shows a similar trend and is not able to cope with problems larger than $N > 50$. Structure *1-5* and *1-10* are clearly more scalable on this benchmark function, being able to solve most of the problem sizes with the fixed setting.

The results on the second experiment are presented in Figure 6.14. The interpretation of the symbols is the same as above. We note that all configurations can solve the problem successfully up to size $N = 90$ as defined in the experimental setup. Larger problem sizes $N > 90$ cause the failure of at least some runs to identify the global optimum (note the empty symbols for sizes 95 and 100). Also, in this scenario, the two multi-model structures *1-5* and *1-10* are the most appropriate choice.

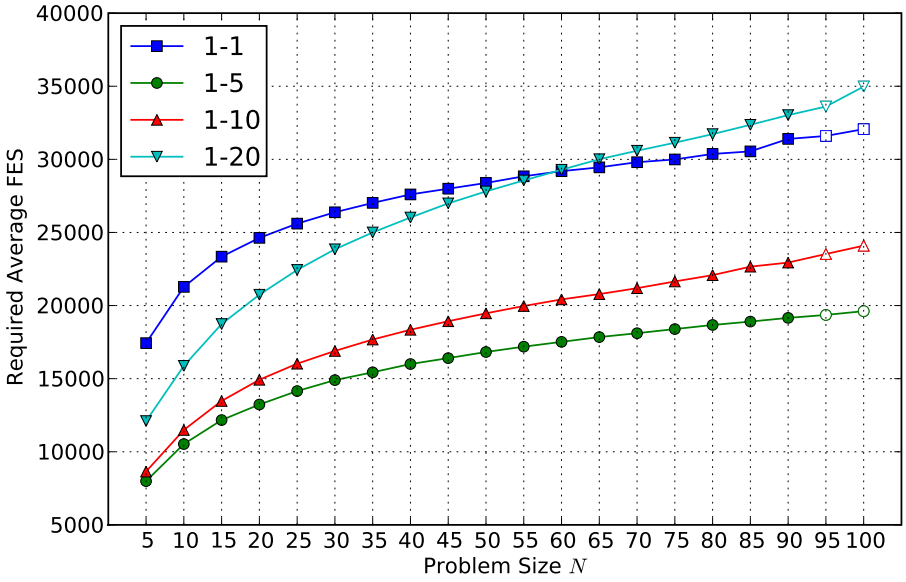


Figure 6.14: The required average number of fitness evaluations (obtained in 25 runs) as a function of the problem size of the sphere benchmark according to scalability experiment II. On all problem sizes, the population structures *1-5* and *1-10* dominate the other two investigated structures. Structure *1-5* requires on average $\approx 57.7\%$ less FES than structure *1-1*.

Structure *1-5* is particularly successful and requires on average $\approx 57.7\%$ fewer FES than structure *1-1* to solve the problem.

Conclusions

From experiment I, it is concluded that cHM-EDA is capable of solving a number of problem sizes using a single fixed setting for the learning rates. Assuming an appropriate population size, a multiple probabilistic model can solve more problem sizes than a single model. It was also demonstrated that the population size clearly impacts the scalability of the algorithm. For low-dimensional problems, a small number of individuals is sufficient; in fact, five to ten individuals represent the most scalable settings for the investigated problem sizes.

If cHM-EDA is configured to solve a large number of problem sizes, a multiple model is also beneficial. More specifically, multiple models require significantly fewer FES to solve the problem than a single model. This observation was made in experiment II. Once more it is noted that a small population size is more suitable, at least on the range of investigated problem sizes, compared to a larger population.

From the experimental setup of experiment II, it is observed that the learning rates θ_σ increase with the number of models used. The single model configuration *1-1* requires a 4.5 times smaller learning rate than structure *1-5*, which in turn is ≈ 1.5 times smaller than structure *1-10*. Due to the direct relationship between learning rate and convergence speed this observation seems very interesting and will be further investigated in the next section.

6.3.3 *Learning rates*

We have seen from the previous experiments that using several probabilistic models instead of a single one in cHM-EDA is beneficial in terms of either scalability or number of required FES to solve a problem. In the latter case, it was observed that the learning rate θ_σ could be larger if multiple models were used. Due to the direct relationship between learning rate and convergence speed this observation is the focus of this section. It is hypothesised that multiple models allow faster learning rates and as a result can speed up the optimisation process.

In the following experiment four population structures are chosen: *1-1* (one group having a single individual), *1-5* (one group having five individuals), *1-10* and *1-20* (one group having ten and 20 individuals, respectively). Once more the simplest test function, namely the sphere function F1, from the CEC'05 benchmark suite is used for this analysis and a number of different problem sizes $N = 10, \ldots, 100$ are considered. The idea of the experimental setup is to identify the optimal (*i.e.* fastest possible) learning rates for each of the four structures in order to solve the sphere function of a specific problem size N in a minimal number of FES. The obtained learning rates and the corresponding required FES can then be compared among the different population structures.

Since the learning rate of the mean shift θ_μ is of low importance on this problem it was fixed to 0.1 for all configurations. For each problem size, and for each of the four structures, the largest possible rate for the standard deviation shift θ_σ is determined that consistently solves the function in 25 independent runs[4]. The required setting was obtained through systematic trial and error.

In Figure 6.15, the learning rate θ_σ and the corresponding required number of FES to solve the problem are presented as functions of the problem size N for each of the four population structures. Regardless of the problem size, the single model *1-1*

4 A run is considered successful if the distance between optimal fitness and achieved fitness is smaller than $\epsilon = 10^{-8}$

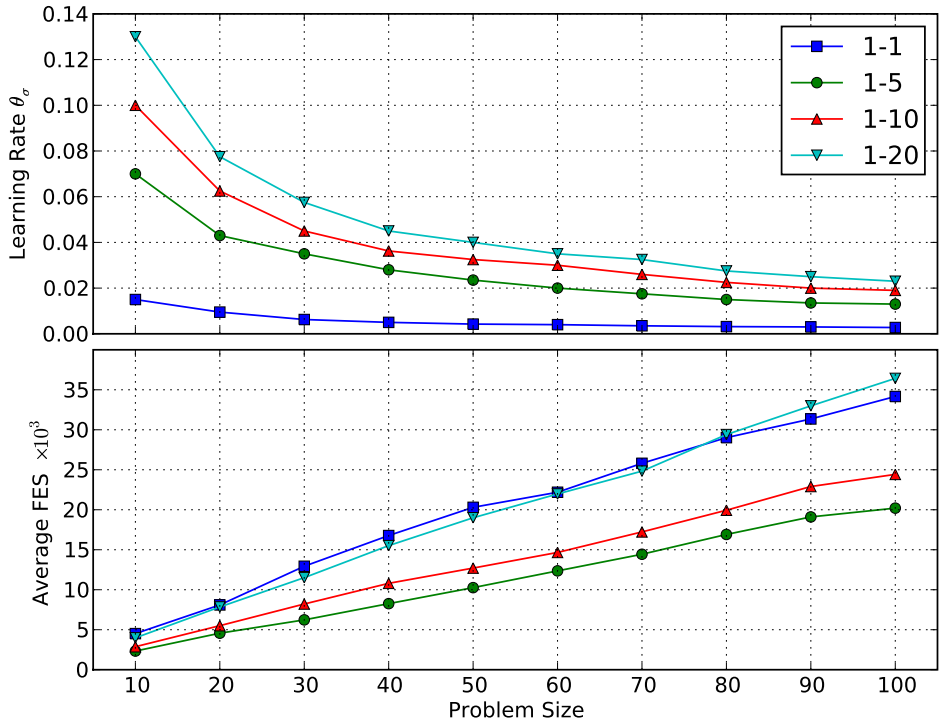


Figure 6.15: The learning rate θ_σ and the corresponding average number of required FES as a function of the problem size N for four different population structures. The optimal learning rate for multiple models is larger compared to the single model (top figure). Thus to a certain extent increasing the number of models allows also faster optimisation (structure *1-5* and *1-10*), *cf.* bottom figure.

reports clearly the smallest learning rate among all tested configurations. We also note that the larger the population size the greater the optimal learning rate becomes. Similarly to earlier experiments, it is observed that population structures *1-5* and *1-10* are better suited on the sphere function than *1-1* and *1-20*, the latter two requiring significantly more FES than the first two.

In the context of fully synchronised probabilistic models in cHM-EDA, multiple models allow a greater learning rate compared to a single model. Thus, increasing the number of models allows to a certain extent a faster optimisation of the considered problem. Nevertheless, if too many models are used, the number of FES increases despite the fast learning rate. On the sphere function, the optimal number of models appear to be five to ten.

Population Structure	Configuration
1-1	$\theta_\mu = 0.1$ $\theta_\sigma = 0.003$
1-5	$\theta_\mu = 0.1$ $\theta_\sigma = 0.0135$
1-10	$\theta_\mu = 0.1$ $\theta_\sigma = 0.02$
1-20	$\theta_\mu = 0.1$ $\theta_\sigma = 0.025$

Table 6.4: Experimental setup of cHM-EDA for the experiments on robustness to fitness noise

6.3.4 *Robustness*

It was earlier pointed out that the capability of a method to handle noisy and inaccurate information is an essential pre-condition to solve real-world problems. In this section, cHM-EDA is tested on a noisy function from the CEC'05 benchmark suite and several different population structures are compared to each other. We are especially interested whether the use of several probabilistic models is beneficial in the context of fitness noise.

The CEC'05 suite contains two noisy test functions: the noisy version of Schwefel's problem 1.2, namely F4, and a noisy hybrid composition function, namely F17. The latter is a multi-modal function with a large number of local optima and a mixture of many different function properties. Since only the effects of noise are under scrutiny here, F17 is less suitable for this analysis. Function F4 on the other hand is uni-modal, scalable and comparatively easy to solve. As seen in the benchmark study, most algorithms can solve this function at least for small problem sizes. Hence, we focus the experimental analysis on this function using $N = 10$ dimensions.

Once more four population structures are chosen: *1-1* (one group having a single individual), *1-5* (one group having five individuals), *1-10* and *1-20* (one group having ten and 20 individuals, respectively). The configuration of these structures is directly adopted from an earlier experiment on scalability (*cf.* section 6.3.2) and are summarised in Table 6.4. It is noted that the presented learning rates are slow enough to consistently solve the sphere function of size $N = 90$. The rates are deliberately chosen to be small in order to compensate for the effects of noise.

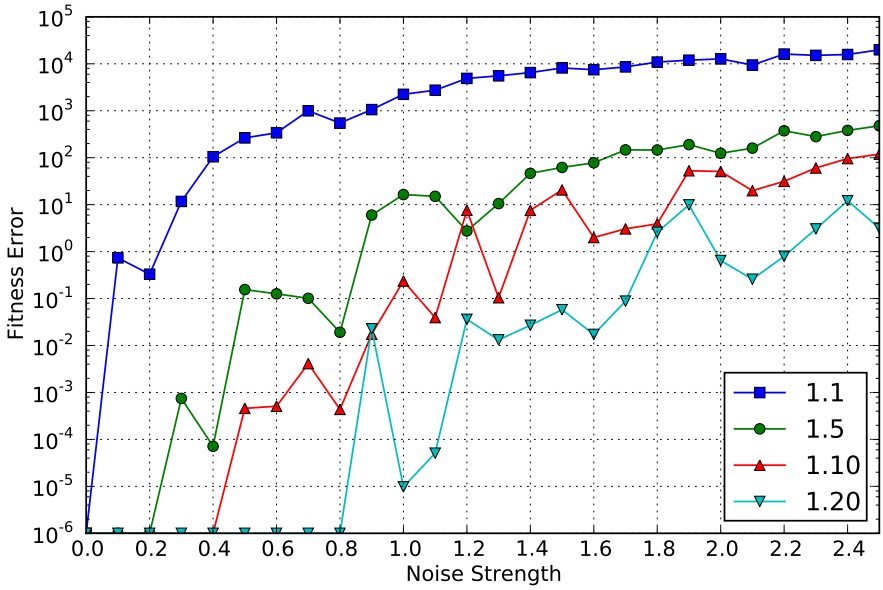


Figure 6.16: The average final fitness errors of four different population structures as a function of the noise strength σ_m. Due to the non-deceptive nature of the applied fitness noise, a higher selective pressure is beneficial on this problem. All configurations use a single group of fully synchronised individuals. Thus larger population sizes allow higher selective pressure and thus better robustness to the fitness noise.

Function F4 is given as:

$$F_4(x) := (\sum_{i=1}^{N}(\sum_{j=1}^{i} z)^2) \times (1 + \sigma_m|\mathcal{N}(0,1)|) + f_{bias} \tag{6.8}$$

where f_{bias} is a scalar constant fitness bias, $z = x - o,\ x = (x_1, \ldots, x_N)$ and o the location of the global optimum. Parameter σ_m indicates the strength of the applied noise. We immediately note that this noise is a multiplicative fitness noise since the effect of the noise decreases for solutions closer to the optimum and increases for more distant ones. At the optimum, the noise is zero.

In the benchmark suite, σ_m is fixed to 0.4. In this study values for σ_m vary in the range $[0, 2.5]$. For each σ_m and for each of the four configurations, 50 runs are performed and the evolution of the fitness error $\epsilon = F_4(x) - f_{bias}$ is recorded. According to the guidelines of the CEC'05 suite, an error of $\epsilon < 10^{-6}$ indicates a successful run.

The final fitness error of a single run is computed as the average of the last $1,000$ fitness values obtained during the evolutionary process. Since an algorithm has not necessarily found the global optimum, but has converged towards some different

point in the landscape, the average of several fitness evaluations is a fair estimate of the achieved real (non-noisy) fitness. In Figure 6.16, the average final fitness errors are presented as a function of the noise strength σ_m. All configurations are able to solve the problem consistently if no noise is present, *i.e.* $\sigma_m = 0$. As soon as noise is introduced, the performance of structure *1-1* is greatly impacted and the method quickly becomes unable to optimise the landscape at all. The multi-model structures are clearly more resistive to the disruptive effects of increasing noise strengths. Here the large population size of *1-20* is the most robust among the tested configurations. The large variations of the performance between different noise levels is due to a low final fitness of single runs and the logarithmic scale of the figure, which emphasise these small absolute differences.

Similar to the vQEA, cHM-EDA also demonstrates good robustness to multiplicative fitness noise. Increasing the number of probabilistic models also improves the robustness of the method to noise on the used test function. Since the noise is not misleading or deceptive, a high selective pressure is beneficial and consequently larger population sizes in a single fully synchronised group are very successful on this problem.

6.4 CONCLUSION

In this chapter, vQEA was extended towards continuous search spaces by replacing the $\mathcal{Q}$bit with a probabilistic model based on Gaussian distributions. All key characteristics of vQEA, namely a multiple probabilistic model, a hierarchical population structure and a convergence dependent learning rule are also part of its extension. The method was named continuous hierarchical model EDA, since the quantum metaphor has become inappropriate in the context of the Gaussian distributions.

The overall performance of cHM-EDA is very competitive especially on difficult, high-dimensional problems. Thus, it is claimed that the method may be a good candidate for handling real world problems. Issues may arise whenever the search space contains flat areas. Here some adaptive update strategies for the learning rate of the standard deviation, as *e.g.* employed in the CMA-ES, might be beneficial.

Along with the benchmark experiments, some practical guidelines for parameter configuration were presented. The standard deviation rate θ_σ is the critical parameter in cHM-EDA, while the mean shift θ_μ can be left constant for most problems. The derived guidelines work well on the 25 test functions of the CEC'05 benchmark suite.

To a certain degree, the multiple probabilistic model allows the simultaneous exploration of several promising areas in the search space. This characteristic is a key difference of cHM-EDA to single model EDA. Compared to a single model cHM-EDA, the multi-model cHM-EDA offers either a higher scalability of a fixed parameter setting or a faster convergence speed towards the optimum due to the use of faster learning rates. Furthermore, multiple models increase the robustness of the method in the context of multiplicative fitness noise.

Future directions may involve the exploration of different population structures. Another interesting idea is the use different learning rates among the groups since some groups move fast in order to quickly identify promising areas in the search space, while other groups move more slowly and maintain enough diversity to perform a localised search in the promising areas spotted by the faster groups. An interesting concept is also a restart strategy as implemented *e.g.* in some evolutionary strategies with the aim to overcome flat areas in the search space.

Chapter 7

OPTIMISING HETEROGENEOUS SEARCH SPACES: A HYBRID VQEA/CHM-EDA MODEL

The proposed vQEA and cHM-EDA share the same algorithmic structure and a combination of the two methods is straightforward and a natural extension towards a heterogeneous optimisation method. Both vQEA and cHM-EDA were shown to be highly competitive methods on their own and it is expected that their combined application in a hybrid algorithm would be beneficial.

The probabilistic model of this combined approach is a concatenation of a string of $\mathcal{Q}$bits and the string of Gaussian distributions as defined for the cHM-EDA. Since the probabilistic model of this algorithm is heterogeneous, the new method is introduced as the *heterogeneous hierarchical model EDA* (hHM-EDA).

Optimising heterogeneous search spaces using probabilistic methods is not new. The Mixed Bayesian Optimisation Algorithm (MBOA) was introduced in (Ocenasek & Schwarz, 2002) as a continuous-discrete optimisation method. Similarly to hHM-EDA, it belongs to the family of EDA and employs specialised probabilistic models to explore the binary and real-valued part of a solution efficiently. Nevertheless, some distinct differences between the two methods exist, which are highlighted and discussed in greater detail as part of this chapter.

The chapter is organised as follows. First, the novel hHM-EDA is presented and its functioning explained, followed by a comparison of the method to MBOA. Similarities and specifics according to the used probabilistic model, model update, sampling, selection and replacement strategies of both methods are explained. Since all the components of hHM-EDA have been individually discussed in the previous chapters, we focus here on the interaction between the combined probabilistic models. Once more the explicit aim is to develop some robust guidelines to determine suitable parameter configurations for the method. The performance of hHM-EDA is compared to a number of binary-only and continuous-only optimisation methods, but also to the continuous-discrete MBOA.

7.1 THE HETEROGENEOUS HIERARCHICAL MODEL EDA

The overall structure of the proposed hHM-EDA is almost identical to vQEA and cHM-EDA. Its behaviour can be decomposed in three different interacting levels, see Figure 7.1.

INDIVIDUALS The lowest level corresponds to *individuals*. An individual i at generation t contains a heterogeneous probabilistic model $\mathcal{H}_i(t)$ and two compound solutions $S_i(t)$ and $A_i(t)$. More precisely, $\mathcal{H}_i$ corresponds to a string of N pairs of models $(Q_i^{(j)}, P_i^{(j)})$:

$$\mathcal{H}_i = \mathcal{H}_i^1 \ldots \mathcal{H}_i^N = \begin{bmatrix} Q_i^{(1)} & \ldots & Q_i^{(N)} \\ P_i^{(1)} & \ldots & P_i^{(N)} \end{bmatrix} \tag{7.1}$$

where P_i denotes the continuous representation of the search space in the form of a string of Gaussian distributions:

$$P_i = P_i^1 \ldots P_i^N = \begin{bmatrix} \mu_i^{(1)} & \ldots & \mu_i^{(N)} \\ \sigma_i^{(1)} & \ldots & \sigma_i^{(N)} \end{bmatrix} \tag{7.2}$$

and Q_i the binary representation in form of a concatenation of $\mathcal{Q}$bits:

$$Q_i = Q_i^1 \ldots Q_i^{\mathrm{N}} = \begin{bmatrix} \alpha_i^1 & \ldots & \alpha_i^{\mathrm{N}} \\ \beta_i^1 & \ldots & \beta_i^{\mathrm{N}} \end{bmatrix} \tag{7.3}$$

The pair $(\mu_i^{(j)}, \sigma_i^{(j)})$ corresponds to the parameters of the distribution of the j^{th} variable of the i^{th} probabilistic model, and $(\alpha_i^{(j)}, \beta_i^{(j)})$ correspond to the probability amplitudes of the j^{th} $\mathcal{Q}$bit of the i^{th} probabilistic model.

Each variable in P_i and Q_i is sampled according to $(\mu_i^{(j)}, \sigma_i^{(j)})$ and $(\alpha_i^{(j)}, \beta_i^{(j)})$ respectively, forming a compound solution $S_i = (C_i, R_i)$, where C_i is a bit vector and R_i a real-valued vector of size N. Hence, $S_i(t)$ represents a configuration in the search space whose quality can be determined using a fitness function f.

Similarly to cHM-EDA, we assume without loss of generality each $r_i^{(j)} \in R_i$ to be defined in the range $[-1, 1]$. As a consequence, each $r_i^{(j)} \in R_i$ follows a *truncated normal distribution* in the range $[-1, 1]$. Truncated normals can be sampled using a simple numerical procedure and the technique is widely adopted in pseudo-random number generation, see *e.g.* (Geweke, 1991) for an efficient implementation.

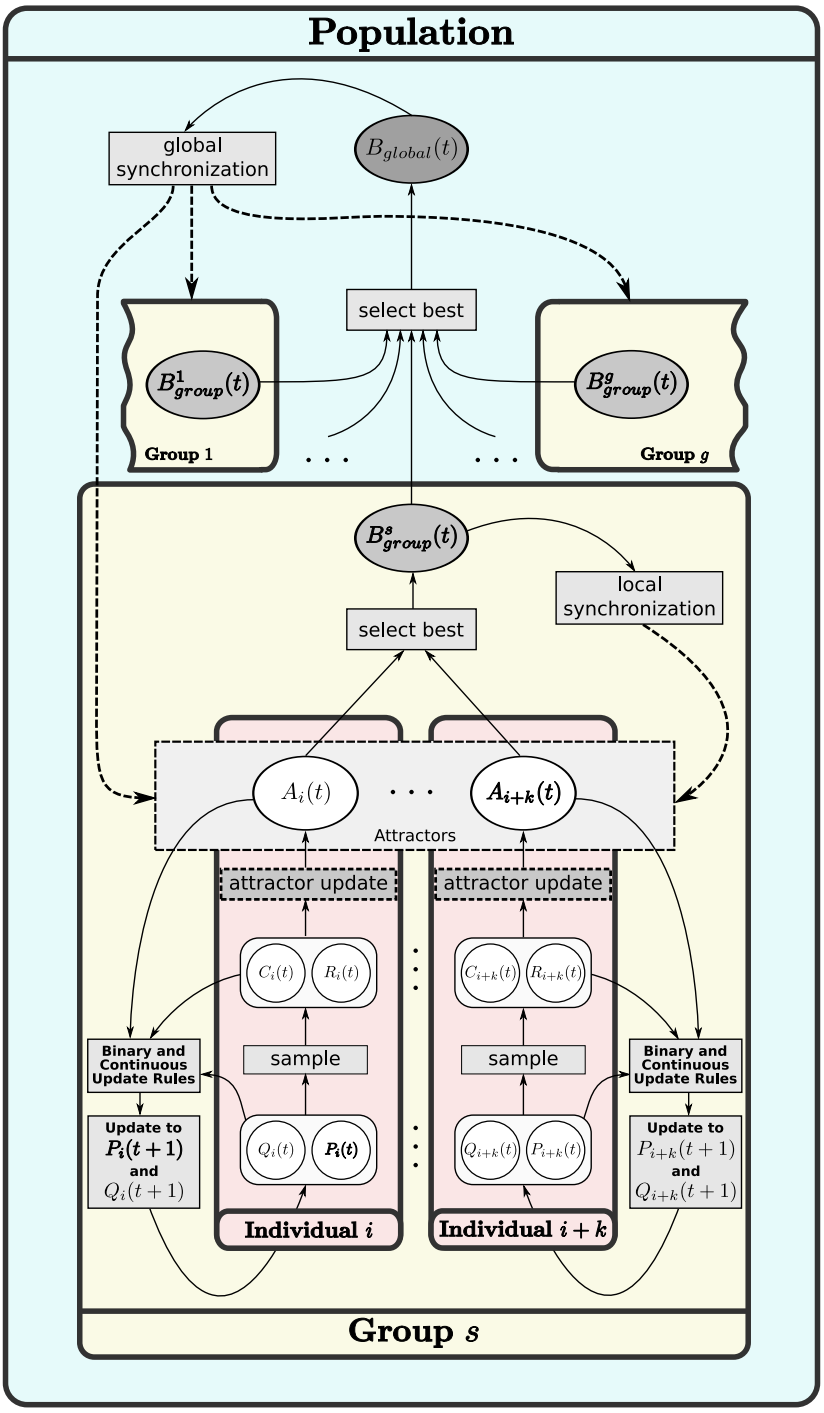


Figure 7.1: Three interacting levels can be distinguished in hHM-EDA: The individual, group and population level.

To each individual i, a solution A_i consisting of a binary and a continuous subcomponent is attached acting as an attractor for $\mathcal{H}_i$. Every generation $S_i(t)$ and A_i are compared in terms of their fitness. If A_i is better than $S_i(t)$, an update operation is applied on the corresponding model $\mathcal{H}_i$. Each representation uses its corresponding

update operator to drive the probabilistic model. The binary probabilistic model Q_i is updated using the rotation gate as employed in vQEA, and the continuous model P_i is modified by the mean and standard deviation shift as introduced for cHM-EDA. Thus the hybrid algorithm requires the setting of three learning rates for the model update: the learning rate $\Delta\theta$ used in the rotation gate to update a $\mathcal{Q}$bit, and the two learning rates θ_μ and θ_σ to update the Gaussian mean and standard deviation respectively.

Similarly to vQEA and cHM-EDA, the update policy of an attractor A_i can follow either an elitist or a non-elitist strategy. The choice of the update policy has great consequences for the algorithm and changes its behaviour completely. Since no experimental condition could be identified that favoured the elitist attractor update policy for vQEA and cHM-EDA, we concentrate on the non-elitist version in hHM-EDA.

GROUPS The second level corresponds to *groups*. The population is divided into g groups each containing k individuals having the ability of synchronising their attractors. For that purpose, the best attractor (in terms of fitness) of a group, denoted B_{group}, is stored at every generation and is periodically distributed to the group attractors. This phase of local synchronisation is controlled by the parameter S_{local}.

POPULATION The set of all $p = g \times k$ individuals forms the *population* and defines the topmost level of the multi-model approach. As for the groups, the individuals of the population can synchronise their attractors, too. For that purpose, the best attractor (in terms of fitness) among all groups, denoted B_{global}, is stored every generation and is periodically distributed to the group attractors. This phase of global synchronisation is controlled by the parameter S_{global}.

7.2 MBOA

Similarly to hHM-EDA, MBOA also belongs to the class of EDA and can be formulated for continuous and discrete search spaces. Thus, we discuss this method here in greater detail. According to the classification of EDA given in (Pelikan et al., 1999), MBOA belongs to the third class of EDA which means it is able to explicitly model multi-variate interactions. The algorithm is based on decision trees (Friedman & Goldszmidt, 1998) that have been successfully employed already in the hierarchical Bayesian Optimisation Algorithm (hBOA) (Pelikan, Goldberg, & Sastry, 2000). Indeed, it was shown in (Ocenasek & Schwarz, 2002) that MBOA is backward-

compatible to hBOA and achieves very similar performance on several deceptive binary benchmark functions.

7.2.1 *Principle*

MBOA attempts to learn a probability distribution $P(X)$ that is approximated by the product of conditional probability distributions for each search variable X_i, $1 \leq i \leq N$, given a set of influencing variables Π_i:

$$P(X) = P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i|\Pi_i) \tag{7.4}$$

The method is initialised through generating a base population of n random individuals. Then a number of $\tau \times n$ promising individuals is selected using a tournament selection, forming a population D. Parameter $\tau \in \mathbb{R}$ is typically set to 0.5. Based on D, the probabilistic model of MBOA is rebuilt every generation. The model consists of a set of N decision trees, one for each search variable X_i. Each tree defines the conditional distributions $P(X_i|\Pi_i)$, where Π_i denotes the set of variables that impact the outcome X_i. The nodes in the i-th decision tree are formed by the variables in Π_i. The i-th tree is recursively constructed by cutting the domain of the variables $\Pi_{ij} \in \Pi_i$ into parts, where X_i is assumed to be mutually independent. In the continuous domain, real-valued split boundaries are defined which create intervals for the variables Π_{ij}. The leafs of the tree are modelled by a uni-variate density function using Gaussian kernels. Thus, the search space is partitioned into subspaces, in which the search variables can be identified by a simple localised search. The Gaussian kernels are used to explore each partition locally.

An offspring population is sampled from the constructed N decision trees, which in turn is used to replace part of the base population. In order to preserve diversity in the population, a restricted tournament replacement is employed as introduced in (Pelikan & Goldberg, 2001).

In Figure 7.2, an example for a learnt model in MBOA is presented for a two-dimensional problem consisting of the two search variables X_1 and X_2. Using Equation 7.4, the joint probability distribution $P(X_1, X_2)$ is factorised into $P(X_1, X_2) = P(X_1)P(X_2|X_1)$. Given the density function $p_1(x_1)$ for the distribution $P(X_1)$, a

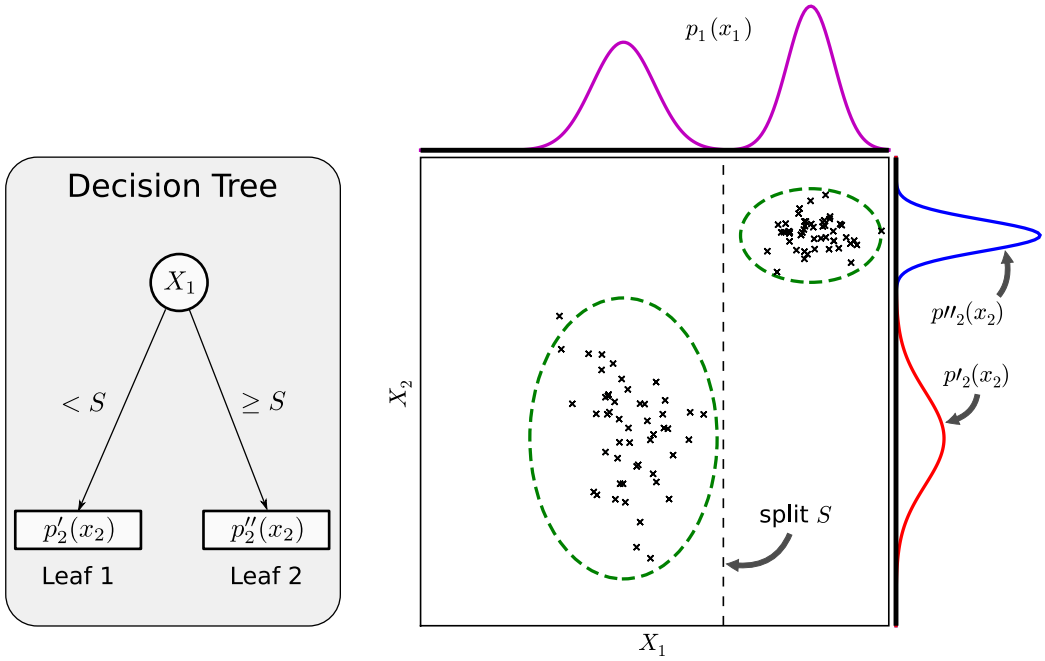


Figure 7.2: Illustration of a trained model in MBOA for a two-dimensional problem. The joint probability distribution $P(X_1, X_2)$ is factorised using Equation 7.4 into $P(X_1, X_2) = P(X_1)P(X_2|X_1)$. The density function $p_1(x_1)$ corresponds to the distribution $P(X_1)$. Value $S \in \mathbb{R}$ is a continuous split boundary, that partitions X_1 into two parts. For each part a different local distribution is defined to sample X_2. Thus, the density of $P(X_2|X_1)$ is given by $p_2(x_2|x_1)$ with $p_2(x_2|x_1) = p'_2(x_2)$, if $x_1 < S$ and $p_2(x_2|x_1) = p''_2(x_2)$, if $x_1 \geq S$. $p'_2(x_2)$ and $p''_2(x_2)$ can be modelled by Gaussian kernels.

decision tree for variable X_2 is learnt. The tree partitions X_1 into two parts, based on a continuous split boundary $S \in \mathbb{R}$. Thus, the density of $P(X_2|X_1)$ is given by

$$p_2(x_2|x_1) = \begin{cases} p'_2(x_2) & \text{if } x_1 < S \\ p''_2(x_2) & \text{if } x_1 \geq S \end{cases} \tag{7.5}$$

The densities $p'_2(x_2)$ and $p''_2(x_2)$ can be modelled by a single Gaussian probability density function, but also a mixture of Gaussian kernels or linear regression models could be employed (Ocenasek & Schwarz, 2002).

Additional information about MBOA can be found in (Kern et al., 2004) and in (Ocenasek & Schwarz, 2002), in which also the actual construction of the decision trees and the computation of the split boundaries are explained in detail. A complete study on MBOA is presented in the PhD dissertation of Ocenasek (2002).

7.2.2 *Comparison to hHM-EDA*

Although both MBOA and hHM-EDA belong to the family of EDA, the methods differ significantly from each other. Here the differences in the employed probabilistic model, the model update, the sampling and replacement strategies and the structure of the population of individuals are discussed.

The probabilistic model used in MBOA is based on N decision trees (one for each search variable), which are rebuilt in every generation of the evolutionary process. Generating these trees is a very complex and expensive operation. Indeed, it was shown in (Ocenasek & Pelikan, 2004), that the model construction easily becomes the most costly operation of the method: experimenting with spin glass benchmarks, MBOA spent nearly 95% of the execution time on building the decision trees. This situation motivated an implementation of the method on parallel hardware (Ocenasek & Pelikan, 2004). As a consequence, the capability of modelling complex variable dependencies comes at the price of a significant computational overhead. Furthermore, for many practical applications this overhead might not even result in any advantage compared to much simpler methods, *cf. e.g.* the work of (Johnson & Shapiro, 2001) on comparing different evolutionary algorithms on feature selection problems.

For hHM-EDA, on the other hand, a much simpler model is used. It belongs to the first class of EDA according to the classification scheme given in (Pelikan et al., 1999) and is based on a number of independent Gaussian and Bernoulli distributions (*i.e.* string of $\mathcal{Q}$bits). Since each individual maintains its own probabilistic model, hHM-EDA is a multi-model EDA which is in contrast to MBOA. The advantages of this unique approach have been discussed at length in the corresponding chapters about vQEA and cHM-EDA respectively. The computational overhead of hHM-EDA according to model management is small and most of the resources are devoted to the evaluation of the fitness function.

The model update in MBOA consists of a complete reconstruction of all N decision trees, while the model in hHM-EDA is updated incrementally and thus evolves during the optimisation process. The current state of the model is a direct result of earlier updates introducing a memory about information of previously visited areas in the search landscape. Compared to MBOA, the model update in hHM-EDA is fast and inexpensive, despite the fact that a multiple probabilistic model is maintained.

In order to sample a population of individuals, MBOA traverses down the decision trees for all search variables and samples the distribution specified at the leaf of the tree. A comparatively large number of individuals is generated, which replace part of

the population created in the previous generation. As mentioned earlier, a restricted tournament replacement is used. Sampling the model in hHM-EDA is straightforward since the distributions are independent. Unlike MBOA, hHM-EDA samples only a single solution from each model.

Finally, hHM-EDA maintains a structured population which allows an information exchange between the individual models while for MBOA no such structure exists.

7.2.3 *Conclusion*

Both MBOA and hHM-EDA are EDA and explore the search space probabilistically. Nevertheless, their characteristics and thus potential applications are very different. MBOA is an advanced and complex algorithm which is specialised to deal with the strong variable interactions inherent in some optimisation problems. The computational overhead of MBOA is significant and the execution time spent on constructing the decision trees may easily rival the time required for the fitness evaluations. Only if the fitness function is expensive itself, the overhead becomes negligible. Thus, MBOA seems suitable for problems with costly fitness functions that also require the discovery of variable linkage in order to be solved properly. For such problems, parallel hardware becomes a necessary requirement.

The proposed hHM-EDA, on the other hand, is a much more light-weight algorithm and its computational overhead is negligible for most problems. The multiple probabilistic model is an original mechanism capable of compensating for a limited number of decision errors due to variable linkage. Thus, hHM-EDA is a flexible, less specialised tool, suitable for a variety of optimisation problems.

7.3 PERFORMANCE ANALYSIS

In this section, the performance of hHM-EDA is evaluated. Results are compared to a selection of contemporary continuous-only and binary-only optimisation methods, along with the already discussed MBOA. For the binary-only optimisers three first-level binary EDA are considered that have been discussed already in chapter 4, namely UMDA (Mühlenbein & Paass, 1996), PBIL (Baluja, 1994) and cGA (Harik et al., 1999). Using binary representations to explore continuous search spaces is a typical scenario in the context of traditional genetic algorithms, *cf. e.g.* the early work in (Michalewicz & Janikow, 1991) and also (Maniezzo, 1994), in which a binary

GA was applied on a heterogeneous optimisation problem. Bit strings of pre-defined length are mapped into real values by a Gray encoding.

Using a continuous representation to explore a binary landscape, on the other hand, is less common. An example can be found in (Leung et al., 2003) where a real-coded GA evolves the topology and the weight matrix of a neural network. Since a continuous representation is used, real values $x \in \mathbb{R}$ of the chromosome are converted into bits using a simple mapping:

$$\delta(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{else} \end{cases} \tag{7.6}$$

This mapping enables a numerical optimiser to explore a binary search space. Due to the excellent performance reported in the previous chapter, the CMA-ES and cHM-EDA are used for the performance analysis presented here.

7.3.1 *Benchmark problem*

Due to the lack of a suitable benchmark suite for heterogeneous optimisation problems, a simple benchmark is proposed here. A minimisation problem is considered that contains two equally sized search landscapes: a binary and a continuous one. The dimensionality (number of variables) of each landscape is denoted by N. Target vectors representing the global optimum of the problem are specified for each landscape: a binary vector $B^* = (b_1^*, \ldots, b_N^*)$ and a continuous vector $R^* = (r_1^*, \ldots, r_N^*)$. A solution for this problem is denoted as $S = (B, R)$, where $B = (b_1, \ldots, b_N)$ and $R = (r_1, \ldots, r_N)$ represent the binary and the real part of the problem respectively. The goal is to evolve a solution S, such that it becomes equivalent to the target solution $S^* = (B^*, R^*)$. More specifically, the fitness function in this problem is defined as the Euclidean distance between the real part R of a solution S to the real part R^* of the target solution S^*. The binary part B of the solution acts as a mask in the computation of the distance: only if bit $b_i = 1$, does the corresponding real value r_i contribute to the computation of the difference. Furthermore, if $b_i \neq b_i^*$ a penalty is added to the overall fitness of the solution. The complete fitness function is described in detail in Algorithm 4. The global optimum is reached if the fitness becomes $f = 0$.

The problem is designed to resemble a typical wrapper-based feature selection scenario. The feature space is represented by the binary solution sub-component while the parameter space of the classification method is reflected by the real-valued sub-component. If a certain bit (feature) $b_i \in B$ is wrongly selected, *i.e.* $b_i \neq b_i^*$, the

Algorithm 4 Computes the fitness f of the solution $S = (B, R)$

Require: $B = (b_1, \ldots, b_N)$ and $R = (r_1, \ldots, r_N)$

```
f ⇐ 0
for i = 1 to N do
    if b_i = 1 and b*_i = 1 then
        d ⇐ (r*_i − r_i)^2
    else if b_i ≠ b*_i then
        d ⇐ (r*_i)^2
    else
        d ⇐ 0
    end if
    f ⇐ f + d
end for
```

solution $S = (B, R)$ receives a penalty $(r_i^*)^2$. Thus, different bits (features) may have a different significance, since different fitness penalties are associated with them. On the other hand, if the bit (feature) is correctly selected, *i.e.* $b_i = b_i^* = 1$, the size of the fitness penalty depends on the quality of the variable (parameter of the classifier) r_i of the real solution part R. Thus, even if the optimisation method correctly selects a certain feature, the fitness penalty may be large if the classification method is poorly parametrised. Both solution sub-components need to *co-operate* in order to minimise the fitness penalties.

In the following experiment, the target solution $S^* = (B^*, R^*)$ was chosen in dependence of the problem size N:

$$\begin{aligned} B^* &= (\overbrace{1, \ldots, 1}^{\times \frac{N}{2}}, \overbrace{0, \ldots, 0}^{\times \frac{N}{2}}) \\ R^* &= (\overbrace{p_{max}, \ldots, p_{min}}^{\times \frac{N}{2},\ \text{equi-distant}}, \underbrace{p_{max}, \ldots, p_{min}}_{\times \frac{N}{2},\ \text{equi-distant}}) \end{aligned} \tag{7.7}$$

The parameters p_{min} and p_{max} denote the minimum and maximum fitness penalty assigned to a certain bit. Penalties are equi-distantly distributed over the first $\frac{N}{2}$ and last $\frac{N}{2}$ elements of the real-valued solution sub-component. In the experiments discussed later in this chapter, $p_{min} = 0.5$ and $p_{max} = 1$ are chosen.

It is noteworthy that, using this configuration, only the first $\frac{N}{2}$ real-valued elements $r_i \in R$ have to be optimised by the algorithm. The other $\frac{N}{2}$ elements become irrelevant in the fitness computation, if the algorithm has evolved zeroes at the last $\frac{N}{2}$ positions of the binary vector.

Since different fitness penalties are assigned to each binary element, all bits correspond to a different marginal fitness contribution. In the GA domain, such a situation is also referred to as *salient* building blocks (Thierens, Goldberg, & Pereira, 1998). Due to the difference of significance, the convergence behaviour of the binary probabilistic model is directly affected. More specifically, a sequential convergence of variables is expected, starting with the ones with the highest salience and finishing with the ones with the lowest salience. This sequential convergence phenomenon is called *domino convergence* and was first mentioned in (Rudnick, 1992).

7.3.2 *Configuring hHM-EDA*

Similarly to the parameter analysis in chapter 6, we focus here on a specific population structure. The structure consisting of ten individuals that are fully synchronised in every generation is directly adopted from previous experiments on vQEA and cHM-EDA. Although this setting has generally reported good optimisation performance, it is noted that this structure might not be necessarily optimal for hHM-EDA. Nevertheless, we restrict the analysis here to this simple configuration only and leave the exploration of more complex population hierarchies for future research.

Learning rates θ_μ and θ_σ

The first series of experiments investigates the hypothesis that the three guidelines developed for cHM-EDA in chapter 6, section 6.2.1, are also working in the context of hHM-EDA. For this analysis, the learning rate of the rotation gate is fixed to specific values $\Delta\theta \in \{0.0005\pi, 0.001\pi, 0.005\pi\}$. For each $\Delta\theta$, the parameters θ_μ and θ_σ are varied and the success rate of hHM-EDA is computed based on 25 independent runs on the proposed heterogeneous benchmark problem. A run is considered successful if the final achieved fitness value is lower than 10^{-5}. The success rate is defined as the ratio between the successful and total number of runs. Different problem sizes N are investigated and a maximum number of $N \times 4 \times 10^3$ FES is allowed for the optimiser.

Figure 7.3 presents the success rate of hHM-EDA in dependence of the two learning rates θ_μ and θ_σ for the problem sizes $N = 25$, $N = 50$, $N = 100$ and $\Delta\theta = 0.001\pi$. The darker the colour in these diagrams, the higher the success rate of the particular parameter setting. It is clearly demonstrated that a variety of settings are suitable for solving the problem. We also note that the setting of the mean shift θ_μ has only a low impact on the performance of the algorithm. It is also only slightly

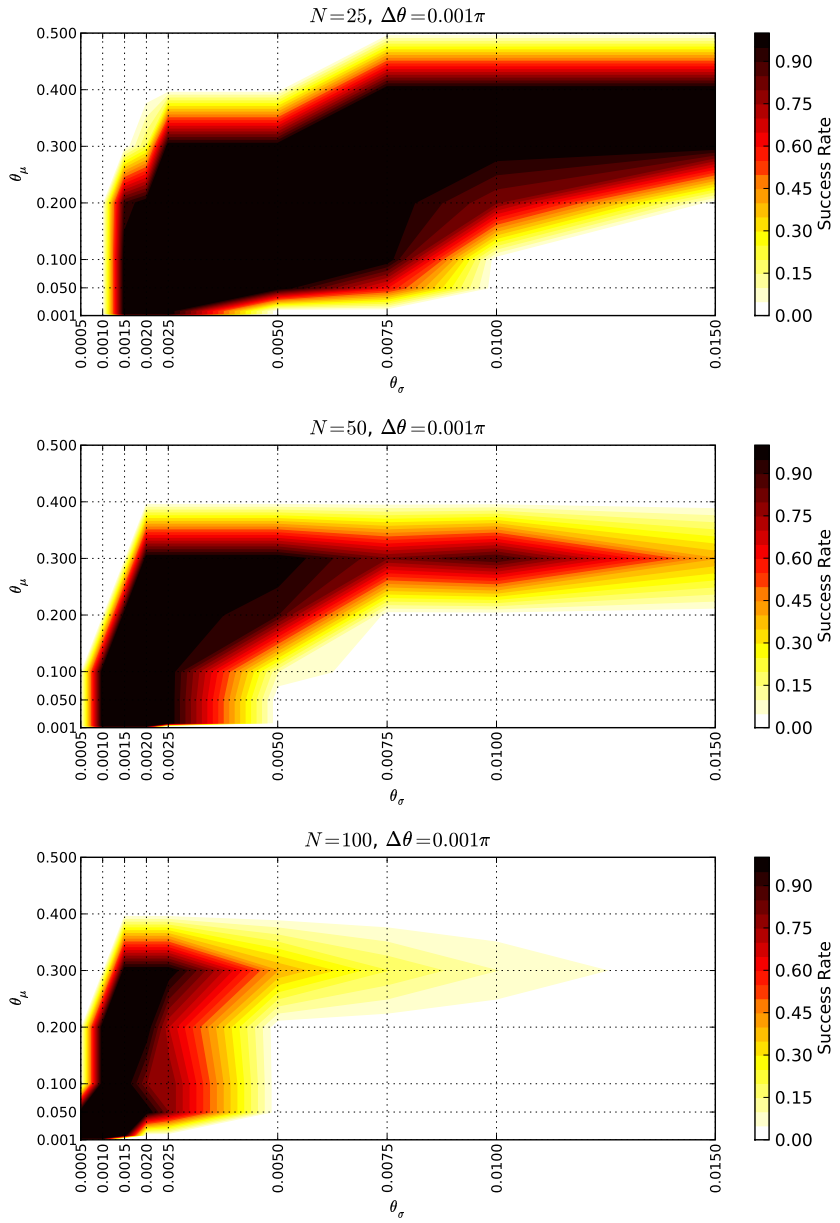


Figure 7.3: The success rate of hHM-EDA in dependence of the two learning rates θ_μ and θ_σ. The parameter $\Delta\theta$ of the rotation gate was fixed to 0.001π. Different problem sizes of the benchmark are presented. The diagrams show the average of 25 independent runs. The low impact of the learning rate θ_μ of the mean shift is clearly demonstrated. The learning rate θ_μ is dependent on the problem size N.

affected by the increase of the problem size N. The learning rate θ_σ, on the other hand, is a critical parameter that strongly depends on the problem size. The larger the size N, the smaller θ_σ has to be set in order to achieve optimal performance. Very

similar results are reported for $\Delta\theta = 0.0005\pi$ and $\Delta\theta = 0.005\pi$[1]. Almost identical observations have been made using cHM-EDA in chapter 6, section 6.2.1.

From these experimental results it is concluded that the three guidelines derived for cHM-EDA are also suitable for configuring hHM-EDA. Fine-tuning the mean shift θ_μ is of low importance while the standard deviation shift θ_σ should be adjusted in dependence with the problem size.

Learning rate $\Delta\theta$

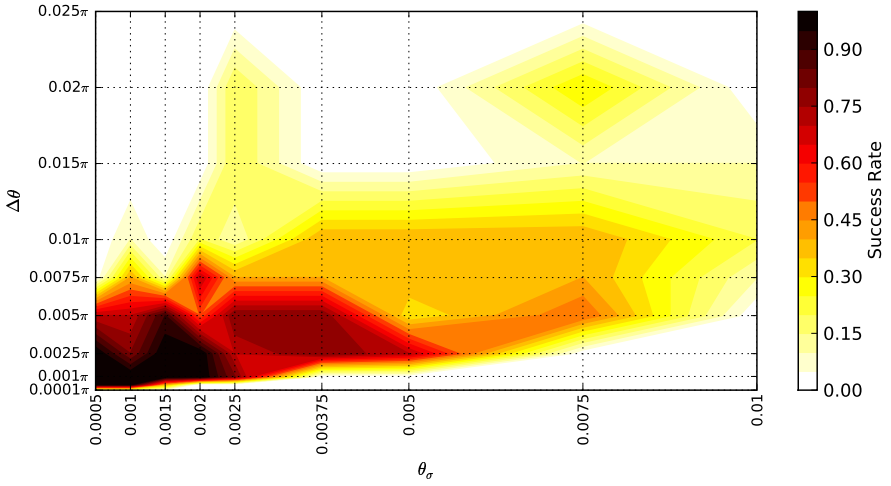
Since the mean shift rate θ_μ has only a low impact on the performance of hHM-EDA, we now focus on the relationship between the learning rate $\Delta\theta$ of the binary model and the standard deviation shift θ_σ. For this analysis θ_μ is fixed to $\hat{\theta}_\mu = 0.05$, which was earlier introduced as the default value for this parameter. Due to the explicit linkage between the binary and continuous search variables, several local optima exist in the fitness landscape of the heterogeneous benchmark problem. A known remedy against premature convergence of QEA and vQEA towards local optima in multi-modal landscapes is the use of a modified rotation gate operator, which was introduced as the H_ϵ gate in (Han & Kim, 2004). For vQEA the H_ϵ gate was already utilised in the performance and noise analysis presented in chapters 3, 4 and 5. In the following experiments, the two configurations $\epsilon = 0$ and $\epsilon = \sin^2(0.02\pi)$ are investigated, where for $\epsilon = 0$ the H_ϵ gate equals to the standard rotation gate, while $\epsilon = \sin^2(0.02\pi)$ was introduced as an appropriate default configuration for H_ϵ in chapter 3.

Figure 7.4 presents the average success rate showing the interdependence of $\Delta\theta$ and θ_σ obtained from 25 independent runs of hHM-EDA on the benchmark problem for a problem size $N = 100$. The darker the colour in these diagrams, the higher the success rate of the particular parameter setting.

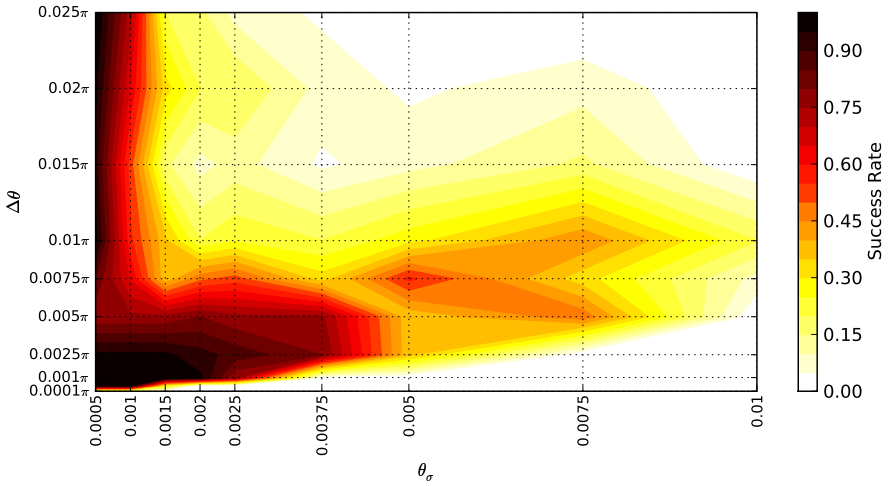
In the case of the standard rotation gate, *cf.* Figure 7.4a, a certain correlation between $\Delta\theta$ and θ_σ is observed. Clearly the best performance is reported when small values for both learning rates are used. If $\Delta\theta$ is increased, θ_σ also needs to increase (and vice versa) in order to maintain a non-zero success rate. Particularly a combination of a small (large) $\Delta\theta$ and a large (small) θ_σ is not suitable for the algorithm.

In the case of the H_ϵ gate, a similar correlation is noted, but additionally another effect impacts the performance of the method, *cf.* Figure 7.4b. Very surprising is the low sensitivity of the algorithm to the learning rate of the binary model. Almost

1 Due to the similarities of the figures, the results for $\Delta\theta = 0.0005\pi$ and $\Delta\theta = 0.005\pi$ are not presented here.



(a) Standard rotation gate ($\epsilon = 0$)



(b) H_ϵ gate, $\epsilon = \sin^2(0.02\pi)$

Figure 7.4: The success rate of hHM-EDA in dependence of the two learning rates $\Delta\theta$ and θ_σ. The learning rate θ_μ was fixed to the default value $\hat{\theta}_\mu = 0.05$. In **(a)** the standard rotation gate was used, which allows the convergence of the probability amplitudes α and β to 0 or 1. Using the H_ϵ gate in **(b)** prevents the complete convergence of the amplitudes, which decreases the sensitivity of hHM-EDA to the parameter $\Delta\theta$. Almost any $\Delta\theta$ is suitable, as long as the standard deviation shift θ_σ is small enough.

any $\Delta\theta$ is suitable, as long as the standard deviation shift θ_σ is small enough. The H_ϵ operator prevents the convergence of the binary probabilistic model towards 1 or 0, and instead defines for the two values $|\alpha|^2$ and $|\beta|^2$ of a $\mathcal{Q}$bit a minimal and a maximal probability, *i.e.* ϵ and $1 - \epsilon$ respectively. Due to the residual probabilities ϵ and $1 - \epsilon$ a certain mechanism is employed by the algorithm that is similar to the

bit-flip mutations used in a GA. With low probabilities, a certain $\mathcal{Q}$bit may collapse towards 1 (or 0), although its amplitudes have evolved close towards 0 (or 1). Thus, at least for some problems, premature convergence of a specific bit due to hitch-hiking phenomena may be compensated through the use of the H_ϵ gate.

In the context of the heterogeneous benchmark problem, the H_ϵ gate is highly advantageous and counteracts hitch-hiking efficiently, since larger learning rates $\Delta\theta$ not only increase the risk of hitch-hiking effects, but at the same time also increase the impact of the mutations on the probabilistic model. If a certain bit-flip mutation is evaluated to be positive, *i.e.* the fitness of the mutated solution improves, the corresponding $\mathcal{Q}$bit is updated towards the mutated bit value. Larger learning rates result in larger model shifts, which in turn increase the probability of mutations for the $\mathcal{Q}$bit in the next generation. Thus, in succeeding generations the state of a $\mathcal{Q}$bit may completely invert due to the impact of earlier mutations.

Since the mutations occur with low probabilities only and are entirely random for each bit, many generations are required to mutate the non-optimal bits in the binary sub-component of a solution. If a certain $\mathcal{Q}$bit $Q_i^{(j)}$ is non-optimally converged, the corresponding continuous model $P_i^{(j)}$ has to maintain enough diversity, *i.e.* the standard deviations $\sigma_i^{(j)}$ need to stay reasonably large, until the desired mutation occurs, in order to be able to optimise the continuous search variable r_j after the bit b_j is mutated. This is due to the fact that the continuous variable r_j only contributes to the fitness computation if the corresponding bit $b_j = b_j^* = 1$. In any other case r_j is irrelevant in the fitness evaluation and its value is subject to genetic drift, since no selective pressure is provided by the fitness function. Thus, the described mutation mechanism works well only for small learning rates θ_σ, which prevents the premature convergence of $\sigma_i^{(j)}$ due to drift before a positive mutation at bit b_j occurs.

Conclusion

The most critical parameter in hHM-EDA is the learning rate θ_σ of the standard deviation shift that should be adjusted according to the number of variables in the problem to solve. Similar to cHM-EDA, the mean shift θ_μ is of low importance and can be fixed to standard values for most problems. Consequently, the three guidelines derived for configuring cHM-EDA are also suitable for hHM-EDA. Furthermore, configuring the learning rate $\Delta\theta$ for updating the binary probabilistic model is straightforward if the H_ϵ gate is used. An appropriate value for the parameter ϵ was presented in the undertaken experiments.

7.3.3 *Benchmark analysis*

In this section, hHM-EDA is applied to the proposed heterogeneous benchmark problem. In order to allow a comparison of results, a number of binary and continuous-only optimisation algorithms, namely UMDA, PBIL, cGA, vQEA, cHM-EDA and CMA-ES, are applied on the same benchmark. Additionally, we investigate the performance of MBOA as discussed above. In all experiments a problem size $N = 100$ is used which should present a certain challenge for the tested algorithms. Each method is allowed to perform a maximum number of $N \times 4 \times 10^3 = 4 \times 10^5$ FES. The search space was limited to the range $[-1, 1]$ for each search variable.

Two configurations of hHM-EDA are considered that are directly adopted from the analysis discussed in section 7.3.2. The first setting follows guideline I which results in the setting $\theta_\mu = \hat{\theta}_\mu = 0.05$ and $\theta_\sigma = \hat{\theta}_\sigma = \frac{1}{10 \times N} = 0.001$. The only difference of the second setting is a slightly faster rate $\theta_\sigma = 0.0015$. Both configurations use a small value, $\Delta\theta = 0.001\pi$, for the H_ϵ gate to update the binary probabilistic model, that was shown to be efficient in the previous analysis.

Optimal configurations for all tested methods were obtained through a comprehensive parameter analysis. In the case of UMDA, the choice of an appropriate population size n is critical. Different sizes in the range $[200, 2500]$ were investigated. The default ratio of 50% for the truncation selection is used. PBIL also requires the setting of a population size which was varied $n \in [50, 300]$. Additional parameters are the learning rate R_l and the mutation shift R_s. Similarly to the noise analysis in chapter 5, we assume $R_l = R_s$. Values were varied $R_l, R_s \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$. In total, 36 different parameter configurations were investigated for PBIL.

The only parameter of cGA is the virtual population size n that was optimised in the range $[150, 1250]$. For vQEA, a single group of ten fully synchronised individuals is tested while the learning rate of a H_ϵ gate[2] is varied $\Delta\theta \in \{0.001, 0.0025, 0.005, 0.0075, 0.01\}$. All binary methods use 12 bits to encode a single real value. A Gray encoding was used for the conversion of bit strings into a continuous value.

The CMA-ES employs special mechanisms that adapt most of its parameters automatically. According to (Auger & Hansen, 2005), only the initial starting points and the initial standard deviation of the method needs to be specified for a given problem. We adopt the strategy given in (Auger & Hansen, 2005) and set the initial

2 Default $\epsilon = \sin^2(0.02\pi)$ was used.

standard deviation to $10^{-2}(B-A)/2$, with $[A,B]^N = [-1,1]^N$ being the search interval of the benchmark. The initial starting points were uniformly sampled in the range $[-0.1, 0.1]^N$, which is slightly different from (Auger & Hansen, 2005), but in favour for the method[3]. No further parameter fine-tuning was attempted for this method. The Java implementation provided by Nikolaus Hansen[4] was used in the experiments.

For cHM-EDA, the default value for $\theta_\mu = \hat{\theta}_\mu$ is used and only $\theta_\sigma \in \{0.00025, \ldots, 0.0015\}$ is varied, which allows a direct comparison to hHM-EDA. The only difference between cHM-EDA and hHM-EDA is the different probabilistic model for the binary solution sub-component of the latter one. All continuous methods use Equation 7.6 to explore the binary solution sub-component.

MBOA only requires the proper setting of its population size n. Sizes are varied $n \in \{50, 100, 125, 150, 200, 250, 300\}$. These values correspond to the size of the base population in MBOA. Every generation, $\tau \times N$ new offspring are generated and evaluated, thus each generation requires the computation of $\tau \times N$ FES instead of N. Parameter $\tau \in \mathbb{R}$ was set to 0.5 as recommended as the default in (Kern et al., 2004). An official implementation of the method in the programming language C++ is provided by Jiri Ocenasek[5].

Results

The results of the parameter analysis can be found in Tables 7.1 and 7.2. For each setting of a method, the best, median, worst and mean performance along with the standard deviation obtained from 25 independent runs is presented in the columns. Additionally, the success rate as defined in section 7.3.2 is given. The most suitable configuration in terms of success rate is highlighted. In the cases where the success rate is not discriminative enough, the mean fitness and number of required FES are considered, in order to determine the most suitable setting.

hHM-EDA, vQEA, UMDA, cHM-EDA and MBOA all report a success rate of 100%. With cGA, 76% of the runs were successful, while not a single run reached the required fitness threshold using PBIL. It is also noted that the binary methods require a rather large population size due to the mapping of 100×12 bits into 100 real values. Because of this mapping, the overall precision of the optimisation is also affected. In the case of cHM-EDA, hHM-EDA, CMA-ES and MBOA, the optimisation was

3 In (Auger & Hansen, 2005) the initial starting points were uniformly drawn from $[A,B]^N$.
4 Available at http://www.lri.fr/~hansen
5 Available at http://jiri.ocenasek.com

Method	Setting	best	med	worst	mean	stdev	success rate
hHM-EDA	θ=0.001, θ_σ=0.001	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	100%
	θ=0.001, θ_σ=0.0015	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	100%
vQEA	θ=0.001	$1.41e-04$	$2.38e-04$	$2.82e-04$	$2.30e-04$	$3.42e-05$	0%
	θ=0.0025	$9.84e-06$	$1.72e-05$	$2.91e-05$	$1.73e-05$	$5.49e-06$	8%
	θ=0.005	$2.89e-06$	$5.41e-06$	$9.08e-06$	$5.52e-06$	$1.50e-06$	100%
	θ=0.0075	$2.41e-06$	$3.40e-06$	$5.30e-06$	$3.71e-06$	$9.42e-07$	100%
	θ=0.01	$2.07e-06$	$3.83e-06$	$5.15e-06$	$3.67e-06$	$7.78e-07$	100%
UMDA	n=200	$1.17e+00$	$1.55e+00$	$3.85e+00$	$1.84e+00$	$6.82e-01$	0%
	n=300	$5.84e-02$	$6.98e-01$	$2.02e+00$	$7.93e-01$	$5.23e-01$	0%
	n=400	$1.84e-02$	$8.69e-02$	$1.06e+00$	$2.66e-01$	$3.05e-01$	0%
	n=500	$7.66e-04$	$1.64e-02$	$7.59e-01$	$1.46e-01$	$2.30e-01$	0%
	n=600	$9.22e-05$	$5.72e-03$	$7.81e-01$	$4.37e-02$	$1.51e-01$	0%
	n=700	$6.04e-04$	$5.08e-03$	$4.28e-01$	$3.89e-02$	$1.03e-01$	0%
	n=800	$3.28e-05$	$1.14e-03$	$2.50e-01$	$1.32e-02$	$4.87e-02$	0%
	n=900	$1.15e-05$	$1.79e-04$	$5.64e-03$	$8.31e-04$	$1.28e-03$	0%
	n=1500	$1.21e-06$	$2.76e-06$	$2.54e-04$	$2.50e-05$	$5.46e-05$	72%
	n=2000	$1.21e-06$	$1.30e-06$	$3.33e-05$	$3.72e-06$	$6.84e-06$	92%
	n=2500	$1.21e-06$	$1.21e-06$	$4.68e-06$	$1.50e-06$	$7.49e-07$	100%
cGA	n=150	$3.72e-01$	$1.30e+00$	$3.12e+00$	$1.41e+00$	$6.70e-01$	0%
	n=250	$1.79e-02$	$3.33e-01$	$1.43e+00$	$4.27e-01$	$4.22e-01$	0%
	n=350	$2.82e-04$	$1.33e-02$	$5.32e-01$	$1.13e-01$	$1.56e-01$	0%
	n=450	$2.06e-04$	$3.83e-03$	$3.82e-01$	$4.12e-02$	$1.02e-01$	0%
	n=550	$1.94e-05$	$4.62e-04$	$5.74e-01$	$3.46e-02$	$1.20e-01$	0%
	n=750	$1.91e-06$	$5.94e-05$	$1.01e-03$	$2.09e-04$	$2.92e-04$	20%
	n=850	$1.42e-06$	$5.53e-06$	$2.92e-01$	$1.17e-02$	$5.73e-02$	72%
	n=900	$1.21e-06$	$4.15e-06$	$3.09e-05$	$7.26e-06$	$7.79e-06$	76%
	n=950	$1.64e-06$	$6.63e-06$	$4.11e-05$	$1.02e-05$	$9.14e-06$	60%
	n=1000	$5.70e-06$	$1.10e-05$	$5.56e-05$	$1.43e-05$	$1.03e-05$	36%
	n=1250	$1.33e-04$	$1.81e-04$	$3.34e-04$	$1.88e-04$	$4.27e-05$	0%
cHM-EDA	θ_σ=0.00025, θ_μ=$\hat{\theta}$	$2.02e-03$	$2.37e-03$	$2.72e-03$	$2.35e-03$	$1.79e-04$	0%
	θ_σ=0.0005, θ_μ=$\hat{\theta}$	$2.44e-07$	$3.13e-07$	$3.45e-07$	$3.07e-07$	$2.61e-08$	100%
	θ_σ=0.00075, θ_μ=$\hat{\theta}$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	100%
	θ_σ=0.001, θ_μ=$\hat{\theta}$	$0.00e+00$	$0.00e+00$	$6.83e-01$	$2.73e-02$	$1.34e-01$	96%
	θ_σ=0.0015, θ_μ=$\hat{\theta}$	$0.00e+00$	$0.00e+00$	$5.55e-01$	$4.56e-02$	$1.31e-01$	88%
CMA-ES		$0.00e+00$	$7.34e-06$	$4.40e-01$	$2.89e-02$	$1.00e-01$	52%
MBOA	N=50	$4.66e+00$	$8.11e+00$	$1.32e+01$	$8.20e+00$	$1.83e+00$	0%
	N=100	$0.00e+00$	$0.00e+00$	$1.38e+00$	$2.61e-01$	$3.59e-01$	56%
	N=125	$0.00e+00$	$0.00e+00$	$5.70e-01$	$6.22e-02$	$1.50e-01$	84%
	N=150	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	100%
	N=200	$3.93e-07$	$9.61e-06$	$2.24e-04$	$2.80e-05$	$4.91e-05$	52%
	N=250	$3.56e-04$	$1.37e-03$	$4.47e-03$	$1.43e-03$	$9.20e-04$	0%
	N=300	$2.74e-03$	$1.08e-02$	$1.64e-02$	$1.05e-02$	$3.71e-03$	0%

Table 7.1: Results of the parameter analysis for hHM-EDA, vQEA, UMDA, cGA, cHM-EDA, CMA-ES and MBOA. Shown is the best, median and worst run obtained from 25 independent runs. Additionally the mean and standard deviation of the runs, along with the success rate is presented (see text for a definition of the success rate). The most suitable setting in terms of success rate for each method is highlighted.

stopped when the fitness value dropped below 10^{-10}. In the tables, this situation is indicated by the value $0.00e+00$.

In Figure 7.5, the fitness evolution of the median run is presented. We note the logarithmic scale of the fitness axis. The proposed hHM-EDA is clearly the fastest optimiser among the tested algorithms on this benchmark, requiring only 12300 FES to achieve the desired solution accuracy of $\epsilon = 10^{-5}$ and 21700 FES to drop below a

Method	Setting	best	med	worst	mean	stdev	success rate
PBIL	n=50, R_l=R_s=0.001	$6.46e+00$	$7.53e+00$	$8.58e+00$	$7.52e+00$	$5.29e-01$	0%
	n=50, R_l=R_s=0.005	$1.17e-02$	$1.81e-02$	$2.56e-02$	$1.81e-02$	$3.17e-03$	0%
	n=50, R_l=R_s=0.01	$2.64e-03$	$3.45e-03$	$5.62e-03$	$3.61e-03$	$7.13e-04$	0%
	n=50, R_l=R_s=0.05	$1.80e-03$	$2.45e-03$	$3.29e-03$	$2.50e-03$	$4.21e-04$	0%
	n=50, R_l=R_s=0.1	$2.37e-03$	$3.93e-03$	$5.09e-03$	$3.89e-03$	$6.35e-04$	0%
	n=50, R_l=R_s=0.2	$5.06e-03$	$9.15e-03$	$1.33e-02$	$9.07e-03$	$1.64e-03$	0%
	n=100, R_l=R_s=0.001	$1.43e+01$	$1.63e+01$	$1.74e+01$	$1.62e+01$	$8.38e-01$	0%
	n=100, R_l=R_s=0.005	$1.14e-01$	$1.49e-01$	$1.66e-01$	$1.46e-01$	$1.25e-02$	0%
	n=100, R_l=R_s=0.01	$3.90e-03$	$5.84e-03$	$7.59e-03$	$5.87e-03$	$9.03e-04$	0%
	n=100, R_l=R_s=0.05	$3.57e-04$	$5.30e-04$	$7.53e-04$	$5.28e-04$	$9.72e-05$	0%
	n=100, R_l=R_s=0.1	$5.43e-04$	$7.14e-04$	$1.06e-03$	$7.44e-04$	$1.50e-04$	0%
	n=100, R_l=R_s=0.2	$8.41e-04$	$1.46e-03$	$2.30e-03$	$1.54e-03$	$3.36e-04$	0%
	n=150, R_l=R_s=0.001	$1.90e+01$	$2.06e+01$	$2.23e+01$	$2.07e+01$	$7.96e-01$	0%
	n=150, R_l=R_s=0.005	$4.76e-01$	$6.14e-01$	$7.37e-01$	$6.09e-01$	$6.72e-02$	0%
	n=150, R_l=R_s=0.01	$1.75e-02$	$2.52e-02$	$2.94e-02$	$2.39e-02$	$3.64e-03$	0%
	n=150, R_l=R_s=0.05	$2.06e-04$	$2.99e-04$	$4.70e-04$	$3.19e-04$	$6.61e-05$	0%
	n=150, R_l=R_s=0.1	$2.26e-04$	$3.47e-04$	$4.52e-04$	$3.42e-04$	$5.27e-05$	0%
	n=150, R_l=R_s=0.2	$3.96e-04$	$7.32e-04$	$1.07e-03$	$7.29e-04$	$1.72e-04$	0%
	n=200, R_l=R_s=0.001	$1.86e+01$	$2.30e+01$	$2.40e+01$	$2.25e+01$	$1.23e+00$	0%
	n=200, R_l=R_s=0.005	$1.50e+00$	$1.80e+00$	$2.08e+00$	$1.80e+00$	$1.64e-01$	0%
	n=200, R_l=R_s=0.01	$5.61e-02$	$7.43e-02$	$9.62e-02$	$7.51e-02$	$1.12e-02$	0%
	n=200, R_l=R_s=0.05	$2.07e-04$	$3.02e-04$	$6.24e-04$	$3.23e-04$	$1.08e-04$	0%
	n=200, R_l=R_s=0.1	$1.44e-04$	$2.60e-04$	$1.11e-02$	$6.80e-04$	$2.13e-03$	0%
	n=200, R_l=R_s=0.2	$1.81e-04$	$4.16e-04$	$4.14e-01$	$1.70e-02$	$8.09e-02$	0%
	n=250, R_l=R_s=0.001	$2.06e+01$	$2.41e+01$	$2.57e+01$	$2.41e+01$	$1.11e+00$	0%
	n=250, R_l=R_s=0.005	$3.01e+00$	$3.51e+00$	$4.37e+00$	$3.56e+00$	$3.08e-01$	0%
	n=250, R_l=R_s=0.01	$1.61e-01$	$1.90e-01$	$2.39e-01$	$1.90e-01$	$1.97e-02$	0%
	n=250, R_l=R_s=0.05	$1.74e-04$	$3.82e-04$	$1.16e-03$	$4.22e-04$	$1.82e-04$	0%
	n=250, R_l=R_s=0.1	$9.19e-05$	$2.52e-04$	$6.13e-04$	$2.68e-04$	$1.19e-04$	0%
	n=250, R_l=R_s=0.2	$1.24e-04$	$3.23e-04$	$3.75e-01$	$1.55e-02$	$7.34e-02$	0%
	n=300, R_l=R_s=0.001	$2.39e+01$	$2.54e+01$	$2.67e+01$	$2.54e+01$	$6.77e-01$	0%
	n=300, R_l=R_s=0.005	$5.10e+00$	$5.61e+00$	$6.77e+00$	$5.72e+00$	$3.98e-01$	0%
	n=300, R_l=R_s=0.01	$3.19e-01$	$4.16e-01$	$5.03e-01$	$4.13e-01$	$4.53e-02$	0%
	n=300, R_l=R_s=0.05	$2.73e-04$	$8.23e-04$	$2.46e-02$	$1.79e-03$	$4.67e-03$	0%
	n=300, R_l=R_s=0.1	$1.17e-04$	$3.25e-04$	$2.56e-02$	$1.73e-03$	$5.19e-03$	0%
	n=300, R_l=R_s=0.2	$9.08e-05$	$3.68e-04$	$9.24e-01$	$3.75e-02$	$1.81e-01$	0%

Table 7.2: Results of the parameter analysis for PBIL. Shown is the best, median and worst run obtained from 25 independent runs. Additionally the mean and standard deviation of the runs, along with the success rate is presented (see text for a definition of the success rate). The most suitable setting in terms of success rate for PBIL is highlighted.

precision of 10^{-10}. The fitness is exponentially minimised resulting in a linear curve on the logarithmic scale of the ordinate.

Particularly interesting is the fitness evolution of PBIL, since a number of stepwise fitness improvements are observed. This behaviour is caused by mutations having a positive impact on the fitness of a solution. Mutations become very important in the later stages of the optimisation process when the probabilistic model has almost converged towards a specific solution candidate in the search space. Mutating a wrongly evolved bit in the binary solution sub-component can result in an especially significant fitness improvement of the overall solution. Since a comparably large mutation shift $R_s = 0.1$ is used, an improvement due to mutation can be efficiently exploited by PBIL.

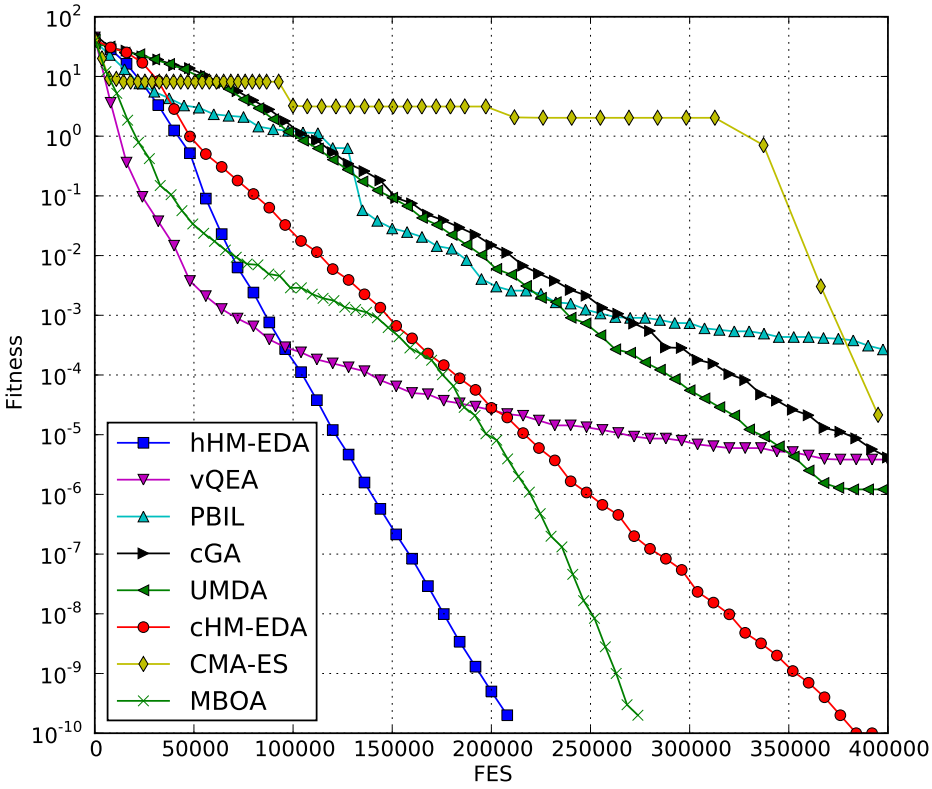
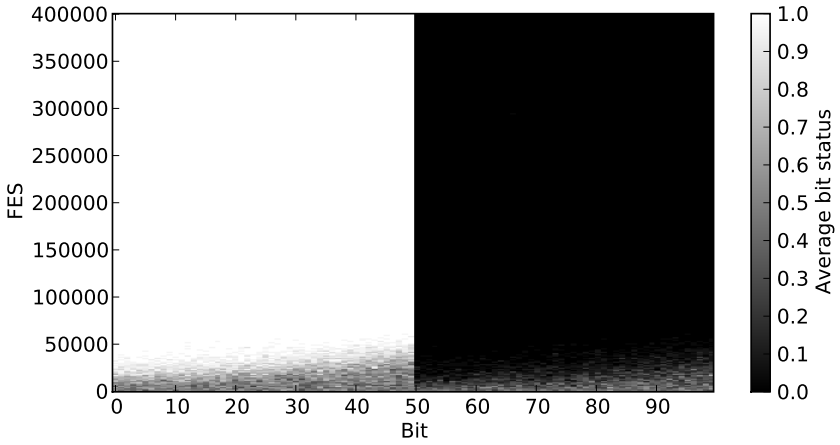


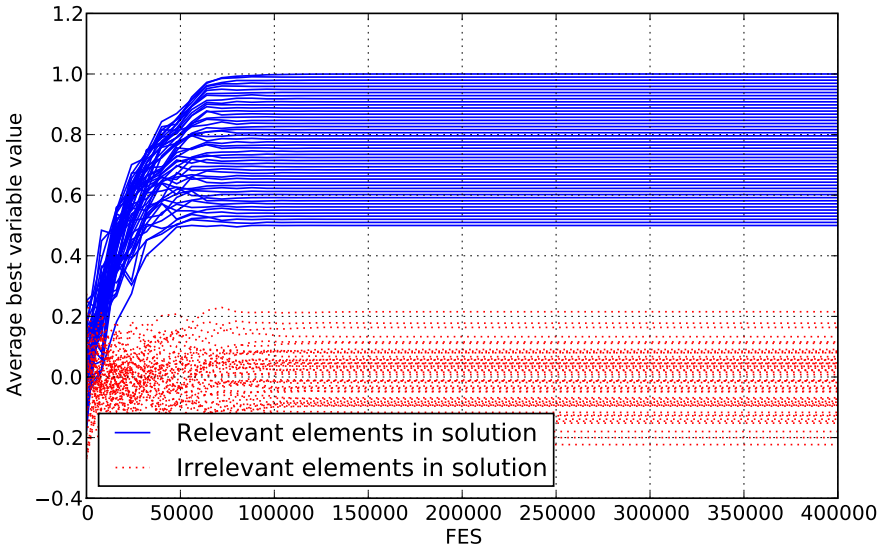
Figure 7.5: Evolution of the median fitness for all tested algorithms on the heterogeneous benchmark problem. Results are obtained from 25 independent runs. Due to the mapping from bit values to the continuous domain, the binary methods allow a minimal solution quality of $\approx 10^{-6}$ only. As the continuous optimisers are more precise, the evolution was stopped when the fitness value dropped below 10^{-10}.

The step-wise fitness evolution of CMA-ES, on the other hand, has an entirely different reason. It reflects the local restarts of the method after getting stuck on some non-optimal solution during the evolutionary process. In the presented median run, CMA-ES performed four independent restarts, the first finishing after $88,483$ FES, the second after $190,799$ FES, the third after $327,562$ FES, while the fourth restart exhausted the maximum number of FES and achieved the best results. That means, if the initial population of CMA-ES represents a solution close to the optimum, the method can converge towards it very quickly. Indeed, the fastest run of CMA-ES required only three restarts and a total of $294,065$ FES to achieve the precision of 10^{-10}.

All 25 runs of cHM-EDA solved the problem reliably in the given maximum number of FES. Since the learning rate $\theta_\sigma = 0.00075$ for cHM-EDA is two times smaller



(a) Evolution of binary solution sub-component



(b) Evolution of continuous solution sub-component

Figure 7.6: Evolution of binary and continuous solution sub-component using hHM-EDA. Results are averaged from 25 independent runs. Dark colours in **(a)** correspond to an average bit status of 0, white colours a status of 1. The domino convergence effect due to different salience of the bits is clearly visible in the figure. Simultaneously the continuous search space is optimised, *cf.* **(b)**. Only the first 50 variables are subject to optimisation, if the binary solution was identified correctly. The irrelevant variables are subject to genetic drift and converge randomly.

than in hHM-EDA, the latter is also significantly faster. The overall fitness evolution of the method is very similar to hHM-EDA.

MBOA, on the other hand, reports a very different convergence behaviour. The optimisation performance is comparatively fast in early stages of the run, but slows

down significantly after $\approx 0.5 \times 10^5$ FES, increases again after $\approx 1.5 \times 10^5$ FES and finally converges towards the optimum at an exponential rate. It was also noted that MBOA is able to explore the binary search space very efficiently. The binary model of the presented median run, for example, converged after only $17,100$ FES, while the remaining $232,800$ FES were used to optimise the continuous model.

This observation suggests a very competitive performance of MBOA on binary optimisation problems, but a comparably slow convergence rate on numerical problems. In (Kern et al., 2004), very similar results are reported. Here, several continuous EA, *i.e.* the Cumulative Step Size Adaptation Evolutionary Strategy (CSA-ES), CMA-ES, the Iterated Density Estimation Evolutionary Algorithm (IDEA) and MBOA were experimentally compared to each other using well-known numerical benchmark problems. Especially on simple uni-modal, separable problems, MBOA was shown to be less competitive than the considered ES. Furthermore, it has been demonstrated in (Kern et al., 2004), that although good results could be obtained on separable multi-modal functions, MBOA was not able to optimise any of the tested non-separable functions at all.

Similar to MBOA, also hHM-EDA follows a step-wise optimisation strategy of its two models. The optimal binary solution sub-component is discovered after $47,000$ FES and the optimisation of the continuous component was finished after $170,000$ additional FES. The evolution of the mean generational best solutions of the binary and the real solution sub-components are presented in Figure 7.6. Results are averaged from the 25 runs of hHM-EDA. The colour in Figure 7.6a reflects the average bit status of each of the 100 bits at a specific generation, where dark colours denote a status of 0, and white colours a status of 1. The domino convergence due to the different salience of the bits is clearly visible in the figure. Bits corresponding to larger fitness penalties converge earlier during the evolutionary process.

Simultaneously the continuous search space is explored, *cf.* Figure 7.6b. If the binary sub-component was successfully optimised, only the first $\frac{N}{2} = 50$ real-valued elements $r_i \in R$ are considered for further optimisation. The last 50 variables are subject to genetic drift and converge randomly.

Computational cost

The tested methods are also compared according to their computational cost. The binary-only algorithms are generally fast, since the computational overhead for managing the simple probabilistic model is low. For vQEA, a multi-model has to be maintained and updated which slightly increases the computational requirements com-

Method	Time in sec	Relative to hHM-EDA
hHM-EDA	8.5 (0.0)	1.0
cHM-EDA	12.0 (0.0)	1.4
vQEA	38.6 (0.9)	4.5
PBIL	18.4 (0.1)	2.2
cGA	26.5 (0.1)	3.1
UMDA	19.8 (0.0)	2.3
CMA-ES	157.5 (5.1)	18.5
MBOA	2740.3 (12.0)	322.6

Table 7.3: Execution time of the tested methods when applied on the heterogeneous benchmark problem of size $N = 100$. In brackets the standard deviation is given. The third column presents the required time in relation to the execution time of hHM-EDA. For example, CMA-ES required ≈ 18.5 more time than hHM-EDA.

pared to PBIL, UMDA and cGA. Also, cHM-EDA and hHM-EDA are fast, since their algorithmic structure and the employed models are very similar to vQEA.

The more costly methods are clearly CMA-ES and MBOA. In terms of CMA-ES, a covariance matrix is generated based on the population of the current generation. Also, the sampling of new solutions according to this covariance matrix adds complexity to the algorithm. MBOA is the most costly among the tested methods here. As discussed earlier, its computational overhead is large and it requires significantly more resources than any of the other methods.

In order to demonstrate the computational cost of all the methods, the execution time for each of them is recorded. It is explicitly noted that the execution time is not a very reliable metric to compare algorithms to each other since it has a number of problems. The results depend not only on the used hardware, but also on the used programming language, the programmer's capabilities to optimise the code and the included software libraries. For example, MBOA is based on a C++ implementation, while all other methods are implemented in Java. Nevertheless, such a comparison can be very informative, if the limitations are known and discussed properly.

All methods apply the same configurations as used in the benchmark analysis. Only the stopping criterion was slightly modified: the algorithms perform the maximum number of FES and are not allowed to stop earlier, even if the success criterion is reached. Thus, all methods evaluate the fitness function $N \times 4 \times 10^3 = 4 \times 10^5$ times. The execution time was averaged over five runs. All experiments are performed on the same machine, which is an Intel Core2 Duo CPU, 3.00GHz, 4GB RAM, running a 64Bit Ubuntu Linux. The C++ code of MBOA was compiled using GCC 4.3.3 and the highest optimisation level.

Table 7.3 presents the measured CPU time for each method required to finish a single run. As expected, all binary methods are approximately equal in their computational demands, vQEA being slightly slower due to the additional probabilistic models. Also cHM-EDA and hHM-EDA report a fast execution time. The very good results of hHM-EDA are attributed to the conditional model update. Only if the sampled solution is worse than the current attractor does an update occur. Since the algorithm converges before the maximum number of FES is reached, no model update occurs in later stages of the run since the attractor and sampled solution are always identical. This situation results in an impressive execution time. If hHM-EDA is configured with a slower learning rate θ_σ in order to prevent the early convergence of the method, the execution time of the algorithm is close to the one for cHM-EDA.

CMA-ES and MBOA require on average ≈ 157 and ≈ 2740 seconds, respectively, to finish the run. Compared to hHM-EDA, these methods are approximately 18 and 322 times slower than hHM-EDA.

7.3.4 *Conclusion*

In this section, the performance of hHM-EDA was experimentally compared to a number of binary and continuous optimisers, along with the heterogeneous method MBOA. Due to the lack of a suitable benchmark suite for mixed problems, a simple test function was proposed that has similarities with the wrapper-based feature selection technique. The complexity of all the methods was discussed and compared in the light of their execution time.

Considering the obtained results on the proposed benchmark, hHM-EDA is clearly a highly competitive algorithm among the presented methods. However, a more detailed analysis on a wider range of test functions will have to be performed to provide further statistical evidence for this claim. Nevertheless, the obtained results have demonstrated a promising proof of concept. The hHM-EDA is a light-weight, fast and reliable optimiser with a negligible computational overhead. Practical guidelines have been presented that allow an easy and intuitive configuration of the method.

7.4 SEPARATION FROM COOPERATIVE CO-EVOLUTION

The simultaneous evolution of two solution parts in hHM-EDA seems very similar to the principle employed in a cooperative co-evolutionary algorithm. Nevertheless, there is a distinct difference between the two, which is discussed in this section.

Although a variety of co-evolutionary methods exist, a generalised cooperative co-evolutionary architecture (CCA) has been introduced only recently in (Potter & Jong, 2000). The co-evolutionary model defines different species, each of them evolving specific parts of a solution, also referred to as sub-components. A complete solution for the problem is formed by combining all the sub-components together. A single fitness criterion evaluates the quality of the complete solution. The evolution of each species proceeds more or less independently from other species. Thus, different representations for each sub-component and even different evolutionary algorithms can work together in this approach.

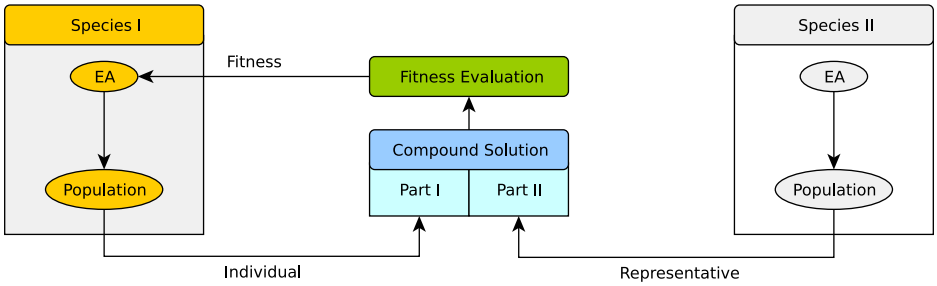
In principle, the two probabilistic models of hHM-EDA could be interpreted as two distinctive species. Each of them represents a separate sub-component of a compound solution and both employ entirely different update operations to drive their probabilistic model. Despite their independent evolution, both representations share a single fitness function and both parts need to collaborate in order to maximise their fitness.

The difference between hHM-EDA and CCA becomes obvious when comparing the actual process of the fitness evaluation of the two. The generalised CCA according to (Potter & Jong, 2000) is shown in Figure 7.7. Here two species are co-evolved and the two diagrams show the fitness evaluation from the perspective of either species I or II. In order to evaluate the individuals of species I, a representative of species II is chosen. This representative is then combined with all individuals of species I and the fitness evaluation occurs. In the opposite fashion, the individuals of the second species are evaluated.

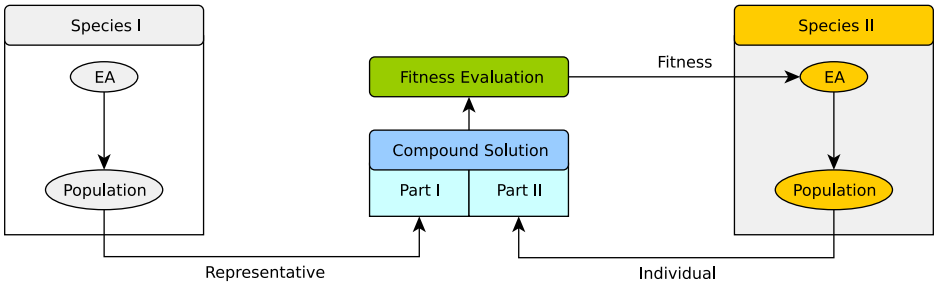
In the case of hHM-EDA, the situation is completely different. Here each individual consists of two parts and thus represents already a complete solution. In contrast to CCA, no representative is chosen from the other models and thus the metaphor of two species is not suitable for hHM-EDA. Figure 7.8 shows the evaluation process of an individual in hHM-EDA.

7.5 CONCLUSION

In this chapter, the two probabilistic models used in vQEA and cHM-EDA were combined forming a new original algorithm that was introduced as the heterogeneous hierarchical model EDA. Due to the lack of proper benchmark problems, a proof of concept in the form of a synthetic test problem was demonstrated. The benchmark shares similarities with a typical wrapper-based feature selection scenario.



(a) Evaluating species I



(b) Evaluating species II

Figure 7.7: The fitness evaluation process employed in the cooperative co-evolutionary architecture from the perspective of species I (top figure) and species II (bottom figure) respectively.

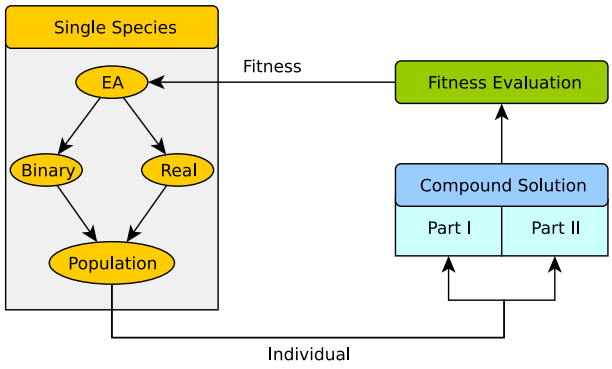


Figure 7.8: The fitness evaluation process employed in the hHM-EDA. All individuals belong to a single species, but each consists of two different sub-components (binary and real). Both sub-components together form a complete solution that is then evaluated by the fitness criterion.

Experimental analysis of eight different optimisation techniques was provided and discussed. In comparison to binary-only and continuous-only optimisation algorithms, hHM-EDA is highly competitive. Even the much more complex continuous-discrete optimiser MBOA required slightly more FES than hHM-EDA to solve the

benchmark reliably. However, the analysis of more test functions is required to provide strong statistical evidence to this claim. Differences and similarities between the MBOA and hHM-EDA were highlighted and discussed.

In terms of complexity, hHM-EDA requires very little algorithmic overhead, especially in comparison to MBOA and CMA-ES. Overall, hHM-EDA is a light-weight, fast and reliable optimisation method that requires the configuration of only very few parameters.

As part of the important integration of hHM-EDA into the current research field on evolutionary computation, the similarities and differences to the generalised cooperative co-evolutionary architecture were discussed. The co-evolving species in CCA are sequentially evaluated by choosing representatives from other species which is in contrast to the fitness evaluation in hHM-EDA. Thus, the interpretation of the two probabilistic models as separate species (in the sense of CCA) is not suitable for the hHM-EDA. Future research might elaborate further on the difference between the approaches and conduct detailed experimental comparisons of their characteristics.

Chapter 8

INTEGRATED FEATURE AND PARAMETER OPTIMISATION FOR AN EVOLVING SPIKING NEURAL NETWORK

This chapter presents the proposed extension of eSNN toward the feature subset selection (FSS) domain. All required methods for this extension, *i.e.* the binary and continuous optimisation algorithms and their hybridisation, were developed and comprehensively tested in the prior chapters. It was shown that both the binary and the continuous optimiser as well as the hybrid version are highly competitive and represent current state-of-the-art in the field of optimisation. The combination of hHM-EDA and eSNN forms an integrated feature and parameter optimisation framework based on the eSNN classification method. Due to the implementation of feature selection, the extension is expected to improve classification accuracy, while the simultaneous optimisation of the eSNN configuration avoids poor parameter choices and promotes the straightforward application of the method to a specific problem domain.

As described earlier, the continuous representation in hHM-EDA is used to optimise the parameter space of eSNN, while the binary representation explores the feature space of the given data set. A bit state of "0" or "1" indicates the absence or presence of the corresponding feature. According to the quantum metaphor of the binary part of hHM-EDA inherited from vQEA, the feature space is explored probabilistically using a *superposition of feature subsets*. Due to this interpretation, the novel eSNN based feature selection framework is named *Quantum-inspired Spiking Neural Network* (QiSNN) framework.

In order to test the functioning of the novel QiSNN framework, the method is experimentally compared to two traditional FSS algorithms. The first is the classical multi-layer perceptron (MLP), and the second is the Naïve Bayesian Classifier (NBC). These methods are used in a wrapper-based fashion similar to the proposed QiSNN framework. vQEA is employed as the selection algorithm, while either MLP or NBC are used to evaluate a given feature subset.

Additionally, we analyse the QiSNN classification performance by exchanging the hHM-EDA for vQEA. This allows a direct comparison of the binary and heterogeneous optimisation performance in the eSNN classification context. The binary-only nature of vQEA requires the conversion of bit strings into real values involving a number of general issues, such as the introduction of granularity into a continuous search space, but also additional computational overhead and encoding issues. Thus, the hHM-EDA is expected to be beneficial in such a scenario.

Altogether, four methods are experimentally compared in this chapter: i) the proposed QiSNN using hHM-EDA specialised on the exploration of heterogeneous search spaces; ii) QiSNN using the the binary-only optimisation algorithm vQEA; iii) a wrapper approach using MLP as the classifier and vQEA for feature selection; and iv) a wrapper approach using NBC as the classifier and vQEA for feature selection.

The analysis of QiSNN is undertaken using synthetic data sets. Such an approach has several advantages. First, the global optimum is known *a priori* and the functioning of the algorithm can be easily validated. Second, the characteristics of the data set are known and all parameters, such as noise and redundancy of features, can be fully controlled by the experimenter. Finally, benchmarks commonly allow inter-comparisons between methods developed in other studies.

The following sections introduce the novel QiSNN framework along with its components. Then, QiSNN is experimentally investigated and compared to the above mentioned algorithms in terms of classification and feature selection performance and computational cost. Finally, we discuss the quality of obtained results followed by the conclusion of this chapter.

8.1 QUANTUM-INSPIRED SPIKING NEURAL NETWORK FRAMEWORK

The proposed QiSNN framework follows the wrapper approach introduced in (Kohavi & Sommerfield, 1995). Kohavi and John (1997) discussed the method in detail. The wrapper methodology is a type of “black box” approach. In its core, it contains a general optimisation algorithm interacting with an induction or classification method. The optimisation task consists in a reliable identification of an optimal feature subset that maximises the classification accuracy determined by the inductor. Thus, the classification method provides a quality measure for a presented feature subset and hence, acts as the fitness function for a general evolutionary algorithm.

Due to the black box character of the classification method and the interacting optimisation algorithm, the wrapper methodology offers a simple, but powerful, feature

selection technique that has become popular in many research areas and domains. See for example, the review on feature selection algorithms in bioinformatics (Saeys, Inza, & Larrañaga, 2007), and the medical case study on the survival of cirrhotic patients presented in (Inza, Merino, et al., 2001) and (Blanco, Inza, Merino, Quiroga, & Larrañaga, 2005). See also (Álvarez et al., 2006), where the wrapper technique for selecting feature subsets for a emotion recognition system based on spoken language was used.

The wrapper approach in the context of EDAs is particularly popular. The study presented in (Inza, Larrañaga, & Sierra, 2001) and the excellent textbook by (Larrañaga & Lozano, 2002) on the matter are worth mentioning here.

The QiSNN framework employs hHM-EDA as the feature selecting optimisation algorithm, while the eSNN classification method represents the inductor. Since hHM-EDA belongs to the class of EDA itself, the QiSNN approach is related to the EDA studies mentioned above. An alternative also investigated in this chapter is the use of vQEA instead of hHM-EDA as the optimiser.

8.1.1 *Integrated feature and parameter optimisation*

QiSNN integrates the feature and parameter optimisation into a single framework. Section 2.7 of chapter 2 has reviewed similar approaches in this context. Valko et al. (2005) identified the fitness function as a crucial step for the successful application of such an integrated approach. It was argued that in the early phase of the optimisation, the parameters are selected randomly. As a result it is very likely that a setting is selected for which the classifier is unable to respond to any input presented. For such settings the fitness value is zero which results in flat areas in the fitness landscape. Hence, a configuration that will allow the network to fire (even if not always correctly) represents a local attractor in the search space that could be difficult to escape in later iterations of the search. (Valko et al., 2005) used a linear combination of several sub-criteria to avoid this problem. Nevertheless, we cannot confirm that the use of much simpler fitness functions leads to any problems in our experiments. Using the classification accuracy on testing samples seemed to work well as it is presented in this thesis and in earlier papers. All parameters of eSNN, namely modulation factor m_l, similarity threshold s_l, potential fraction c_l, $\forall l \in L$, were included in the search space of the optimisation method.

In the context of vQEA, a conversion of bit strings into real values is required. A similar scenario was presented in the previous chapter. This study uses a small

number of Gray-coded bits to approximate parameter configurations of the eSNN method.

8.1.2 *Description of QiSNN*

The proposed QiSNN framework is shown in Figure 8.1. The upper part of the diagram represents the eSNN classification method as comprehensively explained in chapter 2. Note the added binary mask in the second step of the process. This mask, along with a specific configuration of neural and learning parameters, is passed to eSNN from the optimisation method depicted in the lower part of the figure.

The binary mask describes the features to be selected from a real-valued input data vector. Then, the selected features are transformed into a train of spikes using the rank order population encoding technique (see chapter 2 for details). Following the one-pass learning procedure, the connection weights of eSNN are trained according to the given parameter set.

The learning process includes the presentation of all training samples. After the learning, the classification accuracy is determined on a set of test samples. This accuracy provides a quality measure of the feature subset and the used parameter configuration. This quality feedback is passed to the employed optimisation algorithm, *i.e.* either hHM-EDA or vQEA. Based on the quality, the optimiser adapts the search strategy and passes new feature subsets and configurations to eSNN for evaluation. The whole process iterates until a termination criterion is met, *i.e.* a predefined classification accuracy is reached or the maximum number of iterations is exhausted.

8.2 DATA

QiSNN is investigated on the basis of two benchmarks, namely the two-spiral problem and the hypercube data set. A description of the generation and characteristics of the data is presented here.

8.2.1 *Two spirals*

The first benchmark is known as the two-spiral-problem. This problem is composed of two-dimensional data forming two intertwined spirals and was first introduced in (Lang & Witbrock, 1988). It requires the learning of a highly non-linear separation of the input space. The data was frequently used as a benchmark for neural networks

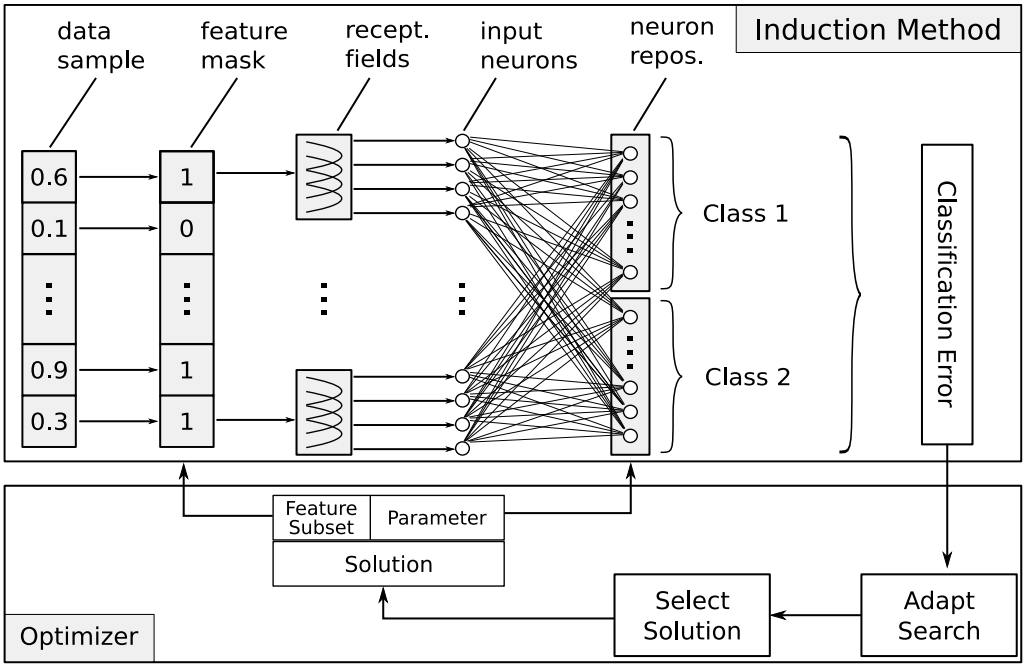


Figure 8.1: The QiSNN framework of tightly coupled feature selection and parameter optimisation of eSNN, integrated with the data. As a first step a feature subset is selected from a real-valued data sample using a bit string acting as a feature mask, where a "1"/"0" in this mask indicates selected/non-selected features of the data vector. Selected vector elements are then mapped into the time domain using a number of Gaussian receptive fields. Based on this transformation input neurons of a eSNN emit spikes at pre-defined firing times, invoking the one-pass learning algorithm of the eSNN. The learning iteratively creates repositories of output neurons, one repository for each class. Here a two-class problem is presented. Based on a set of training samples the eSNN is trained and its quality is determined based on the classification accuracy on a set of testing samples. The classification accuracy is then used as the fitness criterion of the optimisation method. Based on the fitness the search strategy is adapted and a new solution is proposed. The solution includes two parts: A binary feature mask and a set of real-valued parameters for eSNN. The whole process iterates until a termination criterion is met, *i.e.* a pre-defined classification accuracy is reached or the maximum number of iterations is exhausted.

including the analysis of the eSNN method itself (Wysoski, 2008). Since the data contains only two relevant dimensions, we have extended it by adding redundant and random information. The importance of the redundant features is varied: features range from mere copies of the original two spirals to completely random ones. The available information in a feature decreases when stronger noise is applied. The design of the data set is particularly important since it is expected that the eSNN is capable of rejecting features according to their inherent information, *i.e.* the less information a feature carries, the earlier ESNN should be able to exclude the feature

during the evolutionary process. We briefly summarise the data generation below. The situation is similar to the salience of different search variables. See chapter 7 for a comprehensive discussion of this phenomenon,

Data points belonging to two intertwined Archimedean spirals (also known as the arithmetic spiral) were generated and labelled accordingly. The irrelevant dimensions consist of random values chosen from a uniform distribution, covering the entire input space in the range $[-1, 1]$ of the data set. The redundant dimensions are represented by copies of the original spiral points $p = (x, y)^T$, which were perturbed by a Gaussian noise using standard deviation $\sigma = s|p|$, with $|p|$ being the absolute value of vector p and s – a parameter controlling the noise strength. The noise increases linearly for points that are more distant from the spiral origin $(0, 0)^T$. A noisy value p'_i is then defined as the outcome of the p_i-centred Gaussian distributed random variable $\mathcal{N}(p_i, \sigma^2)$, using σ as defined above.

Our final data set contained seven redundant two-dimensional spiral points $(x'_i, y'_i)^T$ and for each a different noise strength parameter $s \in \{0.2, 0.3, \ldots, 0.8\}$ was used, totalling 14 redundant features. Four additional random features $r_1, \ldots, r_4$ were included. Together with the two relevant features of the spirals (x and y), the data set contained 20 features. Figure 8.2 presents the 400 generated samples of the resulting data set for the seven values of s and the original and fully random versions.

8.2.2 *Hypercube*

The second benchmark is the uniform hypercube data set, to our best knowledge first introduced in (Estevez, Tesmer, Perez, & Zurada, 2009). The problem consists of two classes of 400 samples. For each sample, a five-dimensional vector $(r_1, \ldots, r_5)$ is drawn from a uniform distribution. A given pattern belongs to class 1 if $r_i < \alpha\gamma^{i-1}$ for $i = 1, \ldots 5$ and to class 2 otherwise. The parameters were chosen to be $\gamma = 0.8$ and $\alpha = 0.5$.

This data set was created with different ratios of relevant, redundant and random features in order to cover some typical scenarios in the context of feature selection problems. More relevant features (five) were included than used for the spiral data (two). The number of redundant and random features was chosen to be five and 30 respectively, compared to 14 and seven in the spiral data set. In total, 400 samples with 40 features each were used in this data set. The redundant features are linear combinations of the relevant features perturbed by additive Gaussian noise of increasing

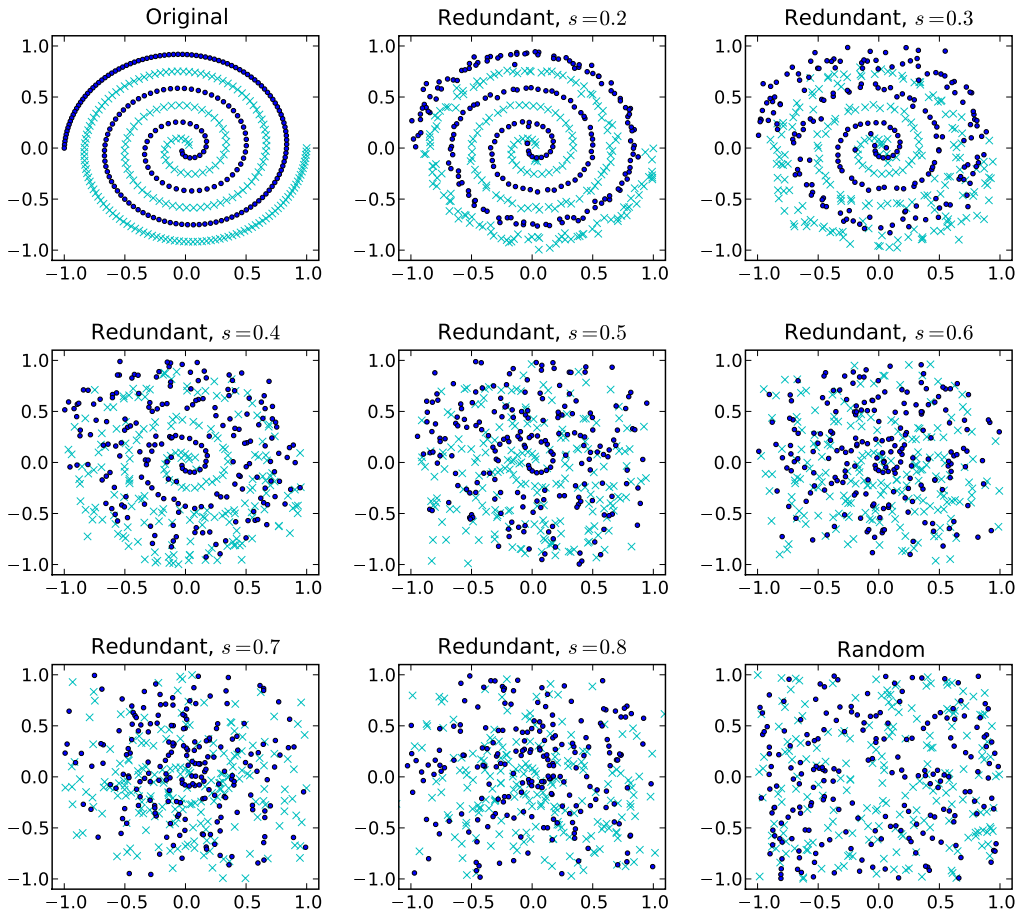


Figure 8.2: The different features of the generated synthetic two-spiral data set for investigating eSNN in the context of a FSS problem. The colours/symbols represent the class label of a given data point. Each figure shows two features (x- and y-axis). All features are combined to form the complete experimental data set. The quality of the redundant features is decreasing as stronger noise is applied. Additionally, four random features are included in the data set (only two of them are shown in the bottom right diagram).

level. The data set was balanced. Estevez et al. (2009), provide a detailed explanation of the data generation.

8.3 PERFORMANCE ANALYSIS

This section investigates the classification and feature selection performance of the proposed QiSNN. Four methods are included in the analysis: the proposed QiSNN using either hHM-EDA or vQEA for the feature and parameter optimisation problem, a wrapper approach using MLP as the classifier and vQEA for feature selection, and a wrapper approach using NBC as the classifier and vQEA for feature selection. First, the experimental setup is described, followed by the presentation and discussion of the results.

8.3.1 *Setup*

Both optimisation algorithms, *i.e.* hHM-EDA and vQEA, use a population structure of ten individuals organised in a single group that is globally synchronised every generation. This setting was reported to be generally suitable for a number of benchmark problems.

In the case of the spiral data set, the learning rate for the binary rotation gate was set to $\theta = \pi/50$. Due to the advantages discovered in chapter 7, the H_ϵ gate in its default setting, *i.e.* $\epsilon = \sin^2(0.02\pi)$, was used for hHM-EDA and vQEA. In terms of hHM-EDA, the rate of the mean and standard deviation shift were chosen to be $\theta_\mu = 0.1$ and $\theta_\sigma = 0.025$ respectively. We note that these learning rates are slightly faster than the default settings of the methods. This is due to the fact that QiSNN requires only six parameters for a two-class problem, and as a consequence, faster learning rates are possible.

A total of 400 generations were performed. Due to the larger problem size, 500 generations were computed for the hypercube problem, using $\theta = \pi/50$ for the binary learning rate and $\theta_\mu = 0.1$ and $\theta_\sigma = 0.05$ for the continuous update operators.

vQEA requires the conversion of bit strings into real values. Four bits per variable offer sufficient flexibility for the parameter space. For the conversion itself a Gray code was used.

A fair comparison between methods requires the appropriate configuration of each classifier used in this study. NBC has no parameters and so no parameter tuning was necessary for this method.

The MLP, on the other hand, involves numerous tunable variables, the most critical ones being the number of hidden neurons, the learning rate, and the momentum term. These parameters were varied in order to determine the best combination. Al-

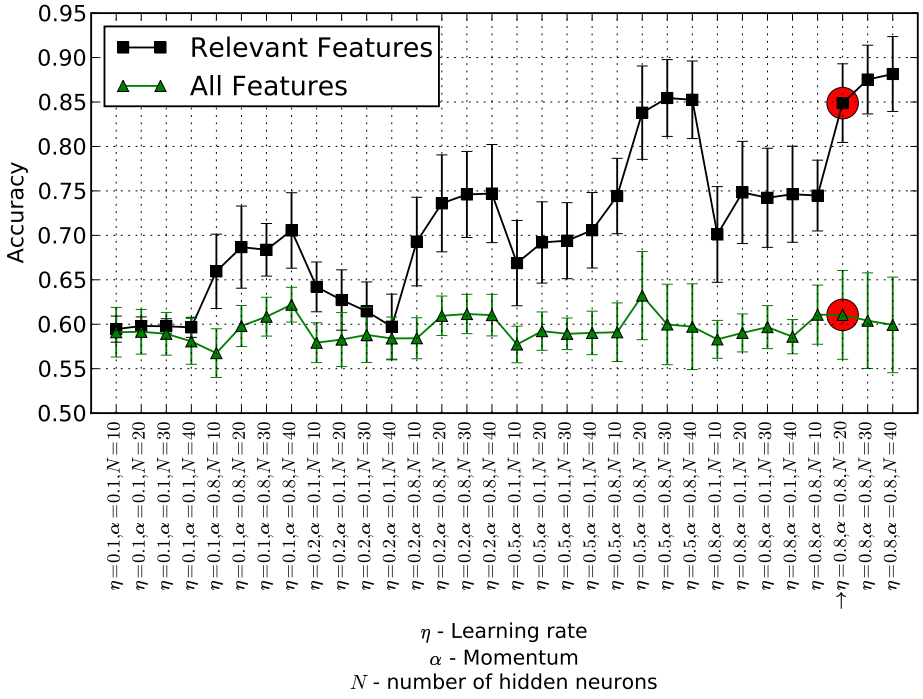


Figure 8.3: The figure shows the accuracy levels achieved by 32 different configurations of a multi-layer perceptron on the two-spiral data set. Each point represents the average of the accuracies obtained in a 10-fold cross-validation experiment. Error bars indicate the standard deviation. All configurations use neurons with sigmoid transfer functions trained in 500 epochs. The lower curve (green triangles) represents the accuracy of the MLP when all 20 features are included in the data set, the upper curve (black squares) the accuracy when only the relevant features are used. The circles (red) indicate the final configuration chosen for the experiments performed in this study. They yield a satisfying compromise between computational cost and classification quality.

together 32 different settings were tested. For each setting, two separate 10-fold cross-validation runs were performed. For the first run, a subset of the data including the relevant features only, was used, while the for the second run all features were involved.

The results on the spiral data set for these two runs and for each of the 32 parameter configurations are presented in Figure 8.3. We note the clear benefit of feature selection for this data set. Only if the MLP is trained on relevant information, does the classification accuracy increase significantly. Thus, appropriate feature selection improves the performance of MLP, which is the key principle exploited in the wrapper approach.

Parameter Name	Value
no. hidden neurons	$N = 20$
activation function	sigmoid
learning rate	$\eta = 0.8$
momentum term	$\alpha = 0.8$
training epochs	500

Table 8.1: Appropriate Parameter configuration for the MLP model, used for comparison with the proposed new method

The chosen setting for the experiments described below is based on a trade-off between computational cost and classification accuracy. The severe additional cost of more hidden neurons is not worth the slight increase of accuracy reported in Figure 8.3. Table 8.1 presents the parameter settings obtained from the parameter study. Using 10-fold cross-validation, the chosen configuration of MLP achieved a satisfying accuracy of 0.849 (standard deviation 0.0634) on the spiral data set containing the two relevant features only. When applied to the full data set using all 20 features, the same configuration resulted in an accuracy of 0.611 (0.0608).

Finding an appropriate setting for the spiral problem appeared to be more difficult compared to the hypercube data set. For this data, changes in the configuration had only small impact on the performance of the classifier. Thus, the same parameter setting for both problems was used. The standard error back-propagation learning algorithm (Rumelhart et al., 1986) was employed to train the connection weights of the network. The weights were initialised to small values in the range $[-0.25, 0.25]$ randomly chosen according to a uniform distribution.

Most of the parameters of QiSNN are optimised during the evolutionary process. For each class $l \in L$, the modulation factor m_l, the similarity threshold s_l, and the proportion factor c_l are optimised. Since both problems contain two classes, six parameters are involved in the QiSNN framework. In terms of the population encoding, the number of receptive fields needs especially careful consideration since it affects the resolution for distinguishing between different input variables. After some preliminary experiments, 20 receptive fields in case of the spiral data and five receptive fields for the hypercube were used. The Gaussian centres were uniformly distributed over the search interval and the variance was set to $\beta = 1.5$.

In order to guarantee statistical significance, 30 independent runs for each investigated classification method were performed. In every generation, all samples of the data set were randomly shuffled and divided into training and testing samples, ac-

cording to a train-to-test ratio of 75%. For the computation of the classification error, we determined the ratio between correctly classified samples and the total number of testing samples.

8.3.2 *Results*

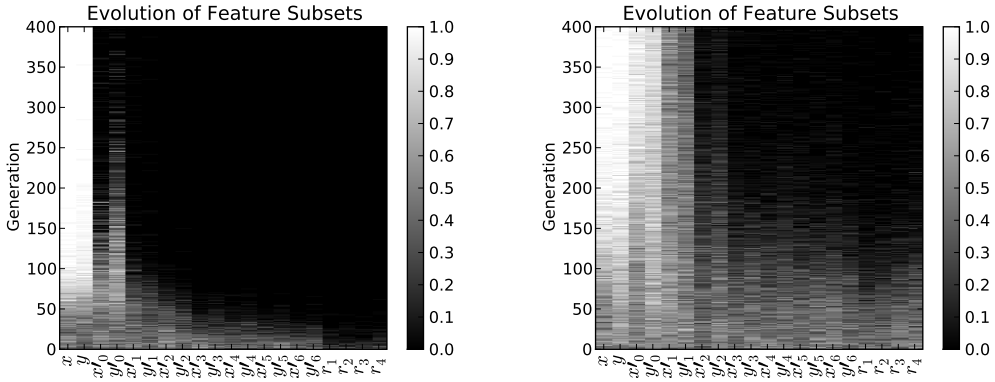
We discuss the results on the two-spiral problem first, followed by the results on the hypercube data.

Spiral data

Figure 8.4 presents the evolution of the average best feature subset in every generation using the two versions of QiSNN, MLP and NBC respectively. The colour of a point in these diagrams reflects how often a specific feature was selected at a certain generation: the lighter the colour, the more often the corresponding feature was selected. It can clearly be seen that, independent of the algorithm used, a large number of features have been discarded during the evolutionary process. Furthermore, all algorithms clearly identify the features x and y to be relevant. All methods except the proposed QiSNN using hHM-EDA select some redundant and/or irrelevant features as well.

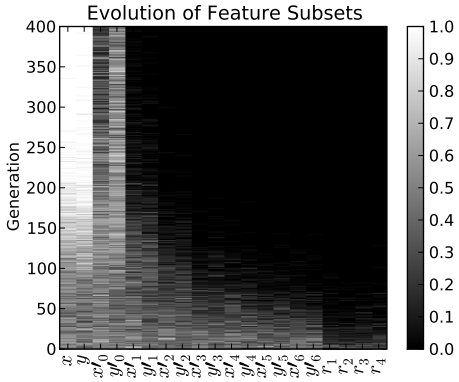
Particularly interesting is the order in which the features have been removed by each algorithm. Both versions of QiSNN (Figure 8.4a and 8.4c) rejected the four random features $r_1, \ldots, r_4$ containing no information almost immediately (in fewer than 20 generations). The redundant features x'_i, y'_i were then rejected one after the other, according to the strength of the inherent noise: the higher the noise, the earlier a feature is identified as irrelevant. We note the excellent performance of QiSNN using hHM-EDA which is clearly able to reject all redundant features in most of the runs. Figure 8.5 compares the evolution of the number of selected features during each generation. While both QiSNN based methods clearly select fewer features than their classical competitors at any stage of the optimisation, the binary optimised QiSNN is outperformed by the heterogeneous version.

It is also interesting to compare the evolution of the classification error for each algorithm, *cf.* Figure 8.6. The gradient in the fitness landscape defined by eSNN appears to be much steeper compared to other algorithms, ranging from completely unfit solutions at the beginning of the evolutionary run toward high quality solutions in later generations. MLP and NBC display a flatter fitness evolution. It is noted that the eSNN starts with no optimisation of its parameters, while MLP and NBC

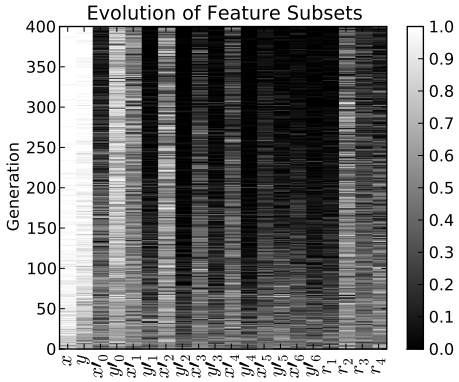


(a) QiSNN using hHM-EDA

(b) MLP



(c) QiSNN (binary) using vQEA



(d) NBC

Figure 8.4: Evolution of feature subsets on the spiral data set. The two relevant features were identified by all methods, indicated by the bright colour of the first two columns in the diagrams. Only the QiSNN using hHM-EDA is able to determine the optimal feature subset consistently.

are properly configured as part of the experimental setup. The fitness gradient may be partially responsible for eSNN turning into a very good quality measure for the feature subsets.

According to the presented results for QiSNN, a strong correlation between classification accuracy and number of features appears advantageous in the context of a feature selection task. Figure 8.7 presents this relationship for each of the investigated induction methods. Each point in the diagram corresponds to a tuple (accuracy, number of features) obtained from the generational best individual of every generation. The colour indicates the generation itself. The lighter the colour, the later the generation in which a given tuple was obtained. In the case of QiSNN (*cf.* Figure 8.7a), a strong relationship between number of features and accuracy can be observed. Even for small decreases of the number of features, significant accuracy

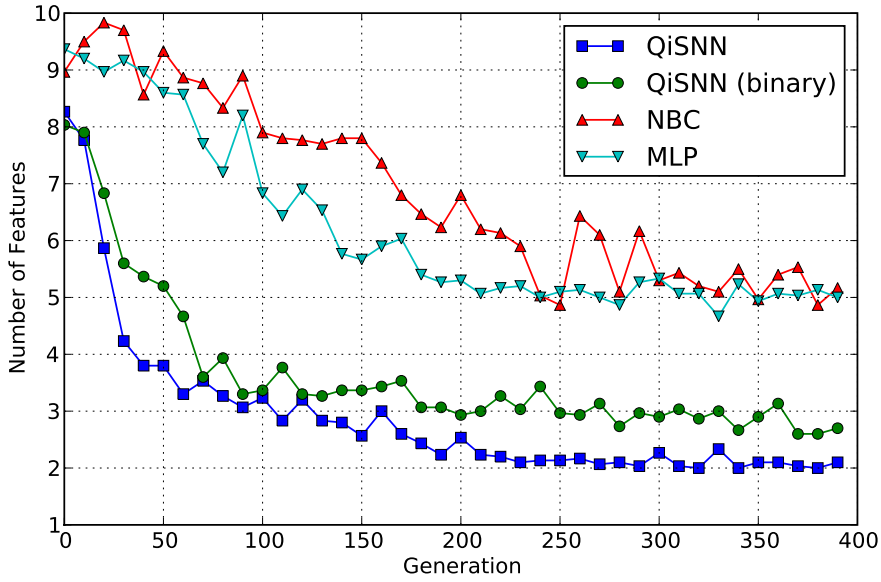


Figure 8.5: Evolution of number of features in the spiral data set. All methods are clearly capable of reducing the number of features. The two versions of QiSNN exclude more features than the classical methods.

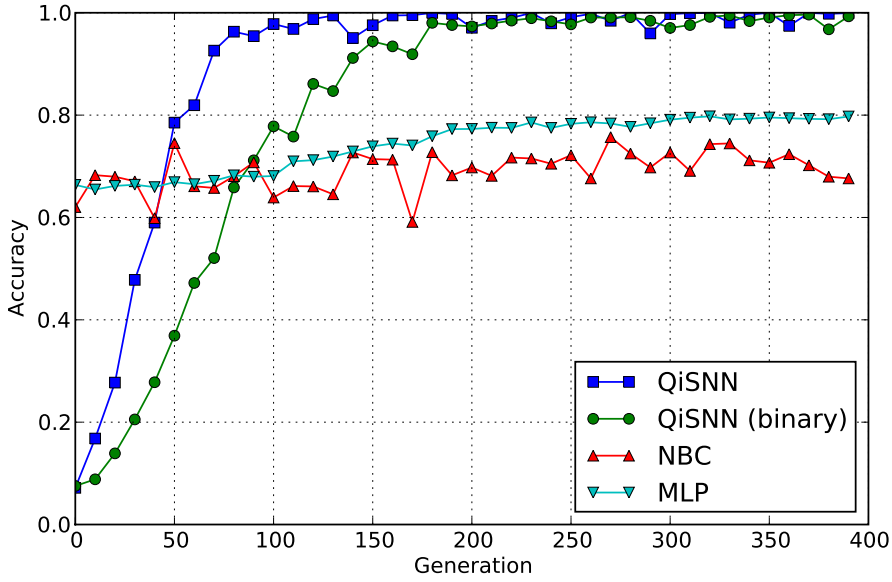
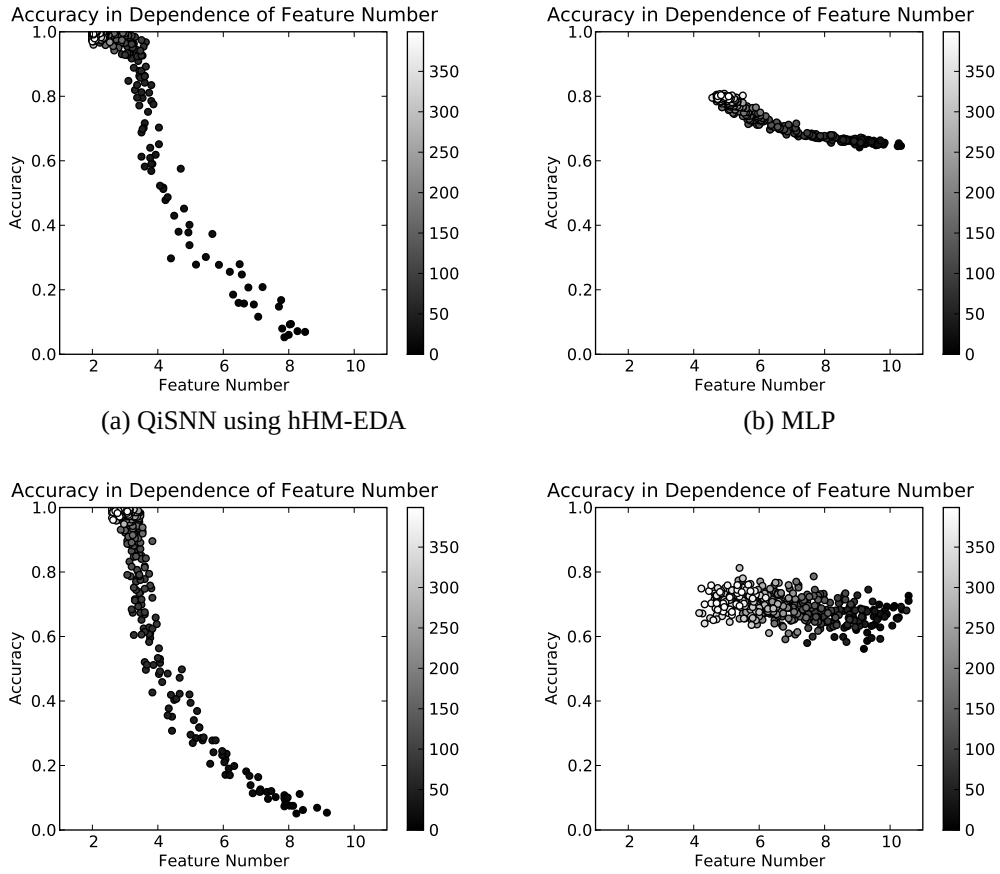


Figure 8.6: Evolution of the average accuracy of the generational best solution on the spiral data set. QiSNN reports excellent classification results, the proposed QiSNN using hHM-EDA being faster than all other tested algorithms.

improvements are reported. The strong correlation between number of features and classification accuracy introduces a gradient and partially reduces *neutrality* in the



(a) QiSNN using hHM-EDA

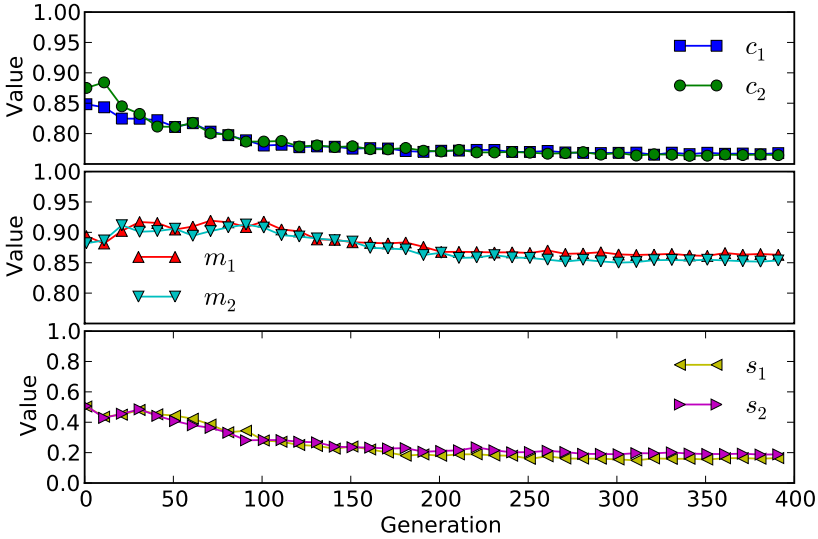
(b) MLP

(c) QiSNN (binary) using vQEA

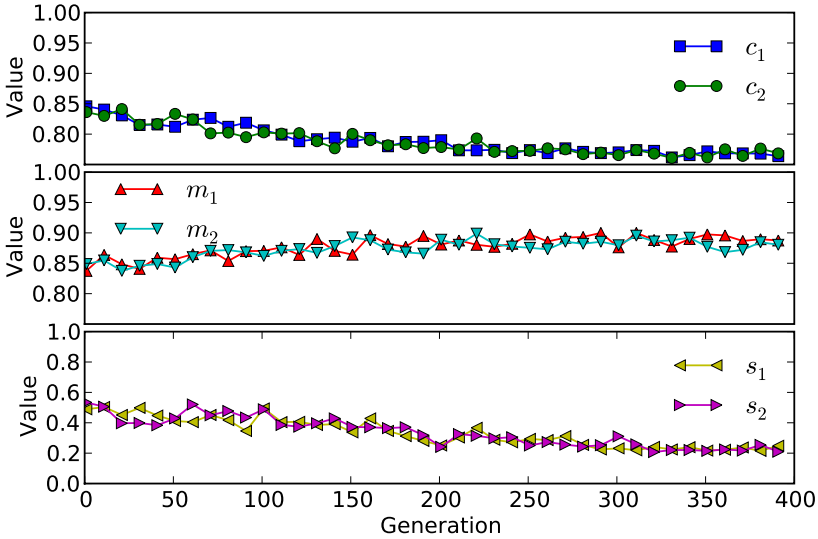
(d) NBC

Figure 8.7: The classification accuracy as a function of the number of features for all tested classifiers on the spiral data set. The different gray levels correspond to the generation in which a given data point was obtained. The lighter the colour, the later the generation. For the eSNN-based classifiers the accuracy is highly dependent on the number of features, which is in contrast to MLP and NBC.

fitness landscape. Removing a redundant or irrelevant feature from the selected subset corresponds to a fitness gain for QiSNN, which may not necessarily be true for the other two tested methods. If the feature removal does not lead to a certain fitness gain, and thus two solutions may have the same fitness value, the fitness landscape has a neutral dimension at the corresponding parameter. Due to genetic drift, the neutral parameter converges randomly, which means a random selection or non-selection of the encoded feature. In the fitness landscape defined by eSNN, neutral dimensions are replaced by a fitness gradient, which allows the identification and exclusion of low quality features from the current subset. As a result, the fitness landscape can be easily climbed by the optimisation algorithm, leading to faster and more consistent convergence towards the optimal feature subset.



(a) QiSNN using hHM-EDA



(b) QiSNN (binary) using vQEA for optimisation

Figure 8.8: Evolution of parameters in the QiSNN framework on the spiral data set. Three parameter pairs are optimised during the evolutionary process. Due to the continuous representation of the parameter space, a smoother exploration is possible (upper figure) compared to the binary optimisation (bottom figure).

Figure 8.8 presents the evolution of the eSNN parameters for the two versions of QiSNN. Although both methods have evolved similar final parameter configurations, the exploration using the continuous representation is much smoother compared to the binary one and allows a finer parameter tuning. Due to the balanced nature of the

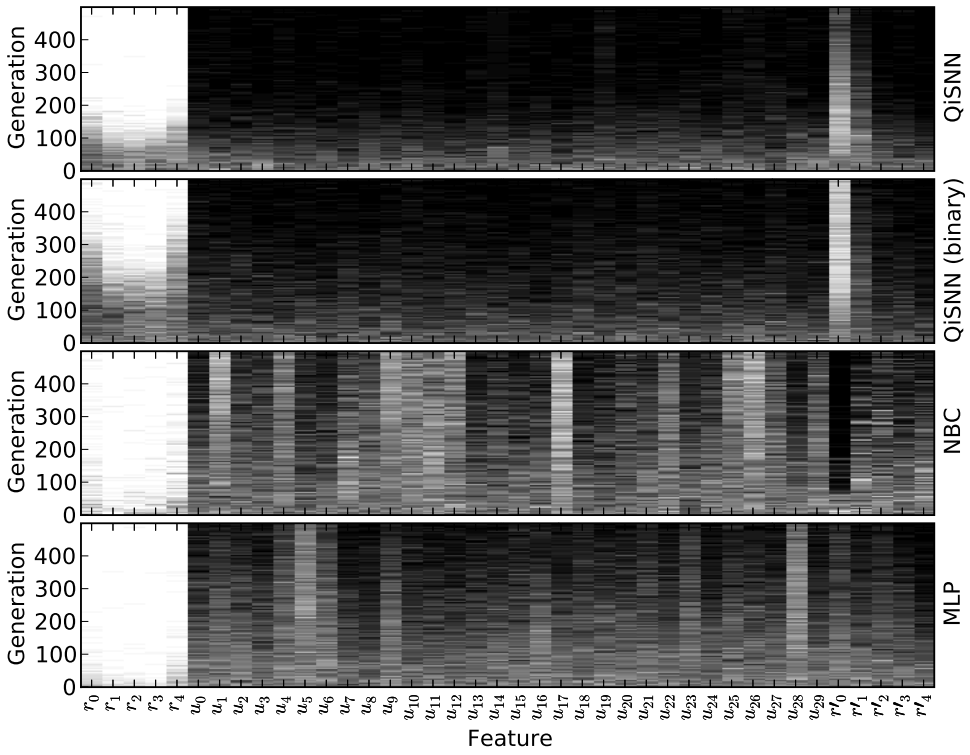


Figure 8.9: Evolution of feature subsets on the hypercube data set. Results on the synthetic hypercube data set averaged over 30 independent optimisation runs. The five relevant features were identified by all methods, indicated by the bright colour of the first columns in the diagram.

data set, the parameter setting for the two classes have evolved to be approximately identical, *i.e.* $c_1 \approx c_2$, $m_1 \approx m_2$ and $s_1 \approx s_2$.

Hypercube

A similar analysis was done for the second benchmark data set. We note that this data set was very easy to solve by any of the tested algorithms. Even without feature selection, MLP and NBC reported very high classification accuracy. Nevertheless, the results are presented here since they show the proper functioning of all tested methods on an additional independent benchmark problem. Figures 8.9-8.13 depict the results on the hypercube problem.

In Figure 8.9, the evolution of the average selected feature subset is shown. Similar to the figures presented on the spiral data above, different levels of greyness reflect how often a specific feature was selected at a certain generation. In these diagrams, the first five features correspond to the relevant features, followed by 30 irrelevant and finally by the five redundant features. All methods clearly identify the five relevant

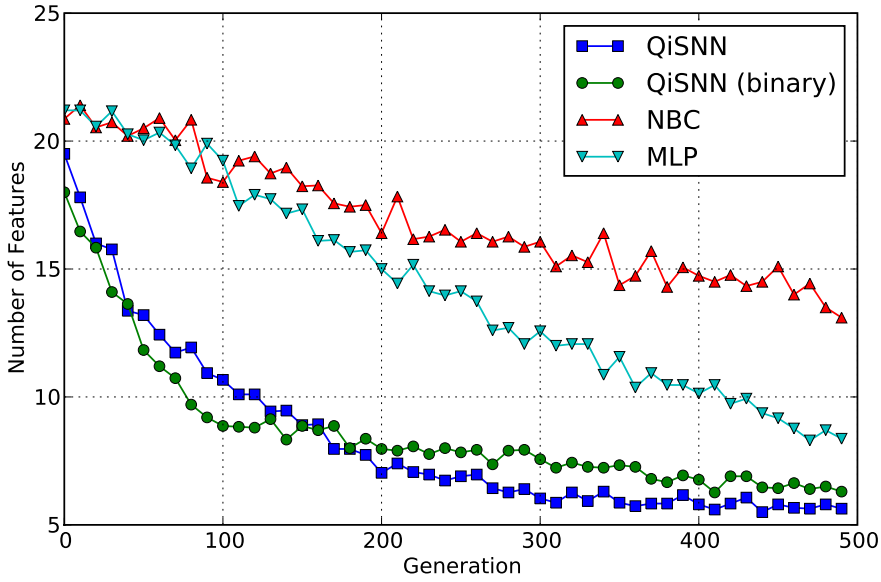


Figure 8.10: Evolution of the number of features in the hypercube data set. All methods are clearly capable to reduce the number of features. The two versions of QiSNN exclude significantly more features than the classical methods.

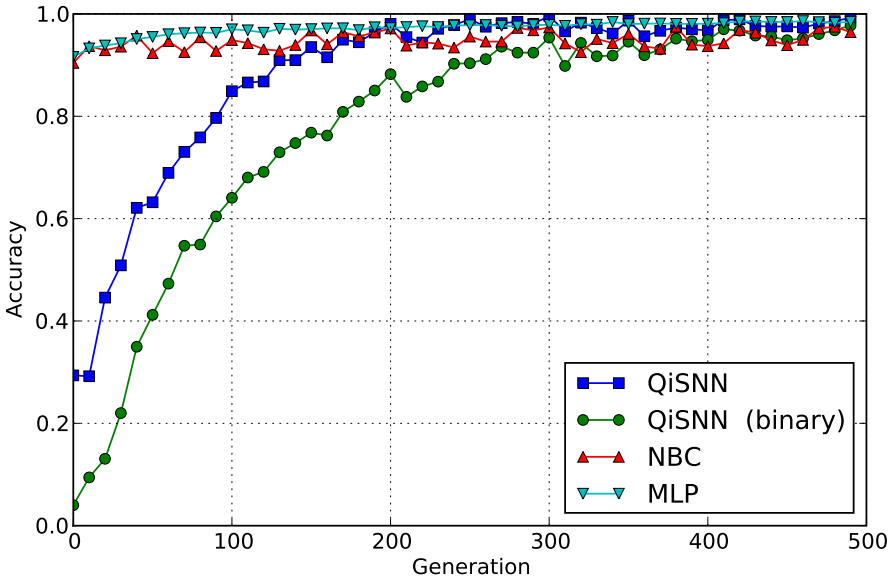
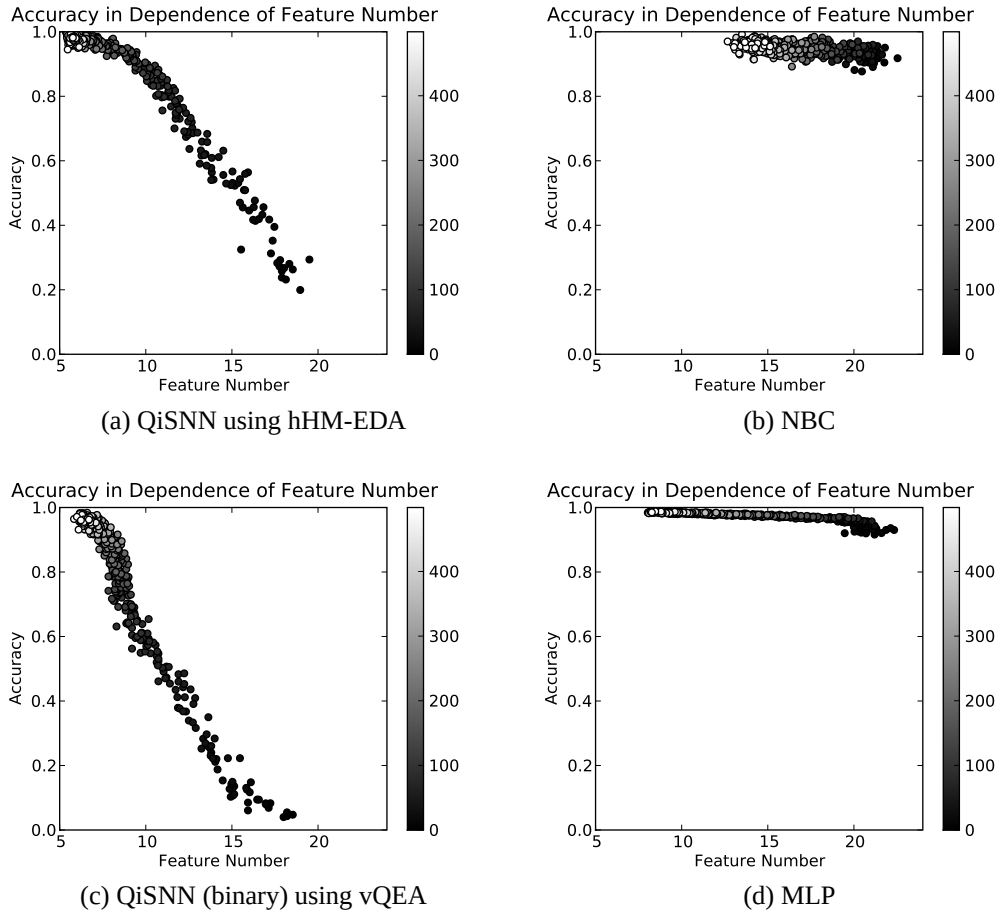


Figure 8.11: Evolution of the average accuracy of the generational best solution on the hypercube data set. All methods report excellent classification results.

variables. Nevertheless, all methods also select some irrelevant/redundant ones. In Figure 8.10, the evolution of the average number of selected features is presented. Both QiSNN methods were capable of decreasing the number of features faster than



(a) QiSNN using hHM-EDA

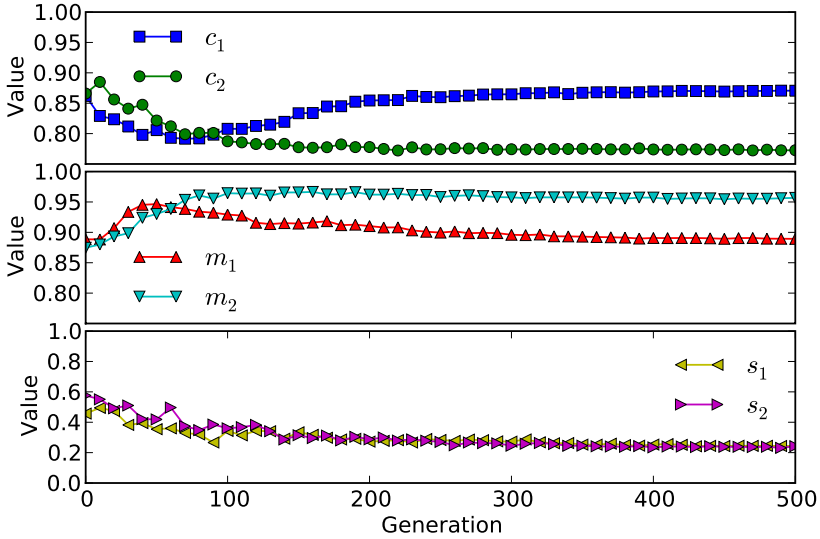
(b) NBC

(c) QiSNN (binary) using vQEA

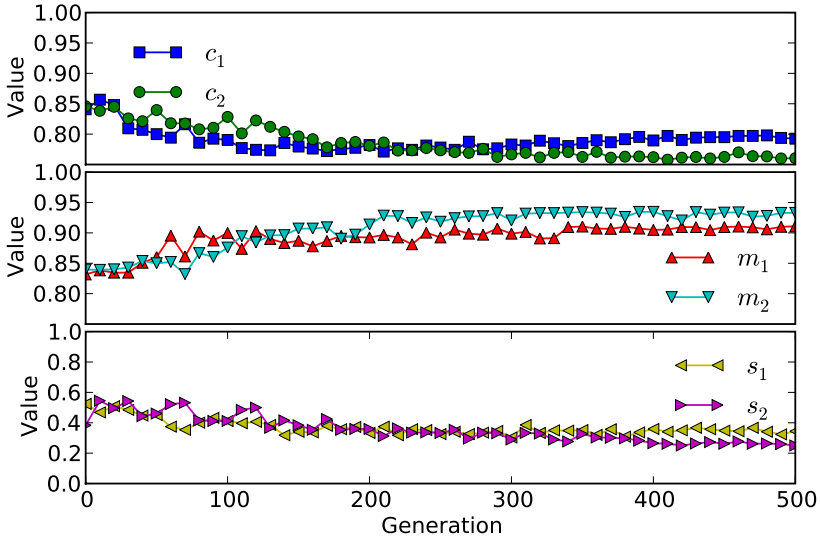
(d) MLP

Figure 8.12: The classification accuracy as a function of the number of features for all tested classifiers on the hypercube data set. The different gray levels correspond to the generation in which a given data point was obtained. The lighter the colour, the later the generation. For the eSNN-based classifiers the accuracy is highly dependent on the number of features, which is in contrast to MLP and NBC.

NBC and MLP, *cf.* Figure 8.10. As depicted in Figure 8.11, NBC and MLP report close to optimal classification accuracy without removing all irrelevant and redundant features. Without the presence of any selective pressure, some features converge randomly due to genetic drift, which has resulted in the selection of some irrelevant features. Since in QiSNN an appropriate parameter setting needs to be evolved during the run, its classification performance is worse than MLP and NBC at the early stage of the evolutionary process. In later generations, the accuracy increases quickly and reaches levels similar to those in the traditional algorithms. Once more, we note the faster convergence of the proposed QiSNN using the heterogeneous optimisation method compared to the binary optimised QiSNN.



(a) QiSNN using hHM-EDA



(b) QiSNN (binary) using vQEA

Figure 8.13: Evolution of parameters in the QiSNN framework on the hypercube data set. Three parameter pairs are optimised during the evolutionary process. Due to the continuous representation of the parameter space, a smoother exploration is possible (upper figure) compared to the binary optimisation (bottom figure).

The previously observed strong correlation between classification accuracy and number of features in QiSNN is also clearly demonstrated on the hypercube data. Figure 8.12 presents this relationship for each of the investigated induction methods. Finally, we want to comment on the parameter evolution obtained from the two QiSNN. Similarly to the case on the spiral data, the continuous parameter optimisa-

tion is much smoother, compared to the binary optimisation. Both optimisers report similar final parameter configurations for the similarity threshold and the modulation factor, only the proportion factor c_l shows a slightly different evolution between the methods.

Complexity

We also want to discuss the computational complexity for each of the algorithms presented here. The fitness evaluation of a feature subset is clearly the most costly part in the wrapper. Depending on the data set, an MLP requires the construction of a rather large neural network, followed by the training of each data sample for 500 epochs using a costly back-propagation procedure and is thus by far the most complex method in this study. The eSNN classifier implements a fast one-pass learning, but additional overhead is required for transforming each data sample into a spike sequence and computing the spike propagation in the network. Due to the simple topology of the network, an efficient network simulation is possible. The NBC requires the lowest computational resources, each training sample is investigated only once and only minimal overhead is necessary, allowing very fast classification.

8.4 PARAMETER EVOLUTION

The previous experiments revealed that the simultaneous optimisation of feature subsets and eSNN parameters is effectively achieved by hHM-EDA. In this section, we focus on the analysis of the parameter evolution in greater detail.

First of all, the interpretation of the eSNN parameters is highlighted. For each class label, three eSNN related variables exist, namely the modulation factor m, the firing threshold fraction c, and the similarity threshold s. See also chapter 2, section 2.6 for a comprehensive description of these variables.

The modulation factor m reflects how strongly a neuron is affected by the temporal order of spike arrival times. In the extreme case of $m = 0$, none of the pre-synaptic spikes contributes to the computation of the post-synaptic potential. For the other extreme, $m = 1$, all pre-synaptic spikes have the same importance and contribute to the computation of the post-synaptic potential equally. Between these two extremes, $0 < m < 1$, the temporal order of spike arrival times is important: the earlier a spike is received by a neuron after the stimulation onset, the stronger its contribution to the post-synaptic potential change.

The modulation also directly controls the maximum stimulation u_{max} of a neuron, since $u_{max} = \sum_j w_j(m)^{order(j)}$, where the sum runs over all pre-synaptic neurons, *cf.* Algorithm 1 on page 40. In fact, the term $\sum_x m^x$ corresponds to a *geometrical series* which converges according to:

$$\sum_{x=0}^{\infty} m^x = \frac{1}{1-m} \tag{8.1}$$

Since a connection weight always satisfies $w_j \leq 1$, the term $\frac{1}{1-m}$ in Equation 8.1 represents an upper bound for u_{max}.

The firing threshold ϑ is defined as a fraction c of the maximum post-synaptic potential, *i.e.* $\vartheta = cu_{max}$. Parameter $0 < c < 1$ describes the sensitivity of the neuron for pre-synaptic spike activity. The smaller the value of c, the lower the firing threshold ϑ and the earlier the post-synaptic response of the neuron occurs.

The similarity threshold s, on the other hand, is not a neural parameter, but instead a parameter of the one-pass learning algorithm in eSNN. The learning algorithm evolves repositories R_l of neurons – one for each class label l. The number of neurons in R_l depends on the value of s, and larger (smaller) s correspond to fewer (more) neurons. Each evolved output neuron is sensitive to a specific input pattern and hence represents a certain area or cluster in the data space. Since parameter s controls the number of neurons in a repository, it indirectly controls the size of the cluster represented by a certain output neuron. The smaller the value s is, the more training samples may activate a specific output neuron and as a consequence, the larger the cluster represented by this neuron becomes. In the extreme case of $s = 1$, for each training sample an individual output neuron is trained, while for $s = 0$, all training samples are mapped to a single output neuron.

8.4.1 *Setup*

For a controlled experimental analysis of the parameter evolution in QiSNN, we study each parameter separately. More specifically, in each experiment, a single parameter is selected for investigation and is then subjected to the optimisation through hHM-EDA. The remaining parameters, on the other hand, are fixed to reasonable default values. For example, selecting the modulation factor m for analysis results in the simultaneous optimisation of m and the feature subsets, while the values for the similarity threshold s and the firing threshold fraction c are fixed to predefined constants $\hat{s}$ and $\hat{c}$, respectively.

The following constants are chosen: $\hat{m} = 0.85$, $\hat{s} = 0.2$ and $\hat{c} = 0.75$. These values are directly adopted from the experimental results obtained on the spiral data set in the previous section. Since three different parameters exist in QiSNN, three experiments were undertaken. For each experiment, 30 independent runs are performed and results averaged. The spiral data set is used for studying the parameter evolution, since it is clearly a more challenging benchmark than the hypercube data set.

8.4.2 *Results*

The next sections discuss the results obtained from the described experimental setup.

Evolution of neuron repositories

First, we demonstrate that the size of the neuron repository is indeed dependent on the similarity threshold s as it was claimed in the previous section. Figure 8.14 shows the number of neurons in the repository R_l in dependence of s_l. A point in the diagram corresponds to a tuple $(s_l, |R_l|)$, where $|R_l|$ is the number of neurons in R_l. The tuple is extracted from the generational best solution obtained in each of the 400 performed generations. The number of neurons in R_l also depends on the size of the corresponding feature subset. For larger subsets, the similarity between samples decreases on average, since a sample contains more elements that may differ from the elements of other data samples. This dependence is reflected by the colour of a point in Figure 8.14. The lighter the colour, the more features are selected by the corresponding solution.

The figure reveals an interesting pattern on how the neuron number evolves during the optimisation process. In the early stage of the optimisation, on average, 10 out of 20 possible features are selected. Note the light colour in Figure 8.14 for values of $s_l \approx 0.5$, $l \in \{1, 2\}$. At this stage of the evolution, for most training samples an individual output neuron is created and stored in the repository. Parameter s does not impact the learning process much, since the input samples are very different to each other due to their comparatively high dimensionality. In the course of the optimisation process, the number of features decreases, which in turn increases the similarity between different samples on average. Due to this increasing similarity, more output neurons are merged and the number of neurons per repository decreases. At this stage of the evolution, most random and redundant features are already excluded from the optimisation. Note the dark coloured points for $|R_l| \approx 100$.

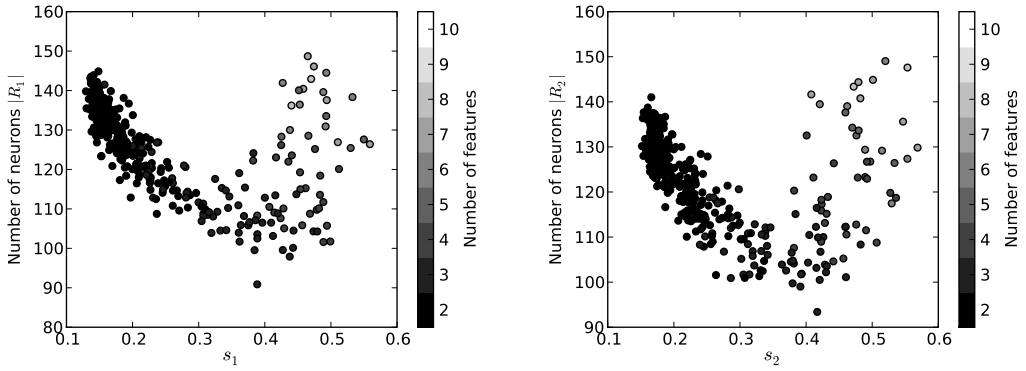


Figure 8.14: The number of neurons in the evolving neuron repository R_l in dependence of the similarity threshold s_l. A point in the diagram corresponds to a tuple $(s_l, |R_l|)$, where $|R_l|$ is the number of neurons in R_l. The colour of a point corresponds to the number of selected features. The lighter the colour, the more features are selected by corresponding solution.

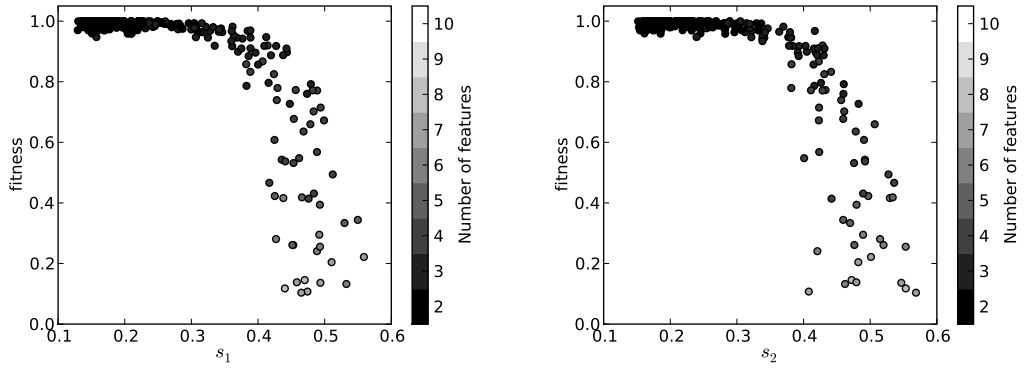
In order to further increase the classification accuracy, an increase of the neuron number in R_l seems beneficial. As stated above, more output neurons correspond to a finer clustering of the data space. Since the spiral data represents a highly non-separable classification problem, reducing the cluster size is meaningful. In the figure, this situation is reflected by a clear trend of s_l towards small values.

We conclude that, due to the dependence of $|R_l|$ on s_l, the neuron number $|R_l|$ increases proportionally with decreasing s_l. Furthermore, if the number of selected features is large, the value of s_l is of low importance, since training samples differ from each other mainly due to their comparatively large dimensionality.

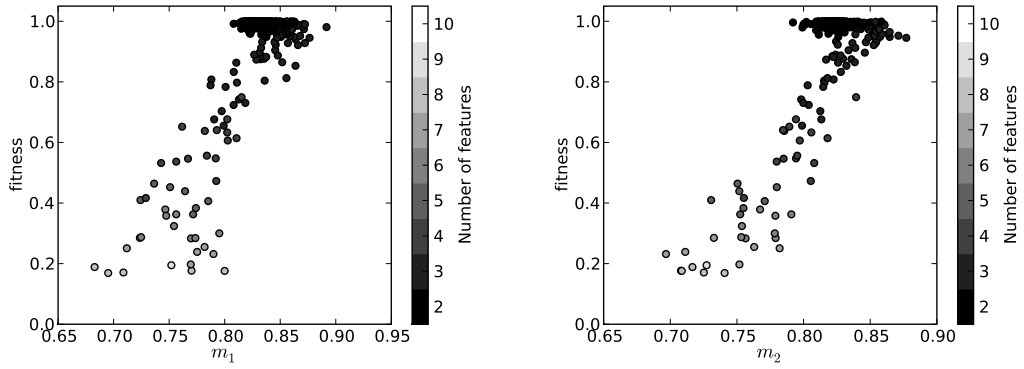
Evolution of the similarity threshold s

Figure 8.15 shows the results obtained from the three experiments described in the experimental setup above. A point in these diagrams corresponds to a tuple (f, x_l), where f denotes a fitness value and $x \in \{s, m, c\}$ denotes the parameter that is optimised through hHM-EDA. The tuple is extracted from the generational best solution. Similarly to Figure 8.14, each point is coloured in dependence of the number of features selected by this solution. The lighter the colour, the more features are selected.

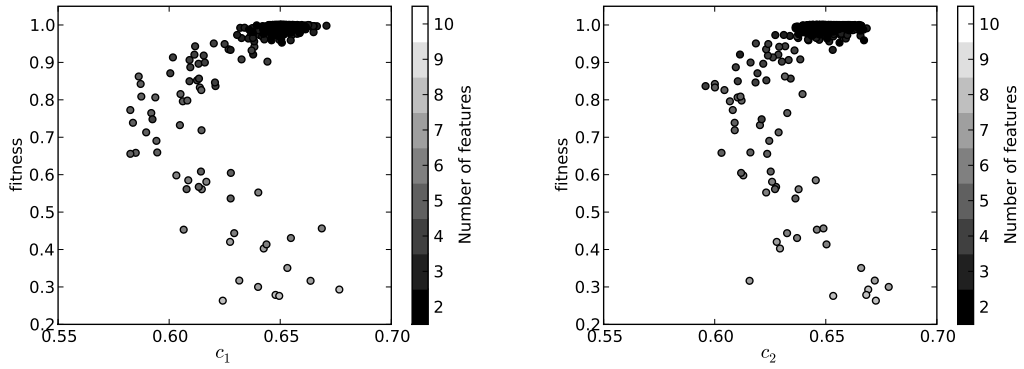
The independent evolution of the parameters shows some interesting patterns. In Figure 8.15a, the generational best fitness in dependence of the similarity threshold s_l is presented. We have concluded earlier that the value of s_l is less important, if the selected feature subset contains many irrelevant and/or redundant features. We observe in the early stage of optimisation, that on average $s \approx 0.5$, which is the ex-



(a) Fitness of the generational best solution in dependence of s_l



(b) Fitness of the generational best solution in dependence of m_l



(c) Fitness of the generational best solution in dependence of c_l

Figure 8.15: The fitness in dependence of the parameter x_l with $x \in \{s, m, c\}$. A point in these diagrams represents a tuple (f, x_l) extracted from the generational best individual. The colour of a point corresponds to the number of selected features. The lighter the colour, the more features are selected by corresponding solution. For each experiment a single parameter is subjected to the optimisation process, while the remaining parameters are fixed to reasonable default values.

pected value of a uniform random variable sampled in the range $[0, 1]$. Parameter s_l is optimised by hHM-EDA in exactly this range. The classification accuracy improves mainly due to the selection of appropriate feature subsets. Note that, according to the experimental setup, only the features and the variable s_l are subject to optimisation. Hence, only the number of features and the value of s_l determine the classification accuracy of a solution.

The importance of s_l is boosted in later stages of the optimisation process. A clear trend of s_l towards smaller values is noted after the removal of most redundant and irrelevant features. We conclude, that s_l is a low salient search variable and its importance depends mainly on the number and quality of selected features.

Evolution of the modulation factor m

The modulation factor is optimised in the range $[0.5, 1]$. The random sampling of the parameter results in an average value of $m \approx 0.75$ at the first generations of the evolutionary process, *cf.* Figure 8.15b. Nevertheless, even at this early stage of the optimisation, a clear trend is noted. The modulation factor increases towards a value of $m \approx 0.85$. At this stage, a high-quality feature subset has also evolved. We observe a decrease of the significance of m, since most values in the range $[0.8, 0.85]$ report excellent classification results.

We conclude, that the modulation factor m is significant especially in early stages of the optimisation. After a high-quality feature subset is identified, the precise modulation is less important and a certain range of values is suitable.

Evolution of the firing threshold fraction c

Similarly to the modulation factor m, the firing threshold fraction c is optimised in the range $[0.5, 1]$. From Figure 8.15c, it is immediately noted that, initially c follows a clear trend towards small values. The classification accuracy improves mainly due to the optimisation of this parameter. An accuracy of ≈ 0.5 is achieved even without the identification of any high-quality feature subset. Note the light coloured tuples at fitness levels $f < 0.5$.

This observation is explained by the fact, that smaller firing thresholds allow the network to *respond* to any presented input, even if this response is not necessarily correct. As explained in chapter 2, section 2.6.2, it is possible that the eSNN classifier remains silent, *i.e.* no output neuron is activated after the presentation of an input sample. This case is considered as a mis-classification. Thus, since a small threshold allows the eSNN to respond to a presented input, this configuration turns the eSNN

into a random classifier. A random classification reports on average an accuracy of ≈ 0.5 and thus, small values represent an attractor in the search space of c.

In later generations of the evolution, suitable feature subsets are identified which results in classification accuracies higher than 0.5. At this stage, larger firing thresholds are beneficial, since they allow a more precise control over the activation of certain output neurons. We observe a changing trend in the evolution of c after the accuracy level has reached $f \approx 0.7$. At the final stages of the optimisation process, fractions $c \approx 0.65$ are suitable to achieve excellent classification results.

We conclude, that the firing threshold fraction c is a high salience search variable. Small values c represent a strong attractor, since they allow the transformation of eSNN into a random classifier.

8.5 ROLE OF NEURAL ENCODING

In this section, we investigate the impact of the encoding parameters on the classification performance of eSNN. As explained in chapter 2, the so-called rank order population encoding is employed in the context of eSNN. This encoding requires the setting of two parameters. Parameter M controls the number of Gaussian receptive fields, while parameter β controls the width (variance) of each Gaussian. We immediately notice that these variables directly affect the network size, since a larger number of receptive fields increases the number of input neurons of the network. As a consequence, also the learning process is affected and thus potentially the classification performance.

There is another important aspect of the neural encoding that is highlighted here. The encoding describes a mapping from a *lower dimensional* real valued input vector space to a *higher dimensional* vector space of spike times. This transformation has the potential to simplify the classification task. In fact, a similar concept is the working principle of numerous other classification methods, *e.g.* Support Vector Machines and the principle of the Echo State Machines and the Liquid State Machines (LSM) (Maass et al., 2002). For example, the LSM also employs spiking neurons for solving classification or time series prediction problems. The idea is to transform a sequence of input spikes into multiple spike trains through the excitation of a large static recurrent SNN. The response from this static network is then interpreted by a readout function. This readout function can be memory-less and even linear, since it is expected that the mapping step has transformed the problem into a linear separable one.

Indeed, it was demonstrated in (Soltic et al., 2008) that an appropriate configuration of the rank order population encoding can significantly simplify the classification problem. A visualisation of the obtained encoded spike trains revealed distinct patterns for samples belonging to a certain class. It was also concluded that the parameter configuration of the encoding method is critical for the functioning of eSNN and that too few or too many receptive fields deteriorate the classification performance.

In order to analyse the relationship between the encoding parameters and the classification behaviour of eSNN we investigate the following experimental setup. For all experiments the spiral data set is used and only the two relevant dimensions x and y are considered. More specifically, the feature selection mechanism of QiSNN is switched off for this study. Only the parameters of eSNN (modulation factor, similarity threshold and firing threshold) are optimised. The parameter configuration for M and β is varied according to $M \in \{10, 20, 30, 50\}$ and $\beta \in \{1, 1.5, 2\}$ totalling in twelve different experiments. The optimisation process is allowed to run 400 generations.

From each of the twelve experiments a typical evolved eSNN is obtained which is considered to be optimally configured and trained. Each eSNN is then tested on $100,000$ test samples (x_i, y_i) which are equally distributed in the data space with $-1 \leq x_i \leq 1$ and $-1 \leq y_i \leq 1$ for $1 \leq i \leq 100,000$. For all samples, the classification output of the evolved eSNN is determined and visualised in Figure 8.16. Each diagram shows the results for one of the twelve performed experiments.

The axes in the diagrams represent the two dimensions (features) of the classification problem, x and y. The colour of a point reflects the classification output of the corresponding eSNN: white points belong to samples which are classified as spiral A, while black points represent samples classified as spiral B. The gray coloured points represent data samples for which no label could be determined, *i.e.* none of the output neurons of the trained eSNN emitted a spike and the network remained silent for the presented input.

It is clearly demonstrated that the configuration of the encoding parameters is critical for the functioning of eSNN, which is coherent with the findings presented in (Soltic et al., 2008). The number of receptive fields affects the ability of eSNN of distinguishing between different input vectors. Thus, parameter M determines the *resolution* of the data space representable by eSNN. If the resolution is not fine-grained enough, the data space can not be optimally partitioned by eSNN and the classification performance decreases, *cf.* the three diagrams where $M = 10$. On the other hand, if the resolution is too fine, the output neurons become increasingly specialised to the presented training samples, which in turn decreases the generalisa-

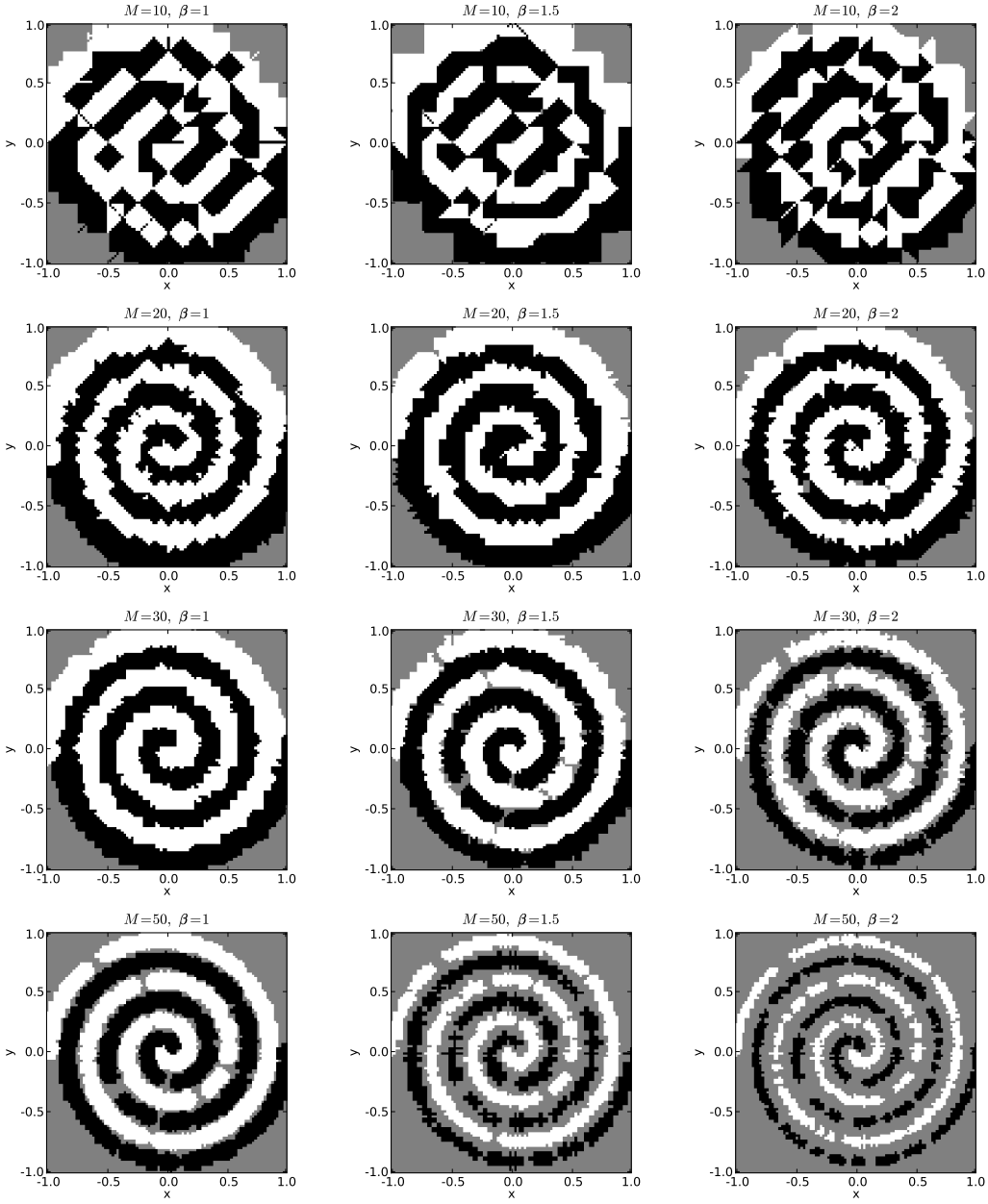


Figure 8.16: Output patterns of trained eSNN using different configurations of the rank order population encoding. The axes represent the two features of the two-spiral data set. White (black) coloured points belong to test samples which are classified as spiral A (B). The gray coloured points represent data samples which eSNN could not classify.

tion ability of eSNN. The diagrams where $M = 50$ illustrate an example for the loss of generalisation. We notice the significant increase of gray areas, *i.e.* unclassified samples, in these figures. The output neurons are strongly specialised and are only sensitive to the data points learnt from the training samples. The ability of eSNN to

interpolate data points between training samples is significantly reduced (notice the gray "gaps" between the two spirals).

The generalisation ability of the output neurons is also strongly affected by the configuration of parameter β. Larger β values increase the specialisation of the neurons. The importance of β increases with the number of receptive fields. For $M = 10$ fields, the value of parameter β is almost irrelevant, while the impact of the variable is clearly visible for $M = 50$.

8.6 CONCLUSION

This chapter proposed an integrated feature and parameter optimisation framework based on the combination of the heterogeneous optimisation algorithm hHM-EDA and the eSNN classification method. According to the quantum metaphor of the binary part of hHM-EDA inherited from vQEA, the feature space is explored probabilistically using a superposition of feature subsets. Due to this interpretation, the novel eSNN based feature selection method was named Quantum-inspired Spiking Neural Network (QiSNN) framework.

The classification and feature selection performance of the method was demonstrated on two synthetic benchmark problems. QiSNN reported excellent results in comparison to traditional wrapper-based feature selection methods. This observation was partly explained by the removed neutrality in the fitness landscape represented by the eSNN classifier.

The benefit of the novel heterogeneous optimiser hHM-EDA was clearly demonstrated. In comparison to the binary representation of vQEA, hHM-EDA allows a faster, smoother and more reliable exploration of the mixed variable search space. The performance difference between vQEA and hMH-EDA is expected to increase with larger problem sizes. However, additional analysis is required to provide statistical evidence for this claim.

The analysis of the parameter evolution in QiSNN revealed some interesting characteristics. The features are clearly the most important parameters in the optimisation process and are commonly optimised first. In general, the eSNN parameters strongly depend on the quality of the selected feature subset. The similarity threshold is of low significance, if inadequate feature subsets are selected, which occurs mainly in the early stage of the optimisation. The modulation factor is important in early generations and becomes less significant after high-quality feature subsets are identified.

It was also noted that the firing threshold fraction is a very critical parameter, since it has the ability to turn eSNN into a random classifier.

The self-adapting nature of QiSNN due to the simultaneous evolution of network parameters and feature subsets represents a highly desirable characteristic in the context of machine learning and knowledge discovery. It promotes a straight-forward application of the framework to specific problem domains without the requirement of expert knowledge in the area of spiking neurons.

The configuration of the neural encoding method is very important for the functioning of the eSNN classifier. It was argued that an appropriate encoding mechanism can simplify the classification task. The parameters involved in the rank order population encoding affect both the separation resolution and the generalisation ability of eSNN. A careful fine-tuning of these parameters in dependence of the data set to be classified can significantly improve the classification performance of eSNN. Thus, besides numerous potential applications for real-world classification problems, future research may include the optimisation of additional variables of the system, *e.g.* the parameters of the employed rank order population encoding method.

Chapter 9

APPLICATION OF QISNN – A CASE STUDY ON ECOLOGICAL MODELLING

This chapter presents the findings of a case study where the QiSNN framework is applied on a real world data set in the context of an ecological modelling problem. For many invertebrate species little is known about their response to environmental variables over large spatial scales. That knowledge is important since it can help to identify critical locations in which a species that has the potential to cause great environmental harm might establish a new damaging population. The usual approach to determine the importance of a range of environmental variables that explain the global distribution of a species is to train or fit a model to its known distribution using environmental parameters measured in areas where the species is present and where it is absent.

In this study, meteorological data that comprised 68 monthly and seasonal temperature, rainfall and soil moisture variables for 206 global geographic sites were compiled from published records. These variables were correlated to global locations where the Mediterranean fruit-fly (*Ceratitis capitata*), a serious invasive species and fruit pest, was recorded at the time of the study, as either present or absent (CABI, 2003). The data set is balanced meaning that it has an equal number of samples for each of the two classes. Motivated by inadequate results (Worner, Lankin, Samarasinghe, & Teulon, 2002; Cocu, Harrington, Rounsevell, Worner, & Hulle, 2005; Watts & Worner, 2006) using a different method, namely the multi-layer perceptron (MLP), this study aims to identify important features relevant for predicting the presence/absence of this insect species. The obtained results may also be of importance to evaluate the risk of invasion of certain species into specific geographical regions.

In the following sections, first the experimental setup is explained, followed by an analysis and discussion of the obtained results. Since this study is undertaken in collaboration with Dr Sue Worner from the Centre for Bio-protection at Lincoln Uni-

versity, Christchurch, New Zealand, the results are also analysed from an ecological point of view.

9.1 EXPERIMENTAL SETUP

In the previous chapter, QiSNN reported promising results on synthetically designed benchmark data. In this case study, QiSNN is investigated in a real world scenario. Similarly to the experiments presented in chapter 8, two optimisation methods for QiSNN are considered: the binary-only optimisation algorithm vQEA and the heterogeneous optimiser hHM-EDA. For a better recognition of these two setups, we refer to them as the heterogeneous and the binary QiSNN, respectively, for the rest of this chapter.

In order to allow a comparison of results, we apply a traditional classification method on the same data set by exchanging the eSNN classifier for the classical naïve Bayesian classifier (NBC). A similar scenario was discussed in the previous chapter: vQEA is used to evolve an appropriate feature subset while the quality of a subset is determined through training a NBC (instead of an eSNN) and reporting its classification accuracy. Based on this evaluation, new feature subsets are selected.

We note that this problem represents a combinatorial optimisation task for which the binary nature of vQEA is well suited. Apart from the discretisation of the data set, which is a requirement for NBC, the method does not require the setting of any other parameters thus no parameter optimisation is needed here.

A number of careful parameter choices have to be made. For all optimisation methods, a population structure of ten individuals organised in a single group is chosen, which is globally synchronised every generation. The parameters for the mean and standard deviation shift in the heterogeneous QiSNN were set to $\theta_{\mu}^{(H)} = 0.1$ and $\theta_{\sigma}^{(H)} = 0.02$ respectively, the learning rate for the binary model was $\theta^{(H)} = \pi/100$. In the binary QiSNN, the learning rate was set to $\theta^{(B)} = \pi/200$, while in combination with the NBC a rate $\theta^{(NBC)} = \pi/100$ worked favourably.

QiSNN automatically adapts its parameters during the evolutionary process. Since the classification task is a two-class problem, six parameters are involved in the optimisation. In the heterogeneous QiSNN this search space is explored by the continuous solution part of the optimiser. The binary vQEA, on the other hand, requires the conversion of bit strings into real values. In the experiments, four bits per variable were enough to offer sufficient flexibility for the parameter space. For the conversion itself, a Gray code was used.

In terms of the population encoding for eSNN, especially the number of receptive fields needs careful consideration since it affects the resolution for distinguishing between different input variables. After some preliminary experiments 10 receptive fields were chosen, the centres uniformly distributed over the interval $[0, 1]$, and the variance controlling parameter $\beta = 1.5$.

In every generation, the 206 samples of the data set were randomly shuffled and divided into a training and testing set, according to a ratio of 75% (154 training and 52 testing samples). The chromosome of each individual in the population was translated into the corresponding parameter and feature space, resulting in the generation of a fully parametrised, but untrained, eSNN or NBC and a feature subset[1]. The created eSNN or NBC of each individual was then independently trained and tested on the appropriate data subsets. For the computation of the classification error, we determined the ratio between correctly classified samples and the total number of testing samples.

Each of the three setups were allowed to evolve over a total number of 4000 generations. In order to guarantee statistical relevance, 30 independent runs using different random seeds were performed for each setup.

9.2 EXPERIMENTAL RESULTS

In Figure 9.1, the evolution of the average best feature subset in every generation is presented using eSNN and NBC as classifiers. The colour of a point in this diagram reflects how often a specific feature was selected at a certain generation. The lighter the colour, the more often the corresponding feature was selected at the given generation. It can clearly be seen that a large number of features have been excluded during the evolutionary process. Many features have been identified to be irrelevant by all algorithms, although also some significant differences between the evolved feature subsets is noticed. Figure 9.1 clearly shows the similarity of the feature subsets obtained by both versions of QiSNN. Nevertheless, the heterogeneous version reports greater consistency in the feature rejection. Also, the latter one selected significantly fewer features than the binary QiSNN and NBC, *cf.* Figure 9.2: on average 14 features were selected using binary QiSNN, ten in case of the heterogeneous QiSNN and 18 using NBC. We will analyse these features from an ecological point of view in the next section. The trend in Figure 9.2 suggests the evolution for the NBC is incomplete. Compared to the binary QiSNN, the heterogeneous version addition-

1 In case of the NBC, only a feature space exists, since no parameters have to be tuned.

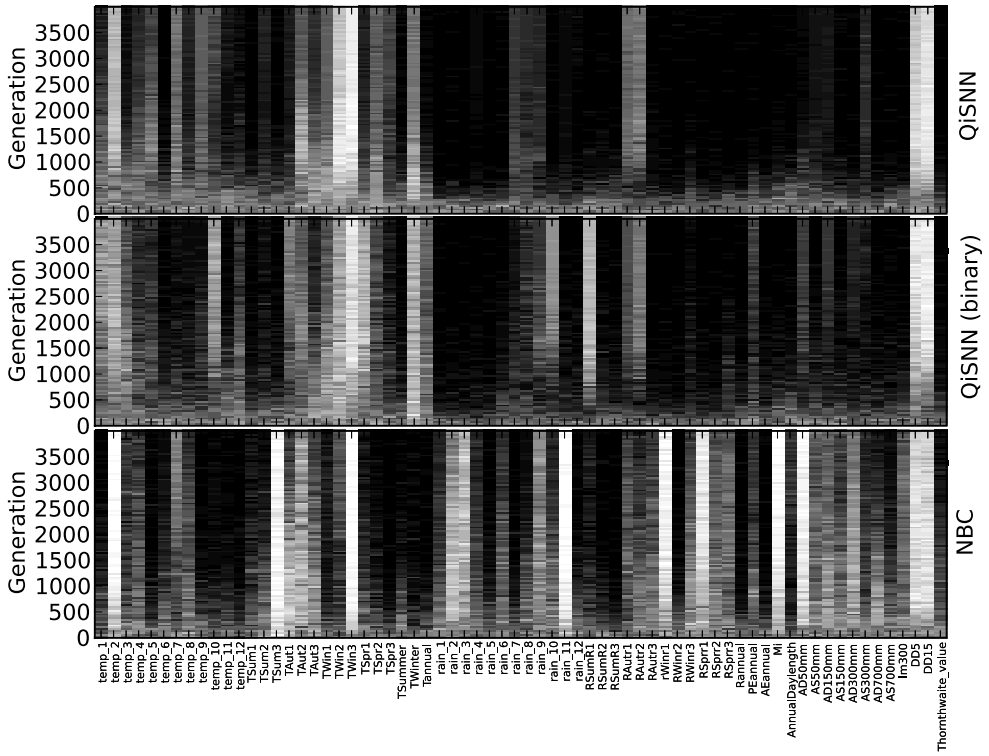


Figure 9.1: Evolution of the feature subsets on the ecological data set. The lighter the colour of a point in the diagram, the more often a specific feature was selected at the given generation. Each point is the average of 30 independent runs.

ally rejected the following features: temp1, temp3, TAut2, TSpr1, Tannual, rain10, RSumR2, PEAnnual. The overall classification accuracy was similar among all tested algorithms.

The eSNN classifier appears to be rather consistent in excluding features, since most of the 30 independent runs have agreed at least about the irrelevant features, hence many black columns appear in the diagram. The situation is different for features that have been identified as relevant in most of the runs. In a small number of runs, exactly these features were considered to be irrelevant, as reflected by the light grey columns in Figure 9.1. For these features, several hypothesis can be derived. We emphasise that the features for which the classifiers are undecided may be not important, but also not misleading during the evolutionary search. Hence, they are randomly included in the final feature subset by any of the runs performed. It is also likely that some features are equally relevant (*i.e.* redundant features), so at least one of them will be selected as a representative of these features by the algorithm. Different runs will most likely select a different feature, thus the final subset is varying.

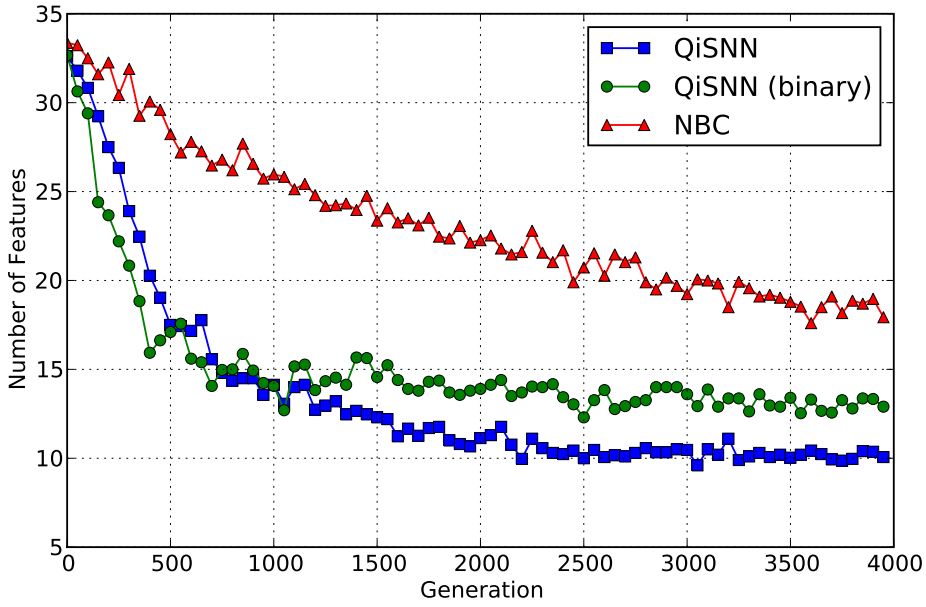


Figure 9.2: The evolution of average feature numbers on the ecological data set. All methods are clearly capable of reducing the number of features. The two versions of QiSNN exclude significantly more features than NBC.

Furthermore, it is possible that some features are present conditional to the presence/absence of others. Hence, the average evolved feature subset can not be consistent in all runs and the ecological analysis of the feature subset should include all features that have been selected more frequently than a certain percentage in all runs performed.

In the case of NBC, an opposite situation can be observed. Some features are clearly found to be relevant in all 30 runs, which is in contrast to the results obtained by QiSNN. However, for many other features, no definite decision can be made, since some of the runs reported a given feature to be relevant, but at the same time an almost equal number of runs reported the exact opposite. The explanations given earlier about redundant and conditional features are true for NBC as well.

It has to be noted that there is a difference in the way NBC and eSNN classify a test sample. NBC *always* reports an answer (either class 0 or class 1). As a result, the classification accuracy of NBC is never lower than approximately 50% which corresponds to a random classification. QiSNN, on the other hand, is also able to *deny* classification (either class 0, class 1 or *undecided*). The latter case is considered to be a mis-classification of the presented sample. Thus, QiSNN has a third classification

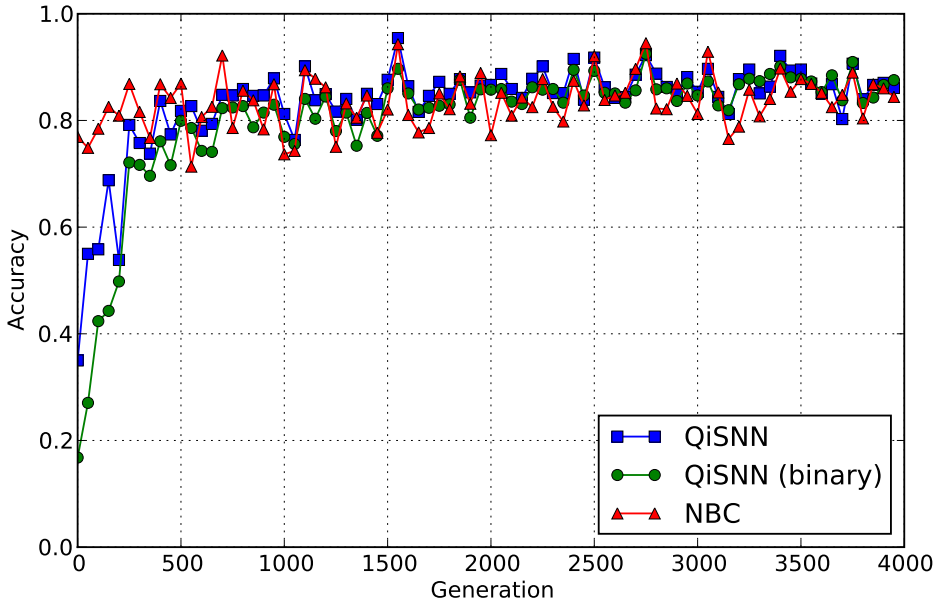


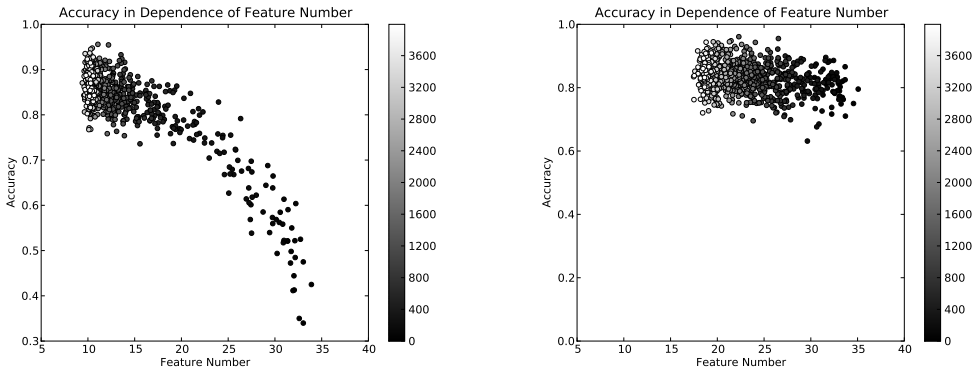
Figure 9.3: The evolution of accuracy on the ecological data set. All methods report similar classification performance after the evolution of 4000 generations.

option, *i.e.* remaining silent, and as a consequence, a random classification would result in an accuracy of 33%, compared to 50% for NBC.

Furthermore, QiSNN starts the evolution with an non-optimised parameter configuration. Thus, the likelihood of mis-classification is large in the early stage of the optimisation. In later generations, this situation changes since QiSNN discovers a working parameter configuration. At this stage of the run, the accuracies of both algorithms can be compared fairly.

This situation is clearly demonstrated in Figure 9.3 which presents the evolution of the average accuracy over 4000 generations. After 500 generation, QiSNN achieves accuracy levels that are similar to NBC. The average accuracy of the best individual in the population after the evolution was constantly above 80% for both tested classifiers, NBC displaying a slightly higher variance during the evolutionary run compared to QiSNN.

It is interesting to see how strongly the classification accuracy depends on the feature number for each of the tested algorithms. In Figure 9.4, this dependence is investigated for the eSNN and NBC classifiers. Since the binary and heterogeneous QiSNN show a similar behaviour here, we chose the heterogeneous QiSNN as the representative for both QiSNN. Each point in the diagram corresponds to a tuple (accuracy, feature number) obtained from the generational best individual of



(a) eSNN (b) NBC

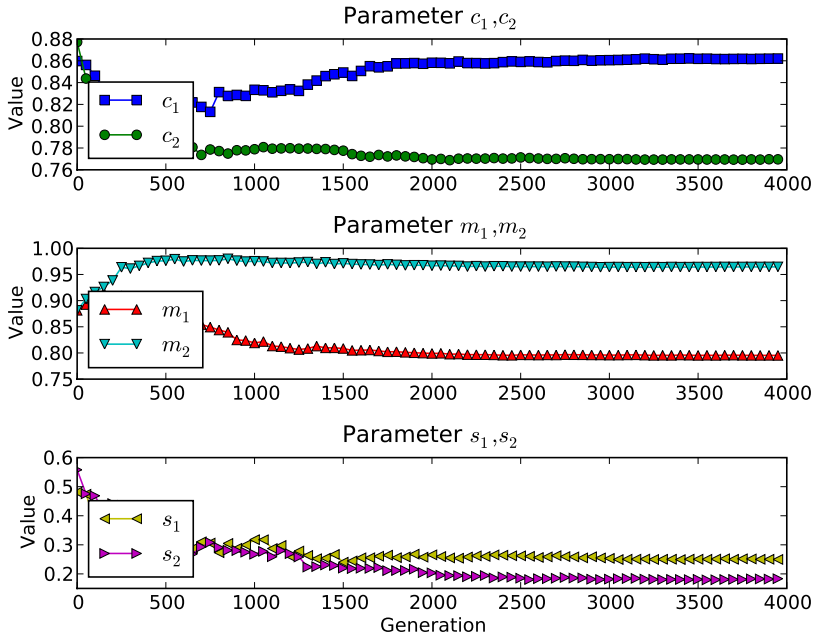
Figure 9.4: The diagrams show the accuracy as a function of the feature number for eSNN (a) and Naïve Bayesian classifier (b). The different grey levels correspond to the generation in which a given data point was obtained. The lighter the colour the later the generation. For eSNN the accuracy is highly dependent on the feature number, which is in strong contrast to NBC.

every generation. The colour indicates the generation itself; the lighter the colour, the later the generation in which a given tuple was obtained. In the case of eSNN (*cf.* Figure 9.4a), a strong relationship between feature number and accuracy can be observed. Even for small decreases of the feature number significant accuracy improvements are reported. Since the evolutionary search is driven by the classification accuracy only, solutions having a small number of features represent a strong attractor in the search space. In the case of NBC, smaller feature subsets are also rewarded by higher classification accuracy. Nevertheless, this award is less obvious compared to the one observed in eSNN, *cf.* Figure 9.4b. Thus, the fitness landscape (in terms of feature number) represented by NBC appears to be flatter than the one represented by eSNN. It is noteworthy that flat fitness landscapes are an undesired property of any fitness function in an evolutionary algorithm.

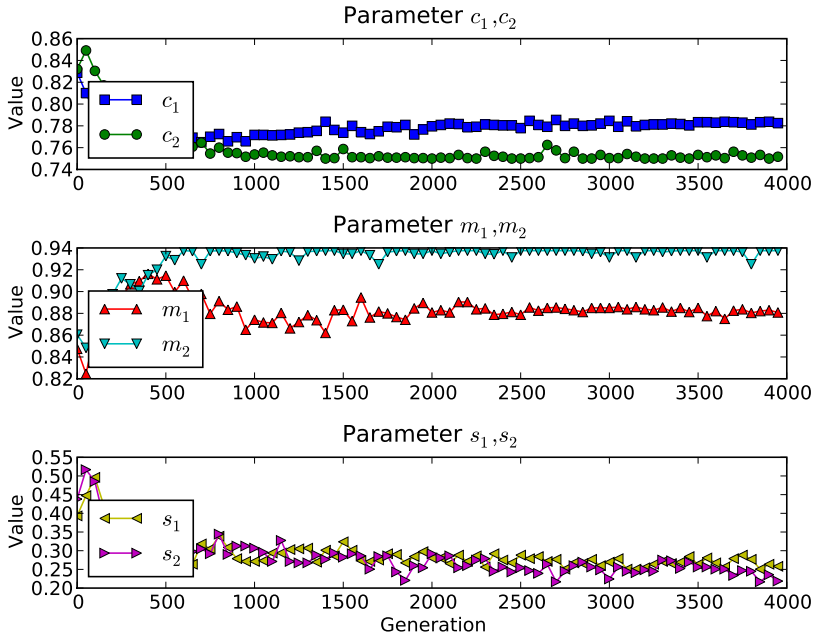
Figure 9.5 presents the evolution of the parameters of the eSNN architecture. All three pairs display a steady trend and evolve constantly towards a certain optimum, not reporting too much variability. We take this as an indicator that these parameters were indeed carefully controlled by the corresponding optimisation algorithm.

9.3 INTERPRETATION FROM ECOLOGICAL POINT OF VIEW

Using the heterogeneous QiSNN, on average only 10 features were selected in a particular evolutionary run. However, since the evolved feature subsets were not



(a) Heterogeneous QiSNN



(b) Binary QiSNN

Figure 9.5: Evolution of parameters in the QiSNN framework on the ecological data set. Three parameter pairs are optimised during the evolutionary process. Due to the continuous representation of the parameter space, the enhanced QiSNN (upper figure) reports a smoother exploration compared to the binary optimisation (bottom figure).

identical in all of the runs and the presence/absence of features is also expected to be conditional on the presence/absence of other features, we have decided to include into the ecological analysis all features that have been selected by at least 20% of the 30 independent runs. Thus, in the case of QiSNN the analysis indicates 25 variables that were considered as being involved in the determination of the classification outcome after the evolution of 4000 generations.

Table 9.1 summarises the final feature subsets obtained by each of the classification methods. A feature is marked as rejected when at least 20% of all performed runs have discarded this feature at the end of the evolutionary run. If a feature was selected in 80% or more of all runs, it is marked as selected. The remaining features have been labelled as "undecided" in the table. As mentioned earlier, the table reflects the fact that eSNN is more consistent in rejecting features than NBC. For this reason, we concentrate our ecological analysis on the results obtained by QiSNN only. The features included in this analysis are presented by the two columns ("Undecided" and "Select") corresponding to the eSNN method in Table 9.1.

Winter (TWIN2, TWIN3, TWINTER) and early spring (TSPR1) temperatures, and early summer rainfall (RSUMR1) were particularly strong features along with the degree-days (DD5 and DD15). Degree-days are the accumulated number of degrees of temperature above a threshold temperature (5° and 15° in this case) over time (in this data set over the whole year). It would be expected that the latter two variables would be closely correlated. These results correspond to another analysis where more conventional statistical and machine learning methods were used to identify the contribution of environmental variables to *C. capitata* presence or absence (Worner, Leday, & Ikeda, 2008). While there is no indication from this analysis whether the features have a negative or positive effect on the distribution of the species, it is known that *C. capitata* is limited by the severity of temperatures in the winter and early spring and extremes of wet or dry conditions in the summer (Vera, Rodriguez, Segura, Cladera, & Sutherst, 2002).

The accuracy of the resulting model on the test set, however, is not only higher than that for the model using the full feature set, but also higher than that found by (Worner et al., 2008) using a range of conventional models. The clear potential for further improvement of classification accuracy with model refinement, as well as automatic optimisation of parameters, makes this an extremely useful approach for the analysis and modelling of complex, noisy ecological data.

Feature	eSNN			NBC			Agreement
	Reject	Undecided	Select	Reject	Undecided	Select	
Temp1		yes		yes			
Temp2		yes				yes	
Temp3		yes			yes		U
Temp4	yes				yes		
Temp5		yes		yes			
Temp6	yes			yes			R
Temp7	yes				yes		
Temp8	yes			yes			R
Temp9	yes			yes			R
Temp10		yes		yes			
Temp11	yes			yes			R
Temp12	yes			yes			R
TSum1	yes			yes			R
TSum2	yes			yes			R
TSum3	yes					yes	
TAut1		yes			yes		
TAut2		yes			yes		U
TAut3	yes			yes			R
TWin1		yes		yes			
TWin2		yes			yes		U
TWin3			yes			yes	S
TSpr1		yes		yes			
TSpr2		yes		yes			
TSpr3	yes			yes			R
TSummer	yes			yes			R
TWinter		yes		yes			
Tannual		yes		yes			
Rain1	yes			yes			R
Rain2	yes				yes		
Rain3	yes				yes		
Rain4	yes				yes		
Rain5	yes			yes			R
Rain6	yes			yes			R
Rain7	yes			yes			R
Rain8		yes		yes			
Rain9		yes			yes		
Rain10		yes		yes			
Rain11	yes					yes	
Rain12	yes			yes			R
RSumR1		yes		yes			
RSumR2	yes			yes			R
RSumR3	yes			yes			R
RAutr1		yes			yes		U
RAutr2		yes		yes			
RAutr3	yes				yes		
RWinr1	yes					yes	
RWinr2	yes			yes			R
RWinr3	yes			yes			R
RSprr1	yes					yes	
RSprr2	yes				yes		
RSprr3	yes				yes		
Rannual	yes			yes			R
PEannual		yes		yes			
AEannual	yes			yes			R
Mi	yes					yes	
ADayLen	yes			yes			R
AD50mm		yes				yes	
AS50mm	yes				yes		
AD150mm		yes			yes		
AS150mm	yes				yes		
AD300mm	yes				yes		
AS300mm	yes			yes			R
AD700mm	yes			yes			R
AS700mm	yes			yes			R
Im300	yes				yes		
DD5			yes			yes	S
DD15			yes		yes		
Thornthw	yes			yes			R
Total (%)	63.2	32.3	4.4	57.4	29.4	13.2	47.0

Table 9.1: Final feature subsets obtained from the ecological experiments. U=Undecided, R=Rejected, S=Selected

9.4 CONCLUSION

In this chapter, the QiSNN feature selection framework was applied on a real world problem in the context of an ecological modelling problem. Results have been compared to the traditional Naïve Bayesian Classifier (NBC). Although no significant

difference in terms of accuracy between the two classification methods was obtained, some important experimental observations were made. While NBC represented a rather flat fitness landscape for the evolutionary algorithm, in which lower numbers of features receive only little fitness rewards, the eSNN used in QiSNN reported a clear correlation between classification accuracy and feature number. As a result, eSNN was capable of decreasing the feature number not only faster than NBC, but was also more consistent in excluding features from the optimisation process. NBC on the other hand appeared to be more consistent in selecting features, while being less consistent in rejecting them. The obtained feature subsets were analysed by an ecological expert and found to be coherent with current knowledge in this area. In a previous analysis, in which conventional statistical methods were applied on this data set without performing any feature selection beforehand, a worse classification accuracy was reported.

Chapter 10

CONCLUSION AND FUTURE DIRECTIONS

This chapter summarises the achievements of the presented research and provides several directions for future work.

10.1 SUMMARY OF ACHIEVEMENTS

This thesis proposed an integrated feature and parameter optimisation framework built upon the evolving spiking neural network architecture. The framework combines an evolutionary optimisation algorithm with an eSNN based classification method following the wrapper approach. Due to the quantum computing metaphor of the employed binary optimisation algorithm, the novel technique was introduced as the Quantum-inspired Spiking Neural Network (QiSNN) framework. The evolutionary process evolves an appropriate feature subset while simultaneously optimising the neural and learning related parameters of the eSNN. The synaptic weights of the neural network are not subject to evolution, but are trained by a fast one-pass learning algorithm instead. The QiSNN framework and part of its analysis was initially published in (Schliebs, Defoin-Platel, & Kasabov, 2009) and later extended in (Schliebs, Defoin-Platel, Worner, & Kasabov, 2009a).

The QiSNN framework offers a number of advantages compared to the individual application of the eSNN classifier. First, the parameters of eSNN are self-adapting promoting the straight-forward application of the method to a specific problem domain. This characteristic is highly desirable for any machine learning and knowledge discovery method, especially in the context of an increasing amount of interdisciplinary research. As a consequence the framework effectively avoids a poor classification performance caused by the choice of inappropriate parameter configurations by the experimenter. Second, feature selection can significantly improve the classi-

fication accuracy of eSNN and enhances the suitability of the method for real-world problems.

The novel method was experimentally investigated on a number of data sets including both synthetic and real world problems. It was shown that the eSNN classifier responds very sensitively to redundant and irrelevant features. It is noteworthy that irrelevant features may decrease the performance of the method significantly. As a consequence, feature selection is very important for eSNN. The sensitivity of eSNN to noise is effectively exploited by the evolutionary optimisation algorithm. Relevant, redundant and irrelevant features were reliably detected in the investigated benchmark problems.

In a case study, the QiSNN framework was applied to an ecological modelling problem. Results were compared to the traditional Naïve Bayesian Classifier (NBC). QiSNN decreased the number of features faster and more consistently than NBC. The obtained feature subsets were analysed by an ecological expert and found to be coherent with current knowledge in this area. The case study was recently published in (Schliebs, Defoin-Platel, Worner, & Kasabov, 2009b) and (Schliebs, Defoin-Platel, Worner, & Kasabov, 2009a). In a previous analysis, in which conventional statistical methods were applied on this data set without performing any feature selection beforehand, a worse classification accuracy was reported.

For the simultaneous evolution of feature subsets and eSNN parameter configurations a specialised evolutionary algorithm was developed that allows the simultaneous exploration of a binary and a continuous search space. This method is a novel and original contribution to the field of evolutionary computation. The algorithm hybridises two additional evolutionary methods and was introduced as the heterogeneous Hierarchical Model Estimation of Distribution Algorithm (hHM-EDA).

In its core, hHM-EDA combines the novel Versatile Quantum-inspired Evolutionary Algorithm (vQEA), and the novel continuous Hierarchical Model Estimation of Distribution Algorithm (cHM-EDA). hHM-EDA was experimentally investigated and its competitive performance was demonstrated when compared to eight different optimisation techniques. Furthermore, guidelines for the configuration of hHM-EDA were developed and tested as part of the analysis. In terms of computational cost, hHM-EDA requires very little algorithmic overhead in contrast to a number of other tested algorithms. Overall, hHM-EDA is a light-weight, fast and reliable optimisation method that is easy to configure and flexible to use. However, the analysis of more test functions is suggested to provide additional statistical evidence to this claim.

The binary representation employed in hHM-EDA is explored by vQEA that was introduced as an improvement over the Quantum-inspired Evolutionary Algorithm

(QEA) previously proposed in (Han & Kim, 2002). An extensive experimental analysis demonstrated that apart from the quantum metaphor, vQEA is an original approach belonging to the class of EDA. The main differentiating feature of vQEA to other EDA approaches is a multiple probabilistic model that is organised in a structured population of individuals. The advantages of manipulating several probability vectors instead of only one were empirically demonstrated. vQEA is an effective optimiser that works with fairly generic settings of its control parameters for a collection of benchmark problems of various sizes, with different levels of interactions between variables and numbers of neutral dimensions. Multiple probability vectors compensate for a finite number of decision errors while the population structure allows an adaptive learning speed and directly controls the diversity of the solutions sampled by vQEA. As part of the thesis, vQEA was first published in (Defoin-Platel et al., 2007) followed by a comprehensive analysis in (Defoin-Platel et al., 2009).

The behaviour and the robustness of vQEA was analysed on several benchmark problems using different noise landscapes. The study revealed a significant benefit of vQEA in comparison to other EDA approaches. It was shown that the selective pressure during the evolutionary process can be controlled by varying the population structure. Small population sizes in combination with few global synchronisation events decrease the selective pressure, while a fully synchronised population structure increases it. This knowledge may prove very important for the additional fine-tuning of parameters on noisy problems.

The continuous representation employed in hHM-EDA is explored by cHM-EDA that was developed as an extension of vQEA towards numerical optimisation. The probabilistic model of vQEA, *i.e.* the $\mathcal{Q}$bit, was replaced with Gaussian distributions. All key characteristics of vQEA, namely a multiple probabilistic model, a hierarchical population structure and a convergence dependent learning rule, are also part of its extension. The method was named continuous hierarchical model EDA, since the quantum metaphor has become inappropriate in the context of the Gaussian distributions.

cHM-EDA was investigated on a state-of-the-art benchmark suite consisting of 25 different test functions covering a variety of different problem characteristics. The functions range from simple separable uni-modal problems, over non-separable, non-linear, non-symmetrical, rotated and scalable functions, to complex hybrid composition functions in which several different function properties are mixed together. Additionally, some noisy benchmarks were considered. The overall performance of cHM-EDA is very competitive, especially on difficult, high dimensional problems. Issues arise when the search space contains flat areas. Here an adaptive learning rate

for the standard deviation update of the model, as *e.g.* employed in some other evolutionary techniques, might be beneficial. Along with the benchmark experiments, some practical guidelines for parameter configuration were presented. Furthermore, the scalability, robustness and convergence speed of cHM-EDA were investigated.

10.2 FUTURE DIRECTIONS

This thesis has contributed to two research areas: QiSNN is a novel method in the field of neural information processing, while the three developed optimisation algorithms belong to the field of evolutionary computation. The developed methods reported promising results and future work is planned to further investigate their characteristics and compare their performance to related methods. Additionally, a number of potential applications are possible.

In the next sections, future work on QiSNN is presented, followed by plans for future work on vQEA, cHM-EDA and hHM-EDA.

10.2.1 *QiSNN*

Here we discuss possible future directions for the QiSNN framework.

Analysis

A detailed analysis of the characteristics of QiSNN in real-world scenarios is suggested. It was already shown in the course of this thesis that the involved optimisation methods are able to handle fitness noise satisfyingly well. However, another important property of real-world data is the *imbalance of data sets*. Experimental analysis is required to investigate the behaviour of QiSNN in such a context. Furthermore, some traditional classification benchmarks based on real-world data sets may be investigated in a future study.

Applications

The QiSNN framework was designed for a straightforward application in different problem domains. The thesis has undertaken already a real-world case study in the context of an ecological modelling problem. Future studies on similar data are planned. Additional *ecological data* may be provided by the Centre for Bioprotection at Lincoln University, Christchurch, New Zealand. Potential other applications may

involve the analysis of *bio-medical problems*, such as gene expression analysis and the prediction of diseases based on clinical data.

For many data sets, the feature space is very large and involves the optimisation of thousands of different features. In this situation the need for more powerful computational resources arises. The implementation of a *distributed version of QiSNN* is required which allows the efficient simulation of SNN on a computer cluster. Another possibility is the use of state-of-the-art computer systems, such as the *BlueFern supercomputer*[1] that was made available to New Zealand scientists recently. This machine is currently the only IBM Blue Gene computer installed in the Southern Hemisphere. The system employs 1024 dual-core CPUs per rack resulting in a total of 4096 cores and a theoretical peak performance of 11.2 Tera-flops. Comprehensive large-scale simulations of SNN may provide an invaluable tool for the future analysis of brain-like neural information processing. Applying supercomputers to neuroscientific problems is a current trend, *cf. e.g.* the Blue Brain Project (Markram, 2006) which is an attempt to reverse engineer a mammalian brain through extensive computer simulations.

Neural models

Recently, numerous studies have suggested a novel paradigm for developing more realistic neural models. In (Kasabov, 2008) the potential of a *probabilistic* spiking neural model was discussed. The principle was further elaborated in (Kasabov, 2010). In these articles it is argued that most current neural models are deterministic which is in contrast to biological neurons. Stochastic elements may enhance the information processing capabilities of spiking neurons. The integration of such a novel neural model into the QiSNN framework would be straightforward since only the deterministic Thorpe neuron would need to be replaced. Due to its relevance for QiSNN, the concept of this probabilistic approach is briefly outlined here.

The probabilistic neural model as presented in (Kasabov, 2010) is schematically shown in Figure 10.1. The potential u_i represents the state of a neuron i. A neuron i is stimulated by the spike activity of pre-synaptic neurons j. Additional to the synaptic connection weights $w_{j,i}$, the probabilistic model has three novel parameters. Parameter $p^c_{j,i}$ represents the probability that a spike from neuron j will reach i, $p^s_{j,i}$ is the probability that synapse (j,i) contributes to potential u_i, and p_i is the probability that neuron i emits an output spike. The overall potential u_i of the neuron can be

1 More information about BlueFern is available at http://www.bluefern.canterbury.ac.nz

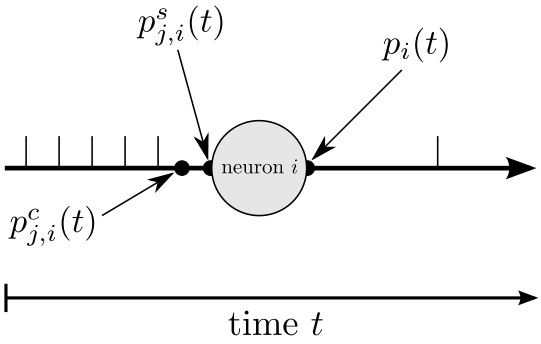


Figure 10.1: A probabilistic spiking neural model according to (Kasabov, 2010). $p^c_{j,i}$ represents the probability that a spike from neuron j will reach i, $p^s_{j,i}$ is the probability that synapse (j, i) contributes to potential u_i, and p_i is the probability that neuron i emits an output spike. See also Equation 10.1. Figure was redrawn from (Kasabov, 2010).

described *e.g.* by means of the spike response model (Gerstner & Kistler, 2002b) which was described in detail in chapter 2:

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_f \overbrace{C \times S}^{\text{Random variables}} \times \epsilon_{ij}(t - t_j^{(f)}) \tag{10.1}$$

where $C = 1$ with probability $p^c_{j,i}$, $S = 1$ with probability $p^s_{j,i}$ and p_i is sampled in dependence of the time t and the state of potential u. The kernel functions η and ϵ follow the interpretation of the spike response model, *cf.* chapter 2 for details. Note that, if all probabilities are set to 1, the model resembles the traditional spike response model.

Kasabov (2010) argues that an integrated probabilistic SNN similar to QiSNN may be very suitable for classification and feature selection problems. Especially in the context of many practical real-world problems involving large amounts of noise, a non-deterministic neuron may demonstrate some interesting characteristics.

Optimisation of additional parameters

Although most parameters in the QiSNN framework are optimised through the evolutionary process of hHM-EDA, additional parameters may be included in this optimisation, *e.g.* the parameters of the *neural encoding method*. The employed rank-order population encoding (explained in chapter 2, section 2.6.1) requires the setting of the number of Gaussian receptive fields and the parameter β which controls the standard

deviation of the Gaussian. Both parameters are critical for the functioning of QiSNN. Furthermore, other neural encoding techniques may be considered.

10.2.2 *Optimisation algorithms*

This thesis has developed three novel optimisation methods that allow the simultaneous evolution of a suitable feature subset and a corresponding parameter set for eSNN. However, due to their generic nature, these methods are also applicable to a variety of general optimisation problems. Several future directions for each of the optimisation techniques are summarised here.

hHM-EDA

Evolutionary algorithms are a powerful optimisation tool and many studies have employed them to enhance neural information processing. In the context of neuroscience these algorithms may be used to *reverse engineer* biological neurons. The technique allows the derivation of novel mathematical neural models whose parameters are adjusted through the use of an evolutionary algorithm in order to fit the model behaviour to some measured biological recordings. Similar approaches are very common in bioinformatical problems such as the reverse engineering of gene regulatory networks. Due to its flexibility, robustness and competitive performance, hHM-EDA seems very suitable for this task.

A concrete example for an application of hHM-EDA is the optimisation of the computational neuro-genetic modelling (CNGM) presented in (Benuskova & Kasabov, 2007). Here, a gene regulatory network (GRN) affects the spike activity of a SNN. Both the GRN and the SNN have parameters that need to be optimised in order to fit the CNGM to a given data set. Benuskova, Jain, Wysoski, and Kasabov (2006) presented a manually fine-tuned CNGM that is capable of reproducing experimental data on long-term potentiation (LTP) occurring in the rat hippocampal dentate gyros. Using hHM-EDA the optimisation process could be automated and the model accuracy increased.

Although the idea of a mixed variable optimisation algorithm is not new, hHM-EDA has only very few competitors at the moment. However, very recent developments in this area have proposed numerous interesting heterogeneous optimisation methods. See *e.g.* the work on genetic algorithms presented in (Rivero et al., 2009) and on a modified version of the Particle Swarm Optimiser (Garro et al., 2009), but also the study on the mixed Ant Colony Optimiser discussed in (Socha, 2004). In or-

der to compare these method and hHM-EDA efficiently with each other, novel mixed-variable test functions and benchmarks are required. A *heterogeneous benchmark suite* with standardised guidelines for presenting results similar to the one proposed in (Suganthan et al., 2005) appears highly suitable for such a comparison study. The suite may include practical real-world scenarios, such as the wrapper-based feature selection or the topology and parameter evolution of neural networks. The comparison of different algorithms could also motivate an international competition on heterogeneous optimisation problems.

The introduced generalised cooperative co-evolutionary architecture (CCA) (Potter & Jong, 2000) offers an interesting scheme for hybridising different evolutionary algorithms. Although it was shown in chapter 7 that the proposed hHM-EDA is not a CGA, cHM-EDA and vQEA could be hybridised following the CGA scheme. This *cooperative co-evolutionary hHM-EDA* might demonstrate interesting properties and should be experimentally compared to the hHM-EDA developed in this thesis.

cHM-EDA

The proposed cHM-EDA has reported very promising results on the CEC'05 benchmark suite. Its performance is generally on par with many highly advanced optimisation algorithms in the field and outperforms numerous methods especially in the context of difficult multi-modal problems. Future improvements include the development of *adaptive learning rates*. Many state-of-the-art algorithms implement such a mechanism, *e.g.* the CMA-ES presented in (Auger & Hansen, 2005). Self-adapting parameters represent a highly desired property of evolutionary algorithms.

Local restart strategies may further improve the optimisation performance of cHM-EDA, especially in the context of large-scale global optimisation problems. Whenever the algorithm is converged to a certain solution or the fitness improves only slowly, the algorithm is reinitialised and starts the search in a different region of the search space. This strategy is efficiently implemented *e.g.* in the Hybrid Real-coded Genetic Algorithm (García-Martínez & Lozano, 2005) and the CMA-ES (Auger & Hansen, 2005). The recently developed benchmark suite presented in (Tang et al., 2007) was specifically designed to compare the properties and performance of different optimisation methods in a high-dimensional optimisation scenario. Since the suite was proposed as part of the special session at the Congress on Evolutionary Computation (CEC) in 2007, an annual major event for the research field, many algorithms have been compared on these test functions. Thus, the analysis of cHM-EDA

based on this benchmark suite allows the inter-comparison of numerous large-scale optimisation methods to cHM-EDA.

The probabilistic multi-model approach may be also interesting for *multi-objective optimisation*. Since each model evolves individually, different areas in the search space are explored simultaneously. Experimental evidence was provided in chapter 6. A mechanism that allows the individual models to merge and split according to already visited solutions, might result in a powerful optimisation tool to solve multi-objective problems. The suitability of EDAs for this problem class was previously discussed in (Lozano, Larrañaga, Inza, & Bengoetxea, 2006).

vQEA

The proposed vQEA was extensively studied in this thesis. Future analysis may further focus on the importance of *different population structures*. It was demonstrated in chapter 5 that the population structure is very important in the context of a noisy and inaccurate fitness evaluation. An adaptive mechanism that automatically adjusts the number of individuals in each group might be beneficial.

Furthermore, the possibility of *heterogeneous groups* should be explored. Here, each group maintains its own learning rate. Hence, different groups explore the search space with different learning speeds. The fast groups generally converge quickly, but allow the efficient identification of promising areas in the search space. The slower groups, on the other hand, maintain diversity in order to perform a local search in these promising areas. Such a mechanism would explicitly introduce different search strategies into vQEA. Similar strategies are also employed in other algorithms, *e.g.* the genetic algorithm presented in (García-Martínez & Lozano, 2005). In principle, this strategy would be suitable for cHM-EDA and hHM-EDA as well.

10.3 CONCLUDING REMARKS

The thesis has embraced two major areas in the field of computational intelligence – the area of neural information processing and the area of evolutionary computation. For both areas, some very recent and exciting directions were explored. Arguably, the development of practical applications based on spiking neurons is currently a hot topic in the research community and numerous specialised conferences, workshops and journals have emerged recently. In light of this trend, the proposed QiSNN framework contributes to the family of contemporary evolving connectionist systems. The integrated feature and parameter optimisation significantly improves the eSNN clas-

sification capabilities and promotes the intuitive and straightforward application of the method in other problem domains.

Evolutionary computation is a traditional companion of neural information processing and both fields have greatly benefited from each other. Among the most advanced and current evolutionary algorithms are the probabilistic approaches, namely EDAs, which have attracted a large and highly productive research community during the last decade. The three novel EDAs developed in the course of the thesis employ multiple probabilistic models to explore the search space which adds an interesting new flavor to the EDA paradigm. The methods represent an original contribution to the field and very promising results have been obtained from an extensive experimental analysis.

The fruitful hybridisation of spiking neurons with evolutionary algorithms in either engineering or neuroscientific applications is very exciting and provides many interesting future directions for research.

Appendix A

FORMAL DESCRIPTION OF USED EDA

A number of classical EDA were implemented and investigated during the preparation of this study. The descriptions of these methods are given below in the format typically used in this field. Additionally, a formalised description of vQEA is presented.

Algorithm 5 vQEA – Versatile Quantum-inspired Evolutionary Algorithm

initialize each Q_i
initialize each A_i
while not termination condition **do**
  for all $i \in [1, p]$ **do**
    sample 1 new solution C_i from Q_i
    evaluate C_i
    if $f(A_i)$ **better than** $f(C_i)$ **then**
      learn_model $\left(A_i, C_i, Q_i\right)$
    end if
    $A_i \leftarrow C_i$
  end for
  check local and global synchronization
end while

function learn_model $\left(A_i, C_i, Q_i\right)$
for all $j \in [1, N]$ **do**
  if $A_i^j \neq C_i^j$ **then**
    if $A_i^j = 1$ **then**
      $Q_i^j \leftarrow$ rotate Q_i^j towards A_i^j using $\Delta\theta$
    else
      $Q_i^j \leftarrow$ rotate Q_i^j towards A_i^j using $-\Delta\theta$
    end if
  end if
end for

Algorithm 6 PBIL – Probabilistic Incremental Learning

initialize the probabilistic model $\mathcal{P}$
while not termination condition **do**
  sample M new solutions from $\mathcal{P}$ into D
  evaluate the elements of D
  select $best$ from D
  for all $j \in [1, N]$ **do**
    $\mathcal{P}^j \leftarrow \mathcal{P}^j \times (1.0 - R_l) + best^j \times R_l$
    if $rand(0, 1] < R_m$ **then**
      $\mathcal{P}^j \leftarrow \mathcal{P}^j \times (1.0 - R_s) + rand(0.0 \textbf{ or } 1.0) \times R_s$
    end if
  end for
end while

Algorithm 7 cGA – Compact Genetic Algorithm

initialize the probabilistic model $\mathcal{P}$
while not termination condition **do**
  sample 2 new solutions from $\mathcal{P}$ into D
  evaluate the elements of D
  select $winner$ and $looser$ from D
  learn_model $\big(winner, looser, \mathcal{P}\big)$
end while

function learn_model $\big(winner, looser, \mathcal{P}\big)$
for all $j \in [1, N]$ **do**
  if $winner^j \neq looser^j$ **then**
    if $winner^j = 1$ **then**
      $\mathcal{P}^j \leftarrow \mathcal{P}^j + 1/n$
    else
      $\mathcal{P}^j \leftarrow \mathcal{P}^j - 1/n$
    end if
  end if
end for

Algorithm 8 UMDA – Uni-variate Marginal Distribution Algorithm

initialize the probabilistic model $\mathcal{P}$
while not termination condition **do**
  sample M new solutions from $\mathcal{P}$ into D
  evaluate the elements of D
  select $L = \alpha * M$ solutions from D into D_s
  for all $j \in [1, N]$ **do**
    $\mathcal{P}^j \leftarrow$ compute marginal frequency at locus i in D_s
  end for
end while

Appendix B

COMPLETE STATISTICS ON CEC'05 BENCHMARK

The CEC-2005 benchmark initiative proposed in (Suganthan et al., 2005) suggests specific guidelines for the presentation of results, which allows a direct comparison of different optimization techniques. The obtained results on the 25 benchmark functions were prepared following these requirements in detail. In the Tables B.1, B.2 and B.3 the objective function errors after 10^3, 10^4, 10^5, and $N \times 10^4$ FES are presented. Table B.4, B.5 and B.6 present the number of required FES to reach a given fixed accuracy level for all successfully solved functions, together with the success rate and the success performance as defined in (Suganthan et al., 2005). Figure B.1 presents the convergence graphs of the objective function errors. Table B.7 summarises the time complexity for the algorithm.

FES	Prob	1	2	3	4	5	6	7	8	9	10	11	12
	min	1.54e+3	3.04e+3	1.32e+7	6.26e+3	9.25e+3	7.00e+7	9.35e+1	2.05e+1	6.02e+1	8.49e+1	1.03e+1	2.25e+4
	7^{th}	3.47e+3	6.17e+3	3.80e+7	1.10e+4	1.04e+4	3.05e+8	1.40e+2	2.06e+1	7.83e+1	1.01e+2	1.16e+1	3.66e+4
	med	4.42e+3	7.22e+3	5.13e+7	1.25e+4	1.13e+4	5.48e+8	2.19e+2	2.07e+1	8.52e+1	1.14e+2	1.20e+1	4.92e+4
1e3	19^{th}	5.25e+3	9.67e+3	7.51e+7	1.43e+4	1.22e+4	7.49e+8	2.61e+2	2.08e+1	9.12e+1	1.25e+2	1.23e+1	6.39e+4
	max	7.05e+3	1.14e+4	9.29e+7	1.76e+4	1.38e+4	1.16e+9	3.49e+2	2.09e+1	1.01e+2	1.35e+2	1.28e+1	8.46e+4
	mean	4.35e+3	7.50e+3	5.41e+7	1.23e+4	1.14e+4	5.66e+8	2.12e+2	2.07e+1	8.32e+1	1.12e+2	1.18e+1	5.14e+4
	std	1.83e+3	2.88e+3	2.79e+7	3.75e+3	1.53e+3	3.74e+8	8.97e+1	1.39e–1	1.37e+1	1.78e+1	8.58e–1	2.15e+4
	min	1.69e–2	6.43e–1	1.22e+6	1.46e+2	9.19e+2	2.36e+4	5.29e–1	2.03e+1	2.50e+1	1.72e+1	1.04e+0	5.90e+0
	7^{th}	2.87e–2	1.29e+0	2.26e+6	2.45e+2	1.07e+3	4.42e+4	6.01e–1	2.04e+1	3.06e+1	2.98e+1	1.34e+0	1.70e+1
	med	3.70e–2	1.42e+0	2.60e+6	3.03e+2	1.17e+3	6.50e+4	7.02e–1	2.05e+1	3.43e+1	3.26e+1	1.43e+0	3.47e+1
1e4	19^{th}	4.23e–2	1.82e+0	3.29e+6	4.10e+2	1.26e+3	7.94e+4	7.38e–1	2.06e+1	3.51e+1	3.90e+1	1.49e+0	9.53e+1
	max	5.37e–2	2.76e+0	4.80e+6	5.14e+2	1.57e+3	1.38e+5	7.97e–1	2.06e+1	3.95e+1	4.06e+1	2.93e+0	7.43e+2
	mean	3.57e–2	1.59e+0	2.83e+6	3.24e+2	1.20e+3	7.00e+4	6.73e–1	2.05e+1	3.29e+1	3.18e+1	1.65e+0	1.79e+2
	std	1.24e–2	6.99e–1	1.19e+6	1.28e+2	2.18e+2	3.88e+4	9.62e–2	1.07e–1	4.85e+0	8.32e+0	6.59e–1	2.84e+2
	min	0.00e+0	0.00e+0	6.53e+3	1.00e–10	8.34e–8	1.24e–2	0.00e+0	2.02e+1	0.00e+0	0.00e+0	0.00e+0	2.29e–2
	7^{th}	0.00e+0	0.00e+0	2.83e+4	4.00e–10	1.35e–7	2.43e–2	0.00e+0	2.03e+1	9.95e–1	0.00e+0	0.00e+0	3.48e+0
	med	0.00e+0	0.00e+0	5.80e+4	5.00e–10	1.41e–7	3.48e–2	0.00e+0	2.03e+1	9.95e–1	9.95e–1	0.00e+0	1.52e+1
1e5	19^{th}	0.00e+0	0.00e+0	1.11e+5	7.00e–10	1.61e–7	5.80e–2	9.86e–3	2.04e+1	2.98e+0	9.95e–1	0.00e+0	4.11e+1
	max	0.00e+0	0.00e+0	2.02e+5	1.00e–9	1.90e–7	4.01e+0	1.72e–2	2.04e+1	4.97e+0	1.99e+0	1.50e+0	7.12e+2
	mean	0.00e+0	0.00e+0	8.12e+4	5.40e–10	1.42e–7	8.28e–1	5.42e–3	2.03e+1	1.99e+0	7.96e–1	3.00e–1	1.54e+2
	std	0.00e+0	0.00e+0	6.98e+4	3.01e–10	3.51e–8	1.59e+0	7.03e–3	8.08e–2	1.78e+0	7.45e–1	6.00e–1	2.79e+2

FES	Prob	13	14	15	16	17	18	19	20	21	22	23	24	25
	min	5.74e+0	4.01e+0	5.05e+2	3.22e+2	3.36e+2	1.16e+3	1.15e+3	1.15e+3	1.23e+3	1.01e+3	1.33e+3	1.20e+3	1.51e+3
	7^{th}	9.35e+0	4.21e+0	7.27e+2	3.77e+2	4.45e+2	1.18e+3	1.17e+3	1.17e+3	1.36e+3	1.10e+3	1.37e+3	1.33e+3	1.62e+3
	med	9.86e+0	4.27e+0	7.52e+2	4.33e+2	4.73e+2	1.21e+3	1.20e+3	1.20e+3	1.40e+3	1.13e+3	1.39e+3	1.37e+3	1.64e+3
1e3	19^{th}	1.10e+1	4.40e+0	8.35e+2	4.60e+2	5.05e+2	1.23e+3	1.23e+3	1.23e+3	1.41e+3	1.15e+3	1.40e+3	1.39e+3	1.68e+3
	max	1.34e+1	4.45e+0	9.42e+2	4.87e+2	5.86e+2	1.27e+3	1.27e+3	1.27e+3	1.44e+3	1.30e+3	1.41e+3	1.40e+3	1.75e+3
	mean	9.87e+0	4.27e+0	7.52e+2	4.16e+2	4.69e+2	1.21e+3	1.20e+3	1.20e+3	1.37e+3	1.14e+3	1.38e+3	1.34e+3	1.64e+3
	std	2.50e+0	1.55e–1	1.45e+2	5.92e+1	8.16e+1	3.96e+1	4.34e+1	4.34e+1	7.50e+1	9.56e+1	3.05e+1	7.37e+1	7.64e+1
	min	8.30e–1	1.92e+0	2.70e+2	1.52e+2	1.36e+2	8.17e+2	8.09e+2	8.21e+2	9.27e+2	7.77e+2	9.71e+2	2.15e+2	1.17e+3
	7^{th}	1.29e+0	2.50e+0	5.13e+2	1.60e+2	1.81e+2	9.83e+2	9.90e+2	9.85e+2	1.01e+3	7.86e+2	1.04e+3	2.26e+2	1.20e+3
	med	1.40e+0	2.73e+0	5.28e+2	1.71e+2	1.85e+2	9.97e+2	1.00e+3	1.00e+3	1.03e+3	7.89e+2	1.07e+3	2.31e+2	1.21e+3
1e4	19^{th}	1.58e+0	2.91e+0	5.45e+2	1.77e+2	1.96e+2	1.02e+3	1.02e+3	1.01e+3	1.05e+3	8.00e+2	1.09e+3	2.35e+2	1.24e+3
	max	1.91e+0	3.41e+0	5.63e+2	1.86e+2	2.10e+2	1.05e+3	1.03e+3	1.03e+3	1.11e+3	8.17e+2	1.21e+3	2.45e+2	1.27e+3
	mean	1.40e+0	2.69e+0	4.84e+2	1.69e+2	1.82e+2	9.73e+2	9.72e+2	9.71e+2	1.03e+3	7.94e+2	1.08e+3	2.31e+2	1.22e+3
	std	3.56e–1	4.88e–1	1.08e+2	1.20e+1	2.51e+1	8.09e+1	8.29e+1	7.65e+1	5.99e+1	1.36e+1	7.65e+1	1.01e+1	3.56e+1
	min	5.69e–1	7.09e–1	0.00e+0	8.44e+1	8.54e+1	8.00e+2	8.00e+2	8.00e+2	8.00e+2	7.30e+2	9.71e+2	2.00e+2	1.03e+3
	7^{th}	8.07e–1	1.92e+0	4.00e+2	9.06e+1	9.14e+1	9.40e+2	9.39e+2	9.40e+2	8.00e+2	7.36e+2	9.71e+2	2.00e+2	1.11e+3
	med	8.61e–1	2.24e+0	4.00e+2	9.14e+1	9.27e+1	9.48e+2	9.46e+2	9.47e+2	8.00e+2	7.44e+2	9.71e+2	2.00e+2	1.12e+3
1e5	19^{th}	1.01e+0	2.70e+0	4.00e+2	9.47e+1	9.49e+1	9.58e+2	9.58e+2	9.58e+2	8.00e+2	7.51e+2	9.71e+2	2.00e+2	1.14e+3
	max	1.75e+0	3.14e+0	4.00e+2	9.72e+1	9.86e+1	1.01e+3	1.01e+3	1.01e+3	8.00e+2	8.00e+2	1.20e+3	2.00e+2	1.17e+3
	mean	1.00e+0	2.14e+0	3.20e+2	9.17e+1	9.26e+1	9.31e+2	9.31e+2	9.31e+2	8.00e+2	7.52e+2	1.02e+3	2.00e+2	1.11e+3
	std	4.00e–1	8.28e–1	1.60e+2	4.33e+0	4.35e+0	7.00e+1	7.04e+1	7.04e+1	0.00e+0	2.50e+1	9.27e+1	0.00e+0	4.50e+1

Table B.1: Function error obtained after 10^3, 10^4 and 10^5 function evaluations (FES) on the 25 test problems in dimension $N = 10$. Given are the minimum, 7^{th}, median, 19^{th}, and maximum value from 25 runs, together with the mean and standard deviation.

FES	Prob	1	2	3	4	5	6	7	8	9	10	11	12
	min	4.03e+4	6.12e+4	4.23e+8	6.03e+4	2.81e+4	1.96e+10	3.14e+3	2.11e+1	4.04e+2	5.37e+2	4.27e+1	1.34e+6
	7^{th}	5.42e+4	8.49e+4	9.77e+8	1.02e+5	3.19e+4	3.33e+10	3.71e+3	2.12e+1	4.28e+2	6.56e+2	4.48e+1	1.58e+6
	med	5.92e+4	9.25e+4	1.18e+9	1.09e+5	3.37e+4	3.58e+10	4.05e+3	2.12e+1	4.43e+2	6.97e+2	4.57e+1	1.71e+6
1e3	19^{th}	6.54e+4	1.02e+5	1.35e+9	1.17e+5	3.58e+4	4.49e+10	4.75e+3	2.13e+1	4.49e+2	7.39e+2	4.69e+1	1.87e+6
	max	7.57e+4	1.16e+5	1.67e+9	1.54e+5	3.82e+4	4.95e+10	4.95e+3	2.13e+1	4.87e+2	7.62e+2	4.83e+1	1.99e+6
	mean	5.90e+4	9.15e+4	1.12e+9	1.09e+5	3.36e+4	3.66e+10	4.12e+3	2.12e+1	4.42e+2	6.78e+2	4.57e+1	1.70e+6
	std	1.18e+4	1.85e+4	4.17e+8	3.01e+4	3.44e+3	1.04e+10	6.68e+2	8.15e–2	2.74e+1	7.95e+1	1.91e+0	2.27e+5
	min	1.78e+3	1.62e+4	2.00e+8	5.22e+4	1.75e+4	2.15e+9	4.77e+1	2.10e+1	2.46e+2	3.14e+2	3.99e+1	1.86e+5
	7^{th}	2.56e+3	2.48e+4	3.49e+8	5.76e+4	1.96e+4	2.84e+9	5.42e+1	2.11e+1	2.68e+2	3.37e+2	4.20e+1	3.21e+5
	med	2.75e+3	2.69e+4	3.92e+8	6.71e+4	2.08e+4	3.24e+9	5.97e+1	2.11e+1	2.77e+2	3.45e+2	4.30e+1	3.31e+5
1e4	19^{th}	2.90e+3	2.93e+4	4.54e+8	6.89e+4	2.16e+4	4.19e+9	6.42e+1	2.11e+1	2.79e+2	3.50e+2	4.38e+1	3.60e+5
	max	3.04e+3	3.31e+4	5.58e+8	7.65e+4	2.30e+4	4.91e+9	7.04e+1	2.12e+1	2.88e+2	3.64e+2	4.51e+1	4.23e+5
	mean	2.61e+3	2.61e+4	3.90e+8	6.45e+4	2.05e+4	3.47e+9	5.92e+1	2.11e+1	2.71e+2	3.42e+2	4.28e+1	3.24e+5
	std	4.42e+2	5.63e+3	1.18e+8	8.57e+3	1.87e+3	9.75e+8	7.86e+0	7.03e–2	1.43e+1	1.66e+1	1.76e+0	7.76e+4
	min	3.00e–10	1.39e–5	6.01e+5	1.84e+2	3.49e+2	3.23e+2	1.51e–8	2.09e+1	4.98e+0	2.01e+0	1.30e–2	2.63e+2
	7^{th}	4.00e–10	2.85e–5	1.56e+6	2.64e+2	3.94e+2	4.51e+2	2.42e–8	2.10e+1	6.97e+0	4.99e+0	1.39e–2	1.10e+3
	med	4.00e–10	3.25e–5	1.70e+6	3.15e+2	4.23e+2	5.06e+2	2.59e–8	2.10e+1	6.97e+0	5.99e+0	1.45e–2	2.99e+3
1e5	19^{th}	4.00e–10	4.63e–5	2.08e+6	3.53e+2	4.57e+2	5.90e+2	3.03e–8	2.10e+1	8.96e+0	6.98e+0	1.51e–2	4.60e+3
	max	5.00e–10	6.36e–5	2.82e+6	4.53e+2	4.89e+2	9.63e+2	3.37e–8	2.11e+1	1.10e+1	1.10e+1	5.16e+0	1.51e+4
	mean	4.00e–10	3.69e–5	1.75e+6	3.14e+2	4.22e+2	5.67e+2	2.58e–8	2.10e+1	7.77e+0	6.19e+0	1.04e+0	4.82e+3
	std	6.32e–11	1.68e–5	7.23e+5	8.97e+1	4.88e+1	2.16e+2	6.32e–9	5.65e–2	2.03e+0	2.91e+0	2.06e+0	5.37e+3

FES	Prob	13	14	15	16	17	18	19	20	21	22	23	24	25
	min	1.23e+2	1.39e+1	1.05e+3	7.80e+2	8.58e+2	1.18e+3	1.18e+3	1.18e+3	1.37e+3	1.41e+3	1.37e+3	1.37e+3	1.69e+3
	7^{th}	2.21e+2	1.41e+1	1.11e+3	9.01e+2	1.07e+3	1.31e+3	1.31e+3	1.31e+3	1.44e+3	1.58e+3	1.44e+3	1.46e+3	1.76e+3
	med	2.96e+2	1.42e+1	1.17e+3	9.85e+2	1.11e+3	1.33e+3	1.32e+3	1.32e+3	1.45e+3	1.61e+3	1.45e+3	1.49e+3	1.77e+3
1e3	19^{th}	3.56e+2	1.43e+1	1.19e+3	1.03e+3	1.14e+3	1.34e+3	1.34e+3	1.34e+3	1.47e+3	1.68e+3	1.47e+3	1.49e+3	1.81e+3
	max	5.18e+2	1.44e+1	1.22e+3	1.10e+3	1.28e+3	1.38e+3	1.38e+3	1.38e+3	1.54e+3	1.72e+3	1.53e+3	1.52e+3	1.86e+3
	mean	3.03e+2	1.42e+1	1.15e+3	9.58e+2	1.09e+3	1.31e+3	1.31e+3	1.31e+3	1.45e+3	1.60e+3	1.45e+3	1.47e+3	1.78e+3
	std	1.33e+2	1.83e–1	6.22e+1	1.10e+2	1.36e+2	6.95e+1	6.81e+1	6.81e+1	5.45e+1	1.10e+2	5.12e+1	5.19e+1	5.71e+1
	min	1.93e+1	1.30e+1	8.39e+2	3.46e+2	3.82e+2	1.06e+3	1.07e+3	1.07e+3	1.20e+3	1.21e+3	1.21e+3	1.17e+3	1.34e+3
	7^{th}	2.15e+1	1.36e+1	8.81e+2	3.88e+2	4.40e+2	1.07e+3	1.08e+3	1.08e+3	1.23e+3	1.26e+3	1.23e+3	1.25e+3	1.35e+3
	med	2.30e+1	1.37e+1	8.88e+2	4.02e+2	4.66e+2	1.08e+3	1.08e+3	1.08e+3	1.23e+3	1.28e+3	1.24e+3	1.26e+3	1.35e+3
1e4	19^{th}	2.36e+1	1.38e+1	8.96e+2	4.17e+2	4.89e+2	1.09e+3	1.09e+3	1.09e+3	1.24e+3	1.29e+3	1.24e+3	1.27e+3	1.36e+3
	max	2.47e+1	1.39e+1	9.17e+2	4.58e+2	6.84e+2	1.11e+3	1.12e+3	1.11e+3	1.26e+3	1.32e+3	1.25e+3	1.29e+3	1.37e+3
	mean	2.24e+1	1.36e+1	8.84e+2	4.02e+2	4.92e+2	1.08e+3	1.09e+3	1.09e+3	1.23e+3	1.27e+3	1.24e+3	1.25e+3	1.35e+3
	std	1.88e+0	3.25e–1	2.58e+1	3.64e+1	1.02e+2	1.66e+1	1.70e+1	1.40e+1	1.95e+1	3.85e+1	1.29e+1	4.19e+1	1.08e+1
	min	2.33e+0	9.99e+0	3.04e+2	7.96e+0	1.20e+1	9.06e+2	9.06e+2	9.04e+2	5.00e+2	8.75e+2	5.34e+2	2.00e+2	2.00e+2
	7^{th}	2.81e+0	1.10e+1	3.05e+2	1.36e+1	1.98e+1	9.08e+2	9.07e+2	9.06e+2	5.00e+2	8.86e+2	5.34e+2	2.00e+2	2.00e+2
	med	3.13e+0	1.13e+1	3.05e+2	1.60e+1	2.34e+1	9.09e+2	9.08e+2	9.07e+2	5.00e+2	8.91e+2	5.34e+2	2.00e+2	2.00e+2
1e5	19^{th}	3.40e+0	1.18e+1	3.05e+2	1.82e+1	2.83e+1	9.09e+2	9.09e+2	9.09e+2	5.00e+2	8.94e+2	5.34e+2	2.00e+2	2.00e+2
	max	4.30e+0	1.26e+1	3.05e+2	2.55e+1	5.42e+2	9.10e+2	9.10e+2	9.09e+2	5.00e+2	9.00e+2	5.34e+2	2.00e+2	2.00e+2
	mean	3.19e+0	1.13e+1	3.05e+2	1.63e+1	1.25e+2	9.08e+2	9.08e+2	9.07e+2	5.00e+2	8.89e+2	5.34e+2	2.00e+2	2.00e+2
	std	6.57e–1	8.62e–1	3.87e–1	5.76e+0	2.09e+2	1.51e+0	1.57e+0	1.87e+0	7.21e–5	8.57e+0	2.87e–4	1.31e–4	8.39e–5

Table B.2: Function error on the test problems for dimension $N = 30$. See description given in Table B.1 for detailed description of the table content.

FES	Prob	1	2	3	4	5	6	7	8	9	10	11	12
	min	1.06e+5	1.81e+5	1.81e+9	2.08e+5	3.71e+4	6.06e+10	6.80e+3	2.12e+1	7.39e+2	1.19e+3	7.57e+1	5.81e+6
	7^{th}	1.33e+5	2.18e+5	3.87e+9	2.95e+5	4.41e+4	8.34e+10	7.97e+3	2.13e+1	8.63e+2	1.35e+3	8.01e+1	7.15e+6
	med	1.43e+5	2.66e+5	4.30e+9	3.37e+5	4.52e+4	9.44e+10	8.36e+3	2.13e+1	8.87e+2	1.43e+3	8.16e+1	7.56e+6
1e3	19^{th}	1.48e+5	2.96e+5	5.11e+9	3.60e+5	4.66e+4	9.93e+10	8.79e+3	2.14e+1	9.02e+2	1.48e+3	8.22e+1	7.81e+6
	max	1.58e+5	3.47e+5	5.89e+9	4.16e+5	4.97e+4	1.27e+11	1.00e+4	2.14e+1	9.23e+2	1.55e+3	8.48e+1	8.69e+6
	mean	1.37e+5	2.61e+5	4.20e+9	3.23e+5	4.46e+4	9.29e+10	8.39e+3	2.13e+1	8.63e+2	1.40e+3	8.09e+1	7.41e+6
	std	1.78e+4	5.79e+4	1.38e+9	6.98e+4	4.17e+3	2.15e+10	1.05e+3	5.99e–2	6.48e+1	1.23e+2	3.01e+0	9.43e+5
	min	2.37e+4	1.00e+5	1.32e+9	1.60e+5	3.10e+4	2.60e+10	3.87e+2	2.12e+1	6.20e+2	7.99e+2	7.28e+1	2.17e+6
	7^{th}	2.64e+4	1.17e+5	1.89e+9	1.82e+5	3.58e+4	3.23e+10	4.73e+2	2.12e+1	6.49e+2	8.99e+2	7.76e+1	2.92e+6
	med	2.81e+4	1.25e+5	2.12e+9	1.94e+5	3.72e+4	3.59e+10	5.22e+2	2.13e+1	6.61e+2	9.19e+2	7.86e+1	3.26e+6
1e4	19^{th}	2.89e+4	1.36e+5	2.37e+9	2.14e+5	3.82e+4	4.09e+10	5.65e+2	2.13e+1	6.70e+2	9.41e+2	7.92e+1	3.36e+6
	max	3.14e+4	1.54e+5	2.77e+9	2.31e+5	3.96e+4	4.25e+10	6.31e+2	2.14e+1	6.84e+2	9.71e+2	8.09e+1	3.51e+6
	mean	2.77e+4	1.27e+5	2.09e+9	1.96e+5	3.63e+4	3.55e+10	5.16e+2	2.13e+1	6.57e+2	9.06e+2	7.78e+1	3.04e+6
	std	2.56e+3	1.80e+4	4.85e+8	2.49e+4	2.96e+3	5.98e+9	8.26e+1	6.53e–2	2.16e+1	5.85e+1	2.71e+0	4.77e+5
	min	1.81e–3	2.92e+0	2.78e+7	1.38e+4	4.81e+3	1.63e+6	1.39e–2	2.11e+1	2.44e+1	5.22e+1	1.31e+0	1.34e+3
	7^{th}	2.26e–3	5.87e+0	3.47e+7	1.99e+4	5.49e+3	2.89e+6	1.73e–2	2.12e+1	2.92e+1	5.64e+1	1.44e+0	7.35e+3
	med	2.38e–3	6.84e+0	3.93e+7	2.26e+4	5.80e+3	3.06e+6	1.84e–2	2.12e+1	3.27e+1	5.82e+1	1.49e+0	1.03e+4
1e5	19^{th}	2.57e–3	9.62e+0	4.28e+7	2.67e+4	6.26e+3	3.36e+6	2.00e–2	2.12e+1	3.35e+1	6.10e+1	1.58e+0	1.89e+4
	max	3.05e–3	1.40e+1	5.28e+7	3.50e+4	6.91e+3	3.86e+6	2.30e–2	2.12e+1	3.77e+1	6.77e+1	5.51e+0	2.40e+4
	mean	2.41e–3	7.85e+0	3.95e+7	2.36e+4	5.85e+3	2.96e+6	1.85e–2	2.12e+1	3.15e+1	5.91e+1	2.26e+0	1.24e+4
	std	4.03e–4	3.76e+0	8.34e+6	7.08e+3	7.07e+2	7.42e+5	3.00e–3	5.11e–2	4.46e+0	5.14e+0	1.62e+0	8.12e+3

FES	Prob	13	14	15	16	17	18	19	20	21	22	23	24	25
	min	7.75e+2	2.36e+1	1.16e+3	1.05e+3	1.10e+3	1.36e+3	1.36e+3	1.36e+3	1.47e+3	1.55e+3	1.51e+3	1.54e+3	1.88e+3
	7^{th}	1.07e+3	2.39e+1	1.24e+3	1.12e+3	1.21e+3	1.41e+3	1.42e+3	1.42e+3	1.53e+3	1.71e+3	1.53e+3	1.60e+3	1.95e+3
	med	1.36e+3	2.40e+1	1.29e+3	1.18e+3	1.25e+3	1.43e+3	1.43e+3	1.43e+3	1.54e+3	1.76e+3	1.55e+3	1.62e+3	1.97e+3
1e3	19^{th}	1.67e+3	2.41e+1	1.31e+3	1.21e+3	1.33e+3	1.44e+3	1.45e+3	1.45e+3	1.56e+3	1.80e+3	1.57e+3	1.63e+3	1.99e+3
	max	1.97e+3	2.42e+1	1.38e+3	1.26e+3	1.41e+3	1.46e+3	1.47e+3	1.47e+3	1.59e+3	1.86e+3	1.58e+3	1.66e+3	2.03e+3
	mean	1.37e+3	2.40e+1	1.28e+3	1.17e+3	1.26e+3	1.42e+3	1.43e+3	1.43e+3	1.54e+3	1.74e+3	1.55e+3	1.61e+3	1.96e+3
	std	4.22e+2	2.09e–1	7.30e+1	7.18e+1	1.06e+2	3.32e+1	3.78e+1	3.77e+1	3.85e+1	1.07e+2	2.48e+1	4.19e+1	5.08e+1
	min	7.29e+1	2.30e+1	8.79e+2	5.50e+2	6.21e+2	1.23e+3	1.24e+3	1.24e+3	1.35e+3	1.31e+3	1.34e+3	1.39e+3	1.46e+3
	7^{th}	8.31e+1	2.35e+1	1.05e+3	6.45e+2	7.68e+2	1.26e+3	1.25e+3	1.26e+3	1.36e+3	1.41e+3	1.36e+3	1.43e+3	1.47e+3
	med	8.76e+1	2.36e+1	1.08e+3	6.71e+2	8.10e+2	1.27e+3	1.27e+3	1.27e+3	1.37e+3	1.43e+3	1.37e+3	1.45e+3	1.48e+3
1e4	19^{th}	1.01e+2	2.37e+1	1.10e+3	6.99e+2	8.34e+2	1.28e+3	1.29e+3	1.28e+3	1.38e+3	1.46e+3	1.38e+3	1.45e+3	1.49e+3
	max	1.10e+2	2.38e+1	1.12e+3	7.34e+2	8.65e+2	1.29e+3	1.30e+3	1.30e+3	1.39e+3	1.49e+3	1.40e+3	1.46e+3	1.52e+3
	mean	9.10e+1	2.35e+1	1.04e+3	6.60e+2	7.80e+2	1.26e+3	1.27e+3	1.27e+3	1.37e+3	1.42e+3	1.37e+3	1.44e+3	1.48e+3
	std	1.33e+1	2.82e–1	8.61e+1	6.23e+1	8.56e+1	2.19e+1	2.44e+1	2.32e+1	1.52e+1	6.21e+1	1.96e+1	2.58e+1	2.09e+1
	min	4.28e+0	1.82e+1	2.53e+2	3.41e+1	5.75e+1	9.22e+2	9.23e+2	9.23e+2	5.01e+2	9.39e+2	5.39e+2	2.02e+2	2.01e+2
	7^{th}	5.01e+0	1.99e+1	3.53e+2	3.95e+1	6.65e+1	9.24e+2	9.25e+2	9.25e+2	5.01e+2	9.45e+2	5.40e+2	2.02e+2	2.01e+2
	med	5.60e+0	2.06e+1	3.56e+2	4.17e+1	7.02e+1	9.25e+2	9.26e+2	9.25e+2	5.01e+2	9.49e+2	5.40e+2	2.03e+2	2.01e+2
1e5	19^{th}	6.16e+0	2.13e+1	4.05e+2	4.45e+1	7.22e+1	9.26e+2	9.26e+2	9.26e+2	5.01e+2	9.52e+2	5.40e+2	2.03e+2	2.01e+2
	max	7.42e+0	2.21e+1	4.06e+2	5.63e+1	8.04e+1	9.29e+2	9.28e+2	9.27e+2	5.01e+2	9.54e+2	5.40e+2	2.03e+2	2.01e+2
	mean	5.69e+0	2.04e+1	3.55e+2	4.32e+1	6.93e+1	9.25e+2	9.26e+2	9.25e+2	5.01e+2	9.48e+2	5.40e+2	2.02e+2	2.01e+2
	std	1.07e+0	1.32e+0	5.58e+1	7.37e+0	7.47e+0	2.07e+0	1.74e+0	1.46e+0	1.49e–1	5.21e+0	3.17e–1	2.45e–1	2.30e–1

Table B.3: Function error on the test problems for dimension $N = 50$. See description given in Table B.1 for detailed description of the table content.

Prob	min	7^{th}	median	19^{th}	max	mean	std	Success Rate	Success Performance
1	$1.77e+03$	$1.81e+03$	$1.82e+03$	$1.83e+03$	$1.85e+03$	$1.82e+03$	$1.92e+01$	100	$1.82e+03$
2	$2.46e+03$	$2.51e+03$	$2.56e+03$	$2.58e+03$	$2.64e+03$	$2.55e+03$	$4.51e+01$	100	$2.55e+03$
4	$7.12e+03$	$7.41e+03$	$7.48e+03$	$7.51e+03$	$7.59e+03$	$7.45e+03$	$1.01e+02$	100	$7.45e+03$
5	$9.07e+03$	$9.15e+03$	$9.25e+03$	$9.27e+03$	$9.34e+03$	$9.22e+03$	$7.28e+01$	100	$9.22e+03$
7	$1.41e+03$	$1.44e+03$	$1.48e+03$	$1.79e+03$	-	$1.51e+03$	$1.33e+02$	84	$1.80e+03$
9	$2.32e+03$	$2.65e+03$	-	-	-	$2.48e+03$	$1.09e+02$	28	$8.87e+03$
10	$2.39e+03$	$2.63e+03$	-	-	-	$2.60e+03$	$9.27e+01$	40	$6.50e+03$
11	$2.44e+03$	$2.48e+03$	$2.50e+03$	$2.54e+03$	-	$2.50e+03$	$3.98e+01$	92	$2.72e+03$
15	$2.76e+03$	-	-	-	-	$2.76e+03$	$0.00e+00$	4	$6.90e+04$

Table B.4: The table presents the number of FES to achieve a fixed accuracy level for dimension $N = 10$. The results obtained in 25 runs were sorted according to required FES. Given are the minimum, 7^{th}, median, 19^{th}, and maximum number of FES from these runs, together with the mean and standard deviation, the success rate (in %) and success performance as specified in (Suganthan et al., 2005).

Prob	min	7^{th}	median	19^{th}	max	mean	std	Success Rate	Success Performance
1	$7.49e+03$	$7.61e+03$	$7.63e+03$	$7.66e+03$	$7.69e+03$	$7.63e+03$	$4.13e+01$	100	$7.63e+03$
2	$1.13e+04$	$1.14e+04$	$1.15e+04$	$1.16e+04$	$1.19e+04$	$1.15e+04$	$1.61e+02$	100	$1.15e+04$
7	$5.52e+03$	$5.61e+03$	$5.64e+03$	$5.67e+03$	$5.71e+03$	$5.63e+03$	$4.73e+01$	100	$5.63e+03$
11	$1.02e+04$	$1.03e+04$	$1.04e+04$	$1.04e+04$	-	$1.04e+04$	$6.07e+01$	92	$1.13e+04$

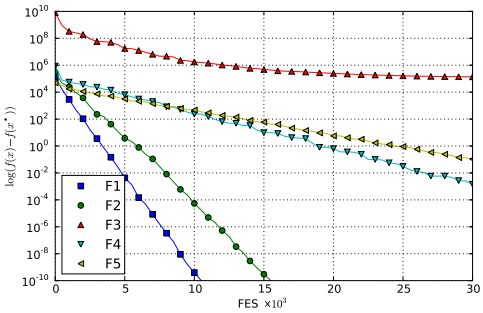
Table B.5: The table presents the number of FES to achieve a fixed accuracy level for dimension $N = 30$. See the caption of Table B.4 for detailed description of the table content.

Prob	min	7^{th}	median	19^{th}	max	mean	std	Success Rate	Success Performance
1	$1.42e+04$	$1.44e+04$	$1.44e+04$	$1.44e+04$	$1.44e+04$	$1.44e+04$	$5.37e+01$	100	$1.44e+04$
2	$2.19e+04$	$2.29e+04$	$2.36e+04$	$2.65e+04$	-	$2.36e+04$	$1.34e+03$	76	$3.11e+04$
7	$1.02e+04$	$1.03e+04$	$1.04e+04$	$1.04e+04$	$1.06e+04$	$1.04e+04$	$8.66e+01$	100	$1.04e+04$
11	$1.94e+04$	$1.97e+04$	$1.97e+04$	$1.98e+04$	-	$1.97e+04$	$9.69e+01$	76	$2.59e+04$

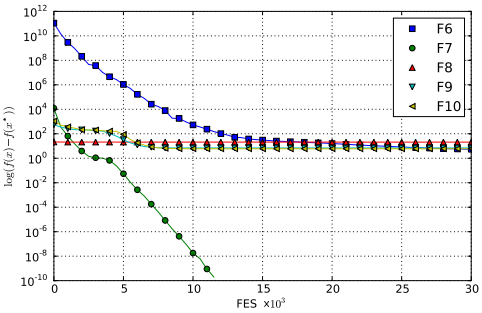
Table B.6: The table presents the number of FES to achieve a fixed accuracy level for dimension $N = 50$. See the caption of Table B.4 for detailed description of the table content.

	$T0$	$T1$	$\hat{T}2$	$(\hat{T}2 - T1)/T0$
$N = 10$		0.520	2.239	11.311
$N = 30$	0.152	1.976	4.217	14.742
$N = 50$		3.580	6.479	19.074

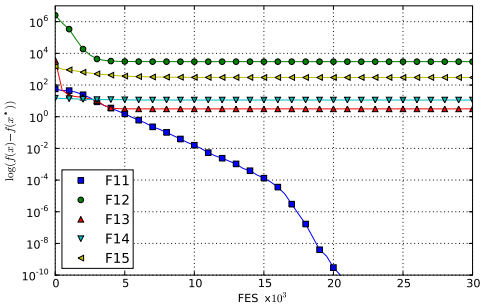
Table B.7: Measured CPU time in seconds according to (Suganthan et al., 2005) using Java 1.6, Ubuntu Linux 9.04 64bit, Intel Core2 Duo 3GHz, 4GB RAM.



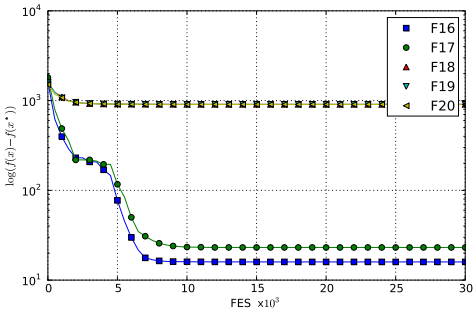
(a) Convergence Graphs for Problem 1-5



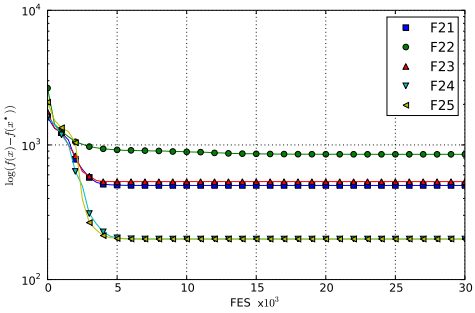
(b) Convergence Graphs for Problem 6-10



(c) Convergence Graphs for Problem 11-15



(d) Convergence Graphs for Problem 16-20



(e) Convergence Graphs for Problem 21-25

Figure B.1: The figure presents the evolution of the objective function error value as a function of the FES for 25 benchmark functions in dimension $N = 30$. Shown is the median value of 25 performed runs.

Appendix C

CONFIGURING FIRST-LEVEL EDAS FOR NOISY FITNESS FUNCTIONS

Maximum noise strength was set to $\sigma_m = \{0, 0.5, 1.5\}$ as the representatives for each noise type. In PBIL the population size $M \in \{10, 20, \ldots, 100\}$ and the learning rate $R_l = \{0.01, 0.05, 0.1, 0.15, 0.2, 0.25\}$ were varied. We set the mutation probability to $R_m = 0.02$ and the mutation shift to $R_s = R_l$, which is the default setting for this algorithm. The only parameter for cGA is the virtual population size n, which we set to $n = \{80, 90, \ldots, 350\}$. For UMDA we tested different values for the population size $M \in \{100, 200, \ldots, 1000\}$, while truncation selection with rate 50% was used. vQEA is a coarse-grained algorithm allowing a complex structure for the population of $\mathcal{Q}$individuals. Four structural settings were investigated: a single $\mathcal{Q}$individual ($\mathsf{vQEA}_{1,1}$), one group of ten fully synchronised $\mathcal{Q}$individuals ($\mathsf{vQEA}_{1,10}$), five groups of two $\mathcal{Q}$individuals synchronised every i-th generation ($\mathsf{vQEA}_{5,2}$) and ten groups of one $\mathcal{Q}$individual synchronised every j-th generation ($\mathsf{vQEA}_{10,1}$). For all vQEA configurations the learning rate has to be set, which we chose out of $\Delta\theta \in \{0.0025, 0.005, 0.01, 0.15, 0.02, 0.03\}$. Additionally the local and global synchronisation period has to be determined. For the local synchronisation period we used always $S_{loc} = 1$, which is the default setting and has been identified to work best. The global synchronisation period was varied according to $S_{glob} \in \{5, 10, 25, 50, 75, 100, 150, 200, 300\}$.

Each setting of all the algorithms was applied to each benchmark and each noise landscape. To satisfy statistical requirements 30 runs were performed for each configuration. Table C.1 presents the identified optimal settings found.

OneMax				
	constant	linear	inverse linear	cosine
PBIL	$M = 10$ $R_l = R_s = 0.01$	$M = 100$ $R_l = R_s = 0.05$	$M = 10$ $R_l = R_s = 0.25$	$M = 10$ $R_l = R_s = 0.01$
cGA	$n = 200$	$n = 250$	$n = 200$	$n = 190$
UMDA	$M = 500$	$M = 500$	$M = 500$	$M = 500$
vQEA$_{1,10}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.03\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{1,1}$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{10,1}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 10$	$\Delta\theta = 0.005\pi$ $S_{glob} = 5$	$\Delta\theta = 0.02\pi$ $S_{glob} = 25$	$\Delta\theta = 0.01\pi$ $S_{glob} = 25$
vQEA$_{5,2}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 25$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 50$	$\Delta\theta = 0.005\pi$ $S_{glob} = 25$
NK-landscapes, $K = 0$				
	constant	linear	inverse linear	cosine
PBIL	$M = 10$ $R_l = R_s = 0.01$	$M = 100$ $R_l = R_s = 0.05$	$M = 50$ $R_l = R_s = 0.25$	$M = 10$ $R_l = R_s = 0.25$
cGA	$n = 190$	$n = 190$	$n = 190$	$n = 190$
UMDA	$M = 500$	$M = 500$	$M = 400$	$M = 500$
vQEA$_{1,10}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.01\pi$ $S_{glob} = 1$	$\Delta\theta = 0.01\pi$ $S_{glob} = 1$
vQEA$_{1,1}$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{10,1}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 100$	$\Delta\theta = 0.005\pi$ $S_{glob} = 300$
vQEA$_{5,2}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 75$	$\Delta\theta = 0.03\pi$ $S_{glob} = 300$
NK-landscapes, $K = 4$				
	constant	linear	inverse linear	cosine
PBIL	$M = 50$ $R_l = R_s = 0.05$	$M = 70$ $R_l = R_s = 0.05$	$M = 10$ $R_l = R_s = 0.01$	$M = 10$ $R_l = R_s = 0.01$
cGA	$n = 220$	$n = 180$	$n = 160$	$n = 180$
UMDA	$M = 300$	$M = 500$	$M = 500$	$M = 500$
vQEA$_{1,10}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 1$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.02\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{1,1}$	$\Delta\theta = 0.005\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 1$
vQEA$_{10,1}$	$\Delta\theta = 0.01\pi$ $S_{glob} = 50$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.03\pi$ $S_{glob} = 50$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 10$
vQEA$_{5,2}$	$\Delta\theta = 0.03\pi$ $S_{glob} = 10$	$\Delta\theta = 0.01\pi$ $S_{glob} = 10$	$\Delta\theta = 0.02\pi$ $S_{glob} = 50$	$\Delta\theta = 0.0025\pi$ $S_{glob} = 10$

Table C.1: Parameter settings

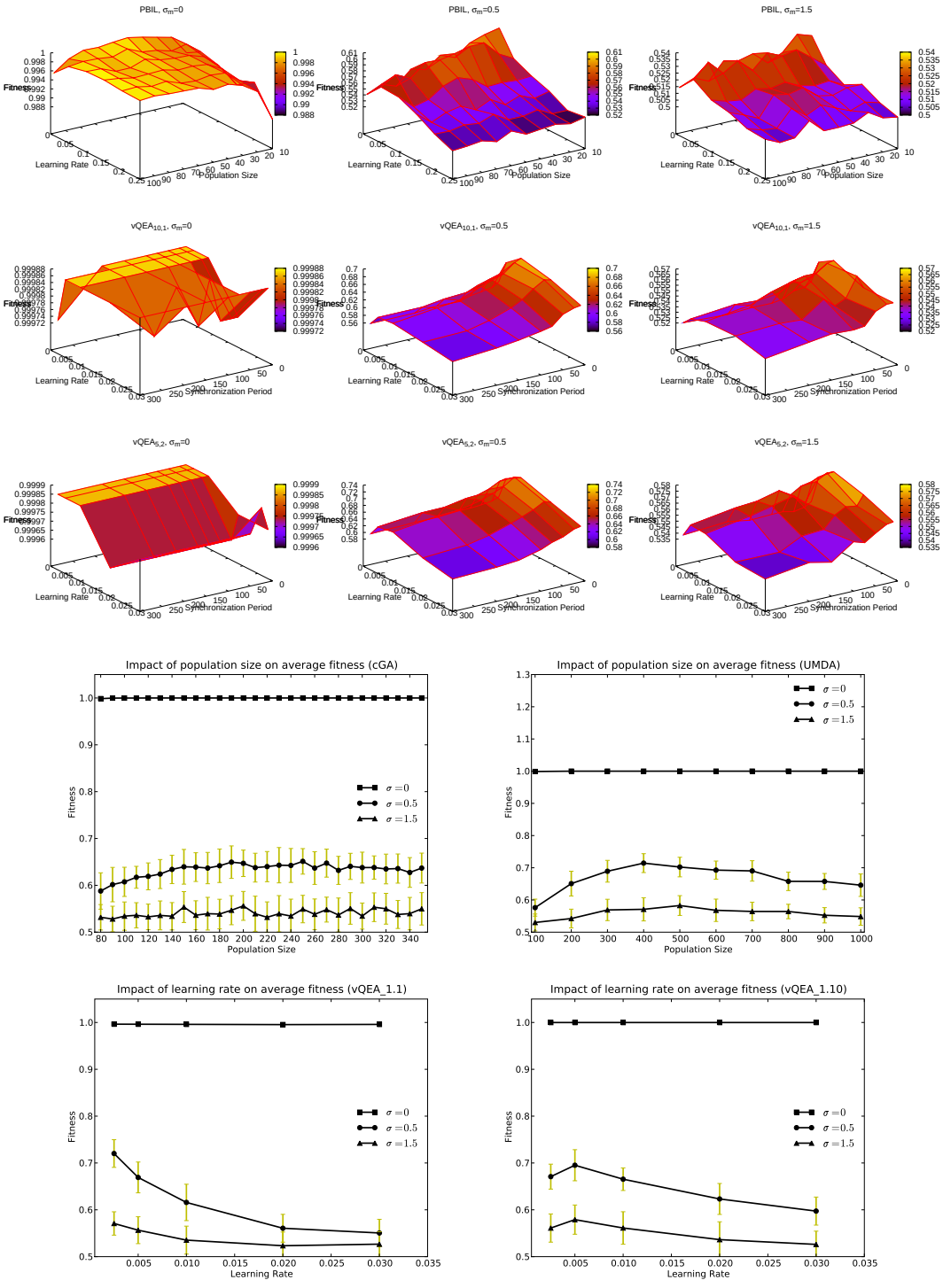


Figure C.1: Results on the OneMax problem with constant noise

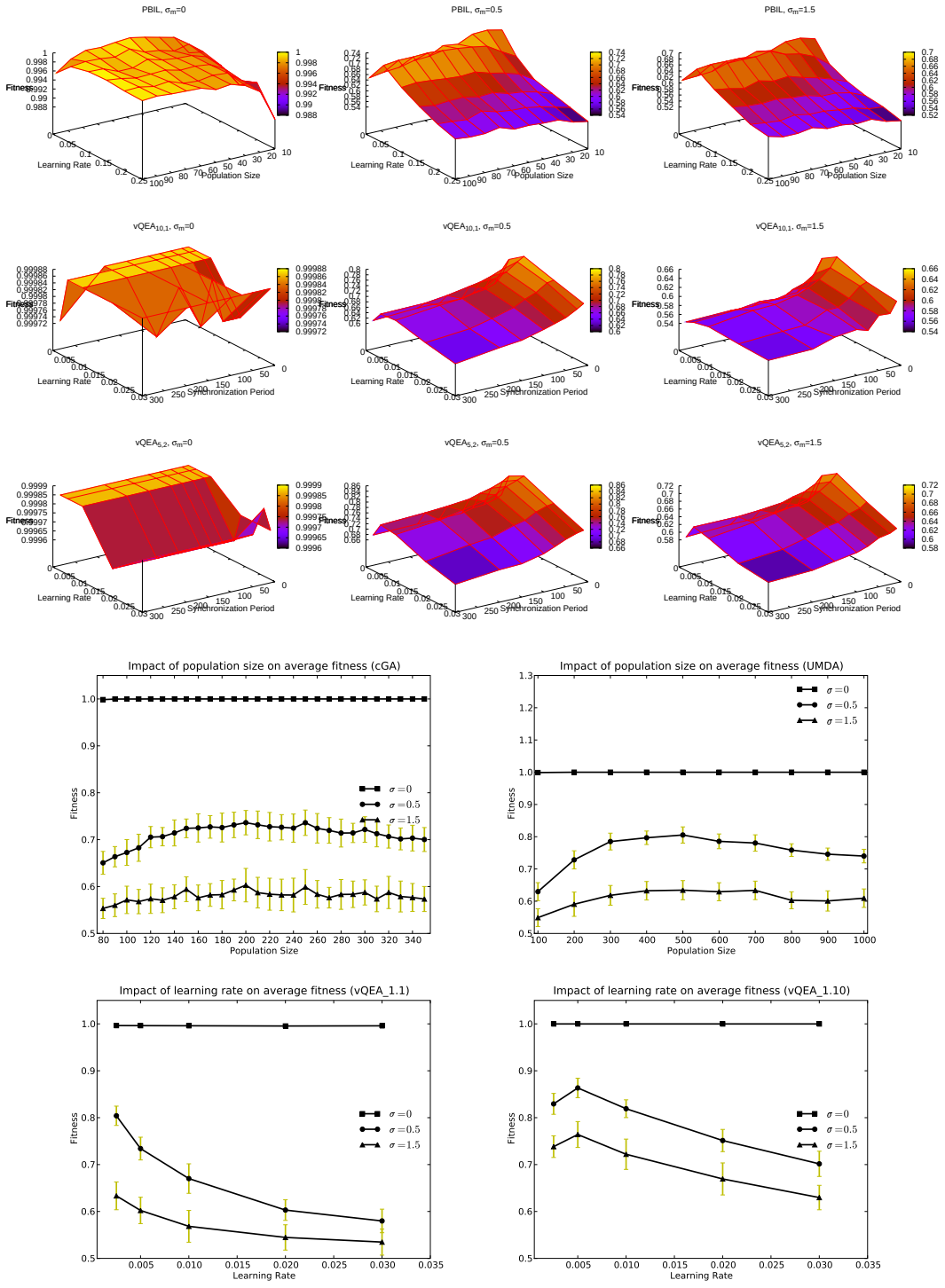


Figure C.2: Results on the OneMax problem with linear noise

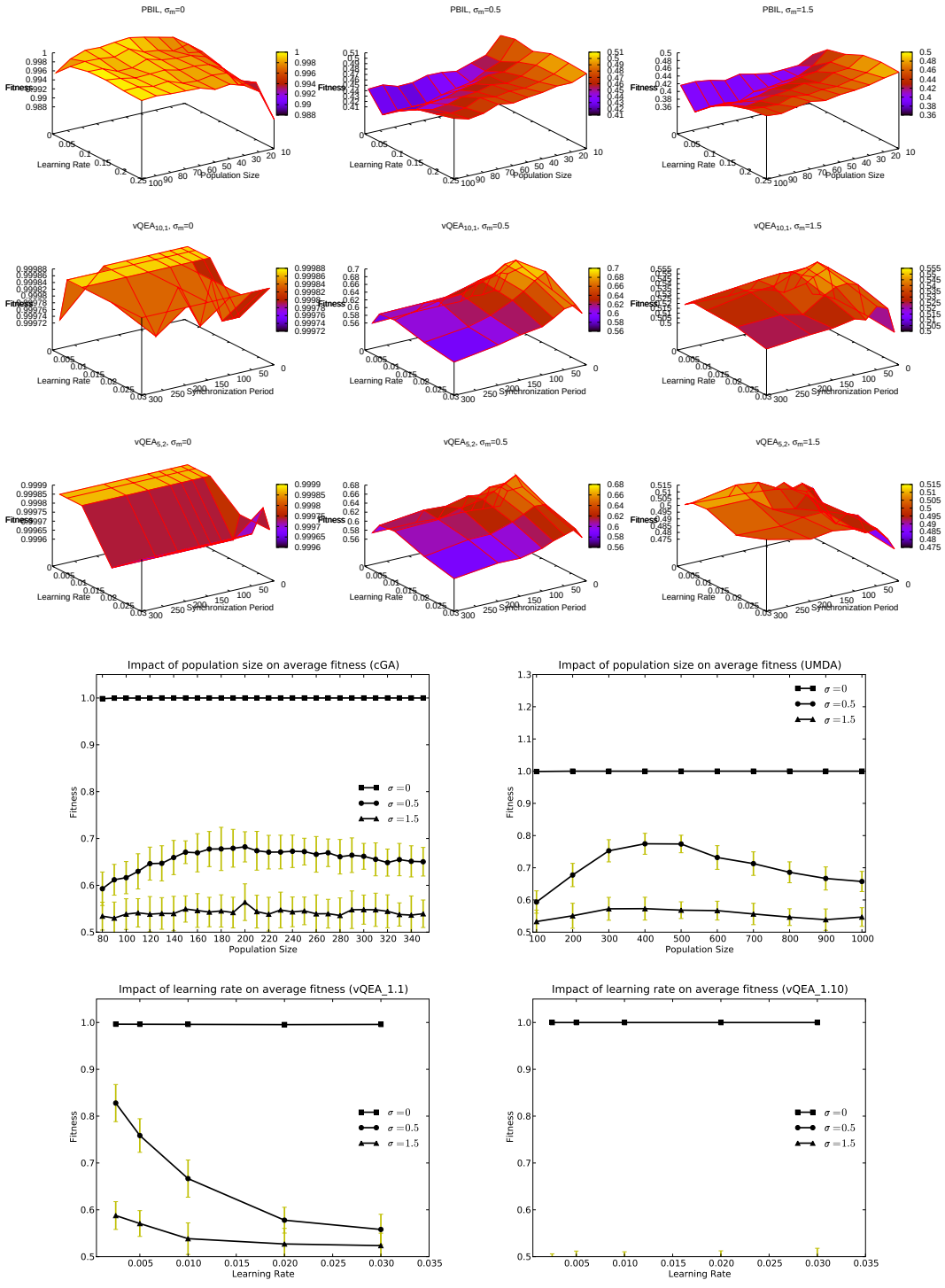


Figure C.3: Results on the OneMax problem with inverse linear noise

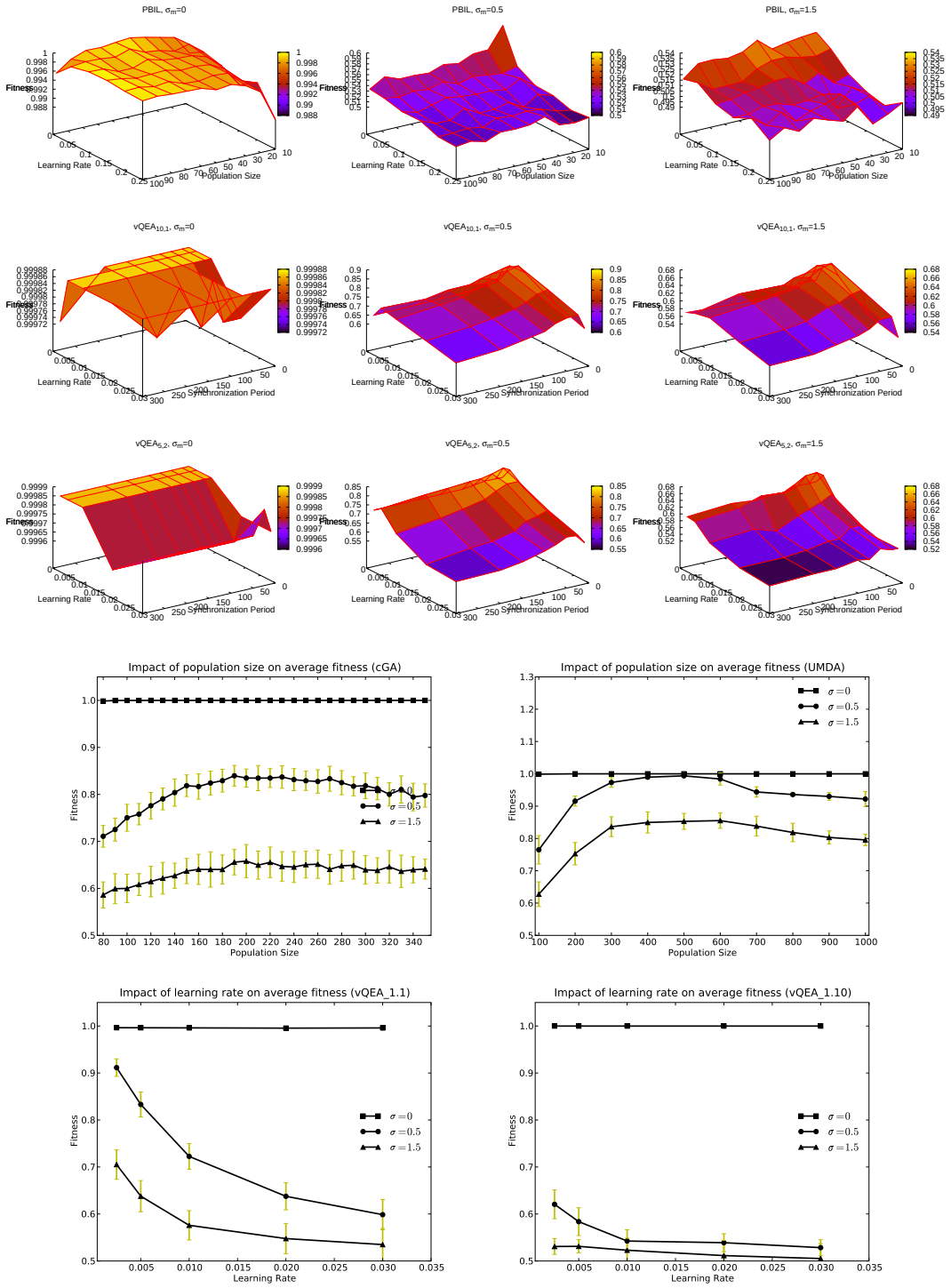


Figure C.4: Results on the OneMax problem with cosine noise

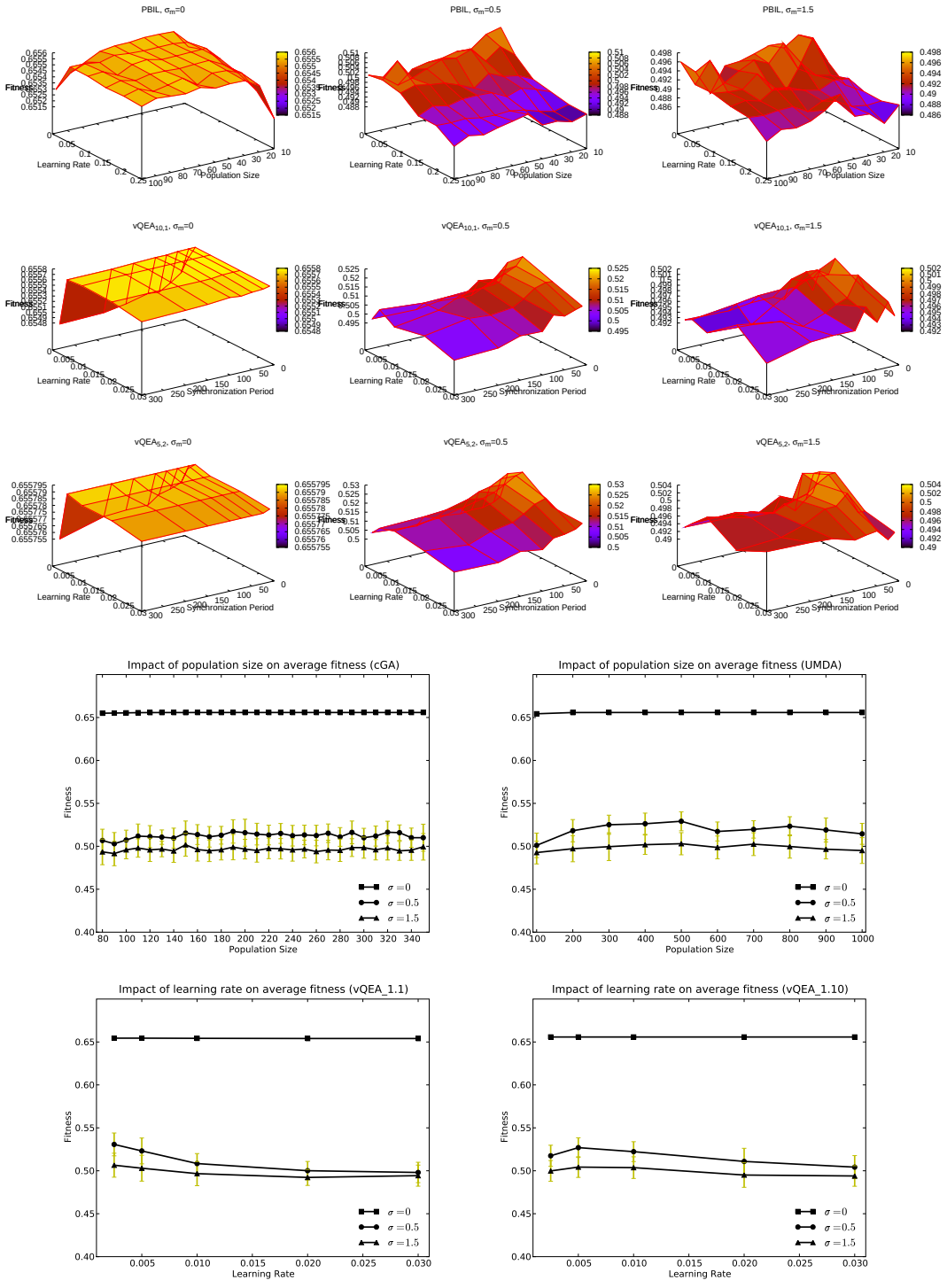


Figure C.5: Results on the K=0 problem with constant noise

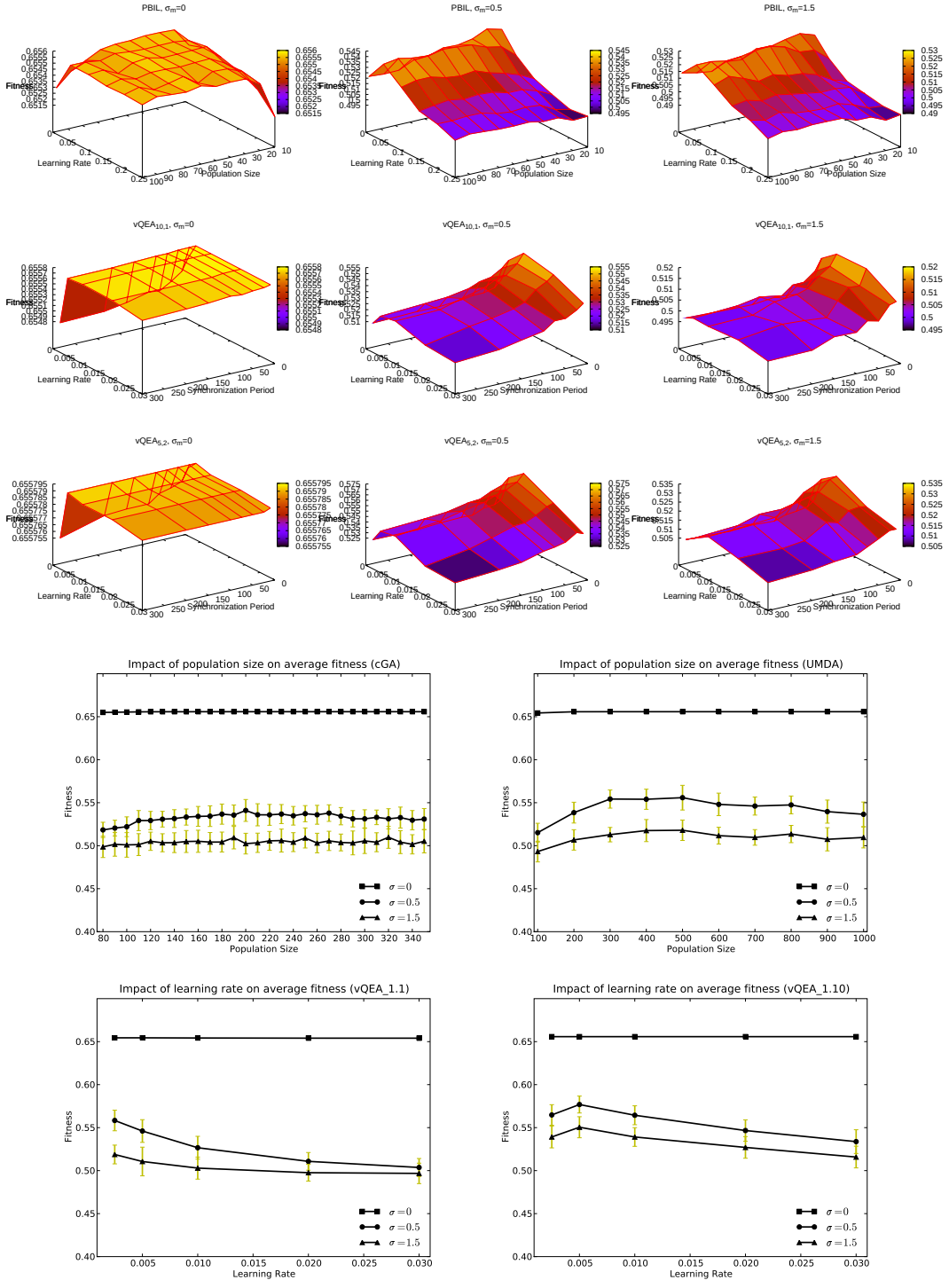


Figure C.6: Results on the K=0 problem with linear noise

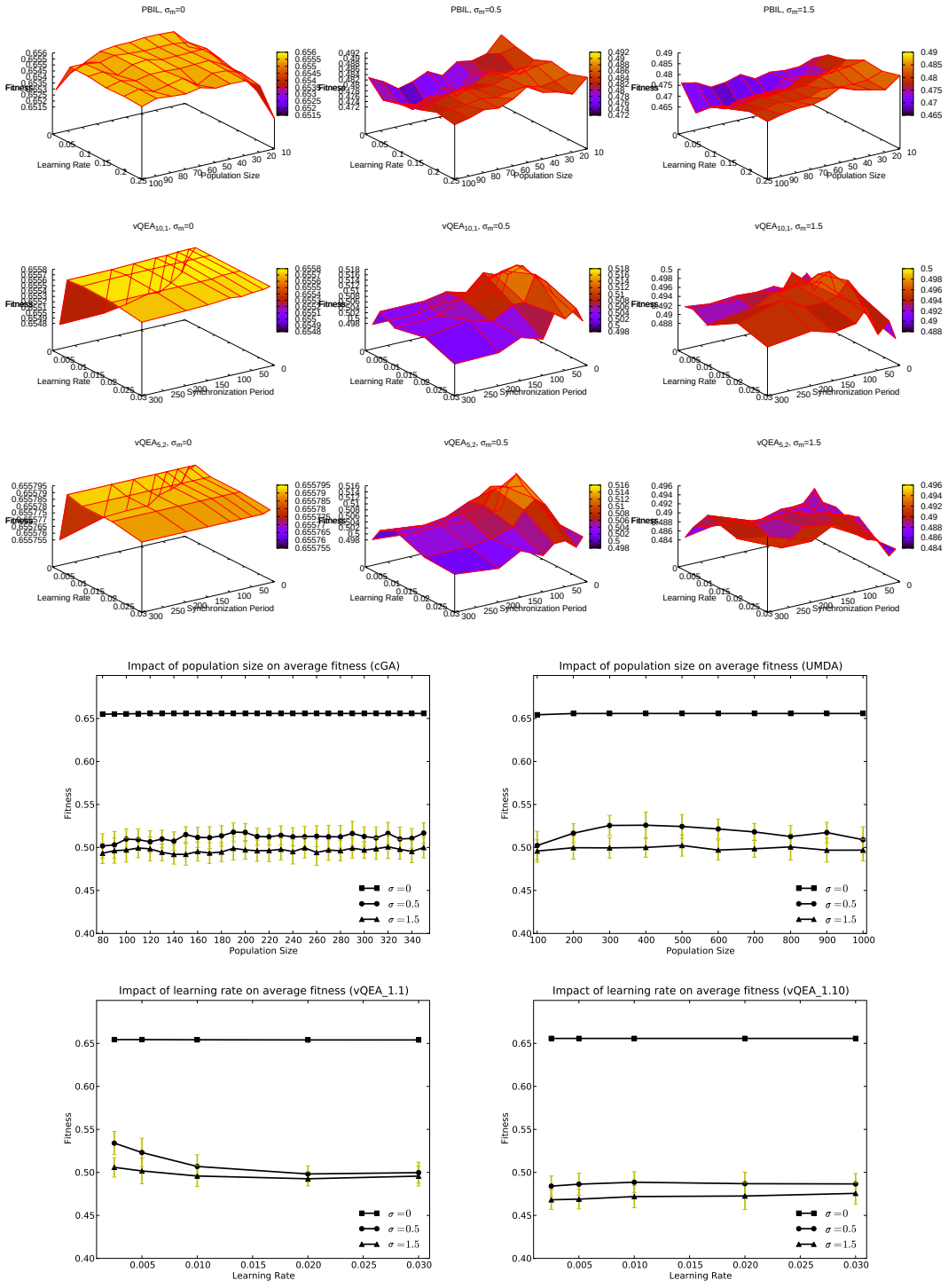


Figure C.7: Results on the K=0 problem with inverse linear noise

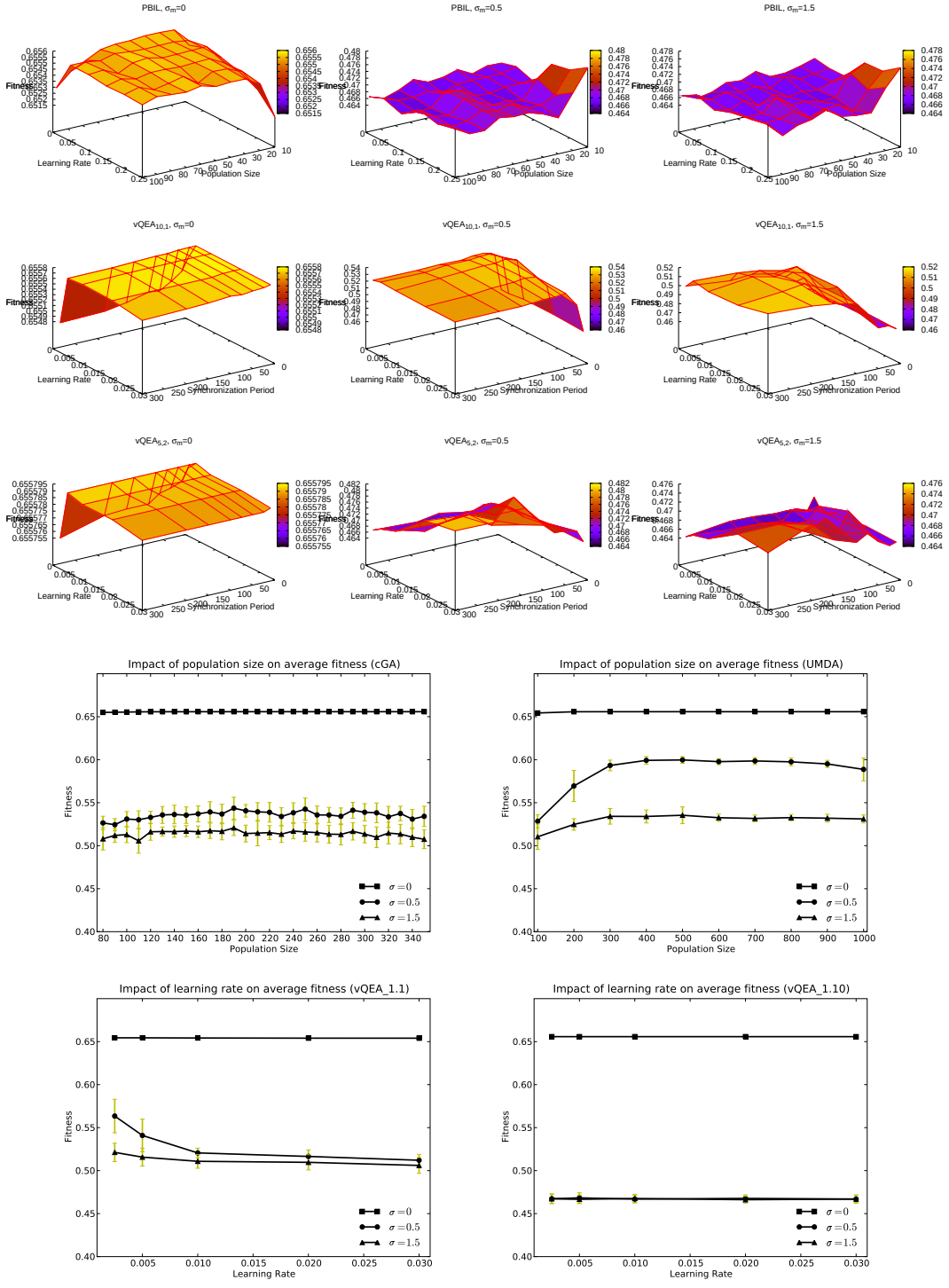


Figure C.8: Results on the K=0 problem with cosine noise

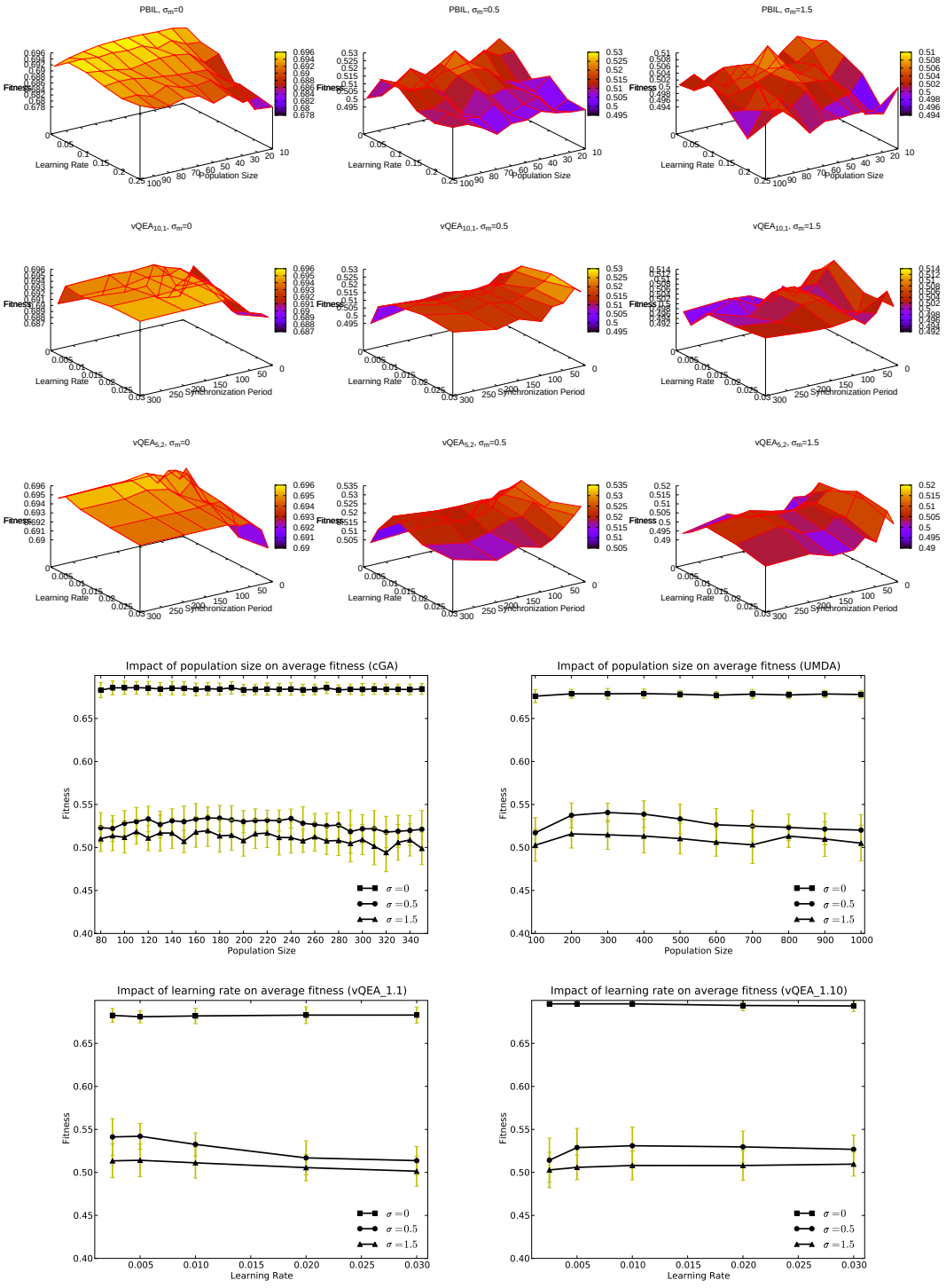


Figure C.9: Results on the K=4 problem with constant noise

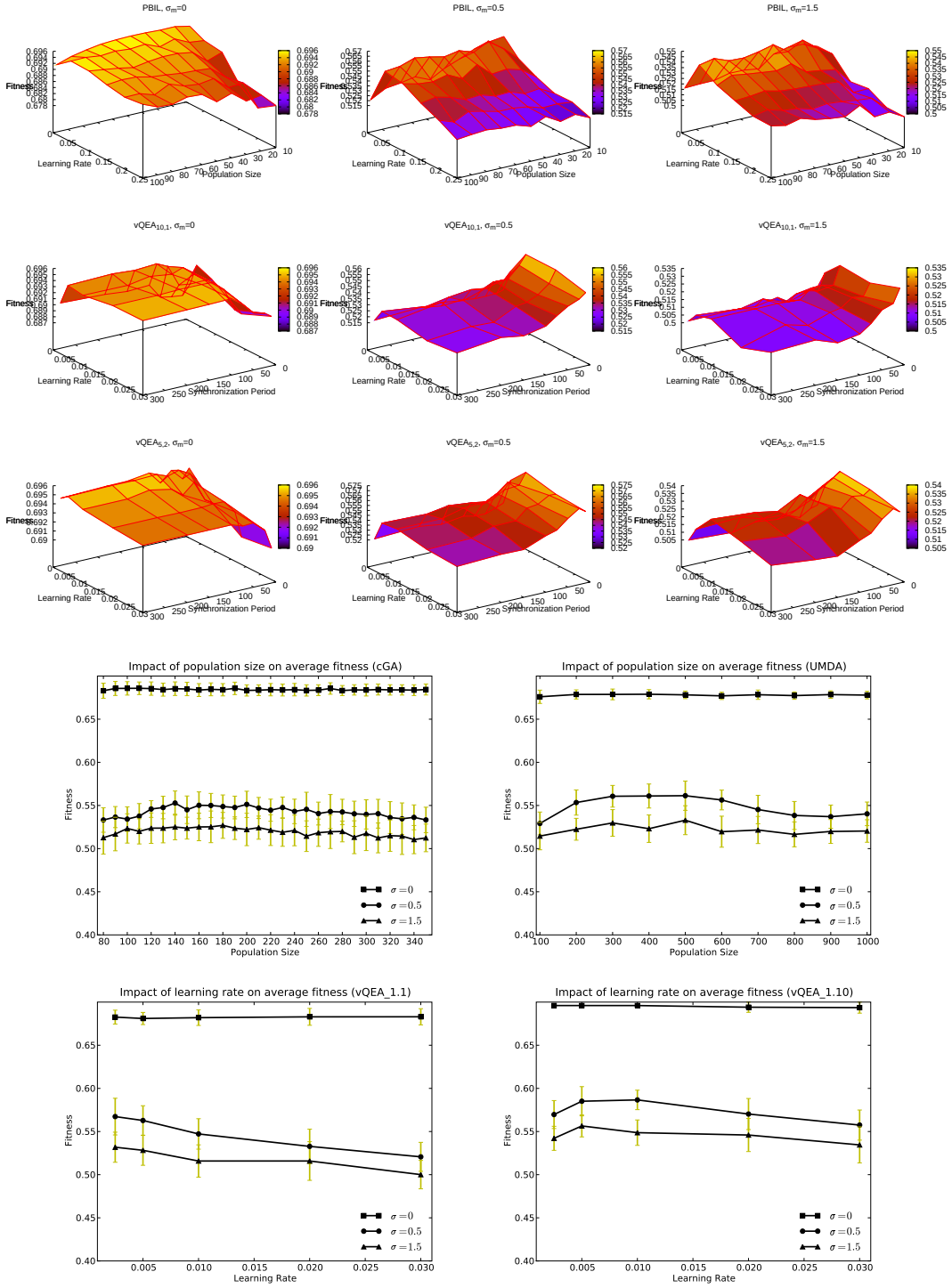


Figure C.10: Results on the K=4 problem with linear noise

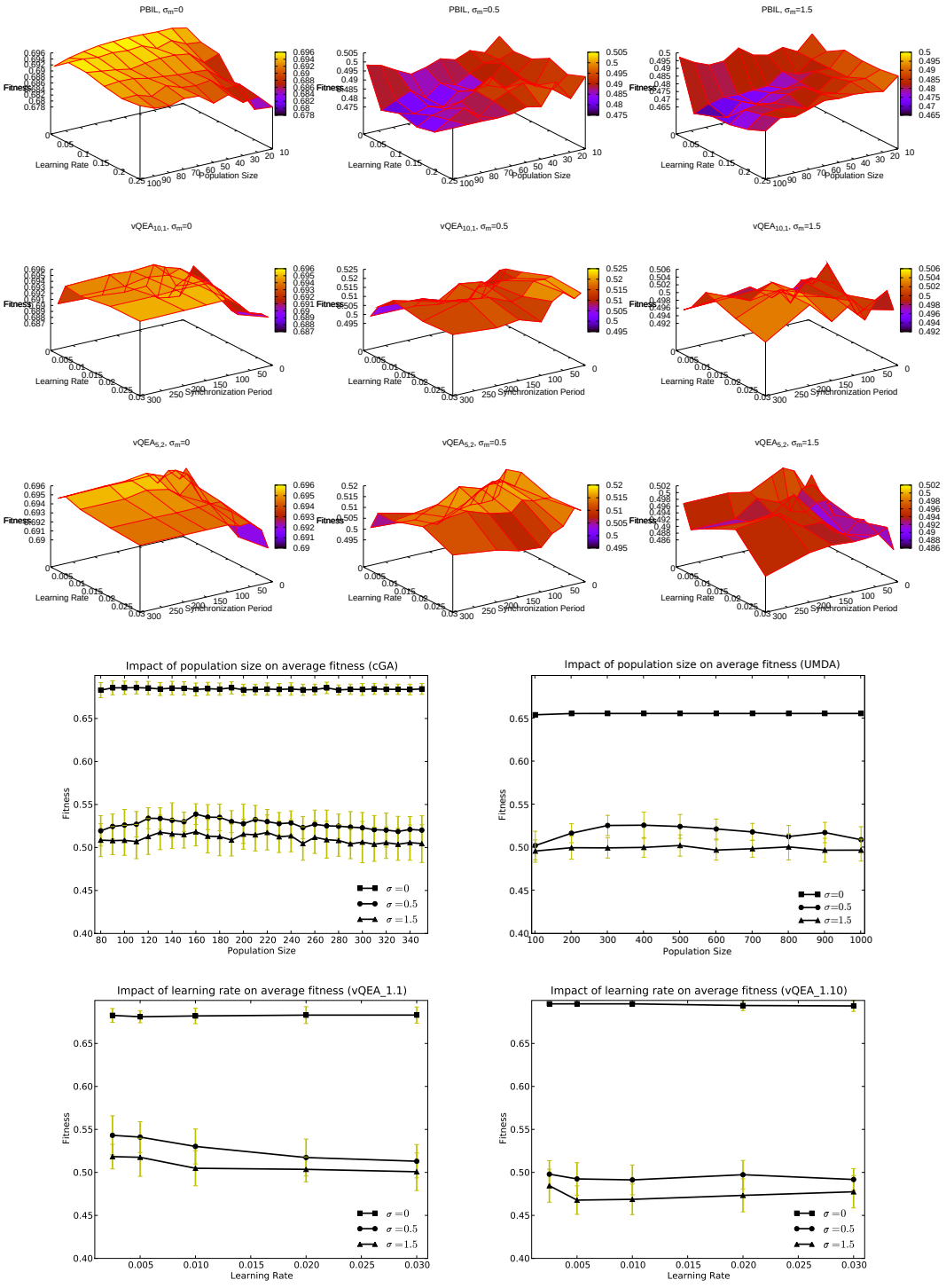


Figure C.11: Results on the K=4 problem with inverse linear noise

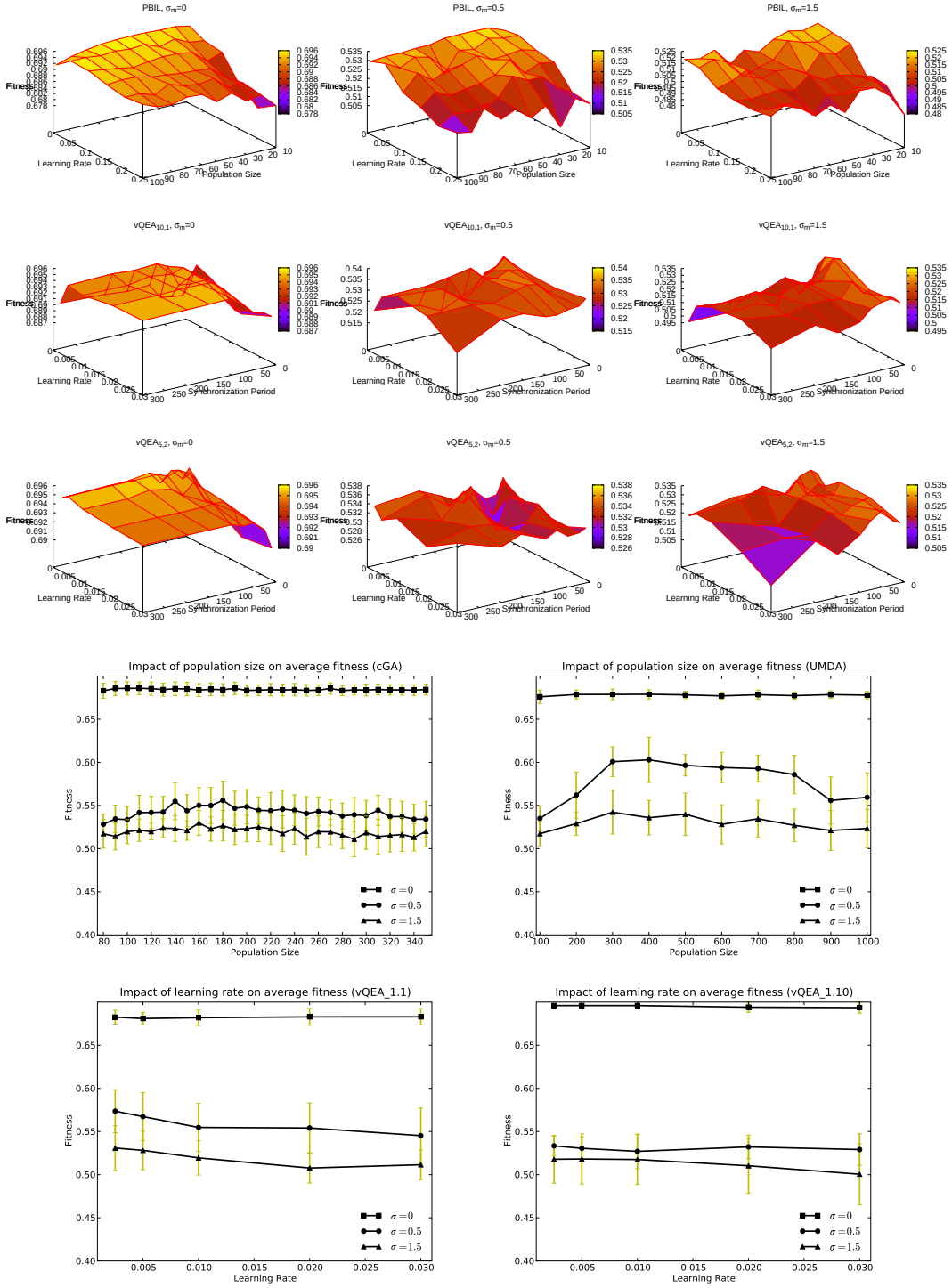


Figure C.12: Results on the K=4 problem with cosine noise

REFERENCES

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin, 50*(5-6).

Adrian, E. D. (1926). The impulses produced by sensory nerve endings. *Journal of Physiology (London), 61*, 49–72.

Ahn, C. W., Goldberg, D. E., & Ramakrishna, R. S. (2003). Multiple-deme parallel estimation of distribution algorithms: Basic framework and application. In *Ppam* (pp. 544–551). Czestochowa, Poland: Springer Berlin / Heidelberg.

Ahn, C. W., Kim, K. P., & Ramakrishna, R. S. (2003a). A memory-efficient elitist genetic algorithm. In *Ppam* (p. 552-559). Czestochowa, Poland: Springer Berlin / Heidelberg.

Ahn, C. W., Kim, K. P., & Ramakrishna, R. S. (2003b). A memory-efficient elitist genetic algorithm. In *Parallel processing and applied mathematics, ppam2003, 5th international conference, proceedings* (pp. 552–559). Czestochowa, Poland: Springer Berlin / Heidelberg.

Ahn, C. W., & Ramakrishna, R. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation, 7*(4), 367–385.

Aizawa, A. N., & Wah, B. W. (1993). Dynamic control of genetic algorithms in a noisy environment. In *Proceedings of the 5th international conference on genetic algorithms* (pp. 48–55). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Aizawa, A. N., & Wah, B. W. (1994). Scheduling of genetic algorithms in a noisy environment. *Evol. Comput., 2*(2), 97–122.

Alba, E., Montes, J. F. A., & Troya, J. M. (1993). Full automatic ANN design: A genetic approach. In *IWANN'93: Proceedings of the International Workshop on Artificial Neural Networks* (pp. 399–404). London, UK: Springer-Verlag.

Álvarez, A., Cearreta, I., López, J. M., Arruti, A., Lazkano, E., Sierra, B., et al. (2006). Feature subset selection based on evolutionary algorithms for automatic emotion recognition in spoken spanish and standard basque language. In P. Sojka, I. Kopecek, & K. Pala (Eds.), *Text, speech and dialogue, 9th international conference, tsd 2006, proceedings* (Vol. 4188, p. 565-572). Brno, Czech Republic: Springer.

Arnold, D. V., & Beyer, H.-G. (2003). A comparison of evolution strategies with other direct search methods in the presence of noise. *Comput. Optim. Appl., 24*(1), 135–159.

Auger, A., & Hansen, N. (2005, Sept.). Performance evaluation of an advanced local search evolutionary algorithm. In *IEEE Congress on Evolutionary Computation* (Vol. 2, p. 1777-1784 Vol. 2). IEEE Press.

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,* (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA: Carnegie Mellon University.

Baluja, S., & Davies, S. (1997). Using optimal dependency-trees for combinational optimization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 30–38). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Baluja, S., & Davies, S. (1998). Fast probabilistic modeling for combinatorial optimization. In *Aaai '98/iaai '98: Proceedings of the 15th national/10th conference on artificial intelligence/innovative applications of artificial intelligence* (pp. 469–476). Menlo Park, CA, USA: American Association for Artificial Intelligence.

Bell, C. C., Han, V. Z., Sugawara, Y., & Grant, K. (1997, May). Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature, 387*, 278-281.

Benuskova, L., Jain, V., Wysoski, S. G., & Kasabov, N. (2006). Computational neurogenetic modeling: a pathway to new discoveries in genetic neuroscience. *Intl. Journal of Neural Systems, 16*(3), 215-227.

Benuskova, L., & Kasabov, N. (2007). *Computational neurogenetic modelling*. NY: Springer.

Beyer, H.-G. (2000). Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering, 186*(2-4), 239 - 267.

Beyer, H.-G., & Sendhoff, B. (2006, Oct.). Functions with noise-induced multimodality: A test for evolutionary robust optimization-properties and performance analysis. *IEEE Transactions on Evolutionary Computation, 10*(5), 507-526.

Bi, G.-q., & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci., 18*(24), 10464-10472.

Bi, G.-q., & Poo, M.-m. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience, 24*(1), 139-166.

Blanco, R., Inza, I., Merino, M., Quiroga, J., & Larrañaga, P. (2005). Feature selection in bayesian classifiers for the prognosis of survival of cirrhotic patients treated with tips. *Journal of Biomedical Informatics, 38*(5), 376 - 388.

Bliss, T. V. P., & Gardner-Medwin, A. R. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path. *J Physiol.*, *232*(2), 357-374.

Bliss, T. V. P., & Lomo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *J Physiol.*, *232*(2), 331-356.

Bohte, S. M., Kok, J. N., & Poutré, J. A. L. (2000). SpikeProp: backpropagation for networks of spiking neurons. In *ESANN* (p. 419-424).

Bohte, S. M., Kok, J. N., & Poutré, J. A. L. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1-4), 17-37.

Bonet, J. S. de, Isbell, C. L., Jr., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in Neural Information Processing Systems* (Vol. 9, p. 424430). Cambridge, MA: The MIT Press.

Bosman, P. A., & Thierens, D. (2000). Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In *Parallel Problem Solving From Nature - PPSN VI* (pp. 767–776). London, UK: Springer.

Bower, J. M., & Beeman, D. (1995). *The book of genesis*. New York, USA: Springer-Verlag.

Brunel, N., Chance, F. S., Fourcaud, N., & Abbott, L. F. (2001, Mar). Effects of synaptic noise and filtering on the frequency response of spiking neurons. *Physical Review Letters*, *86*, 2186-2189.

Brunel, N., & Rossum, M. C. W. van. (2007). Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biol. Cybern.*, *97*(5), 337–339.

Burkitt, N. (2006a). A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biol. Cybern.*, *95*(1), 1–19.

Burkitt, N. (2006b). A review of the integrate-and-fire neuron model: II. inhomogeneous synaptic input and network properties. *Biol. Cybern.*, *95*(2), 97-112.

CABI. (2003). *Crop protection compendium, global module, 5th edition.* Wallingford, UK.

Carnevale, N. T., & Hines, M. L. (2006). *The NEURON book*. New York, NY, USA: Cambridge University Press.

Castellani, M. (2006). ANNE - a new algorithm for evolution of artificial neural network classifier systems. In *Ieee congress on evolutionary computation, cec'06* (p. 3294-3301).

Castellani, M., & Marques, N. (2008). FeaSANNT - an embedded evolutionary feature selection approach for neural network classifiers. *VIMation Journal*, *1*,

46-53.

Castellani, M., & Rowlands, H. (2009). Evolutionary artificial neural network design and training for wood veneer classification. *Engineering Applications of Artificial Intelligence*, *22*(4-5), 732 - 741.

Cocu, N., Harrington, R., Rounsevell, M., Worner, S., & Hulle, M. (2005). Geographical location, climate and land use influences on the phenology and numbers of the aphid, myzus persicae, in europe. *Journal of Biogeography*, *32*(4), 615-632.

Cordón, O., Fernández de Viana, I., Herrera, F., & Moreno, L. (2000, September 8-9). A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System. In M. Dorigo, M. Middendoff, & T. Stützle (Eds.), *From Ant Colonies to Artificial Ants: Proceedings of the Second International Workshop on Ant Algorithms* (pp. 22–29). Brussels, Belgium: Springer.

da Cruz, A. A., Vellasco, M., & Pacheco, M. (2006, July). Quantum-inspired evolutionary algorithm for numerical optimization. In *IEEE Congress on Evolutionary Computation, CEC'06* (p. 2630- 2637). Vancouver, Canada: IEEE Press.

de Sousa, H. C., & Riul Jr., A. (2002). Using MLP networks to classify red wines and water readings of an electronic tongue. *Neural Networks, Brazilian Symposium on*, *0*, 13.

Defoin-Platel, M., Schliebs, S., & Kasabov, N. (2007). A versatile quantum-inspired evolutionary algorithm. In *IEEE Congress on Evolutionary Computation, CEC'07* (pp. 423–430). Singapore: IEEE Press.

Defoin-Platel, M., Schliebs, S., & Kasabov, N. (2009, Dec.). Quantum-inspired evolutionary algorithm: A multimodel EDA. *Evolutionary Computation, IEEE Transactions on*, *13*(6), 1218-1232.

delaOssa, L., Gámez, J. A., & Puerta, J. M. (2006). Initial approaches to the application of islands-based parallel EDAs in continuous domains. *J. Parallel Distrib. Comput.*, *66*(8), 991–1001.

Delorme, A., Gautrais, J., VanRullen, R., & Thorpe, S. (1999). *SpikeNET: A simulator for modeling large networks of integrate and fire neurons.*

Delorme, A., Perrinet, L., & Thorpe, S. J. (2001). Networks of integrate-and-fire neurons using rank order coding B: Spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing*, *38-40*, 539-545.

Delorme, A., & Thorpe, S. J. (2001). Face identification using one spike per neuron: resistance to image degradations. *Neural Networks*, *14*(6-7), 795-803.

Delorme, A., & Thorpe, S. J. (2003, Nov). SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons. *Network: Computation in Neural Systems*, *14*, 613-627.

Deng, D., & Kasabov, N. K. (2000). ESOM: An algorithm to evolve self-organizing maps from on-line data streams. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN'00* (Vol. 6, p. 3-8). Como, Italy: IEEE Press.

Di Pietro, A., While, L., & Barone, L. (2004, 20-23 June). Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In *IEEE Congress on Evolutionary Computation, CEC'04* (pp. 1254–1261). Portland, Oregon: IEEE Press.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, *26*(1), 29–41.

Draa, A., Batouche, M., & Talbi, H. (2004a). A quantum-inspired differential evolution algorithm for rigid image registration. In *International Conference on Computational Intelligence, ICCI 2004, Proceedings* (pp. 408–411). Istanbul, Turkey: International Computational Intelligence Society.

Draa, A., Batouche, M., & Talbi, H. (2004b). A quantum-inspired differential evolution algorithm for rigid image registration. *Transactions on Engineering, Computing and Technology*, 408–411.

Egger, V., Feldmeyer, D., & Sakmann, B. (1999). Coincidence detection and changes of synaptic efficacy in spiny stellate neurons in rat barrel cortex. *Nature Neuroscience*, *2*, 1098-105.

Eiben, A., & Jelasity, M. (2002, May). A critical note on experimental research methodology in ec. In *Congress on evolutionary computation, cec '02* (Vol. 1, p. 582-587). HI, USA: IEEE Press.

Estevez, P., Tesmer, M., Perez, C., & Zurada, J. (2009, Feb.). Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, *20*(2), 189-201.

Fitzpatrick, J. M., & Grefenstette, J. (1988). Genetic algorithms in noisy environments. *Mach. Lang.*, *3*(2-3), 101–120.

Floreano, D., Epars, Y., Zufferey, J.-C., & Mattiussi, C. (2006). Evolution of spiking neural circuits in autonomous mobile robots: Research articles. *Int. J. Intell. Syst.*, *21*(9), 1005–1024.

Floreano, D., & Mattiussi, C. (2001). Evolution of spiking neural controllers for autonomous vision-based robots. In *ER '01: Proceedings of the International*

Symposium on Evolutionary Robotics From Intelligent Robotics to Artificial Life (pp. 38–61). London, UK: Springer-Verlag.

Florian, R. V. (2005). A reinforcement learning algorithm for spiking neural networks. In *SYNASC '05: Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (p. 299). Washington, DC, USA: IEEE Computer Society.

Florian, R. V. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, *19*(6), 1468-1502.

Forrest, S., & Mitchell, M. (1992). Relative building-block fitness and the building block hypothesis. In *FOGA* (p. 109-126). Colorado, USA: Morgan Kaufmann.

Friedman, N., & Goldszmidt, M. (1998). Learning bayesian networks with local structure. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models* (pp. 421–459). Norwell, MA, USA: Kluwer Academic Publishers.

Futuyma, D. J. (1998). *Evolutionary biology*. MA, USA: Sinauer, Sunderland.

Gallagher, M., & Frean, M. (2005). Population-based continuous optimization, probabilistic modelling and mean shift. *Evol. Comput.*, *13*(1), 29–42.

García-Martínez, C., & Lozano, M. (2005, Sept.). Hybrid real-coded genetic algorithms with female and male differentiation. In *IEEE Congress on Evolutionary Computation, CEC'05* (Vol. 1, p. 896-903 Vol.1). IEEE Press.

Garro, B. A., Sossa, H., & Vazquez, R. A. (2009). Design of artificial neural networks using a modified particle swarm optimization algorithm. *International Joint Conference on Neural Networks, IEEE - INNS - ENNS*, *0*, 938-945.

Gent, I. P., Grant, S. A., MacIntyre, E., Prosser, P., Shaw, P., Smith, B. M., et al. (1997). *How not to do it* (Tech. Rep. No. Report 97.27). Department of Computer Science, University of Strathclyde, Glasgow, Scottland.

Gerstner, W. (1999). Spiking neurons. In *Pulsed neural networks* (pp. 1–53). Cambridge, MA, USA: MIT Press.

Gerstner, W. (2000). Population dynamics of spiking neurons: Fast transients, asynchronous states, and locking. *Neural Comput.*, *12*(1), 43–89.

Gerstner, W., Hemmen, J. L. van, & Cowan, J. D. (1996). What matters in neuronal locking? *Neural Computation*, *8*(8), 1653–1676.

Gerstner, W., & Kistler, W. K. (2002a). Mathematical Formulations of Hebbian Learning. *Biological Cybernetics*, *87*(5-6), 404–415.

Gerstner, W., & Kistler, W. M. (2002b). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge, MA: Cambridge University Press.

Geweke, J. (1991). Efficient simulation from the multivariate normal and student-t distributions subject to linear constraints and the evaluation of constraint probabilities. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface* (pp. 571–578). Seattle, Washington: American Statistical Association, New York.

Glackin, B. P., McGinnity, T. M., Maguire, L. P., Wu, Q., & Belatreche, A. (2005). A novel approach for the implementation of large scale spiking neural networks on FPGA hardware. In *Computational Intelligence and Bioinspired Systems, 8th International Work-Conference on Artificial Neural Networks* (p. 552-563). Barcelona, Spain: Springer.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning* (1 ed.). Reading, Mass.: Addison-Wesley Professional.

Goldberg, D. E. (1990). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems, 5*, 139–167.

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*. Norwell, MA, USA: Kluwer Academic Publishers.

Goldberg, D. E., Deb, K., & Clark, J. H. (1991, 2-4). Genetic algorithms, noise, and the sizing of populations. In G. Rzevski & R. A. Adey (Eds.), *Applications of Artificial Intelligence in Engineering VI* (pp. 3–16). Oxford, UK: Elsevier Applied Science, London, UK.

González, C., Lozano, J. A., & Larrañaga, P. (2000). Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, *12*, 465–479.

Han, K.-H. (2003). *Quantum-inspired evolutionary algorithm*. Unpublished doctoral dissertation, Korea Advanced Institute of Science and Technology (KAIST).

Han, K.-H., & Kim, J.-H. (2002, December). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation, 6*(6), 580-593.

Han, K.-H., & Kim, J.-H. (2003). On setting the parameters of quantum-inspired evolutionary algorithm for practical application. In *Congress on Evolutionary Computation. CEC '03* (Vol. 1, p. 178-194). Canberra, Australia: IEEE Press.

Han, K.-H., & Kim, J.-H. (2004). Quantum-inspired evolutionary algorithms with a new termination criterion, H_ϵ gate, and two phase scheme. *IEEE Transactions on Evolutionary Computation, 8*(2), 156-169.

Han, K.-H., & Kim, J.-H. (2006). On the analysis of the quantum-inspired evolutionary algorithm with a single individual. In *IEEE Congress on Evolutionary Computation, CEC'06* (p. 16-21). Vancouver, Canada: IEEE Press.

Harik, G. R. (1999). *Linkage learning via probabilistic modeling in the ECGA* (Tech. Rep. No. 99010). IlliGAL, University of Illinois at Urbana-Champaign.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1999, November). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation, 3*(4), 287-297.

Hebb, D. O. (Ed.). (1949). *The organization of behavior*. New York: Wiley.

Hey, T. (1999, 06). Quantum computing: an introduction. *Computing & Control Engineering Journal, 10*, 105-112.

Hintz, K., & Spofford, J. (1990, Sep). Evolving a neural network. In *5th IEEE International Symposium on Intelligent Control* (p. 479-484 vol.1).

Hodgkin, A. L., & Huxley, A. F. (1952, August). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology, 117*(4), 500–544.

Hooker, J. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics, 1*, 33–42.

Iannella, N., & Kindermann, L. (2005). Finding iterative roots with a spiking neural network. *Information Processing Letters, 95*(6), 545-551.

Iglesias, J., Eriksson, J., Grize, F., Tomassini, M., & Villa, A. E. (2005). Dynamics of pruning in simulated large-scale spiking neural networks. *Biosystems, 79*(1-3), 11 - 20.

Iglesias, J., & Villa, A. E. (2007). Effect of stimulus-driven pruning on the detection of spatiotemporal patterns of activity in large neural networks. *Biosystems, 89*(1-3), 287 - 293.

Iglesias, J., & Villa, A. E. P. (2006). Neuronal cell death and synaptic pruning driven by spike-timing dependent plasticity. In S. D. Kollias, A. Stafylopatis, W. Duch, & E. Oja (Eds.), *International Conference on Artificial Neural Networks* (p. 953-962). Heidelberg Germany: Springer.

Inza, I., Larrañaga, P., & Sierra, B. (2001). Feature subset selection by bayesian networks: a comparison with genetic and sequential algorithms. *International Journal of Approximate Reasoning, 27*(2), 143 - 164.

Inza, I., Merino, M., Larrañaga, P., Quiroga, J., Sierra, B., & Girala, M. (2001). Feature subset selection by genetic algorithms and estimation of distribution algorithms: A case study in the survival of cirrhotic patients treated with TIPS. *Artificial Intelligence in Medicine, 23*(2), 187 - 205.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. on Neural Networks, 14*(6), 1569-1572.

Izhikevich, E. M. (2004, Sept.). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks, 15*(5), 1063-1070.

Izhikevich, E. M. (2006). *Dynamical systems in neuroscience: The geometry of excitability and bursting (computational neuroscience)* (1 ed.). The MIT Press.

Izhikevich, E. M., & Edelman, G. M. (2008, March). Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences, 105*(9), 3593–3598.

Jang, J.-S., Han, K.-H., & Kim, J.-H. (2004, June). Face detection using quantum-inspired evolutionary algorithm. In *Congress on Evolutionary Computation, CEC'04* (p. 2100-2106). Portland, Oregon: IEEE Press.

Janikow, C. Z., & Michalewicz, Z. (1991). An experimental comparison of binary and floating point representations in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms* (p. 31-36). USA: Morgan Kaufmann Press.

Jin, Y., & Branke, J. (2005). Evolutionary optimization in uncertain environments – A survey. *IEEE Transactions on Evolutionary Computation, 9*(3), 303-317.

Johnson, A., & Shapiro, J. L. (2001, Oct). The importance of selection mechanisms in distribution estimation algorithms. In *In Proceedings of the 5th International Conference on Artificial Evolution AE01*. London, UK: Springer.

Kandel, E. R. (2000). *Principles of neural science*. McGraw-Hill Education.

Kasabov, N. (1998a). ECOS: Evolving connectionist systems and the ECO learning paradigm. In S. Usui & T. Omori (Eds.), *The Fifth International Conference on Neural Information Processing, ICONIP'98* (p. 1232-1235). Kitakyushu, Japan: IOA Press.

Kasabov, N. (1998b). The ECOS framework and the ECO learning method for evolving connectionist systems. *JACIII, 2*(6), 195-202.

Kasabov, N. (1998c). Evolving fuzzy neural networks-algorithms, applications and biological motivation. In T. Yamakawa & G. Matsumoto (Eds.), *Methodologies for the Conception Design and Application of Soft Computing* (p. 271-274). Singapore: World Scientific.

Kasabov, N. (2006). *Evolving connectionist systems: The knowledge engineering approach*. NJ, USA: Springer-Verlag New York, Inc.

Kasabov, N. (2007). *Evolving connectionist systems: The knowledge engineering approach* (Second ed.). Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Kasabov, N. (2008). Integrative probabilistic evolving spiking neural networks utilising quantum inspired evolutionary algorithm: A computational framework. In *Advances in Neuro-Information Processing, 15th International Conference, ICONIP* (p. 3-13). Heidelberg, Germany: Springer.

Kasabov, N. (2010). To spike or not to spike: A probabilistic spiking neuron model. *Neural Networks, 23*(1), 16-19.

Kasabov, N., & Song, Q. (2002, Apr). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems, 10*(2), 144-154.

Kasinski, A. J., & Ponulak, F. (2005). Experimental demonstration of learning properties of a new supervised learning method for the spiking neural networks. In *Artificial Neural Networks: Biological Inspirations - ICANN 2005, 15th International Conference* (Vol. 3696, p. 145-152). Warsaw, Poland: Springer.

Kasinski, A. J., & Ponulak, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural networks. *Int. J.of Applied Mathematics and Computer Science, 16*, 101-113.

Kauffman, S. A. (1993). *The origins of order. Self-organization and selection in evolution*. New-York: Oxford University Press.

Kempter, R., Gerstner, W., & van Hemmen, J. L. (1999, Apr). Hebbian learning and spiking neurons. *Phys. Rev. E, 59*(4), 4498–4514.

Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., & Koumoutsakos, P. (2004). Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing, 3*(1), 77–112.

Kima, K., Hwang, J., Han, K.-H., Kim, J.-H., & Park, K. (2003, March). A quantum-inspired evolutionary computing algorithm for disk allocation method. *IEICE Transactions on Information and Systems, E86-D*(3), 645–649.

Kincaid, R., Griffith, M., Sykes, R., & Sobieszczanski-Sobieski, J. (2004). Bell-curve genetic algorithm for mixed continuous and discrete optimization problems. *Structural and Multidisciplinary Optimization, 26*, 396-405(10).

Kistler, W. M. (2002). Spike-timing dependent synaptic plasticity: a phenomenological framework. *Biological Cybernetics, 87*(5-6), 416-427.

Kistler, W. M., Seitz, R., & Hemmen, J. L. van. (1998). Modeling collective excitations in cortical tissue. *Phys. D, 114*(3-4), 273–295.

Knight, B. W. (1972). Dynamics of encoding in a population of neurons. *J. Gen. Physiol.*(59), 734-766.

Knoblauch, A. (2005). Neural associative memory for brain modeling and information retrieval. *Inf. Process. Lett., 95*(6), 537-544.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(1-2), 273 - 324.

Kohavi, R., & Sommerfield, D. (1995). Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of*

the First International Conference on Knowledge Discovery and Data Mining (KDD-95) (p. 192-197). Montreal, Canada: AAAI Press.

Lang, K. J., & Witbrock, M. J. (1988). Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Models Summer School San Mateo, Morgan Kauffmann (Ed.)* (p. 52-59). San Mateo, USA: Morgen Kauffmann Press.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen.*(9), 620-635.

Larrañaga, P., Etxeberria, E., Lozano, J. A., & Peña, J. M. (1999). *Optimization by learning and simulation of bayesian and gaussian networks* (Tech. Rep. No. EHU-KZAA-4/99). Basque Country, Spain: University of the Basque Country.

Larrañaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Boston: Kluwer Academic Publishers.

Lestienne, R. (1995). Determination of the precision of spike timing in the visual cortex of anaesthetised cats. *Biological Cybernetics*, *74*(1), 55-61.

Leung, F., Lam, H., Ling, S., & Tam, P. (2003, Jan). Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, *14*(1), 79-88.

Li, Y., & Jiao, L. (2005). Quantum-inspired immune clonal algorithm. In *Artificial immune systems* (pp. 304–317). Berlin/Heidelberg: Springer.

Liu, F., Li, S., Liang, M., & Hu, L. (2006). Wideband signal DOA estimation based on modified quantum genetic algorithm. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, *89*, 648-653.

Liu, J., Sun, J., & Xu, W. (2006). Improving quantum-behaved particle swarm optimization by simulated annealing. In *Computational Intelligence and Bioinformatics* (Vol. 4115/2006, pp. 130–136). Berlin/Heidelberg: Springer.

Loiselle, S., Rouat, J., Pressnitzer, D., & Thorpe, S. (2005, Aug). Exploration of rank order coding with spiking neural networks for speech recognition. In *IEEE International Joint Conference on Neural Networks, IJCNN '05* (Vol. 4, p. 2076-2080).

Lozano, J. A., Larrañaga, P., Inza, I., & Bengoetxea, E. (2006). *Towards a new evolutionary computation: Advances on estimation of distribution algorithms (studies in fuzziness and soft computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Maass, W. (1994). Lower bounds for the computational power of networks of spiking neurons. *Electronic Colloquium on Computational Complexity (ECCC)*, *1*(19).

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659 - 1671.

Maass, W. (1999). Computing with spiking neurons. In *Pulsed neural networks* (pp. 55–85). Cambridge, MA, USA: MIT Press.

Maass, W., & Bishop, C. M. (Eds.). (1999). *Pulsed neural networks*. Cambridge, MA, USA: MIT Press.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531-2560.

Madera, J., Alba, E., & Ochoa, A. (2006). A parallel island model for estimation of distribution algorithms. In P. L. E. J.A. Lozano (Ed.), *Towards a New Evolutionary Computation. Advances in the Estimation of Distribution Algorithms* (Vol. 192, pp. 159–186). New York: Springer Berlin / Heidelberg.

Maguire, L. P., McGinnity, T. M., Glackin, B., Ghani, A., Belatreche, A., & Harkin, J. (2007). Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomput.*, *71*(1-3), 13–29.

Malsburg, C. von der. (1981). *The correlation theory of brain function* (Tech. Rep. No. Internal Report 81-2). Göttingen, Germany: MPI für Biophysikalische Chemie.

Maniezzo, V. (1994, Jan). Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, *5*(1), 39-53.

Markon, S., Arnold, D., Back, T., Beielstein, T., & Beyer, H.-G. (2001). Thresholding – a selection operator for noisy ES. In *Congress on Evolutionary Computation, CEC'01* (Vol. 1, p. 465-472).

Markram, H. (2006). The blue brain project. *Nature Rev. Neurosci.*, *7*, 153-160.

Markram, H., Lubke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, *275*(5297), 213-215.

Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2008, 01). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE*, *3*(1), e1377.

Meffin, H., Burkitt, A., & Grayden, D. (2004). An analytical model for the "Large, Fluctuating Synaptic Conductance State" typical of neocortical neurons in vivo. *Journal of Computational Neuroscience*, *16*, 159-175(17).

Menneer, T., & Narayanan, A. (1995). *Quantum-inspired neural networks* (Tech. Rep. No. R329). Exeter, UK: Department of Computer Science, University of Exeter.

Michalewicz, Z., & Janikow, C. Z. (1991). Genetic algorithms for numerical optimization. *Statistics and Computing*, *1*(2), 75-91.

Miller, B. L., & Goldberg, D. E. (1996). Genetic algorithms, selection schemes, and the varying effects of noise. *Evol. Comput., 4*(2), 113–131.

Mininno, E., Cupertino, F., & Naso, D. (2008, April). Real-valued compact genetic algorithms for embedded microcontroller optimization. *IEEE Transactions on Evolutionary Computation, 12*(2), 203-219.

Monmarché, N., Ramat, E., Dromel, G., Slimane, M., & Venturini, G. (1999, Jan). *On the similarities between AS, BSC and PBIL: toward the birth of a new meta-heuristic* (Rapport interne No. 215). E3i Tours: Laboratoire d'Informatique de l'Université de Tours.

Mühlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation, 5*(3), 303-346.

Mühlenbein, H., Mahnig, T., & Rodriguez, A. O. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics, 5*(2), 215–247.

Mühlenbein, H., & Paass, G. (1996). From recombination of genes to the estimation of distributions I. binary parameters. In *PPSN* (p. 178-187).

Narayanan, A., & Moore, M. (1996). Quantum inspired genetic algorithms. In *International Conference on Evolutionary Computation* (pp. 61–66).

Natschläger, T., Maass, W., & Markram, H. (2002). The "Liquid Computer": A Novel Strategy for Real-Time Computing on Time Series. *Special Issue on Foundations of Information Processing of TELEMATIK, 8*(1), 39–43.

Natschläger, T., Markram, H., & Maass, W. (2003). Computer models and analysis tools for neural microcircuits. In *Neuroscience Database*s*: A Practical Guide*, (pp. 123–138). Kluwer Academic Publishers.

Nawrot, M. P., Schnepel, P., Aertsen, A., & Boucsein, C. (2009). Precisely timed signal transmission in neocortical networks with reliable intermediate-range projections. *Frontiers in Neural Circuits, 3*(2), 1–11.

Nelson, M., & Rinzel, J. (1995). The Hodgkin-Huxley model. In J. M. Bower & D. Beeman (Eds.), *The book of genesis* (p. 27-51). New York, USA: Springer-Verlag.

Ocenasek, J. (2002). *Parallel estimation of distribution algorithms*. Unpublished doctoral dissertation, Faculty of Information Technology, Brno University of Technology, Brno, Czech Rep.

Ocenasek, J., & Pelikan, M. (2004). Parallel mixed bayesian optimization algorithm: A scaleup analysis. *CoRR, cs.NE/0406007*.

Ocenasek, J., & Schwarz, J. (2002). Estimation of distribution algorithm for mixed continuous-discrete optimization problems. In *2nd Euro-International Sym-*

posium on Computational Intelligence (p. 227-232). Kosice, Slovakia: IOS Press.

Oliker, S., Furst, M., & Maimon, O. (1993). Design architectures and training of neural networks with a distributed genetic algorithm. In *IEEE International Conference on Neural Networks* (Vol. Vol. 1, p. 199-202).

Pelikan, M., Goldberg, D., & Lobo, F. (1999, Septemper). *A survey of optimization by building and using probabilistic model* (Tech. Rep. No. 99018). IlliGAL.

Pelikan, M., & Goldberg, D. E. (2001). *Escaping hierarchical traps with competent genetic algorithms* (IlliGAL Report No. 2001003). Urbana, IL: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Pelikan, M., Goldberg, D. E., & Cantú-paz, E. E. (2000). Linkage problem, distribution estimation, and bayesian networks. *Evol. Comput.*, *8*(3), 311–340.

Pelikan, M., Goldberg, D. E., & Sastry, K. (2000). *Bayesian optimization algorithm, decision graphs, and occam's razor* (IlliGAL Report No. 2000020). Urbana, IL: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, & P. K. Chawdhry (Eds.), *Advances in Soft Computing - Engineering Design and Manufacturing* (pp. 521–535). London: Springer-Verlag.

Pelikan, M., Sastry, K., Butz, M. V., & Goldberg, D. E. (2006). Hierarchical BOA on random decomposable problems. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 431–432). New York, NY, USA: ACM.

Perrinet, L., Delorme, A., Samuelides, M., & Thorpe, S. J. (2001). Networks of integrate-and-fire neuron using rank order coding A: How to implement spike time dependent Hebbian plasticity. *Neurocomputing*, *38-40*, 817-822.

Ponulak, F. (2005). *ReSuMe – new supervised learning method for spiking neural networks* (Tech. Rep.). Poznań, Poland: Institute of Control and Information Engineering, Poznań University of Technology.

Ponulak, F. (2008). Analysis of the resume learning process for spiking neural networks. *Applied Mathematics and Computer Science*, *18*(2), 117-127.

Ponulak, F., & Kasinski, A. J. (2006). Generalization properties of spiking neurons trained with ReSuMe method. In *ESANN 2006, 14th European Symposium on Artificial Neural Networks* (p. 629-634).

Posik, P. (2005, Sept.). Real-parameter optimization using the mutation step co-evolution. In *IEEE Congress on Evolutionary Computation, CEC'05* (Vol. 1,

p. 872-879 Vol.1). IEEE Press.

Potter, M. A., & Jong, K. A. D. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, *8*, 1–29.

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Rieke, F., Warland, D., Steveninck, R. de Ruyter van, & Bialek, W. (1999). *Spikes: exploring the neural code*. Cambridge, MA, USA: MIT Press.

Riul, A., Sousa, H. C. de, Malmegrim, R. R., Santos, D. S. dos, Carvalho, A. C. P. L. F., Fonseca, F. J., et al. (2004). Wine classification by taste sensors made from ultra-thin films and using neural networks. *Sensors and Actuators B: Chemical*, *98*(1), 77 - 82.

Rivero, D., Dorado, J., Fernández-Blanco, E., & Pazos, A. (2009). A genetic algorithm for ANN design, training and simplification. In *IWANN '09: Proceedings of the 10th International Work-Conference on Artificial Neural Networks* (pp. 391–398). Berlin, Heidelberg: Springer-Verlag.

Rönkkönen, J., Kukkonen, S., & Price, K. (2005, Sept.). Real-parameter optimization with differential evolution. In *IEEE Congress on Evolutionary Computation, CEC'05* (Vol. 1, p. 506-513 Vol.1). IEEE Press.

Rudnick, W. M. (1992). *Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks*. Unpublished doctoral dissertation, Beaverton, OR, USA.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* (pp. 318–362). Cambridge, MA, USA: MIT Press.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507-2517.

Sanderson, C., & Paliwal, K. K. (2003). Noise compensation in a person verification system using face and multiple speech features. *Pattern Recognition*, *36*(2), 293 - 302.

Sastry, K., Goldberg, D. E., & Llora, X. (2007). Towards billion-bit optimization via a parallel estimation of distribution algorithm. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation* (pp. 577–584). New York, NY, USA: ACM Press.

Schliebs, S., Defoin-Platel, M., & Kasabov, N. (2009). Integrated feature and parameter optimization for an evolving spiking neural network. In M. Köppen, N. K. Kasabov, & G. G. Coghill (Eds.), *Advances in Neuro-Information Pro-*

cessing, 15th International Conference (Vol. 5506, p. 1229-1236). Heidelberg, Germany: Springer.

Schliebs, S., Defoin-Platel, M., Worner, S., & Kasabov, N. (2009a). Integrated feature and parameter optimization for an evolving spiking neural network: Exploring heterogeneous probabilistic models. *Neural Networks, 22*(5-6), 623 - 632.

Schliebs, S., Defoin-Platel, M., Worner, S., & Kasabov, N. (2009b). Quantum-inspired feature and parameter optimisation of evolving spiking neural networks with a case study from ecological modeling. In *International Joint Conference on Neural Networks, IEEE - INNS - ENNS* (Vol. 0, p. 2833-2840). Los Alamitos, CA, USA: IEEE Computer Society.

Schrauwen, B., & van Campenhout, J. (2004, 11). Improving SpikeProp: Enhancements to an error-backpropagation rule for spiking neural networks. In *Proceedings of the 15th ProRISC Workshop.*

Schwefel, H.-P. (1981). *Numerical optimization of computer models*. New York, NY, USA: John Wiley & Sons, Inc.

Sebag, M., & Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature* (pp. 418–427). London, UK: Springer-Verlag.

Segundo, J. P., Moore, G. P., Stensaas, L. J., & Bullock, T. H. (1963). Sensitivity Of Neurones In Aplysia to Temporal Pattern of Arriving Impulses. *J Exp Biol, 40*(4), 643-667.

Sendhoff, B., Beyer, H.-G., & Olhofer, M. (2002). On noise induced multi-modality in evolutionary algorithms. In L.Wang, K. Tan, T. Furuhashi, J.-H. Kim, & X.Yao (Eds.), *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning - SEAL* (Vol. 1, p. 219-224).

Servet, I., Travé-Massuyès, L., & Stern, D. (1998). Telephone network traffic overloading diagnosis and evolutionary computation techniques. In *AE '97: Selected Papers from the Third European Conference on Artificial Evolution* (pp. 137–144). London, UK: Springer-Verlag.

Seung, H. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron, 40*(6), 1063 - 1073.

Shapiro, J. L. (2005). Drift and scaling in estimation of distribution algorithms. *Evol. Comput., 13*(1), 99–123.

Shapiro, J. L. (2006). Diversity loss in general estimation of distribution algorithms. In *PPSN* (pp. 92–101).

Socha, K. (2004). ACO for Continuous and Mixed-Variable Optimization. In (pp. 25–36). Berlin / Heidelberg: Springer.

Soltic, S. (2009). *Evolving connectionist systems for adaptive decision support with application in ecological data modelling*. Unpublished doctoral dissertation, Auckland University of Technology.

Soltic, S., Wysoski, S., & Kasabov, N. (2008). Evolving spiking neural networks for taste recognition. In *IEEE World Congress on Computational Intelligence (WCCI), Hong Kong* (p. 2091-2097).

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, *11*(4), 341–359.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., et al. (2005). *Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization* (Tech. Rep.). Singapore: Nanyang Technological University.

Talbi, H., Draa, A., & Batouche, M. (2006). A novel quantum-inspired evaluation algorithm for multi-source affine image registration. *Int. Arab J. Inf. Technol.*, *3*(1), 9-15.

Tang, K., Yao, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M., et al. (2007). *Benchmark functions for the CEC'2008 special session and competition on large scale global optimization* (Tech. Rep.). China: Nature Inspired Computation and Applications Laboratory, USTC.

Thierens, D., Goldberg, D., & Pereira, A. (1998, May). Domino convergence, drift, and the temporal-salience structure of problems. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence* (p. 535-540). IEEE Press.

Thorpe, S. J. (1990). Spike arrival times: A highly efficient coding scheme for neural networks. In R. Eckmiller, G. Hartmann, & G. Hauske (Eds.), *Parallel processing in neural systems, international conference on* (p. 91-94). North-Holland: Elsevier.

Thorpe, S. J. (1997). How can the human visual system process a natural scene in under 150ms? On the role of asynchronous spike propagation. In *ESANN*. D-Facto public.

Thorpe, S. J., Delorme, A., & Rullen, R. van. (2001). Spike-based strategies for rapid processing. *Neural Networks*, *14*(6-7), 715-725.

Thorpe, S. J., Fize, D., & Marlot, C. (1996, Jun). Speed of processing in the human visual system. *Nature*, *381*, 520-522.

Thorpe, S. J., & Gautrais, J. (1996). Rapid visual processing using spike asynchrony. In *Advances in Neural Information Processing Systems 9, NIPS* (p. 901-907). Denver, CO, USA: MIT Press.

Thorpe, S. J., & Gautrais, J. (1998). Rank order coding. In *CNS '97: Proceedings of the 6th annual conference on Computational neuroscience: trends in research, 1998* (pp. 113–118). New York, NY, USA: Plenum Press.

Thorpe, S. J., Guyonneau, R., Guilbaud, N., Allegraud, J.-M., & VanRullen, R. (2004). SpikeNet: real-time visual processing with one spike per neuron. *Neurocomputing*, *58-60*, 857 - 864.

Tiňo, P., & Mills, A. J. S. (2006). Learning beyond finite memory in recurrent networks of spiking neurons. *Neural Computation*, *18*(3), 591-613.

Tovee, M. J., Rolls, E. T., Treves, A., & Bellis, R. P. (1993). Information encoding and the responses of single neurons in the primate temporal visual cortex. *J Neurophysiol*, *70*(2), 640-654.

Valko, M., Marques, N. C., & Castelani, M. (2005). Evolutionary feature selection for spiking neural network pattern classifiers. In B. et al. (Ed.), *Proceedings of 2005 Portuguese Conference on Artificial Intelligence* (pp. 24–32). IEEE Press.

Van Rullen, R., Gautrais, J., Delorme, A., & Thorpe, S. (1998, November). Face processing using one spike per neurone. *Biosystems*, *48*(1-3), 229–239.

Van Rullen, R., & Thorpe, S. J. (2001). Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Comput.*, *13*(6), 1255–1283.

Venayagamoorthy, G. K., & Singhal, G. (2005, December). Quantum-inspired evolutionary algorithms and binary particle swarm optimization for training MLP and SRN neural networks. *Journal of Computational and Theoretical Nanoscience*, *2*(4), 561–568.

Vera, M. T., Rodriguez, R., Segura, D. F., Cladera, J. L., & Sutherst, R. W. (2002). Potential geographical distribution of the mediterranean fuit fly, ceratitis capitata (diptera:tephritidae) with emphasis on argentina and australia. *Environmental entomology*, *31*(6), 1009-1022.

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391 - 403.

Verstraeten, D., Schrauwen, B., & Stroobandt, D. (2005). Isolated word recognition using a liquid state machine. In *ESANN* (p. 435-440).

Villa, A. E. P., Tetko, I. V., Hyland, B., & Najem, A. (1999). Spatiotemporal activity patterns of rat cortical neurons predict responses in a conditioned task. *Proceedings of the National Academy of Sciences of the United States of America*, *96*(3), 1106-1111.

Wang, X., Hou, Z.-G., Zou, A., Tan, M., & Cheng, L. (2008). A behavior controller based on spiking neural networks for mobile robots. *Neurocomputing*, *71*(4-6), 655 - 666.

Watts, M. (2009, May). A decade of Kasabov's evolving connectionist systems: A review. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *39*(3), 253-269.

Watts, M., & Worner, S. (2006). Using mlp to determine abiotic factors influencing the establishment of insect pest species. In *International Joint Conference on Neural Networks, IJCNN '06* (p. 1840-1845). Vancouver, Canada: IEEE.

Weinberger, E. D. (1996). *NP completeness of kauffman's NK model, a tuneably rugged fitness landscape* (Tech. Rep. No. 96-02-003). Santa Fe Institute.

White, D., & Ligomenides, P. A. (1993). GANNet: A genetic algorithm for optimizing topology and weights in neural network design. In *IWANN '93: Proceedings of the International Workshop on Artificial Neural Networks* (pp. 322–327). London, UK: Springer-Verlag.

Whitley, D., Rana, S., Dzubera, J., & Mathias, K. E. (1996). Evaluating evolutionary algorithms. *Artificial Intelligence*, *85*(1-2), 245 - 276.

Wiersma, C. A. G. (1951). A Bifunctional Single Motor Axon System of a Crustacean Muscle. *J Exp Biol*, *28*(1), 13-21.

Wineberg, M., & Oppacher, F. (2003). Metrics for population comparisons in evolutionary computation systems. In M. Hamza (Ed.), *Intelligent systems and control* (Vol. 388, p. 24). Salzburg, Austria.

Worner, S., Lankin, G., Samarasinghe, S., & Teulon, D. (2002). Improving prediction of aphid flights by temporal analysis of input data for an artificial neural network. *New Zealand Plant Protection*, *55*, 312-316.

Worner, S., Leday, G., & Ikeda, T. (2008). Uncertainty analysis and ensemble selection of statistical and machine learning models that predict species distribution. In *Ecological informatics.* Cancun, Mexico.

Wysoski, S. G. (2008). *Evolving spiking neural networks for adaptive audiovisual pattern recognition*. Unpublished doctoral dissertation, Auckland University of Technology. (http://hdl.handle.net/10292/390)

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2006b). On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition.

In *Artificial Neural Networks ICANN 2006* (p. 61-70). Berlin / Heidelberg: Springer.

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2007). Text-independent speaker authentication with spiking neural networks. In *ICANN (2)* (Vol. 4669/2007, p. 758-767). Berlin / Heidelberg: Springer.

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2008a). Adaptive spiking neural networks for audiovisual pattern recognition. In *Neural Information Processing: 14th International Conference, ICONIP 2007* (pp. 406–415). Berlin, Heidelberg: Springer-Verlag.

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2008b). Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing, 71*(13-15), 2563 - 2575.

Wysoski, S. G., Benuskova, L., & Kasabov, N. K. (2006a). Adaptive learning procedure for a network of spiking neurons and visual pattern recognition. In *Advanced Concepts for Intelligent Vision Systems* (p. 1133-1142). Berlin / Heidelberg: Springer.

Xie, X., & Seung, H. S. (2004, Apr). Learning in neural networks by reinforcement of irregular spiking. *Phys. Rev. E, 69*(4), 041909.

Xin, J., & Embrechts, M. (2001). Supervised learning with spiking neural networks. In *International Joint Conference on Neural Networks, IJCNN '01* (Vol. 3, p. 1772-1777). IEEE Press.

Yuan, B., & Gallagher, M. (2003, Dec.). Playing in continuous spaces: some analysis and extension of population-based incremental learning. In *Congress on Evolutionary Computation, CEC'03* (Vol. 1, p. 443-450). Australia: IEEE Press.

Yuan, B., & Gallagher, M. (2005, Sept.). Experimental results for the special session on real-parameter optimization at CEC 2005: a simple, continuous eda. In *IEEE Congress on Evolutionary Computation, CEC'05* (Vol. 2, p. 1792-1799). USA: IEEE Press.

Zhang, Q. (2004). On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm. *IEEE Trans. Evolutionary Computation, 8*(1), 80-93.

Zhang, Q., & Mühlenbein, H. (2004). On the convergence of a class of estimation of distribution algorithms. *IEEE Trans. Evolutionary Computation, 8*(2), 127–136.

Zhang, Q., Sun, J., & Tsang, E. (2005). An evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation, 9*(2), 192–201.

Zhang, Q., Sun, J., Tsang, E., & Ford, J. A. (2002). Estimation of distribution algorithm based on mixture: preliminary experimental results. In *Proceedings of UKCI'02* (pp. 251–257).

Zhou, S., & Sun, Z. (2005a). Can ensemble method convert a 'weak' evolutionary algorithm to a 'strong' one? In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, CIMCA-IAWTIC'06* (pp. 68–74). Washington, DC, USA: IEEE Computer Society.

Zhou, S., & Sun, Z. (2005b). A new approach belonging to EDAs: Quantum-inspired genetic algorithm with only one chromosome. In *ICNC (3)* (pp. 141–150).

Zhou, W., Zhou, C., Huang, Y., & Wang, Y. (2005). Analysis of gene expression data: Application of quantum-inspired evolutionary algorithm to minimum sum-of-squares clustering. In *Rough sets, fuzzy sets, data mining, and granular computing* (p. 383-391). Springer Berlin / Heidelberg.

Zuppicich, A., & Soltic, S. (2009). FPGA implementation of an evolving spiking neural network. In M. Köppen, N. K. Kasabov, & G. G. Coghill (Eds.), *Advances in Neuro-Information Processing, 15th International Conference* (Vol. 5506, p. 1129-1136). Heidelberg, Germany: Springer.