

Computational Intelligence in Control

**Masoud Mohammadian
Ruhul Amin Sarker
Xin Yao**

IDEA GROUP PUBLISHING

Computational Intelligence in Control

Masoud Mohammadian, University of Canberra, Australia
Ruhul Amin Sarker, University of New South Wales, Australia
Xin Yao, University of Birmingham, UK



IDEA GROUP PUBLISHING

Hershey • London • Melbourne • Singapore • Beijing

Acquisition Editor: Mehdi Khosrowpour
Senior Managing Editor: Jan Travers
Managing Editor: Amanda Appicello
Development Editor: Michele Rossi
Copy Editor: Maria Boyer
Typesetter: Tamara Gillis
Cover Design: Integrated Book Technology
Printed at: Integrated Book Technology

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: <http://www.idea-group.com>

and in the United Kingdom by
Idea Group Publishing (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 3313
Web site: <http://www.eurospan.co.uk>

Copyright © 2003 by Idea Group Inc. All rights reserved. No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher.

Library of Congress Cataloging-in-Publication Data

Mohammadian, Masoud.

Computational intelligence in control / Masoud Mohammadian, Ruhul Amin Sarker and Xin Yao.

p. cm.

ISBN 1-59140-037-6 (hardcover) -- ISBN 1-59140-079-1 (ebook)

1. Neural networks (Computer science) 2. Automatic control. 3. Computational intelligence. I. Amin, Ruhul. II. Yao, Xin, 1962- III. Title.

QA76.87 .M58 2003

006.3--dc21

2002014188

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.



NEW from Idea Group Publishing

- **Digital Bridges: Developing Countries in the Knowledge Economy**, John Senyo Afele/ ISBN:1-59140-039-2; eISBN 1-59140-067-8, © 2003
- **Integrative Document & Content Management: Strategies for Exploiting Enterprise Knowledge**, Len Asprey and Michael Middleton/ ISBN: 1-59140-055-4; eISBN 1-59140-068-6, © 2003
- **Critical Reflections on Information Systems: A Systemic Approach**, Jeimy Cano/ ISBN: 1-59140-040-6; eISBN 1-59140-069-4, © 2003
- **Web-Enabled Systems Integration: Practices and Challenges**, Ajantha Dahanayake and Waltraud Gerhardt ISBN: 1-59140-041-4; eISBN 1-59140-070-8, © 2003
- **Public Information Technology: Policy and Management Issues**, G. David Garson/ ISBN: 1-59140-060-0; eISBN 1-59140-071-6, © 2003
- **Knowledge and Information Technology Management: Human and Social Perspectives**, Angappa Gunasekaran, Omar Khalil and Syed Mahbubur Rahman/ ISBN: 1-59140-032-5; eISBN 1-59140-072-4, © 2003
- **Building Knowledge Economies: Opportunities and Challenges**, Liaquat Hossain and Virginia Gibson/ ISBN: 1-59140-059-7; eISBN 1-59140-073-2, © 2003
- **Knowledge and Business Process Management**, Vlatka Hlupic/ISBN: 1-59140-036-8; eISBN 1-59140-074-0, © 2003
- **IT-Based Management: Challenges and Solutions**, Luiz Antonio Joia/ISBN: 1-59140-033-3; eISBN 1-59140-075-9, © 2003
- **Geographic Information Systems and Health Applications**, Omar Khan/ ISBN: 1-59140-042-2; eISBN 1-59140-076-7, © 2003
- **The Economic and Social Impacts of E-Commerce**, Sam Lubbe/ ISBN: 1-59140-043-0; eISBN 1-59140-077-5, © 2003
- **Computational Intelligence in Control**, Masoud Mohammadian, Ruhul Amin Sarker and Xin Yao/ISBN: 1-59140-037-6; eISBN 1-59140-079-1, © 2003
- **Decision-Making Support Systems: Achievements and Challenges for the New Decade**, M.C. Manuel Mora, Guisseppi Forgionne and Jatinder N.D. Gupta/ISBN: 1-59140-045-7; eISBN 1-59140-080-5, © 2003
- **Architectural Issues of Web-Enabled Electronic Business**, Nansi Shi and V.K. Murthy/ ISBN: 1-59140-049-X; eISBN 1-59140-081-3, © 2003
- **Adaptive Evolutionary Information Systems**, Nandish V. Patel/ISBN: 1-59140-034-1; eISBN 1-59140-082-1, © 2003
- **Managing Data Mining Technologies in Organizations: Techniques and Applications**, Parag Pendharkar/ ISBN: 1-59140-057-0; eISBN 1-59140-083-X, © 2003
- **Intelligent Agent Software Engineering**, Valentina Plekhanova/ ISBN: 1-59140-046-5; eISBN 1-59140-084-8, © 2003
- **Advances in Software Maintenance Management: Technologies and Solutions**, Macario Polo, Mario Piattini and Francisco Ruiz/ ISBN: 1-59140-047-3; eISBN 1-59140-085-6, © 2003
- **Multidimensional Databases: Problems and Solutions**, Maurizio Rafanelli/ISBN: 1-59140-053-8; eISBN 1-59140-086-4, © 2003
- **Information Technology Enabled Global Customer Service**, Tapio Reponen/ISBN: 1-59140-048-1; eISBN 1-59140-087-2, © 2003
- **Creating Business Value with Information Technology: Challenges and Solutions**, Namchul Shin/ISBN: 1-59140-038-4; eISBN 1-59140-088-0, © 2003
- **Advances in Mobile Commerce Technologies**, Ee-Peng Lim and Keng Siau/ ISBN: 1-59140-052-X; eISBN 1-59140-089-9, © 2003
- **Mobile Commerce: Technology, Theory and Applications**, Brian Mennecke and Troy Strader/ ISBN: 1-59140-044-9; eISBN 1-59140-090-2, © 2003
- **Managing Multimedia-Enabled Technologies in Organizations**, S.R. Subramanya/ISBN: 1-59140-054-6; eISBN 1-59140-091-0, © 2003
- **Web-Powered Databases**, David Taniar and Johanna Wenny Rahayu/ISBN: 1-59140-035-X; eISBN 1-59140-092-9, © 2003
- **E-Commerce and Cultural Values**, Theerasak Thanasankit/ISBN: 1-59140-056-2; eISBN 1-59140-093-7, © 2003
- **Information Modeling for Internet Applications**, Patrick van Bommel/ISBN: 1-59140-050-3; eISBN 1-59140-094-5, © 2003
- **Data Mining: Opportunities and Challenges**, John Wang/ISBN: 1-59140-051-1; eISBN 1-59140-095-3, © 2003
- **Annals of Cases on Information Technology – vol 5**, Mehdi Khosrowpour/ ISBN: 1-59140-061-9; eISBN 1-59140-096-1, © 2003
- **Advanced Topics in Database Research – vol 2**, Keng Siau/ISBN: 1-59140-063-5; eISBN 1-59140-098-8, © 2003
- **Advanced Topics in End User Computing – vol 2**, Mo Adam Mahmood/ISBN: 1-59140-065-1; eISBN 1-59140-100-3, © 2003
- **Advanced Topics in Global Information Management – vol 2**, Felix Tan/ ISBN: 1-59140-064-3; eISBN 1-59140-101-1, © 2003
- **Advanced Topics in Information Resources Management – vol 2**, Mehdi Khosrowpour/ ISBN: 1-59140-062-7; eISBN 1-59140-099-6, © 2003

Excellent additions to your institution's library! Recommend these titles to your Librarian!

To receive a copy of the Idea Group Publishing catalog, please contact (toll free) 1/800-345-4332, fax 1/717-533-8661, or visit the IGP Online Bookstore at:
[<http://www.idea-group.com>]

Note: All IGP books are also available as ebooks on netlibrary.com as well as other ebook sources. Contact Ms. Carrie Stull at cstull@idea-group.com to receive a complete list of sources where you can obtain ebook information or IGP titles.

Computational Intelligence in Control

Table of Contents

Preface	vii
---------------	-----

SECTION I: NEURAL NETWORKS DESIGN, CONTROL AND ROBOTICS APPLICATION

Chapter I. Designing Neural Network Ensembles by Minimising Mutual Information	1
---	----------

Yong Liu, The University of Aizu, Japan

Xin Yao, The University of Birmingham, UK

*Tetsuya Higuchi, National Institute of Advanced Industrial
Science and Technology, Japan*

Chapter II. A Perturbation Size-Independent Analysis of Robustness in Neural Networks by Randomized Algorithms	22
---	-----------

C. Alippi, Politecnico di Milano, Italy

Chapter III. Helicopter Motion Control Using a General Regression Neural Network	41
---	-----------

*T. G. B. Amaral, Superior Technical School of Setúbal - IPS
School, Portugal*

M. M. Crisóstomo, University of Coimbra, Portugal

*V. Fernão Pires, Superior Technical School of Setúbal - IPS
School, Portugal*

Chapter IV. A Biologically Inspired Neural Network Approach to Real-Time Map Building and Path Planning	69
--	-----------

Simon X. Yang, University of Guelph, Canada

**SECTION II: HYBRID EVOLUTIONARY SYSTEMS FOR
MODELLING, CONTROL AND ROBOTICS APPLICATIONS**

**Chapter V. Evolutionary Learning of Fuzzy Control in
Robot-Soccer 88**
*P.J. Thomas and R.J. Stonier, Central Queensland University,
Australia*

Chapter VI. Evolutionary Learning of a Box-Pushing Controller ... 104
*Pieter Spronck, Ida Sprinkhuizen-Kuyper, Eric Postma and
Rens Kortmann, Universiteit Maastricht, The Netherlands*

**Chapter VII. Computational Intelligence for Modelling and
Control of Multi-Robot Systems 122**
M. Mohammadian, University of Canberra, Australia

**Chapter VIII. Integrating Genetic Algorithms and Finite Element
Analyses for Structural Inverse Problems 136**
D.C. Panni and A.D. Nurse, Loughborough University, UK

SECTION III: FUZZY LOGIC AND BAYESIAN SYSTEMS

**Chapter IX. On the Modelling of a Human Pilot Using Fuzzy
Logic Control 148**
*M. Gestwa and J.-M. Bauschat, German Aerospace Center,
Germany*

Chapter X. Bayesian Agencies in Control 168
*Anet Potgieter and Judith Bishop, University of Pretoria,
South Africa*

**SECTION IV: MACHINE LEARNING, EVOLUTIONARY
OPTIMISATION AND INFORMATION RETRIEVAL**

**Chapter XI. Simulation Model for the Control of Olive Fly
Bactrocera Oleae Using Artificial Life Technique 183**
*Hongfei Gong and Agostinho Claudio da Rosa, LaSEEB-ISR,
Portugal*

Chapter XII. Applications of Data-Driven Modelling and Machine Learning in Control of Water Resources	197
<i>D.P. Solomatine, International Institute for Infrastructural, Hydraulic and Environmental Engineering (IHE-Delft), The Netherlands</i>	
Chapter XIII. Solving Two Multi-Objective Optimization Problems Using Evolutionary Algorithm	218
<i>Ruhul A. Sarker, Hussein A. Abbass and Charles S. Newton, University of New South Wales, Australia</i>	
Chapter XIV. Flexible Job-Shop Scheduling Problems: Formulation, Lower Bounds, Encoding and Controlled Evolutionary Approach ..	233
<i>Imed Kacem, Slim Hammadi and Pierre Borne, Laboratoire d'Automatique et Informatique de Lille, France</i>	
Chapter XV. The Effect of Multi-Parent Recombination on Evolution Strategies for Noisy Objective Functions	262
<i>Yoshiyuki Matsumura, Kazuhiro Ohkura and Kanji Ueda, Kobe University, Japan</i>	
Chapter XVI. On Measuring the Attributes of Evolutionary Algorithms: A Comparison of Algorithms Used for Information Retrieval	279
<i>J.L. Fernández-Villacañás Martín, Universidad Carlos III, Spain</i>	
<i>P. Marrow and M. Shackleton, BTexttract Technologies, UK</i>	
Chapter XVII. Design Wind Speeds Using Fast Fourier Transform: A Case Study	301
<i>Z. Ismail, N. H. Ramli and Z. Ibrahim, Universiti Malaya, Malaysia</i>	
<i>T. A. Majid and G. Sundaraj, Universiti Sains Malaysia, Malaysia</i>	
<i>W. H. W. Badaruzzaman, Universiti Kebangsaan Malaysia, Malaysia</i>	
About the Authors	321
Index	333

Preface

This book covers the recent applications of computational intelligence techniques for modelling, control and automation. The application of these techniques has been found useful in problems when the process is either difficult to model or difficult to solve by conventional methods. There are numerous practical applications of computational intelligence techniques in modelling, control, automation, prediction, image processing and data mining.

Research and development work in the area of computational intelligence is growing rapidly due to the many successful applications of these new techniques in very diverse problems. “Computational Intelligence” covers many fields such as neural networks, (adaptive) fuzzy logic, evolutionary computing, and their hybrids and derivatives. Many industries have benefited from adopting this technology. The increased number of patents and diverse range of products developed using computational intelligence methods is evidence of this fact.

These techniques have attracted increasing attention in recent years for solving many complex problems. They are inspired by nature, biology, statistical techniques, physics and neuroscience. They have been successfully applied in solving many complex problems where traditional problem-solving methods have failed. These modern techniques are taking firm steps as robust problem-solving mechanisms.

This volume aims to be a repository for the current and cutting-edge applications of computational intelligent techniques in modelling control and automation, an area with great demand in the market nowadays.

With roots in modelling, automation, identification and control, computational intelligence techniques provide an interdisciplinary area that is concerned with learning and adaptation of solutions for complex problems. This instantiated an enormous amount of research, searching for learning methods that are capable of controlling novel and non-trivial systems in different industries.

This book consists of open-solicited and invited papers written by leading researchers in the field of computational intelligence. All full papers have been peer review by at least two recognised reviewers. Our goal is to provide a book

that covers the foundation as well as the practical side of the computational intelligence.

The book consists of 17 chapters in the fields of self-learning and adaptive control, robotics and manufacturing, machine learning, evolutionary optimisation, information retrieval, fuzzy logic, Bayesian systems, neural networks and hybrid evolutionary computing.

This book will be highly useful to postgraduate students, researchers, doctoral students, instructors, and partitioners of computational intelligence techniques, industrial engineers, computer scientists and mathematicians with interest in modelling and control.

We would like to thank the senior and assistant editors of Idea Group Publishing for their professional and technical assistance during the preparation of this book. We are grateful to the unknown reviewers for the book proposal for their review and approval of the book proposal. Our special thanks goes to Michele Rossi and Mehdi Khosrowpour for their assistance and their valuable advise in finalizing this book.

We would like to acknowledge the assistance of all involved in the collation and review process of the book, without whose support and encouragement this book could not have been successfully completed.

We wish to thank all the authors for their insights and excellent contributions to this book. We would like also to thank our families for their understanding and support throughout this book project.

M. Mohammadian, R. Sarker and X. Yao

SECTION I:

**NEURAL
NETWORKS
DESIGN, CONTROL
AND ROBOTICS
APPLICATION**

Chapter I

Designing Neural Network Ensembles by Minimising Mutual Information

Yong Liu

The University of Aizu, Japan

Xin Yao

The University of Birmingham, UK

Tetsuya Higuchi

National Institute of Advanced Industrial Science and Technology, Japan

ABSTRACT

This chapter describes negative correlation learning for designing neural network ensembles. Negative correlation learning has been firstly analysed in terms of minimising mutual information on a regression task. By minimising the mutual information between variables extracted by two neural networks, they are forced to convey different information about some features of their input. Based on the decision boundaries and correct response sets, negative correlation learning has been further studied on two pattern classification problems. The purpose of examining the decision boundaries and the correct response sets is not only to illustrate the learning behavior of negative correlation learning, but also to cast light on how to design more effective neural network ensembles. The experimental results showed the decision boundary of the trained neural network ensemble by negative correlation learning is almost as good as the optimum decision boundary.

INTRODUCTION

In single neural network methods, the neural network learning problem is often formulated as an optimisation problem, i.e., minimising certain criteria, e.g., minimum error, fastest learning, lowest complexity, etc., about architectures. Learning algorithms, such as backpropagation (BP) (Rumelhart, Hinton & Williams, 1986), are used as optimisation algorithms to minimise an error function. Despite the different error functions used, these learning algorithms reduce a learning problem to the same kind of optimisation problem.

Learning is different from optimisation because we want the learned system to have best generalisation, which is different from minimising an error function. The neural network with the minimum error on the training set does not necessarily have the best generalisation unless there is an equivalence between generalisation and the error function. Unfortunately, measuring generalisation exactly and accurately is almost impossible in practice (Wolpert, 1990), although there are many theories and criteria on generalisation, such as the minimum description length (Rissanen, 1978), Akaike's information criteria (Akaike, 1974) and minimum message length (Wallace & Patrick, 1991). In practice, these criteria are often used to define better error functions in the hope that minimising the functions will maximise generalisation. While better error functions often lead to better generalisation of learned systems, there is no guarantee. Regardless of the error functions used, single network methods are still used as optimisation algorithms. They just optimise different error functions. The nature of the problem is unchanged.

While there is little we can do in single neural network methods, there are opportunities in neural network ensemble methods. Neural network ensembles adopt the divide-and-conquer strategy. Instead of using a single network to solve a task, a neural network ensemble combines a set of neural networks which learn to subdivide the task and thereby solve it more efficiently and elegantly. A neural network ensemble offers several advantages over a monolithic neural network. First, it can perform more complex tasks than any of its components (i.e., individual neural networks in the ensemble). Secondly, it can make an overall system easier to understand and modify. Finally, it is more robust than a monolithic neural network and can show graceful performance degradation in situations where only a subset of neural networks in the ensemble are performing correctly. Given the advantages of neural network ensembles and the complexity of the problems that are beginning to be investigated, it is clear that the neural network ensemble method will be an important and pervasive problem-solving technique.

The idea of designing an ensemble learning system consisting of many subsystems can be traced back to as early as 1958 (Selfridge, 1958; Nilsson, 1965). Since the early 1990s, algorithms based on similar ideas have been developed in many different but related forms, such as neural network ensembles

(Hansen & Salamon, 1990; Sharkey, 1996), mixtures of experts (Jacobs, Jordan, Nowlan & Hinton, 1991; Jacobs & Jordan, 1991; Jacobs, Jordan & Barto, 1991; Jacobs, 1997), various boosting and bagging methods (Drucker, Cortes, Jackel, LeCun & Vapnik, 1994; Schapire, 1990; Drucker, Schapire & Simard, 1993) and many others. There are a number of methods of designing neural network ensembles. To summarise, there are three ways of designing neural network ensembles in these methods: independent training, sequential training and simultaneous training.

A number of methods have been proposed to train a set of neural networks independently by varying initial random weights, the architectures, the learning algorithm used and the data (Hansen et al., 1990; Sarkar, 1996). Experimental results have shown that networks obtained from a given network architecture for different initial random weights often correctly recognize different subsets of a given test set (Hansen et al., 1990; Sarkar, 1996). As argued in Hansen et al. (1990), because each network makes generalisation errors on different subsets of the input space, the collective decision produced by the ensemble is less likely to be in error than the decision made by any of the individual networks.

Most independent training methods emphasised independence among individual neural networks in an ensemble. One of the disadvantages of such a method is the loss of interaction among the individual networks during learning. There is no consideration of whether what one individual learns has already been learned by other individuals. The errors of independently trained neural networks may still be positively correlated. It has been found that the combining results are weakened if the errors of individual networks are positively correlated (Clemen & Winkler, 1985). In order to decorrelate the individual neural networks, sequential training methods train a set of networks in a particular order (Drucker et al., 1993; Opitz & Shavlik, 1996; Rosen, 1996). Drucker et al. (1993) suggested training the neural networks using the boosting algorithm. The boosting algorithm was originally proposed by Schapire (1990). Schapire proved that it is theoretically possible to convert a weak learning algorithm that performs only slightly better than random guessing into one that achieves arbitrary accuracy. The proof presented by Schapire (1990) is constructive. The construction uses filtering to modify the distribution of examples in such a way as to force the weak learning algorithm to focus on the harder-to-learn parts of the distribution.

Most of the independent training methods and sequential training methods follow a two-stage design process: first generating individual networks, and then combining them. The possible interactions among the individual networks cannot be exploited until the integration stage. There is no feedback from the integration stage to the individual network design stage. It is possible that some of the independently designed networks do not make much contribution to the integrated system. In

order to use the feedback from the integration, simultaneous training methods train a set of networks together. Negative correlation learning (Liu & Yao, 1998a, 1998b, 1999) and the mixtures-of-experts (ME) architectures (Jacobs et al., 1991; Jordan & Jacobs, 1994) are two examples of simultaneous training methods. The idea of negative correlation learning is to encourage different individual networks in the ensemble to learn different parts or aspects of the training data, so that the ensemble can better learn the entire training data. In negative correlation learning, the individual networks are trained simultaneously rather than independently or sequentially. This provides an opportunity for the individual networks to interact with each other and to specialise.

In this chapter, negative correlation learning has been firstly analysed in terms of minimising mutual information on a regression task. The similarity measurement between two neural networks in an ensemble can be defined by the explicit mutual information of output variables extracted by two neural networks. The mutual information between two variables, output F_i of network i and output F_j of network j , is given by

$$I(F_i; F_j) = h(F_i) + h(F_j) - h(F_i, F_j) \quad (1)$$

where $h(F_i)$ is the entropy of F_i , $h(F_j)$ is the entropy of F_j , and $h(F_i, F_j)$ is the joint differential entropy of F_i and F_j . The equation shows that joint differential entropy can only have high entropy if the mutual information between two variables is low, while each variable has high individual entropy. That is, the lower mutual information two variables have, the more different they are. By minimising the mutual information between variables extracted by two neural networks, they are forced to convey different information about some features of their input. The idea of minimising mutual information is to encourage different individual networks to learn different parts or aspects of the training data so that the ensemble can learn the whole training data better.

Based on the decision boundaries and correct response sets, negative correlation learning has been further studied on two pattern classification problems. The purpose of examining the decision boundaries and the correct response sets is not only to illustrate the learning behavior of negative correlation learning, but also to cast light on how to design more effective neural network ensembles. The experimental results showed the decision boundary of the trained neural network ensemble by negative correlation learning is almost as good as the optimum decision boundary.

The rest of this chapter is organised as follows: Next, the chapter explores the connections between the mutual information and the correlation coefficient, and

explains how negative correlation learning can be used to minimise mutual information; then the chapter analyses negative correlation learning via the metrics of mutual information on a regression task; the chapter then discusses the decision boundaries constructed by negative correlation learning on a pattern classification problem; finally the chapter examines the correct response sets of individual networks trained by negative correlation learning and their intersections, and the chapter concludes with a summary of the chapter and a few remarks.

MINIMISING MUTUAL INFORMATION BY NEGATIVE CORRELATION LEARNING

Minimisation of Mutual Information

Suppose the output F_i of network i and the output F_j of network j are Gaussian random variables. Their variances are σ_i^2 and σ_j^2 , respectively. The mutual information between F_i and F_j can be defined by Eq.(1) (van der Lubbe, 1997, 1999). The differential entropy $h(F_i)$ and $h(F_j)$ are given by

$$h(F_i) = [1 + \log(2\pi\sigma_i^2)] / 2 \quad (2)$$

and

$$h(F_j) = [1 + \log(2\pi\sigma_j^2)] / 2 \quad (3)$$

The joint differential entropy $h(F_i, F_j)$ is given by

$$h(F_i, F_j) = 1 + \log(2\pi) + \log|\det(\Sigma)| \quad (4)$$

where Σ is the 2-by-2 covariance matrix of F_i and F_j . The determinant of Σ is

$$\det(\Sigma) = \sigma_i^2 \sigma_j^2 (1 - \rho_{ij}^2) \quad (5)$$

where ρ_{ij} is the correlation coefficient of F_i and F_j

$$\rho_{ij} = E[(F_i - E[F_i])(F_j - E[F_j])] / (\sigma_i^2 \sigma_j^2) \quad (6)$$

Using the formula of Eq.(5), we get

$$h(F_i, F_j) = 1 + \log(2\pi) + \log[\sigma_i^2 \sigma_j^2 (1 - \rho_{ij}^2)] / 2 \quad (7)$$

By substituting Eqs.(2), (3), and (7) in (1), we get

$$I(F_i; F_j) = -\log(1 - \rho_{ij}^2) / 2 \quad (8)$$

From Eq.(8), we may make the following statements:

1. If F_i and F_j are uncorrelated, the correlation coefficient ρ_{ij} is reduced to zero, and the mutual information $I(F_i; F_j)$ becomes very small.
2. If F_i and F_j are highly positively correlated, the correlation coefficient ρ_{ij} is close to 1, and mutual information $I(F_i; F_j)$ becomes very large.

Both theoretical and experimental results (Clemen et al., 1985) have indicated that when individual networks in an ensemble are unbiased, average procedures are most effective in combining them when errors in the individual networks are negatively correlated and moderately effective when the errors are uncorrelated. There is little to be gained from average procedures when the errors are positively correlated. In order to create a population of neural networks that are as uncorrelated as possible, the mutual information between each individual neural network and the rest of the population should be minimised. Minimising the mutual information between each individual neural network and the rest of the population is equivalent to minimising the correlation coefficient between them.

Negative Correlation Learning

Given the training data set $D = \{(\mathbf{x}(1), \mathbf{y}(1)), \dots, (\mathbf{x}(N), \mathbf{y}(N))\}$, we consider estimating \mathbf{y} by forming a neural network ensemble whose output is a simple averaging of outputs F_i of a set of neural networks. All the individual networks in the ensemble are trained on the same training data set D

$$F(n) = \frac{1}{M} \sum_{i=1}^M F_i(n) \quad (9)$$

where $F_i(n)$ is the output of individual network i on the n th training pattern $\mathbf{x}(n)$, $F(n)$ is the output of the neural network ensemble on the n th training pattern, and M is the number of individual networks in the neural network ensemble.

The idea of negative correlation learning is to introduce a correlation penalty term into the error function of each individual network so that the individual network can be trained simultaneously and interactively. The error function E_i for individual i on the training data set D in negative correlation learning is defined by

$$E_i = \frac{1}{N} \sum_{n=1}^N \left[\frac{1}{2} (F_i(n) - y(n))^2 + \lambda p_i(n) \right] \quad (10)$$

where N is the number of training patterns, $E_i(n)$ is the value of the error function of network i at presentation of the n th training pattern and $y(n)$ is the desired output of the n th training pattern. The first term in the right side of Eq. (10) is the mean-squared error of individual network i . The second term p_i is a correlation penalty function. The purpose of minimising p_i is to negatively correlate each individual's error with errors for the rest of the ensemble. The parameter λ is used to adjust the strength of the penalty.

The penalty function p_i has the form

$$p_i(n) = - (F_i(n) - F(n))^2 / 2 \quad (11)$$

The partial derivative of E_i with respect to the output of individual i on the n th training pattern is

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F_i(n) - y(n) - \lambda (F_i(n) - F(n)) \quad (12)$$

where we have made use of the assumption that the output of ensemble $F(n)$ has constant value with respect to $F_i(n)$. The value of parameter λ lies inside the range $0 \leq \lambda \leq 1$ so that both $(1 - \lambda)$ and λ have nonnegative values. BP (Rumelhart et al., 1996) algorithm has been used for weight adjustments in the mode of pattern-by-pattern updating. That is, weight updating of all the individual networks is performed simultaneously using Eq. (12) after the presentation of each training pattern. One complete presentation of the entire training set during the learning process is called an *epoch*. Negative correlation learning from Eq. (12) is a simple extension to the standard BP algorithm. In fact, the only modification that is needed is to calculate an extra term of the form $\lambda (F_i(n) - F(n))$ for the i th neural network.

From Eqs.(10), (11) and (12), we may make the following observations:

1. During the training process, all the individual networks interact with each other through their penalty terms in the error functions. Each network F_i minimises not only the difference between $F_i(n)$ and $y(n)$, but also the difference between $F(n)$ and $y(n)$. That is, negative correlation learning considers errors what all other neural networks have learned while training a neural network.
2. For $\lambda=0.0$, there are no correlation penalty terms in the error functions of the individual networks, and the individual networks are just trained independently using BP. That is, independent training using BP for the individual networks is a special case of negative correlation learning.
3. For $\lambda=1$, from Eq.(12) we get

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F(n) - y(n) \quad (13)$$

Note that the error of the ensemble for the n th training pattern is defined by

$$E_{ensemble} = \frac{1}{2} \left(\frac{1}{M} \sum_{i=1}^M F_i(n) - y(n) \right)^2 \quad (14)$$

The partial derivative of $E_{ensemble}$ with respect to F_i on the n th training pattern is

$$\frac{\partial E_{ensemble}}{\partial F_i(n)} = \frac{1}{M} (F(n) - y(n)) \quad (15)$$

In this case, we get

$$\frac{\partial E_i(n)}{\partial F_i(n)} \propto \frac{\partial E_{ensemble}}{\partial F_i(n)} \quad (16)$$

The minimisation of the error function of the ensemble is achieved by minimising the error functions of the individual networks. From this point of view, negative correlation learning provides a novel way to decompose the learning task of the ensemble into a number of subtasks for different individual networks.

ANALYSIS BASED ON MEASURING MUTUAL INFORMATION

In order to understand why and how negative correlation learning works, this section analyses it through measuring mutual information on a regression task in three cases: noise-free condition, small noise condition and large noise condition.

Simulation Setup

The regression function investigated here is

$$f(\mathbf{x}) = \frac{1}{13}[10 \sin(\pi x_1 x_2) + 20(x_3 - \frac{1}{2})^2 + 10x_4 + 5x_5] - 1 \quad (17)$$

where $\mathbf{x}=[x_1, \dots, x_5]$ is an input vector whose components lie between zero and one. The value of $f(\mathbf{x})$ lies in the interval $[-1, 1]$. This regression task has been used by Jacobs (1997) to estimate the bias of mixture-of-experts architectures and the variance and covariance of experts' weighted outputs.

Twenty-five training sets, $(\mathbf{x}^{(k)}(l), y^{(k)}(l))$, $l = 1, \dots, L$, $L = 500$, $k = 1, \dots, K$, $K = 25$, were created at random. Each set consisted of 500 input-output patterns in which the components of the input vectors were independently sampled from a uniform distribution over the interval $(0, 1)$. In the noise-free condition, the target outputs were not corrupted by noise; in the small noise condition, the target outputs were created by adding noise sampled from a Gaussian distribution with a mean of zero and a variance of $\sigma^2 = 0.1$ to the function $f(\mathbf{x})$; in the large noise condition, the target outputs were created by adding noise sampled from a Gaussian distribution with a mean of zero and a variance of $\sigma^2 = 0.2$ to the function $f(\mathbf{x})$. A testing set of 1,024 input-output patterns, $(\mathbf{t}(n), d(n))$, $n = 1, \dots, N$, $N = 1024$, was also generated. For this set, the components of the input vectors were independently sampled from a uniform distribution over the interval $(0, 1)$, and the target outputs were not corrupted by noise in all three conditions. Each individual network in the ensemble is a multi-layer perceptron with one hidden layer. All the individual networks have 5 hidden nodes in an ensemble architecture. The hidden node function is defined by the logistic function

$$\varphi(y) = \frac{1}{1 + \exp(-y)} \quad (18)$$

The network output is a linear combination of the outputs of the hidden nodes.

For each estimation of mutual information among an ensemble, 25 simulations were conducted. In each simulation, the ensemble was trained on a different training set from the same initial weights distributed inside a small range so that different simulations of an ensemble yielded different performances solely due to the use of different training sets. Such simulation setup follows the suggestions from Jacobs (1997).

Measurement of Mutual Information

The average outputs of the ensemble and the individual network i on the n th pattern in the testing set, $(\mathbf{t}(n), d(n))$, $n = 1, \dots, N$, are denoted and given respectively by

$$\overline{F}(\mathbf{t}(n)) = \frac{1}{K} \sum_{k=1}^K F^{(k)}(\mathbf{t}(n)) \quad (19)$$

and

$$\overline{F}_i(\mathbf{t}(n)) = \frac{1}{K} \sum_{k=1}^K F_i^{(k)}(\mathbf{t}(n)) \quad (20)$$

where $F^{(k)}(\mathbf{t}(n))$ and $F_i^{(k)}(\mathbf{t}(n))$ are the outputs of the ensemble and the individual network i on the n th pattern in the testing set from the k th simulation, respectively, and $K=25$ is the number of simulations. From Eq.(6), the correlation coefficient between network i and network j is given by

$$\rho_{ij} = \frac{\sum_{n=1}^N \sum_{k=1}^K \left(F_i^{(k)}(\mathbf{t}(n)) - \overline{F}_i(\mathbf{t}(n)) \right) \left(F_j^{(k)}(\mathbf{t}(n)) - \overline{F}_j(\mathbf{t}(n)) \right)}{\sqrt{\sum_{n=1}^N \sum_{k=1}^K \left(F_i^{(k)}(\mathbf{t}(n)) - \overline{F}_i(\mathbf{t}(n)) \right)^2 \sum_{n=1}^N \sum_{k=1}^K \left(F_j^{(k)}(\mathbf{t}(n)) - \overline{F}_j(\mathbf{t}(n)) \right)^2}} \quad (21)$$

From Eq.(8), the integrated mutual information among the ensembles can be defined by

$$E_{mi} = -\frac{1}{2} \sum_{i=1}^M \sum_{j=1, j \neq i}^M \log(1 - \rho_{ij}^2) \quad (22)$$

We may also define the integrated mean-squared error (MSE) on the testing set as

$$E_{mse} = \frac{1}{N} \sum_{n=1}^N \frac{1}{K} \sum_{k=1}^K (F^{(k)}(t(n)) - d(n))^2 \quad (23)$$

The integrated mean-squared error E_{train_mse} on the training set is given by

$$E_{train_mse} = \frac{1}{L} \sum_{l=1}^L \frac{1}{K} \sum_{k=1}^K (F^{(k)}(x^{(k)}(l)) - y^{(k)}(l))^2 \quad (24)$$

Results in the Noise-Free Condition

The results of negative correlation learning in the noise-free condition for the different values of λ at epoch 2000 are given in Table 1. The results suggest that both E_{train_mse} and E_{test_mse} appeared to decrease with the increasing value of λ . The mutual information E_{mi} among the ensemble decreased as the value of λ increased when $0 \leq \lambda \leq 0.5$. However, when λ increased further to 0.75 and 1, the mutual information E_{mi} had larger values. The reason of having larger mutual information at $\lambda = 0.75$ and $\lambda = 1$ is that some correlation coefficients had negative values and the mutual information depends on the absolute values of correlation coefficients.

In order to find out why E_{train_mse} decreased with increasing value of λ , the concept of capability of a trained ensemble is introduced. The capability of a trained ensemble is measured by its ability of producing correct input-output mapping on the training set used, specifically, by its integrated mean-squared error E_{train_mse} on the training set. The smaller E_{train_mse} is, the larger capability the trained ensemble has.

Results in the Noise Conditions

Table 2 and Table 3 compare the performance of negative correlation learning for different strength parameters in both small noise (variance $\sigma^2 = 0.1$) and large

Table 1: The results of negative correlation learning in the noise-free condition for different λ values at epoch 2000

λ	0	0.25	0.5	0.75	1
E_{mi}	0.3706	0.1478	0.1038	0.1704	0.6308
E_{test_mse}	0.0016	0.0013	0.0011	0.0007	0.0002
E_{train_mse}	0.0013	0.0010	0.0008	0.0005	0.0001

noise (variance $\sigma^2 = 0.2$) conditions. The results show that there were same trends for E_{mi} , E_{test_mse} and E_{train_mse} in both noise-free and noise conditions when $\lambda \leq 0.5$. That is, E_{mi} , E_{test_mse} and E_{train_mse} appeared to decrease with the increasing value of λ . However, E_{test_mse} appeared to decrease first and then increase with the increasing value of λ .

In order to find out why E_{test_mse} showed different trends in noise-free and noise conditions when $\lambda = 0.75$ and $\lambda = 1$, the integrated mean-squared error E_{train_mse} on the training set was also shown in Tables 1, 2 and 3. When $\lambda = 0$, the neural network ensemble trained had relatively large E_{train_mse} . It indicated that the capability of the neural network ensemble trained was not big enough to produce correct input-output mapping (i.e., it was underfitting) for this regression task. When $\lambda = 1$, the neural network ensemble trained learned too many specific input-output relations (i.e., it was overfitting), and it might memorise the training data and therefore be less able to generalise between similar input-output patterns. Although the overfitting was not observed for the neural network ensemble used in the noise-free condition, too large capability of the neural network ensemble will lead to overfitting for both noise-free and noise conditions because of the ill-posedness of any finite training set (Friedman, 1994).

Choosing a proper value of λ is important, and also problem dependent. For the noise conditions used for this regression task and the ensemble architected used, the performance of the ensemble was optimal for $\lambda = 0.5$ among the tested values of λ in the sense of minimising the MSE on the testing set.

Table 2: The results of negative correlation learning in the small noise condition for different λ values at epoch 2000

λ	0	0.25	0.5	0.75	1
E_{mi}	6.5495	3.8761	1.4547	0.3877	0.2431
E_{test_mse}	0.0137	0.0128	0.0124	0.0126	0.0290
E_{train_mse}	0.0962	0.0940	0.0915	0.0873	0.0778

Table 3: The results of negative correlation learning in the large noise condition for different λ values at epoch 2000

λ	0	0.25	0.5	0.75	1
E_{mi}	6.7503	3.9652	1.6957	0.4341	0.2030
E_{test_mse}	0.0249	0.0235	0.0228	0.0248	0.0633
E_{train_mse}	0.1895	0.1863	0.1813	0.1721	0.1512

ANALYSIS BASED ON DECISION BOUNDARIES

This section analyses the decision boundaries constructed by both negative correlation learning and the independent training. The independent training is a special case of negative correlation learning for $\lambda=0.0$ in Eq.(12).

Simulation Setup

The objective of the pattern classification problem is to distinguish between two classes of overlapping, two-dimensional, Gaussian-distributed patterns labeled 1 and 2. Let Class 1 and Class 2 denote the set of events for which a random vector \mathbf{x} belongs to patterns 1 and 2, respectively. We may then express the conditional probability density functions for the two classes:

$$f_x(x) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} \|\mathbf{x} - \mu_1\|^2\right) \quad (25)$$

where mean vector $\mu_1 = [0,0]^T$ and variance $\sigma_1^2 = 1$.

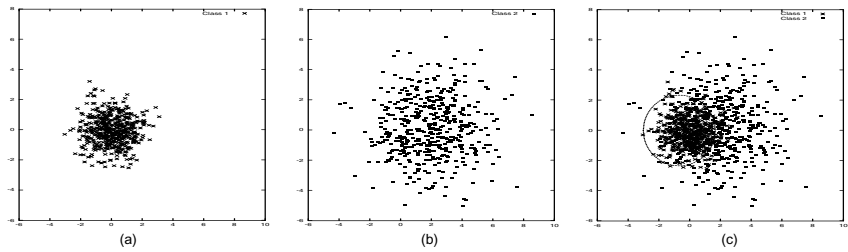
$$f_x(x) = \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{1}{2\sigma_2^2} \|\mathbf{x} - \mu_2\|^2\right) \quad (26)$$

where mean vector $\mu_2 = [0,0]^T$ and variance $\sigma_2^2 = 4$. The two classes are assumed to be equiprobable; that is $p_1 = p_2 = 1/2$. The costs for misclassifications are assumed to be equal, and the costs for correct classifications are assumed to be zero. On this basis, the (optimum) Bayes classifier achieves a probability of correct classification $p_c = 81.51$ percent. The boundary of the Bayes classifier consists of a circle of center $[-2/3, 0]^T$ and radius $r = 2.34$; 1000 points from each of two processes were generated for the training set. The testing set consists of 16,000 points from each of two classes.

Figure 1 shows individual scatter diagrams for classes and the joint scatter diagram representing the superposition of scatter plots of 500 points from each of two processes. This latter diagram clearly shows that the two distributions overlap each other significantly, indicating that there is inevitably a significant probability of misclassification.

The ensemble architecture used in the experiments has three networks. Each individual network in the ensemble is a multi-layer perceptron with one hidden layer. All the individual networks have three hidden nodes in an ensemble architecture.

Figure 1: (a) Scatter plot of Class 1; (b) Scatter plot of Class 2; (c) Combined scatter plot of both classes; the circle represents the optimum Bayes solution



Both hidden node function and output node function are defined by the logistic function in Eq.(18).

Experimental Results

The results presented in Table 4 pertain to 10 different runs of the experiment, with each run involving the use of 2,000 data points for training and 32,000 for testing. Figures 2 and 3 compare the decision boundaries constructed by negative

Figure 2: Decision boundaries formed by the different networks trained by the negative correlation learning ($\lambda = 0.75$): (a) Network 1; (b) Network 2; (c) Network 3; (d) Ensemble; the circle represents the optimum Bayes solution

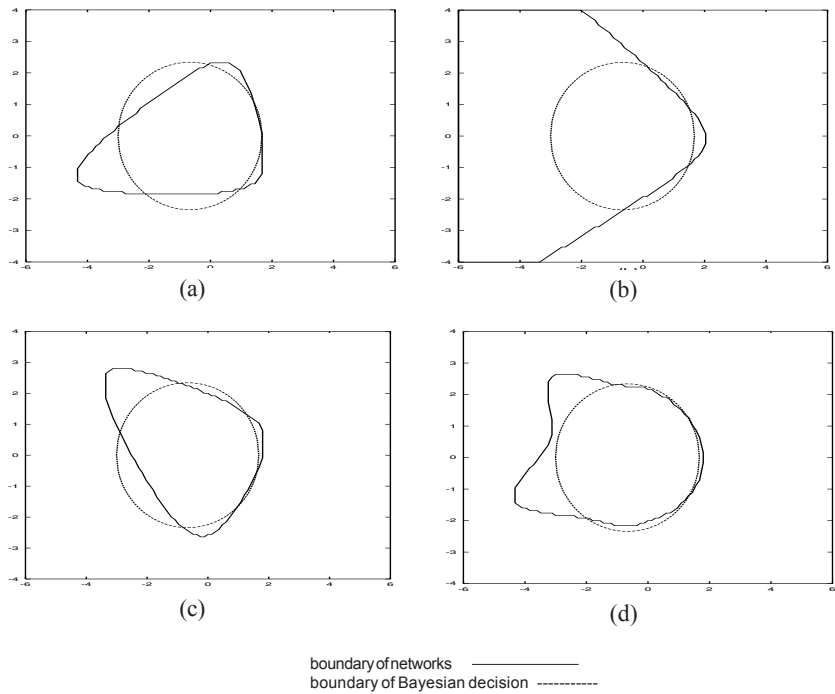
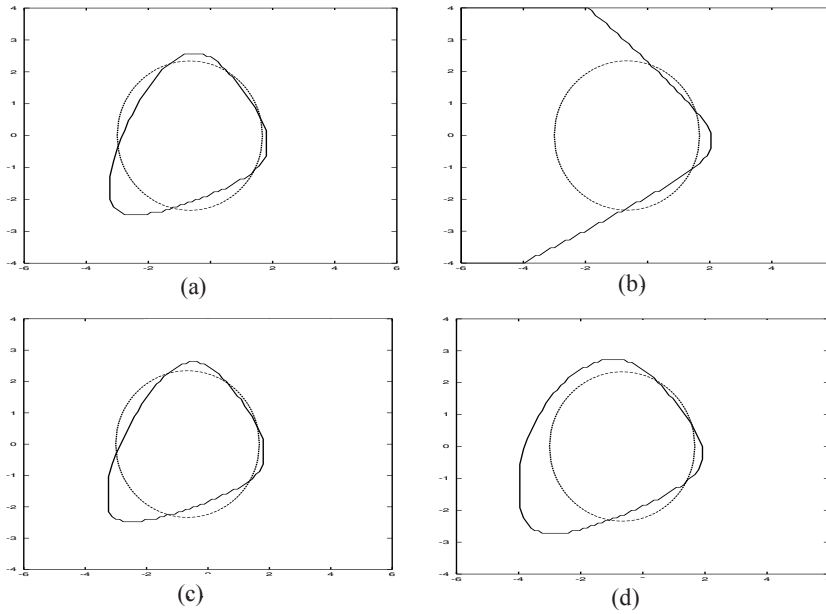


Figure 3: Decision boundaries formed by the different networks trained by the independent training (i.e., $\lambda = 0.0$ in negative correlation learning): (a) Network 1; (b) Network 2; (c) Network 3; (d) Ensemble; the circle represents the optimum Bayes solution



boundary of networks —————
boundary of Bayesian decision - - - - -

correlation learning and the independent training. In comparison of the average correct classification percentage and the decision boundaries obtained by the two ensemble learning methods, it is clear that negative correlation learning outperformed the independent training method. Although the classification performance of individual networks in the independent training is relatively good, the overall performance of the entire ensemble was not improved because different networks, such as Network 1 and Network 3 in Figure 3, tended to generate the similar decision boundaries.

The percentage of correct classification of the ensemble trained by negative correlation is 81.41, which is almost equal to that realised by the Bayesian classifier. Figure 2 clearly demonstrates that negative correlation learning is capable of constructing a decision between Class 1 and Class 2 that is almost as good as the optimum decision boundary. It is evident from Figure 2 that different individual networks trained by negative correlation learning were able to specialise to different parts of the testing set.

Table 4: Comparison between negative correlation learning (NCL) ($\lambda = 0.75$) and the independent training (i.e., $\lambda = 0.0$ in negative correlation learning) on the classification performance of individual networks and the ensemble; the results are the average correct classification percentage on the testing set over 10 independent runs

Methods	Net 1	Net 2	Net 3	Ensemble
NCL	81.11	75.26	73.09	81.03
Independent Training	81.13	80.49	81.13	80.99

ANALYSIS BASED ON THE CORRECT RESPONSE SETS

In this section, negative correlation learning was tested on the Australian credit card assessment problem. The problem is how to assess applications for credit cards based on a number of attributes. There are 690 patterns in total. The output has two classes. The 14 attributes include 6 numeric values and 8 discrete ones, the latter having from 2 to 14 possible values. The Australian credit card assessment problem is a classification problem which is different from the regression type of tasks, whose outputs are continuous. The data set was obtained from the UCI machine learning benchmark repository. It is available by anonymous ftp at [ics.uci.edu](ftp://ics.uci.edu) (128.195.1.1) in directory/pub/machine-learning-databases.

Experimental Setup

The data set was partitioned into two sets: a training set and a testing set. The first 518 examples were used for the training set, and the remaining 172 examples for the testing set. The input attributes were rescaled to between 0.0 and 1.0 by a linear function. The output attributes of all the problems were encoded using a *1-of-m* output representation for m classes. The output with the highest activation designated the class. The aim of this section is to study the difference between negative correlation learning and independent training, rather than to compare negative correlation learning with previous work. The experiments used such a single train-and-test partition.

The ensemble architecture used in the experiments has 4 networks. Each individual network is a feedforward network with one hidden layer. Both hidden node function and output node function are defined by the logistic function in Eq.(18). All the individual networks have 10 hidden nodes. The number of training

epochs was set to 250. The strength parameter λ was set to 1.0. These parameters were chosen after limited preliminary experiments. They are not meant to be optimal.

Experimental Results

Table 5 shows the average results of negative correlation learning over 25 runs. Each run of negative correlation learning was from different initial weights. The ensemble with the same initial weight setup was also trained using BP without the correlation penalty terms (i.e., $\lambda = 0.0$ in negative correlation learning). Results are also shown in Table 5. For this problem, the simple averaging defined in Eq.(9) was first applied to decide the output of the ensemble. For the simple averaging, it was surprising that the results of negative correlation learning with $\lambda = 1.0$ were similar to those of independent training. This phenomenon seems contradictory to the claim that the effect of the correlation penalty term is to encourage different individual networks in an ensemble to learn different parts or aspects of the training data. In order to verify and quantify this claim, we compared the outputs of the individual networks trained with the correlation penalty terms to those of the individual networks trained without the correlation penalty terms.

Table 5: Comparison of error rates between negative correlation learning ($\lambda = 1.0$) and independent training (i.e., $\lambda = 0.0$ in negative correlation learning) on the Australian credit card assessment problem; the results were averaged over 25 runs. “Simple Averaging” and “Winner-Takes-All” indicate two different combination methods used in negative correlation learning, Mean, SD, Min and Max indicate the mean value, standard deviation, minimum and maximum value, respectively

	Error Rate	Simple Averaging	Winner-Takes-All
$\lambda = 1.0$	Mean	0.1337	0.1195
	SD	0.0068	0.0052
	Min	0.1163	0.1105
	Max	0.1454	0.1279
$\lambda = 0.0$	Mean	0.1368	0.1384
	SD	0.0048	0.0049
	Min	0.1279	0.1279
	Max	0.1454	0.1512

Table 6: The sizes of the correct response sets of individual networks created respectively by negative correlation learning ($\lambda = 1.0$) and independent training (i.e., $\lambda = 0.0$ in negative correlation learning) on the testing set and the sizes of their intersections for the Australian credit card assessment problem; the results were obtained from the first run among the 25 runs

$\lambda = 1.0$			$\lambda = 0.0$		
$\Omega_1 = 147$	$\Omega_2 = 143$	$\Omega_3 = 138$	$\Omega_1 = 149$	$\Omega_2 = 147$	$\Omega_3 = 148$
$\Omega_4 = 143$	$\Omega_{12} = 138$	$\Omega_{13} = 124$	$\Omega_4 = 148$	$\Omega_{12} = 147$	$\Omega_{13} = 147$
$\Omega_{14} = 141$	$\Omega_{23} = 116$	$\Omega_{24} = 133$	$\Omega_{14} = 147$	$\Omega_{23} = 147$	$\Omega_{24} = 146$
$\Omega_{34} = 123$	$\Omega_{123} = 115$	$\Omega_{124} = 133$	$\Omega_{34} = 146$	$\Omega_{123} = 147$	$\Omega_{124} = 146$
$\Omega_{134} = 121$	$\Omega_{234} = 113$	$\Omega_{1234} = 113$	$\Omega_{134} = 146$	$\Omega_{234} = 146$	$\Omega_{1234} = 146$

Two notions were introduced to analyse negative correlation learning. They are the correct response sets of individual networks and their intersections. The correct response set S_i of individual network i on the testing set consists of all the patterns in the testing set which are classified correctly by the individual network i . Let Ω_i denote the size of set S_i , and $\Omega_{i_1 i_2 \dots i_k}$ denote the size of set $S_{i_1} \cap S_{i_2} \cap \dots \cap S_{i_k}$. Table 6 shows the sizes of the correct response sets of individual networks and their intersections on the testing set, where the individual networks were respectively created by negative correlation learning and independent training. It is evident from Table 6 that different individual networks created by negative correlation learning were able to specialise to different parts of the testing set. For instance, in Table 6 the sizes of both correct response sets S_2 and S_4 at $\lambda = 1.0$ were 143, but the size of their intersection $S_2 \cap S_4$ was 133. The size of $S_1 \cap S_2 \cap S_3 \cap S_4$ was only 113. In contrast, the individual networks in the ensemble created by independent training were quite similar. The sizes of correct response sets S_1 , S_2 , S_3 and S_4 at $\lambda = 0.0$ were from 147 to 149, while the size of their intersection set $S_1 \cap S_2 \cap S_3 \cap S_4$ reached 146. There were only three different patterns correctly classified by the four individual networks in the ensemble.

In simple averaging, all the individual networks have the same combination weights and are treated equally. However, not all the networks are equally important. Because different individual networks created by negative correlation learning were able to specialise to different parts of the testing set, only the outputs of these specialists should be considered to make the final decision about the ensemble for this part of the testing set. In this experiment, a winner-takes-all method was applied to select such networks. For each pattern of the testing set,

the output of the ensemble was only decided by the network whose output had the highest activation. Table 5 shows the average results of negative correlation learning over 25 runs using the winner-takes-all combination method. The winner-takes-all combination method improved negative correlation learning significantly because there were good and poor networks for each pattern in the testing set, and winner-takes-all selected the best one. However, it did not improve the independent training much because the individual networks created by the independent training were all similar to each other.

CONCLUSIONS

This chapter describes negative correlation learning for designing neural network ensembles. It can be regarded as one way of decomposing a large problem into smaller and specialised ones, so that each sub-problem can be dealt with by an individual neural network relatively easily. A correlation penalty term in the error function was proposed to minimise mutual information and encourage the formation of specialists in the ensemble.

Negative correlation learning has been analysed in terms of mutual information on a regression task in the different noise conditions. Unlike independent training which creates larger mutual information among the ensemble, negative correlation learning can produce smaller mutual information among the ensemble. Through minimisation of mutual information, very competitive results have been produced by negative correlation learning in comparison with independent training.

This chapter compares the decision boundaries and the correct response sets constructed by negative correlation learning and the independent training for two pattern classification problems. The experimental results show that negative correlation learning has a very good classification performance. In fact, the decision boundary formed by negative correlation learning is nearly close to the optimum decision boundary generated by the Bayes classifier.

There are, however, some issues that need resolving. No special considerations were made in optimisation of the size of the ensemble and strength parameter λ in this chapter. Evolutionary ensembles with negative correlation learning for optimisation of the size of the ensemble had been studied on the classification problems (Liu, Yao & Higuchi, 2000).

REFERENCES

- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Appl. Comp.*, AC-19, 716-723.

- Clemen, R. T., & Winkler, R. L. (1985). Limits for the precision and value of information from dependent sources. *Operations Research*, 33:427-442.
- Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y. & Vapnik, V. (1994). Boosting and other ensemble methods. *Neural Computation*, 6:1289-1301.
- Drucker, H., Schapire, R. & Simard, P. (1993). Improving performance in neural networks using a boosting algorithm. In Hanson, S. J., Cowan, J. D. & Giles, C. L. (Eds.), *Advances in Neural Information Processing Systems 5*, pp. 42-49. San Mateo, CA: Morgan Kaufmann.
- Friedman, J. H. (1994). An overview of predictive learning and function approximation. In V. Cherkassky, J. H. Friedman, and H. Wechsler, (Eds.), *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, pp. 1-61. Springer-Verlag, Heidelberg, Germany.
- Hansen, L. K. & Salamon, P. (1990). Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(10):993-1001.
- Jacobs, R. A. (1997). Bias/variance analyses of mixture-of-experts architectures. *Neural Computation*, 9:369-383.
- Jacobs, R. A. & Jordan, M. I. (1991). A competitive modular connectionist architecture. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, (Eds.), *Advances in Neural Information Processing Systems 3*, pp. 767-773. Morgan Kaufmann, San Mateo, CA.
- Jacobs, R. A., Jordan, M. I. & Barto, A. G. (1991). Task decomposition through competition in a modular connectionist architecture: the what and where vision task. *Cognitive Science*, 15:219-250.
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J. & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79-87.
- Jordan, M. I. & Jacobs, R. A. (1994). Hierarchical mixtures-of-experts and the em algorithm. *Neural Computation*, 6:181-214.
- Liu, Y. & Yao, X. (1998a). Negatively correlated neural networks can produce best ensembles. *Australian Journal of Intelligent Information Processing Systems*, 4:176-185.
- Liu, Y. & Yao, X. (1998b). A cooperative ensemble learning system. In *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98)*, pages 2202-2207. IEEE Press, Piscataway, NJ, USA.
- Liu, Y. & Yao, X. (1999). Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):716-725.
- Liu, Y., Yao, X., & Higuchi, T. (2000). Evolutionary ensembles with negative correlation learning. *IEEE Trans. on Evolutionary Computation*, 4(4):380-725.

- Nilsson, N. J. (1965). *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. New York: McGraw Hill.
- Opitz, D. W. & Shavlik, J. W. (1996). Actively searching for an effective neural network ensemble. *Connection Science*, 8:337-353.
- Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465-471.
- Rosen, B. E. (1996). Ensemble learning using decorrelated neural networks. *Connection Science*, 8:373-383.
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. I, pp. 318-362. MIT Press, Cambridge, MA.
- Sarkar, D. (1996). Randomness in generalization ability: A source to improve it. *IEEE Trans. on Neural Networks*, 7(3):676-685.
- Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5:197-227.
- Selfridge, O. G. (1958). Pandemonium: a paradigm for learning. *Mechanisation of Thought Processes: Proceedings of a Symp.* Held at the National Physical Lab., pp. 513-526. HMSO, London.
- Sharkey, A. J. C. (1996). On combining artificial neural nets. *Connection Science*, 8:299-313.
- van der Lubbe, J. C. A. (1997). *Information Theory*. Cambridge: Cambridge University Press.
- van der Lubbe, J. C. A. (1999). *Information Theory*. (2nd ed) Prentice-Hall International, Inc.
- Wallace, C. S., & Patrick, J. D. (1991). *Coding Decision Trees*. Technical Report 91/153, Department of Computer Science, Monash University, Clayton, Victoria 3168, Australia, August.
- Wolpert, D. H. (1990). A mathematical theory of generalization. *Complex Systems*, 4:151-249.

Chapter II

A Perturbation Size-Independent Analysis of Robustness in Neural Networks by Randomized Algorithms

C. Alippi
Politecnico di Milano, Italy

ABSTRACT

This chapter presents a general methodology for evaluating the loss in performance of a generic neural network once its weights are affected by perturbations. Since weights represent the “knowledge space” of the neural model, the robustness analysis can be used to study the weights/performance relationship. The perturbation analysis, which is closely related to sensitivity issues, relaxes all assumptions made in the related literature, such as the small perturbation hypothesis, specific requirements on the distribution of perturbations and neural variables, the number of hidden units and a given neural structure. The methodology, based on Randomized Algorithms, allows reformulating the computationally intractable problem of robustness/sensitivity analysis in a probabilistic framework characterised by a polynomial time solution in the accuracy and confidence degrees.

INTRODUCTION

The evaluation of the effects induced by perturbations affecting a neural computation is relevant from the theoretical point of view and in developing an embedded device dedicated to a specific application.

In the first case, the interest is in obtaining a reliable and easy to be generated measure of the performance loss induced by perturbations affecting the weights of a neural network. The relevance of the analysis is obvious since weights characterise the “knowledge space” of the neural model and, hence, its inner nature. In this direction, a study of the evolution of the network’s weights over training time allows for understanding the mechanism behind the generation of the knowledge space. Conversely, the analysis of a specific knowledge space (fixed configuration for weights) provides hints about the relationship between the weights space and the performance function. The latter aspect is of primary interest in recurrent neural networks where even small modifications of the weight values are critical to performance (e.g., think of the stability of an intelligent controller comprising a neural network and issues leading to robust control).

The second case is somehow strictly related to the first one and covers the situation where the neural network must be implemented in a physical device. The optimally trained neural network becomes the “golden unit” to be implemented within a finite precision representation environment as it happens in mission-critical applications and embedded systems. In these applications, behavioural perturbations affecting the weights of a neural network abstract uncertainties associated with the implementation process, such as finite precision representations (e.g., truncation or rounding in a digital hardware, fixed or low resolution floating point representations), fluctuations of the parameters representing the weights in analog solutions (e.g., associated with the production process of a physical component), ageing effects, or more complex and subtle uncertainties in mixed implementations.

The sensitivity/robustness issue has been widely addressed in the neural network community with a particular focus on specific neural topologies.

More in detail, when the neural network is composed of linear units, the analysis is straightforward and the relationship between perturbations and the induced performance loss can be obtained in a closed form (Alippi & Briozzo, 1998). Conversely, when the neural topology is non-linear, which is mostly the case, several authors assume the small perturbation hypothesis or particular hypothesis about the stochastic nature of the neural computation. In both cases, the assumptions make the mathematics more amenable with the positive consequence that a relationship between perturbations and performance loss can be derived (e.g., see Alippi & Briozzo, 1998; Pichè, 1995). Unfortunately, these analyses introduce hypotheses which are not always satisfied in all real applications.

Another classic approach requires expanding with Taylor the neural computation around the nominal value of the trained weights. A subsequent linearised analysis follows which allows for solving the sensitivity issue (e.g., Pichè, 1995). Anyway, the validity of such approaches depend, in turn, on the validity of the small perturbation hypothesis: how to understand a priori if a perturbation is small for a given application?

In other applications the small perturbation hypothesis cannot be accepted being the involved perturbations everything but small. As an example we have the development of a digital embedded system. There, the designer has to reduce as possible the dimension of the weights by saving bits; this produces a positive impact on cost, memory size and power consumption of the final device.

Differently, other authors avoid the small perturbation assumption by focusing the attention on very specific neural network topologies and/or introducing particular assumptions regarding the distribution of perturbations, internal neural variables and inputs (Stevenson, Winter & Widrow, 1990; Alippi, Piuri & Sami, 1995).

Other authors have considered the sensitivity analysis under the small perturbation hypothesis to deal with implementation aspects. In this case, perturbations are specifically related to finite precision representations of the interim variables characterising the neural computation (Holt & Hwang, 1993; Dunder & Rose, 1995).

Differently from the limiting approaches provided in the literature, this chapter suggests a robustness/sensitivity analysis in the large, i.e., without assuming constraints on the size or nature of the perturbation; as such, small perturbation situations become only a subcase of the theory. The analysis is general and can be applied to all neural topologies, both static and recurrent in order to quantify the performance loss of the neural model when perturbations affect the model's weights.

The suggested sensitivity/robustness analysis can be applied to *All* neural network models involved in system identification, control signal/image processing and automation-based applications without any restriction. In particular, the analysis allows for solving the following problems:

- Quantify the robustness of a generically trained neural network by means of a suitable, easily to be computed and reliable robustness index;
- Compare different neural networks, solving a given application by ranking them according to their robustness;
- Investigate the criticality of a recurrent model (“stability” issue) by means of its robustness index;

- Study the efficacy and effectiveness of techniques developed to improve the robustness degree of a neural network by inspecting the improvement in robustness.

The key elements of the perturbation analysis are Randomised Algorithms—RAs- (Vidyasagar, 1996, 1998; Tempo & Dabbene, 1999; Alippi, 2002), which transform the computationally intractable problem of evaluating the robustness of a generic neural network with respect to generic, continuous perturbations, in a tractable problem solvable with a polynomial time algorithm by resorting to probability.

The increasing interest and the extensive use of Randomised Algorithms in control theory, and in particular in the robust control area (Djavan, Tulleken, Voetter, Verbruggen & Olsder, 1989; Battarcharyya, Chapellat & Keel, 1995; Bai & Tempo, 1997; Chen & Zhou, 1997; Vidyasagar, 1998; Tempo & Dabbene, 1999, Calafiore, Dabbene & Tempo, 1999), make this versatile technique extremely interesting also for the neural network researcher.

We suggest the interested reader to refer to Vidyasagar (1998) and Tempo and Dabbene (1999) for a deep analysis of the use of RAs in control applications; the author forecasts an increasing use of Randomised Algorithms in the analysis and synthesis of intelligent controllers in the neural network community.

The structure of the chapter is as follows. We first formalise the concept of robustness by identifying a natural and general index for robustness. Randomised Algorithms are then briefly introduced to provide a comprehensive analysis and adapted to estimate the robustness index. Experiments then follow to shed light on the use of the theory in identifying the robustness index for static and recurrent neural models.

A GENERAL ROBUSTNESS/SENSITIVITY ANALYSIS FOR NEURAL NETWORKS

In the following we consider a *generic* neural network implementing the $\hat{y} = f(\hat{\theta}, x)$ function where $\hat{\theta}$ is the weight (and biases) vector containing all the trained free parameters of the neural model.

In several neural models, and in particular in those related to system identification and control, the relationship between the inputs and the output of the system are captured by considering a regressor vector ϕ , which contains a limited time-window of actual and past inputs, outputs and possibly predicted outputs.

Of particular interest, in the zoo of neural models, are those which can be represented by means of the model structures $\hat{y}(t) = f(\varphi)$ where function $f(\cdot)$ is a regression-type neural network, characterised by N_φ inputs, N_h non-linear hidden units and a single effective linear/non-linear output (Ljung, 1987; Hertz, Krog & Palmer, 1991; Hassoun, 1995; Ljung, Sjöberg & Hjalmarsson, 1996).

The absence/presence of a dynamic in the system can be modelled by a suitable number of delay elements (or time lags), which may affect inputs (time history on external inputs u) system outputs (time history on $y(t)$) on predicted outputs (time history on $\hat{y}(t)$) or residuals (time history on $e(t) = \hat{y}(t) - y(t)$). Where it is needed $y(t)$, $\hat{y}(t)$ and $e(t)$ are vectorial entities, a component for each independent distinct variable.

Several neural model structures have been suggested in the literature, which basically differ in the regressor vector. Examples are, NARMAX and NOE topologies. NARMAX structure can be obtained by considering both past inputs and outputs of the system to infer $y(t)$. We have:

$$\varphi = [u(t), u(t-1), \dots, u(t-n_u), y(t-1), \dots, y(t-n_y), \dots, e(t-1), \dots, e(t-n_e)]$$

Differently, the NOE structure processes only past inputs and predicted outputs, i.e.:

$$\varphi = [u(t), u(t-1), \dots, u(t-n_u), \hat{y}(t-1), \dots, \hat{y}(t-n_y)]$$

Static neural networks, such as classifiers, can be obtained by simply considering external inputs:

$$\varphi = [u(t), u(t-1), \dots, u(t-n_u)]$$

Of course, different neural models can be considered, e.g., fully recurrent and well fit with the suggested robustness analysis.

A general, perturbation size independent, model-independent robustness analysis requires the evaluation of the loss in performance induced by a generic perturbation, in our analysis affecting the weights of a generic neural network. We denote by $y_\Delta(x) = f_\Delta(\theta, \Delta, x)$ the mathematical description of the perturbed computation and by $\Delta \in D \subseteq \mathbb{R}^p$ a generic p -dimensional perturbation vector, a component for each independent perturbation affecting the neural computation $\hat{y}(t)$. The perturbation space D is characterised in stochastic terms by providing the probability density function pdf_D .

To measure the discrepancy between $y_\Delta(x)$ and $y(t)$ or $\hat{y}(t)$, we consider a generic loss function $U(\Delta)$. In the following we only assume that such performance loss function is measurable according to Lebesgue with respect to D . Lebesgue measurability for $U(\Delta)$ allows us for taking into account an extremely large class of loss functions.

Common examples for U are the Mean Square Error—MSE—loss functions

$$U(\Delta) = \frac{1}{N_x} \sum_{i=1}^{N_x} (\hat{y}(x_i) - \hat{y}(x_i, \Delta))^2 \quad \text{and} \quad U(\Delta) = \frac{1}{N_x} \sum_{i=1}^{N_x} (y(x_i) - \hat{y}(x_i, \Delta))^2. \quad (1)$$

More specifically, (1)-left compares the perturbed network with \hat{y} , which is supposed to be the “golden” error-free unit while (1)-right estimates the performance of the error-affected (perturbed) neural network (generalisation ability of the perturbed neural model).

The formalisation of the impact of perturbation on the performance function can be simply derived:

Definition: Robustness Index

We say that a neural network is robust at level $\bar{\gamma}$ in D , when the robustness index $\bar{\gamma}$ is the minimum positive value for which

$$U(\Delta) \leq \bar{\gamma}, \forall \Delta \in D, \forall \gamma \geq \bar{\gamma} \quad (2)$$

Immediately, from the definition of robustness index we have that a generic neural network NN_1 is more robust than NN_2 if $\bar{\gamma}_1 < \bar{\gamma}_2$ and the property holds independently from the topology of the two neural networks.

The main problem related to the determination of the robustness index $\bar{\gamma}$ is that we have to compute $U(\Delta)$, $\forall \Delta \in D$ if we wish a tight bound. The $\bar{\gamma}$ -identification problem is therefore intractable from a computational point of view if we relax all assumptions made in the literature as we do.

To deal with the computational aspect we associate a dual probabilistic problem to (2):

Robustness Index: Dual Problem We say that a neural network is robust at level $\bar{\gamma}$ in D with confidence η , when $\bar{\gamma}$ is the minimum positive value for which

$$\Pr(U(\Delta) \leq \bar{\gamma}) \geq \eta \quad \text{holds} \quad \forall \Delta \in D, \forall \gamma \geq \bar{\gamma} \quad (3)$$

The probabilistic problem is weaker than the deterministic one since it tolerates the existence of a set of perturbations (whose measure according to Lebesgue is $1 - \eta$) for which $u(\Delta) > \bar{\gamma}$. In other words, not more than 100η % of perturbations $\Delta \in D$ will generate a loss in performance larger than $\bar{\gamma}$.

Probabilistic and deterministic problems are “close” to each other when we choose, as we do, $\eta = 1$. Note that $\bar{\gamma}$ depends only on the size of D and the neural network structure.

The non-linearity with respect to Δ and the lack of a priori assumptions regarding the neural network do not allow computing (2) in a closed form for the general perturbation case. The analysis, which would imply testing $U\Delta$ in correspondence with a continuous perturbation space, can be solved by resorting to probability according to the dual problem and by applying Randomised Algorithms to solve the robustness/sensitivity problem.

RANDOMIZED ALGORITHMS AND PERTURBATION ANALYSIS

In this paragraph we briefly review the theory behind Randomised Algorithms and adapt them to the robustness analysis problem.

In the following we denote by $p_\gamma = \Pr\{U(\Delta) \leq \gamma\}$ the probability that the loss in performance associated with perturbations in D is below a given—but arbitrary—value γ .

Probability p_γ is unknown, cannot be computed in a close form for a generic U function and neural network topology, and its evaluation requires exploration of the whole perturbation space D .

The unknown probability p_γ can be estimated by sampling D with N independent and identically distributed samples Δ_i ; extraction must be carried out according to the *pdf* of the perturbation.

For each sample Δ_i we then generate the triplet

$$\{\Delta_i, U(\Delta_i), I(\Delta_i)\}_{i=1, N} \text{ where } I(\Delta_i) = \begin{cases} 1 & \text{if } U(\Delta_i) \leq \gamma \\ 0 & \text{if } U(\Delta_i) > \gamma \end{cases} \quad (4)$$

The true probability p_γ can now simply be estimated as

$$\hat{p}_N = \frac{1}{N} \sum_{i=1}^N I(\Delta_i) \quad (5)$$

Of course, when N tends to infinity, \hat{p}_N converges to p_γ . Conversely, on a finite data set of cardinality N , the discrepancy between \hat{p}_N and p_γ exists and can be

simply measured as $|p_\gamma - \hat{p}_N|$. $|p_\gamma - \hat{p}_N|$ is a random variable which depends on the particular extraction of the N samples since different extractions of N samples from D will provide different estimates for \hat{p}_N . By introducing an accuracy degree ε on $|p_\gamma - \hat{p}_N|$ and a confidence level $1 - \delta$ (which requests that the $|p_\gamma - \hat{p}_N| \leq \varepsilon$ inequality is satisfied at least with probability $1 - \delta$), our problem can be formalised by requiring that the inequality

$$\Pr\{|p_\gamma - \hat{p}_N| \leq \varepsilon\} \geq 1 - \delta \quad (6)$$

is satisfied for $\forall \gamma \geq 0$. Of course, we wish to control the accuracy and the confidence degrees of (6) by allowing the user to choose the most appropriate values for the particular need. Finally, by extracting a number of samples from D according to the Chernoff inequality (Chernoff, 1952)

$$N \geq \frac{\ln \frac{2}{\delta}}{2\varepsilon^2} \quad (7)$$

we have that $\Pr\{|p_\gamma - \hat{p}_N| \leq \varepsilon\} \geq 1 - \delta$ holds for $\forall \gamma \geq 0, \forall \delta, \varepsilon \in [0, 1]$.

As an example, by considering 5% in accuracy and 99% in confidence, we have to extract 1060 samples from D ; with such choice we can approximate p_γ with \hat{p}_N introducing the maximum error 0.05 ($\hat{p}_N - 0.05 \leq p_\gamma \leq \hat{p}_N + 0.05$) and the inequality holds at least with the probability 0.99.

Other bounds can be considered instead of the Chernoff's one as suggested by Bernoulli and Bienaymè, (e.g., see Tempo & Dabbene, 1999). Nevertheless, the Chernoff's bound improves upon the others and, therefore, should be preferred if we wish to keep minimal the number of samples to be extracted. The Chernoff bound grants that:

- N is independent from the dimension of D (and hence it does not depend on the number of perturbations we are considering in the neural network);
- N is linear in $\ln \frac{1}{\delta}$ and $\frac{1}{\varepsilon^2}$ (hence it is polynomial in the accuracy and confidence degrees).

As a consequence, the dual probabilistic problem related to the identification of the robustness index $\bar{\gamma}$ can be solved with randomised algorithms and therefore with a polynomial complexity in the accuracy and the confidence degrees independently from the number of weights of the neural model network. In fact, by expanding the (6) we have that

$$\Pr\{p_\gamma - \hat{p}_N \leq \varepsilon\} \geq 1 - \delta \equiv \Pr\left\{\left|\Pr(u(\Delta) \leq \gamma) - \frac{1}{N} \sum_i I(\Delta_i)\right| \leq \varepsilon\right\} \geq 1 - \delta \quad (8)$$

If accuracy ε and confidence δ are small enough, we can confuse p_γ and \hat{p}_N by committing a small error. As a consequence, the dual probabilistic problem requiring $p_\gamma \geq \eta$ becomes $\hat{p}_N \geq \eta$. We surely assume ε and δ to be small enough in subsequent derivations.

The final algorithm, which allows for testing the robustness degree $\bar{\gamma}$ of a neural network, is:

1. *Select ε and δ sufficiently small to have enough accuracy and confidence.*
2. *Extract from D , according to its pdf, a number of perturbations N as suggested by (7).*
3. *Generate the indicator function $I(\Delta)$ and generate the estimate $\hat{p}_N = \hat{p}_N(\gamma)$ according to (5).*
4. *Select the minimum value γ_η from the $\hat{p}_N = \hat{p}_N(\gamma)$ function so that $\hat{p}_N(\gamma_\eta) = 1$ is satisfied $\forall \gamma \geq \gamma_\eta$. γ_η is the estimate of the robustness index $\bar{\gamma}$.*

Note that with a simple algorithm we are able to estimate in polynomial time the robustness degree $\bar{\gamma}$ of a generic neural network. The accuracy in estimating $\bar{\gamma}$ can be made arbitrarily good at the expense of a larger number of samples as suggested by Chernoff's bound.

APPLYING THE METHODOLOGY TO STUDY THE ROBUSTNESS OF NEURAL NETWORKS

In the experimental section we show how the robustness index for neural networks can be computed and how it can be used to characterise a neural model. After having presented and experimentally justified the theory supporting Randomised Algorithms, we will focus on the following problems:

- test the robustness of a given static neural network (robustness analysis);
- study the relationships between the robustness of a static neural network and the number of hidden units (structure redundancy);
- analyse the robustness of recurrent neural networks (robustness/stability analysis).

In the following experiments we consider perturbations affecting weights and biases of a neural network defined in D and subject to uniform distributions. Here,

a perturbation Δ_i affecting a generic weight w_i must be intended as a relative perturbation with respect to the weight magnitude according to the multiplicative perturbation model $w_{i,p} = w_i(1 + \Delta_i)$, $\forall i = 1, n$. A $t\%$ perturbation implies that Δ_i is drawn from a symmetrical uniform distribution of extremes

$$\left[-\frac{t}{100}, \frac{t}{100} \right];$$

a 5% perturbation affecting weights and biases composing vector $\hat{\theta}$ implies that each weight/bias is affected by an independent perturbation extracted from the $[-0.05, 0.05]$ interval and applied to the nominal value according to the multiplicative perturbation model.

Experiment 1: *The impact of ϵ , δ and N on the evaluation of the robustness index*

The reference application to be learned is the simple error-free function

$$y = -x \cdot \sin(x^2) + \frac{e^{-0.23 \cdot x}}{1 + x^4}, \quad x \in [-3, 3]$$

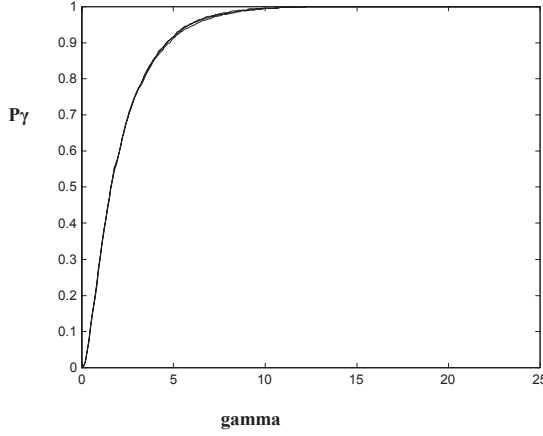
A set of 41 training data have been extracted from the function domain according to a uniform distribution. We considered static feedforward neural networks with hidden units characterised by a hyperbolic tangent activation function and a single linear output. Training was accomplished by considering a Levenberg-Marquardt algorithm applied to an MSE training function; a test set was considered during the training phase to determine the optimal stopping point so as to monitor the upsurge of overfitting effects.

We discovered that all neural networks with at least 6 hidden units are able to solve the function approximation task with excellent performance.

In this experiment we focus the attention on the neural network characterised by 10 hidden units. After training we run the robustness algorithm by considering 7.5% of perturbations (weights are affected by perturbations up to 7.5% of their magnitude) and we chose $\epsilon = 0.02$ and $\delta = 0.01$ from which we have to extract $N = 6624$ samples from D . We carried out three extractions of N samples and, for each set, we computed the related $\hat{p}_N = \hat{p}_N(\gamma)$ curve.

The $\hat{p}_N = \hat{p}_N(\gamma)$ curves are given in Figure 1. As we can see the curves are very close to each other. In fact, we know from the theory, that the estimated probability belongs to a neighbourhood of the true one according to the

Figure 1: $\hat{p}_N = \hat{p}_N(\gamma)$ for three different runs



$\hat{p}_N - 0.02 \leq p_\gamma \leq \hat{p}_N + 0.02$ relationship. A single curve is therefore enough to characterise the robustness of the envisaged neural network and there is no need to consider multiple runs. By inspecting Figure 1 we obtain that the estimate of the robustness index $\bar{\gamma}$ is $\gamma_\eta = 11$ which implies that $U(\Delta) \leq 11, \forall \Delta \in D$ with high probability.

We wish now to study the impact of N on $\hat{p}_N = \hat{p}_N(\gamma)$ by considering three runs with different ε and δ according to Table 1.

The $\hat{p}_N = \hat{p}_N(\gamma)$ curves are given in Figure 2. It is interesting to note, at least for the specific application, that even with low values of N , the estimates for $\hat{p}_N = \hat{p}_N(\gamma)$ and γ_η are reasonable and not far from each other. We should anyway extract the number of samples according to Chernoff's inequality.

Experiment 2: Testing the robustness of a given neural network

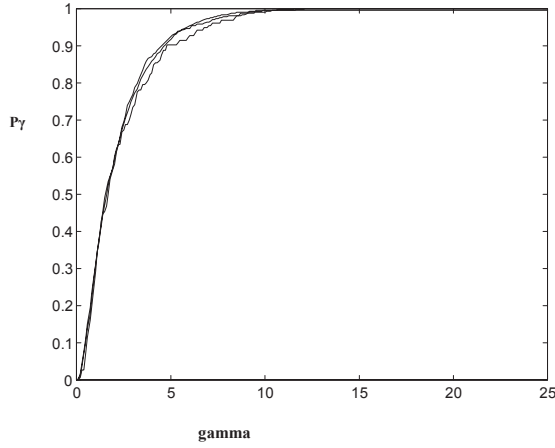
In the second experiment we test the robustness of the 10 hidden units network by considering its behaviour once affected by stronger perturbations (larger D) and, in particular, for perturbations 1%, 3%, 5%, 10%, 30%. We selected $\varepsilon=0.02$ and $\delta=0.01$.

The $\hat{p}_\gamma = \hat{p}_\gamma(\gamma)$ function corresponding to the different perturbations is given in Figure 3.

Table 1: ε , δ and N

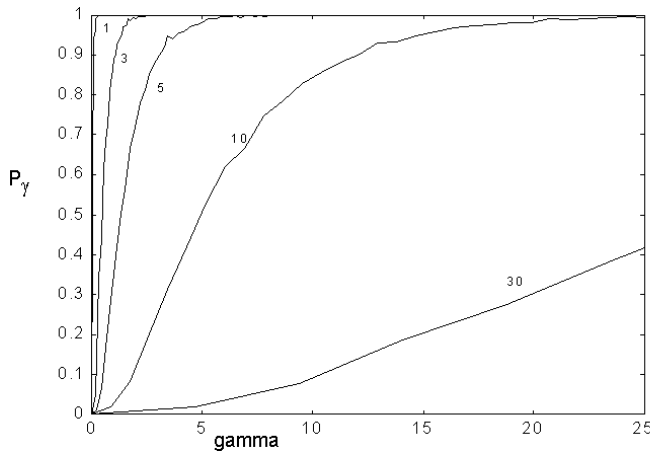
ε	δ	N
0.02	0.01	6624
0.05	0.05	738
0.1	0.1	150

Figure 2: $\hat{p}_N = \hat{p}_N(\gamma)$ for different runs with parameters given in Table 1



Again, from its definition, $\bar{\gamma}$ is the smallest value for which $\hat{p}_\gamma = 1$, $\gamma \geq \bar{\gamma}$; as an example, if we consider the 5% perturbation case, $\bar{\gamma}$ assumes a value around 7. It is obvious, but interesting to point out that, by increasing the strength of perturbation (i.e., by enlarging the extremes of the uniform distribution characterising the *pdf* of D), $\bar{\gamma}$ increases. In fact, stronger perturbations have a worse impact on the performance loss function since the error-affected neural network diverges from the error-free one. Conversely, we see that small perturbations, e.g., the 1% one, induce a very small loss in performance since the robustness index γ_η is very small.

Figure 3: \hat{p}_γ as a function of γ for the 10 hidden units neural network



Experiment 3: Testing the robustness of a hierarchy of performance-equivalent neural networks

Once we have identified the robustness degree of a neural network solving an application, we can investigate whether it is possible to improve the robustness degree of the application by considering a sort of structural redundancy or not. This issue can be tackled by considering the neural hierarchy $M: M_1 \subseteq M_2 \dots \subseteq M_k \dots$ where M_k represents a neural network with k hidden units.

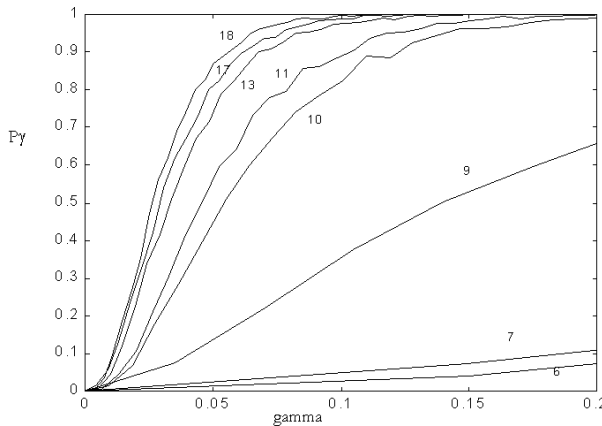
To this end, we consider a set of performance-equivalent neural networks, each of which is able to solve the application with a performance tolerable by the user. All neural networks are characterised by a different topological complexity (number of hidden units).

The $\hat{p}_\gamma = \hat{p}_\gamma(\gamma)$ curves parameterised in the number of hidden units are given in Figure 4 in the case of 1% perturbation. We can see that by increasing the number of hidden units, $\bar{\gamma}$ decreases. We immediately realise that neural networks with a reduced number of hidden units are, for this application, less robust than the ones possessing more degrees of freedom. Large networks provide, in a way, a sort of spatial redundancy: information characterising the knowledge space of the neural networks is distributed over more degrees of freedom.

We discovered cases where a larger neural network was less robust than a smaller one: in such a case probably the complex model degenerates into a simpler one.

The evolution of $\bar{\gamma}$ over the number of hidden units parameterised with respect to the different perturbations 5%, 10% and 30% is given in Figure 5. We note that the minimal network, namely the smallest network able to solve the application, is not the more robust one for this application (in fact it possesses large values for the $\bar{\gamma}$ s).

Figure 4: \hat{p}_γ over γ and parameterised in the number of hidden units



This trend—verified also with other applications—suggests that the robustness degree of the neural network improves on the average by increasing the number of hidden units (spatial redundancy). Anyway, with a small increase in the topological complexity (e.g., by considering the 13 hidden units model instead of the 6 one), we obtain a significant improvement according to the robustness level. There is no need to consider more complex neural networks since the improvement in robustness is small.

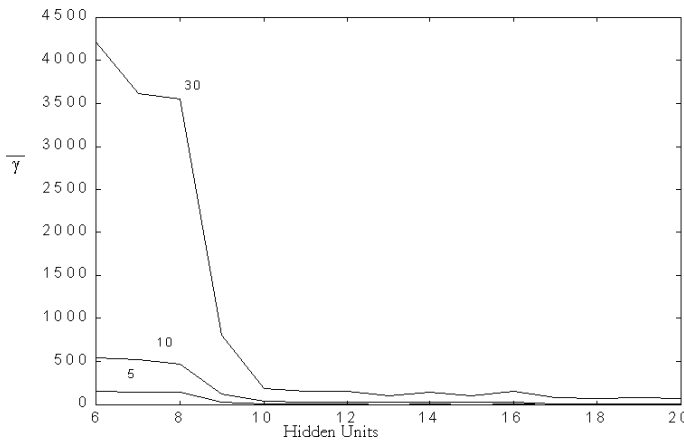
Experiment 4: Testing the robustness of recurrent neural networks

The goal of the last experiment is to study the robustness/stability of recurrent neural networks with the suggested theory. The chosen application refers to the identification of the open-loop stable, nonlinear, continuous system suggested in Norgaard (2000). The input and the corresponding output sequence of the system to be identified is given in Figure 6.

We first considered an NOE recurrent neural network with 5 hidden units characterised by the regressor vector $\varphi = [u(t-1), u(t-2), \hat{y}(t-1), \hat{y}(t-2)]$. The non-linear core of the neural network is a static regression type neural network as the one considered in the function approximation experiments. The topology of the NOE network is given in figure 7.

Once trained, the network we applied the methodology to estimates the robustness of the recurrent model. The $\hat{p}_\gamma = \hat{p}_\gamma(\gamma)$ curve, evaluated with $\varepsilon = \delta = 0.05$, for the 0.1% perturbation case is given in Figure 8. As we could have expected, differently from the static function approximation application, the recurrent NOE neural network is sensitive even to small perturbations affecting the knowledge space of the network.

Figure 5: $\bar{\gamma}$ as function of the hidden units, $\varepsilon = 0.04$, $\delta = 0.01$



We identified the dynamic system with a NARMAX neural network characterised by 5 hidden units and the structure given in Figure 9. For such topology we selected the regressor vector

$$\varphi = [u(t-1), u(t-2), y(t-1), y(t-2), e(t-1), e(t-2)]$$

Figure 10 shows the $\hat{p}_\gamma = \hat{p}_\gamma(\gamma)$ curve. It is interesting to note that the NARMAX neural network is less robust than the corresponding NOE model. The basic reason for such behaviour is due to the fact that the recurrent model does not receive directly as input the fed-back network output but only the residual e .

During training the NOE model must somehow learn more deeply the concept of stability since even small variations of weights associated with the training phase weights update would produce a trajectory diverging from the system output to be mimicked. This effect is due to the pure fed-back structure of the NOE model which

Figure 6: The input and the corresponding output of the dynamic system

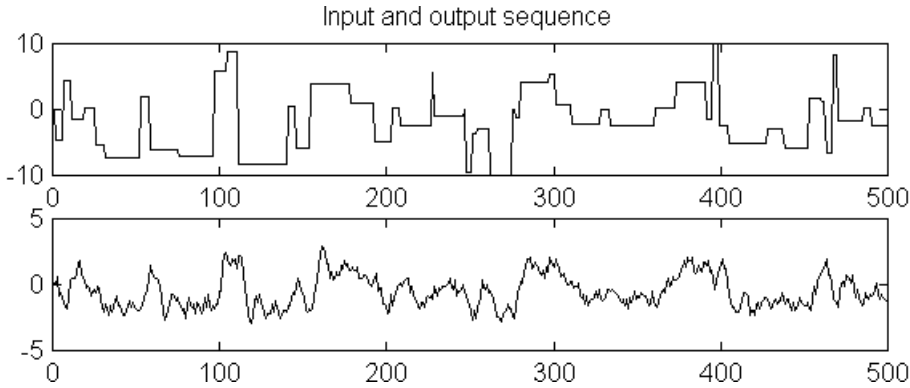


Figure 7: The considered NOE neural network

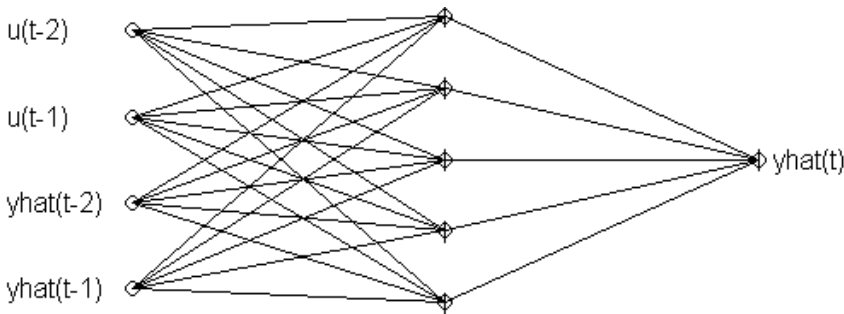


Figure 8: The \hat{p}_γ function for the NOE neural network

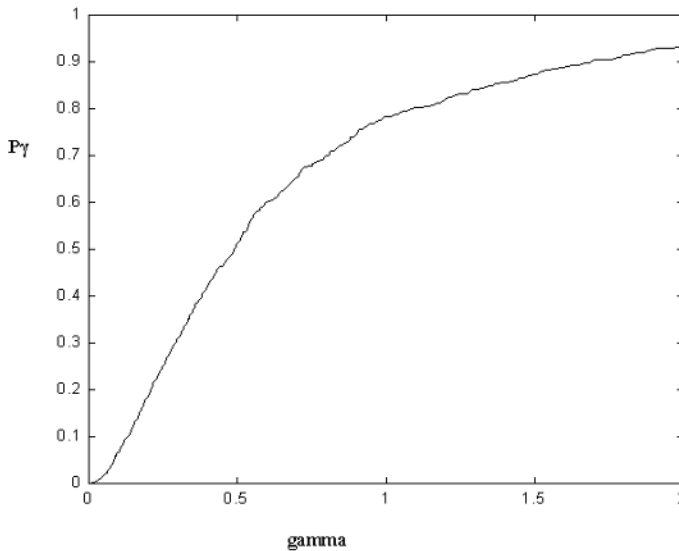
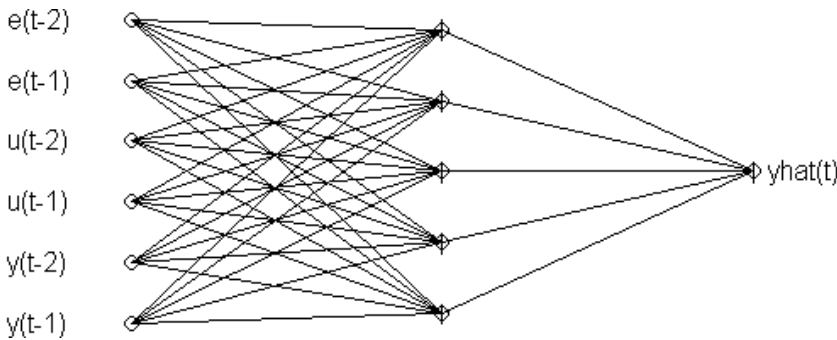
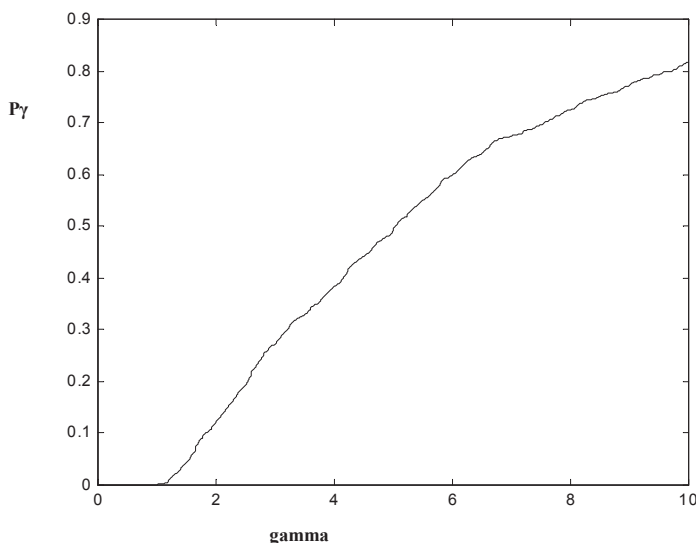


Figure 9: The considered NARMAX neural network



receives as inputs past predicted output and not direct information from the process. Interestingly, this requires the neural model to implicitly learn, during the training phase, the concept of robustness as proven by the $\hat{p}_\gamma = \hat{p}_\gamma(\gamma)$ curve. Conversely, the NARMAX model has a smoother and less complex training phase since it receives fresh information directly from the process (y values) which help the neural model to be stable. As such, the training procedure will not search for weights configuration particularly robust since small deviations, which could make the system unstable, will be directly stabilised by the true information coming from the process.

Figure 10: The \hat{p}_γ function for the NARMAX neural network



CONCLUSION

The main results of the chapter can be summarised as follows. Once given a trained neural network:

- the effects of perturbations affecting the network weights can be evaluated regardless of the topology and structure of the neural network, the strength of the perturbation by considering a probabilistic approach;
- the robustness/sensitivity analysis can be carried out with a Poly-time algorithm by resorting to Randomised Algorithms;
- the analysis is independent from the figure of merit considered to evaluate the loss in performance induced by the perturbations.

REFERENCES

- Alippi, C. (2002). Randomized Algorithms: A system-level, Poly-time analysis of robust computation. *IEEE Transactions on Computers*, 51(7).
- Alippi, C. & Briozzo, L. (1998). Accuracy vs. precision in digital VLSI architectures for signal processing. *IEEE Transactions on Computers*, 47(4).

- Alippi, C., Piuri, V. & Sami, M. (1995). Sensitivity to Errors in Artificial Neural Networks: A Behavioural Approach. *IEEE Transactions on Circuits and Systems: Part 1*, 42(6).
- Bai, E., Tempo, R. & Fu, M. (1997). Worst-case properties of the uniform distribution and randomized algorithms for robustness analysis. *IEEE-American Control Conference*, Albuquerque, NM.
- Bhattacharyya, S.P., Chapellat, H. & Keel, L.H. (1995). *Robust Control: The Parametric Approach*. Englewood Cliffs, NJ: Prentice Hall.
- Calafiore, G., Dabbene, F. & Tempo, R. (1999). Uniform sample generation of vectors in lp balls for probabilistic robustness analysis. In *Recent Advances in Control*, Springer-Verlag.
- Chen, X. & Zhou, K. (1997). On the probabilistic characterisation of model uncertainty and robustness. *Proceedings IEEE-36th Conference on Decision and Control*, San Diego, CA.
- Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Stat.* 23.
- Djavidan, P., Tulleken, H., Voetter, M., Verbruggen, H. & Olsder, G. (1989). Probabilistic Robust Controller Design. *Proceedings IEEE-28th Conference on Decision and Control*, Tampa, FL.
- Dundar, G., Rose, K. (1995). The effects of Quantization on Multilayer Neural Networks, *IEEE Transactions of Neural Networks*. 6(6).
- Hassoun, M.H. (1995). *Fundamentals of Artificial Neural Networks*, The MIT Press.
- Hertz, J., Krogh, A. & Palmer, R.G. (1991). *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Co.
- Holt, J. & Hwang, J. (1993). Finite precision error analysis of neural network hardware implementations. *IEEE Transactions on Computers*. 42(3).
- Ljung, L. (1987). *System Identification, theory for the user*, Prentice-Hall.
- Ljung, L., Sjöberg, J. & Hjalmarsson, H. (1996). On neural networks model structures in system identification. In *Identification, Adaptation, Learning*. NATO ASI series.
- Norgaard, M. (2000). *Neural Network-Based System Identification Toolbox*.
- Piché, S. (1995). The selection of weights accuracies for Madalines. *IEEE Transactions on Neural Networks*. 6(2).
- Stevenson, M., Winter, R. & Widrow, B. (1990). Sensitivity of Feedforward neural networks to weights errors, *IEEE Transactions on Neural Networks*. 1(1).
- Tempo, R. & Dabbene, F. (1999). Probabilistic Robustness Analysis and Design of Uncertain Systems. *Progress in Systems and Control Theory*. 25.

Vidyasagar, M. (1996). *A Theory of Learning and Generalisation with Applications to Neural Networks and Control Systems*. Berlin: Springer-Verlag.

Vidyasagar, M. (1998). Statistical learning Theory and Randomized algorithms for Control. IEEE-Control systems.

Chapter III

Helicopter Motion Control Using a General Regression Neural Network

T.G.B. Amaral

Superior Technical School of Setúbal – IPS, Portugal

M.M. Crisóstomo

University of Coimbra, Portugal

V. Fernão Pires

Superior Technical School of Setúbal – IPS, Portugal

ABSTRACT

This chapter describes the application of a general regression neural network (GRNN) to control the flight of a helicopter. This GRNN is an adaptive network that provides estimates of continuous variables and is a one-pass learning algorithm with a highly parallel structure. Even with sparse data in a multidimensional measurement space, the algorithm provides smooth transitions from one observed value to another. An important reason for using the GRNN as a controller is the fast learning capability and its non-iterative process. The disadvantage of this neural network is the amount of computation required to produce an estimate, which can become large if many training instances are gathered. To overcome this problem, it is described as a clustering algorithm to produce representative exemplars from a group of training instances that are close to one another reducing the computation amount to obtain an estimate. The reduction of training data used by the GRNN can make it possible to separate the obtained representative

exemplars, for example, in two data sets for the coarse and fine control. Experiments are performed to determine the degradation of the performance of the clustering algorithm with less training data. In the control flight system, data training is also reduced to obtain faster controllers, maintaining the desired performance.

INTRODUCTION

The application of a general regression neural network to control a non-linear system such as the flight of a helicopter at or near hover is described. This general regression neural network in an adaptive network that provides estimates of continuous variables and is a one-pass learning algorithm with a highly parallel structure. Even with sparse data in a multidimensional measurement space, the algorithm provides smooth transitions from one observed value to another. The automatic flight control system, through the longitudinal and lateral cyclic, the collective and pedals are used to enable a helicopter to maintain its position fixed in space for a long period of time. In order to reduce the computation amount of the gathered data for training, and to obtain an estimate, a clustering algorithm was implemented. Simulation results are presented and the performance of the controller is analysed.

HELICOPTER MOTION CONTROL

Recently, unmanned helicopters, particularly large-scale ones, have been expected not only for the industrial fields such as agricultural spraying and aerial photography, but also for such fields as observation, rescuing and fire fighting. For monotonous and dangerous tasks, an autonomous flight control of the helicopter is advantageous.

In general, the unmanned helicopter is an example of an intelligent autonomous agent. Autonomous flight control involves some difficulties due to the following:

- it is non-linear;
- flight modes are cross-coupled;
- its dynamics are unstable;
- it is a multivariate (i.e., there are many input-output variables) system;
- it is sensitive to external disturbances and environmental conditions such as wind, temperature, etc;
- it can be used in many different flight modes (e.g., hover or forward flight), each of which requires different control laws;
- it is often used in dangerous environments (e.g., at low altitudes near obstacles).

These characteristics make the conventional control difficult and create a challenge to the design of intelligent control systems.

For example, although helicopters are non-linear systems, NN controllers are capable of controlling them because they are also inherently non-linear. The instabilities that result from time delays between changes in the system input and output can be addressed with the previous learning of the network with a set of data that represents the pilots knowledge to stabilize the helicopter. Linear NN can be implemented to compensate the cross-couplings between control inputs, mainly when the helicopter makes a significant change in its flight.

Therefore, a supervised general regression neural network can be used to control the flight modes of an unmanned helicopter. The regression is the least-mean-squares estimation of the value of a variable based on data samples. The term *general regression* implies that the regression surface is not restricted by being linear. If the values of the variables to be estimated are future values, the general regression network (GRNN) is a predictor. If they are dependent variables related to input variables in a process, system or plant, the GRNN can be used to model the process, system or plant. Once the system is modelled, a control surface can be defined in terms of samples of control variables that, given a state vector of the system, improve the output of the system. If a GRNN is trained using these samples, it can estimate the entire control surface, becoming a controller. A GRNN can be used to map from one set of sample points to another. If the target space has the same dimension as the input space, and if the mapping is one-to-one, an inverse mapping can easily be formed using the same examples. When the variables to be estimated are for intermediate values between given points, then the GRNN can be used as an interpolator.

In all cases, the GRNN instantly adapts to new data points. This could be a particular advantage for training robots to emulate a teacher or for any system whose model changes frequently.

SYSTEM MODELLING

The helicopter control is one of the popular non-linear educational control problems. Due to its highly non-linear dynamics, it gives the possibility to demonstrate basic features and limits of non-linear control concepts. Sugeno (1997, 1998) developed a fuzzy-logic based control system to replace the aircraft's normal set of control inputs. Other researchers, such as Phillips et al. (1994), Wade et al (1994), and Wade and Walker (1994), have developed fuzzy logic flight controls describing systems that include mechanisms for discovering and tuning fuzzy rules in adaptive controllers. (Larkin, 1984) described a model of an autopilot controller based on fuzzy algorithms. An alternative approach to real-time control of an

autonomous flying vehicle based on behavioral, or reactive, approach is proposed by Fagg et al. (1993). A recurrent neural network used to forward modeling of helicopter flight dynamics was described by Walker and Mo (1994). The NN-based controllers can indirectly model human cognitive performance by emulating the biological processes underlying human skill acquisition.

The main difference between NN-based controllers and conventional control systems is that, in the NN case, systems are built from indirectly representations of control knowledge similar to those employed by skilled humans, while in the conventional design case, a deep analytical understanding of the system dynamics is needed. The ability of humans to pilot manned helicopters with only the qualitative knowledge indicate that NN-based controllers with similar capabilities can also be developed.

The helicopter can be modelled as a linear system around trim points, i.e., a flight with no accelerations and no moments. The state space equations are a natural form, which can represent the helicopter motion. The general mathematical model is given by:

$$\dot{x} = Ax + Bu_c$$

$$y = Cx + Du_c$$

where x , u_c and y are the state vector, control vector and output vector, respectively.

The helicopter used to simulate the flight in hover position was a single main rotor helicopter of 15,000 pounds. The control and state vectors are defined as:

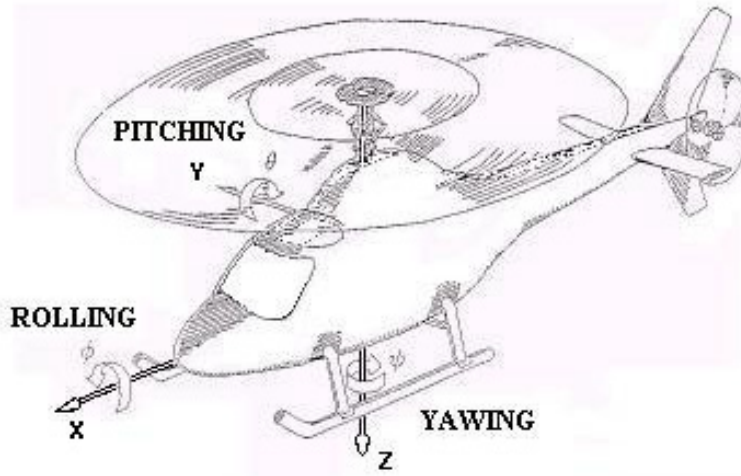
$$u_c^T = [\delta_a \ \delta_b \ \delta_c \ \delta_d] \quad (1)$$

$$x^T = [u \ v \ w \ p \ q \ r \ \phi \ \theta \ \varphi \ x \ y \ z] \quad (2)$$

where

δ_a is the collective control [*inches*];
 δ_b and δ_c are the longitudinal and lateral cyclic controls, respectively [*inches*];

Figure 1: Helicopter coordinates



δ_d is the pedal control [inches];
 u, v and w are the perturbation linear velocities [ft/sec];
 p, q and r are the perturbation angular velocities [rad/sec];
 ϕ, θ and φ are the perturbation euler angles for roll, pitch and yaw [rad];
 x, y and z are the perturbation linear displacements over the ground [ft].

Figure 1 shows the coordinate system to describe the motion of the helicopter. The origin of the helicopter axes is placed on the center of gravity.

The thrust of the main rotor, thus mainly the vertical acceleration, is controlled by the collective control (δ_a). The pitching moment, that is, nose pointing up or down, is controlled by the longitudinal cyclic control (δ_b). The rolling moment, that is, right wing tip down, left wing tip up, and vice versa, is controlled by the lateral cyclic control (δ_c). The yawing moment, that is, nose left and right, is controlled by the pedal control (δ_d).

The corresponding differential equations that represent the behavior of the helicopter in hover position are:

$$\begin{aligned}
 \frac{du}{dt} = & -0.069u - 0.032v + 116.8p + 1168.5q - 6.15r - 32.19\theta + 0.118\delta_a \\
 & - 2.79\delta_b - 0.25\delta_c + 0.0043\delta_d
 \end{aligned}$$

$$\begin{aligned}\frac{dv}{dt} = & 0.017u - 0.085v - 0.0021w - 430.5p + 381.3q + 30.75r + 32.14\phi \\ & + 0.023\theta - 0.14\delta_a - \delta_b + 0.665\delta_c - 1.39\delta_d\end{aligned}$$

$$\begin{aligned}\frac{dw}{dt} = & -0.0021v - 0.257w + 7.99p + 46.74q + 135.3r + 1.85\phi - 0.404\theta \\ & - 9.23\delta_a - 0.107\delta_b - 0.01\delta_c\end{aligned}$$

$$\begin{aligned}\frac{dp}{dt} = & 0.45u - 0.687v - 0.0021w - 6027.2p + 5043.16q + 664.2r - 1.82\delta_a \\ & - 13.7\delta_b + 8.58\delta_c - 5.15\delta_d\end{aligned}$$

$$\begin{aligned}\frac{dq}{dt} = & 0.665u + 0.429v - 0.043w - 1537.5p - 15744.5q - 12.3r - 0.966\delta_a \\ & + 37.13\delta_b + 3.43\delta_c + 0.75\delta_d\end{aligned}$$

$$\begin{aligned}\frac{dr}{dt} = & -0.0214u + 0.515v + 0.0064w - 369.0p - 44.28q - 1266.9r + 25.97\delta_a \\ & - 0.15\delta_b + 0.075\delta_c + 40.78\delta_d\end{aligned}$$

$$\frac{d\phi}{dt} = p$$

$$\frac{d\theta}{dt} = q$$

$$\frac{d\varphi}{dt} = r$$

$$\frac{dx}{dt} = u$$

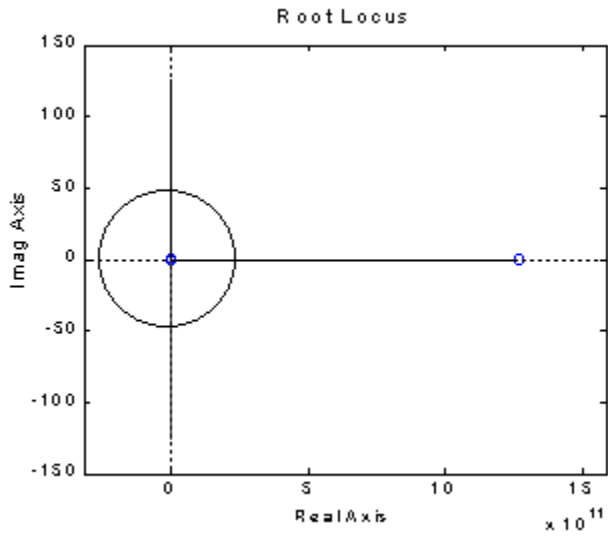
$$\frac{dy}{dt} = v$$

$$\frac{dz}{dt} = w$$

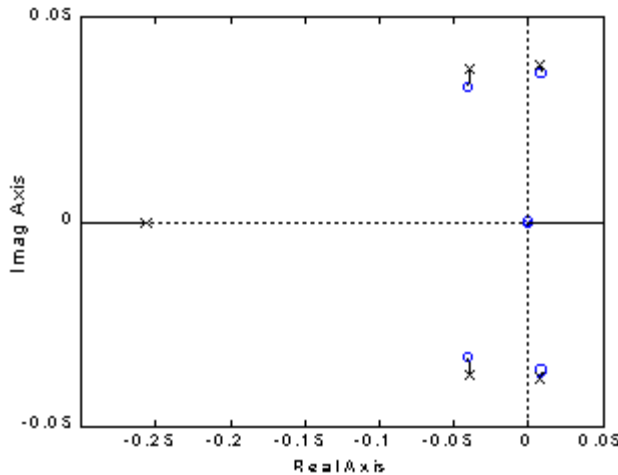
Since each motion is not independent of δ_a , δ_b , δ_c and δ_d , there exists a cross-coupling.

Figure 2 shows the root locus for the model described above. Figure 2(a) shows the root locus, considering the collective control as the input and the vertical displacement as the output. In Figure 2(c), the longitudinal cyclic and the forward displacement are the input and the output, respectively. Figure 2(e) shows the root locus considering the lateral cyclic as the input and the lateral displacement as the output. Figures 2(b), (d) and (f) show the zoom of the region near the imaginary axis as well as the roots that dominate the transient response. In general, the contribution

Figure 2: Root locus of the helicopter model

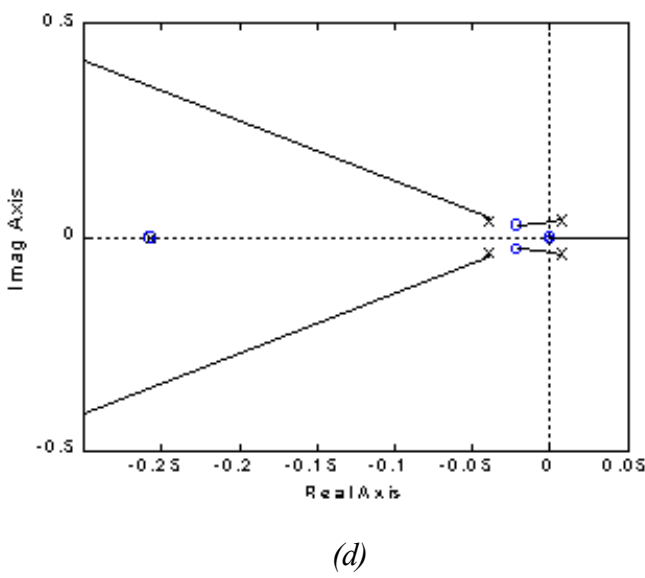
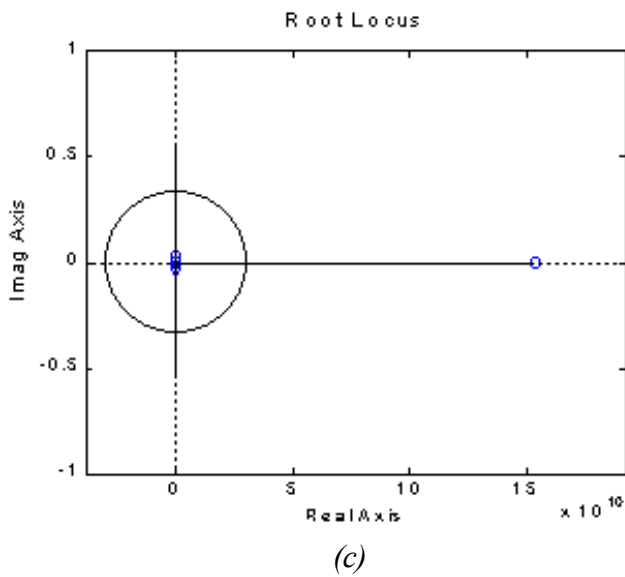


(a)



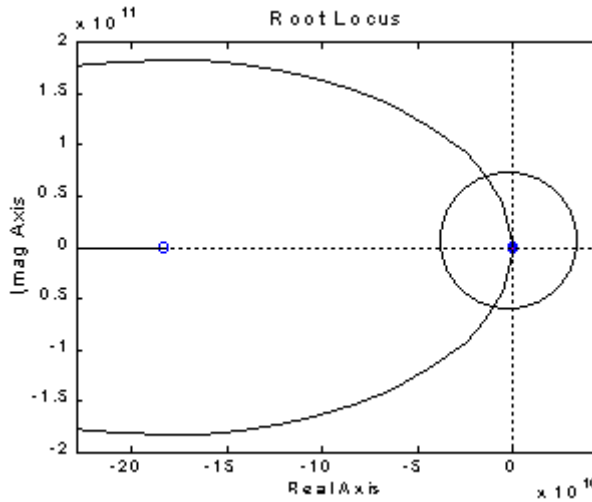
(b)

Figure 2: Root locus of the helicopter model (continued)

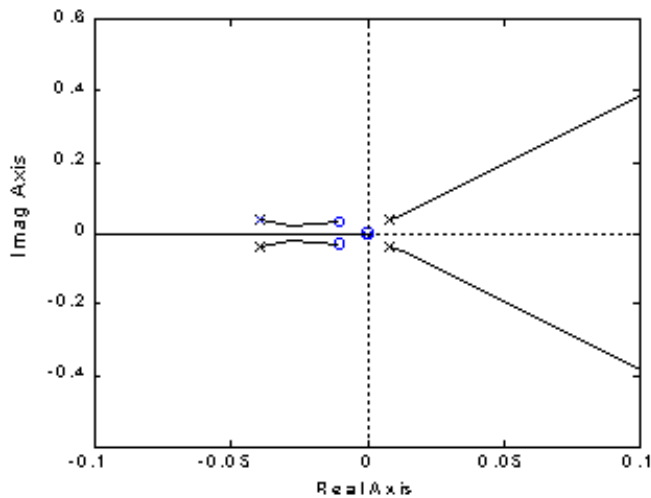


in the time response of roots that lie relatively far to the left in the s-plane will be small. These three Figures clearly show that some of the eigenvalues corresponding to the helicopter model are in the right side of the s-plane, with positive real-part values, making the system unstable.

Figure 2: Root locus of the helicopter model (continued)



(e)



(f)

GENERAL REGRESSION NEURAL NETWORK

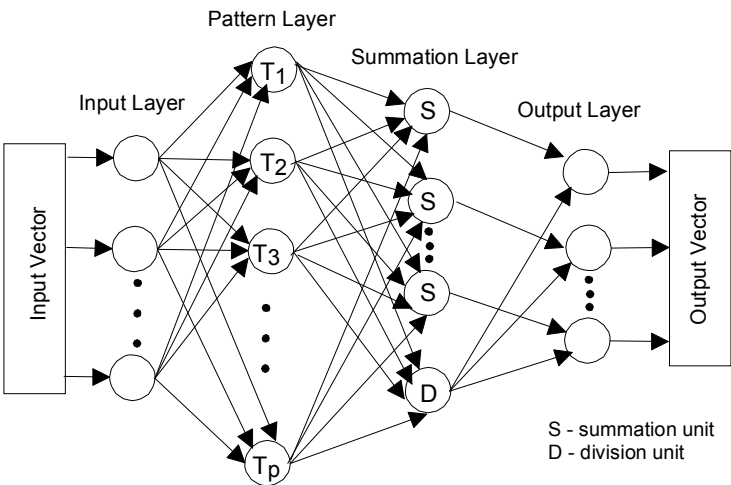
The generalized regression neural networks are memory-based feed-forward networks originally developed in the statistics literature by Nadaraya (1964) and known as Nadaraya-Watson kernel regression. Then the GRNN was ‘re-discovered’ by Specht (1991) and Chen, C. (1996), with the desired capability of

learning in a one-shot manner from incoming training data being independent upon time-consuming iterative algorithms. This quick learning ability allows the GRNN to adapt to changing system parameters more rapidly than other methods such as genetic algorithms, back-propagation or reinforcement learning. This is achieved by the estimation of continuous variables with a single training instance and refining this estimation in a non-iterative manner since the training data is added to the network. Therefore, this neural network can be used as an intelligent controller for the autonomous helicopter.

GRNN Architecture

The GRNN is a special extension of the radial basis function network. This neural network is based on nonlinear regression theory consisting of four layers: the input layer, the pattern layer, the summation layer and the output layer (see Figure 3). It can approximate any arbitrary mapping between the input and output vectors. While the neurons in the first three layers are fully connected, each output neuron is connected only to some processing units in the summation layer. The summation layer has two different types of processing units: the summation units and the division unit. The number of summation units in the summation layer is always the same as the number of GRNN output units. The division unit only sums the weighted activations of the pattern units without using any activation function. Each of the output units is connected only to its corresponding summation unit and to the division unit. There are no weights in these connections. The function of the output units consists of a simple division of the signal coming from the summation unit by the signal coming from the division unit.

Figure 3: Topology of the generalized regression neural network



Consider X and Y independent and dependent variables respectively. The regression of Y on X is the computation of the most probable value of Y for each value of X based on a finite number of possibly noisy measurements of X and the associated values of Y . The variables X and Y can be vectors. In parametric regression, some functional form with unknown parameters, a_p , is assumed and the values of the parameters are chosen to make the best fit to the observed data. For example, in linear regression, the output Y is assumed to be a linear function of the input X , and the unknown parameters a_i are linear coefficients. In nonparametric regression, no assumption about the statistical structure of incoming training data is made.

The equation form used for the GRNN is presented in (3) and (4) (Specht, 1991). The resulting regression, which involves summations over the observations, is directly applicable to problems involving numerical data.

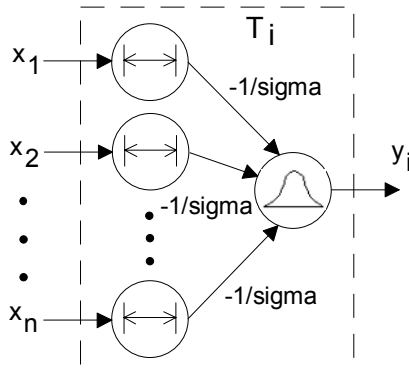
$$D_i^2 = (X - X_i)^T (X - X_i) \quad (3)$$

$$\hat{Y}(X) = \frac{\sum_{i=1}^n Y_i \exp\left(\frac{-D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^n \exp\left(\frac{-D_i^2}{2\sigma^2}\right)} \quad (4)$$

X_i and Y_i are earlier samples that compose the training data set and n represents the number of training data.

The estimated output $\hat{Y}(X)$ is a weighted average of all the observed values Y_i , where each observed value is weighted exponentially according to the Euclidean distance between X and X_i . The smoothing parameter σ (or bandwidth) controls how tightly the estimate is made to fit the data. Figure 4 shows the shapes of the i^{th} pattern unit node in the pattern layer T_i of the GRNN for three different values of the smoothing parameter. The output value in each pattern unit T_i is given by the following expression:

$$T_i = \exp\left(-\|X - X_i\|^2 / 2\sigma^2\right) \quad (5)$$

Figure 4: Structure of the i^{th} pattern unit

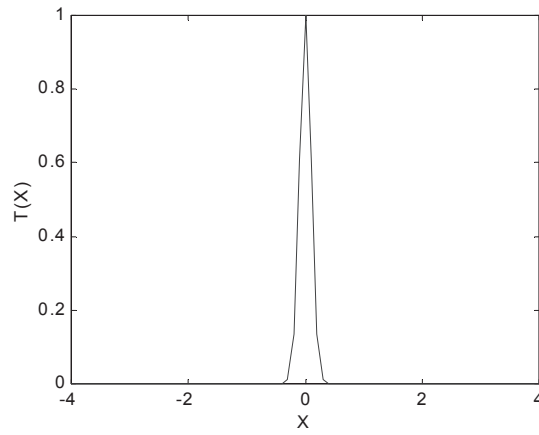
where $T_i : \mathcal{R}^n \rightarrow \mathcal{R}$ and $i = 1, \dots, p$ represents all the pattern units in the pattern layer.

When the smoothing parameter σ is made large, the estimate is forced to be smooth and in the limit becomes a multivariate Gaussian with covariance $\sigma^2 I$. On the other hand, a smaller σ allows the estimate to more closely fit the data and assume non-Gaussian shapes (see Figure 5(a)). In this case, the disadvantage happens when the wild points could have a great effect on the estimate. As σ becomes large, $\hat{Y}(X)$ assumes the value of the sample mean of the observed Y_i (see Figure 5(c)). When σ goes to 0, $\hat{Y}(X)$ becomes the value of the Y_i associated with the data closest to X . For intermediate values of σ , all values of Y_i are used, but those corresponding to observed values closer to X are given heavier weight (see Figure 5(b)).

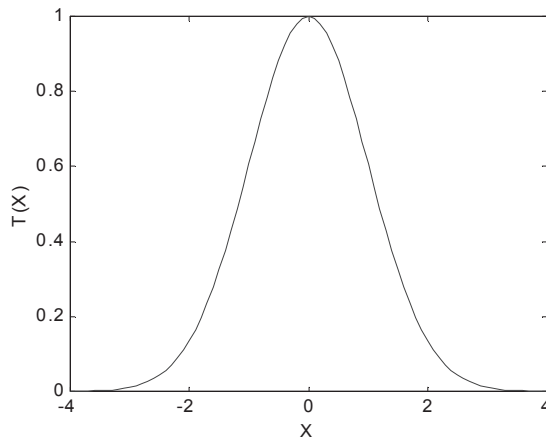
If we have much confidence in the input data, a small σ can be used to give greater weight to individual measurements. However, if there is uncertainty in the input data due to noise, a large σ must be used to reduce the effect of spurious measurements on an estimate. The optimisation of the smoothing parameter is critical to the performance of the GRNN. Usually this parameter is chosen by the cross-validation procedure or by esoteric methods that are not well known in the neural net literature.

In the next subsections it will be shown the application of the GRNN to model a piecewise linear function, and to control an unmanned helicopter. It is also discussed the clustering algorithm to obtain the representative samples between the training data (Lefteri, 1997).

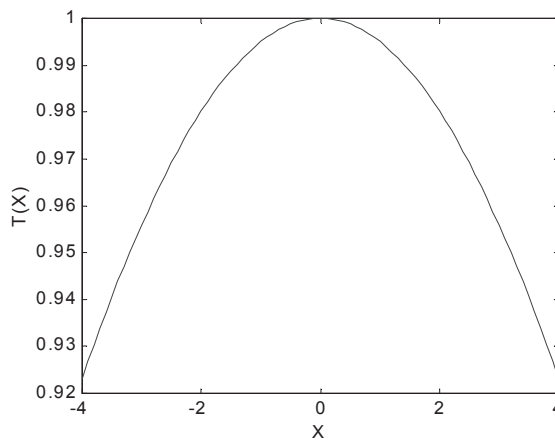
Figure 5: Possible shapes for different smoothing parameter values



(a) $\sigma = 0.1$



(b) $\sigma = 1$



(c) $\sigma = 10$

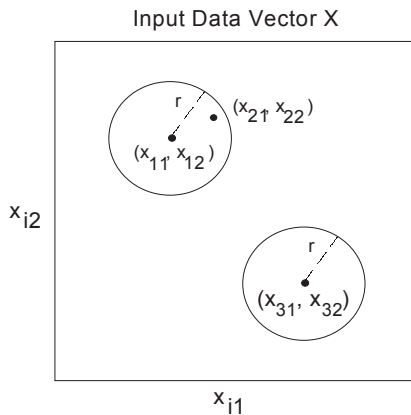
Clustering and Adaptation to Nonstationary Statistics

For some problems, the number of sample points (X, Y) may be not sufficient since it is desired to use all the data obtainable directly in the estimator (4). In other problems, the number of sample points can be sufficiently large, becoming no longer practical to assign a separate node to each sample. There exist various clustering techniques that can be used to group samples. Therefore the group can be represented by only one node (Moody & Darken, 1989), (Burrascano, 1991), and (Tou & Gonzalez, 1974). A sample point belongs to a group if the distance between this sample point and the cluster center is less than a specific value. This value, which can be considered the radius of influence r of the cluster, must need to be specified before the training starts.

In the developed clustering algorithm, representative samples are produced from a group of training instances that are close to one another. The training is completed after presenting to the GRNN input layer, only once, each input-output vector pair from the training set. The Euclidean distance to obtain the representative samples and to reject all the other data points was used in the algorithm.

Suppose that we have a set of n data samples $\{(X_i, Y_i) \in (X, Y); i = 1, \dots, n\}$ where X and Y represent the input and output data sets, respectively. X_i and Y_i are a 2D vector (x_{i1}, x_{i2}) and a single value, respectively. Initially, the first sample point (X_1, Y_1) in the training set becomes the center of the cluster of the first pattern unit at X . The next sample point is then compared with this center of the first pattern unit, and it is assigned to the same cluster (pattern unit) if its distance from this center is less than the prespecified radius of influence. Then, equation (7) should be updated for this cluster. Otherwise, if the distance $|X - X_i|$ is higher than r , then the sample becomes the center of the cluster of the next pattern unit. Figure 6 shows how the clustering algorithm works considering three points (x_{11}, x_{12}) , (x_{21}, x_{22}) and (x_{31}, x_{32}) .

Figure 6: Cluster generation



x_{32}). The first two points belong to the same cluster because the distance between them is less than r . The third point is the center of the new cluster since the distance is higher than r .

In the same manner, all the other sample points are compared one-by-one with all pattern units already set, and the whole pattern layer is thus gradually built. During this training, the determined values of individual elements of the center clusters are directly assigned to the weights in connections between the input units and the corresponding pattern units.

After the determination of the cluster centers, the equation (4) can then be rewritten as (Specht, 1991):

$$\hat{Y}(X) = \frac{\sum_{i=1}^p A_i \exp\left(\frac{-D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^p B_i \exp\left(\frac{-D_i^2}{2\sigma^2}\right)} \quad (6)$$

where

$$\begin{cases} A_i(k) = A_i(k-1) + Y_j \\ B_i(k) = B_i(k-1) + 1 \end{cases} \quad (7)$$

The value $p < n$ represents the number of clusters. $A_i(k)$ and $B_i(k)$ are the coefficients for the cluster i after k samples. $A_i(k)$ is the sum of the Y values and $B_i(k)$ is the number of samples assigned to cluster i . The $A_i(k)$ and $B_i(k)$ coefficients are completely determined in one iteration for each data sample.

Reducing the Number of Clusters in Dynamic Systems

If the network is used to model a system with changing characteristics, it is necessary to eliminate the clusters that were not updated during a period of time. The justification for this is that in the dynamic systems appears new cluster centers that represent the new behavior of the model. Then, the number of clusters will increase and also the computation time to produce an output. Since the A and B coefficients can be determined by using the recursive equations (7), it is introduced a forgetting function allowing to reduce the number of clusters as shown in the following expressions,

$$\begin{cases} A_i(k) = A_i(k-1)\exp\left(-\frac{t_i}{\tau}\right) + \left(1 - \exp\left(-\frac{t_i}{\tau}\right)\right)Y_j \\ B_i(k) = B_i(k-1)\exp\left(-\frac{t_i}{\tau}\right) + \left(1 - \exp\left(-\frac{t_i}{\tau}\right)\right) \end{cases} \quad (8)$$

and

$$\begin{cases} A_i(k) = A_i(k-1)\exp\left(-\frac{t_i}{\tau}\right) \\ B_i(k) = B_i(k-1)\exp\left(-\frac{t_i}{\tau}\right) \end{cases} \quad (9)$$

Equation (8) is the update expression when a new sample is assigned to the cluster i . Equation (9) is applied to all other clusters. The parameters t and τ are the time passed after the last update of the cluster i and a constant that determines when the cluster disappears after the last update, respectively. Figure 7 shows the exponential decay and increase functions represented by solid and dashed lines respectively. The exponential decay function will attenuate all the coefficients A and B of the clusters. The increasing exponential function allows the new sample data to have an influence in the local area around its assigned cluster center.

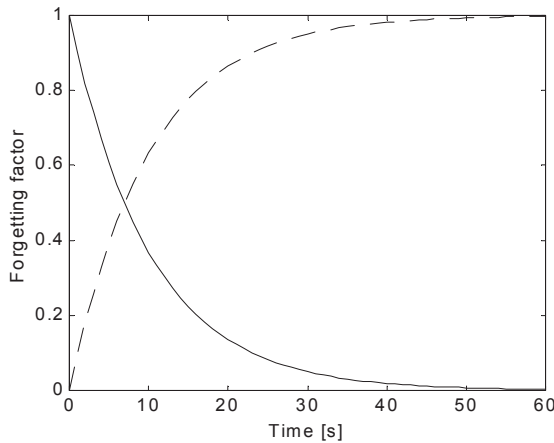
When the coefficient B is zero then the corresponding cluster would be eliminated. For example, considering Figure 7, if the cluster i is not updated during 60 seconds then the cluster i (and its associated A_i and B_i coefficients) will be eliminated.

Comparison with other Non-Linear Regression Techniques

The advantages of GRNN relative to other non-linear regression techniques are:

1. The network learns in one pass through the data and can generalize from samples as soon as they are stored.
2. With the increasing number of observed samples, the estimates converge to the conditional mean regression surfaces. However, using a few number of samples, it forms very reasonable regression surfaces.

Figure 7: Exponential decay function



3. The estimate is limited within a range defined by the minimum and maximum of the observations.
4. The estimate cannot converge to poor solutions corresponding to local minima of the error criterion.
5. A software simulation is easy to develop and to use.
6. The network can provide a mapping from one set of sample points to another. If the mapping is one-to-one, an inverse mapping can easily be generated from the same sample points.
7. The clustering version of GRNN, equation (6), limits the numbers of nodes. Optionally it can provide a mechanism for forgetting old data.

The main disadvantage of GRNN is the amount of computation required to produce an estimate, since it can become large if many training instances are gathered. To overcome this problem it was implemented a clustering algorithm. This additional processing stage brings the questions regarding when to do the initial clustering and if the re-clustering should be done after additional training data is gathered.

GRNN-Based Model

The described GRNN type has many potential uses as models and inverse models (see Figure 8). A simple problem with one independent variable is used as an example to show how the regression technique is applied to modelling a system. Suppose that we have a piecewise linear function and training instances taken from this function (see Figure 9) (Montgomery, 1999). The samples $X_i = [-4, -3, -2, -1, 0, 1, 2, 3, 4]$ and $Y_i = [-1, -1, -1, -1, -1, 0, 1, 1, 1]$ are represented by circles.

Since GRNN always estimates using a weighted average of the given samples, the estimate is always within the observed range of the dependent variable. In the input range, the estimator takes on a set of curves that depend on σ , each of which is a reasonable approximation to the piecewise linear function (see Figure 9). Figure 10 shows the GRNN estimates of this function for different sigma values. For $\sigma = 0.5$ the curve is the best approximation. A small sigma allows the estimate to more closely fit the training data, while a large sigma produces a smoother estimate. It is possible to over fit the data with very small values of σ .

Besides the advantages of the GRNN when compared with other non-linear regression techniques, there exists another four benefits from the use of the GRNN. First, it has a non-iterative, fast-learning capability. Second, the smoothing parameter, σ , can be made large to smooth out noisy data or made small to allow the estimated regression surface to be as non-linear as required to more closely approximate the actual observed training data. Third, it is not necessary to specify the form of a regression equation. Finally, the addition of new samples to the training data set does not require re-calibrating the model. The disadvantage of the network is the

Figure 8: Modelling the system using GRNN

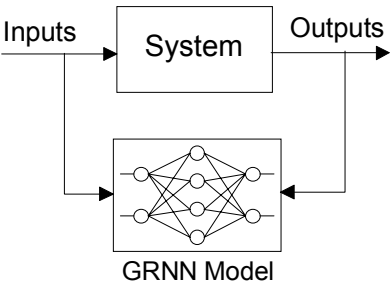


Figure 9: Linear function with training set

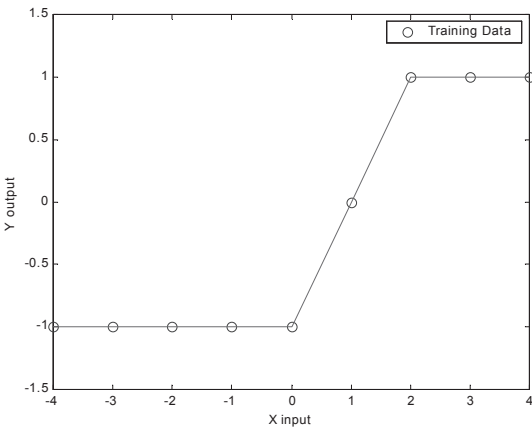
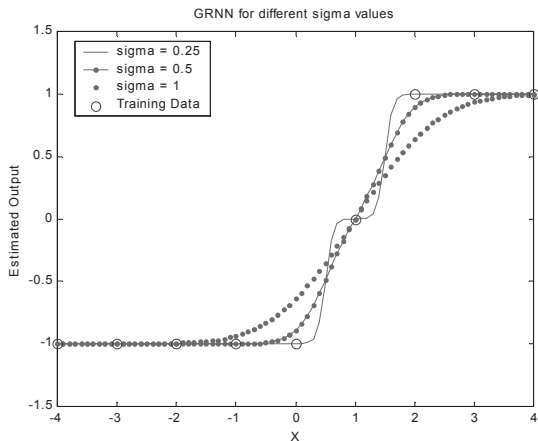


Figure 10: Example of GRNN application



difficulty to analyze and to provide a clear understanding of how its behavior is related to its contents. The inexistence of an intuitive method for selecting the optimal smoothing parameter is also a difficult task to solve.

GRNN-Based Controller

The non-linear control helicopter and robotic systems are particularly good application areas that can be used to demonstrate the potential speed of neural networks implemented in parallel hardware and the adaptability of instant learning. First, the GRNN learns the relation between the state vector of the system and the control variables. After the GRNN-based model is trained, it can be used to determine control inputs. One way in which the neural network could be used to control a system is shown in Figure 11.

The GRNN is not trained in the traditional neural network sense where weights are adjusted iteratively. Rather, all training data is stored in memory (thus the term memory-based network) and only when the output is necessary for a different input

Figure 11: A GRNN controller

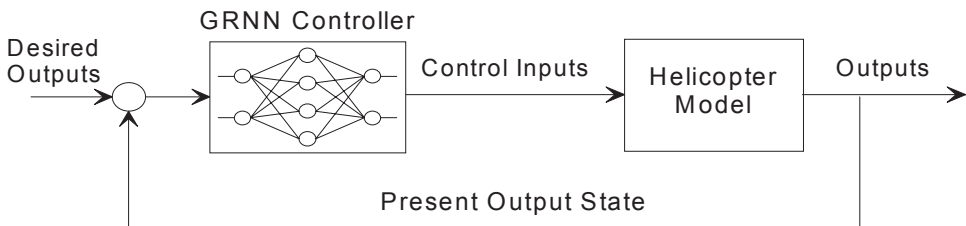


Table 1: Vector prototype for each control input

Control Inputs	State Variables
Collective	$\{\delta_a, z, w\}$
Longitudinal cyclic	$\{\delta_b, x, u, \theta, q\}$
Lateral cyclic	$\{\delta_c, y, v, \phi, p\}$
Pedals	$\{\delta_d, \varphi, r\}$

a new computation is performed. In controlling the helicopter each data training is a vector with the input variables and the corresponding output for each controller. Only the samples that represent the cluster centers are used to populate the network. The reduction of training data used by the GRNN is an important problem because we can obtain a faster controller maintaining a good performance. It is also possible to separate the obtained representative clusters' center, for example, in two data sets for the coarse and fine control.

To control the helicopter flight mode in hover position, four data sets corresponding to each input control were used. In each data set exists a set of vectors that correspond to the representative clusters obtained after the clustering algorithm is applied. The vector structure in each data set is given in Table 1.

SIMULATED RESULTS

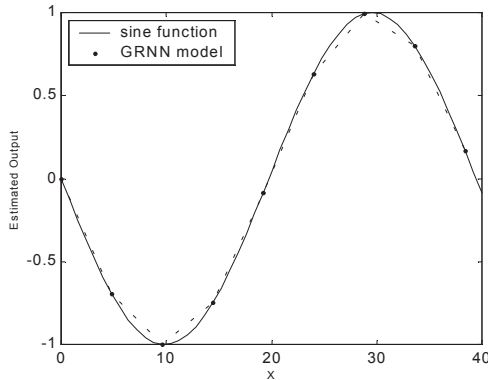
An experiment was performed to determine the extent to which performance of the clustering algorithm degrades with less training data. Figure 12 shows the output of the sine function and the model when the identification procedure was carried out for only nine patterns instead of the fifty-five used to represent the sine function.

Figures 13 and 14 show the open loop responses of the helicopter displacements and euler angles corresponding to impulse control inputs in the longitudinal and lateral cyclic. First it was applied an impulse in the longitudinal control and then in the lateral control input. The initial conditions for the helicopter model are as follows:

$$u_0 = 0, v_0 = 0, w_0 = 0, \phi_0 = -0.0576, \theta_0 = 0.0126, \varphi_0 = 0.$$

The initial conditions corresponding to the derivatives of the state variables described above are zero.

Figure 12: Output of sine function (solid line) and of GRNN model (dotted line) after training with only nine patterns



Figures 15 to 23 shows the system response using the GRNN controller using the data set for each controller. Each data set contains the representative clusters obtained after the clustering process. Figures 15 to 17 show the displacement of the helicopter in the longitudinal, lateral and vertical axis, respectively. These three Figures show that the higher displacement changes occur in the forward and lateral axis rather than in the vertical axis. This happens because the impulse control inputs were applied to the longitudinal and lateral cyclic which are the commands to control the forward and lateral displacements of the helicopter.

Figure 18 shows the trajectory of the helicopter in the 2D plan. The arrows indicate the direction of the helicopter displacement after applied impulses in the control inputs. After approximately five minutes the helicopter is stabilized in the initial position (i.e. $x = y = z = 0$).

Figure 13: Simulated results of (x, y, z) to an impulse control input, in the longitudinal cyclic

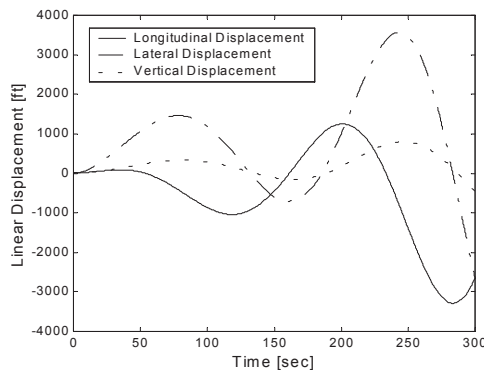


Figure 14: Simulated results of (ϕ, θ, φ) to an impulse control input, in the longitudinal cyclic

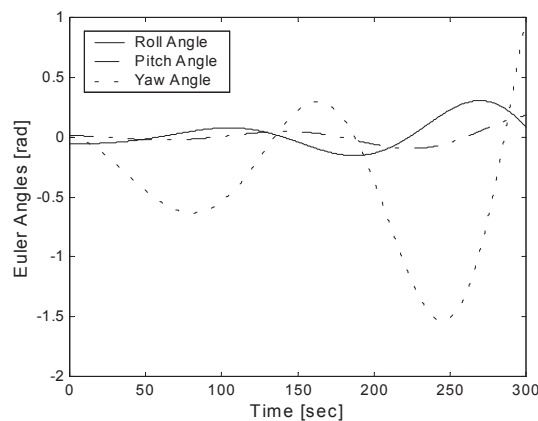


Figure 19 shows the roll, pitch and yaw angles. Even when the initial conditions of the roll and pitch angles are different from zero, these angles stabilize, permitting the control of the helicopter.

Figures 20 to 23 show the control inputs applied to the helicopter. The control inputs were limited to $\pm 5V$ for simulate practical limitations of the actuators. Since the higher perturbation occurs in the longitudinal and lateral displacements than it was the longitudinal and lateral cyclic control inputs the actuators with higher performance.

Figure 15: Simulated results of x for an impulse control input, in the longitudinal and lateral cyclic

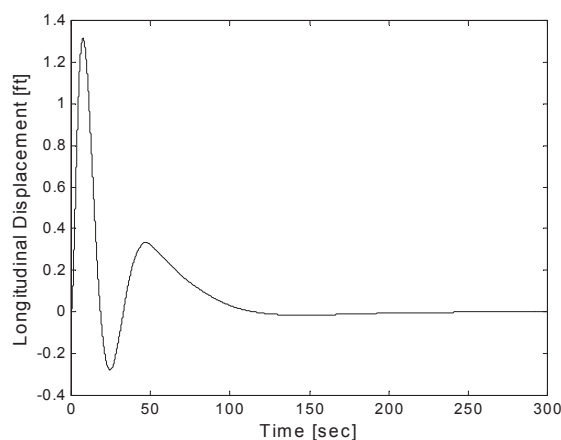


Figure 16: Simulated results of y for impulse control inputs, in the longitudinal and lateral cyclic

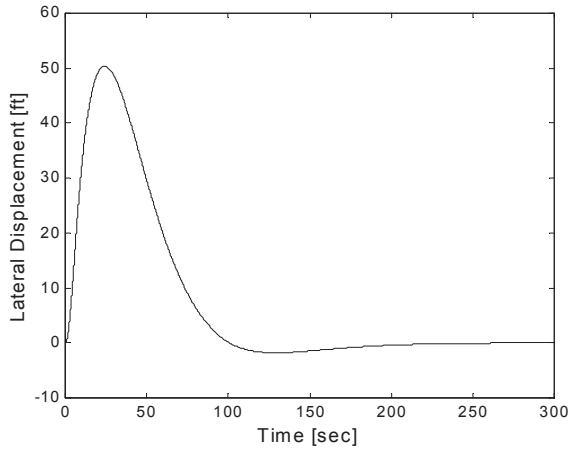
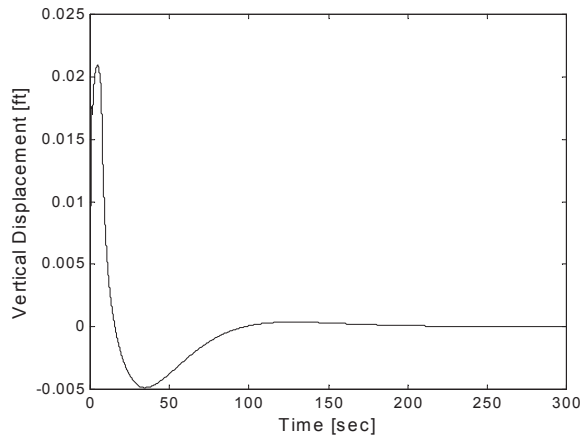


Figure 17: Simulated results of z for impulse control inputs, in the longitudinal and lateral cyclic



Since for each flight mode it can be used one distinct GRNN controller, then it is not necessary to reduce the number of clusters. Each controller has a cluster set that represents the dynamic behavior of the helicopter for the specific flight mode.

CONCLUSIONS

To control the displacement of a single main rotor helicopter of a 15,000-pound using the longitudinal, lateral and collective control inputs, three GRNN

Figure 18: Trajectory of the helicopter in the (x, y) plan

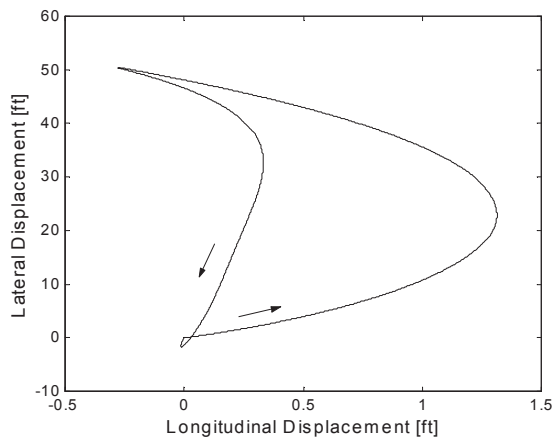
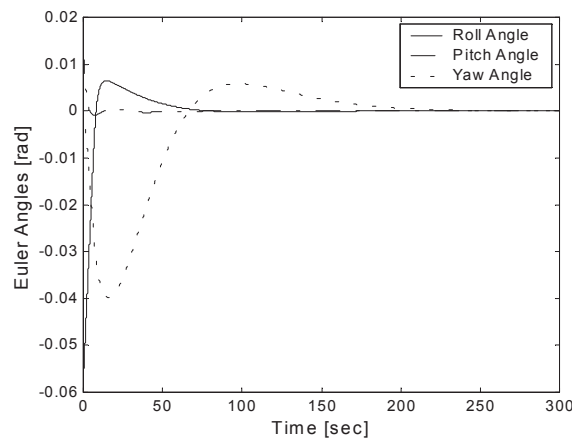


Figure 19: Simulated results of (ϕ, θ, φ) to an impulse control input, in the longitudinal and lateral cyclic



controllers have been used. The direction of the helicopter nose was controlled by the pedals control input using another GRNN controller. With these controllers it was possible to enable the helicopter to maintain its stationary position for a long period of time. The advantage of the GRNN controller is the fast-learning capability and the non iterative process. For many gathered training instances, the computation amount became large. Therefore, to overcome this problem a clustering algorithm was implemented.

Figure 20: Simulated result of the collective control input

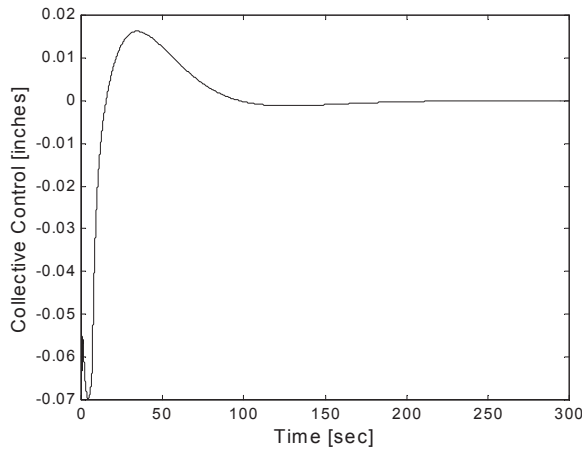
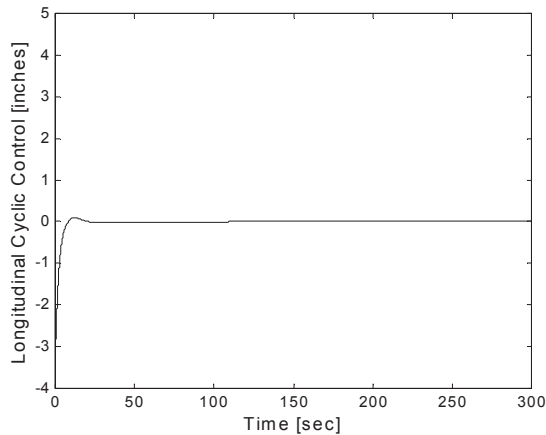


Figure 21: Simulated result of the longitudinal cyclic control



FUTURE DIRECTIONS

A fuzzy algorithm can also control the helicopter. Fuzzy rule base systems are linguistic in nature and can be inspected by a human expert. However, GRNN and fuzzy algorithms could be used together. Fuzzy logic gives a common framework for combining numerical training data and expert linguistic knowledge along with the compact transparency and computational efficiency of rule bases, while the GRNN

Figure 22: Simulated result of the lateral cyclic control

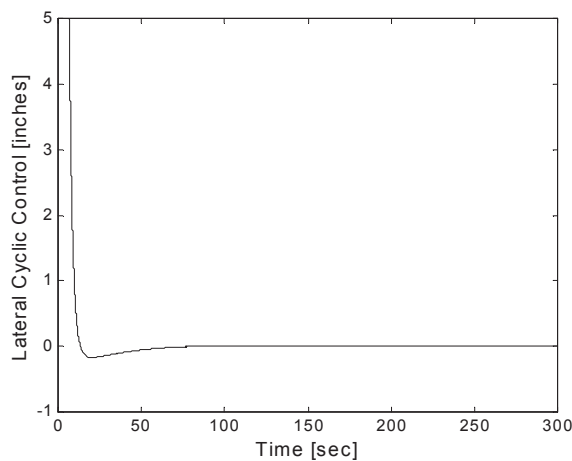
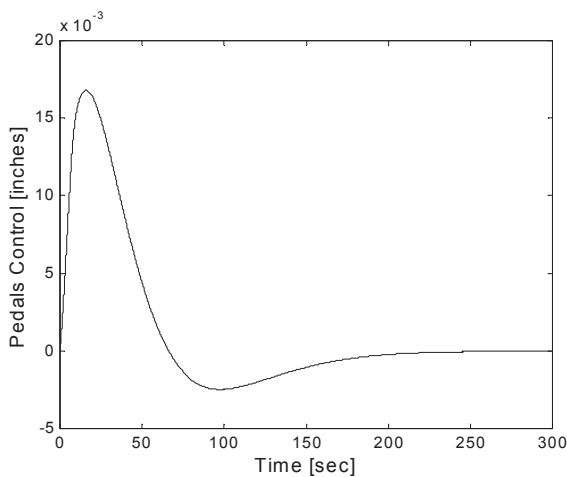


Figure 23: Simulated result of the pedals control



gives the approach rapid adaptive capability. For this reason, one of the future research directions may be the hybrid fuzzy logic/GRNN approach.

ACKNOWLEDGMENTS

The authors would like to thank Professor Mark Dreier from Bell Helicopter Textron, USA. He graciously sent to us his Matlab/Simulink system containing a helicopter model describing the dynamic behaviour of the helicopter in hover flight mode.

REFERENCES

- Burrascano, P. (1991, July). Learning vector quantization for the probabilistic neural network, *IEEE Trans. Neural Network*, (2), 458-461.
- Chen, C. H. (1996). *Fuzzy Logic and Neural Network Handbook*. McGraw-Hill.
- Fagg, A., Lewis, M., Montgomery, J. & Bekey, G. (1993). The USC autonomous flying vehicle: An experiment in real-time behavior-based control. *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, July.
- Larkin, L. (1984). A fuzzy logic controller for aircraft flight control. *Proceedings 23rd Conference on Decision and Control*. Las Vegas, NV. December.
- Montgomery, J. F. (1999). Learning Helicopter Control through Teaching by Showing. PhD dissertation. University of Southern California, Los Angeles, CA.
- Moody, J., & Darken, C. (1989). Fast learning in networks of locally-tuned processing units, *Neural Computation*, (1). 281-294.
- Nadaraya, E. A. (1964), On estimating regression, *Theory Probab. Application* 10, 186-190.
- Phillips, C., Karr, C. & Walker, G. (1994). A genetic algorithm for discovering fuzzy logic rules. *Proceedings International Fuzzy Systems and Intelligent Controls Conference*, March.
- Specht, D.F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6), November, 568-576.
- Sugeno, M. (1997). *The Industrial Electronics Handbook*. CRC Press, 1127-1138.
- Sugeno, M. (1998). Recent advances in fuzzy control: Stability issues and application to an unmanned helicopter. *World Automation Congress*, Alaska, May.
- Tou, J. T. & Gonzalez, R. C. (1974). *Pattern Recognition Principles*. Reading, MA: Addison-Wesley.
- Tsoukalas, L.H. & Uhrig, R.E. (1997). *Fuzzy and Neural Approaches in Engineering*. A volume in the Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control. Simon Haykin, Series Editor.
- Wade, R. & Walker, G. (1994). Fuzzy logic adaptive controller-helicopter (FLAC-H): A multi-platform, rotary-winged aerial robotic control system. *19th Army Science Conference*. Orlando, FL, June.
- Wade, R., Walker, G. & Phillips, C. (1994). Combining genetic algorithms and

aircraft simulations to tune fuzzy rules in a helicopter control system. *Advances in Modelling and Simulation Conference*, Huntsville, AL, April.

Walker, G. & Mo, S. (1994). Forward modelling of helicopter flight dynamics using recurrent neural networks, *19th Army Science Conference*, Orlando, FL, June.

Chapter IV

A Biologically Inspired Neural Network Approach to Real-Time Map Building and Path Planning

Simon X. Yang
University of Guelph, Canada

ABSTRACT

A novel biologically inspired neural network approach is proposed for real-time simultaneous map building and path planning with limited sensor information in a non-stationary environment. The dynamics of each neuron is characterized by a shunting equation with both excitatory and inhibitory connections. There are only local connections in the proposed neural network. The map of the environment is built during the real-time robot navigation with its sensor information that is limited to a short range. The real-time robot path is generated through the dynamic activity landscape of the neural network. The effectiveness and the efficiency are demonstrated by simulation studies.

INTRODUCTION

Real-time path planning with collision free in a non-stationary environment is a very important issue in robotics. There are a lot of studies on the path planning for robots using various approaches. Most of the previous models use global methods to search the possible paths in the workspace (e.g., Lozano-Perez, 1983; Zelinsky,

1994; Al-Sultan & Aliyu, 1996; Li & Bui, 1998). Ong and Gilbert (1998) proposed a new searching-based model for path planning with penetration growth distance, which searches over collision paths instead of the free workspace. Most searching-based models can deal with static environment only and are computationally complicated when the environment is complex. Some of the early models deal with static environment only, and may suffer from undesired local minima (e.g., Ilari & Torras, 1990; Zelinsky, 1994; Glasius et al., 1994). Some previous robot motion planning models require the prior information of the non-stationary environment, including the varying target and obstacles. For example, Chang and Song (1997) proposed a virtual force guidance model for dynamic motion planning of a mobile robot in a predictable environment, where an artificial neural network is used to predict the future environment through a relative-error-back-propagation learning.

Several neural network models were proposed to generate real-time trajectory through learning (e.g., Li & Ogmen, 1994; Beom & Cho, 1995; Glasius et al., 1994; 1995; Zalama, Gaudiano & Lopez Coronado, 1995; Chang & Song, 1997; Gaudiano et al., 1996; Yang, 1999; Yang & Meng, 2000a, 2000b, 2001). The learning based approaches suffer from extra computational cost because of the learning procedures. In addition, the planned robot motion using learning based approaches is not optimal, especially during the initial learning phase of the neural network. For example, Zalama et al. (1995) proposed a neural network model for the navigation of a mobile robot, which can generate dynamical trajectory with obstacle avoidance through unsupervised learning.

Glasius et al. (1995) proposed a neural network model for real-time trajectory formation with collision free in a non-stationary environment. However, this model suffers from slow dynamics and cannot perform properly in a fast changing environment. Inspired by Hodgkin and Huxley's (1952) membrane equation and the later developed Grossberg's (1988) shunting model, Yang and Meng (2000a) proposed a neural network approach to dynamical trajectory generation with collision free in an arbitrarily changing environment. These models are capable of planning a real-time optimal path in non-stationary situations without any learning process. But the planned paths in Glasius et al. (1995) and Yang and Meng (2000a) do not take into account the clearance from obstacles, which is demanded in many situations. By introducing inhibitory lateral connections in the neural network, Yang and Meng (2000b) proposed a new model for path planning with safety consideration, which is capable of generating a "comfortable" path for a mobile robot, without suffering either the "too close" (narrow safety margin) or the "too far" (waste) problems. However, the models in Ilari and Torras (1990), Zelinsky (1994), Zalama et al. (1995), Glasius et al. (1995) and Yang and Meng (2000a, 2000b) assume that the workspace is known, which is not practically feasible in many applications.

In this chapter, a novel biologically inspired neural network approach, based on the model in Yang and Meng (2000b) for path planning of mobile robots with completely known environment, is proposed for real-time simultaneous map building and path planning of mobile robots in a dynamic environment, where the environment is assumed completely unknown. The state space of the topologically organized neural network is the Cartesian workspace, where the dynamics of each neuron is characterized by a shunting equation that was derived from Hodgkin and Huxley's (1952) membrane model for a biological system. The robot navigation is based on the target location, and robot sensor readings that are limited to a short range. The real-time robot path is generated from the neural activity landscape of the neural network that adapts changes according to the target location and the known map of the workspace. A map of the environment is built in real time when the robot is moving toward the target, where the sensor readings are obtained from the onboard sensors of the mobile robot that are limited to a certain local range only.

THE MODEL

In this section, the originality of the proposed neural network approach is briefly introduced. Then, the philosophy of the proposed neural network approach and the model algorithm are presented. Finally, the stability of the proposed model is proven using both qualitative analysis and a Lyapunov stability theory.

Originality

Hodgkin and Huxley (1952) proposed a computational model for a patch of membrane in a biological neural system using electrical circuit elements. This modeling work, together with other experimental work, led them to a Nobel Prize in 1963 for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane. In Hodgkin and Huxley's (1952) membrane model, the dynamics of voltage across the membrane, V_m , is described using a state equation technique such as:

$$C_m \frac{dV_m}{dt} = -(E_p + V_m)g_p + (E_{Na} + V_m)g_{Na} - (E_K + V_m)g_K \quad (1)$$

where C_m is the membrane capacitance. Parameters E_K , E_{Na} and E_p are the Nernst potentials (saturation potentials) for potassium ions, sodium ions and the passive leak current in the membrane, respectively. Parameters g_K , g_{Na} and g_p represent the conductance of potassium, sodium and passive channels, respectively. This model

provided the foundation of the shunting model and led to a lot of model variations and applications (Grossberg, 1988).

By setting $C_m = 1$, substituting $x_i = E_p + V_m$, $A = g_p$, $B = E_{Na} + E_p$, $D = E_k - E_p$, $S_i^e = g_{Na}$ and $S_i^i = g_K$ in Eqn. (1), a typical shunting equation is obtained (Ogmen & Gagne, 1990a, 1990b) as:

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)S_i^e(t) - (D + x_i)S_i^i(t) \quad (2)$$

where variable x_i is the neural activity (membrane potential) of the i -th neuron. Parameters A , B and D are non-negative constants representing the passive decay rate, the upper and lower bounds of the neural activity, respectively. Variables S_i^e and S_i^i are the excitatory and inhibitory inputs to the neuron (Ogmen & Gagne, 1990a, 1990b; Yang, 1999). This shunting model was first proposed by Grossberg to understand the real-time adaptive behavior of individuals to complex and dynamic environmental contingencies (Grossberg, 1973, 1982, 1983, 1988), and has a lot of applications in biological and machine vision, sensory motor control, and many other areas (e.g., Grossberg, 1982, 1988; Ogmen & Gagne, 1990a, 1990b; Ogmen, 1993; Zalama et al., 1995; Gaudiano et al., 1996; Yang, 1999).

Model Algorithm

The fundamental concept of the proposed model is to develop a neural network architecture, whose dynamic neural activity landscape represents the limited knowledge of the dynamically varying environment from onboard robot sensors. By properly defining the external inputs from the varying environment and internal neural connections, the target and obstacles are guaranteed to stay at the peak and the valley of the activity landscape of the neural network, respectively. The target globally attracts the robot in the whole state space through neural activity propagation, while the obstacles have only local effect in a small region to avoid collisions and to achieve the clearance from obstacles. The real-time collision-free robot motion is planned through the dynamic activity landscape of the neural network.

The neural network architecture of the proposed model is a discrete topologically organized map that is used in several neural network models (Glasius et al., 1995; Yang & Meng, 2000a, 2000b). The proposed model is expressed in a finite (F -) dimensional (F -D) state space, which can be either the Cartesian workspace or the configuration joint space of a multi-joint manipulator. The location of the i -th neuron at the grid in the F -D state space, denoted by a vector $q_i \in R^F$, represents

a position in the workspace or a configuration in the joint space. The target globally attracts the robot through neural activity propagation, while the obstacles push the robot only locally in a small region to avoid collision. To take into account the clearance from obstacles, there are both excitatory and inhibitory lateral connections. The dynamics of the i -th neuron in the neuron network is given by a shunting equation:

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \left([I_i]^+ + \sum_{j=1}^k w_{ij} [x_j]^+ \right) - (D + x_i) \left([I_i]^- + \sum_{j=1}^k v_{ij} [x_j - \sigma]^- \right) \quad (3)$$

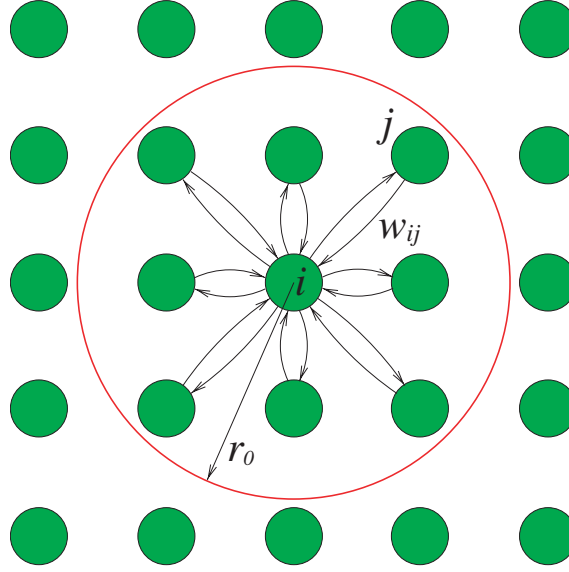
where the excitatory and inhibitory inputs are $[I_i]^+ + \sum_{j=1}^k w_{ij} [x_j]^+$ and $[I_i]^- + \sum_{j=1}^k v_{ij} [x_j - \sigma]^-$, respectively. The external input I_i to the i -th neuron is defined as: $I_i = E$, if there is a target; $I_i = -E$, if there is an obstacle; $I_i = 0$, otherwise, where E is a very large positive constant over its total lateral input. Unlike those models in Yang and Meng (2000a, 2000b) where the whole environment is assumed to be completely known, the proposed model assumes that initially the environment is *completely unknown*, except that the robot knows the target location. Thus the external input I_i depends on the known information of the environment from its onboard sensors whose capacity is limited to a certain local range. A map of the environment is building from the sensor information during the real-time robot navigation.

The function $[a]^+$ is a linear-above-threshold function defined as, $[a]^+ = \max(a, 0)$, and the non-linear function $[a]^-$ is defined as $[a]^- = \max(-a, 0)$. The weights of the excitatory and inhibitory connections, w_{ij} and v_{ij} , from the i -th neuron to the j -th neuron are defined as:

$$w_{ij} = f(|q_i - q_j|) \text{ and } v_{ij} = \beta w_{ij} \quad (4)$$

respectively, where β is a positive constant, $\beta \in [0, 1]$, and $|q_i - q_j|$ represents the Euclidean distance between vectors q_i and q_j in the state space. Function $f(a)$ is a monotonically decreasing function, such as a function defined as: $f(a) = \mu/a$, if $0 < a < r_0$; $f(a) = 0$, if $a \geq r_0$, where μ and r_0 are positive constants. Therefore, it is obvious that the neural connection weights w_{ij} and v_{ij} are symmetric. The neuron has only local connections in a small region $(0, r_0)$, i.e., its receptive field is the space whose distance to the i -th neuron is less than r_0 . The neurons located within the receptive field of the i -th neuron are referred as its *neighboring neurons*. The parameter k is the total number of neighboring neurons of the i -th neuron. Parameter σ is the threshold of the inhibitory lateral neural connections. The threshold of the

Figure 1: Schematic diagram of the neural network for robot path planning when the state space is 2D; the i -th neuron has only 8 lateral connections to its neighboring neurons that are within its receptive field



excitatory connections is chosen as a constant zero. A schematic diagram of the neural network in 2D is shown in Figure 1, where r_0 is chosen as $r_0=2$. The receptive field of the i -th neuron is represented by a circle with a radius of r_0 .

The proposed neural network characterized by Eqn. (3) guarantees that the positive neural activity can propagate to the whole state space. However, the negative activity stays locally only in a small region, due to the existence of the threshold σ of the inhibitory lateral connections. Therefore, the target *globally* influences the whole state space to attract the robot, while the obstacles have only *local* effect to avoid collision. In addition, by choosing different β and/or σ values, the local influence from the obstacles is adjusted, and a suitable strength of clearance from obstacles is selected. Therefore, the proposed model is capable of planning the shortest path from the starting position to the target, or a safer path, or the safest path, depending on the different requirement.

The positions of the target and obstacles may vary with time. The activity landscape of the neuron network dynamically changes due to the varying external inputs and the internal lateral connections. The optimal path is generated from the dynamic activity landscape by a gradient ascent rule. For a given *present position* in the workspace or in the robot manipulator joint space, denoted by q_p , the *next position* q_n (also called “command position”) is obtained by:

$$p_n \Leftarrow x_{p_n} = \max \{x_j, j = 1, 2, \dots, k\} \quad (5)$$

where k is the number of the neighboring neurons, i.e., all the possible next positions of the present position q_p . After the *present position* reaches its *next position*, the *next position* becomes a *new present position*. The present position *adaptively* changes according to the varying environment. The speed of the robot can be defined as a function of its distance to the nearest obstacle, e.g., a function defined as:

$$v = \begin{cases} v_m, & \text{if } d \geq d_0 \\ v_m d / d_0, & \text{otherwise} \end{cases} \quad (6)$$

where v_m is the maximum robot speed, d_0 is a positive constant and d_0 is the Euclidean distance from robot to its nearest obstacle.

The dynamic activity landscape of the topologically organized neural network is used to determine *where* the next robot position should be. However, *when* the robot moves to the next position is determined by the robot moving speed. In a static environment, the activity landscape of the neural network will reach a steady state, which will later be proven using the Lyapunov stability theory. Mostly the robot reaches the target much earlier than the activity landscape reaches the steady state of the neural network. When a robot is in a dynamically changing environment, the neural activity landscape will never reach a steady state. Due to the very large external input constant E , the target and the obstacles stay at the peak and the valley of the activity landscape of the neural network, respectively. The robot keeps moving toward the target with obstacle avoidance till the designated objectives are achieved.

Stability Analysis

In the shunting model in Eqn. (2) and (3), the neural activity x_i increases at a rate of $(B - x_i)S_i^e$, which is not only proportional to the excitatory input S_i^e , but also proportional to an auto gain control term $(B - x_i)$. Thus, with an equal amount of input S_i^e , the closer the values of x_i and B are, the slower x_i increases. When the activity x_i is below B , the excitatory term is positive, causing an increase in the neural activity. If x_i is equal to B , the excitatory term becomes zero, and x_i will no longer increase no matter how strong the excitatory input is. In case the activity x_i exceeds B , $B - x_i$ becomes negative and the shunting term pulls x_i back to B . Therefore, x_i is forced to stay below B , the upper bound of the neural activity. Similarly, the inhibitory term forces the neural activity to stay above the lower bound $-D$.

Therefore, once the activity goes into the finite region $[-D, B]$, it is guaranteed that the neural activity will stay in this region for any value of the total excitatory and inhibitory inputs (Yang, 1999).

The stability and convergence of the proposed model can also be rigorously proven using a Lyapunov stability theory. From the definition of $[a]^+$, $[a]^-$ and v_{ij} , Eqn. (3) is rewritten into Grossberg's general form (Grossberg, 1988):

$$\frac{dx_i}{dt} = a(x_i) \left(a(x_i) - \sum_{j=1}^N c_{ij} d_j(x_j) \right) \quad (7)$$

by the following substitutions:

$$a_i(x_i) = \begin{cases} B - x_i, & \text{if } x_i \geq 0 \\ D + x_i & \text{if } x_i < 0 \end{cases}$$

$$b_i(x_i) = \frac{1}{a_i(x_i)} (B[I_i]^+ - C[I_i]^- - (A + [I_i]^+ + [I_i]^-)x_i)$$

$$c_{ij} = -w_{ij}$$

and

$$d_j(x_j) = \begin{cases} x_j, & \text{if } x_j \geq 0 \\ \beta(x_j - \sigma), & \text{if } x_j < \sigma \\ 0, & \text{otherwise} \end{cases}$$

Since the neural connection weight is symmetric, $w_{ij} = w_{ji}$, then $c_{ij} = c_{ji}$ (symmetry). Since B and D are non-negative constants and $x_i \in [-D, B]$, then $a_i(x_i) \geq 0$ (positivity). Since $d'_j(x_j) = 1$ at $x_j > 0$; $d'_j(x_j) = \beta$ at $x_j > \sigma$; and $d'_j(x_j) = 0$, otherwise, then $d_j(x_j) \geq 0$ (monotonicity). Therefore, Eqn. (5) satisfies all the three stability conditions required by Grossberg's general form (Grossberg, 1988). The Lyapunov function candidate for Eqn. (7) can be chosen as:

$$v = -\sum_{i=1}^N \int^{x_i} b_i(y_i) d'_i(y_i) dy_i + \frac{1}{2} \sum_{j,k=1}^N c_{kj} d_j(x_j) d_k(x_k) \quad (8)$$

The derivative of v along all the trajectories is given as:

$$\frac{dv}{dt} = -\sum_{i=1}^N a_i d_i' (b_i - \sum_{j=1}^N c_{ij} d_j)^2$$

Since $a_i \geq 0$ and $d_i' \geq 0$, then $dv/dt \leq 0$ along all the trajectories. The rigorous proof of the stability and convergence of Eqn. (7) can be found in Grossberg (1983). Therefore, the proposed neural network system is stable. The dynamics of the network is guaranteed to converge to an equilibrium state of the system.

SIMULATIONS

To demonstrate the effectiveness of the proposed neural network model, the proposed neural network model for real-time map building and path planning is applied to a room-like environment in Figure 2A, where the static obstacles are shown by solid squares. The target is located at position (30,30), while the starting position of the robot is at (4,4). The neural network has 50×50 topologically organized neurons, with zero initial neural activities. The model parameters are chosen as: $A = 10$ and $B = D = 1$ for the shunting equation; $\mu = 1$, $\beta = 0.8$, $\sigma = -0.8$ and $r_0 = 2$ for the lateral connections; and $E = 100$ for the external inputs. Three cases are carried out. In the first case, same as the models in Yang and Meng (2000a, 2000b), it is assumed that the environment is completely known by some additional sensors in the workspace. The generated robot path is shown in Figure 2A, where the robot is represented by circles. It shows that the robot can reach the target without obstacle avoidance. The neural activity landscape when the robot arrives at the target location is shown in Figure 2A, where the peak is at the target location, while the valleys are at the obstacle locations.

In the second case, the environment is assumed to be completely unknown, except the target location. The onboard robot sensors can “see” in a limited range within a radius of $R = 5$ (see the circle in lower right corner of Figure 3A). As shown in Figure 3A, initially the robot sees some obstacles on its back, but there are no obstacles in its front direction toward the target. Thus the robot moves straight forward to the target. However, when the robot arrives at location (16,16), it starts to sense the obstacle in its front. When the robot arrives at (18,18), the built map is shown in Figure 3A, where a few obstacles are detected by its onboard sensors, and the activity landscape is shown in Figure 3B. The robot is moving toward the target along a collision-free path; more and more obstacles were detected. When the robot arrives at (18,41), the built map and the activity landscape are shown in

Figure 2: Path planning with safety consideration when the environment is completely known--A: the dynamic robot path; B: the activity landscape when the robot reaches the target

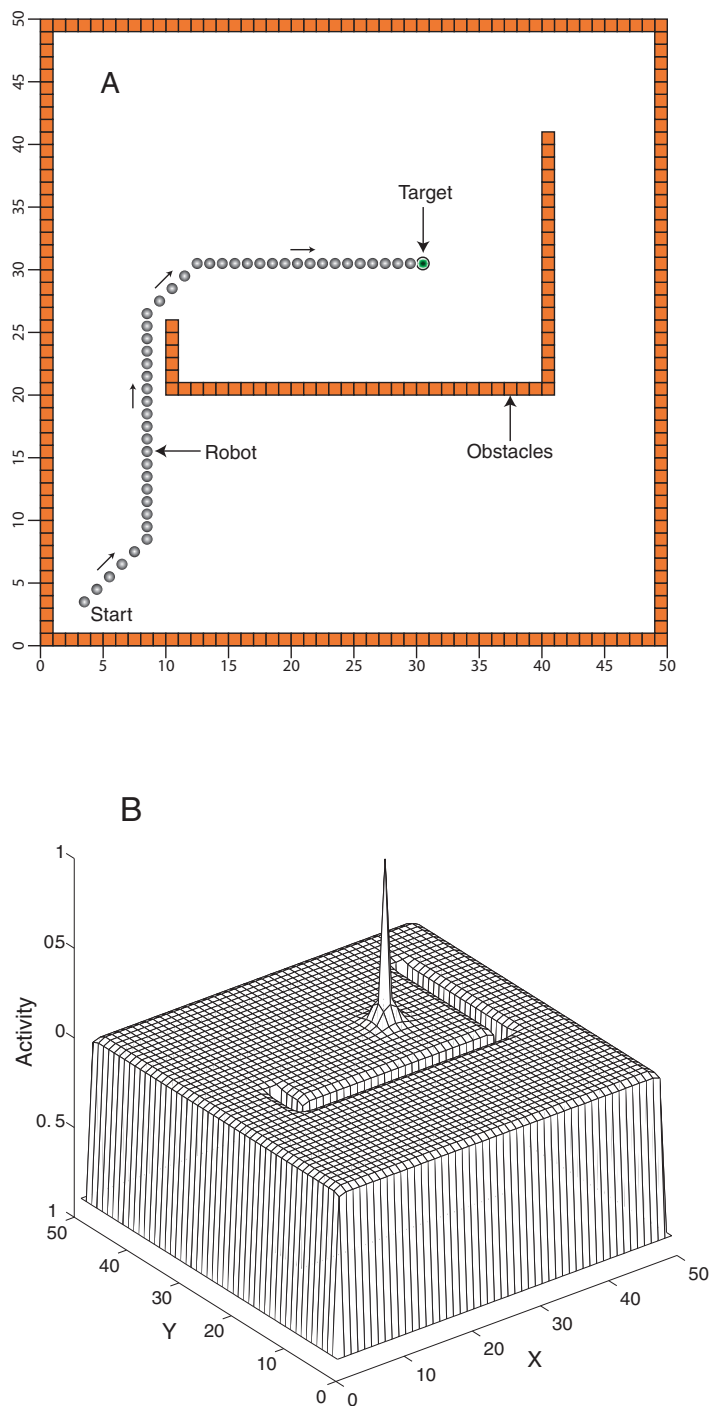


Figure 3: Map building and path planning with sensor information limited to a radius or $R=5$ --A and B: the dynamic robot path and the environment map (A) and the neural activity landscape (B) when the robot arrives at (18,18); C and D: the dynamic robot path and the environment map (C) and the neural activity landscape (D) when the robot arrives at (18,41)

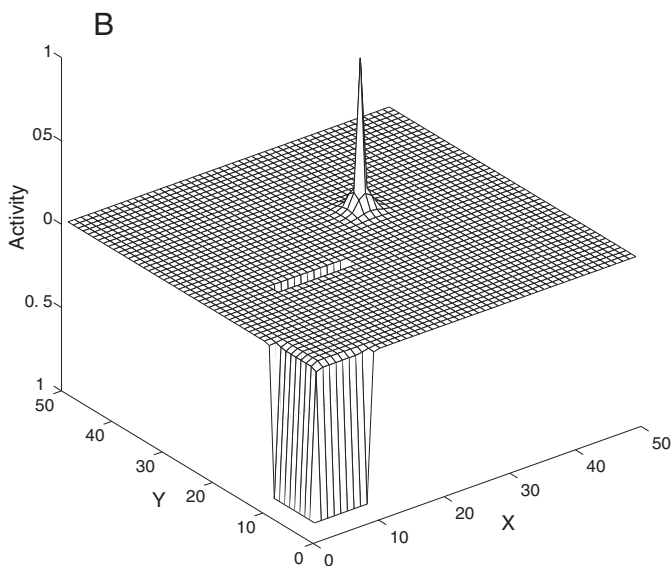
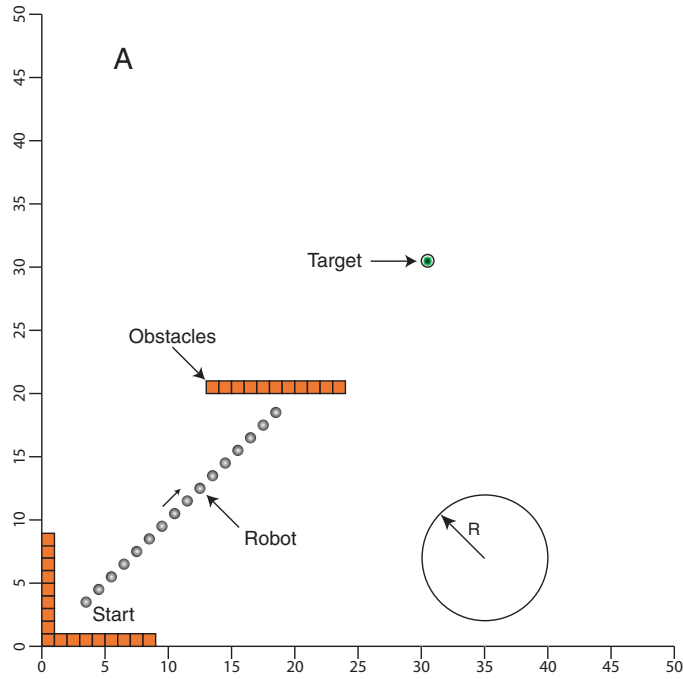
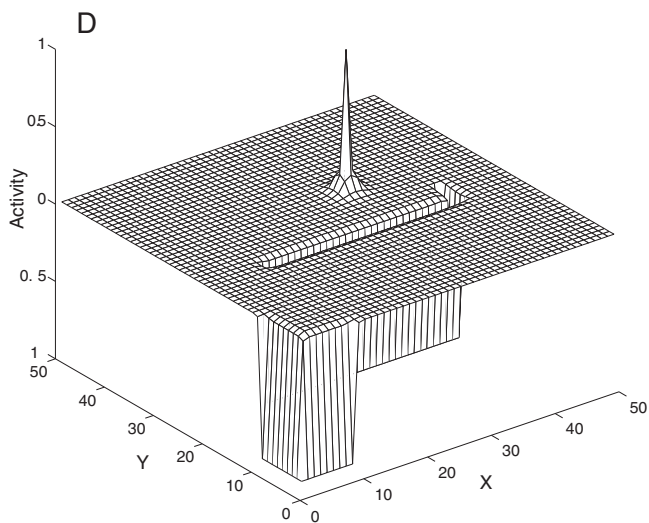
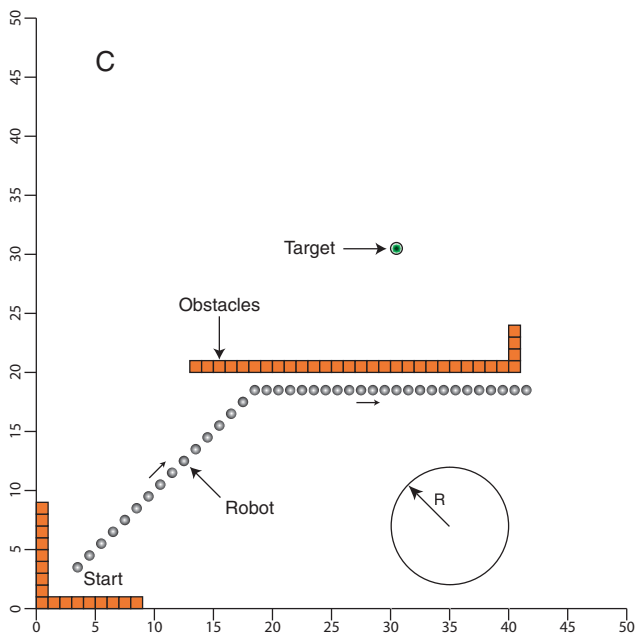


Figure 3: (continued) (C) the neural activity landscape (D) when the robot arrives at (18,41)



*Figure 4: Map building and path planning with limited sensor information--
A: the robot path and the built map in same case in Figure 3; B: the robot path and the built map when there are obstacles suddenly placed in its front to close to the gate to the target when the robot arrives at (42,41) (marked by an arrow) in the case in left panel*

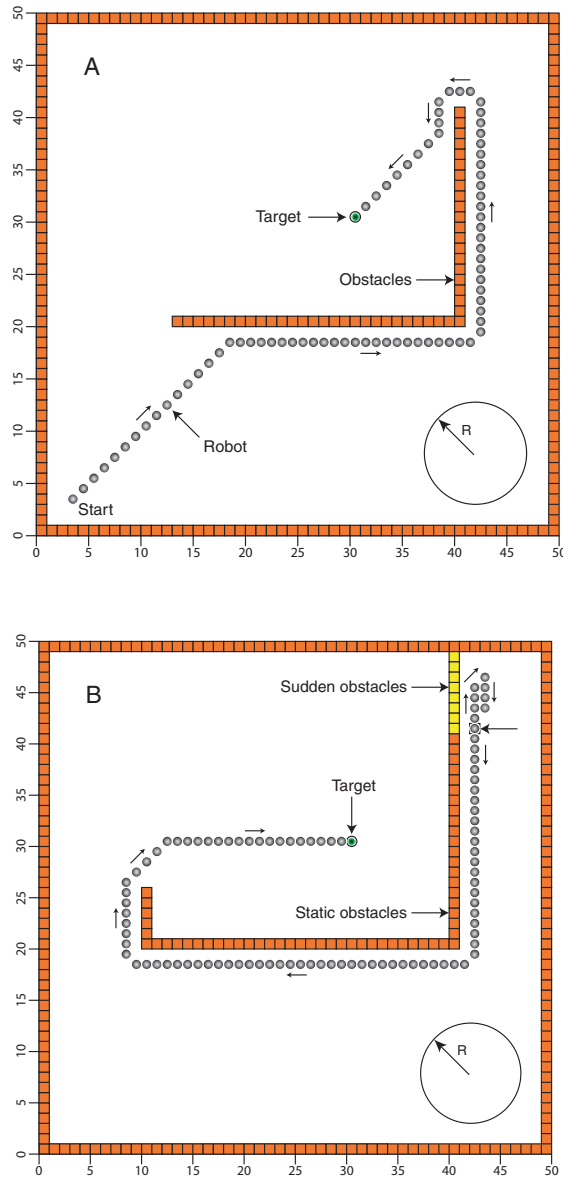


Figure 3C and Figure 3D, respectively. The robot continues to move toward the target. The robot traveling path to reach the target and the built map are shown in Figure 3A.

In the third case, the condition initially is the same as in Case 2. However, when the robot arrives at (42, 41), there are obstacles suddenly placed in front of the robot, which close the gate for the robot to reach the target. The robot has to move around there, and finally the robot has to move back, pass around the obstacles and finally reach the target from the other side. The robot path after the sudden obstacles were placed is shown in Figure 4B. It shows that the robot is capable of reaching the target with obstacle clearance.

DISCUSSIONS

In this section, the parameter sensitivity of the proposed neural network model will be discussed. Then a simple model characterized by an additive equation is obtained from the proposed shunting model by removing the auto gain control terms and lumping together the excitatory and inhibitory inputs.

Parameter Sensitivity

Parameter sensitivity is a very important factor to be considered when a model is proposed or evaluated. An acceptable model should be robust, i.e., not very sensitive to changes in its parameter values. There are only few parameters in the proposed model in Eqn. (3). The upper and lower activity bounds B and D , the receptive field parameter r_0 and the external input constant E are not important factors. The passive decay rate A determines the transient response of the neurons, which is very important for the model dynamics, particularly when the target and the obstacle are varying fast. The lateral connection weight parameter μ is also an important factor, which determines the propagation of the neural activity in the neural network. The relative inhibitory lateral connection parameter β and the threshold of the inhibitory connections σ determine the strength of the clearance from obstacles. They are very important factors as well. A detailed discussion through description and simulation of the model parameters can be found in Yang and Meng (2000b, 2001).

The proposed model is not very sensitive to the variations of model parameters and the connection weight function. The parameters can be chosen in a very wide range. The weight function can be any monotonically decreasing function (Yang & Meng, 2001).

Model Variation

If the excitatory and inhibitory connections in the shunting equation in Eqn. (3) are lumped together and the auto gain control terms are removed, then a simpler form can be obtained from Eqn. (3):

$$\frac{dx_i}{dt} = -Ax_i + I_i + \sum_{j=1}^k w_{ij}[x_j]^+ - \sum_{j=1}^k v_{ij}[x_j - \sigma]^-, \quad (9)$$

This is an additive equation (Grossberg, 1988). The term $I_i + \sum_{j=1}^k w_{ij}[x_j]^+ - \sum_{j=1}^k v_{ij}[x_j - \sigma]^-$ represents the total inputs to the i -th neuron from the external and internal connections. The non-linear functions $[a]^+$, $[a]^-$ and the threshold σ are defined as the same as earlier in this chapter, which together guarantee that the positive neural activity can propagate to the whole workspace while the negative activity can propagate locally in a small region only. Thus the target globally attracts the robot in the whole workspace, while the obstacles have only local effects to achieve clearance from obstacles. Therefore this additive model satisfies the fundamental concepts of the proposed approach described earlier in this chapter. It is capable of simultaneously planning robot path and building environment map in most situations. From the definition of $[a]^+$, $[a]^-$ and v_{ij} Eqn. (9) can be further rewritten into a compact form as:

$$\frac{dx_i}{dt} = -Ax_i + I_i + \sum_{j=1}^k w_{ij}d(x_j) \quad (10)$$

where $d(x_j)$ is defined in Eqn. (7). The stability of this additive model can also be proven using a Lyapunov stability theory, although its neural activity does not have any bounds. Eqn. (9) can be rewritten into Grossberg's general form in Eqn. (7) by variable substitutions. It is easy to prove that Eqn. (9) satisfies all the three stability conditions of Eqn. (7) (Grossberg, 1988; Yang, 1999, Yang & Meng, 2001). Therefore this additive neural network system is stable.

There are many important differences between the shunting model and the additive model, although the additive model is computationally simpler. By rewriting them into the general form in Eqn. (7), unlike the additive model in Eqn. (9), the amplification function $a_i(x_i)$ of the shunting model in Eqn. (3) is not a constant, and the self-signal function $b_i(x_i)$ is non-linear. The shunting model in Eqn. (3) has two *auto gain control* terms, $(B-x_i)$ and $(D+x_i)$, which result in that the dynamics of

Eqn. (3) remain sensitive to input fluctuations (Grossberg, 1988). Such a property is important for the real-time robot path planning when the target and obstacles are varying. In contrast, the dynamics of the additive equation may saturate in many situations (Grossberg, 1988). Furthermore, the activity of the shunting model is bounded in the finite interval $[-D, B]$, while the activity in the additive model does not have any bounds (Grossberg, 1988; Ogmen & Gagne, 1990a, 1990b; Yang & Meng, 2000a, 2000b, 2001).

CONCLUSION

In this chapter, a novel biologically inspired neural network model is proposed for the real-time map building and path planning with safety consideration. Several points are worth noticing about the proposed model:

- The strength of the clearance from obstacles is adjustable. By changing suitable model parameters, this model is capable of planning the shortest path, or a comfortable path, or the safest path (Yang & Meng, 2000b).
- The algorithm is computationally efficient. The map is built during the robot navigation, and the robot path is planned through the dynamic neural activity landscape *without* any prior knowledge of the dynamic environment, *without* explicitly searching over the free space or the collision paths, *without* explicitly optimizing any cost functions and *without* any learning procedures.
- The model can perform properly in an arbitrarily varying environment, even with a sudden environmental change, such as suddenly adding or removing obstacles.
- The model is biologically plausible. The neural activity is a continuous analog signal and has both upper and lower bounds. In addition, the continuous activity prevents the possible oscillations related to parallel dynamics of discrete neurons (Glasius et al., 1995; Marcus, Waugh & Westervelt, 1990).
- This model is not very sensitive to the model parameters and the connection weight function. The parameters can be chosen in a very wide range. The weight function can be any monotonically decreasing function.

ACKNOWLEDGMENTS

This work was supported by Natural Sciences and Engineering Research Council (NSERC) and Materials and Manufacturing Ontario (MMO) of Canada.

REFERENCES

- Al-Sultan, K. S. & Aliyu, D. S. (1996). A new potential field-based algorithm for path planning. *J. of Intelligent and Robotic Systems*, 17(3), 265-282.
- Beom, H. R. & Cho, H. S. (1995). Sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *IEEE Trans. on Systems, Man, and Cybernetics*, 25(3), 464-477.
- Chang, C. C. & Song, K. T. (1997). Environment prediction for a mobile robot in a dynamic environment. *IEEE Trans. on Robotics and Automation*, 13(6), 862-872.
- Gaudiano, P., Zalama, E. & Lopez Coronado, J. (1996). An unsupervised neural network for low-level control of a mobile robot: Noise resistance, stability, and hardware implementation. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 26(3), 485-496.
- Glasius, R., Komoda, A. & Gielen, S. C. A. M. (1994). Population coding in a neural net for trajectory formation. *Network: Computation in Neural Systems*, 5, 549-563.
- Glasius, R., Komoda, A. & Gielen, S. C. A. M. (1995). Neural network dynamics for path planning and obstacle avoidance. *Neural Networks*, 8(1), 125-133.
- Grossberg, S. (1973). Contour enhancement, short-term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, 52, 217-257.
- Grossberg, S. (1982). *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*. Boston: Reidel Press.
- Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Trans. Systems, Man, and Cybernetics*, 13(5), 815-926.
- Grossberg, S. (1988). Non-linear neural networks: Principles, mechanisms, and architecture. *Neural Networks*, 1, 17-61.
- Hodgkin, A. L. & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology London*, 117, 500-544.
- Ilari, J. & Torras, C. (1990). 2nd path planning: A configuration space heuristic approach. *International Journal of Robotics Research*, 9(1), 75-91.
- Li, L. & Ogmen, H. (1994). Visually guided motor control: Adaptive Sensorimotor mapping with on-line visual-error correction. In: *Proceedings of the World Congress on Neural Networks*. 127-134.
- Li, Z. X. & Bui, T. D. (1998). Robot path planning using fluid model. *Journal of Intelligent and Robotic Systems*, 21, 29-50.

- Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE Trans. Computers*, 32, 108-320.
- Marcus, C. M., Waugh, F. R. & Westervelt, R. M. (1990). Associative memory in an analog iterated-map neural network. *Physical Review A*, 41(6), 3355-3364.
- Ogmen, H. & Gagne, S. (1990a). Neural models for sustained and on-off units of insect lamina. *Biological Cybernetics*, 63, 51-60.
- Ogmen, H. & Gagne, S. (1990b). Neural network architecture for motion perception and elementary motion detection in the fly visual system. *Neural Networks*, 3, 487-505.
- Ong, C. J. & Gilbert, E. G. (1998). Robot path planning with penetration growth distance. *Journal of Robotic Systems*, 15(2), 57-74.
- Yang, S. X. & Meng, M. (2000a). Neural network approach to real-time collision-free motion planning. *Neural Networks*, 13(2), 133-148.
- Yang, S. X. & Meng, M. (2000b). An efficient neural network method for real-time motion planning with safety consideration. *Robotics and Autonomous Systems*, 32(2-3), 115-128.
- Yang, S. X. & Meng, M. (2001). Neural network approaches to dynamic collision-free trajectory generation. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 31 (3), 302-318.
- Yang, X. (1999). *Neural Network Approaches to Real-Time Motion Planning and Control of Robotic Systems*. PhD dissertation, University of Alberta, Canada.
- Zalama, E., Gaudiano, P. & Lopez Coronado, J. (1995). A real-time, unsupervised neural network for the low-level control of a mobile robot in a non-stationary environment. *Neural Networks*, 8, 103-123.
- Zelinsky, A. (1994). Using path transforms to guide the search for findpath in 2nd *International Journal of Robotics Research*, 13(4), 315-325.

SECTION II:

**HYBRID
EVOLUTIONARY
SYSTEMS FOR
MODELLING,
CONTROL
AND ROBOTICS
APPLICATIONS**

Chapter V

Evolutionary Learning of Fuzzy Control in Robot-Soccer

P.J.Thomas and R.J.Stonier
Central Queensland University, Australia

ABSTRACT

In this chapter an evolutionary algorithm is developed to learn a fuzzy knowledge base for the control of a soccer micro-robot from any configuration belonging to a grid of initial configurations, to hit the ball along the ball to goal line of sight. A relative coordinate system is used. Forward and reverse mode of the robot and its physical dimensions are incorporated, as well as special considerations to cases when in its initial configuration, the robot is touching the ball.

INTRODUCTION

An important aspect of fuzzy logic application is the determination of a fuzzy logic knowledge base to satisfactorily control the specified system, whether this is derivable from an appropriate mathematical model or just from system input-output data. Inherent in this are two main problems. The first is to obtain an adequate knowledge base (KB) for the controller, usually obtained from expert knowledge, and second is that of selection of key parameters defined in the method.

The KB is typically generated by expert knowledge but a fundamental weakness with this static acquisition is that it is frequently incomplete, and its control strategies are conservative. To overcome this one approach is to construct self-organising fuzzy logic controllers (Yan, 1994). These self-organising fuzzy logic controllers are used mainly for the creation and modification of the rule base. Of interest is the question of how this self-organisation and adaptation can be carried out in an automated fashion. One way is to incorporate genetic/evolutionary algorithms to form genetic fuzzy systems, (Karr, 1991; Thrift, 1991; Cordón, 1995).

Evolutionary learning of fuzzy controllers in a three-level hierarchical, fuzzy logic system to solve a collision-avoidance problem in a simulated two-robot system is discussed in Mohammadian (1998a). A key issue is that of learning knowledge in a given layer sufficient for use in higher layers. We need to find a KB that is effective, to some acceptable measure, in controlling the robot to its target from ‘any’ initial configuration. One way is to first learn a set of local fuzzy controllers, each KB learned by an evolutionary algorithm from a given initial configuration within a set of initial configurations spread uniformly over the configuration space. These KBs can then be *fused* through a *fuzzy amalgamation* process (Mohammadian, 1994, 1998b; Stonier, 1995a, 1995b), into the global (final), fuzzy control knowledge base. An alternative approach (Mohammadian, 1996; Stonier, 1998), is to develop an evolutionary algorithm to learn directly the ‘final’ KB by itself over the region of initial configurations.

In this chapter we use this latter approach and incorporate special attributes to cover the difficult cases for control when the robot is close and touching the ball. A relative coordinate system is used and terms are introduced into the fitness evaluations that allow both forward and reverse motion of the soccer robot. We define the robot soccer system, the design of the fuzzy controller, the design of the evolutionary algorithm and finish with a short presentation of results for control of the robot from a far distance from the ball and from configurations close and touching the ball.

ROBOT SOCCER SYSTEM

The basic robot soccer system considered is that defined for the Federation of Robot-Soccer Association, Robot World Cup (www.fira.net). All calculations for vision data processing, strategies and position control of the robots are performed on a centralised host computer. Full specifications of hardware, software and basic robot strategies that are employed in this type of micro-robot soccer system can be found in Kim (1998).

Kinematics

The kinematics of a wheelchair-style robot is given by Equation 1 from Jung (1999).

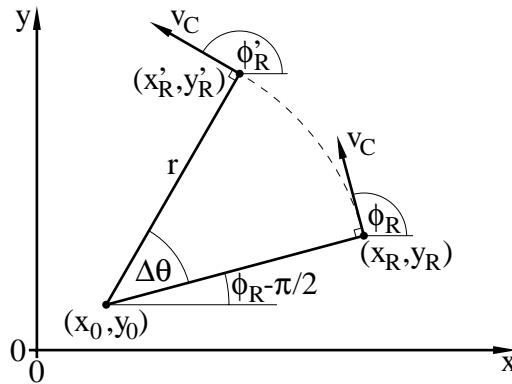
$$\begin{bmatrix} v_C \\ \omega \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ -1/L & 1/L \end{bmatrix} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \quad (1)$$

where v_L is the instantaneous speed at the left wheel of the robot, v_R is the instantaneous speed at the right wheel of the robot, L is the wheel base length, v_C is the instantaneous speed of the robot centre, ω is the instantaneous angular speed about the instantaneous point of rotation (x_0, y_0) . The radius of the arc r is determined from $v_C = r \omega$, which is the distance between (x_0, y_0) and v_C .

Let the next robot position be approximated by a small time interval Δt . Assume v_L and v_R are constant over this interval. If $\omega = 0$, the robot is moving in a straight line. Equation 2 gives the next robot position using linear displacement $\Delta s = v_C \Delta t$:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\phi}_R \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ 0 \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\phi_R) \\ \Delta s \sin(\phi_R) \\ \phi_R \end{bmatrix} \quad (2)$$

Figure 1: Curvilinear formulae symbols



When $\omega \neq 0$, the robot scribes an arc. Curvilinear robot paths are calculated using translation, rotation and translation Equation 3. Refer to Figure 1 for the following derivation:

First, determine the point of rotation (x_0, y_0) :

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} - r \begin{bmatrix} \cos(\phi_R - \pi / 2) \\ \sin(\phi_R - \pi / 2) \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} + r \begin{bmatrix} -\sin(\phi_R) \\ \cos(\phi_R) \end{bmatrix}$$

Translate point of rotation to origin:

$$\begin{bmatrix} x_R^1 \\ y_R^1 \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = r \begin{bmatrix} \sin(\phi_R) \\ -\cos(\phi_R) \end{bmatrix}$$

Rotate about the z-axis (counter-clockwise positive):

$$\begin{bmatrix} x_R^2 \\ y_R^2 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} x_R^1 \\ y_R^1 \end{bmatrix} = r \begin{bmatrix} \sin(\phi_R + \Delta\theta) \\ -\cos(\phi_R + \Delta\theta) \end{bmatrix}$$

Translate origin to point of rotation:

$$\begin{bmatrix} x_R' \\ y_R' \end{bmatrix} = \begin{bmatrix} x_R^2 \\ y_R^2 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \end{bmatrix} + r \begin{bmatrix} \sin(\phi_R + \Delta\theta) - \sin(\phi_R) \\ -\cos(\phi_R + \Delta\theta) + \cos(\phi_R) \end{bmatrix}$$

Finish off by adding in robot angle correction:

$$\begin{bmatrix} x_R' \\ y_R' \\ \phi_R' \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ \phi_R \end{bmatrix} + \begin{bmatrix} \sin(\phi_R + \Delta\theta) & -\sin(\phi_R) & 0 \\ -\cos(\phi_R + \Delta\theta) & \cos(\phi_R) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ r \\ \Delta\theta \end{bmatrix} \quad (3)$$

where $\phi'_R \in [0, 2\pi)$ is necessarily constrained for input into the fuzzy system. The following parameter values were used: $L = 68.5(mm)$, $\Delta t = 1/60(s)$ $\Delta t = 1/60(s)$. The playable region was defined as a rectangle from coordinate $(0,0)$ to $(1500,1300)$ (measurements in mm.), ignoring the goal box at each end of the field.

FUZZY CONTROL SYSTEM DESIGN

The discussion on kinematics above shows, excluding momentum and friction, that only two variables, the velocity of the left and right wheels, $y_1 = v_L$ and $y_2 = v_R$, control the motion of the robot. These two variables are taken as output of the fuzzy control system now described below. Input variables for the fuzzy control system are taken to be the position of the robot relative to the ball, described by $n = 3$ variables $x_1 = d^2$, $x_2 = \theta$ and $x_3 = \phi$ as shown in Figure 2.

These relative coordinates were used in preference to Cartesian coordinate variables for a number of reasons, one being that it reduced the number of rules in the fuzzy KB. Distance squared was used to reduce the calculation cost by not using a square root function, effectively applying a “more or less” hedge. The angle of the robot relative to the ball goal line was used instead of the ball robot line because of positional error arising from image capture pixel size in determining the position of each object. The vision system has an inherent $\pm 4.5 mm$ error caused by pixel size. The pixel size error causes the angle of the line error to be inversely proportional to the distance between the points used to calculate the line. However, one of the points used to calculate the \overline{BG} line is at the centre of the goal line. The allowable angle range when close to the goal offsets the error caused in determining the line. The vision system error has negligible effect on placing the ball into the goal when using \overline{BG} as a reference.

Figure 2: Relative coordinate parameters

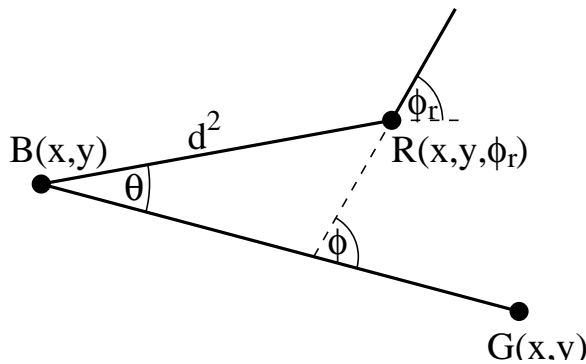


Figure 3: Fuzzy input sets

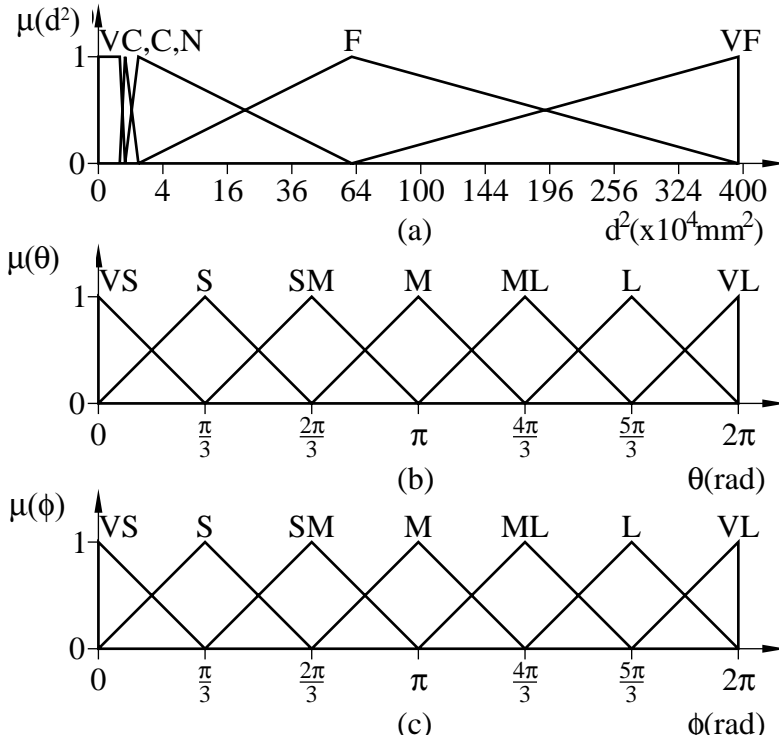


Figure 3 shows the fuzzy input sets used to define the “attack ball strategy.” For all rules seven sets are defined for both angles θ and ϕ : VS is Very Small, S is Small, SM is Small Medium, M is Medium, ML is Medium Large, L is Large and VL is Very Large. Five sets are defined for distance squared: VC is Very Close, C is Close, N is Near, F is Far and VF is Very Far.

The values of y_1 and y_2 are taken to be integers lying in the interval $[-128, 127]$. We take 256 B_k output fuzzy sets each corresponding to centre $\bar{y}_k = -128 + (k - 1)$ for $k = 1, \dots, 256$. In this case the name of the sets are the same as the output centres \bar{y}_k of the sets.

The purpose of taking 256 B_k output fuzzy sets instead of 255, $B_k \in [-127, 127]$, is a technical issue to allow the use of a binary mutation operator in the evolutionary algorithm. The velocities are in fact capped to $v_L, v_R \in [-127, 127]$ before transmission to the robots.

Taking a large number of output sets serves three purposes:

- (i) It does not affect the computational cost of the fuzzy controller; the solution can be as fine as it needs to be.

- (ii) The \bar{y}_k are the control values used for the left and right wheel motors - eliminating conversion.
- (iii) It reduces erratic behaviour of the evolutionary algorithm (finer control) during mutation.

There were $7 \times 7 \times 5 = 245$ rules in a complete fuzzy knowledge base for this system. In general, the j^{th} fuzzy rule has the form:

If (x_1 is A_1^j and x_2 is A_2^j and x_3 is A_3^j)

Then (y_1 is B_1^j and y_2 is B_2^j)

where A_k^j , $k=1,2,3$ are normalised fuzzy sets for input variables x_k , $k=1,2,3$, and where B_m^j , $m=1,2$ are normalised fuzzy sets for output variables y_m , $m=1,2$.

Given a fuzzy rule base with M rules, a fuzzy controller as given in Equation 4 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables:

$$y_k = \frac{\sum_{j=1}^M \bar{y}_k^j \prod_{i=1}^n \mu_{A_i^j}(x_i)}{\sum_{j=1}^M \prod_{i=1}^n \mu_{A_i^j}(x_i)} \quad (4)$$

where \bar{y}_k^j are centres of the output sets B_k^j .

These values, 490 of them, are typically unknown and require determination in establishing valid output for controls to each wheel of the robot. Since there is lack of *a priori* knowledge about the system control, we used evolutionary algorithms (EAs) (Michalewicz, 1994) to search for an acceptable solution.

EVOLUTIONARY LEARNING

Our objective here is to learn a rule base for the control of this system. The first problem is how to formulate the knowledge base as a string in the population.

Each output fuzzy set is represented by an integer in the interval $[-128, 127]$. We can form an individual string \underline{P} as a string of $2M = 490$ consequents (integers under the identification above):

$$\underline{s} = \{s_1^1, s_2^1, \dots, s_1^k, s_2^k, \dots, s_1^M, s_2^M\}$$

where $s_j, j = 1, 2$ is an integer in the interval $[-128, 127]$.

The population at generation t , $P(t) = \underline{s}^n : n = 1, \dots, N$, where N is the number of individuals in the population. The population at the next generation $P(t+1)$ was built using a full replacement policy. Tournament selection, with n_T being the tournament size, determined two parent strings for mating in the current generation. Geometric crossover with probability p_c was used for generating two child strings from the parent strings, for insertion in the next generation's population. In each string, the integer components were stored as two's complement byte-sized quantities, and binary mutation was undertaken on each string in the new population with probability p_m . (Elitism was not used, for it was found to cause premature convergence of the algorithm.)

Fitness evaluation of each individual was calculated by scribing a path using the fuzzy controller and stopping when either:

- (i) iteration (time steps) reached a prescribed limit (100), or
- (ii) the path exceeded the maximum allowable distance from the ball, or
- (iii) the robot collides with the ball.

In (iii) care needs to be taken recognising the finite size of the robot. The robot is a square with size of 80 mm and the ball has a diameter of 42.7 mm . Detecting a collision of the robot and ball is made by calculating the distance of the ball centre $(x_B, y_B) = (750, 650)$ perpendicular to the line in the direction of the robot d_{NL} passing through the centre of the updated position of the robot (x'_R, y'_R) , and the distance of the ball d_{AL} projected onto that line. These values are determined as:

$$d_{AL} = \frac{|(x_B - x'_R) + m(y_B - y'_R)|}{\sqrt{m^2 + 1}}$$

$$d_{NL} = \frac{|(y_B - y'_R) - m(x_B - x'_R)|}{\sqrt{m^2 + 1}}$$

where m is the gradient of the line passing through the robot centre, in the direction of the robot. The following quantity is used to define an exclusion region determined by the physical size of the robot:

A flag "HitBall" is raised when the following condition is true:

IF ((($d_{NL} < 40$) AND ($d_{AL} < 61.35$)) OR (($d_{NL} < 61.35$)
AND ($d_{AL} < 40$)) OR ($r_{corner}^2 < 61.35^2$)) THEN (HitBall = TRUE)

where, $r_{corner}^2 = (d_{AL} - 40)^2 + (d_{NL} - 40)^2$.

The final position of the path was used to evaluate the fitness of each individual as given by Equation 5:

$$\sum_C (T_1 + T_2 + T_3 + T_4) \quad (5)$$

The fitness is calculated as a sum of a number of different quantities, over a set C of initial starting configurations, each configuration specifying robot coordinates x_R, y_R , and angle ϕ describing the orientation of the robot relative to the BG line.

There were 273 initial configurations. The first 28 are defined with the robot touching the ball on each of its four sides in seven different orientations θ around the ball. The remaining initial configurations were defined with seven θ angles on five distance rings from the ball with seven ϕ angles.

The first 28 configurations $c \in [0, 28)$, are given by:

$x_R = x_B + 61.35 \cos(\theta)$, $y_R = y_B + 61.35 \sin(\theta)$, and ϕ with:

$\theta \in \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3, 7\pi/3\}$, and $\phi \in \{0, \pi/2, \pi, 3\pi/2\}$.

The remaining initial configurations $c \in [28, 273)$, are given by:

$x_R = x_B + d \cos(\theta)$, $y_R = y_B + d \sin(\theta)$, and ϕ with:

$d \in \{77.92, 122.75, 310.4, 784.94, 1900\}$,

$\theta \in \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3, 7\pi/3\}$,

$\phi \in \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3, 7\pi/3\}$.

The first quantity in the fitness sum is $T_1 = d^2(R, DP)$. It is the final squared distance between the robot centre R and the destination point $DP = (688.5, 650)$ when the path is terminated as described above. The term is used to determine accuracy of the fuzzy controller to control the system to the desired destination configuration.

The second term T_2 is the iteration count for each path. This quantity is used to minimise the time taken to reach the desired destination configuration.

The third quantity is $T_3 = 1000 \sin^2(\phi)$ where ϕ is the final angle of the robot relative to line \overline{BG} . This term is included to enable forward facing and reverse facing solutions to be accepted at the final destination.

The fourth quantity T_4 is a penalty function that is only applied for those configurations $c \in [0, 28)$. It is described in Equation 6:

$$T_4 = \begin{cases} 10\,000 & \text{if } \theta \in [11\pi/12, 13\pi/12) \text{ and } \sin^2(\phi) > 0.25 \\ 10\,000 & \text{if } \theta \notin [11\pi/12, 13\pi/12) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

It is a constant penalty used to drive the solution away from paths that hit the ball when considering the first 28 initial configurations. Without this penalty, the best solutions obtained via evolutionary learning are invariably ones that try to run through the ball.

The evolutionary algorithm was terminated after a prescribed number of generations. The best individual, that is, the one with the minimum fitness, is taken as the “best” fuzzy logic controller determined by the algorithm for this problem.

RESULTS

The evolutionary algorithm was found to easily learn a fuzzy controller for when fitness was evaluated for a single initial configuration.

Establishing learning over a set of initial configurations from $c=0$ to $c=28$ where the robot was placed in contact with the ball was difficult; appropriate sets of evolutionary parameters needed to be defined with a mutation schedule to ensure diversity in the population at different stages in the learning, for example after every 1,000 generations. The reason for the difficulty was that the algorithm tended to lock fuzzy control into always forward or always reverse motion of the robot, with the consequence that not the shortest distance path was achieved, and invariable penalty constraints were broken.

Learning the fuzzy control over the set of all configurations incorporated the difficulties that had to be overcome for those configurations close to the ball. The algorithm tended to lock into local minima when considering multiple configurations. The local minima existed due to the algorithm finding a good single path amongst the many that influenced nearby paths.

Typical values in simulations were: the size of the population $N = 200$, probability of crossover $p_c = 0.6$ and the number of tournament contestants $n_t = 8$. Mutation probability was defined as a schedule: $p_m = 0.05 - 0.00048(\text{gen} \bmod (1000))$, which decreased mutation with increasing generation number and recommenced with high mutation every 1,000 generations. The evolutionary algorithm was usually run for batches of 10,000 generations.

Due to limited space we present here a few results obtained from the evolutionary learning of the knowledge base, two examples showing the control of the robot from a large distance from the ball, and two showing control of the robot touching the ball.

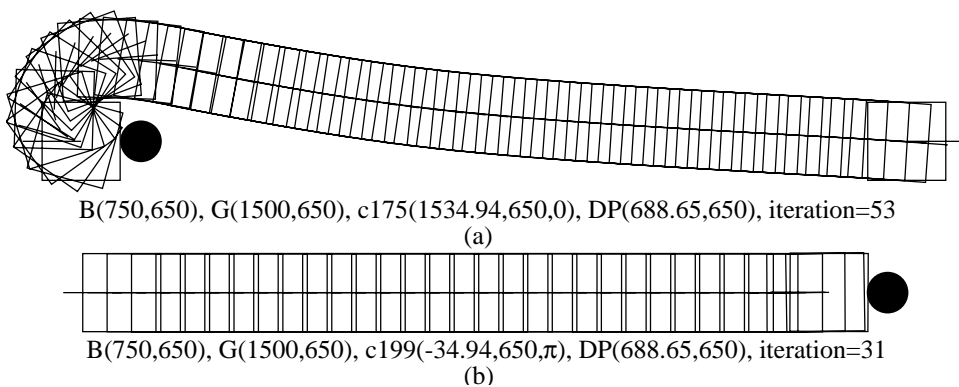
Learning of the Fuzzy Control at a Distance

Figure 4(a) shows the path from initial configuration $c = 175$. For this initial configuration the robot is placed to the far right of the ball and on the ball to goal line. It took 53 time steps to reach the destination point DP with a final angle of ϕ (rad).

In Figure 4(b) the path from initial configuration $c = 199$, with the robot to the far left, facing away from the ball on the ball to goal line, took just 31 time steps to reach the destination point DP with a final angle of ϕ (rad). Note that in both cases the robot approached the destination in reverse mode.

These graphs are typical of the fuzzy control of the robot starting from initial positions at a “large” distance from the ball. Destination and final angle accuracy was excellent. Evolutionary learning was quite rapid, with acceptable solutions resulting in smooth paths to the destination started appearing within a few hundred generations. Further learning resulted in faster control to the destination.

Figure 4: Long distance paths



Learning of Fuzzy Control Close to the Ball

Learning of the fuzzy control of the robot close to the ball was more difficult. Figure 5(a) shows the path from initial configuration $c=1$ which took 19 time steps to reach the destination point DP with final angle of 0 (rad). In Figure 5(b) the path from the initial configuration $c=13$ took 24 time steps to reach the destination point DP with a final angle of 0 (rad). In both cases the robot approached the final destination in forward mode.

The starting initial configurations in which the robot was touching the ball were the most difficult to learn, for they were responsible for the majority of the penalty function evaluations in the fitness calculations for each individual of the evolutionary algorithm. The hardest initial configuration to learn was $c=13$.

COMMENTS

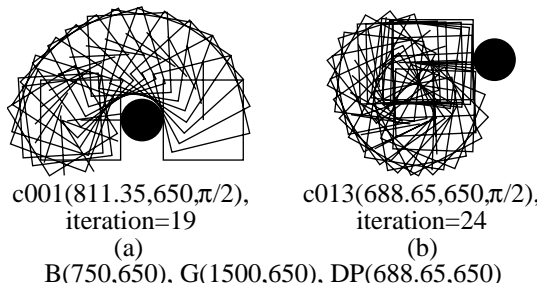
This chapter detailed the learning of a fuzzy logic controller to control a soccer robot to a point behind the ball in the direction of the goal by using an evolutionary algorithm. Learning of a single robot path was very easy. However, learning of several paths from different initial configurations caused many difficulties.

Several starting configuration evaluations caused the final approach of all paths to be either forward or reverse facing. To achieve the final approach heading, the evolutionary algorithm learnt to use chatter and high momentum turns. If a restriction on turning was applied, the algorithm learnt to execute low momentum turns.

The cause of these difficulties was identified as:

- (i) Insufficient number of inputs to the fuzzy system, the rule base could not cater for all of the information need to control the robot to forward and reverse facing solutions.
- (ii) Multi-criteria optimisation problems caused by summing all terms from all path evaluations forming the fitness value.

Figure 5: Short distance paths



NEW RESEARCH

The research presented here has since been extended and a brief description is given. The full analysis and results will be published elsewhere.

The number of input variables for the given problem was extended to include left wheel and right wheel velocities, with the number of membership sets for each variable extended to 7, yielding $7^5 = 16807$ rules in the complete fuzzy knowledge base.

The output variables were changed from left and right wheel velocities to changes in these velocities, namely: Δv_L and Δv_R , with final wheel velocities $v'_L = v_L + \Delta v_L$ and $v'_R = v_R + \Delta v_R$. The number of output member sets was reduced to eight $B_k \in [-28, 28]$, $k = 0, \dots, 7$.

The number of initial configuration was increased to a grid: $x = -750 + 100(k - 1)$ for $k = 1, \dots, 31$ and $y = -650 + 100(k - 1)$ for $k = 1, \dots, 27$ excluding the ball position. Each grid point has five angles: $\theta = 2(k - 1)\pi / 5$ for $k = 1, \dots, 5$. The total number of initial configurations is therefore $C = 5(31 \times 27 - 1) = 4180$. All initial configurations start with zero, left and right, wheel velocity.

Each output fuzzy set is represented by an integer in the interval $[0, 7]$. An individual string \underline{P} is now of length $2M = 33614$ consequents: $\underline{s} = \{s_1^1, s_2^1, \dots, s_1^k, s_2^k, \dots, s_1^M, s_2^M\}$ where s_j , $j = 1, 2$ is an integer in the interval $[0, 7]$.

Evolutionary learning was again used with a population of size $N = 2000$, full replacement policy, tournament selection with size $n_t = 3$ and one point crossover with probability $p_c = 0.6$. Elitism was now used, with the 10 best individuals carried from the old population to the new population. An incremental mutation operator with probability $p_m = 0.01$ replaced the binary mutation used previously. This mutation operator increments/decrements s_k by one with equal probability and has boundary checking, that is, if $s_k = 0$, it was incremented to $s_k = 1$, and if $s_k = 7$, it was decremented to $s_k = 6$.

The final position of the path was again used to evaluate the fitness of each individual as given by Equation 7:

$$\sum_C (\alpha_1 T_1 + \alpha_2 T_2 + \alpha_3 T_3 + \alpha_4 T_4) \quad (7)$$

where $\alpha_1 = 1.0$, $\alpha_2 = 1.0$, $\alpha_3 = 100.0$, $\alpha_4 = 0.0$. The weight coefficients for the first two terms were equal, the terminal angle coefficients was heavily weighted and constant penalty was turned off.

A new learning for the algorithm was implemented as follows. The evolutionary algorithm was run sequentially through the full number of initial configurations, being allowed to run for 10 generations at each configuration before moving to the next. It was stopped after a total of 500,000 generations in all were completed.

The results obtained in the final “best” fuzzy knowledge were excellent, obtaining very smooth continuous paths to the target with both forward and reverse facing in the final position depending on the initial configuration. Only a very small number of aberrations existed, but the paths to the target were still acceptable. Due

Figure 6: Long distant path

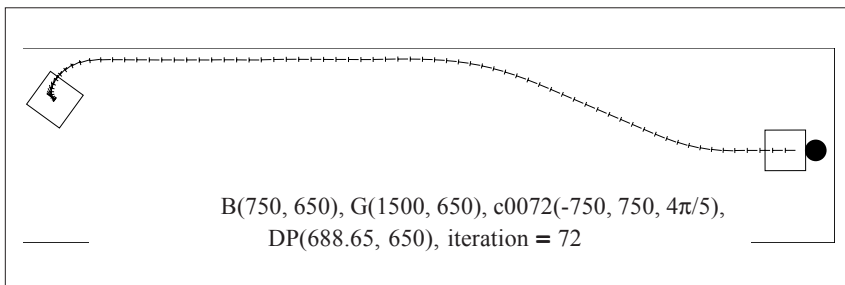
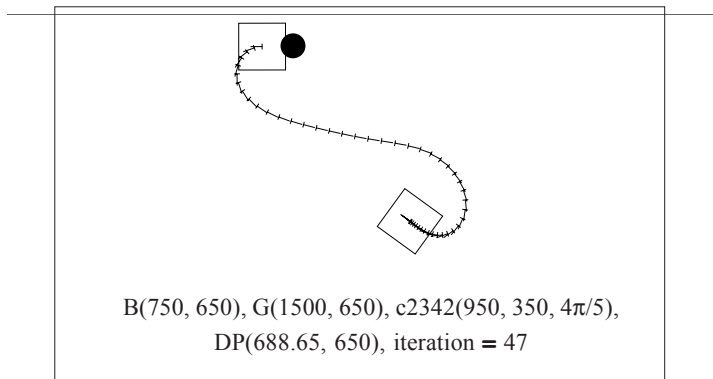


Figure 7: Short distant path



to limited reporting space, we show only two of the many images obtained in Figures 6 and 7. Note one has final approach to the ball forward facing, the other reverse facing; one is from a far distance and one is close to the ball.

This research is being further extended for initial input left and right wheel velocities lying in the full range of admissible values.

REFERENCES

- Cordón, O. & Herrera, F. (1995). A general study on genetic fuzzy systems. In Winter, G & Cuesta, P. (Eds.), *Genetic Algorithms in Engineering and Computer Science*, John Wiley and Sons, 33-57.
- Jung, M.-J., Shim, H.-S., Kim, H.-S. & Kim, J.-H. (1999). The omni-directional mobile robot OmniKity-Y (OK-I) for RoboSOT category. In Stonier, R.J. & Kim, J.H. (Eds) *Proceedings of the FIRA Robot World Cup 1998*, 37-42.
- Karr, C.L. (1991). Applying genetics. *AI Expert*, 38-43.
- Kim, J.H. (1998). *Lecture Notes on Multi-Robot Cooperative System Development*. Green Publishing Co.
- Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs* (2nd Edition). New York: Springer Verlag.
- Mohammadian, M. (1998). *Genetic learning of fuzzy control rules in hierarchical and multi-layer fuzzy logic systems with application to mobile robots*, PhD Thesis, Central Queensland University, Rockhampton, Australia.
- Mohammadian, M. & Stonier, R.J. (1994). Generating fuzzy rules by genetic algorithms. *Proceedings of 3rd International Workshop of Robot and Human Communication*, Nagoya, Japan, 362-367.
- Mohammadian, M. & Stonier, R.J. (1996). Fuzzy rule generation by genetic learning for target tracking. *Proceedings of the 5th International Intelligent Systems Conference*, Reno, Nevada, 10-14.
- Mohammadian, M. & Stonier, R.J. (1998). Hierarchical fuzzy control, *Proceedings of the 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, 621-629.
- Stonier, R.J. (1995). Adaptive learning using genetic algorithms and evolutionary programming in robotic systems, *First Korea-Australia Joint Workshop on Evolutionary Computing*, 183-198.
- Stonier, R.J. & Mohammadian, M. (1995). Self learning hierarchical fuzzy logic controller in multi-robot systems. *Proceedings of the IEA Conference Control95*, Melbourne, Australia, 381-386.

- Stonier, R.J. & Mohammadian, M. (1998). Knowledge acquisition for target capture, *Proceedings of the International Conference on Evolutionary Computing ICEC'98*, Anchorage, Alaska, 721-726.
- Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. *Proceedings of the 4th International Conference on Genetic Algorithms*, 509-513.
- Yan, J., Ryan, M. & Power, J. (1994). *Using Fuzzy Logic*, New York: Prentice Hall.

Chapter VI

Evolutionary Learning of a Box-Pushing Controller

Pieter Spronck, Ida Sprinkhuizen-Kuyper, Eric Postma and Rens Kortmann
Universiteit Maastricht, The Netherlands

Abstract

In our research we use evolutionary algorithms to evolve robot controllers for executing elementary behaviours. This chapter focuses on the behaviour of pushing a box between two walls. The main research question addressed in this chapter is: how can a neural network learn to control the box-pushing task using evolutionary-computation techniques? In answering this question we study the following three characteristics by means of simulation experiments: (1) the fitness function, (2) the neural network topology and (3) the parameters of the evolutionary algorithm. We find that appropriate choices for these characteristics are: (1) a global external fitness function, (2) a recurrent neural network, and (3) a regular evolutionary algorithm augmented with the doping technique in which the initial population is supplied with a solution to a hard task instance. We conclude by stating that our findings on the relatively simple box-pushing behaviour form a good starting point for the evolutionary learning of more complex behaviours.

Introduction

Imagine a cat on a rooftop. It has just spotted a juicy pigeon sitting on a window sill and is wondering how to catch this prey. The situation is tricky: there are two routes the cat can take, both of them involving walking on narrow ledges and requiring daring tricks of balance. The cat decides to take the shortest route and tries to lower itself onto a ledge beneath. While trying to do so, it notices that the chance of toppling over and falling three stories down onto a busy shopping street is becoming increasingly more realistic. The cat now decides to abandon its plan and sets its senses to something more practical.

From a traditional Artificial Intelligence point of view, this navigation problem is not that difficult. The start and goal positions are known, the possible routes are clear, and apparently, the cat has developed a plan to catch the bird. However, the successful execution of the plan critically depends on the cat's low-level interactions with its environment, rather than its high-level planning capabilities. Hitherto, the Artificial Intelligence community has given little attention to low-level control mechanisms (e.g., equilibrium controllers) as compared to high-level controllers (e.g., symbolic problem-solving systems).

Low-level controllers are typically needed for autonomous systems dealing with elementary tasks in dynamic partially observable environments. They form the foundation of high-level controllers executing more complex tasks in the environment (Brooks, 1986). In this chapter we focus on the elementary task of pushing a box between two walls. The box-pushing task was originally introduced (albeit in a slightly different form) by Lee, Hallam and Lund (1997). Pushing an object is an important aspect of robot soccer, a modern platform for autonomous systems research (Asada & Kitano, 1999), and underlies many more complex behaviours such as target following, navigation and object manipulation. While the deceptively simple task of pushing an object is usually disregarded in favour of the seemingly more challenging task of determining a strategic position, pushing is far from trivial and deserves at least as much attention as the strategic task.

The main research question addressed in this chapter is: how can a neural network learn to control the box-pushing task using evolutionary-computation techniques? In answering this question we study the following three characteristics by means of simulation experiments: (1) the fitness function, (2) the neural network topology and (3) the parameters of the evolutionary algorithm.

The outline of the remainder of this chapter is as follows. First, we discuss some background on the use of neural networks and evolutionary algorithms in learning to control a robot. Then we describe the goal of the research in terms of the three characteristics discussed above (i.e., the fitness function, the neural-network

topology and the parameters of the evolutionary algorithm). We give an overview of the standard procedure used in our simulation experiments, followed by a presentation and discussion of the results. Finally, we draw conclusions.

Background

This section provides some background on the approach pursued in our experiments by discussing the use of neural and evolutionary computation techniques in controlling an autonomous robot.

Neural networks offer useful models for learning to control autonomous robots. Although there exist many effective learning algorithms for automatically setting the weights of the network, most algorithms require a carefully prepared set of training examples consisting of pairs of input and desired-output patterns. For freely moving robots in a semi-realistic environment, the preparation of a training set is rather tedious. It is not uncommon that a set of training examples cannot be generated at all. For instance, if the environment in which the controller has to work is (partially) unknown, a training set cannot take into account all the situations that the controller may encounter.

An alternative way to determine the neural network weights is by employing evolutionary algorithms (Bäck, 1996; Yao, 1995). Evolutionary algorithms have many advantages over regular training methods especially for controlling robots (Arkin, 1998). Besides the fact that evolutionary algorithms offer the ability to learn both the weight values and the topology (whereas regular training methods often are limited to determining only the weight values), the only requirement for evolutionary algorithms to work is the availability of a so-called fitness function. A fitness function is an evaluation function indicating the relative success of one solution compared to all the others. Such a fitness function can be defined on a set of examples, comparable to the training set used in most regular training methods. Alternatively, the fitness function can be defined as the quality of the actual behaviour of a neural-network-controlled robot during a test run. This last form of evolutionary learning is called *Genetic Reinforcement Learning* and was relatively unknown until Darrell Whitley (1993) introduced it in his experiments with the GENITOR algorithm.

The main disadvantage of evolutionary algorithms is their inherent unpredictability with respect to the computation time and the quality of the final solution found. It is very hard to predict the time before the algorithm comes up with a satisfactory solution to the problem at hand, and it is often difficult to judge whether significantly better solutions are possible (Goldberg, 1989). In view of these limitations, we

decided to study the application of evolutionary algorithms to a relatively simple box-pushing task extensively in order to find an optimal configuration for the neural controller and the evolutionary algorithm.

Goal

The goal of our study is to find an optimal design of the evolutionary experiments in order to optimise the quality of a box-pushing controller. In particular, our aim is to increase our understanding of how to set up the experiments in an optimal way so that we can be more effective in designing neural controllers for more complex tasks.

In the box-pushing task, the robot is faced with the task of pushing a box as far as possible between two walls in a limited period of time. The inputs of the neural network are the signals received by the robot's proximity sensors. The output of the network rotates the wheels of the robot.

As exemplified in our research question, the following three questions of the experimental set-up are addressed in our research.

1. *What is a suitable fitness function for the evolutionary algorithm?* The fitness function is a measurement of the success of a solution proposed by the evolutionary algorithm, and is the single most important aspect of any evolutionary system.
2. *What is an appropriate neural network topology to solve the box-pushing task?* The main candidates for the neural network are the feedforward and recurrent network topologies. The main advantage of a feedforward network is that it is quick and simple to evolve. The main advantage of a recurrent network is that it has the ability to store and recall values from previous cycles of the robot run.
3. *What is a proper choice for the parameters of the evolutionary algorithm?*

In the next section, we discuss the experimental procedure followed in answering these questions.

Experimental Procedure

The Robot

The robot we used in our studies is of the Khepera type (Mondada et al., 1993). For this robot a good simulation is publicly available. Controllers developed with this simulation have shown to work well in practice. We used the simulator on

Figure 1: Screenshot from the Unix Khepera simulator. To the left the robot world, with the walls (rectangular blocks), the robot (grey circle) and the box (black circle). The small black spots indicate the starting positions used for the robot (the lower three spots) and the box (the upper three spots).

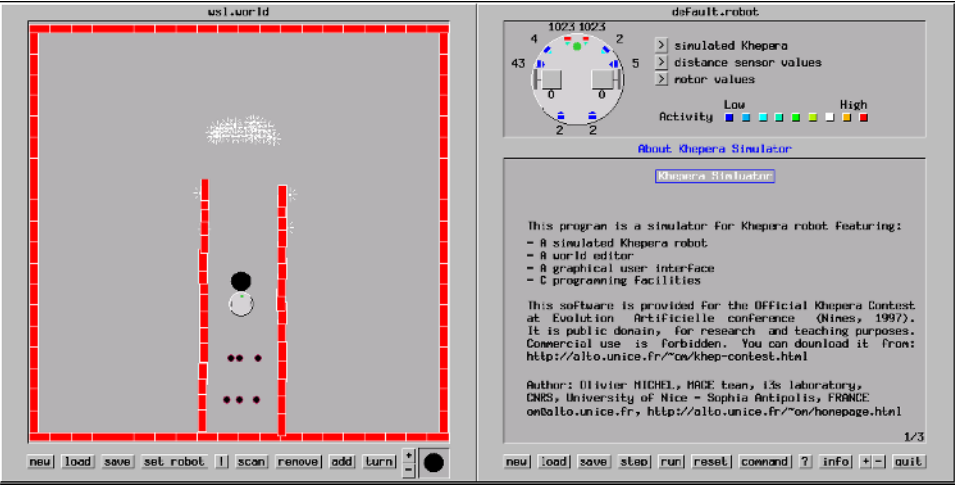
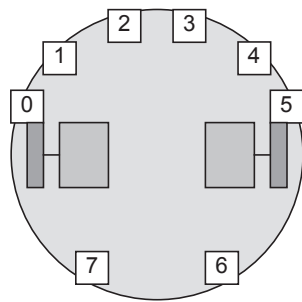


Figure 2: Overview of the Khepera robot



Neural controller inputs:

0 to 7: Sensor 0 to 7 distance values.
8: Sensor 0 - 1
9: Sensor 1 - 2
10: Sensor 2 - 3
11: Sensor 3 - 4
12: Sensor 4 - 5
13: Sensor 6 - 7

Inputs 8 to 13 are the edge detectors.

its original Unix-platform (Figure 1) and also ported it to a Windows-based environment called “Elegance” (Spronck & Kerckhoffs, 1997), that is particularly suited to experiment with different configurations for the evolutionary algorithm.

The Khepera robot possesses eight infra-red light and proximity sensors (Figure 2). In our experiments, we disregarded the light sensors and only used the proximity values (except for the experiments for determining a suitable fitness function). The sensors are numbered clockwise from 0 to 7 starting from the left of the robot. Sensors 2 and 3 point forward, while sensors 6 and 7 point backwards. To control the robot, the two wheels of the robot are supplied with input values

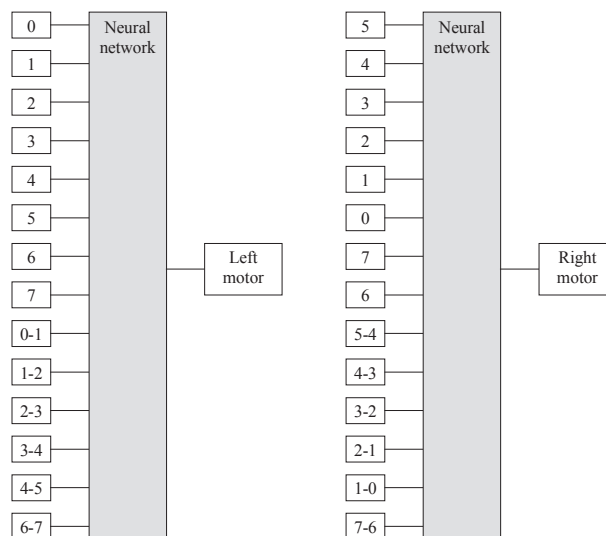
ranging from -10 to $+10$ in integer numbers, where the sign of the input value determines the direction of rotation of the wheels. The robot moves in discrete steps and requires new inputs at each step.

As can be seen from Figure 1, the robot is placed within a short distance from the box, close to the back wall. In order to move the box as far as possible from its initial position, the only viable pushing direction is towards the top of the area. The two walls are rough. The robot may use a wall as support while pushing the box, but then has to deal with the roughness that may cause the box to get stuck.

The Controller

The neural controller we use has 14 inputs. Eight inputs are delivered by the proximity sensors. The other six are defined as the differences between the values of neighbouring proximity sensors (leaving out the differences between sensors 5 and 6 and between sensors 0 and 7, because these pairs are too widely separated). We call these (virtual) sensors “edge detectors,” because they deliver large inputs when an edge (i.e., a spatial discontinuity in proximity values) is detected. In a mathematical sense, the virtual sensors are redundant. However, in our earlier

Figure 3: Exploiting the mirror symmetry of the robot to derive a neural controller for one wheel from the neural controller for the other wheel. The left network drives the left motor, the right network the right motor. The network inputs are proximity values derived from the Khepera robot shown in Figure 2. The neural networks are equal, but the inputs have been moved around and the signs of the edge-detecting inputs have been switched.



studies we found them to be beneficial to the evolution process (Sprinkhuizen-Kuyper, 2001). The use of edge-detecting inputs is inspired by biological visual systems which are more sensitive to differences than to the absolute level of light intensity. For instance, in the human visual system, edge-detecting neurons are omni-present (Cornsweet, 1970). Detecting edges is an important prerequisite for visual-guided behaviour, and allows the controller to distinguish the box from the walls.

Since the robot has two wheels, either a neural network with two outputs is needed, or two separate neural networks are needed. Because of the symmetric placement of the sensors around the robot (see Figure 2), a mirrored copy of a neural network that drives one of the wheels can be used to drive the other wheel. The mirrored copy of the network requires exchanged inputs for the proximity and virtual sensors. In addition, the signals delivered by the virtual edge-detecting sensors have to be negated (see Figure 3). The use of two (almost) identical networks reduces the number of free parameters considerably which makes the search for a solution easier. The output of the neural networks is mapped onto a legal interval for the motor values by using a sigmoid transfer function.

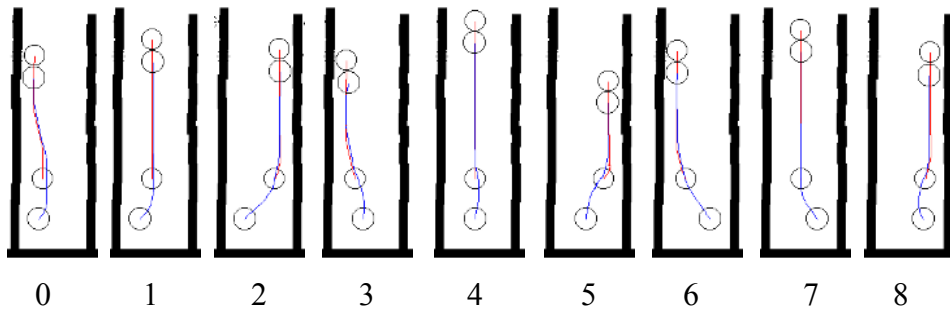
The neural activation functions employed in our neural controller are defined as linear functions. Although the use of linear transfer functions in a multi-layer network does not make much sense from a mathematical viewpoint (a multi-layer network of linear layers can always be reduced to a single linear-layer network), preliminary results revealed only small differences in performance with the (traditional) non-linear transfer functions (Sprinkhuizen-Kuyper, 2001). It turns out that the extra layer may help the evolution process by keeping the connection weights relatively small.

The Evolutionary Algorithm

The evolutionary algorithm used in our experiments works on a population of about 100 individuals. Tournament selection with size 3 is used to get parents for the genetic operators. Newly generated individuals are inserted back in the original population, using a crowding scheme with a factor of 3. We used elitism to prevent loss of the best solutions, and the evolution process continues until no significant increase in fitness is visible any more. Usually this takes 25 to 35 generations.

The chromosome representing a neural network consists of an array of “connection genes.” Each connection gene represents a single possible connection of the network and is defined as a pair of one bit and one real number. The bit represents the presence or absence of a connection and the real value specifies the weight of the connection. During absence, the weight value is maintained in the chromosome which facilitates the evolution process by functioning as a “memory” for removed weight values.

Figure 4: The nine different starting positions used in the experiments. The lines and ending positions illustrate typical box-pushing behaviour.



Selection of Task Instances

For many tasks, the detailed characteristics of the environment are unknown in advance. The evolution of a controller for such a task needs to take the unknown environment into account. By randomly generating task instances (environments) for each controller to be tested, it is ensured that most situations are tested at some point. However, with randomly generated instances evolutionary selection tends to favour lucky controllers over good controllers. Therefore, a better approach is to pre-select a number of specific task instances for the controller to work on, and to define the fitness as a function on the results achieved on those task instances (e.g., the mean fitness). Of course, care should be taken that these task instances form a good sample from the distribution of all instances (i.e., they cover all relevant aspects of the task as a whole).

Based on these considerations we determine the fitness on a pre-selected number of relevant task instances. For our experiments, we use nine different starting positions for the robot and the box as input for the fitness determination (see Figure 4). Defining the origin as the upper left corner (in the world as shown in Figure 1), the x and y coordinates range from 0 to 1000. The three coordinates of the starting positions of the robot are defined as (470,900), (500,900) and (540,900) and those of the box are (480,800), (500,800) and (545,800). Combining starting positions of the robot and box yields nine distinct starting positions. The robot is allowed to push the box as far as possible for 100 time steps. To average out the noise inherent in the simulator, we calculated the fitness by taking the mean of 100 trials for each of the nine starting positions.

Experiments

The experiments performed on the box-pushing task focus on the three questions stated previously, i.e., (1) What is a suitable fitness function for the evolutionary algorithm? (2) What is the best neural network topology to solve the

box-pushing task? (3) What is the best choice for the parameters of the evolutionary algorithm? Below, we discuss the experiments and their results for each of these questions.

The Fitness Function

The fitness of an individual solution can be assessed from different viewpoints. For the first series of experiments, we varied the fitness measure along two dimensions:

- *Global vs. local*. A global fitness measure assesses an individual based on the difference between begin and end state of both the individual and its environment. A local fitness measure, instead, follows the behaviour of the individual at every time step in the simulation.
- *Internal vs. external*. The internal fitness takes the agent's perspective, i.e., it only uses information delivered by the sensors of the robot. In contrast, an external fitness measure takes a "bird's eye" perspective, i.e., it measures the positions of the robot and the box in environmental coordinates.

Combining the global versus local and the internal versus external dimensions, we arrive at the following four different combinations for defining the fitness function:

1. *Global external* (GE). The fitness is defined as the distance between the starting and end position of the box minus half the distance between the robot and the box at the end position.
2. *Local external* (LE). The local fitness (determined at each time step) is defined as the sum of the distance change of the box minus half the change in distance between the robot and the box. The overall local external fitness value is defined as the sum of the local fitness values at each time step.
3. *Global internal* (GI). To allow the robot to derive a fitness measure, it needs some landmark. We added a collection of lights at the end of the desired pushing route. The light intensity sensed by the robot is inversely proportional to the distance to the goal. At the end of the run, the fitness is calculated as the sum of two normalised sums: the values delivered by the two frontal distance sensors (a high value means an object is close, so if the front sensors have a high value, the robot is pushing against something) and the values delivered by the four front light sensors.
4. *Local internal* (LI). For the local internal fitness function, we employ a variation of the function used by Lee et al. (1997). At each time step a local result is calculated by adding the normalised sum of the two frontal distance sensors to half the normalised average of the motor values minus one-third of the normalised rotation component of the motor values. As a result, pushing

Table 1: Cross-comparison of the best controller evolved with each of the four fitness functions tested with each of the four fitness functions. Underlined: comparison with the same measure as evolved with. Bold: best score.

Evolved controller	Cross-comparison			
	GE	LE	GI	LI
GE	<u>286.7</u>	346.6	1.75	132.7
LE	278.8	<u>341.0</u>	1.70	132.3
GI	245.3	304.4	<u>1.61</u>	125.2
LI	168.3	238.9	1.34	<u>139.5</u>

against something and forward motion add to the fitness, whereas turning behaviour is costly in terms of fitness.

We tested the four fitness functions on a feedforward neural controller without hidden nodes. We used a straightforward configuration for the evolutionary algorithm, using only a uniform crossover and a simple mutation operator, without the option of changing the architecture by removing connections. We generated controllers with each of the fitness functions, and then cross-compared the resulting controllers with the other fitness functions. The cross-comparison results of the best controllers are given in Table 1. We found that controllers evolved from the global external viewpoint, evaluated with the other fitness functions, performed significantly better than the controllers evolved with those other fitness functions. This is a fortunate result since evolution runs using the global external viewpoint required the least number of computations, and the global external fitness measure is easily implemented.

The other fitness measures also did quite well in cross-comparison (though they did not yield better results than achieved with the global external measure), except for the local internal fitness measure (see Table 1). The local internal fitness measure was copied from Lee et al. (1997), and works well in the absence of walls, but is of limited use for our experimental set-up: pushing full-speed against a wall results in an undesirable high evaluation score with this fitness measure.

The success of the global external fitness measure is not a surprising result. An external fitness measure can use more objective information than an internal fitness measure and is therefore often more reliable and easier to implement (for instance, in our box-pushing experiments for the internal fitness functions, we needed to add an extra landmark to the world). As far as local fitness measures are concerned, having established the “best strategy” to deal with a task, the fitness of a controller can be determined at each of the constituent steps. However, in practice the best strategy is not known if it were, it could be programmed directly and learning would not be necessary. Using a global fitness measure which judges fitness by the

performance of a controller on the task as a whole leaves the derivation of the strategy to the learning algorithm, instead of making possibly incorrect presumptions about it when implementing a local fitness measure.

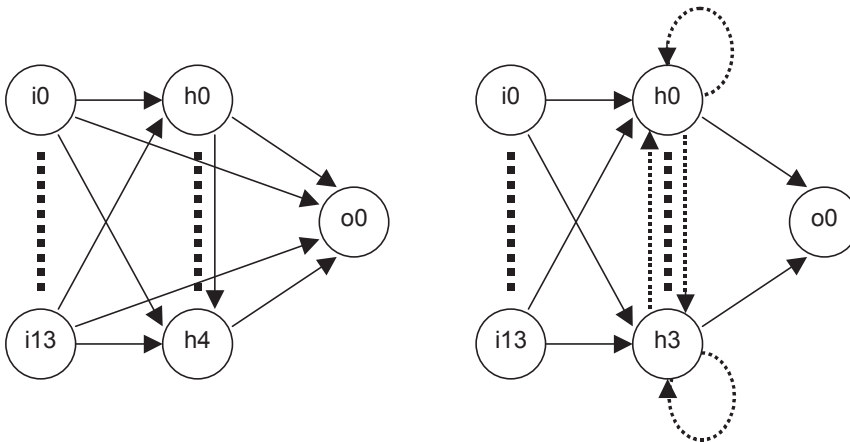
We therefore conclude that a global external fitness measure is best suitable for evolving box-pushing behaviour between two walls from scratch, not only from an experimental but also from a theoretical viewpoint. For the remainder of our experiments, we used the global external fitness measure.

Neural Controller Configuration

To determine the best neural controller topology, we compared two neural networks:

- A *feedforward network with five hidden nodes* (Figure 5, left). This feedforward network is the most general feedforward configuration (with five hidden nodes) possible. It is more general than the more common layered feedforward network, since every connection is allowed as long as a feedforward flow of data through the network is possible. For that purpose, the hidden nodes in the network are ordered and connections between hidden nodes are restricted to run from “lower” nodes towards “higher” nodes. Also, the input nodes may be directly connected to the output nodes.
- A *layered recurrent network with four hidden nodes* (Figure 5, right). This network is less general than a completely recurrent network, because only

Figure 5: Illustration of a feedforward network (left) and a recurrent network (right) as used in the experiments. The hidden nodes in the feedforward network are ordered. Connections between hidden nodes in this network are restricted in the sense that “lower” nodes are only allowed to connect to “higher” nodes. Recurrent connections are represented by dotted arrows.

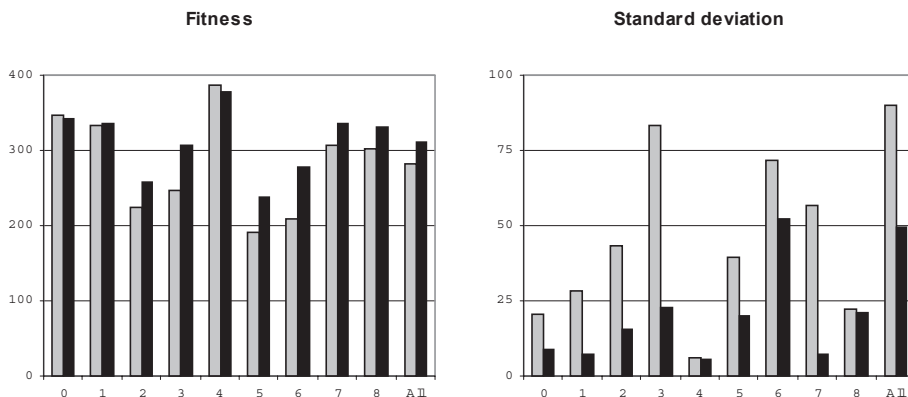


recurrent connections within a layer are allowed. This was done to ease the implementation of recurrent inputs. The recurrent connections are used to input the node values from the previous cycle through the network into the new cycle. As such, they function as a memory for previous node values. The number of connections in the recurrent network with four hidden nodes is slightly higher than in the feedforward network with five hidden nodes.

We used the same evolutionary algorithm to develop these neural controllers as we did in the previous experiment. To increase the evolution speed, only the fitness results of the candidates for the best individual were averaged over 100 trials. The fitness values of the other individuals were obtained by averaging over 10 trials. Subsequently, we experimented with different genetic operators. These experiments did not yield results that differed significantly from those reported below. However, since the new genetic operators added the ability to vary the size of the hidden layer, these experiments did reveal that for our problem a hidden layer consisting of three nodes yields optimal results.

As can be concluded from Figure 6, the experiments showed that recurrent controllers tend to perform better (have a higher fitness) and more reliable (have a lower standard deviation) than feedforward controllers. We also discovered the

Figure 6: Mean values of the results of the best feedforward (grey bars) and recurrent (black bars) controllers, averaged over 7 evolution runs. The left chart gives the fitness values, the right chart the standard deviation. On the horizontal axis the different starting positions are presented, with the mean values over all starting position presented as “all.” The overall values are as follows: for the feedforward controller, the fitness is 282.9 with a standard deviation of 90.2; for the recurrent controller, the fitness is 311.2 with a standard deviation of 49.6.



probable reason for that. When observing how the wheels of a feedforward controlled robot move, we noticed that they almost exclusively take on one of the two extreme control values: -10 and $+10$, which initiate the behaviours “very fast backwards” and “very fast forwards.” This behaviour works well when pushing the box without hindrance, but makes it difficult for the robot to manoeuvre itself out of a problematic position. This causes the robot controlled by a feedforward network to get stuck regularly. A recurrent controller, on the other hand, shows more subtle behaviour, and even though it sporadically gets stuck, it does so less often than the feedforward controller. This explains both the higher fitness and the lower standard deviation of recurrent controllers.

Some parallels can be drawn between global characteristics of our successful controller and those of the biological visual systems. The processing of information in visual systems is mainly bi-directional. At the level of neural networks, reciprocal connections are the rule rather than the exception (Shepherd, 1990). Even at the level of brain systems, the communication among systems is bi-directional. In agreement with these biological characteristics, our most successful controllers are recurrent controllers with linear feedback. In biological systems, linear feedback is characteristic of temporal filters, short-term memory and noise suppression mechanisms.

The Evolutionary Algorithm

Our third series of experiments aims at enhancing the evolutionary algorithm in order to significantly increase the final fitness values reached.

We previously sketched the basic evolutionary algorithm. The algorithm employs the following six genetic operators, one of which was randomly selected at each evolution step:

1. Uniform crossover
2. Biased weight mutation (Montana & Davis, 1989) with a probability of 10% to change each weight, in a range of $[-0.3, +0.3]$
3. Biased nodes mutation (Montana & Davis, 1989), changing just one node within the same range as the biased weight mutation
4. Nodes crossover (Montana & Davis, 1989) picking half the nodes of each parent
5. Node existence mutation (Spronck & Kerckhoffs, 1997), with a probability of 95% to remove a node and a probability of 5% to completely activate a node
6. Connectivity mutation (Spronck & Kerckhoffs, 1997), with each connection having a 10% probability to flip.

For the crossover operators, the best of the children was added to the population, and the other one removed. Thierens et al.’s (1993) treatment of

competing conventions was used to support the crossover operators. Our earlier experiments showed that the kind of genetic operators used does not really influence the final results (even the simplest operators do not yield significantly different results); but the six genetic operators listed above provide a reasonable evolution rate and the ability to reduce the size of the neural controllers.

We found that changes to the genetic operators and changes to minor parameters of the algorithm (such as population size, the selection mechanism and the maximum number of generations) would not bring significant changes to the best fitness values reached. However, we also found that the final solution found would often have more problems with the “harder” task instances (starting positions) than with the easier task instances. This is not surprising, because standard evolutionary-learning algorithms tend to favour a larger number of solutions to easy problem instances over a smaller number of solutions to hard ones (Spronck et al., 2001).

In multi-objective evolutionary learning (Fonseca & Fleming, 1995), a solution with multiple objectives has to be found. Our task instances (the nine starting positions) correspond to the separate objectives in multi-objective learning. The main difference between these two forms of learning is that for multi-objective learning, the separate objectives are usually very different, whereas in task learning the objective is the same for each of the task instances. Therefore, with task learning problems, it is more than likely that a solution to one of the task instances also works well on some other task instances. This is especially true for the solutions to hard instances because they often incorporate solutions to the easier instances.

A technique sometimes used in multi-objective evolutionary learning is “doping.” Here solutions for some of the objectives, for instance generated using well-known heuristics, are inserted into the initial population (see, for instance, Matthews, 2000). This helps the algorithm evolve a solution for all objectives which incorporates the pre-supplied solutions to single objectives. Note that “doping” of initial populations does not work when applying evolutionary algorithms to a single task because the doped solution will not be further improved in terms of fitness.

Since we suspected that solutions to the harder task instances in our box-pushing problem would probably encompass characteristics required to solve the easier tasks, we proposed that doping our initial populations with a solution for one of the harder task instances could lead to higher final fitness rates. To test this, two steps had to be undertaken: the hardest task instances needed to be identified, and a good solution for one of these task instances had to be generated and inserted in the initial population for our regular evolutionary algorithm.

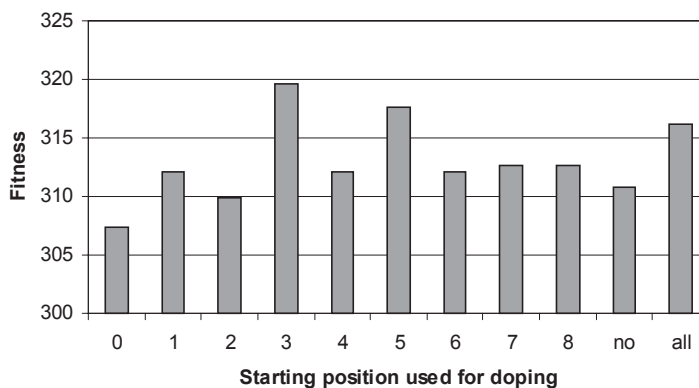
Good solutions for each of the starting positions were evolved using the standard evolutionary algorithm. We allowed 75 (instead of 35) generations to be created and (obviously) the fitness was defined for a single starting position only. We defined the hardest starting positions as those that took the longest to converge

and even then could not reach a very high fitness value. Starting positions 3 and 5 (see Figure 4) proved to be the most difficult in this respect. It should be noted that one would expect starting positions 2 and 6 to be more difficult since for these positions the separation between the robot and the box is the largest, but positions 3 and 5 appear to be more difficult due to the roughness of the walls and the angle under which the robot first tends to push the box against one of the walls.

We then ran a number of regular experiments (with 35 generations), whereby for each initial population one of the winning individuals on an isolated starting position was inserted. For each of these, we ran five or six experiments. We also ran a series of experiments where we doped the initial population with a good solution for each of the nine starting positions.

The results for the experiments with doping with a single starting position, without doping, and with doping with all starting positions, are graphically compared in Figure 7. The average fitness for all of the experiments with the doped populations settles around the same fitness value as the experiments without doping, namely 311, except for those doped with a winning individual for one of the two hardest starting positions (3 and 5). Doping with starting position 3 yielded an average fitness of 320, and doping with starting position 5, an average fitness of 318. Both fitness values obtained are significant improvements over our previous results (preliminary experiments showed that the standard error of the mean (Cohen, 1995) for evolved controllers with 100 experiments is about 1.3, so the results have an accuracy of about 2.5 fitness points with 95% reliability). Furthermore, the overall best result over all experiments was achieved with doping of starting position 5, namely a fitness of 323.

Figure 7: Fitness values of experiments with doping with a single starting position (“0” to “8”), without doping (“no”) and with doping with all starting positions (“all”).



Doping with all nine starting positions led to an average fitness of 316. This is not a significant difference with the results achieved for doping with starting positions 3 or 5 (especially not since for the experiments with doping of all nine starting positions also, an individual with a fitness of 323 was found).

To test whether doping with a hard starting position always manages to achieve these high fitness results, we generated four different versions of a winning controller for starting position 5. All of these had about equal, high fitness ratings on the isolated starting position. Four repetitions of experiments with doping with each of these individuals were executed. In three experiments the final result was equal to what we found in our first test with doping with a solution to starting position 5. The fourth, however, achieved only the same average results we achieved in our experiments without doping.

These results show that for the box-pushing task, doping of an initial population, with a winning individual that has been evolved on one of the hard task instances, generally leads to better overall results than an evolution without doping. However, the results of the experiments with alternative versions of a winning individual for starting position 5 show that a fitness improvement over experiments without doping is not guaranteed, since it is possible to dope with an “unlucky” individual.

It is important to remark that our experiment differs from standard machine learning experiments in the sense that our aim is to determine the effect of doping rather than to determine generalization performance. In the latter case, inclusion of the starting position on which the algorithm is trained in the test data would give a positively biased and wrong indication of generalization performance. Our results should therefore be interpreted in terms of overall task performance due to doping, rather than in terms of generalization performance. Incidentally, we know that the generalization effect *is* present because doping with a solution for one of the hard starting positions not only improves the results of the final controller on that particular starting position, but also on most of the other starting positions, especially the harder ones. We confirmed this with experiments in a more deterministic environment.

Conclusion

Our experiments aimed at answering the following question. How can a neural network learn to control the box-pushing task using evolutionary-computation techniques? We investigated three characteristics of relevance to evolving robot controllers, i.e., the fitness measure, the neural network topology and the parameters of the evolutionary algorithm. On these three characteristics and their associated questions, we conclude the following.

1. A global external fitness measure gives the best results and is easiest to implement.
2. A recurrent neural controller, while more difficult to implement and evolve, gives significantly better results than a feedforward controller.
3. While neither changes to most parameters of the evolutionary algorithm nor the selection of genetic operators significantly influence the final results of the evolution process, doping the initial population with an individual that solves one of the harder task instances improves upon the final fitness reached.

Clearly, future work has to establish to what extent these conclusions generalize to other (more complex) tasks. In preliminary studies on an entirely different and more complex foraging task involving the capture of food and avoidance of poison, we obtained similar results. Therefore, we are confident that our results generalize beyond the box-pushing task. The simplicity of the box-pushing task allowed us to analyse the results of our experiments and draw conclusions on the fitness function, the neural network topology and parameters of the evolutionary algorithm. We therefore conclude that the detailed analysis of evolving a controller on a simple task forms a good starting point for more complex tasks.

References

- Arkin, R. (1998). *Behavior-Based Robotics*. Cambridge: MIT-press.
- Asada, M. & Kitano, H. (1999). The Robocup challenge. *Robotics and Autonomous Systems*, 29(1):3-12.
- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press.
- Brooks, R.A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 14-23.
- Cohen, P.R. (1995). *Empirical Methods for Artificial Intelligence*. MIT Press.
- Cornsweet, T.N. (1970). *Visual Perception*. Academic Press.
- Fonseca, C.M. & Fleming, P.J. (1995). An overview of evolutionary algorithms in multi-objective optimization. *Evolutionary Computation*, 3(1):1-16.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley Publishing Company.
- Lee, W-P., Hallam, J. & Lund, H.H. (1997). Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. *Proceedings of IEEE 4th International Conference on Evolutionary Computation*, IEEE Press.

- Matthews, K.B., Craw, S., Elder, S., Sibbald, A.R. & MacKenzie, I. (2000). Applying genetic algorithms to multi-objective land use planning. *Proceedings of the Genetics and Evolutionary Computation Conference (GECCO 2000)*, 613-620, Morgan Kaufman.
- Mondada, F., Franzi, E. & Jenne, P. (1993). Mobile robot miniaturisation: A tool for investigating in control algorithms. In Yoshikawa, T. & Miyazaki, F. (Eds.) *Proceedings of the Third International Symposium on Experimental Robotics*, 501-513. Berlin: Springer Verlag.
- Montana, D. & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 762-767, CA: Morgan Kaufman.
- Shepherd, G.M. (1990). *The Synaptic Organization of the Brain (Third Edition)*. Oxford: Oxford University Press.
- Sprinkhuizen-Kuyper, I.G. (2001). *Artificial Evolution of Box-pushing Behaviour*, Report CS 01-02, Universiteit Maastricht, Faculty of General Sciences, IKAT/Department of Computer Science, Maastricht, The Netherlands.
- Spronck, P. & Kerckhoffs, E. (1997). Using genetic algorithms to design neural reinforcement controllers for simulated plants. In Kaylan, A. & Lehmann, A. *Proceedings of the 11th European Simulation Conference*, 292-299.
- Spronck, P., Sprinkhuizen-Kuyper, I.G. & Postma, E.O. (2001). Infused evolutionary learning. In Hoste, V. & de Pauw, G. *Proceedings of the Eleventh Belgian-Dutch Conference on Machine Learning*, 61-68. University of Antwerp.
- Thierens, D., Suykens, J., Vandewalle, J. & De Moor, B. (1993). Genetic weight optimization of a feedforward neural network controller. In Albrechts, R.F., Reeves, C.R. & Steel, N.C. *Artificial Neural Nets and Genetic Algorithms*. 658-663, New York: Springer-Verlag.
- Whitley, D., Dominic, S., Das, R. & Anderson, C. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, (13), 103-128. Kluwer Academy Publishers.
- Yao, X. (1995). Evolutionary artificial neural networks. In Kent et al. (Ed) *Encyclopedia of Computer Science and Technology*. (33) 137-170. New York: Marcel Dekker Inc.

Software Availability

The Khepera simulator is available from <http://diwww.epfl.ch/lami/team/michel/khep-sim/>.

The program “Elegance” is available from <http://www.cs.unimaas.nl/p.spronck/>.

Chapter VII

Computational Intelligence for Modelling and Control of Multi-Robot Systems

M. Mohammadian
University of Canberra, Australia

ABSTRACT

With increased application of fuzzy logic in complex control systems, there is a need for a structured methodological approach in the development of fuzzy logic systems. Current fuzzy logic systems are developed based on individualistic bases and cannot face the challenge of interacting with other (fuzzy) systems in a dynamic environment. In this chapter a method for development of fuzzy systems that can interact with other (fuzzy) systems is proposed. Specifically a method for designing hierarchical self-learning fuzzy logic control systems based on the integration of genetic algorithms and fuzzy logic to provide an integrated knowledge base for intelligent control of mobile robots for collision-avoidance in a common workspace. The robots are considered as point masses moving in a common work space. Genetic algorithms are employed as an adaptive method for learning the fuzzy rules of the control systems as well as learning, the mapping and interaction between fuzzy knowledge bases of different fuzzy logic systems.

INTRODUCTION

Fuzzy logic systems are increasingly used in control applications. Their ability to cope with uncertainty inherent in complex systems makes them an attractive

method for solving complex, uncertain and dynamic systems. Current fuzzy logic systems are developed based on the individualistic systems. These systems are unable to face the challenge of interaction that might be necessary between different fuzzy logic systems solving a complex problem. There is a need for a structured approach to design fuzzy logic systems for controlling complex systems consisting of multiple fuzzy logic systems and fuzzy knowledge bases as is the case in a hierarchical fuzzy logic system.

In general hierarchical fuzzy logic systems consist of several fuzzy logic systems, each performing a specific task which are combined to form a system to solve a complex task. These controllers interact with each other to solve the problem at hand. The output of each fuzzy logic controller in a hierarchical fuzzy logic system has an effect on other fuzzy logic systems and consequently to the final output of the system.

The division of individual fuzzy systems required to solve complex problems demands that those problems be decomposed and distributed among different fuzzy logic systems. One of the main limitations on the application of hierarchical fuzzy logic systems in complex problem-solving domains is the lack of methods to structure the development of the hierarchical fuzzy logic systems. This is an important issue when considering the robustness and efficiency of the system. A well structured hierarchical fuzzy logic system can perform its tasks more efficiently.

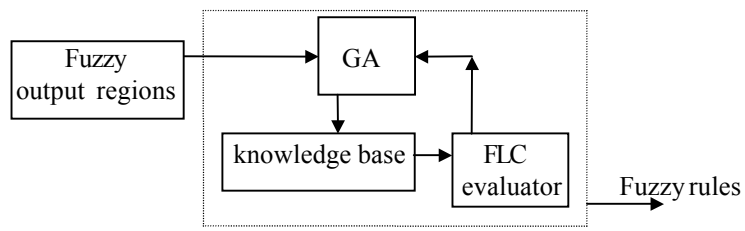
The design of the fuzzy knowledge base of complex fuzzy logic systems are based upon human experience and the operator's knowledge of the system to be controlled (Lee, 1990). The fuzzy rules are formulated by a trial and error method which is not only time consuming but also does not guarantee 'optimal' fuzzy rules for the system. Incorporating genetic algorithms into the design of a fuzzy logic system ensures automatic generation of fuzzy rules for a fuzzy logic system.

This chapter is organised as follows: in the next section the learning of fuzzy rules of fuzzy logic systems using genetic algorithms are described. The application of this learning method to control a simulated multi-robot system using hierarchical fuzzy logic systems is considered and simulation results are presented. Conclusions are then drawn and further research directions are given.

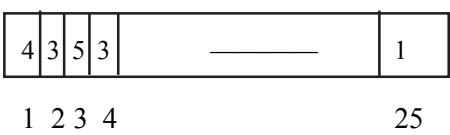
LEARNING OF FUZZY LOGIC SYSTEM USING GENETIC ALGORITHMS

Earlier in the following papers (Mohammadian, 1994; Mohammadian, 1996; Stonier, 1998) an integrated architecture consisting of genetic algorithms and fuzzy logic for automatic rule generation of fuzzy logic systems was proposed. A block diagram of the fuzzy rule generation architecture is shown in Figure 1.

Figure 1: Fuzzy-GA rule generator architecture

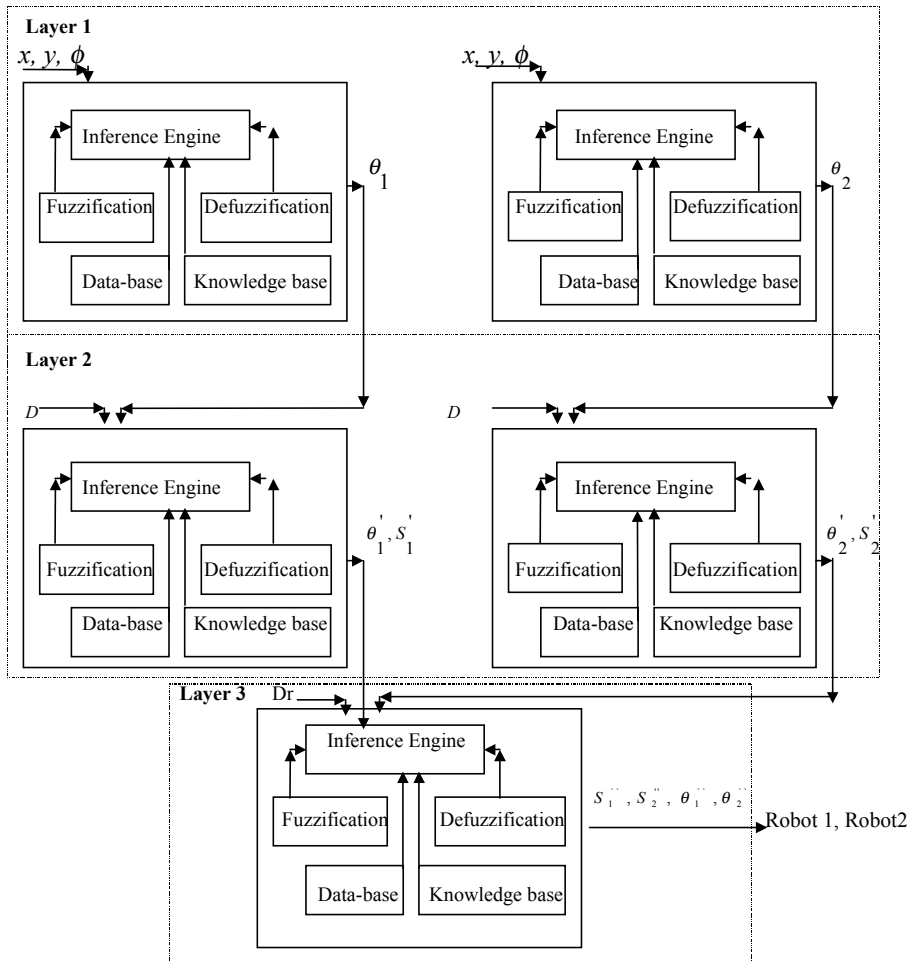


Let us consider a fuzzy logic controller with two inputs (x and y) and a single output (z). As a first step to generating the fuzzy rules, the domain intervals of the input and output variables are divided into different regions, called fuzzy sets. The number of fuzzy sets is application dependent. Assume that x , y and z are all divided into five fuzzy regions each, with x and y denoted by the linguistic terms **VL**, **LO**, **MD**, **HI**, **VH** and z denoted by the linguistic terms **VS**, **SM**, **MD**, **HI**, **VH**. A fuzzy membership function is assigned to each fuzzy set. Since x and y are divided into five fuzzy sets each, a maximum of twenty five fuzzy rules can be written for the fuzzy logic system. The consequent for each fuzzy rule is determined using a genetic algorithm. In order to do so, the output fuzzy sets need to be encoded. It is not necessary to encode the input fuzzy sets because they are static. The fuzzy rules relating the input variables (x and y) to the output variable (z) have twenty five possible combinations. The consequent of each fuzzy rule can be any one of the five output fuzzy sets. The output fuzzy sets are encoded by assigning a number for each fuzzy set for example, 1 = **VS** (Very Small), 2 = **SM** (Small), 3 = **MD** (Medium), 4 = **HI** (High) and 5 = **VH** (Very High). The genetic algorithm randomly encodes each output fuzzy set into a number ranging from 1 to 5 for all possible combinations of the input fuzzy variables. An individual string (chromosome) can then be represented in the following way:



Each string is a member of a population, and a population of size n has n number of individual strings randomly encoded by genetic algorithm. Each individual string is then decoded into the output linguistic terms. The set of fuzzy rules thus developed is evaluated by the fuzzy logic controller based upon a fitness value which is specific to the system. The fitness value is application dependent. At the end of each generation, copies of the best performing string from the parent generation are included in the next generation to ensure that the best performing strings are not lost. The genetic algorithm then performs the process of selection, crossover and

Figure 3: A three-layer hierarchical fuzzy logic system for controlling the multi-robot system



Hierarchical Fuzzy Logic Systems for Multi-Robot Control

In this section we use the method proposed in the above section to develop a hierarchical fuzzy logic system to control a simulated multi-robot system. Figure 2 shows the diagram of simulated robots and their targets in the workspace.

A hierarchical fuzzy logic system consisting of three layers is developed to control and guide the robots to their target. In the first layer, a fuzzy logic system is developed for each robot that controls each robot from any position in the workspace to its target. The knowledge on how to control the robot is learned by a self-learning method using genetic algorithms (see Figure 1). In the second layer a new fuzzy logic system is developed for each robot that learns to control the speed

and slow down the robots when arriving to their target. The second layer fuzzy logic controllers work with the first layer fuzzy logic controllers, and the knowledge base of the second layer fuzzy logic controllers are learned upon the first fuzzy logic system's knowledge base. A mapping of the knowledge bases in the first layer to the second layer is learned using genetic algorithms. In this way the system adapts itself to the current knowledge (fuzzy control rules) available and uses this knowledge to improve the performance of the system by learning new concepts. In this case the current knowledge is how to control each robot to its target, and the new concepts are the knowledge learned about how to control the speed of robots and slow them down at their targets. For the third layer a new fuzzy logic system is developed to avoid collision of the robots working in common workspace. Here a mapping of the knowledge base of the second layer fuzzy logic controllers to the third layer fuzzy logic controllers is learned using genetic algorithms. Figure 3 shows the structure of the Hierarchical Fuzzy Logic Control system (HFLCs).

The first genetic algorithm learns the fuzzy knowledge base to control each robot to its target using constant speed. Below a description of the fuzzy knowledge base for layer one is given. For each robot there are three inputs x , y , ϕ and single output θ . Here x , y are the cartesian coordinates in the plane, and ϕ is the heading angle relative to the x-axis, and θ is the control steering angle also measured relative to the x-axis (Mohammadian, 1994; Stonier, 1998).

We divide the domain regions for x , y , and θ into 5, 5, 7 and 7 regions (fuzzy sets) respectively and assign appropriate linguistic variables. For each region (fuzzy

Figure 4: Fuzzy regions and the corresponding membership functions of x , y , ϕ and θ

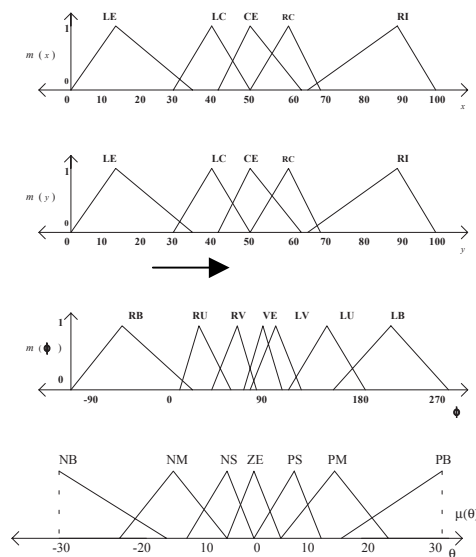
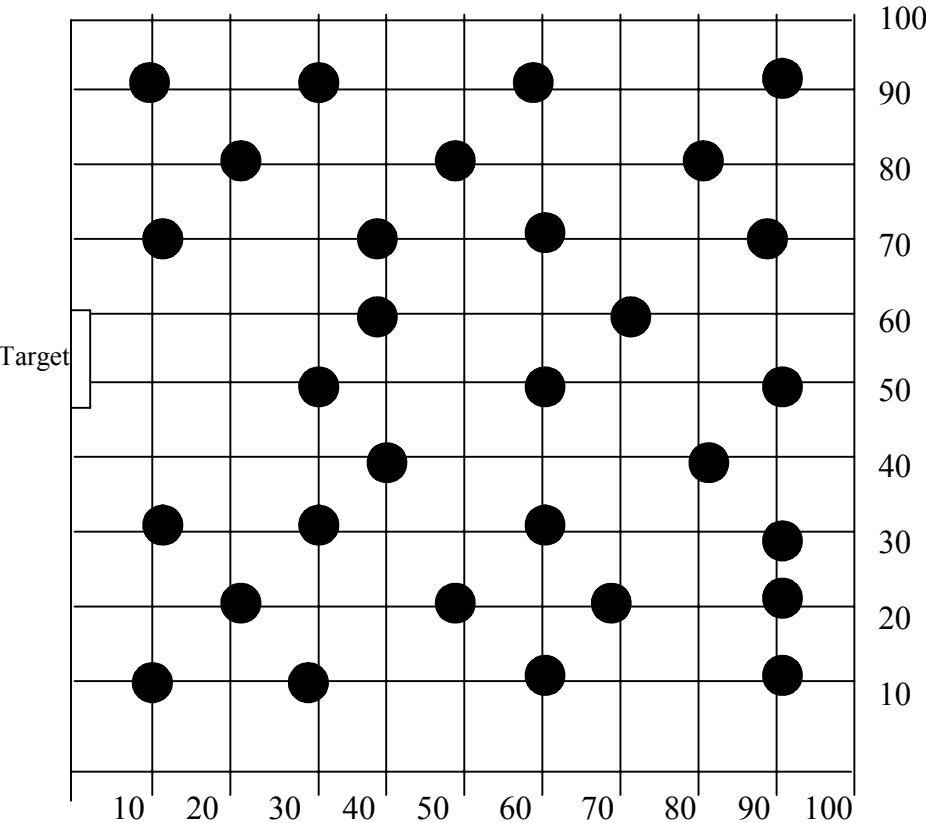


Figure 5: Spread of the initial configurations in the workspace for robot₁

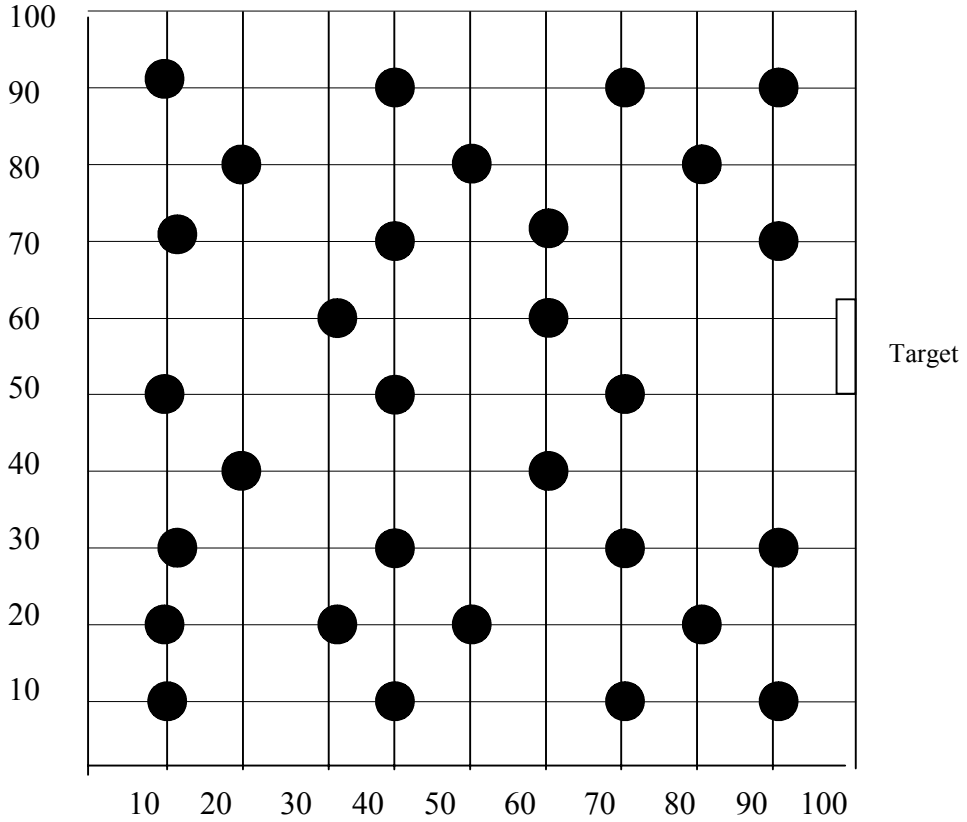


set), a fuzzy membership function is assigned; the shape is assumed to be triangular, see (Figure 4).

A grid of 180 configurations is chosen on the plane and associated with each point there are 6 chosen heading directions. For this analysis the heading directions are -50° , 45° , 90° , 135° and 240° . Figure 5 shows the spread of initial configurations in the workspace and the target.

The robot is started random at an initial configuration chosen from the 180 available initial configurations. Genetic algorithm was used to determine a set of fuzzy rules that will best determine the steering angle to control the robot to the desired target (with final heading angle 180° for the first robot and 0° for the second robot) in the first layer, (see Figure 2). Figure 6 shows the spread of initial configurations in the workspace and the target.

Figure 6: Spread of the initial configurations in the workspace for robot₂



The two individual fuzzy rule bases, one for each robot, constitute the full fuzzy knowledge base in layer 1. In all there are 175 rules learned for the control of each robot to its target.

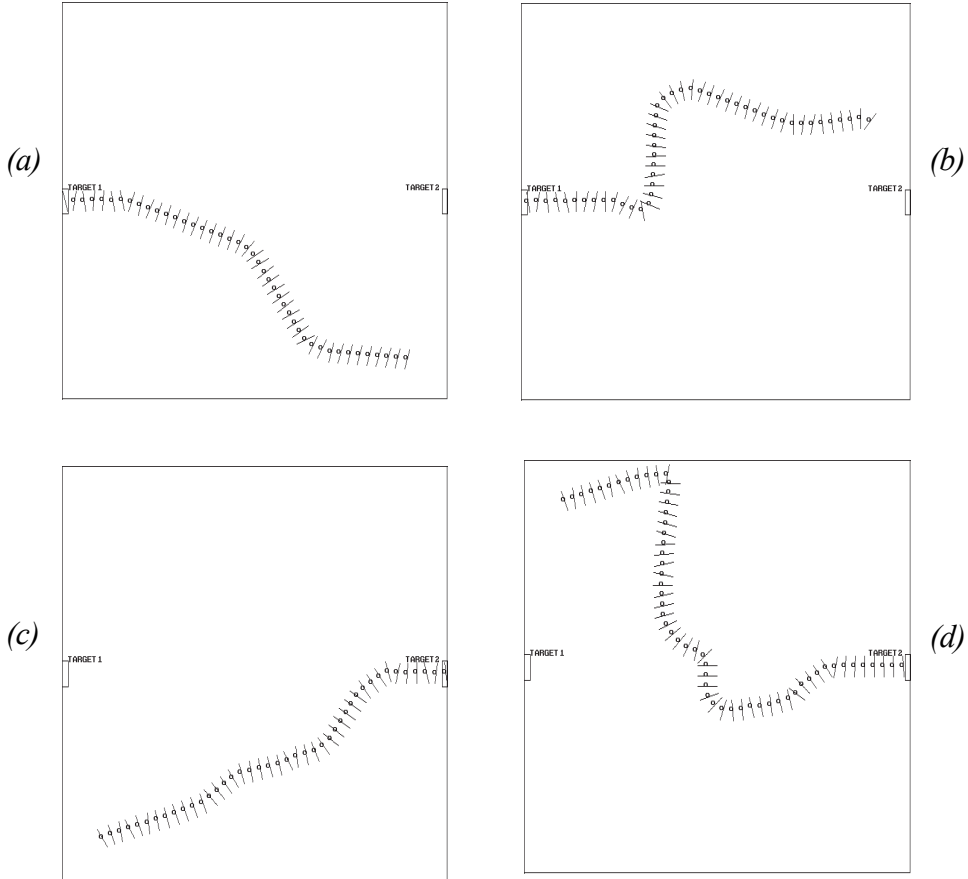
The objective function for the first layer HFLC system is calculated as:

$$Objective = \sqrt{(x_r - x_t)^2 + (y_r - y_t)^2 + (\phi_r - \phi_t)^2}$$

where x_r, y_r, ϕ_r and x_t, y_t, ϕ_t are the robot's coordinates and the heading angle and the target's coordinates respectively. We wish to minimise the *objective* function. An illustration of how the knowledge contained in the final fuzzy knowledge base controls the robots to their targets is shown in Figure 7. Good trajectories were obtained from all initial configurations.

Figure 7: Robot trajectory from initial configurations:

(a) $(90, 10, 135^\circ)$, (b) $(90, 70, 135^\circ)$; and for robot₂ (c) $(90, 10, 135^\circ)$, (d) $(70, 70, 135^\circ)$



We now consider the adaptive learning of the rules in the second layer of the hierarchical fuzzy logic controller using genetic algorithm. Having learned the fundamental knowledge base for steering control of each robot to its target, we would like to control the speed of each robot to its target. The objective here is to develop another fuzzy knowledge base for a fuzzy logic system to determine corrections to steering angles and speed of each robot while approaching its target. There are two inputs which is the distance D between the robot and its target and the current steering angle of each robot and output, $\theta'_i, S'_i, i = 1, 2$, which is the correction to the speed and steering angle of each robot for the next iteration.

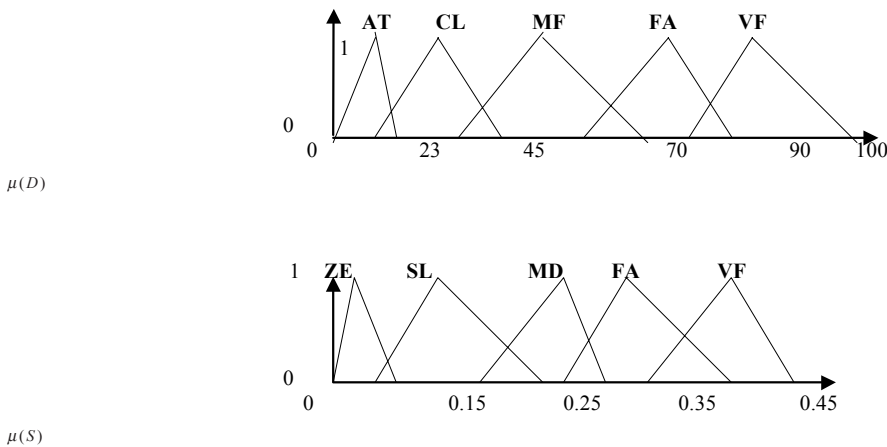
Figure 8 shows the fuzzy sets and fuzzy membership functions of D and S . Intervals of definition D, S, θ and θ' , are each divided into $5 \times 5 \times 7 \times 7$ regions (fuzzy membership of θ and θ' are the same). There are thirty five fuzzy rules for the second layer fuzzy knowledge base. The fuzzy knowledge base can be formed as a 5 by 7 table with cells to hold the two outputs for the corresponding actions that must be taken, given the conditions corresponding to D and θ are satisfied. For example, a fuzzy rule may look like :

If $D = \mathbf{VC}$ and $\theta = \mathbf{VS}$, then $S' = \mathbf{VS}$ also $\theta' = \mathbf{PB}$

The choice of output control signal to be set for each fuzzy rule is made by genetic algorithm. The genetic algorithm then performs a self-directed search, learning fuzzy rules for the second fuzzy knowledge base of the HFLC. The learning is performed quickly and automatically with no need for operational guidance other than the fitness values supplied to it by the HFLC.

Again here a grid of 180 configurations is chosen on the plane, and associated with each point there are 6 chosen heading directions (same as for the first layer). The analysis then proceeds as in the first layer, a genetic algorithm is used to determine a set of fuzzy rules that will best determine the steering angle and speed corrections to drive a robot to its desired target from a randomly chosen initial triplet (x, y, ϕ) with speed equal to zero at the target. The same fitness function that was used in the first layer is used, but a penalty of 1,000 is added if the speed of the robot when it has reached its target is greater than zero. The genetic algorithm is then allowed to evolve through the normal processes of reproduction, crossover and mutation to minimise this fitness. Again an 'elite' option was used in developing a

Figure 8: The fuzzy sets and membership functions of D and S



new population from the old and prescaling used to improve convergence. The best chromosome in the final population defines the fuzzy knowledge base for this, the second layer HFLLC. An illustration of how the knowledge contained in the second layer fuzzy knowledge base is shown in Figure 9 and 10. Next we consider the adaptive learning of the rules in the third layer of the hierarchical fuzzy logic controller using the genetic algorithm.

Having learned the fundamental rule bases for steering control and speed of each robot to its target, the objective here is to develop another fuzzy knowledge base to determine corrections to steering angles θ for each robot to avoid the collision of robots while approaching their target. In this layer there are three inputs

Figure 9: Robot trajectory of robot₁ from initial configurations: (a) (90, 60, 45°) and (b) (90, 10, -50°)

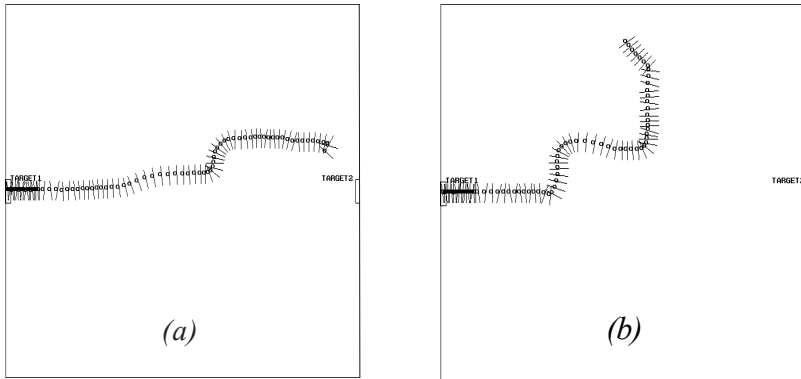


Figure 10 Robot trajectory of robot₂ from initial configurations: (a) (50, 90, -50°), and (b) (60, 90, 90°)

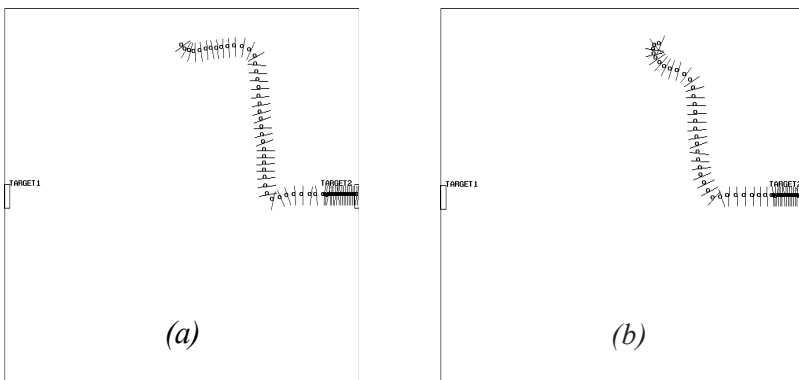
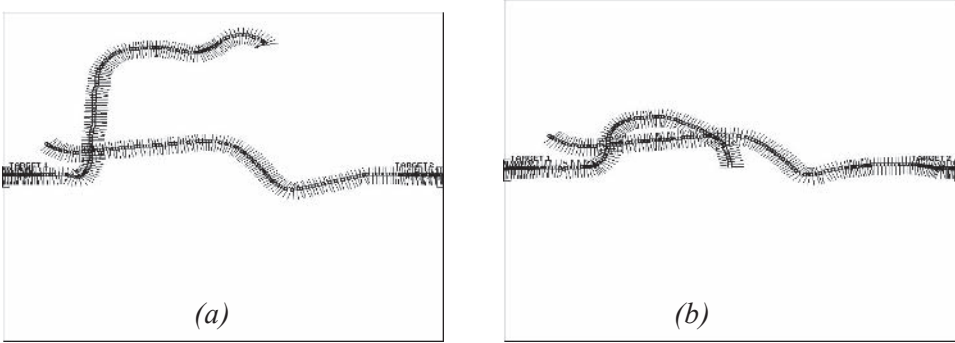


Figure 11: (a) Robot trajectories at initial configuration:
 (a) $robot_1 = (50, 50, 45^\circ)$, $robot_2 = (10, 60, -50^\circ)$ and
 (b) $robot_1 = (60, 50, 90^\circ)$, $robot_2 = (10, 60, -50^\circ)$



Dr , θ_1' and θ_2' , with outputs θ_1'' and θ_2'' , S_1'' and S_2'' for each robot. Here Dr is the calculated physical distance between robots, θ_1'' and θ_2'' is a correction to the steering angle of each robot and S_1'' and S_2'' is the correction to the speed of each robot.

We divided the domain regions for Dr , θ_1' and θ_2' , into 5, 7 and 7 regions respectively and assigned appropriate linguistic variables, as before. Again for each region a fuzzy membership function is assigned, and the shape is assumed to be triangular. We use genetic algorithm to learn the fuzzy knowledge base for the third layer of HFLC. The fitness of each chromosome is calculated as for second layer fuzzy logic system and a penalty (of 1,000) is added if the robots collide. In all simulations the genetic algorithm had a population size of 100 with mutation rate = 0.01 and crossover rate = 0.6. The genetic algorithm was run for 500 generations. Figure 11 shows the robot trajectories using HFLC with three layers. The robots arrive to their target while avoiding collision, and their speed is reduced as they approach their target.

Using a hierarchical fuzzy logic control system and a genetic fuzzy rules generator architecture, quick convergence to a collision free path in learning the fuzzy rules of the third layer was observed. This can be attributed to the fact that the architecture was able to learn and modify rules in the third layer fuzzy rule base using knowledge in the second layer fuzzy rule bases.

CONCLUSION

We proposed using HFCLC for a multi-robot control problem. We tackled the complexity of the multi-robot control problem by dividing the problem into smaller manageable parts. By using HFCLC the number of control laws is reduced. In the first layer of HFCLC, ignoring the possibility of collision, steering angles for the control of each robot to their associated target were determined by genetic algorithms. In the second layer genetic algorithm was used to determine adjustments to steering angle and speed of each robot to control the speed of the robot when arriving to its target. Next another layer is developed to adjust the speed and steering angle of the robots to avoid collision of the robots. If only one fuzzy logic system was used to solve this problem with the inputs x, y, ϕ of each robot and D , each with the same fuzzy sets described in this chapter then there would be 153,125 fuzzy rule needed for its fuzzy knowledge base. Using a HFCLC system we have a total number of 1,645 fuzzy rules for this system. The hierarchical concept learning using the proposed method makes easier the development of fuzzy logic control systems, by encouraging the development of fuzzy logic controllers where the large number of systems parameters inhibits the construction of such controllers.

ACKNOWLEDGMENT

The guidance, support and assistance of R. J. Stonier from Central Queensland University, Australia, to complete this project is greatly appreciated and acknowledged.

REFERENCES

- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley.
- Lee C. C. (1990). Fuzzy logic in control systems : fuzzy logic controller- part 1, *IEEE Transaction on Systems, Man and Cybernetics*, 20(2), 404-418.
- Lee C. C. (1990). Fuzzy logic in control systems: fuzzy logic controller—part 2, *IEEE Transaction on Systems, Man and Cybernetics*, 20(2) 419-435.
- Mohammadian, M., Kingham, M., Hassan, M.. (1996). Adaptive holding policies for IP over ATM networks using fuzzy logic and genetic algorithms, *IEEE International Conference on Communication System*, Westin Stamford, Singapore.
- Mohammadian, M. & Stonier, R.J. (1994). Generating fuzzy rules by genetic algorithms, *Proceedings of 3rd IEEE International Workshop on Robot and Human Communication*, Nagoya, 362-367.
- Raju., G.V.S. & Zhou., J. (1993). Adaptive Hierarchical Fuzzy Controller, *IEEE Transactions on Systems, Man and Cybernetics*, 973-980.

- Stonier, R.J. & Mohammadian, M. (1995). Intelligent hierarchical control for obstacle-avoidance, *Conference on Computational Techniques and Applications: CTAC95*.
- Stonier, R. J. & Mohammadian, M. (1998). Knowledge acquisition for target capture, *Proceedings of the International Conference on Evolutionary Computing ICEC'98*, Anchorage, Alaska, 721-726.
- Wang L. (1997). *A Course in Fuzzy Systems and Control*, Prentice Hall.

Chapter VIII

Integrating Genetic Algorithms and Finite Element Analyses for Structural Inverse Problems

D.C. Panni and A.D. Nurse
Loughborough University, UK

ABSTRACT

A general method for integrating genetic algorithms within a commercially available finite element (FE) package to solve a range of structural inverse problems is presented. The described method exploits a user-programmable interface to control the genetic algorithm from within the FE package. This general approach is presented with specific reference to three illustrative system identification problems. In two of these the aim is to deduce the damaged state of composite structures from a known physical response to a given static loading. In the third the manufactured lay-up of a composite component is designed using the proposed methodology.

INTRODUCTION

Inverse analyses have a variety of applications in structural mechanics in which unknowns in a structure are determined using system identification techniques. These techniques allow the state of the structure to be deduced from the observed

response to given inputs. Depending on the problem to be solved, the unknowns to be determined may be the material properties, applied loads, boundary conditions or even the geometry of the specimen.

In general, inverse system identification techniques involve updating an analytical model representing the structure, where the difference between some measure of analytical response and equivalent experimental response is minimised. In this sense, the inverse problem can also be considered an optimisation problem. Central to the analysis is the appropriate selection of an analytical model that can accurately predict the response of the structure, and an efficient and robust optimisation algorithm for updating the model. Both of these components can be programmed for specific problems, however the programming of an analytical model is largely problem dependent and can be cumbersome. This is particularly true of structures in which the geometry and to a lesser extent the material properties are complex. Therefore it is considered, beneficial to develop a robust tool that can be readily applied to a wide range of structural inverse problems without being concerned with the complexity of geometry and/or material properties.

The method described here exploits the versatility of the *LUSAS* finite element package (distributed by FEA Ltd., v13.3, 2001, www.lusas.com) by integrating it as an object within a genetic algorithm (GA). The principal advantage of this approach is that the broad functionality of the finite element application can be used to model many structural scenarios, without needing to know the exact form of the analytical model. It is sufficient to enter the geometry, the loading and the boundary conditions without explicitly stating the form of the analytical model. This is handled inside the FE code and effectively hidden to the analyst. It is expected that this work will provide the basis of future automated and robust inverse analysis in a wide range of applications.

The described method is applicable to many structural problems in which the state of the structure is unknown. GAs offer a powerful means of finding the global optima of functions, particularly those in which the solution space is irregular or discontinuous. One area in which system identification problems can be of considerable benefit is in damage detection and quantification. Another application is the design of the manufacturing lay-up for producing composites optimised for strength and/or lightweight rigidity. This chapter will pay particular interest to this latter application and will reference two examples of how the described method has been successfully applied in the former. Of particular benefit to this method is the fact that the result of the algorithm is an updated finite element model that represents the actual structure that can subsequently be used under in-situ loading conditions.

Early investigations into inverse damage detection were summarised by Hajela and Soeiro (1990), who considered damage in the analytical model to be represented by a local, reduced elastic modulus. The unknowns to be solved were

typically a set of continuous damage parameters between lower and upper limits of 0 and 1 respectively. This approach presents the advantage that the general form of the analytical model remains the same between iterations. However, some damage scenarios cannot be easily represented by a continuous variable between upper and lower constraints. Often, damage is discontinuous and of a discrete nature, complicated by the fact that the form of the stiffness matrix may change as the model is updated. An investigation by Louis et al. (1997) used a GA and the boundary element method to locate damage represented by rectangular cut-outs in a plate, while Chou and Ghaboussi (1997) combined the FE method and GAs to resolve damage in truss structures. Another form of system identification problem in which the distribution of inclusions in a structure was deduced using GAs and an FE model was investigated by Schoenauer et al. (1997). While these and similar investigations are largely problem specific, Rodic & Gresovnik (1998) described a general computer method, in which the Elfen package was combined with a programmable shell to solve inverse structural problems.

This chapter focuses on the use of the FE package to solve structural inverse problems of a generalised nature that unlike the above references requires no problem specific coding once the GA has been set up to identify the attributes of the FE model that require optimisation.. It can be readily adapted for use as a powerful design tool to optimise the design of structures subject to physical and performance constraints as well as being available to determine damaged properties of a structure from its response to loading as shown by Sherratt et al. (2001) and Panni & Nurse (2001).

GENETIC ALGORITHMS

Genetic algorithms are a set of powerful optimisation tools that are used in a wide range of disciplines and engineering applications. They take their inspiration from Darwin's theory of natural selection and the survival of the fittest. Unlike gradient based optimisation techniques, GAs operate on a 'population' of solutions and are well suited to the optimisation of discrete or non-continuous functions. A wide range of suitable texts is available to the interested reader (e.g., Goldberg, 1989) and only a rudimentary introduction is presented here.

At the heart of the algorithm is the concept that string-like (genotype) representations of actual numerical solutions (phenotype) to a potential problem are manipulated using simple genetic operators. In most cases this is an appropriate binary representation but more sophisticated representations have been successfully used. To begin, an initially random population of potential solutions is generated.

Each solution in the population is then converted into its genotype representation and assigned a numerical fitness value based on how well that particular solution minimises an objective function. Subsequent generations are developed by probabilistically selecting pairs of parent solutions to combine and produce offspring solutions in the next generation. Combination of the parents to produce offspring is achieved using a ‘crossover’ operator that combines certain characteristics of both parent solutions. The selection of parents is biased in favour of those that have the best fitness values. In this way, subsequent generations exhibit the beneficial characteristics of the previous generation, and any characteristics that lead to poor fitness values are rapidly excluded from the population. In general the average fitness of subsequent generations can be seen to rapidly improve, and the fittest solution in each population should converge to that value which minimises the objective function.

A pseudo-code representation of the GA to solve inverse problems is listed below:

1. *Input GA control parameters, (size of population, probabilities of mutation and crossover, etc.)*
2. *Generate a population of random solutions representing possible damage scenarios*
3. *For each individual*
 Convert solutions into suitable coded form
 Build FE model
 Solve FE model
 Extract response data
 Evaluate fitness
4. *Generate subsequent population of coded solutions, based on the fittest individuals from the previous generation applying crossover, reproduction and mutation*
5. *Loop steps 3-4 for fixed number of iterations or until specified convergence criterion is met*

The essential advantage of genetic algorithms is the capacity to very quickly search the complete solution space and locate optimal areas. These optimal areas can then be rapidly exploited to find the global optima.

The two genetic operators at the heart of a genetic algorithm are crossover and mutation. In crossover, the parents are split at a random location and swapped over so that half of the solution comes from each parent. In mutation, a random bit in the string is inverted from a ‘1’ to ‘0’ and vice versa. Each of these operations is applied with a given probability.

IMPLEMENTATION

The method uses the *LUSAS* finite element package. This commercially available general-purpose finite element application incorporates a user programmable interface, which was designed to help automate repetitive FE actions.

The interface allows ASCII script (command) files to be executed within the application. These script files contain the GA code and the *LUSAS*-specific instructions for building the geometry of the model, solving it and interrogating the results database to extract the response data. The output response data is then used to formulate the objective function and determine the fitness of that particular solution.

The scriptable interface is built around a set of core objects which an applications programmer can manipulate using specific methods. Both VBSCRIPT and JSCRIPT can be used to write the scripts. Although these scripting languages are relatively basic, they have sufficient functionality to describe even complicated genetic algorithms.

The script file takes the general form shown below:

```
$ENGINE=scripting engine
{script body}
{procedures and functions}
```

The script body contains the main sub-routines (see below) that control the GA, while the procedures and functions represent often-repeated standard routines that are called from more than one of the sub-routines in the script body:

<i>Input_Data</i>	(Requests user input of GA parameters)
<i>Generate_Original_Population</i>	(Randomly generates initial population)
<i>Evaluate_Population</i>	(Builds and solves FE models, extracts data and evaluates solution fitness values)
<i>Generate_New_Population</i>	(Uses genetic operators to build a new population)
<i>Output_GA_Data</i>	(Monitors the progress of the GA)

On executing the command file from within *LUSAS*, the user is prompted to input the control parameters of the GA including the size of the population. No further user input is required, and data regarding the progress of the algorithm is output to a results text file.

EXAMPLES

Damage Detection

The following examples demonstrate how the method has been used to resolve a number of different damage scenarios incorporating different encoding methods and *a priori* knowledge.

Firstly, with reference to damage detection in composites, the central idea of the approach is that an analytical model representing a loaded structure is systematically updated to minimise the difference in structural response between the model and equivalent experimentally obtained data. Mathematically, the problem is stated as:

$$\text{Minimise objective function,} \quad f = d_{exp} - d_{ana}$$

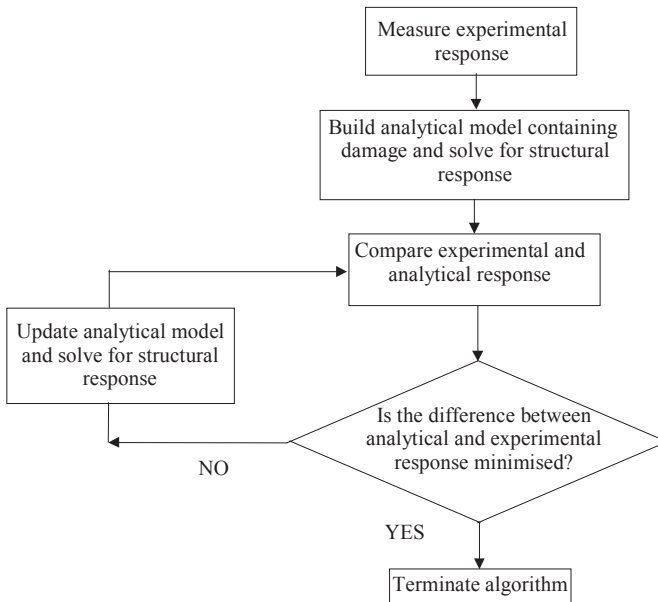
where d_{exp} is a vector of j experimentally determined structural responses, z ; and d_{ana} is the corresponding vector of j analytically determined responses, Z .

$$d_{exp} = [z_1, z_2, \dots, z_j]^T$$

$$d_{ana} = [Z_1, Z_2, \dots, Z_j]^T$$

Experimental input data is fed into a GA, which automatically uses it to iteratively update the analytical model and minimise the above objective function.

Figure 1: General approach to resolve inverse system identification problems in which the damaged state of a structure is sought



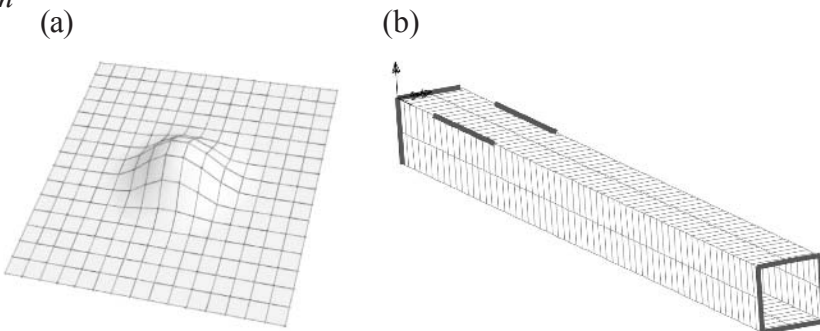
On termination of the algorithm, the analytical model should represent the actual damage that is present in the structure.

A general system identification procedure is represented as a flow chart in Figure 1.

In the first instance, we wish to fully define a delamination within a laminated composite panel using whole-field measurements of out-of-plane deformation when the panel is subject to vacuum loading. In this case, both the delamination depth and shape are the unknowns to be solved. The delamination is encoded in the GA by converting the surface of the panel into a 16×16 square grid, Figure 2(a). Each of these 256 grids can either contain a delamination or not. Therefore the coding comprises a 256 long binary string with each bit representing a grid. A '1' represents the presence of a delamination while a '0' represents no delamination. The remaining unknown is the depth. Since there are a finite number of integer depths that a delamination can occur at, the depth is represented by appending an integer to the front of the binary bitstring. This encoding of the information allows various shapes of delamination to be considered—with greater resolution afforded by a finer mesh. The fitness value can be determined by evaluating the difference in peak displacement values for the experimental and analytical models. However, this does not guarantee convergence to a unique solution. Therefore, the fitness function is constrained using penalty functions that accommodate *a priori* knowledge of the solution. In this case, the pattern of the surface displacement plots is approximated using a series of inequality rules. Any solution that violates any rule has a penalty function appended to the fitness function. In this manner the GA favours those solutions that best match the overall pattern of the experimental data and ensure convergence to a unique solution.

Another application involves the detection of longitudinal cracks along the corners of pultruded composite beams, Figure 2(b) using a set of reduced beam bending stiffnesses obtained from 3-point bend tests as input for the GA. Assuming

Figure 2: (a) Deformation pattern of composite laminate surface under vacuum loading; (b) FE mesh including longitudinal cracks in a box-section beam



the *a priori* knowledge that the cracks appear in pairs along the edges of a surface and that they are of equal length, the crack information can be encoded in terms of a surface, a length of crack and the starting point. By definition the model is discretised in terms of the finite element mesh and therefore, rather than represent crack length and starting position in terms of continuous values, they can be represented in terms of discrete numbers of elements along the length of the beam. These integer values are then converted into binary code and concatenated to represent the full solution code. This application is described in more detail in Sherratt et al. (2001).

Manufacturing Lay-Up Design

The high strength-to-weight ratio properties of fibre-matrix composites are a function of the orientation of different laminae within the sandwiched structure. Consequently, the design engineer must carefully select and optimise where possible the orientation of the fibres to produce the desired strength and stiffness characteristics. Though in current practice this is still something of a ‘black art’ and new designs rely much on ‘tried and trusted’ solutions.

Here, the FE/GA tool is applied to a laminated box-section beam shown in Figure 3 (dimensions in mm) with the aim of finding the laminate lay-up that minimises boom-tip deflection subject to given design and strength constraints. The objective function to be minimised is formulated as a penalised boom-tip displacement, which includes a penalty term for violation of the strength constraint. Since the aim of the problem is to find a stacking sequence, which comprises a list of ply orientations for both the box-section flanges and the webs which minimises the overall deflection of the box-section, it is necessary to develop a chromosomal or genotype representation for the design variables. The stacking sequence or lay-up of the laminate is simply a list of ply fibre orientations that make up the entire thickness of the laminate.

Figure 3: Geometry and loading of box section beam for minimum deflection lay-up design

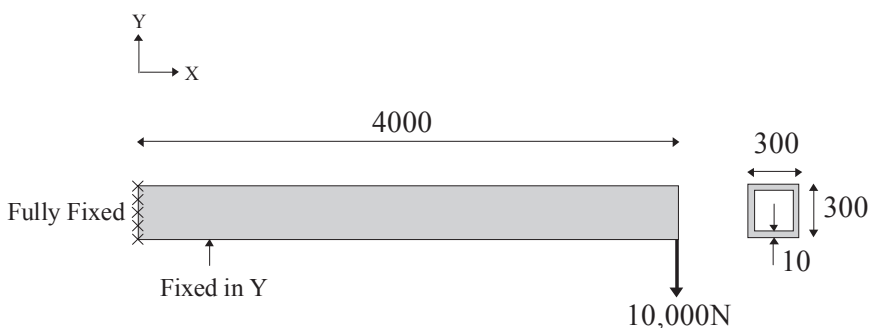
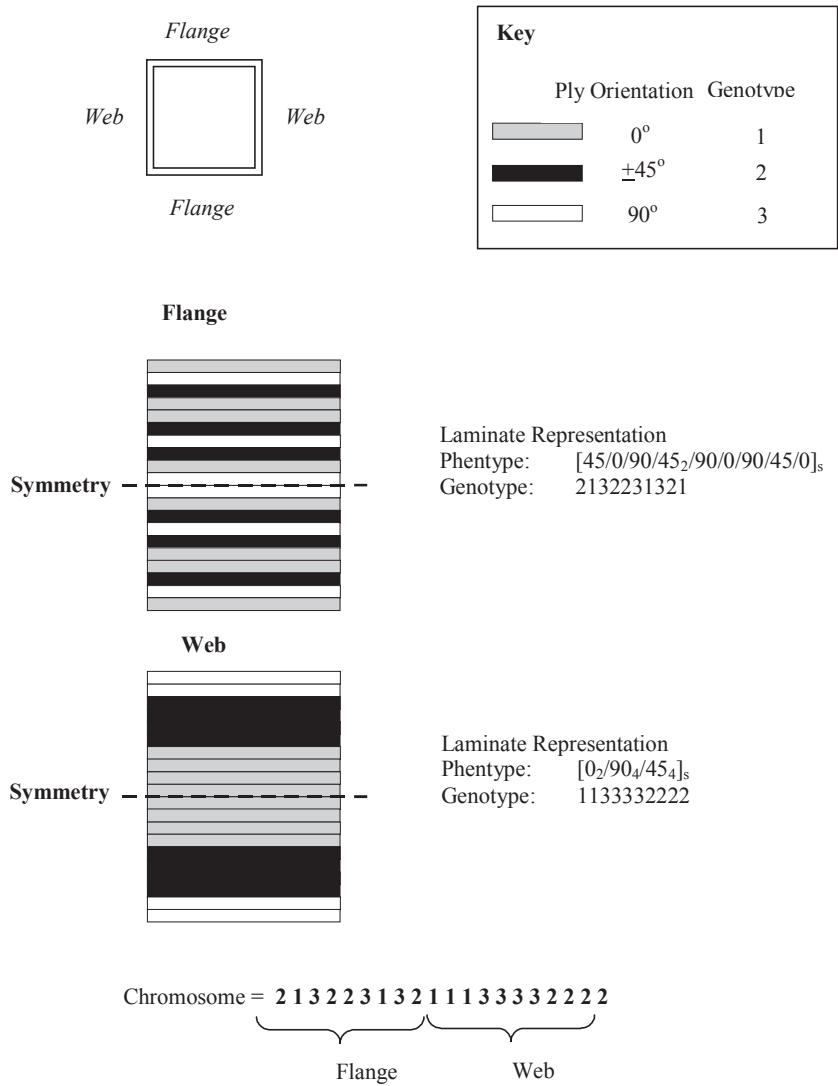
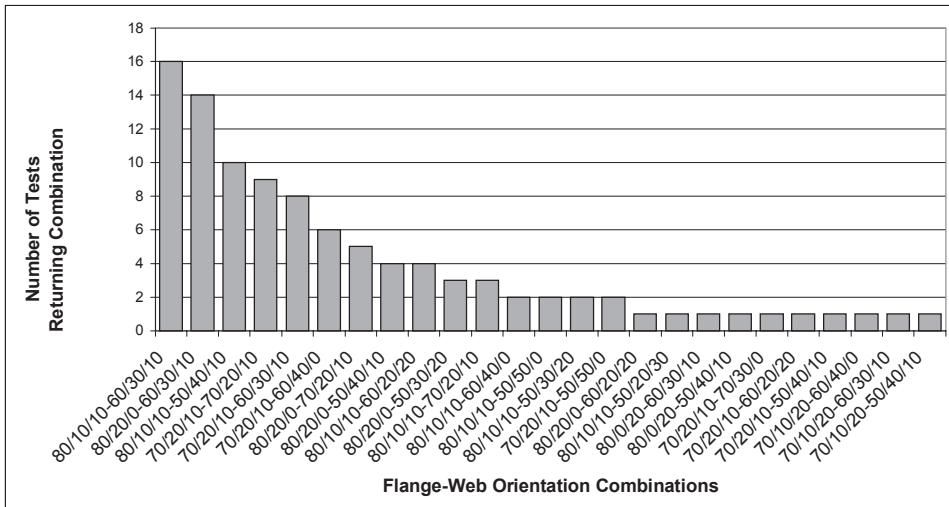


Figure 4: Chromosome definition for box-section beam lay-up optimisation



Three discrete ply orientations will be adopted (0° , $\pm 45^\circ$ and 90°). This lends itself well to representation using a three integer string-like genotype code with each integer in the string representing a different ply orientation (Figure 4). Additional constraints are enforced by the particular genotype encoding adopted and by introducing a repair strategy that eliminates infeasible solutions from the gene-pool and replaces them with feasible ones. The method uses a 3 digit real encoding system to represent the discrete laminate ply orientations in both the flange and webs.

Figure 5: Distribution of algorithm test runs versus orientation combination



The problem is a classical laminated composite optimisation problem and similar problems have been previously addressed, though only to simple geometry panels subject to either in-plane and/or out of plane bending. Significantly less research has been performed on more complex geometry models, without which it is unlikely that industry will adopt the method. In this particular area the work represents a significant contribution to the present body of knowledge.

Analysis of the results from the perspective of proportions of ply orientations in the laminate shows a clear convergence among the tests to a smaller set of solutions that have the same proportion of 0° , $\pm 45^\circ$ and 90° fibres. These are summarised in Figure 5.

The fittest individual found by the GA in 100 test runs is in good agreement with results that might be expected using sound engineering principles. This indicates that the approach adopted here may prove to be a valuable tool in this type of design optimisation problem.

FUTURE TRENDS

The principal advantage of this method is the relative ease of implementation, which requires only knowledge of GAs and simple script language programming. It avoids the relatively complex and tedious task of manually assembling and solving FE problems. Although the authors recognise that the present method may be computationally expensive, since it requires many standard FE solutions to be solved, this may be offset in the future with the increasing availability of computing

power. Furthermore, the inherent parallelism in GAs means that the described method may significantly benefit from developments in parallel computing.

CONCLUSIONS

A method for integrating GAs within a finite element environment has been presented. This approach allows the possibility of applying the undoubted benefits of GAs to a wide variety of structural problems. This present chapter has described how an integrated GA and FE analysis can be used to solve structural problems associated with composites. However, the same method can be used to solve a whole range of system identification for structural inverse problems in which unknowns such as applied load may be resolved. A further application of this method is the automated optimisation of structural design, in which a structure and a material are designed subject to physical, cost and manufacturing constraints.

REFERENCES

- Chou, J.-H. & Ghaboussi, J. (1997). Structural damage detection and identification using genetic algorithms. *Intelligent Engineering Systems Through Artificial Neural Networks*, 7, 395-400.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley Publishing Co.
- Hajela, P. & Soeiro, F.J. (1990). Recent developments in damage detection based on system identification methods. *Structural Optimization*, 2, 1-10.
- Louis, S.J., et al. (1997). Flaw detection and configuration with genetic algorithms. In Dasgupta, D. & Michalewicz, Z. (Eds.), *Evolutionary Algorithms in Engineering Applications*. Springer Verlag.
- LUSAS v13.3 (2001). London: FEA Ltd Kingston
- Panni, D.C. & Nurse, A.D. (2001), Integrating genetic algorithms and the finite element method to solve structural inverse problems. *International Conference on Computational Intelligence for Computer Modelling, Control, and Automation*, Las Vegas, July 10–12, 38-46.
- Rodic, T. & Gresovnik, I. (1998). A computer system for solving inverse and optimization problems. *Engineering Computations*, 15(7), 893-907.
- Schoenauer, M., et al. (1997). Identification of mechanical inclusions. In: Dasgupta, D. & Michalewicz, Z. (Eds.), *Evolutionary Algorithms in Engineering Applications*. Springer Verlag.
- Sherratt, P.J., Panni, D.C. & Nurse, A.D. (2001). Damage assessment of composite structures using inverse analysis and genetic algorithms. *Key Engng. Matls*, 204-205, 409-418.

SECTION III:

**FUZZY LOGIC
AND
BAYESIAN
SYSTEMS**

Chapter IX

On the Modelling of a Human Pilot Using Fuzzy Logic Control

M. Gestwa and J.-M. Bauschat
German Aerospace Center, Germany

ABSTRACT

This chapter discusses the possibility to model the control behaviour of a human pilot by fuzzy logic control. For this investigation a special flight task is considered, the ILS tracking task, and an evaluation pilot has to perform this task in a ground based flight simulator. During the ILS tracking task all necessary flight data are stored in a database and additionally the pilot commands are recorded. The development of the described fuzzy controller (the fuzzy pilot) is based on cognitive analysis by evaluating the recorded flight data with the associated pilot comments. Finally the fuzzy pilot is compared with the human pilot and it can be verified that the fuzzy pilot and the human pilot are based on the same control concept.

INTRODUCTION

It is a must for manned real-time simulations to take Man/Machine Interface (MMI) aspects into account. The demanded quality of the MMI-simulation depends on the particular aim of the simulation. Up to now, however, no clear

answer is given to the question: how realistic the real-time simulation at least has to be in relation to a certain flight task? One typical example is the application of motion cues. Simulated motion is not necessary in the case of so-called Flight Training Devices (FTDs) used generally for initial and procedure training. These less complex simulators replicate the actual aircraft cockpit, but do not provide a visual system or motion system. On the other hand, it is well known that the pilot's behaviour is influenced by the aircraft motion in the case of high precision tasks, e.g., when he has to perform an ILS-approach under bad weather conditions such as heavy wind shear, turbulence and gusts (see, e.g., Bussolari et al., 1984; Schänzer et al., 1995). High gain tasks increase the workload of the pilot significantly. Hence, a deeper understanding of these subjectively sensed influences on the pilot's reactions is necessary.

Particular aspects of MMI problems are covered at the Institute of Flight Research of the German Aerospace Center (DLR) by a project named AIDA (Airborne Identification and Development of simulation fidelity criteria using ATTAS). It deals with the comparison of ground-based simulation and real flight. Different experienced commercial pilots have to perform well defined tasks on two simulators with different equipment standards and on the DLR in-flight simulator ATTAS. The results are used twofold: (1) to classify the pilot tasks and (2) to define the demands on the simulator equipment to enable an adequate conduction of the given task. The project also provides additional information related to pilot workload aspects, which leads to a better understanding of the man/machine interface between pilot and aircraft (see Bauschat, 2000).

The data gained from the simulator sessions and flight-tests are evaluated in various ways to gain as much information as possible. Data evaluation in the time domain delivers a good idea about the quality of a task solution. The assessment of the solution quality is additionally supported by statistical evaluations. Pilot's effort to solve a task can be described in the frequency domain, where power spectral density data are used. But the investigation of the individual strategies pilots are using to achieve a good performance during a particular task makes it necessary to model the pilot and the MMI. Investigations based on pilot models support a better understanding of the interaction between pilot and MMI. Sub-models describing particular behaviour patterns, which have been found evaluating the AIDA database, should be easily added to the pilot model. Such a sub-model may, for example, include the influence of the pilot's subjective impressions on his task performance.

With respect to the idea of the AIDA project, it was soon clear that a knowledge-based method should be used to model the control characteristic of a pilot. In this particular case a fuzzy logic approach has been chosen. Fuzzy logic provides a lot of benefits, because verbal descriptions which an expert has given can

be introduced directly into a controller approach. The controller itself is easy to handle from the point of view of an engineer, which means it is easy to modify.

BACKGROUND

In the field of airplane system technology, investigations focusing on Human Factors or Man/Machine Interface have played an important role for decades. Airplane handling qualities can only be evaluated properly, if pilots and their subjective impressions are taken into account. Basic work has been done here by Cooper et al. (1969). The so-called Cooper/Harper Rating Scale provides the evaluation pilot with a tool to rate handling qualities and own effort between 1 (excellent) and 10 (not controllable). Pilot models with different degrees of complexity have been used for theoretical investigations based on computer simulations (see, e.g., McRuer, 1988).

A typical example for the necessity of investigations based on real-time ground simulations is described by Ashkenas et al. (1983). During one of the first free flights of the shuttle orbiter, a PIO incident occurred during the approach to the runway. PIO stands for Pilot Induced Oscillation and can be observed in airplanes with fly-by-wire control systems. PIO is characterised by disharmonic pilot control inputs leading to unintended heavy aircraft motions. Mainly system time delays in combination with closed loop pilot tasks can trigger PIO prone situations. Duda (1997) describes the effect in greater detail. The PIO problem of the space shuttle has been investigated with the help of a very simple real-time simulator and has been solved. The effect observed in the ground-based simulation was modelled and simulated additionally using a linear pilot model.

How realistic a flight simulation is depends on the quality of the simulator. Fixed-based simulators are useful if the pilot has to perform tasks where the motion cues have minimal importance. Typical start or landing procedures, under Instrumental Flight Rules (IFR) conditions, with minimum external disturbances, can be easily performed.

Moving based simulators give a pilot a more realistic impression of a flying aircraft if the simulators are additionally equipped with a good visual system.

However, deficiencies of ground based simulators are well-known (see, e.g., Harper, 1991). Some of them are as follows:

- In the case of a fixed-based simulator, there are no proprioceptive cues.
- A motion-system has physical limits and therefore some cues are more or less suppressed (i.e., only 10-15% of the real roll acceleration is available).
- Some cues, such as the load factor in the vertical aircraft axis, are missing.
- Because of washout filtering, some cues are generated which never appear in a real airplane. The design of washout filters is still a kind of black art.

- The harmonisation between aircraft motion-system and visual-system dynamics is a problem area.
- The workload of the pilot in a simulator and in real flight is generally different. Investigations concerning PIO effects, for instance, have shown this.

The influence of these effects on a pilot need further investigation and is done within the AIDA framework. If the effects can be modelled, they can be implemented within the software system of the above mentioned MMI model.

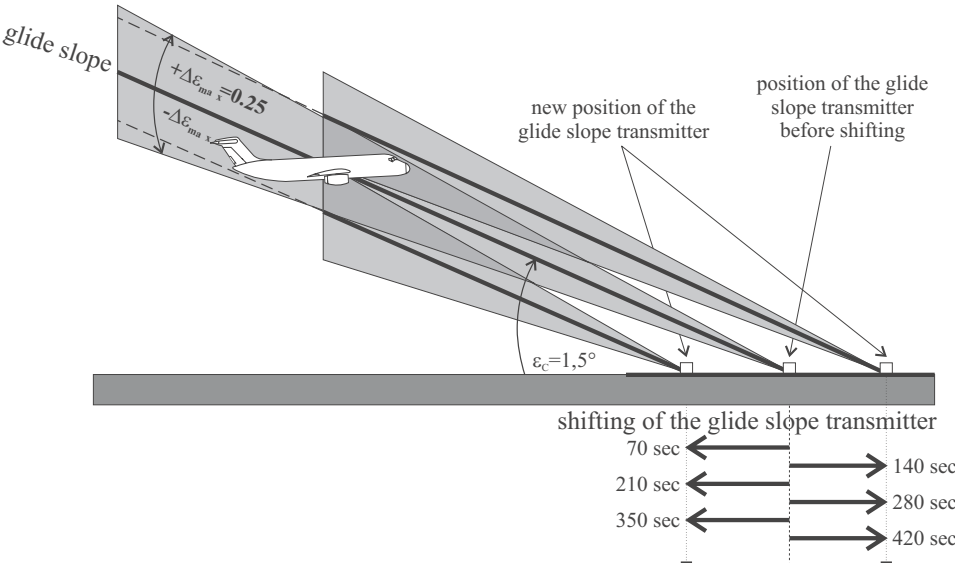
Most of the pilot models mentioned at the beginning of this section have the disadvantage that their potential concerning adaptation to a particular pilot performing a particular task is not very high. At the DLR Institute of Flight Research, it was decided to find a suitable pilot model approach, which satisfies most of the demands in the above mentioned AIDA project. Broad discussions with soft computing experts and an intensive literature study (e.g., Zadeh, 1964; Mamdani, 1974) ended in the decision to try a fuzzy logic control approach.

THE ILS TRACKING TASK

In order to get the necessary data for the development of the *fuzzy pilot*, a particular flight task was performed in a fixed base ground simulator (no motion system). During the task all necessary data are stored, e.g., flight path angle, airspeed, altitude, etc. Additionally the pilot documented control commands and his strategy. These pilot commentaries were recorded with a dicta phone. This particular flight task is called *ILS tracking task* (abbrev. ITT) and can be described in the following way (see Bauschat, 2000):

The ITT consists of seven phases. At the beginning the aircraft flies with the target speed established on the glide slope. After 70 sec the glide slope transmitter shifts to a new position so that the glide slope indicator on the display in the cockpit moves downwards or upwards to its maximal deflection. In the case that the aircraft is above the glide slope, the pilot has to reduce the altitude. For this maneuver he has 70 seconds. After this procedure the glide slope transmitter is shifted again so that the glide slope indicator moves upward to its maximal deflection. Now the aircraft is under the glide slope and the pilot has to climb with the aircraft. Again the pilot has 70 seconds to compensate the glide slope deviation. In the next phase the glide path indicator moves downward again, afterwards upward again, etc. The whole ITT task requires 490 seconds, whereby the glide path indicator moves three times downward and three times upward in a given sequence.

Figure 1: Artificial ILS



The implementation of the ITT in the flight simulator is based on a synthetic glide slope transmitter resp. a synthetic navigation system. On the artificial glide slope, the aircraft has a glide path angle of $1,5^\circ$. The maximal deviation amounts to $0,25^\circ$ (see Figure 1). At the first transmitter movement, the pilot has to compensate an altitude difference of 196 m. The described ITT focuses only the longitudinal motion of the aircraft.

DEVELOPMENT OF THE *FUZZYPILOT*

The pilot has to observe a lot of different instruments in the cockpit. To find out the information an evaluation pilot uses to perform the ITT, he has to fill in a questionnaire. In this questionnaire the pilot describes the priority of the instruments he needed. A given scale is divided into ten priorities, which are subdivided into three classes again (see Table 1).

This information is a basis to find the measurements for the *fuzzypilot*. The circled numbers in Table 1 are the priorities of a professional pilot who has performed the ITT. The priorities of the pilot show that five indicators are important to perform the ITT. By three of them the signal dynamic is important, too. All *very important* indicators listed in Table 1 can be used for the *fuzzypilot* model. The

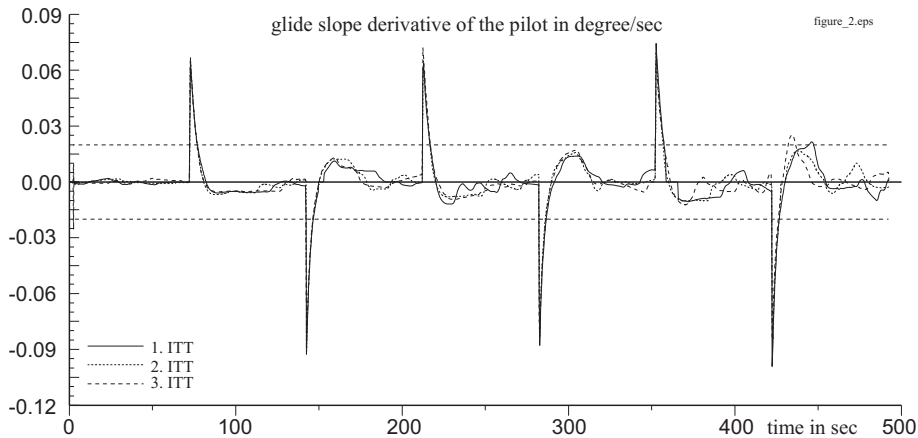
Table 1: Questionnaire of the instrument priority

(1) Priority of the instrument										
	very important			important				unimportant		
pitch	1	2	3	4	⑤	6	7	8	9	10
speed trend	1	2	③	4	5	6	7	8	9	10
speed difference	1	2	③	4	5	6	7	8	9	10
glide slope	①	2	3	4	5	6	7	8	9	10
flight path angle	①	2	3	4	5	6	7	8	9	10
altitude	1	2	3	4	5	6	7	8	9	⑩
vertical speed	1	2	3	4	5	⑥	7	8	9	10
DME	1	2	③	4	5	6	7	8	9	10
(2) Priority of the instrument dynamic										
	very important			important				unimportant		
pitch	1	2	3	4	5	⑥	7	8	9	10
speed trend	1	2	3	4	5	6	7	⑧	9	10
speed difference	1	2	③	4	5	6	7	8	9	10
glide slope	①	2	3	4	5	6	7	8	9	10
flight path angle	1	2	3	④	5	6	7	8	9	10
altitude	1	2	3	4	5	6	7	8	9	⑩
vertical speed	1	2	3	4	5	6	7	8	9	⑩
DME	1	2	③	4	5	6	7	8	9	10

dynamics of the indicators can be substituted with the ratio of difference without information loss. It is obtained by

$$\dot{X} = \frac{X_{t+\Delta t} - X_t}{\Delta t}$$

Between some information exists a dependence, which can be used to reduce the number of measurements. For example the flight path angle can be derived from the glide slope and its change due to time. If an aircraft is established on the glide slope with the proper airspeed, the flight path angle will be proper too. Consequently the flight path angle can be disregarded as a measurement. The distance (DME) can be derived from the sensitivity of the glide slope indicator. A reduction of the distance causes a fast movement and a frequent change of the glide slope indicator. The commentary of the pilot reflects this reduction, too. So, the number of measurements can be reduced from seven to four. These are the glide slope $\Delta\epsilon$, the derivative of the glide slope $\Delta\dot{\epsilon}$, the speed difference Δv and the derivative of the speed difference $\Delta\dot{v}$. The control commands of the *fuzzy pilot* are defined by the control elements in the cockpit. Consequently the *fuzzy pilot* delivers a side stick command and a thrust command.

Figure 2: Glide slope derivative

Specification of the Linguistic Terms and their Fuzzy Sets

In the following subsections the specification of the linguistic terms and their associated fuzzy sets are described. As examples the glide slope derivative and the side stick command are explained in detail. The specification is based on the pilot commentary and the recorded flight data.

Derivative of the Glide Slope $\Delta\dot{\epsilon}$

In the diagram of the glide slope derivative, the movement of the glide slope transmitter can be clearly recognised. After the glide slope transmitter has moved, the derivative reduces its value always in the range from $-0.02^\circ/\text{sec}$ to $0.02^\circ/\text{sec}$ (see the marked range in Figure 2).

The investigation of the pilot reaction shows that he doesn't make any control command immediately after the glide slope transmitter starts to move. This behaviour is normal because the pilot knows that he cannot follow the glide slope directly and that a new phase of the ITT starts. So, the pilot waits until a quasi-stabilised situation is indicated and then he starts to compensate the glide slope deviation.

To model this effect the *fuzzy pilot* has a separate controller. The measurement of this controller is the glide slope derivative with the universe $[-0.1^\circ/\text{s}, 0.1^\circ/\text{s}]$, and in agreement with Figure 2, the three linguistic terms *below*, *zero* and *above* are defined. This fuzzy set overlaps at the broken lines. Table 2 includes the definition points.

Table 2: Points of the measurement

μ	below	zero	above
0	-0.100	-0.025	0.015
1	-0.025	-0.015	0.025
1	-0.025	0.015	0.025
0	-0.015	0.025	0.100

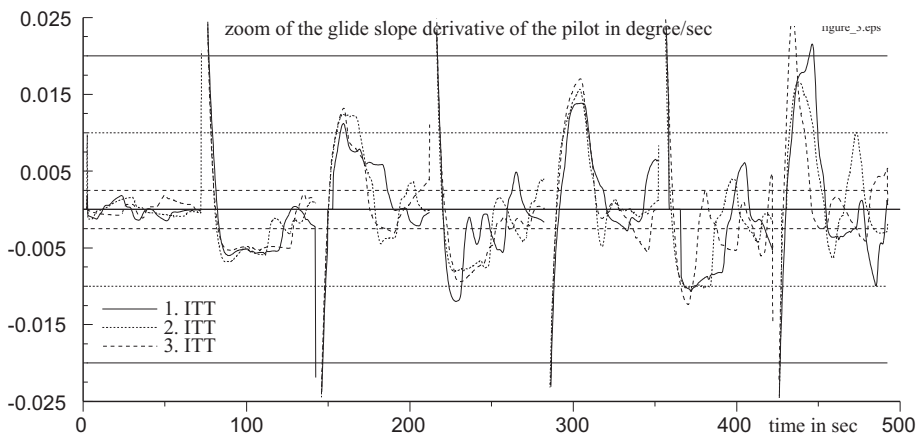
The control commands of the separate controller are the two linguistic terms *yes* and *no* which indicate that the glide slope transmitter has moved. So, the rule base contains the three rules:

IF	$\Delta \dot{\epsilon}$ IS above	THEN	shifting IS <i>yes</i>
IF	$\Delta \dot{\epsilon}$ IS zero	THEN	shifting IS <i>no</i>
IF	$\Delta \dot{\epsilon}$ IS below	THEN	shifting IS <i>yes</i>

Now the linguistic terms of the glide slope derivative have to be defined in the universe $[-0.025^\circ/\text{s}, 0.025^\circ/\text{s}]$. To describe the strategy of the glide slope derivative in this universe, the area between the two broken lines in Figure 2 is enlarged in Figure 3.

In Figure 3 six horizontal lines and the zero line can be seen. Based on this classification the universe is divided into six areas and each area represents a special situation. First the three areas above the zero line will be explained:

Figure 3: Zoom of the marked area of Figure 2



- $[0.01^\circ/\text{s}, 0.02^\circ/\text{s}]$:
The glide slope transmitter has reached a new position. The motion of the glide slope indicator is normal. The pilot starts to stabilize the aircraft on the glide slope.
- $[0.0025^\circ/\text{s}, 0.01^\circ/\text{s}]$:
The aircraft is in stationary descent or climb. The glide slope deviation is reduced.
- $[0.0^\circ/\text{s}, 0.0025^\circ/\text{s}]$:
The aircraft is near to the glide slope and has to be stabilized on the glide slope. According to the commands of the pilot the airplane oscillates around the glide slope.
- $0^\circ/\text{s}$:
On the glide slope the derivative should have a value of zero. Consequently the aircraft is in a stationary state.

This description can be transferred to the area below the zero line. With the help of the commentaries of the pilot seven linguistic terms can be defined. The terms are:

*descent rapidly, descent, descent slightly, zero, climb slightly,
climb, climb heavily*

and the determining points of their fuzzy sets are summarized in Table 3.

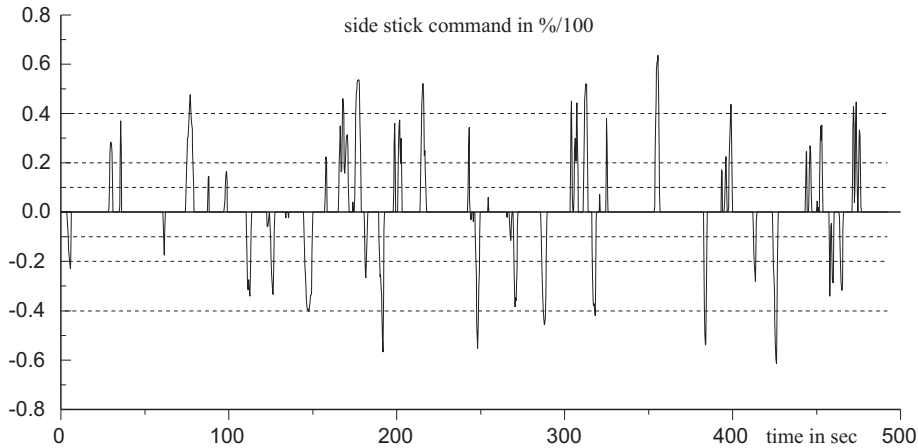
Side Stick Command q_{ss}

The side stick deflections are given as inputs to a rate-command system and Figure 4 shows the side stick commands of an ITT. It can be seen that the pilot commands have mainly the shape of short peaks. This control behaviour is typical for a pilot who is using a rate command system. A rate command system is a flight control system which stabilises an aircraft on a pitch angle the pilot has commanded. To model this control behaviour the maximum method will be used for defuzzification because this method causes a pulsed behaviour (see Kruse et. al., 1993; Kahlert et. al, 1993). With this approach the output of the *fuzzy pilot* will be defined by the maximum of the fuzzy-set. To find these maximum an investigation of the pilot side stick commands is helpful. The maxima can be defined roughly within three positive and three negative classes (dashed lines in Figure 4).

Table 3: Points of the glide slope deviation

μ	descent rapidly	descent	descent slightly	zero	climb slightly	climb	climb rapidly
0	-0.0250	-0.0150	-0.0050	-0.0010	0.0	0.0020	0.0120
1	-0.0150	-0.0120	-0.0025	0.0	0.0025	0.0075	0.0150
1	-0.0150	-0.0075	-0.0025	0.0	0.0025	0.0120	0.0150
0	-0.0120	-0.0020	0.0	0.001	0.0050	0.0150	0.0250

Figure 4: Side stick command of the pilot



With the values in Figure 4 and the pilot comments in addition the number of linguistic term of the side stick commands can be derived. They are:

pull heavily, pull, pull slightly, zero, push slightly, push, push heavily

The universe of the side stick command is defined by the side stick signal with the interval $[-1, 1]$. The linguistic terms are represented by triangular fuzzy sets and are presented in Table 4.

After all fuzzy sets of the measurements and the control commands are defined, the structure of the *fuzzy pilot* can be developed (see Figure 5).

Definition of the Rule Base

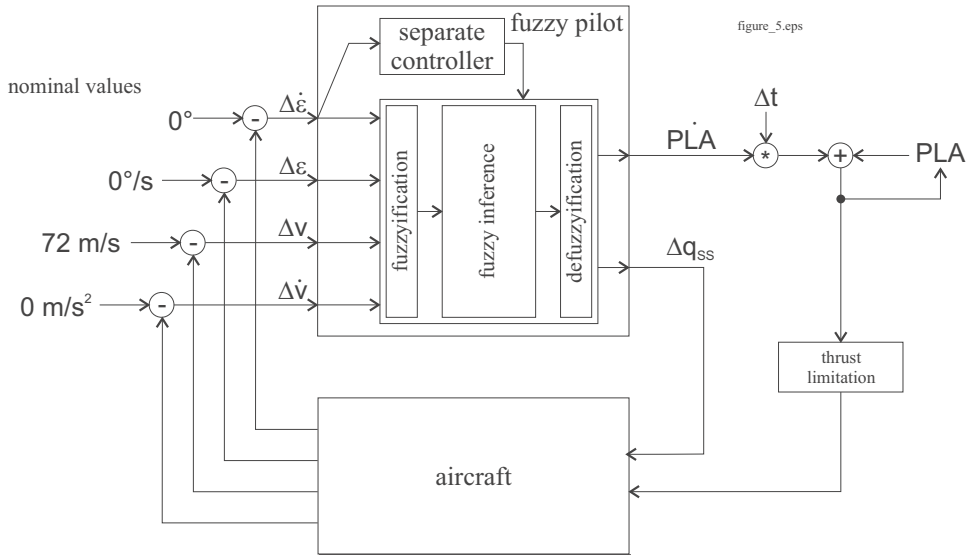
Now it is necessary to define the rule base of the *fuzzy pilot* using the specified linguistic terms. First the basic strategy of the pilot control behaviour should be determined from the time histories (see Figure 6) and the pilot comments. The basic strategy of the pilot to perform the ITT can be divided into three phases:

In the first phase the glide slope transmitter is moving. The absolute value of the glide slope derivative is large. Since he knows that he cannot follow the glide slope indicator, he waits until the indication moves slowly.

Table 4: Points of the side stick command

μ	pull heavily	pull	pull slightly	null	push slightly	push	push heavily
0	-1.0	-0.4	-0.2	-0.1	0.0	0.1	0.2
1	-0.4	-0.2	-0.1	0.0	0.1	0.2	0.4
0	-0.2	-0.1	0.0	0.1	0.2	0.4	1.0

Figure 5: Structure of the fuzzy pilot

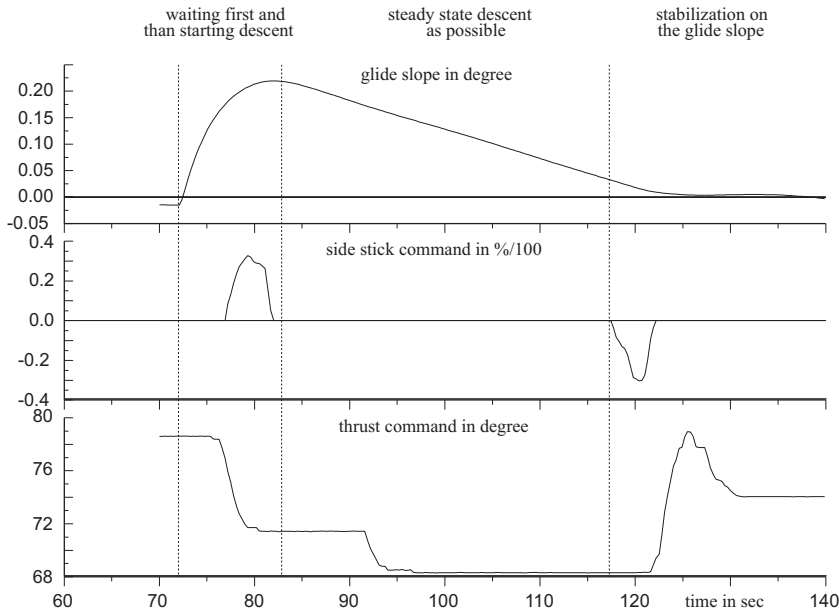


In the second phase the glide slope indicator moves slowly. Now the pilot starts compensating the glide slope deviation. Depending on the actual situation, he initiates a descent or climb. In this phase he is reducing the glide slope deviation very fast. He brings the aircraft as fast as possible close to the glide slope. During the descent or climb, the pilot only has to wait and to observe.

In the third phase the aircraft is near the glide slope. Now the pilot has to stabilize the aircraft on the glide slope. For this procedure he stops the descent or climb by pulling or pushing the side stick. Consequently the descent or climb is interrupted and the aircraft will be stabilized on the glide slope as well as possible. In this state only slight glide slope deviations have to be compensated by the pilot.

Evaluating the airspeed difference it is remarkable that in some situation the power lever has reached the lower limit but the speed is still too high. In this situation the pilot can reduce the speed difference only with the side stick. If he pulls the side stick, the aircraft interrupts the descent and the glide slope deviation is not decreasing. But it is the task of the pilot to compensate the glide slope deviation as quickly as possible. So, he has to accept the interim speed deviation. Furthermore a strong relationship between the side stick and the thrust command exists which arises from the energy balance and is considered by the pilot. The energy balance is the sum of kinetic and potential energy:

Figure 6: Strategy of the pilot



$$E_{\text{total}} = E_{\text{pot}} + E_{\text{kin}} = mgh + \frac{1}{2}mv^2$$

During a small interval the weight of the aircraft changes very slowly. With the assumption the potential energy depends only on the altitude and kinetic energy depends only on the speed. In this case the pilot pushes the side stick until the aircraft reduces the altitude. Potential energy will be transformed in kinetic energy. Without a thrust command the energy balance is constant and consequently the speed increases. According to this the pilot pulls the side stick without a thrust command. The aircraft reduces the speed and increases the altitude. The control strategy and the comments of the pilot reflect this fact and can be described as follows:

- If the pilot pulls the side stick, the corresponding thrust command results from the following aspects: If the aircraft has a positive speed difference, no thrust is given because the climb reduces the speed difference. If the speed difference is roughly equal zero, a very small amount of thrust has to be set to hold the speed. If the aircraft has a negative speed difference, thrust has to be given because the climb will increase the existing speed difference.
- If the pilot pushes the side stick, the thrust command results from the following aspects: If the aircraft has a negative speed difference, no thrust is given

because the descent reduces the speed difference. If the speed difference is roughly equal to zero, thrust has to be reduced a little bit to hold the speed. If the aircraft has a positive speed difference, thrust has to be reduced because the descent will increase the existing speed difference.

- The pilot increases only the thrust. The engines of the aircraft are beyond the centre of gravity. On account of this an increment of the thrust produces a pitch up moment. To compensate this upward movement, the pilot gives a small pitch down command.
- The pilot reduces only the thrust and the aircraft pitches, because a reduction of the thrust produces a pitch down moment. To compensate this downward movement, the pilot gives a small pitch up command.

The rule base of the *fuzzy pilot* has to be designed taking the above mentioned aspects into account. To define additional necessary rules an iterative process has to be included into the design process. The development of the *fuzzy pilot* starts with only one rule and the other rules are defined one after the other. If a situation during the ITT occurs where the *fuzzy pilot* has no rule, the *algorithm* aborts the ITT and reports the current flight state. Then the new rule can be defined by analysing the current flight state. The rules on the next page are defined with this method.

The control behaviour of these eight rules are shown in Figure 7 on the base of the first movement of the glide slope transmitter. This method was used to define the whole rule base of the *fuzzy pilot*.

The development of the *fuzzy pilot* model is based on the information gained from one pilot. This has the consequence that only the specific control characteristics of this subject will be matched. Taking the control characteristic of other pilots into account makes model modifications necessary. However, the main part of the rule base can be used unmodified because most of the rules are based on flight mechanic equations.

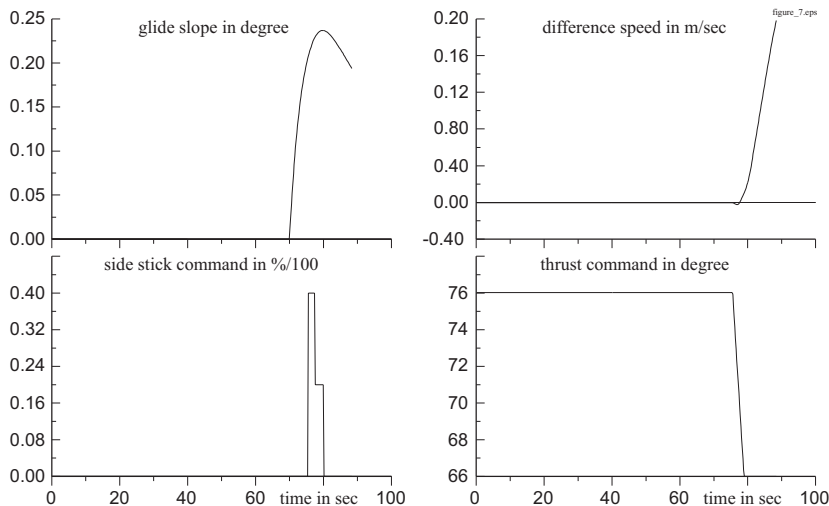
THE FUZZY PILOT IN COMPARISON WITH THE HUMAN PILOT

How the *fuzzy pilot* performs the ITT can be seen in Figure 8. The fuzzy logic system compensates all glide slope deviations caused by the movements of the transmitter.

The *fuzzy pilot* stabilises the aircraft on the glide slope with the demanded target speed.

IF	$\Delta \epsilon$ IS zero \wedge $\Delta \dot{\epsilon}$ IS zero \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS zero \wedge shifting IS no
THEN	q_{ss} IS zero \wedge $\dot{P}LA$ IS hold
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS climb \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS sink \wedge shifting IS no
THEN	q_{ss} IS push heavily \wedge $\dot{P}LA$ IS raise
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS climb \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS sink \wedge shifting IS no
THEN	q_{ss} IS push heavily \wedge $\dot{P}LA$ IS raise slightly
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS climb \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS climb \wedge shifting IS no
THEN	q_{ss} IS push \wedge $\dot{P}LA$ IS raise
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS climb slightly \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS climb \wedge shifting IS no
THEN	q_{ss} IS push \wedge $\dot{P}LA$ IS raise
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS zero \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS climb \wedge shifting IS no
THEN	q_{ss} IS push \wedge $\dot{P}LA$ IS raise heavily
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS descent rapidly \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS climb \wedge shifting IS no
THEN	q_{ss} IS null \wedge $\dot{P}LA$ IS hold
IF	$\Delta \epsilon$ IS over \wedge $\Delta \dot{\epsilon}$ IS descent \wedge Δv IS zero \wedge $\Delta \dot{v}$ IS climb \wedge shifting IS no
THEN	q_{ss} IS null \wedge $\dot{P}LA$ IS hold

Figure 7: A control result of the fuzzy pilot



It was further investigated in this project, if a fuzzy logic control approach is a suitable method to model the human control behaviour. An assessment of the control behaviour of the *fuzzypilot* makes a comparison with the control behaviour of the human pilot necessary. The human control behaviour is additionally influenced by different environmental aspects (see Bubb, 1992). It is not possible to take all these aspects into account in this chapter, so the comparison is based here on pilot's control strategy.

Comparison of the Glide Slope Compensation Strategy

Example glide slope deviations of the pilot and the *fuzzypilot* are compared. For this purpose the deviation of the pilot and the *fuzzypilot* during the ITT will be plotted in one diagram. Figure 9 shows that the curves of the glide slope deviations are quite similar.

During the 2nd, 4th and 5th transmitter movements, the two curves are matching acceptably. On the basis of the mean value and the standard deviation of the glide slope deviation, it can be assessed how the pilot and the *fuzzy pilot* maintain the glide slope. Table 5 shows that all mean values can be found in the proximity of the ideal mean value zero and all standard deviations are in an acceptable range.

Figure 8: Control policy of the fuzzy pilot by the ITT

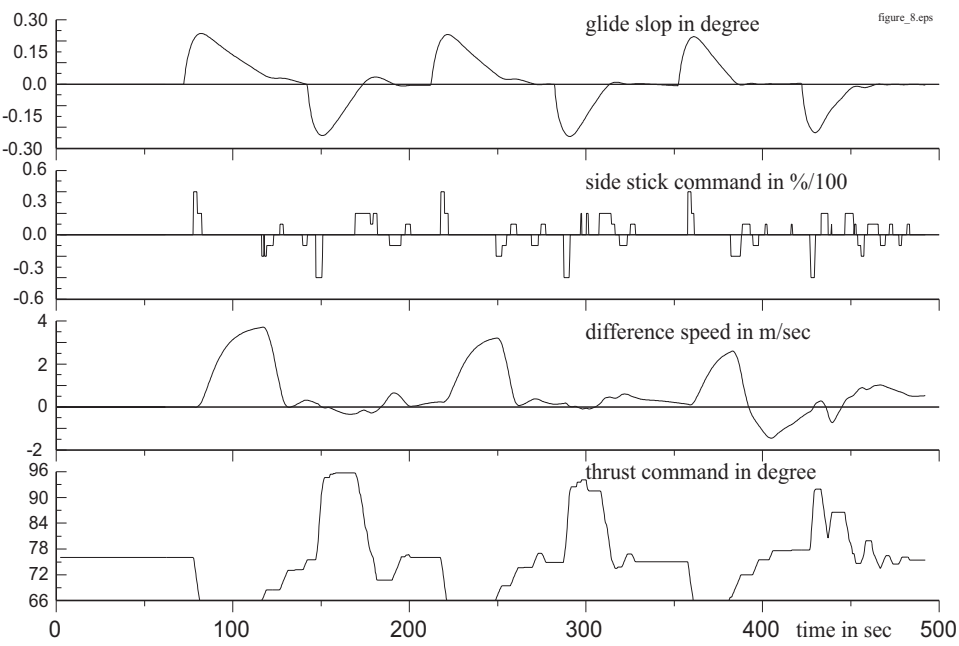


Figure 9: Glide slope deviation of the pilot and the fuzzy pilot

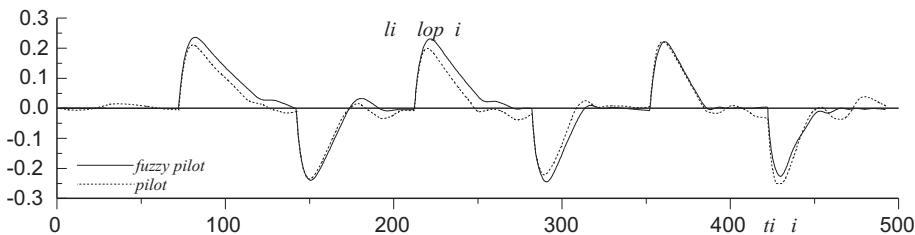


Table 5: Mean value and standard deviation of the glide slope signal $\Delta\epsilon$

	<i>fuzzy pilot</i>	pilot (1. ITT)	pilot (2. ITT)	pilot (3. ITT)
μ	0.0099	0.00083	0.0014	0.0083
σ	0.1010	0.10600	0.0940	0.0970
$\sigma_{\mu=0}$	0.0103	0.01130	0.0088	0.0095

Comparison of the Side Stick Commands

The *fuzzy pilot* commands are short pulsed side stick inputs like those of the human pilot. This can be seen clearly in Figure 10, which shows the side stick commands of the pilot and the *fuzzy pilot* for the ITT. Figure 10 illustrates also that the activity of the *fuzzy pilot* is similar to the human pilot. Table 6 also reflects this result, because the mean value and the standard deviation are nearly identical.

The maximum values of the side stick commands of the *fuzzy pilot* are acceptable. The commands of the human pilot can be characterized as jerky, short inputs in contrast to the *fuzzy pilot*, which prefers weak, long inputs. But the result of the commands is the same. However, at the end of the ITT, the *fuzzy pilot* is more active than at the beginning. This higher activity can be described by the cone-effect of the glide slope signal. The sensitivity of the glide slope signals increases with decreasing distance to the glide slope transmitter.

Figure 10: Side stick command of the pilot and the fuzzy pilot

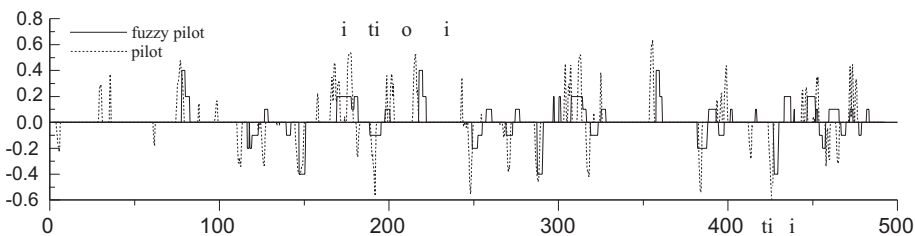


Table 6: Mean values and standard deviations of the side stick command

	<i>fuzzy pilot</i>	pilot (1. ITT)	pilot (2. ITT)	pilot (3. ITT)
μ	0.0061	0.0068	0.0062	0.0058
σ^2	0.1020	0.1410	0.1360	0.1470
$\sigma_{\mu=0}^2$	0.0110	0.0200	0.0180	0.0220

Comparison of the Control Strategy

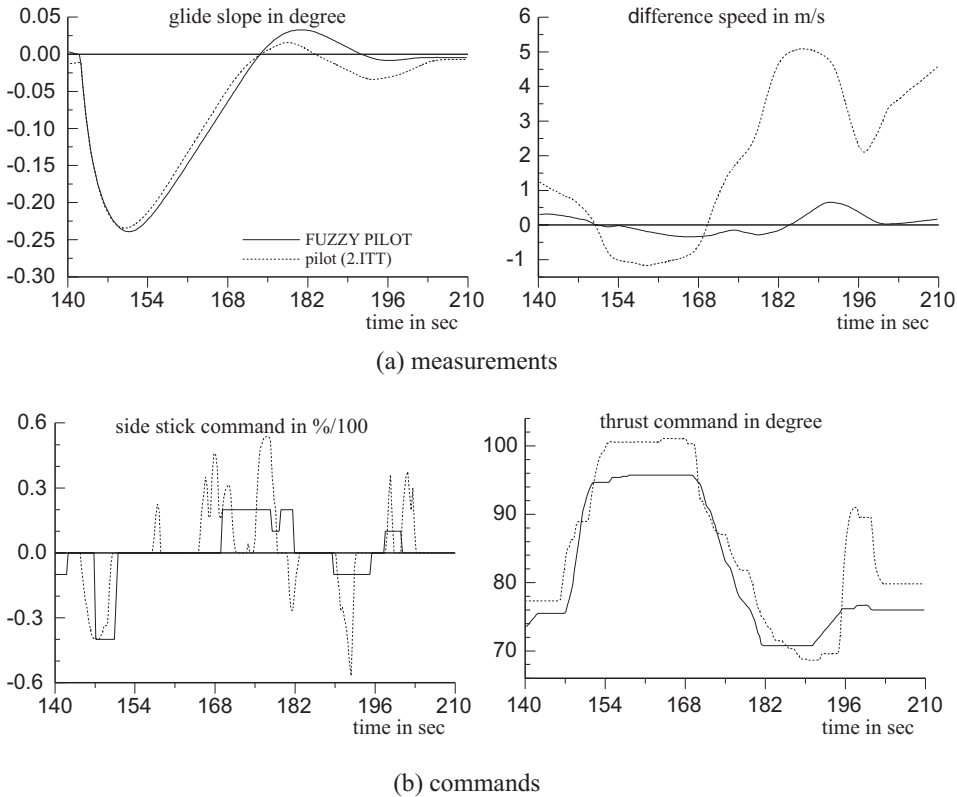
The comparison between the human control strategy and the control strategy of the *fuzzy pilot* implies, that the previously discussed measurements can be seen as an inherent part of a control concept. This concept defines how the pilot has to react in a situation and determines his control behaviour. During the ITT the situation is described by means of the glide slope and the speed. The control strategy is characterized by the side stick command and the thrust command. In the following example one ITT section is evaluated with respect to the control strategy.

The reaction of the *fuzzy pilot* is very similar to the reaction of the pilot (see Figure 11). Both react on a glide slope transmitter shifting with a stationary climb. During the stationary climb, the pilot waits until the aircraft is near the glide slope. Both begin to interrupt the climb so that the aircraft is stabilized on the glide slope. Both pilots stop the climb too early. The aircraft overshoots the glide slope. In this situation the pilot and the *fuzzy pilot* as well push the side stick to return on the glide slope. In the final phase of the ITT both pilots succeed to stabilize the aircraft on the glide slope. In this part of the experiment it can be observed that a side stick command is coupled with a thrust command. The commands *push* and *thrust reduction* as well as *pull* and *thrust increase* define a control unit. It is noticeable that the pilot pushes the side stick during the climb, because the aircraft overshoots the glide slope (see Figure 11 from 168 to 182 seconds). The *fuzzy pilot* has this control behaviour, too. The flight section in Figure 11 points out that the control strategy of both pilots during a negative, maximal movement of the glide slope transmitter is very similar. The previously described control strategy can be observed in every phase of the tracking task. An investigation of all opposed reactions of the *fuzzy pilot* and the human pilot shows that they are based on different flight states and so both reactions are correct. Although some differences between the reaction of the human pilot and the *fuzzy pilot* exist, the control strategy of the *fuzzy pilot* was correct and proper during this particular task.

FUTURE TRENDS

The important message of this investigation is that fuzzy control can be used to model the control behaviour of a human pilot. The next aim is the adaptation of the

Figure 11: Control strategy of the fuzzy pilot in comparison with the one of the human pilot



control behaviour to individual pilots. The *fuzzy pilot* can be used as a pilot model for defined flight tasks. The adaptation of the *fuzzy pilot* can be done using neural networks or evolution strategy. During the adaptation process the fuzzy sets of the *fuzzy pilot* have to be optimised. Before the rule base can be adapted all rules have to be checked. It has to be found out if a rule describes a particular pilot or if it is a general rule, which is describing the control characteristic of a typical pilot (e.g. an altitude overshoot with constant airspeed causes the pilot to decrease the thrust).

The comparison of the control behaviour is another aspect which can be optimised. In this project the control behaviour is compared with a simple method. The difference between two signals can usually be computed with the least squares method. In the case of the *fuzzy pilot*, this method cannot be used, because the *fuzzy pilot* and the human pilot did not start their control activities exactly at the same time. The use of the least squares method would result in an unrealistic difference. But with respect to the comparison of the control strategy, a small time

delay is not as important as the reaction. Therefore the time warp method can be used, because it is applied to find similar time histories in database. The time warp method makes it possible to define a fitting weighting function for an optimisation of the *fuzzy pilot*.

Fuzzy clustering makes it possible to derive fuzzy rules directly out of the cluster. For this purpose a cluster is interpreted as one rule and the accessory fuzzy sets can be evaluated by the projection of the cluster on the axes (Höppner et al., 1997). In the case of the modelling of a human pilot, fuzzy clustering can probably be used to compute the *fuzzy pilot* directly out of the recorded flight data. Therefore the structure of the *fuzzy pilot* can be used to initialise and control the clustering. Attention should be paid to the fact that the projection of the cluster contains an information gap. However it has to be noticed that the data points are dependent on themselves. The data point at the time step t_n is depending on the data point t_{n-1} , etc.

CONCLUSION

The results of the comparison of the *fuzzy pilot* and the human pilot show that

- The *fuzzy pilot* fulfils the requirements of the ITT to hold the aircraft with a fixed target speed on the glide slope.
- The measurements and the control commands of both pilots are very similar in magnitude and trend.
- Control behaviour of the *fuzzy pilot* is based on the control strategy of the human pilot
- The *fuzzy pilot* has primarily the same reaction time as the human pilot.

In this work it could be proven that the developed *fuzzy pilot* uses the same control strategy as the human pilot. It could also be shown that the control commands of the *fuzzy pilot* indicate the same characteristics as those of the pilot. Furthermore the *fuzzy pilot* has in some situation the same reaction time as the pilot so that this aspect of the human control behaviour is also taken into account.

REFERENCES

- Ashkenas, I.L., Hoh, R.H. & Teper, G.L. (1983), Analyses of shuttle orbiter approach and landing, *Journal of Guidance and Control*, 6(6), 448-455.
- Bauschat, J.-M. (2000). An investigation on the dependence of pilot workload and flight simulation fidelity level, *Third International Conference on Engineer-*

- ing Psychology and Cognitive Ergonomics* - Edinburgh October 25-27, Volume 5.
- Budd, H. (1992). *Menschliche Zuverlässigkeit*. Landsberg/Lech: Ecomed.
- Bussolari, S.R. & Lee, P.D.A. (1986). The effects of flight simulator motion on pilot performance and simulator acceptability in transport category aircraft, *Proceedings 2ème Colloque International, La Sécurité Arienne*, 361-371.
- Cooper, G.E. & Harper, R.P. (1969) *The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities*, NASA TN D-5153.
- Duda, H. (1997). Prediction of pilot-in-the-loop oscillations due to rate saturation, *Journal of Guidance, Navigation, and Control*, 20(3).
- Enders, J.H. (1989). *The Human Element—The Key to Safe Civil Operations in Adverse Weather*, Advisory Group of Aerospace Research and Development (AGARD), Conference Proceedings Number 470, K2-1 to K2-7.
- Harper, R.H. (1991). *The Evolution of In-Flight Simulation at Calspan*. CD DGLR-91-05, Paper 91-05-01.
- Höppner, F., Klawonn, F. & Kruse, R. (1997). *Fuzzy-Clusteranalyse*. Braunschweig, Wiesbaden: Vieweg-Verlag (Computational Intelligence).
- Kahlert, J. & Frank, H. (1993). *Fuzzy-Logik und Fuzzy-Control*, Braunschweig, Wiesbaden: Vieweg.
- Kruse, R., Gebhardt, J. & Klawonn, F. (1993). *Fuzzy-Systeme*, Stuttgart: Teubner.
- Mamdani, E.H. (1974) Application of the fuzzy algorithms for control of a simple dynamic plant, *Proceedings of the Institution of Electrical Engineers - Control and Science*, 121(12), 1585-1588.
- McRuer, D.T. (1988). *Pilot Modeling*, AGARD Lecture Series No. 157, Advances in Flying Qualities.
- Schänzer, G. & Krüger, J. (1995). *Delayed Pilot Response in Windshear*, AGARD CP 577, 28-1 to 29-9.
- Zadeh, L.A. (1964). Fuzzy sets, *Information and Control*, (8), 338-353.

Chapter X

Bayesian Agencies in Control

Anet Potgieter and Judith Bishop
University of Pretoria, South Africa

ABSTRACT

Most agent architectures implement autonomous agents that use extensive interaction protocols and social laws to control interactions in order to ensure that the correct behaviors result during run-time. These agents, organized into multi-agent systems in which all agents adhere to predefined interaction protocols, are well suited to the analysis, design and implementation of complex systems in environments where it is possible to predict interactions during the analysis and design phases. In these multi-agent systems, intelligence resides in individual autonomous agents, rather than in the collective behavior of the individual agents. These agents are commonly referred to as “next-generation” or intelligent components, which are difficult to implement using current component-based architectures.

In most distributed environments, such as the Internet, it is not possible to predict interactions during analysis and design. For a complex system to be able to adapt in such an uncertain and non-deterministic environment, we propose the use of agencies, consisting of simple agents, which use probabilistic reasoning to adapt to their environment. Our agents collectively implement distributed Bayesian networks, used by the agencies to control behaviors in response to environmental states. Each agency is responsible for one or more behaviors, and the agencies are structured into heterarchies according to the topology of the underlying Bayesian networks. We refer to our agents and agencies as “Bayesian agents” and “Bayesian agencies.”

Due to the simplicity of the Bayesian agents and the minimal interaction between them, they can be implemented as reusable components using any current component-based architecture. We implemented prototype Bayesian agencies using Sun's Enterprise JavaBeans™ component architecture.

INTRODUCTION

For a system to exhibit computational intelligence, it must be able to learn from and adapt to changes in its environment. Most distributed environments are characterized by uncertainty and non-determinism. Bayesian networks provide the ideal mechanism for systems inhabiting environments such as these, to learn from, reason about and adapt to changes in their environment. Our research focuses on the implementation of distributed Bayesian networks using simple agents organized into agencies. These agencies are structured into heterarchies according to the structure of the Bayesian networks that they collectively implement. Each agency is responsible for one or more behaviors. We call these agents and agencies “Bayesian agents” and “Bayesian agencies.”

This chapter is organized as follows: it begins by giving a background on the underlying technologies that we use in our research. We define agents, agencies, heterarchies, intelligence and artificial life. We further describe Bayesian networks, Bayesian belief propagation and Bayesian learning algorithms.

Next we describe how these agencies collectively adapt to environmental states using a simple Web personalization example. We further describe emergent belief propagation in the Bayesian agencies and a prototype implementation thereof using a component-based architecture.

Then we describe future research and in finally we give our conclusion.

BACKGROUND

Agents, Agencies and Heterarchies

There are two different approaches to the definition of the concepts of agents in the research community. In the first (most popular) approach, agents are autonomous entities as reflected in the following definition:

An agent is an encapsulated computer system situated in some environment and capable of flexible, autonomous action in that environment in order to meet its design objectives (Jennings, 2001).

In the second approach, which was started by Minsky (1988), simple, unintelligent agents are organized into agencies, which in turn can be organized into

hierarchies or heterarchies. The intelligence lies in the behavior of the agencies and not the individual agents. The intelligence of the agencies emerges from interactions between the simple agents and the environment. Agencies organized into a heterarchy can achieve more than agencies organized into a hierarchy (Minsky, 1988). We view a hierarchy as a simplified heterarchy.

Interaction is at the core of emergence and emergence causes intelligence (Brooks, 1991). Most autonomous agents do not exhibit emergent behavior, as the emergence is restricted by complex interaction protocols (Wooldridge, Jennings & Kinny, 2000) and social laws that agents must follow (Zambonelli, Jennings, Omicini & Wooldridge, 2000). Autonomous agents that do react to their environment using emergent behavior include the Subsumption Architecture (Brooks, 1985) and autonomous agents controlled by Behavior Networks (Maes, 1989). These agents consist of simple (sub)agents and their behavior emerges from interactions between the simple (sub)agents and the environment. The (sub)agents in Brooks' Subsumption Architecture are task-accomplishing behaviors, and Maes implemented behaviors as simple (sub)agents organized into single autonomous agents using Behavior Networks.

Behavior Networks consist of nodes, representing behaviors, linked together by causal links. Each node represents the selection of a particular behavior as an emergent property of an underlying process. The behaviors are simple (sub)agents that activate and inhibit each other according to causal links. During execution, activation accumulates in the nodes that represent the best actions to take, given the current environmental states and the global goals of the autonomous agent (Maes, 1989).

Intelligence

Traditional artificial intelligence attempts to model the world, and then reason about it using complex control strategies. The control strategies can either be centralized or distributed. The traditional distributed control strategies are very complex and have high communication overhead.

Brooks (1991) started a new trend in Artificial Intelligence with his key ideas of "situatedness, embodiment, intelligence and emergence." According to Brooks, the world is its own best model, and it cannot be modeled successfully. An agent must be situated in the world and must react to inputs from the world rather than attempt to model the world internally. Situatedness can be described as the state of being in the world. Embodiment is the method by which situatedness is achieved, namely to be in the world and to react to inputs from the world. Intelligence and emergence are tightly interwoven—intelligence emerges out of interactions between behaviors and the environment.

Artificial Life

Artificial life is often described as attempting to understand high-level behavior from low-level rules (Liekens, 2000) in systems such as ant colonies, swarms, flocks of birds and schools of fish. In these systems, coherent behavior emerges from the interactions between the individual organisms in order to collectively learn from and adapt to their environments. The basic laws governing emergent behavior in nature can be used to achieve computational intelligence. As an example, Dorigo, Di Caro and Gambardella (1999) developed algorithms based on an ant colony for collective optimization.

An ant colony is an example in nature of an agency consisting of simple agents. The ants (simple agents) collectively reason about the state of their environment and food sources using pheromone trails. Ants are capable of finding a shortest path from a food source to the nest given a set of external constraints. The variables of interest in their problem domain are the food sources, the nest and obstacles, linked together by different paths between the different food sources and the nest. The constraints are obstacles that the ants might encounter along the way. Ants “learn” and maintain the shortest path between the nest and a food source, given certain obstacles, by using pheromone trails. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to choose a path rich in pheromone rather than a poorer one (Dorigo et al., 1999). Only through collective behavior do ants achieve their global goal namely to gather food using the shortest paths between the food sources and the nest. This is an example of collective probabilistic reasoning found in nature.

Bayesian networks provide the ideal mechanism for collective probabilistic reasoning in computational systems. In the next section, we will give a background on Bayesian networks, belief propagation and learning.

Bayesian Networks

Bayesian networks provide a powerful technology for probabilistic reasoning and statistical inference that can be used to reason in uncertain environments. These networks are “direct representations of the world, not of reasoning processes” (Pearl & Russel, 2000). A Bayesian network (BN) is a directed acyclic graph that represents a set of random variables in an application domain. The nodes represent variables of interest. The links represent informational or causal dependencies among the variables. The dependencies are given in terms of conditional probabilities of states that a node can have given the values of the parent nodes (Dechter, 1996; Pearl & Russel, 2000). Each probability reflects a degree of belief rather than a frequency of occurrence.

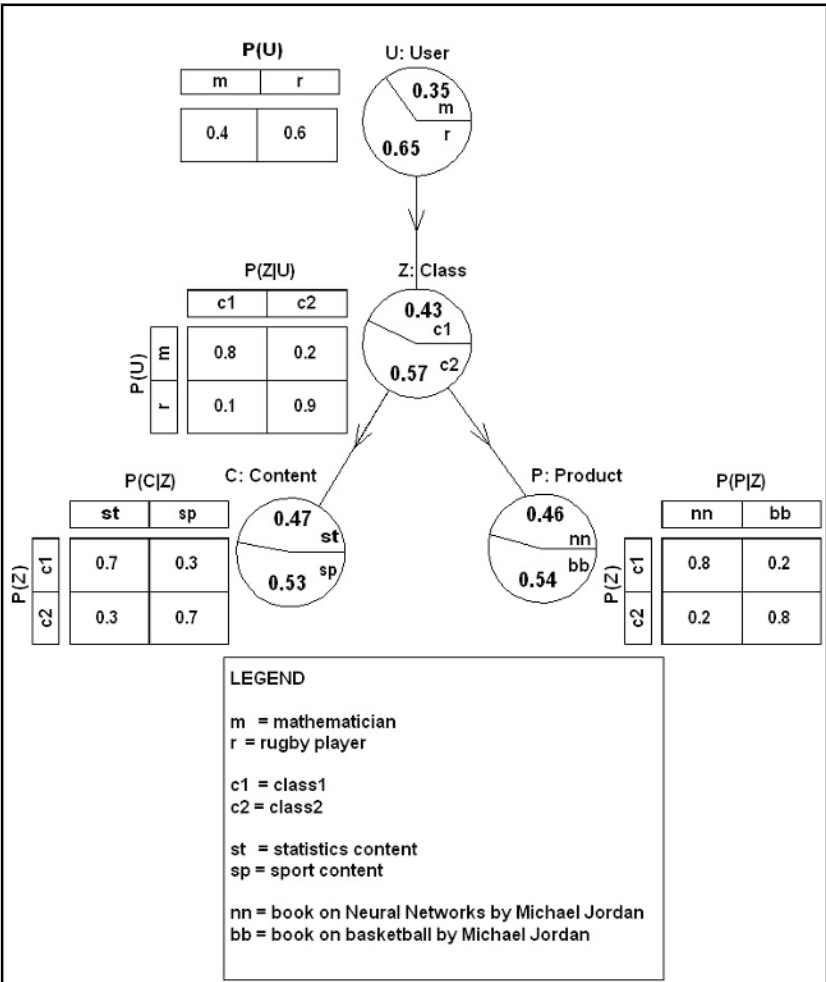
Dechter (1996) uses the following notation: Let $X = \{X_1, \dots, X_n\}$ be a set of random variables. A BN is a pair (G, P) where G is a directed acyclic graph and

$P = \{P_i\}$. Each P_i is the conditional probability matrix associated with X_i where $P_i = \{P(X_i | pa(X_i))\}$ and where $pa(X_i)$ represents the parents of X_i . An assignment $(X_1 = x_1, \dots, X_n = x_n)$ can be abbreviated to $x = (x_1, \dots, x_n)$. The BN represents a global joint probability distribution over X having the product form

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | pa(x_i)) \tag{1}$$

Figure 1 illustrates a Bayesian network based on the user-words aspect model proposed by Popescul, Ungar, Pennock and Lawrence (2001). This network models the relationship between users (U), the contents of browsed Web pages characterized in terms of concepts (C) and products bought from these pages (P).

Figure 1: Bayesian network



This simple model includes three-way co-occurrence data among two users, two products and two concepts.

The users are represented by $u \in U = \{\textit{mathematician}(m), \textit{rugby player}(r)\}$, the products by $p \in P = \{\textit{book authored by Michael Jordan on neural networks}(nn), \textit{book authored by Michael Jordan on basketball}(bb)\}$ and the concepts inferred from the Web pages the users viewed by $c \in C = \{\textit{statistics}(st), \textit{sport}(sp)\}$. The users (U), products (P) and concepts (C) form observations (u, c, p), which are associated with a latent variable class (Z). This simple model has two latent classes, $z \in Z = \{\textit{class1}(c1), \textit{class2}(c2)\}$. In Figure 1, the conditional probability matrices are shown next to their nodes.

The example Bayesian network represents the joint distribution:

$$P(u, z, c, p) = P(u)P(z | u)P(c | z)P(p | z) \quad (2)$$

From (1) and (2) it can be seen that the global distribution is described in terms of local distributions. Pearl and Russel (2000) refer to equation (1) as the “global semantics” of a Bayesian network, and further describe the “local semantics,” which asserts that each variable is independent of its nondescendants in the network given its parents. For example, in Figure 1:

$$P(c | u, z) = P(c | z) \quad (3)$$

The localized nature of Bayesian networks as well as its local semantics makes this technology ideal for distributed implementation.

Using Bayes rule, an equivalent joint distribution for (2) is given by:

$$P(u, z, c, p) = P(z)P(u | z)P(c | z)P(p | z) \quad (4)$$

The joint distribution over users, contents and products is given by:

$$P(u, c, p) = \sum_z P(z)P(u | z)P(c | z)P(p | z) \quad (5)$$

In a changing environment, some variables can have values that change over time. In dynamic Bayesian networks, multiple copies of the variables are represented, one for each time step (Pearl & Russel, 2000).

Belief Propagation

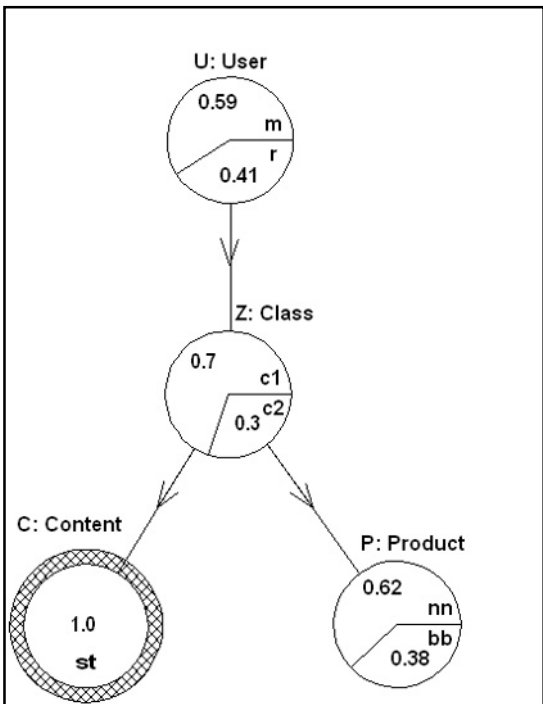
Belief propagation is the process of finding the most probable explanation (MPE) in the presence of evidence (e). Dechter (1996) defines the MPE as the maximum assignment x in:

$$\max_x P(x) = \max_x \prod_{i=1}^n P(x_i | pa(x_i), e) \tag{6}$$

Figure 2 illustrates the results of belief propagation in the presence of evidence. Node C, the evidence node, is circled. The new beliefs updated during belief propagation are indicated on nodes P, Z and U. In the presence of the evidence, namely that a user is interested in statistical concepts, the probability that he will be interested in a book on neural networks authored by professor Michael Jordan rises from 0.46 to 0.62.

Belief propagation is NP-hard (Pearl & Russel, 2000; Dechter, 1996). Judea Pearl developed a belief propagation algorithm for tree-structured Bayesian networks (Carnegie Mellon University, 1991). This algorithm was extended to general multi-connected networks by different researchers. Pearl and Russel

Figure 2: Belief propagation in the presence of evidence



describe three main approaches, namely cycle-cutset conditioning, join-tree propagation (also called the tree-clustering approach) and variable elimination. Cycle-cutset conditioning and join-tree propagation work well for sparse networks with small cycle-cutsets or clusters (Dechter, 1996). Variable elimination is based on non-serial dynamic programming algorithms (Dechter, 1996), which suffer from exponential space and exponential time difficulties. Dechter combined elimination and conditioning in order to address the problems associated with dynamic programming.

The belief propagation algorithms for general multi-connected networks generally have two phases of execution. In the first phase, a secondary tree is constructed. This can for example be a “good” cycle-cutset used during conditioning (Becker et al., 2000) or an optimal junction tree used by tree-clustering algorithms (Jensen, Jensen & Dittmer, 1994). In the second phase, the secondary tree structure is used for inference. Finding a “good” cycle-cutset is NP-complete (Becker et al.) and finding an optimal junction tree is also NP-complete (Becker & Geiger, 1996). A number of approximation algorithms for the secondary trees were developed, as in Becker et al. and Becker and Geiger (1996).

Diez (1996) describes a local conditioning algorithm that uses the original Bayesian network during belief propagation and detects loops using the DFS (Depth-First Search) algorithm.

Bayesian Learning

Bayesian learning can be viewed as finding the local maxima on the likelihood surface defined by the Bayesian network variables (Russel, Binder, Koller & Kanazawa, 1995). Assume that the network must be trained from D , a set of data cases D_1, \dots, D_m generated independently from some underlying distribution. In each data case, values are given for some subset of the variables; this subset may differ from case to case—in other words, the data can be incomplete. Russel et al. (1995) describe the learning task as the calculation of the parameters ω of the conditional probability matrices that best model the data. If the assumption is made that each setting of ω is equally likely a priori, the maximum likelihood model is appropriate. This means that $P_{\omega}(D)$ must be maximized. In other words, the probability assigned by the network to the observed data when the parameters of the conditional probability matrices are set to ω must be maximized.

Examples of learning algorithms include a local gradient-descent learning algorithm (Russel et al.) and the EM algorithm. Popescul et al. (2001) illustrated the EM algorithm applied to their three-way aspect model. Applied to our three-way aspect model, these calculations will change as follows:

Let $n(u, c, p)$ be the number of times that a user u , interested in concepts c , bought product p . This can be calculated from $n(u, c, p) = n(u, c) \times n(c, p)$, where

$n(u, c)$ is the number of times that a user u accessed Web pages containing concepts c and $n(c, p)$ = the number of times product p was bought by users interested in concepts c . In the EM algorithm, a local maximum is found for the log likelihood L of the data (Popescul et al., 2001), which is:

$$L = \sum_{u, c, p} n(u, c, p) \log P(u, c, p) \quad (7)$$

where $P(u, c, p)$ is given by equation (5).

BAYESIAN AGENCIES

Collectively Adapting to Changes in Uncertain Environments

Computational intelligence emerges from the interactions between agents that collectively learn from, reason about and adapt to changing environmental states. Most agent architectures implement autonomous agents that use extensive interaction protocols and social laws to control interactions in order to ensure that the correct behaviors result during runtime. These agents, organized into multi-agent systems in which agents adhere to pre-defined interaction protocols, are well suited to the analysis, design and implementation of complex systems in environments where interactions can be predicted. As the intelligence resides in the individual agents, autonomous agents are commonly referred to as “next-generation” or intelligent components, which are difficult to implement using current component-based architectures.

We believe that the only way to adapt to changes in uncertain environments is through emergent behavior as in Minsky (1988), Maes (1989) and Brooks (1991). Our work is based on Minsky’s concepts of agents, agencies and heterarchies. We adapted Minsky’s definition of agents, agencies and heterarchies as follows: an agency consists of a society of agents that inhabit some complex dynamic environment, where the agents collectively sense and act in this environment so that the agency accomplishes what its composite agents set out to accomplish by interacting with each other. If agents in a society belong to more than one agency, the set of “overlapping” agencies forms a heterarchy. Collective intelligence of a heterarchy emerges through the interaction of the agents within the agencies in the heterarchy.

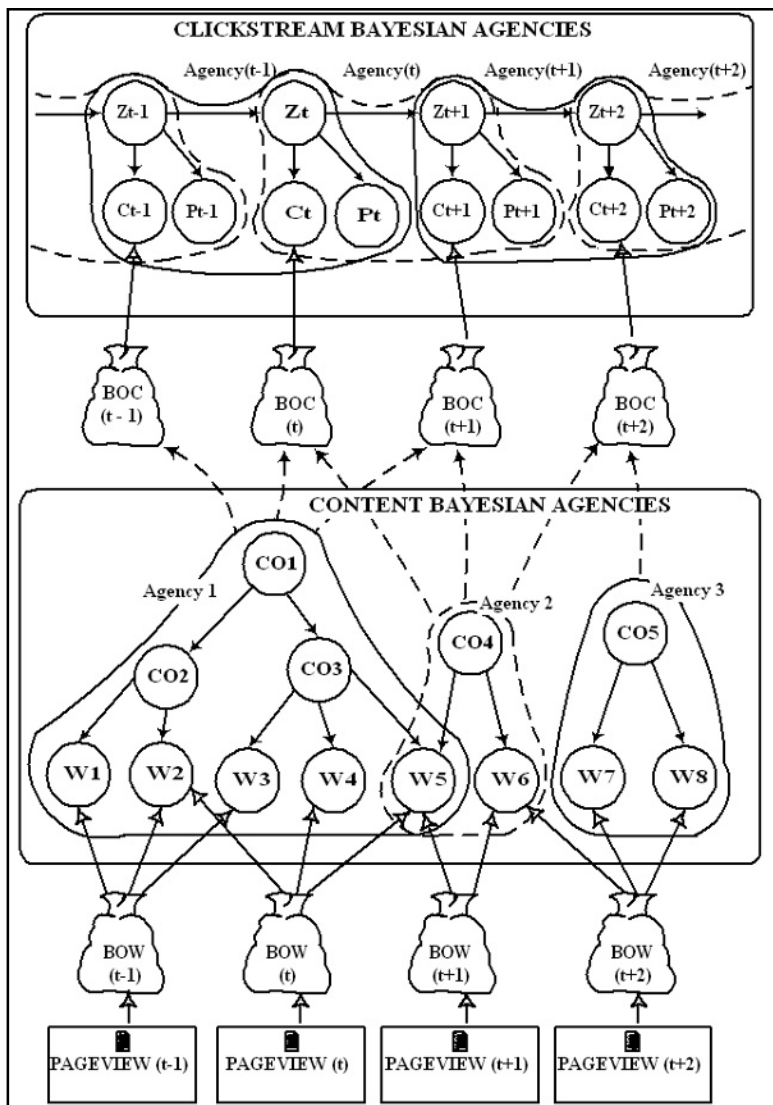
Our agencies collectively implement distributed Bayesian networks in order to control behaviors in uncertain environments. These agencies are structured into heterarchies according to the topology of the underlying Bayesian networks

collectively implemented by them. We refer to our agents and agencies as “Bayesian agents” and “Bayesian agencies.”

In the Behavior Networks defined by Maes (1989), each node determines the activation of a particular behavior as an emergent property of an underlying process. In a similar way, each Bayesian agency determines the activation of one or more behaviors as an emergent property of belief propagation in the subtree of that agency.

Figure 3 illustrates the use of Bayesian agencies in a simplified Web personalization application. There are two sets of Bayesian agencies in this

Figure 3. Bayesian agencies in Web personalization



example, namely the clickstream and the content Bayesian agencies. The clickstream Bayesian agencies collectively implement a dynamic Bayesian network, modeling a two-way contents-product aspect model at each time step. The content Bayesian agencies collectively implement a Bayesian network that models a hierarchical concept model, representing the relationships between words extracted from Web pages and higher-level concepts, at different levels of abstraction.

During time step t , a bag of words, $BOW(t) = \{w_2, w_4, w_5\}$, was extracted from the $PageView(t)$ that a user browsed. $BOW(t)$ was then presented to the content Bayesian agencies that collectively reduced the dimensionality of the words to a bag of concepts $BOC(t) = \{co_1, co_4\}$. Content agency 1 added concept co_1 to $BOC(t)$ and content agency 2 added co_4 to $BOC(t)$ because their beliefs exceeded a certain threshold as a result of belief propagation. Each BOC is “filled” through the emergent behavior of the content agencies. Clickstream agency(t) used $BOC(t)$ as evidence together with products purchased from $PageView(t)$ to predict the contents and products that might interest the user next. The behavior associated with clickstream agency(t) is the personalization of $PageView(t+1)$. $BOW(t+1)$ is extracted from the personalized $PageView(t+1)$ viewed by the user. This process is repeated until the session ends.

Emergent Belief Propagation Using Bayesian Agencies

Emergent belief propagation is the collective behavior of the Bayesian agents, while collectively solving the joint probability distribution of the Bayesian network distributed between them in response to evidence received from the environment.

Every link in the Bayesian network is managed by a Bayesian agent. These agents are organized into agencies, where each agency is responsible for one or more behaviors. Each Bayesian agency implements a subtree of the underlying Bayesian network. The agencies are structured into heterarchies. For example, in Figure 3, the clickstream agencies form a heterarchy, and the concept agencies form a hierarchy (a simple heterarchy).

Belief propagation is localized within the Bayesian agencies. Bayesian agents can only propagate beliefs between themselves and their direct neighbors along links defined by the underlying subtree for that agency. These agents cooperate to find the local MPE of the subtree of the agency it belongs to by communicating λ and Π messages amongst themselves as in Judea Pearl’s belief propagation algorithm (Carnegie Mellon University, 1991). Within an agency, each agent communicates a λ message on each of the incoming links of its parent nodes, and a Π message on each of the outgoing links of its child node. An agent that belongs to more than one agency must contribute to the local MPEs of all the agencies it

belongs to in such a way that the maximum local MPE of each agency is found. Collectively the agents in a heterarchy of agencies will maximize the global MPE of the underlying Bayesian network.

In our current implementation the belief propagation is performed on the original network and loops are handled using local conditioning as described by Diez (1996).

A Component-Based Approach to Emergent Belief Propagation in Bayesian Agencies

The simplicity of the Bayesian agents as well as the minimal interaction between them allowed us to implement them as re-usable components using Sun's Enterprise JavaBeans™ component architecture. In our prototype implementation, the Bayesian agents are implemented using message beans and the links are implemented using JMS queues. The topology of the Bayesian networks, the subtrees, the conditional probability matrices and the beliefs are maintained in a database and administrated by entity beans.

FUTURE RESEARCH

Future research will include the refinement of belief propagation by replacing the DFS algorithm with the self-organization of agents to handle loops in networks. Incremental emergent learning from evidence received from the environment will also form a very important part of our future research. This emergent Bayesian learning will enable the Bayesian agents to collectively “discover” or “mine” relationships in data and to self-organize according to the evolving topology of the Bayesian network.

CONCLUSION

We have shown how Bayesian agencies, based on artificial life principles, can control behaviors in uncertain environments. These agencies consist of simple agents that collectively implement distributed Bayesian networks, which the agencies use to control behaviors in response to environmental states. With our prototype implementation, using Sun's Enterprise JavaBeans™ component architecture, we have shown that due to the simplicity of the Bayesian agents and the minimal interaction between them, they can be implemented as re-usable components using any commercially available component architecture.

REFERENCES

- Becker, A. & Geiger, A. (1996). *A Sufficiently Fast Algorithm for Finding Close to Optimal Junction Trees*. Retrieved March 8, 2001, <http://www.cs.technion.ac.il/~dang/>.
- Becker, A. Bar-Yehuda, R. & Geiger, D. (2000). Randomized algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research*, 12, 219-234. Retrieved March 7, 2001, <http://www.cs.washington.edu/research/jair/abstracts/becker00a.html>.
- Brooks, R. A. (1985). *A Robust Layered Control System for a Mobile Robot [MIT AI Memo 864]*. Retrieved July 18, 2000, <http://www.ai.mit.edu/people/brooks/papers.html>.
- Brooks, R. A. (1991). *Intelligence Without Reason [MIT AI Memo 1293]*. Retrieved July 18, 2000, <http://www.ai.mit.edu/people/brooks/papers.html>.
- Carnegie Mellon University. (1991). *BAYES: Tree-Structured Bayesian Belief Network*. Retrieved May 5, 2001, <http://www.cs.cmu.edu/~mkant/Public/util/areas/reasonng/probabl/bayes/bayes.aug>.
- Dechter, R. (1996). *Bucket Elimination: A Unifying Framework for Probabilistic Inference [UAI96]*. Retrieved October 8, 2000, <http://www.ics.uci.edu/~dechter/publications/>.
- Diez, F. J. (1996). Local conditioning in Bayesian networks. *Artificial Intelligence*, 87, 1-20. Retrieved January 17, 2001, <http://www.dia.uned.es/~fjdiez>.
- Dorigo, M., Di Caro, G. & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, 5(2), 137-172. Retrieved May 5, 2001, <http://iridia.ulb.ac.be/~mdorigo/ACO/TSP>.
- Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4), 35-41.
- Jennings, N. R. & Wooldridge, M. (2000). *Agent-Oriented Software Engineering [Handbook of Agent Technology]*. Retrieved January 23, 2001, <http://www.elec.qmw.ac.uk/dai/pubs/#2000>.
- Jensen, F., Jensen, F. V. & Dittmer, S. L. (1994). *From Influence Diagrams to Junction Trees [Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence]*. Retrieved February 13, 2001, <http://www.cs.auc.dk/research/DSS/abstracts/jencen:jensen:dittmer:94.html>.
- Lieken, A. (2000). *Artificial Life?* Retrieved March 20, 2001, <http://alife.org/index.php?page=alife&context=alife>.
- Maes, P. (1989). *How to Do the Right Thing [MIT AI Memo 1180]*. Retrieved July 18, 2000, <http://agents.www.media.mit.edu/groups/agents/publications>.
- Minsky, M. (1988). *The Society of Mind* (First Touchstone ed.). New York: Simon & Schuster Inc.

- Pearl, J. & Russel, S. (2000). *Bayesian Networks*. Retrieved May 5, 2001, http://bayes.cs.ucla.edu/csl_papers.html.
- Popescul, A., Ungar, L. H., Pennock, D. M. & Lawrence, S. (2001). *Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments*. Retrieved January 28, 2002, <http://www.cis.upenn.edu/~popescul/publications.html>.
- Russel, S. J., Binder, J., Koller, D. & Kanazawa, K. (1995). *Local Learning in Probabilistic Networks with Hidden Variables [Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence]*. Retrieved September 19, 2000, <http://robotics.stanford.edu/~koller/papers/apnijcai.html>.
- Wooldridge, M., Jennings, N. R. & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3), 15. Retrieved December 17, 2000, <http://www.ecs.soton.ac.uk/~nrj/pubs.html#1998>.
- Zambonelli, F., Jennings, N. R., Omicini, A. & Wooldridge, M. (2000). *Agent-Oriented Software Engineering for Internet Applications [Coordination of Internet Agents]*. Retrieved January 23, 2001, <http://www.csc.liv.ac.uk/~mjw/pubs/>.

SECTION IV:

**MACHINE
LEARNING,
EVOLUTIONARY
OPTIMISATION
AND
INFORMATION
RETRIEVAL**

Chapter XI

Simulation Model for the Control of Olive Fly *Bactrocera Oleae* Using Artificial Life Technique

Hongfei Gong and Agostinho Claudio da Rosa
LaSEEB-ISR, Portugal

ABSTRACT

*In this chapter we present a novel method for modelling of the development of olive fly—*Bactrocera oleae* (Gmelin)—based on artificial life technique. The fly's artificial life model consists of a set of distinct agents, each representing one phase in the insect's lifecycle. Each agent is defined mainly by two internal state variables: health and development. Simulation results have provided development times and mortality rates that closely resemble those observations in biological experiment. The model presented has proven to offer good results in replicating the insect's behaviour under monitored climatic conditions. The model's potential uses are discussed.*

INTRODUCTION

Pest management and control, as it is readily understood, are vital to a sustained agricultural production since, without it, long-term reliable income cannot

be ensured. Nowadays chemical protection is the most widely used method for pest control. However, control methods relying on the use of chemical products pose a health risk for man and animals, unnecessary treatments increase production costs, cause more environmental pollution and can lead to the development of resistance to pesticide. Therefore, it becomes more and more important and necessary to know or estimate the state of pest population, because if the control action is applied at the correct moment, a reduced number of pesticide treatments can achieve the same level of pest control.

Olive growing is an important activity for the economic, social and ecological well-being of the Mediterranean region. It represents a relatively cheap source of high quality vegetable fat, and its importance spans the areas of agriculture and food industry. In Portugal this crop represents a significant proportion of the total agricultural production. The olive fly, *Bactrocera oleae*, is generally considered the most damaging of the insect pests that attack the olive trees. Its attacks may potentially account for 50-60% of the total insect pest damage, causing a reduction in the number and/or size of the fruits, with a subsequent reduction in yield and quality of the fruit and oil (Michelakis, 1986; Bento, 1999).

Simulation models have been introduced as a way to assess its current state of pest population and estimate the risk based on climatic data, especially in some cases, the ideal timing for treating a crop is a certain stage of the infestation's lifecycle that is not easily detectable in the field. The quality of a decision support tool concerning the timing and kind of crop-protection actions are highly dependent on the effectiveness of the simulation model used to assess and forecast the development of crop pests.

BACKGROUND

Traditional mathematical and statistical population dynamics analysis methods may satisfactorily reproduce the observed behavior. Most of these models aim at describing the evolution of the parasite population as a whole, using statistical interpolation or differential equations methods, in order to find a set of equations and parameters that correctly fit the available test data. The main problems of this approach are the lack of biological significance of the resulting systems and the difficulty in testing the resulting models, requiring extensive periods of climatic and biological data to increase the confidence in the system. Most of the time, it always fails to establish a correspondence between its low-level causes and the macroscopic parameters involved in the model (Pitzalis, 1984; Croveti, 1982; Dicola & Croveti, 1984).

Furthermore, using a single simulation model as the sole indicator for crop control decision should be avoided; combination of both traditional assessment

methods and other models will increase the necessary confidence level of the decisions to be made. In addition, the interaction between the different players may only be captured by complex and non-linear models that are difficult to manipulate, integrate or optimize.

In recent years, availability of affordable computational power has allowed the appearance of new approaches to ecological system analysis. One such approach is artificial life. Ecological systems, by nature, are composed by a set of interdependent entities, namely living and non-living beings. A typical artificial life approach to studying such systems consists of creating artificial beings, or agents, that resemble as closely as possible the real beings that live in the natural ecological system. Also, lower-level entities are modeled explicitly and interact freely with each other (Langton, 1992; Noble, 1997). Whole population behavior emerges as a result of the free interaction of the agents in the artificial environment. For a good reference on emergence of order in complex systems, see the work of Stuart Kauffman (1993), which gives a rich and compelling picture of the principle of self-organization and selection in evolution.

An artificial life modeling approach addresses the problem by focusing efforts in individual agent modeling, incorporating information on each of the system's constituents and the laws through which they interact. The modeling are bottom-up explicit simulations of basic players of the target ecosystem, like plants, fungus, insects or animals, left to evolve in an artificial environment fed with climatic and biological data. Artificial life modeling of different biological systems has been described in recent years, such as with flocks of bird (Craig, 1992), schools of fish (Craig, 1992; Terzopoulos, 1996), basic cells (Carnahan, 1997), and forests (Lange, 1998; Wilber & Shapiro, 1997), with success. These examples are based on cellular automata (von Neumann, 1966), proven to be a very powerful concept for the simulation of local interaction.

As for the application to pest control, an epidemiological model for the asexual phase of grapevine Downy Mildew, *Plasmopara viticola*, has been developed using artificial life technique. Downy Mildew is one of the most destructive diseases of grapevines occurring in most grape-growing areas of the world, with the reduction in yield and the quality of the fruit and wine.

The model simulates the evolution of a population of artificial fungi in artificial environment vineyard conditions (Costa & Rosa, 1997). Detection of primary infections is one of the most important goals of an effective control of Downy Mildew by fungicides. The initial peak observed in the artificial germinated oospore curve (oospore_gm in Figure 1) by the end of April indicates the possibility of primary infection occurrence once favorable climatic conditions are present. Fungicide application before this time would be premature, useless and potentially harmful for auxiliary fauna.

Figure 1: Artificial oospores population

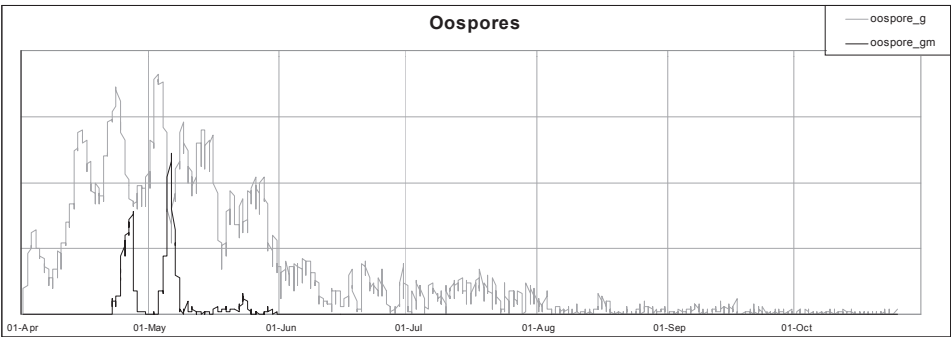
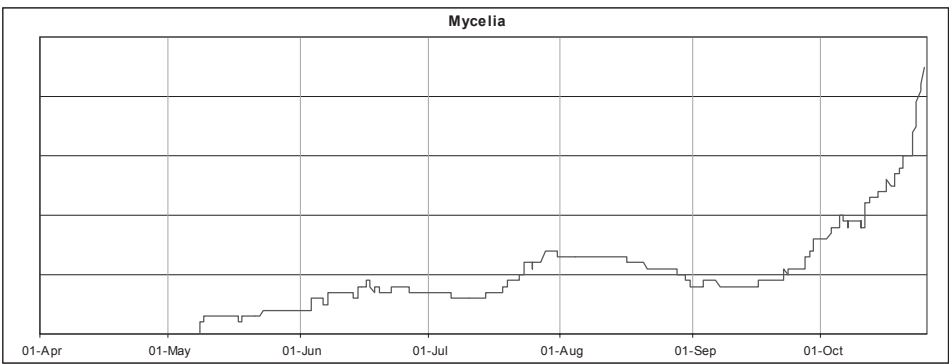


Figure 2: Artificial mycelia



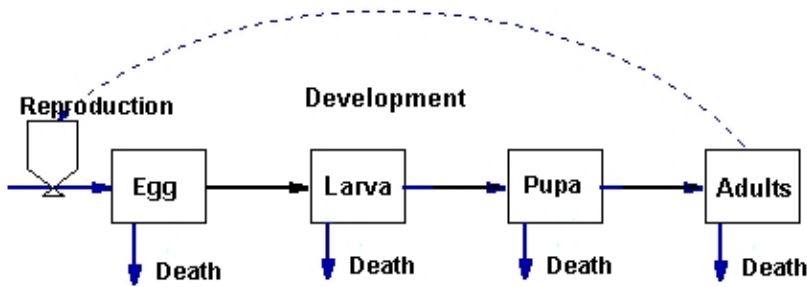
As shown in Figure 2, primary infections, shown by the appearance of artificial mycelia, actually occurred shortly after the germination of the first artificial oospore, due to favorable climatic conditions (temperature above 10°C and rainfall above 10mm). In reality the mycelia will develop inside of the leaf tissue, which is not easy to detect. The visible lesions will only appear due to the destruction of the cellular leaf tissue and the germination of oospore; when this happens, the fungicide treatment can only limit the damage.

MAIN THRUST OF THE CHAPTER

The Olive Fly’s Lifecycle

The olive fly’s lifecycle is composed of four distinct phases: egg, larva, pupa and adult (Figure 3). The first three phases are called pre-imaginary phases. The first two phases must develop inside the olive fruit. Also, it is important to consider that the pupal phase is the one that better resists low temperature values in all pre-imaginary phases.

Figure 3: Olive fly's lifecycle



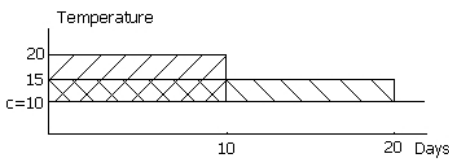
Adult insects must be considered separately, depending on their sex. Female adults pass through three stages: pre-oviposition, oviposition and post-oviposition. Male adults pass through a premature stage before reaching sexual maturity.

Heat Unit Accumulation Concept

The heat unit accumulation concept, also referred to as the degree-day method, is widely used in agronomy to express the relationship between temperature values submitted to a plant and its development time. And it is applicable to insects.

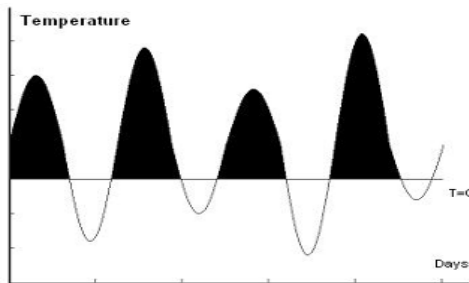
At a constant temperature, the degree-days is formulated as: $D^{\circ} = T(y - c)$; the insect needs to accumulate certain degree days to evolve into the next phase. When the degree-days reach Thermal Constant K , D° is the physiological time (degree-days) required to complete development of a stage, T is the number of days required

Figure 4: Illustration of the degree-days under a) constant temperatures; b) variable temperatures



$$D^{\circ} = 10(20 - 10) = 100$$

$$D^{\circ} = 20(15 - 10) = 100$$



$$D^{\circ} = \sum_{t=1}^T \Delta D(t) = 100$$

Table 1: The thermal constant for the olive fly's phases

Table 1	Egg to Adult	Egg	Larva	Pupa
Thermal Constant (D°)	375.03	48.66	125.3	201.7

to complete development at temperature y , and c is the threshold temperature, below which no development occurs. In other words, degree-days is the product of time and the temperature degree above the threshold temperature. The computation for degree-days under constant temperature is illustrated in Figure 4.a.

The theory is easily extended to fluctuating temperature as indicated in Figure 4.b, which shows curve of daily temperature and the daily integral of $\Delta D(t)$ above C . The method to compute degree-day is to integrate the area under the curve above C , which is formulated as:

$$D^{\circ} = \sum_{t=1}^T \Delta D(t)$$

As a poikilothermic animal, lacking an efficient body temperature control mechanism, the insect would be expected to complete its development when the sum of D° reach its thermal constant. This is the simplest and most widely used method for predicting the physiological age and time for populations of poikilothermic organisms.

Table 1 shows each phase's thermal constant of olive fly is expressed in D° , which stands for degree-days. In order to evolve into the next phase, the insect needs to accumulate the amount of energy during its development (Crovetti, 1982).

Description of the Model

The fly's artificial life model consists of a set of distinct agents, each representing one phase in the insect's lifecycle. In order to represent the insect throughout its lifecycle, seven different agents must be created, each replicating the corresponding phase's behaviour: egg, larva, pupa, male adult, female adult in the pre-oviposition, oviposition and post-oviposition periods.

Oviposition of a new egg in reality will correspond to the creation of a new egg agent. An insect's transition from one phase to another corresponds to the creation of a new agent depicting the insect in the following phase and the destruction of the agent that represented the insect in the completed phase.

At a certain instant, each agent is defined by its two internal state variables: Health (H) and Development (D). These state variables bear a concrete

correspondence to the insect's condition: H encapsulates the insect's general physical condition, and D represents its accumulated energy, using the degree-day concept above mentioned.

On agent creation, the H variable starts with an optimum health value. If by any reason it reaches zero, then the artificial insect dies. The artificial insect's D variable starts with a zero value at the beginning of a certain phase, and will rise until it reaches the phase's thermal constant. At that moment, the insect will have accumulated enough energy to pass unto the next phase of its cycle. The agent that represented the insect in the passed phase will be erased and a new one created, representing the insect in the following phase, with convenient initial H value and zero D value, meaning that it has not yet accumulated any energy.

State changing rules are explained if one follows the steps taken in one iteration. Each agent of the artificial insect will be left to evolve in a series of consecutive iterations, which is a certain time period that provides new, different temperature and relative humidity values. Figure 5 depicts the main steps taken in one iteration.

Firstly, the set of $(Temperature, Relative Humidity) = (T, RH)$ values for the present iteration will be evaluated through a function *Damage*, a $R^2 \rightarrow R$ function that establishes a correspondence between temperature and relative humidity values and the amount of damage, which will affect the artificial insect's H counter (Step 1) on each phase. The *Damage* function was synthesized from consideration and experimental data of the available literature. The approximate shape of the

Figure 5: Iteration scheme

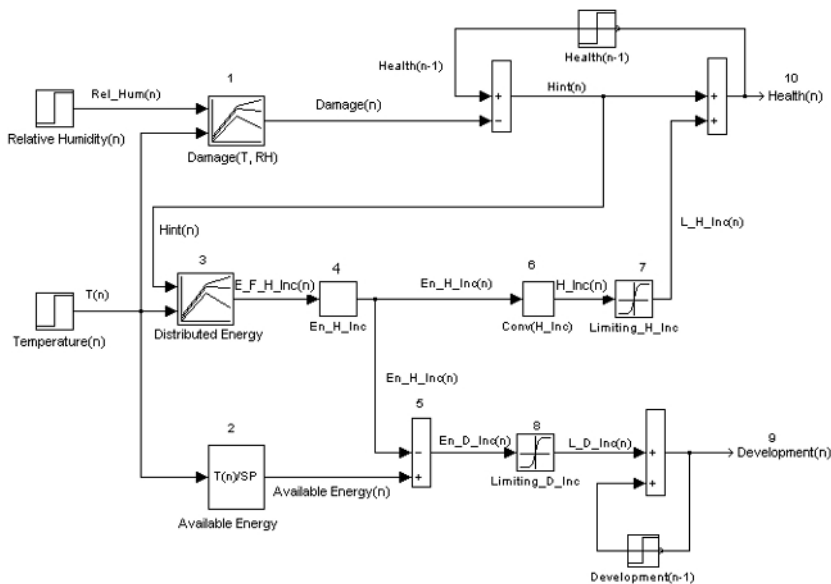
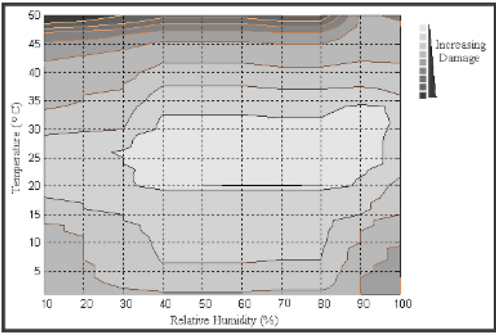


Table 2: Description of the variables in the model

H_{int}	An intermediate H value to evaluate whether or not the insect has sustained lethal damage and dies
Available_Energy	Energy available to insect at that iteration; Energy is measured in degree-days
Distribute_Energy	Function will calculate the fraction of <i>Available Energy</i> used in health restoration
En_F_H_inc	Energy fraction used in Health increment
En_H_inc	Amount of Energy used in Health restoration
En_D_inc	The remaining Energy used in Development
Conv(En_H_inc)	Energy to Health conversion function
H_inc	Health increment from the Energy

Figure 6: Function Damage (T, RH)



function presents in Figure 6. It is visible that lower damage regions in the (T, RH) plane correspond to optimum climatic conditions, and higher damage correspond to extreme climatic conditions.

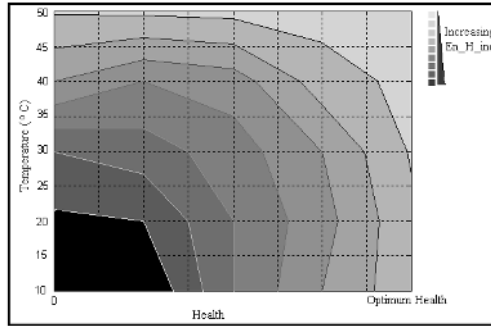
One calculates, thus, an intermediate H value, H_{int} (Step 1a). It is at this moment that the H_{int} is checked to evaluate whether or not the insect has sustained lethal damage and dies.

$$H_{int} = H - \text{Damage}(T, RH)$$

The following step is to calculate the energy available to the insect at that iteration (Step 2). Energy is measured in heat units, namely degree-days.

$$\text{Available_Energy} = T / S_p$$

Figure 7: *Distribute_Energy* (H_{int} , T)



Where T is the iteration's temperature value and Sampling period $S_p = 24$ /the real time span for one iteration.

Note that the temperature value T is used as a mean temperature value for the time represented by one iteration, and that it has not been subtracted from the zero development temperature value, c , as in the calculation of the above mentioned modified average temperature. This means that the artificial insect will still receive some small amount of energy even if temperatures fall below c . Null development rates observed at those temperatures will be achieved by the simultaneous effects of high damage values and small available energy.

It is logical that upon being damaged, the insect will react by trying to repair the inflicted damage, consuming some of the available energy. How much energy is used in health restoration, that is, in increasing the H variable value, is decided by calculating the value of *Distribute_Energy* function, also a $R^2 \rightarrow R$ function that has input variable (H_{int} , T), shown in Figure 7. This function will calculate the fraction of *Available Energy* that will be used to restore health. (Step 3):

$$En_F_H_inc = Distribute_Energy(H_{int}, T)$$

The following expression, En_H_inc , determines the amount of Energy used in Health increment (Step 4):

$$En_H_inc = En_F_H_inc \times Available_Energy$$

The remaining energy, En_D_inc , will be used in development (Step 5), increasing D value:

$$En_D_inc = Available_Energy - En_H_inc$$

The next step will be to determine the intermediate health increment, H_inc , correspondent to the amount of energy used in health restoration, En_H_inc (Step 6). That is the role of $Conv$, energy to health conversion function:

$$H_inc = Conv(En_H_inc)$$

From a biological point of view, it isn't reasonable to assume possible too large health increments. It is, thus, necessary to limit H_inc to a certain maximum value, as a function of both H_{int} and D values, as different health recuperation efficiencies exist at different health and development values (Step 7).

En_D_inc will directly be used in raising D value, after it has passed through a limiting function, analogous in purpose to the one used in limiting H_inc (Step 8).

Finally, D value is inspected and compared to the phase's thermal constant, which has been shown in Table 1. If D is greater than it, then the insect will have completed its development and pass unto the next phase.

To achieve the different environmental influence, every time that *Damage* function is evaluated (Step 1), a random value following a zero mean normal distribution is added to the obtained value.

CONCLUSION

In this simulation, the iteration's time was 2 hours, namely the model will be fed by new temperature and relative humidity values every 2 hours. A typical behaviour of an artificial insect's internal state variables, health and development, is presented in Figure 8, during a simulation.

In order to compare the simulation results with available experimental data, the development for N artificial insects was simulated in the same time, which implied an initial insect population size N ; the average of each artificial insect's development time and sum of dead insect were counted at the end of simulation. Mortality rates were obtained from the dead fraction of the initial population at the end.

In Table 3 we present the obtained results for pupa's development in population size 100, with variable constant temperature and relative humidity of

Figure 8: Internal state variables during simulation

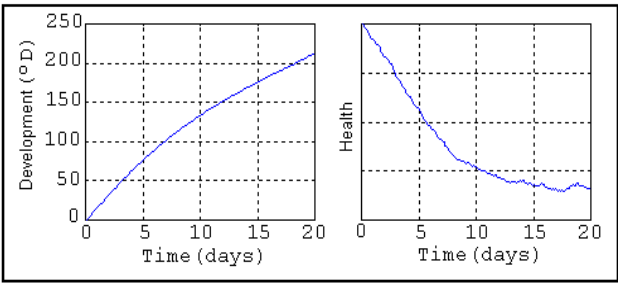


Table 3: Simulation results for pupa's development in population size 100

Table 3	Development Time (Days)		Mortality Rate (%)	
Constant Temperature (Celsius)	Observation from Crovetti (82a)	Simulation Results	Observation from Crovetti (82a)	Simulation Results
10	89.6	87.89	70.2	71
13	54.8	53.62	46.4	42
15	39.3	38.43	22.6	21
16	30.4	28.88	19.2	20
17	27.9	27.33	7.9	8
18	25.9	24.95	6.3	6
21	17.5	16.45	3.2	3
22	14.8	14.70	16	14
23	14.7	13.71	18.3	17
25	11.8	11.31	21.2	21
26	10.5	10.82	35.6	31
27	10.9	10.28	39.8	37
28	10.6	10.47	64.5	61
29	10.1	10.10	46.2	53
30	9.97	9.72	85.4	86
31	9.24	9.29	96.2	96

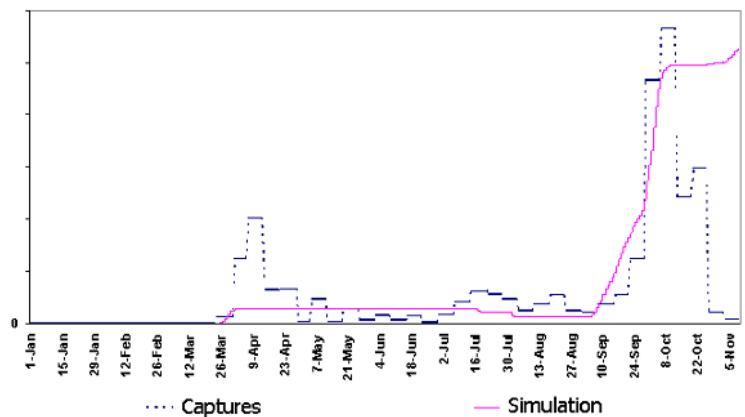
75%(±10%), comparable to those experimental data from Croveti (1982). The root mean square error of Development Time and Mortality Rate are 0.85 and 2.70%.

In order to apply the model to the olive protection in Portugal, manual parameter tuning is performed on the *Damage* and *Distribute_Energy* functions, with slightly decreases for the *Damage* under relative humidity of 75% and

Table 4: Simulation results for pupa's development for olive fly subspecies at north of Portugal

Table 4	Development Time (Days)		Mortality Rate (%)	
Constant Temperature (Celsius)	Observation from Portugal	Simulation Results	Observation from Portugal	Simulation Results
10	89.62	90.01	20	19
13	57	57.08	12	13
15	38.9	39.51	10	10
16	29.8	30.16	10	10
17	27.8	28.34	10	10
18	25	25.41	5	6
21	17.5	17.66	0	0
22	14.7	14.66	0	0
23	14.4	14.33	0	0
25	11.85	11.83	0	0
26	11.85	11.58	0	0
27	10.9	10.91	10	11
28	10.6	10.33	20	21
29	10.25	10.02	30	31
30	9.93	9.66	38	38
31	9.45	9.41	45	47
35	-	8.33	-	52

Figure 9: Illustration for the simulation of the insect population dynamics



increases for the *Distribute_Energy*. The simulation results also fit the observation data very well. In Table 4, the simulation results in population size 100 closely resemble those data observed in lab experiment, which is from an olive fly subspecies at the north of Portugal. The root mean square error of Development Time and Mortality Rate are 0.298 and 0.791%.

The model was also fed by real monitored climatic data of 1993 from north of Portugal with variable temperature and relative humidity. Simulation results have provided development times and mortality rates. The simulation results resemble the data trends of adults capture by yellow panel traps.

Availability of weather forecast up to several days allows the prediction of the pest development. This information is a very important component for the risk evaluation and treatment decision processes in terms of economical and environmental costs.

The knowledge of the pesticide and control method action curves enables the precise time positioning of the treatment in order to achieve maximum efficacy.

FUTURE TRENDS

The model has shown its potential for further research on insect population dynamics. The improvement of the insect's agent definition will use a distributed starting value of Health and an initial population where the insect's age is normally distributed.

Furthermore, the simulated artificial agents evolving in the olive tree, olive fruit and micro-environmental changes due to the olive's state will also be considered.

For example, the existence of the food providing the necessary nutrients will be simulated; the oviposition should also consider the picking time and the residuum of the fruit, and so on.

In this approach, the large number of parameters must be tuned manually. The proposed solution is to use evolutionary algorithms, possibly genetic algorithms, in distributed configurations, to search for good combinations of parameters, obviously restricted to biologically plausible values (Goldberg, 1989). Each agent of the artificial insect will be defined and coded by “chromosome.”

As an added benefit, the same technique can be used to adapt the model to other regions not initially considered. With an automatic parameter adjustment using genetic algorithms, the model would account for regional differences in insect species, that is, the existence of subspecies of insects that behave somewhat differently one from another. Subspecies, with different reactions to environmental variables or different sensitivity to control, will be represented in the difference of internal artificial agents.

As every model should always be subject to a rigorous validation phase, the simulation results of population development should then be compared with those data obtained in field experiments, such as the trapped insects number, the data of field sampling investigation, etc., thus improve its accuracy and confirm the validity of the model.

The output of the model will integrate with a control measure to help protection decision making. With this decision support tool, the farmers and pest control advisors can select a given climatic data range from the climatic database, or with the weather prediction, then execute the simulation; it can help the user pinpoint an adequate program for olive fly control, minimizing the damage, economic loss and the pollution of pesticide.

ACKNOWLEDGMENT

Research work supported by a grant SFRH/BD/4936/2001 from the National Foundation of Science and Technology.

REFERENCES

- Bento, A. (1999). A contribution to the knowledge of *Bactrocera oleae* in Trás-os-Montes region (Northeastern Portugal)—phenology, losses and control, *Acta Horticulturae*, (474), 541-544.
- Carnahan et al. (1997). Computer simulation of Dispersal by *Anopheles Gambiae* in West Africa. *Artificial Life V*. MIT Press, 387-394.

- Costa, J. & Rosa, A. (1998). Artificial life modelling of Downy Mildew of the grapevine. *Journal of Zhejiang Agricultural University*, 24(5), 509-516.
- Craig W. R. (1992). An evolved, vision-based model of obstacle avoidance behavior. *Artificial Life III*, 327-346.
- Craig, W. R. *Homepage of Boids*, <http://www.red3d.com/cwr/boids/>.
- Crovetti, A., Quaglia, F. & Rossi, E. (1982). The heat-units accumulation method for forecasting the *Dacus Oleae* (Gmel.) lifecycle: Results of a study carried out in a biotope of the Southern Tuscany during the years 1980-1982, *Frustula Entomologica, nuova serie X*, 109-117.
- Dicola, G. & Crovetti, A. (1984). Numerical simulation of the population development in *Dacus oleae* (Gmelin). *Integrated Pest Controls in Olive-Groves*, 128-135.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publisher.
- Lange et al. (1997). Investigating forest growth model results on evolutionary time scales. *Artificial Life VI*, MIT Press, 418-420.
- Langton, C. G. (1992). *Introduction, Artificial Life II*, 3-23.
- Michelakis. (1986). Bio-ecological data of the olive fly (*Dacus Oleae* Gmel.) in Crete-Greece. *Proceedings of 2nd International Symposium Fruit Flies*, 397-406.
- Noble, J. (1997). The scientific status of artificial life, *European Conference on Artificial Life*.
- Pitzalis, M. (1984). Bioclimatology and insect development forecast: Degree-days and phenophases of *Dacus Oleae* (Gmel.). *Integrated Pest Control in Olive-Groves, Proceedings of the CEC/FAO/IOBC International Joint Meeting*. Pisa April 3-6, 1984, 84-93.
- Stuart Kauffman, (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.
- Terzopoulos et al. (1996). Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life IV*. MIT Press, 17-27.
- Von Neumann, J. & Arthur, W. B. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press
- Wilber, P. & Shapiro, H. (1997). An artificial life approach to host-parasite interactions. *Ecological Modelling*, 101(1), 113-122.

Chapter XII

Applications of Data-Driven Modelling and Machine Learning in Control of Water Resources

D.P. Solomatine

International Institute for Infrastructural, Hydraulic and Environmental Engineering
(IHE-Delft), The Netherlands

ABSTRACT

Traditionally, management and control of water resources is based on behavior-driven or physically based models based on equations describing the behavior of water bodies. Since recently models built on the basis of large amounts of collected data are gaining popularity. This modelling approach we will call data-driven modelling; it borrows methods from various areas related to computational intelligence—machine learning, data mining, soft computing, etc. The chapter gives an overview of successful applications of several data-driven techniques in the problems of water resources management and control. The list of such applications includes: using decision trees in classifying flood conditions and water levels in the coastal zone depending on the hydrometeorological data, using artificial neural networks (ANN) and fuzzy rule-based systems for building controllers for real-time control of water resources, using ANNs and M5 model trees in flood control, using chaos theory in predicting water levels for ship guidance, etc. Conclusions are drawn on the applicability of the mentioned methods and the future role of computational intelligence in modelling and control of water resources.

INTRODUCTION

A *model* can be defined as a simplified representation of reality with an objective of its explanation or prediction. In engineering, the term *model* is used traditionally in one of two senses:

- (a) a mathematical model based on the description of behaviour (often physics, or first-order principles) of a phenomenon or system under study, referred to later as behavioural (also process, or physically based) models;
- (b) a model built of material components or objects, which is often referred to as scale (or physical) model (since it is usually smaller than the real system).

These views of a model are widely adopted and taught. Understandingly, in social and economical studies, scale modelling would be a difficult undertaking, but behavioural models based on mathematical descriptions of processes are widely spread and used.

Traditionally, management and control of water resources was based on good understanding of the underlying processes and use so-called “physically based” (or “knowledge-driven,” behavioral) models. These could be for example models based on Navier-Stokes' equation describing behavior of water in particular circumstances. Examples are surface (river) water 1D models, coastal 2D models, groundwater models, etc. Equations are solved using finite-difference, finite-element or other schemes, and results—normally water levels, discharges—are presented to decision makers. Often such models are called *simulation models*. Knowledge-driven models can be also “social,” “economic,” etc.

On the contrary, a “*data-driven*” model of a system is defined as a model connecting the system state variables (input, internal and output variables) with only a limited knowledge of the details about the “physical” behavior of the system. “Hybrid models” combine both types of models.

It should be stated that the process of modelling includes studying the system, posing the problem, data preparation, building the model (normally a machine learning model), testing the model, using the model, interpreting the results and, possibly, reiterating. In this chapter we will consider only the techniques for data-driven modelling proper.

Techniques used in data-driven modelling originate in various areas (often overlapping):

- machine learning (decision trees, Bayesian and instance-based methods, neural networks, reinforcement learning);
- soft computing, and in particular fuzzy rule-base systems induced from data;
- data mining (uses methods of machine learning and statistics);
- methods of non-linear dynamics and chaos theory (often considered as part of time series analysis, but which are actually oriented towards the analysis of large data sets).

In this chapter the main techniques used for data-driven modelling will be mentioned, and an overview of their use in the problems of water resources management and control will be given.

MACHINE LEARNING AS THE BASIS OF DATA-DRIVEN MODELLING

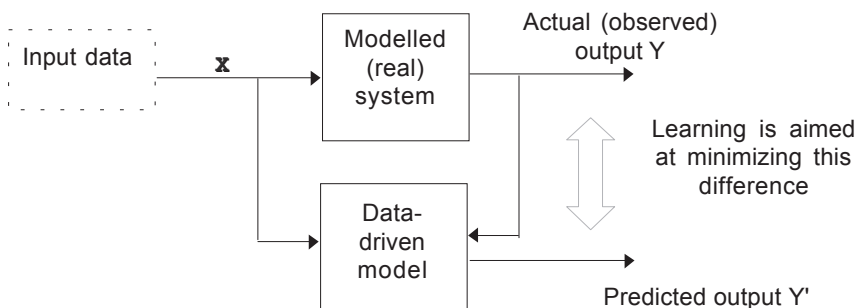
Machine learning is the main source of methods for the data-driven modelling problem (Figure 1). A *machine learning* method is an algorithm that estimates hitherto unknown mapping (or dependency) between a system's inputs and its outputs from the available data (Mitchell, 1998). By data we understand the known samples that are combinations of inputs and corresponding outputs. As such a dependency is discovered, it can be used to predict (or effectively *deduce*) the future system's outputs from the known input values.

There are four main styles of learning considered:

- *Classification*—On the basis of classified examples, a way of classifying unseen examples is to be found.
- *Association*—Association between features (which combinations of values are most frequent) is to be identified.
- *Clustering*—Groups of objects (examples) that are “close” are to be identified.
- *Numeric prediction*—Outcome is not a class, but a numeric (real) value. Often it is called regression.

The oldest area of estimating dependencies from data is statistics, as represented by multivariate regression and classification. In the 60s and 70s, new techniques which were often not based on the assumptions of “well-behaved” statistical distributions of random processes started to emerge, and these were used in many successful applications. Among these techniques were: pattern recognition and cluster analysis, methods trying to imitate the human brain and perception, like

Figure 1: Learning in data-driven modelling



neural networks and fuzzy systems (Tsoukalas & Uhrig, 1997), genetic programming (Koza, 1992), decision trees (Quinlan, 1992), and reinforcement learning (Watkins & Dayan, 1992).

In statistics the following four types of data are considered: nominal, ordinal, interval and ratio (real-valued). In machine learning, for simplicity, we often speak only of two data types: nominal (classes) and real-valued. We will divide the applications with respect to the main data types involved—nominal or real-valued.

PROBLEMS DESCRIBED BY NOMINAL DATA

Classification is treated often as finding classes of data points $\{a_i\} \in \mathbb{R}^n$. Classes must be such that points in a class are close to each other in some sense, and classes are far from each other. *Clustering* is finding groups (subsets) of data without assigning them to particular classes.

Among the most important methods currently used, the following can be mentioned:

- partition-based clustering (K-means, fuzzy C-means, based on Euclidean distance);
- density-based spatial clustering DBScan (for clusters of arbitrary shapes);
- SOF maps (Kohonen neural networks) clustering;
- Bayesian classification;
- decision trees classification (Quinlan, 1992; Witten & Frank, 2000);
- support vector machines (SVM) classification.

Research into extending statistically based induction principles resulted in the increased interest to classification methods such as Bayesian learning. Another important development in this area is *statistical learning theory* (Vapnik, 1998).

Statistical Learning Theory (Support Vector Machines)

Statistical learning theory is associated with the research performed in the 1960–80s in the Institute for Control Problems of the Russian Academy of Sciences in the department of Aizerman (see, for example, Aizerman et al., 1963). Later these early results were extended and generalised to provide the subject currently known as *statistical learning theory* of Vapnik (1998) which serves as a basis of *support vector machine (SVM)* technique. Vapnik's theory is based on solid principles and allows for generalising and the finding of common elements among various machine learning techniques.

Statistical learning theory made an important step: instead of trying to choose the approximating function based on how well it reproduces the *training set* (minimizing the *empirical risk*), it chooses the function that reproduces well also the

verification set (thus minimizing the *structural risk*). On the basis of statistical learning theory, a method of building the discriminating surfaces based on the so-called *support vector machines* (SVM) was developed. This theory and the SVM methods show the superior qualities in complex classification problems.

Decision Trees

A decision tree is quite a simple tool (however effective) representing a step-wise decision-making process about assigning an instance to a predetermined class. In this tree-like structure, the nodes contain the conditions on the attributes' values, and the leaves—the classes. The choice of an attribute on which partitioning is performed is based on the maximum information gain (the expected reduction in entropy caused by partitioning the examples according to this attribute).

Decision trees classify instances by sorting them down the trees from the root to some leaf node that provides the classification of the instance. Each node in the trees specifies a test of some attribute of the instance, and each branch descending from node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the trees, testing the attribute specified by this node, then moving down the tree's branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node.

Classification and clustering plays an important role in pattern recognition, that is identification of a class to which a new data point (pattern) could be attributed. One of the popular applications of classification is classifying a customer to a particular class of consumer behavior, or a company described by a number of parameters can be classified to classes like “very safe,” “safe,” “unsafe,” “unknown.”

Practical Applications

A number of examples of using classification and clustering in water management and control were reported:

- Hall et al. (2000) used SOFM for classifying catchments into groups based on their 12 characteristics, and then applying ANN to model the regional flood frequency.
- Hannah et al. (2000) used clustering for finding groups of hydrographs on the basis of their shape and magnitude; clusters are then used for classification by experts.
- In a similar fashion Harris et al. (2000) applied clustering to identify the classes of river regimes.
- Frapporti et al. (1993) used the method of fuzzy c-means clustering in the problem of classifying shallow Dutch groundwater sites into homogeneous groups.

- The use of fuzzy classification in the problem of soil classification on the basis of cone penetration tests (CPTs) is addressed by Zhang et al. (1999).

Our experience of using classification methods includes the following:

- Using self-organizing feature maps (Kohonen neural networks) as clustering methods, and SVM as classification method in aerial photos interpretation. In this application various classification methods were used in the problem of interpreting an aerial photo of the size of 4387x2786 pixels. Four land cover classes were identified—wood and scrub, agriculture, meadow and urban area (Velickov, Solomatine, Yu & Price, 2000).
- Using decision trees in classifying surge water levels in the coastal zone depending on the hydrometeorological data (Solomatine, Velickov, Rojas & Wust, 2000).
- Classification of the river flow levels according to their severity in the problem of flood control (the constructed decision tree is given above).

Classification of Flows in a Flood Management Problem

This latest example of a decision tree built for making prediction of river flow class (low, medium or high flow) is given below. The data of a catchment in Southern Europe included hourly data on rainfall R , evapotranspiration E and runoff R . The problem posed was to predict the class of flow 3 hours ahead. The variables for building a decision tree model were selected on the basis of correlation analysis (RE stands for the effective rainfall, that is rainfall minus evapotranspiration):

- inputs: $RE_t, RE_{t-1}, RE_{t-2}, RE_{t-3}, Q_t, Q_{t-1}$
- output: class of Q_{t+3} (L, M or H).

The set of 1,854 examples was used for training and 300 for verification. Value for Low flow (denoted in the tree as L) was chosen to be $50 \text{ m}^3/\text{s}$, for Medium flow (denoted as M)— $350 \text{ m}^3/\text{s}$ and for High (H)— $750 \text{ m}^3/\text{s}$. The built tree (shown horizontally) follows:

Decision tree in a flood control problem

```

 $Q_t \leq 51.45$ 
|
|    $RE_{t-1} \leq 0.6686$ : L (815.0/10.0)
|   |
|   |    $Q_t \leq 25.59$ : L (24.0)
|   |   |
|   |   |    $Q_t > 25.59$ : M (24.0/7.0)
|   |
|   |
|   |
 $Q_t > 51.45$ 
|
|    $RE_{t-1} \leq 2.3955$ 
|   |
|   |    $Q_t \leq 59.04$ 
|   |   |
|   |   |    $RE_t \leq -0.0255$ 
|   |   |
|   |   |

```

```

| | | Q_t <= 52.67: L (5.0)
| | | Q_t > 52.67: M (7.0/1.0)
| | RE_t > -0.0255: M (63.0/4.0)
| Q_t > 59.04
| | Q_{t-1} <= 348.55: M (271.0)
| | Q_{t-1} > 348.55
| | | Q_t <= 630.2: M (7.0)
| | | Q_t > 630.2: H (3.0)
RE_{t-1} > 2.3955
| | Q_t <= 247.68
| | | RE_t <= 3.3031: M (3.0)
| | | RE_t > 3.3031: H (7.0/3.0)
| | Q_t > 247.68: H (9.0)

```

The problem of classification of flows was solved using several classification methods; their performance is given in the following table.

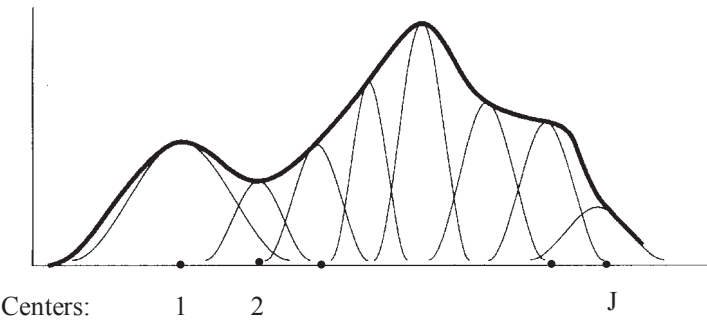
Interpretation of Aerial Photos

Another application of clustering and classification techniques was oriented towards interpretation of aerial photos with the help of self-organizing feature maps (SOFM), vector quantization, SVM and C4.5 decision trees (Velickov et al., 2000). The performance of several classifiers—vector quantization VQ (part of SOFM approach), C4.5 decision tree algorithm and SVM—was compared on a problem of an aerial photo interpretation; see Table 2 and Figure 3. For VQ and SVM, the SOFM clustering was applied first. The best results were achieved with the SVM classifier using the radial basis functions.

Table 2: Performance of classifiers in aerial photo interpretation (% of the misclassified examples)

VQ	C4.5	SVMs		
		radial basis function	simple polynomial kernel	full polynomial kernel
3.12	2.87	0.19	0.78	0.39

Figure 3: Approximation by RBFs in one dimension



PROBLEMS DESCRIBED BY REAL-VALUED DATA

Most engineering problems are formulated with the view of real-valued data. The problem of prediction of real-valued variables is also called a *regression problem*. Often a choice of engineers is to apply linear methods like linear regression and more sophisticated variation for time series, but still linear, a family of ARIMA methods.

RBF: A Method of Combining Functions Inspired by Function Fitting

Since machine learning aims at finding a function that would best approximate some given function, it can be seen also as a problem of function fitting, and this prompts for the use of the corresponding methods already available. This problem was studied in mathematics for more than 150 years. In function fitting (Gershenfeld, 2000) an attempt is made to find functions that, being combined, would approximate a given function (or a data set).

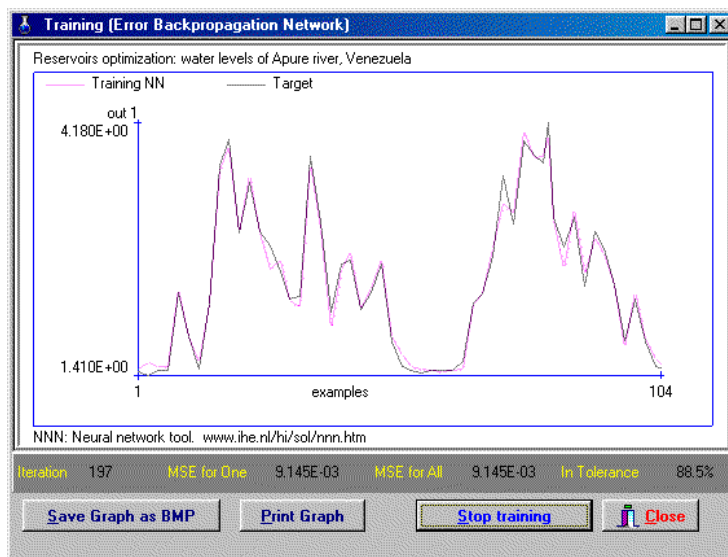
Traditionally, linear (as in linear regression) or polynomial functions were used. Examples could be *splines*—combinations of functions that use cubic polynomials to match the values and first and second derivatives (at the boundaries) of the function to be approximated. Another example is orthogonal so-called orthogonal polynomial functions, e.g., Chebyshev polynomials.

The problem of combining complex polynomial functions is that in order to improve the fit, it is necessary to add higher-order terms, and they may diverge very quickly (being a good approximator close to one point, and a very bad one a bit further). Matching data requires a delicate balancing of the coefficients, resulting in a function that is increasingly “wiggly.” Even if it passes near the training data, it will be useless for interpolation or extrapolation. This is particularly true for functions that have sharp peaks (which are actually so frequent in many engineering applications).

Radial basis functions (RBFs) (quite simple in nature) could be seen as a sensible alternative to the use of complex polynomials. Consider a function $z = f(\mathbf{x})$, where \mathbf{x} is a vector $\{x_1, \dots, x_I\}$ in I -dimensional space. The idea is approximate—a function $z = f(\mathbf{x})$ by another function $F(\mathbf{x})$ in a proximity to some “representative” locations (centers) $\mathbf{w}_j, j=1, \dots, J$ (Figure 4).

The question is then how to find the position of centers \mathbf{w}_j and the “height parameter” of the functions $F(\mathbf{x})$. This can be done by building a *radial-basis function (RBF) neural network* (Schalkoff, 1997); its training allows us to identify the unknown parameters.

Figure 4. ANN reproducing the behaviour of one-dimensional river model (water level), Apure river basin (Venezuela): training



MLP: A Method of Combining Functions Inspired by the Workings of the Brain

An important step towards non-linearity in function representation was a multi-layer perceptron (MLP), the most widely used class of artificial neural networks (ANNs). It has been mathematically proven that adding up simple functions, as an ANN does, allows for universal approximation of functions (Kolmogorov, 1957). After the mid-1980s, when methods for finding these functions (training an MLP) were found, it became the most successful machine learning method currently known. Various types of ANNs are widely used in numerical prediction and also in classification.

ANNs are applied in many areas, including pattern recognition, voice recognition, cryptography, prediction of stock market trends, consumer behavior, etc. ANN can be also used to replicate the behavior of complex models. This may be needed when models are slow but the results of modelling are needed in real time. An application of ANN, in approximating the behavior of a complex hydrodynamic model (25 inputs and 1 output), is considered by Solomatine and Torres (1996); see Figures 5 and 6. Neural network code developed for that application can be run across the Internet (see www.ihe.nl/hi/sol). Experience shows that ANNs, being highly non-linear models, are superior to regression and ARIMA models.

Figure 5: ANN reproducing the behaviour of one-dimensional river model (water level), Apure river basin (Venezuela): verification

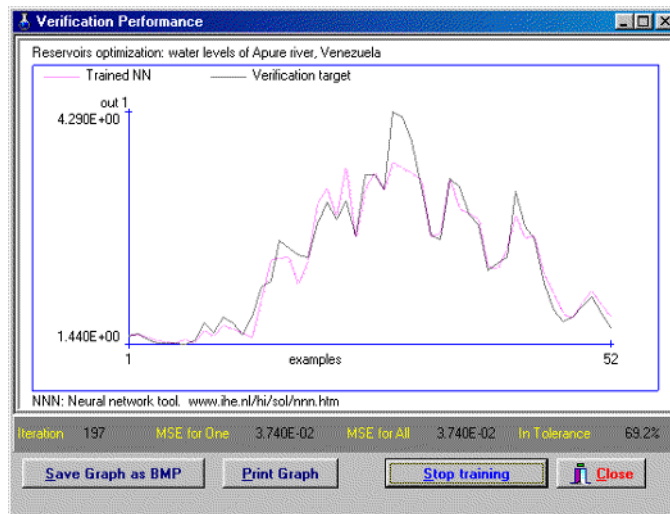
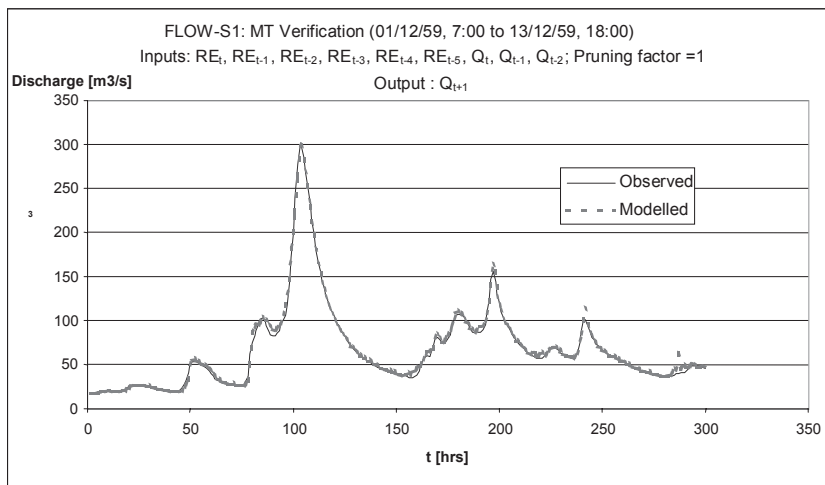


Figure 6: M5 model tree (pruned) in predicting flow $Q(t+1)$



M5 Model Tree: A Method of Combining Functions Inspired by a Decision Tree

Decision trees can be generalised to regression trees and model trees that can deal with continuous attributes. Trees-structured regression is built on the assumption that the functional dependency is not constant in the whole domain, but can be approximate as such on smaller subdomains. For the continuous variables, these subdomains then can be searched for and characterized with some “local” model.

Depending on the nature of such a model, there are two types of trees used for

numerical prediction:

- If the local model gives an average value of the instances for this local scope (zero-order model), then the overall approach is called a *regression tree*. Regression trees were introduced in the CART system of Breiman et al. (1984). CART, for “classification and regression trees,” incorporated a decision trees inducer for discrete classes very much like that of C4.5, which was developed independently, as well as a scheme for inducing regression trees.
- If a local model is a linear regression function of the input variables, then the overall approach is called a *model tree*. There are two (similar) approaches known:
- M5 models trees (Quinlan, 1992) implemented in Cubist software (www.rulequest.com) and, with some changes, Weka software (Witten & Frank, 2000). Some details of the M5 algorithms are given by Wang and Witten (1997).
- Approach by Friedman (1991) in his MARS (*multiple adaptive regression splines*) algorithm implemented as MARS software (www.salford-software.com).

Note that the model tree approach is oriented at building a linear model locally, so that overall it can be said that this piece-wise linear model has some properties of a non-linear model.

The construction of model trees is similar to that used in construction of decision trees, although the splitting criterion is different. Each leaf then represents a local model and in principle could be (locally) more accurate than a global model (even a non-linear one, e.g., a neural network) trained on the whole data set. The M5 model trees splitting criterion is SDR (standard deviation reduction), which is used to determine which attribute is the best to split the portion T of the training data that reaches a particular node:

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

where T, T_1, T_2, \dots are the sets that result from splitting the node according to the chosen attribute; $sd(.)$ is the standard deviation.

The linear regression method is based on an assumption of linear dependencies between input and output. In an M5 model tree, a step towards non-linearity is made—since it builds a model that is locally linear, but overall is non-linear. In fact an M5 tree is a modular model—it consists of modules that are responsible for

modelling particular subspace of the input space. Model trees may serve as an alternative to ANNs (which are *global* models), are often almost as accurate as ANNs and have important advantages:

- Training of MT is much faster than ANN, and it always converges.
- The results can be easily understood by decision makers.
- By applying *pruning* (that is making trees smaller by combining subtrees in one node), it is possible to generate a range of MTs—from an inaccurate but simple linear regression (one leaf only) to a much more accurate but complex combination of local models (many branches and leaves).

Practical Applications

Data-driven methods, especially neural networks, know dozens of successful applications in the water sector. The use of ANNs to model the rainfall-runoff process is addressed in the works of Hsu et al. (1995), Minns and Hall (1996), Abrahart and See (2000), and in a collection of papers edited by Govindaraju and Ramachandra Rao (2000).

Our experience of using machine learning methods in real-valued prediction includes:

- replicating behavior of hydrodynamic/hydrological model of Apure river basin (Venezuela) with the objective of using the ANN in model-based optimal control of a reservoir (Solomatine & Torres, 1996);
- modelling a channel network using ANN (Price et al., 1998);
- building ANN-based intelligent controller for real-time control of water levels in a polder (Lobbrecht & Solomatine, 1999);
- modelling rainfall-runoff process with ANNs (Dibike, Solomatine & Abbott, 1999);
- surge water level prediction in the problem of ship guidance using ANN;
- reconstructing stage-discharge relationship using ANN (Bhattacharya & Solomatine, 2000);
- using M5 model trees to predict discharge in a river (see example below);
- using SVMs in prediction of water flows for flood management (Dibike, Velickov, Solomatine & Abbott, 2001).

Here we will mention the application of model trees to the same data set as was mentioned in the application of classification algorithms. It includes the hourly data on rainfall and flow in a catchment for 3 months. Training set includes 1,854, and the verification set—300 instances. The problem is to predict the discharge value Q_{t+1} for the next hour. Analysis of the catchment and the mutual dependencies between variables allowed for selecting the following input variables: effective rainfall (RE) for times t , $t-1$, $t-2$, $t-3$, $t-4$, $t-5$, and discharges Q at times t ,

t-1, t-2, t-3.

Multiple linear regression model was built and has the following form:

$$Q_{t+1} = 0.842 + 1.04RET + 5.05RET-1 - 1.23RET-2 - 0.0842RET-3 + 0.419RET-4 - 0.429RET-5 + 1.87Q_t - 1.2Q_{t-1} + 0.295Q_{t-2}$$

with the RMSE on a verification set of 82.6 m³/s.

M5 model tree was built for the same problem with the help of Weka software (Witten & Frank, 2000) and it is shown below:

```

Qt <= 59.4 :
|   Qt <= 32.5 : LM1 (1011/1.64%)
|   Qt > 32.5 : LM2 (396/6.17%)
Qt > 59.4 :
|   Qt <= 128 :
|   |   Qt <= 87.5 :
|   |   |   RET-3 <= 0.264 : LM3 (170/5.44%)
|   |   |   RET-3 > 0.264 :
|   |   |   |   RET-2 <= 1.13 : LM4 (36/6.43%)
|   |   |   |   RET-2 > 1.13 :
|   |   |   |   |   RET-3 <= 1.45 : LM5 (9/20.4%)
|   |   |   |   |   RET-3 > 1.45 :
|   |   |   |   |   |   RET-4 <= 1.35 : LM6 (3/10%)
|   |   |   |   |   |   RET-4 > 1.35 : LM7 (3/22.2%)
|   |   |   Qt > 87.5 :
|   |   |   |   Qt <= 103 :
|   |   |   |   |   RET <= 0.121 :
|   |   |   |   |   |   Qt <= 95 :
|   |   |   |   |   |   |   Qt-2 <= 88.1 : LM8 (5/9.21%)
|   |   |   |   |   |   |   Qt-2 > 88.1 : LM9 (18/7.53%)
|   |   |   |   |   |   Qt > 95 : LM10 (19/7.04%)
|   |   |   |   |   RET > 0.121 :
|   |   |   |   |   |   RET <= 1.68 :
|   |   |   |   |   |   |   RET-5 <= 0.167 : LM11 (2/6.82%)
|   |   |   |   |   |   |   RET-5 > 0.167 : LM12 (5/17.1%)
|   |   |   |   |   |   RET > 1.68 :
|   |   |   |   |   |   |   RET <= 3.83 : LM13 (2/7.08%)
|   |   |   |   |   |   |   RET > 3.83 : LM14 (2/0.144%)
|   |   |   |   Qt > 103 : LM15 (50/7.77%)
|   |   Qt > 128 : LM16 (123/38.6%)

```

Models at the leaves:

```

LM1: Qt+1 = 0.0388 + 0.0176RET + 0.0535RET-1 + 0.00866RET-2 +
      0.037RET-3 + 1.01Qt - 0.0127Qt-1 + 0.00311Qt-2
LM2: Qt+1 = -0.221 + 0.0277RET + 1.68RET-1 + 0.0411RET-2 + 7.3RET-
      3 + 1Qt - 0.0127Qt-1 + 0.00311Qt-2

```

```

LM3:  Qt+1 = 3.33 + 0.284REt + 0.761REt-1 + 0.927REt-2 + 0.43REt-3
      - 0.488REt-4 - 0.0852REt-5 + 1.04Qt - 0.147Qt-1 + 0.0351Qt-2
LM4:  Qt+1 = 7.14 + 0.452REt + 0.761REt-1 + 7.46REt-2 + 1.04REt-3 -
      1.1REt-4 - 0.0852REt-5 + 0.983Qt - 0.147Qt-1 + 0.0351Qt-2
LM5:  Qt+1 = 35.9 + 0.771REt + 0.761REt-1 + 7.72REt-2 + 2.69REt-3 -
      1.1REt-4 - 0.0852REt-5 + 0.622Qt - 0.147Qt-1 + 0.0351Qt-2
LM6:  Qt+1 = 39.5 + 0.452REt + 0.761REt-1 + 7.72REt-2 + 2.92REt-3 -
      1.1REt-4 - 0.0852REt-5 + 0.622Qt - 0.147Qt-1 + 0.0351Qt-2
LM7:  Qt+1 = 38.8 + 0.452REt + 0.761REt-1 + 7.72REt-2 + 2.92REt-3 -
      1.1REt-4 - 0.0852REt-5 + 0.622Qt - 0.147Qt-1 + 0.0351Qt-2
LM8:  Qt+1 = 29.3 + 2.58REt + 1.14REt-1 + 0.241REt-2 + 0.186REt-3 -
      0.3REt-4 - 1.51REt-5 + 1.02Qt - 0.422Qt-1 + 0.085Qt-2
LM9:  Qt+1 = 37.1 + 2.58REt + 1.14REt-1 + 0.241REt-2 + 0.186REt-3 -
      0.3REt-4 - 1.51REt-5 + 1.02Qt - 0.422Qt-1 - 0.0197Qt-2
LM10: Qt+1 = 34.2 + 2.58REt + 1.14REt-1 + 0.241REt-2 + 0.186REt-3
      - 0.3REt-4 - 1.51REt-5 + 1.03Qt - 0.422Qt-1 + 0.0148Qt-2
LM11: Qt+1 = 32.8 + 4.1REt + 3.85REt-1 + 0.241REt-2 + 0.186REt-3 -
      0.3REt-4 - 2.76REt-5 + 1.11Qt - 0.422Qt-1 - 0.0524Qt-2
LM12: Qt+1 = 32.6 + 4.1REt + 3.85REt-1 + 0.241REt-2 + 0.186REt-3 -
      0.3REt-4 - 2.76REt-5 + 1.11Qt - 0.422Qt-1 - 0.0524Qt-2
LM13: Qt+1 = 35.9 + 4.1REt + 4.28REt-1 + 0.241REt-2 + 0.186REt-3 -
      0.3REt-4 - 2.76REt-5 + 1.11Qt - 0.422Qt-1 - 0.0524Qt-2
LM14: Qt+1 = 36 + 4.1REt + 4.28REt-1 + 0.241REt-2 + 0.186REt-3 -
      0.3REt-4 - 2.76REt-5 + 1.11Qt - 0.422Qt-1 - 0.0524Qt-2
LM15: Qt+1 = 9.39 + 1.37REt + 3.78REt-1 + 0.241REt-2 + 0.186REt-3
      - 0.3REt-4 - 0.473REt-5 + 1.66Qt - 0.97Qt-1 + 0.211Qt-2
LM16: Qt+1 = 0.432 + 3.99REt + 3.24REt-1 - 0.04REt-2 + 1.76Qt -
      1.07Qt-1 + 0.257Qt-2

```

(here the RMSE on a verification set dropped down to just $3.85\text{ m}^3/\text{s}$.)

This tree was found to be too complex. The pruned (reduced) model tree (to 3 rules) from the total of 16 rules is shown below:

```

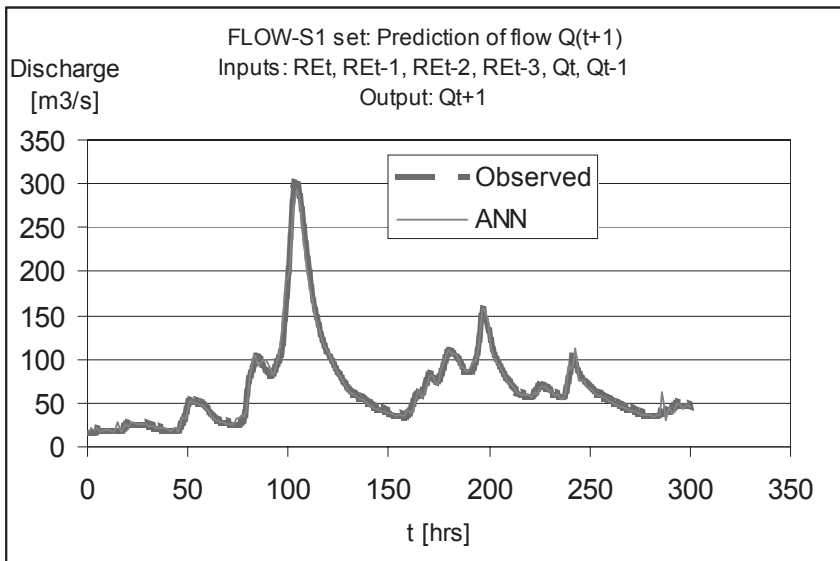
Qt <= 59.4 :
|   Qt <= 32.5 : LM1 (1011/1.64%)
|   Qt > 32.5 : LM2 (396/6.17%)
Qt > 59.4 : LM3 (447/23.5%)

LM1:  Qt+1 = 0.0388 + 0.0108REt + 0.0535REt-1 + 0.0173REt-2 +
      0.0346REt-3 + 1.01Qt - 0.0127Qt-1 + 0.00311Qt-2
LM2:  Qt+1 = -0.221 + 0.0108REt + 1.68REt-1 + 0.0626REt-2 + 7.3REt-
      3 + 1Qt - 0.0127Qt-1 + 0.00311Qt-2
LM3:  Qt+1 = 3.04 + 2.46REt + 4.97REt-1 - 0.04REt-2 + 1.75Qt -
      1.08Qt-1 + 0.265Qt-2

```

The RMSE on a verification set dropped even further down to $3.6\text{ m}^3/\text{s}$. (The reduction of the error may show that the original large tree was overfit.) Figure 7 shows the predicted discharge against the measured one.

Figure 7: ANN (MLP) in predicting flow $Q(t+1)$: 9 inputs, 5 hidden nodes, 1 output



Note that the final pruned model tree model does not use variables RE_{t-3} , RE_{t-4} , RE_{t-5} . The tree actually generated three models which are associated with the three levels of flow: very low (below 32.5 m³/s), low (from 32.5 to 59.6 m³/s), and high (above 59.5 m³/s). The analysis of regression coefficients may help the hydrological analysis: for example, it can be seen that in LM3 (high flows), the influence of the previous values of flows (Q_{t-1} and Q_{t-2}) is much higher than in the models for lower flows.

This final tree is very simple, understandable, needs even less input variables

Figure 8: ANN (MLP) in predicting flow $Q(t+3)$

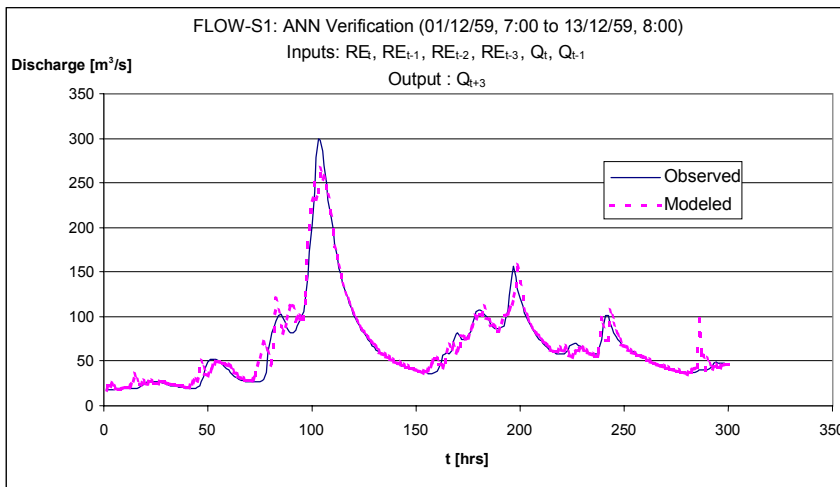
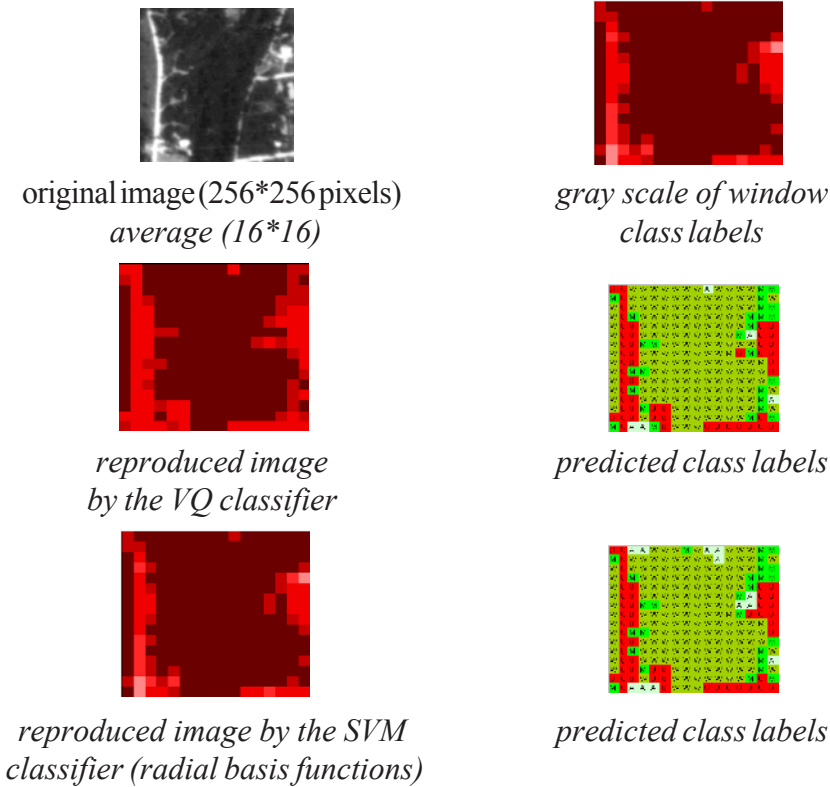


Figure 9: Interpretation of an aerial photo using Kohonen network (SOFM) and SVM classifiers (Velickov et al., 2000); the predicted class labels are shown: W: Wood and scrub, A: Agriculture, M: Meadow, U: Urban area



to feed and gives the lowest verification error. Its accuracy was found to be practically the same as of the ANN (RMSE=5.3 m^3/s) (Figure 8). It is interesting to see how the accuracy decreases (however, not considerably) for the problem of predicting flow $Q(t+3)$, that is 3 hours ahead (Figure 9; this ANN has 6 inputs instead of 9 due to the fact that there are not enough past measurements that would influence the flow 3 hours ahead).

FUZZY RULE-BASED SYSTEMS

The notion of fuzzy logic was introduced by L. Zadeh in 1965 and since then fuzzy systems are used in many applications, including, for example, controllers of washing machines or video cameras. Ideas of fuzzy logic are also used in building data-driven models trained on large data sets. Fuzzy rule-based systems (FRBSs) can be built by interrogating humans, or by processing the historical data. We used the second type of rules, and the basics of this approach are described in the books of Bardossy and Duckstein (1995), and Kosko (1993, 1997). Applications of

FRBS in water resources can be found in Pesti et al. (1996), Pongracz et al. (1999) and Abebe et al. (1999).

Practical Applications

Fuzzy logic had a significant number of applications; some of them are mentioned below:

- Bardossy et al. (1995) modeled the movement of water in the unsaturated zone using a fuzzy rule-based approach. Data generated by numerical solution of Richard's equation was used as examples to train (i.e., formulate the rules of) a fuzzy rule-based model.
- Bardossy and Duckstein (1995) also used adaptive fuzzy systems to model daily water demand time series in the Ruhr basin, Germany, and used fuzzy rules to predict future water demand. The approach used three input variables: the day of the week (working day or holiday), the daily maximum temperature and the general weather conditions of the previous days.
- Pesti et al. (1996) proposed a methodology for predicting regional droughts from large-scale atmospheric patterns. A fuzzy rule-based model was used to predict the so-called Palmer's Drought Severity Index (PDSI) of the New Mexico area based on atmospheric circulation patterns of the United States. With past records split for training and verification, good predictive abilities were reported in addition to easy implementation, simple coding and little computer time.

Our experience includes:

- Abebe, Solomatine and Venneker (1999) used FRBS for prediction of precipitation events.
- Abebe, Guinot and Solomatine (2000) used fuzzy logic approach in the analysis of groundwater model uncertainty.
- Lobbrecht and Solomatine (1999) used FRBS in building an intelligent controller for water management in polder areas.

NON-LINEAR DYNAMICS: CHAOS THEORY

Chaos theory (formulated by Lorentz in 1963) appeared to be an excellent predictive tool for time series. It uses only the time series itself, without the use of other related variables, so it is applicable when the time series carries enough information about the behavior of the system.

Chaos comprises a class of signal intermediate between regular sinusoidal or quasiperiodic motions and unpredictable, truly stochastic behavior (Tsonis, 1992). Chaotic systems are treated as "slightly predictable" and normally are

studied in the framework of non-linear system dynamics. With conventional linear tools such as Fourier transform, chaos looks like “noise,” but chaos has structure seen in the phase (state) space. The main reason for applying chaos theory is the existence of methods permitting prediction of the future positions of the system in the state space. The predictive capacity of chaos theory is by far better than any of the linear models like ARIMA.

Practical Application

We used chaos theory to predict the surge water level in the North Sea close to Hook of Holland; the data set included measurements of surge for 5 years with the 10-minute interval. For two-hours prediction the error was as low as 10 cm and superceded the methods used earlier (Solomatine et al., 2000).

CONCLUSION

Data-driven methods (in other words, methods of machine learning and data mining) have proven their applicability in many areas, including the financial sector, customer resource management, engineering, etc. Our experience shows their applicability to a wide range of problems associated with control of water resources. Normally a particular domain area will benefit from data-driven modelling if:

- there is a considerable amount of data available;
- there are no considerable changes to the modeled system during the period covered by modelling;
- it is difficult to build knowledge-driven simulation models, or in particular cases they are not adequate enough;
- there is a necessity to validate the results of simulation models with other types of models.

Successful analysis and prediction should be always based on the use of various types of models. For example, our experience shows that M5 model trees, combining local and global properties, could be close in accuracy to neural networks (being global, that is trained on the whole data set), and are more easily accepted by decision makers due to their simplicity.

The future is seen in using the *hybrid* models combining models of different types and following different modelling paradigms. It can be foreseen that the computational intelligence (machine learning) will be used not only for building data-driven models, but also for building optimal and adaptive *model structures* of such hybrid models.

ACKNOWLEDGMENTS

Part of this work was performed in the framework of the project “Data mining, knowledge discovery and data-driven modelling” of the *Delft Cluster* research programme supported by the Dutch government. The author is grateful to his co-authors and colleagues at IHE-Delft for the fruitful cooperation.

REFERENCES

- Abebe, A.J., Guinot, V. & Solomatine, D.P. (2000). Fuzzy alpha-cut vs. Monte Carlo techniques in assessing uncertainty in model parameters, *Proceedings of the 4th International Conference on Hydroinformatics*, Iowa City, USA, July.
- Abebe, A.J., Solomatine, D.P., & Venneker, R. Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events, *Hydrological Sciences Journal*, (45).
- Abrahart, R.J. & See, L. (2000). Comparing neural network and autoregressive moving average techniques for the provision of continuous river flow forecast in two contrasting catchments. *Hydrological Processes*, (14), 2157-2172.
- Aizerman, M., Braverman, E. & Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning, *Automation and Remote Control*. (25), 821-837.
- Bhattacharya, B. (2000). *Machine Learning in Real-Time Control of Regional Water Systems*. MSc Thesis. IHE-Delft, The Netherlands.
- Bhattacharya, B. & Solomatine, D.P. (2000). Application of artificial neural network in stage-discharge relationship, *Proceedings of the 4th International Conference on Hydroinformatics*, Iowa City, USA, July.
- Dibike, Y., Solomatine, D.P. & Abbott, M.B. (1991). On the encapsulation of numerical-hydraulic models in artificial neural network, *Journal of Hydraulic Research*, No. 2, 147-161.
- Dibike, Y.B., Velickov, S., Solomatine, D.P. & Abbott, M.B. (2001). Model induction with support vector machines: Introduction and applications. *Journal of Computing in Civil Engineering*, American Society of Civil Engineers (ASCE), 15(3), 208-216.
- Gershenfeld, N.A. (2000). *The Nature of Mathematical Modelling*. Cambridge University Press.
- Govindaraju, R.S. & Ramachandra Rao, A. (Eds.). (2001). *Artificial Neural Networks in Hydrology*. Dordrecht, Kluwer.
- Hall, M. J. & Minns, A. W. (1999). The classification of hydrologically homoge-

- neous regions. *Hydrol. Sci. J.*, (44), 693-704.
- Kolmogorov, A.N. (1957). On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, (114), 953-956.
- Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge.
- Leonard, T. & Hsu, J.S.J. (1999). *Bayesian Methods: An Analysis for Statisticians and Interdisciplinary Researchers*, Cambridge University Press.
- Lobrecht, A.H. & Solomatine, D.P. (1999). Control of water levels in polder areas using neural networks and fuzzy adaptive systems, In D. Savic, G. Walters (Eds), *Water Industry Systems: Modelling and Optimization Applications*, Research Studies Press Ltd., 509-518.
- Mitchell, T.M. (1998). *Machine learning*. McGraw-Hill.
- Price, R.K., Samedov, J. & Solomatine, D.P. (1998). Network modelling using artificial neural networks, *Proceedings International Conference on Hydroinformatics*, Balkema, Rotterdam.
- Quinlan, J.R. (1992). *C4.5: Program for Machine Learning*, Morgan Kaufmann.
- Solomatine, D.P. & Torres, L.A. (1996). Neural network approximation of a hydrodynamic model in optimizing reservoir operation. *Proceedings of the 2nd International Conference on Hydroinformatics*, Zurich, 201-206 September 9-13.
- Solomatine, D.P., Rojas, C., Velickov, S. & Wust, H. (2000). Chaos theory in predicting surge water levels in the North Sea, *Proceedings of the 4th International Conference on Hydroinformatics*, Iowa City, USA, July.
- Tsonis, A.A. (1992). *Chaos: From Theory to Applications*, New York: Plenum Press.
- Tsoukalas, L.H. & Uhrig, R.E. (1997). *Fuzzy and Neural Approaches in Engineering*, New York: John Wiley & Sons.
- Vapnik, V.N. (1998). *Statistical Learning Theory*, New York: John Wiley & Sons.
- Velickov, S. & Solomatine, D.P. (2000). Predictive data mining: Practical examples, *2nd Workshop on AI methods in Civil Engineering*, Cottbus, March.
- Velickov, S., Solomatine, D.P., Yu, X. & Price, R.K. (2000). Application of data mining techniques for remote sensing image analysis, *Proceedings of the 4th International Conference on Hydroinformatics*, Iowa City, USA, July.
- Watkins, C. & Dayan, P. (1992). Q-learning, *Machine Learning*, 3(8), 279-292.
- Witten, I.H. & Frank, E. (2000). *Data Mining*, Morgan Kaufmann Publishers.
- Zadeh, L.A. (1965). Fuzzy sets. *Inf. Control*, (8), 338-353.

Chapter XIII

Solving Two Multi-Objective Optimization Problems Using Evolutionary Algorithm

Ruhul A. Sarker, Hussein A. Abbass and Charles S. Newton
University of New South Wales, Australia

ABSTRACT

Being capable of finding a set of pareto-optimal solutions in a single run is a necessary feature for multi-criteria decision making, Evolutionary algorithms (EAs) have attracted many researchers and practitioners to address the solution of Multi-objective Optimization Problems (MOPs). In a previous work, we developed a Pareto Differential Evolution (PDE) algorithm to handle multi-objective optimization problems. Despite the overwhelming number of Multi-objective Evolutionary Algorithms (MEAs) in the literature, little work has been done to identify the best MEA using an appropriate assessment methodology. In this chapter, we compare our algorithm with twelve other well-known MEAs, using a popular assessment methodology, by solving two benchmark problems. The comparison shows the superiority of our algorithm over others.

INTRODUCTION

Multi-objective optimization problems (MOPs) optimize a set of conflicting objectives simultaneously. MOPs are a very important research topic, not only because of the multi-objective nature of most real-world decision problems, but

also because there are still many open questions in this area. In fact, there is no one universally accepted definition of *optimum* in MOP as opposed to single-objective optimization problems, which makes it difficult to even compare results of one method to another. Normally, the decision about what the *best* answer is corresponds to the so-called human decision maker (Coello Coello, 1999).

Traditionally, there are several methods available in the *Operational Research* (OR) literature for solving MOPs as mathematical programming models (Coello Coello, 1999). None of the OR methods treat all the objectives simultaneously which is a basic requirement in most MOPs. In addition, these methods handle MOPs with a set of impractical assumptions such as linearity and convexity.

In MOPs, there is no single optimal solution, but rather a set of alternative solutions. These solutions are optimal in the wider sense since there are no other solutions in the search space that are superior to (*dominate*) them when all objectives are simultaneously considered. They are known as pareto-optimal solutions. Pareto-optimality is expected in MOPs to provide flexibility for the human decision maker.

Recently, *evolutionary algorithms* (EAs) were found to be useful for solving MOPs (Zitzler & Thiele, 1999). EAs have some advantages over traditional OR techniques. For example, considerations for convexity, concavity and/or continuity of functions are not necessary in EAs, whereas they form a real concern in traditional OR techniques. Although EAs are successful, to some extent, in solving MOPs, the methods appearing in the literature vary a lot in terms of their solutions and the way of comparing their best results with other existing algorithms. In other words, there is no well-accepted method for MOPs that will produce a good set of solutions for all problems. This motivates the further development of good approaches to MOPs.

Recently, we developed a novel *Differential Evolution* (DE) algorithm for MOPs (Abbass, Sarker & Newton, 2001). The approach showed promising results when compared with the *Strength Pareto Evolutionary Algorithm* (SPEA) (Zitzler & Thiele, 1999) for two benchmark problems. However there are several other known methods such as Fonseca and Fleming's genetic algorithm (FFGA) (Fonseca & Fleming, 1993), Hajela's and Lin's genetic algorithm (HLGA) (Hajela & Lin, 1992), Niche Pareto Genetic Algorithm (NPGA) (Horn, Nafpliotis & Goldberg, 1994), Non-dominated Sorting Genetic Algorithms (NSGA) (Srinivas & Deb, 1994), Random Sampling Algorithm (RAND) (Zitzler & Thiele, 1999), Single Objective Evolutionary Algorithm (SOEA) (Zitzler & Thiele, 1999), Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985) and Pareto Archived Evolution Strategy (PAES) (Knowles & Corne, 1999, 2000). There are several versions of PAES like PAES, PAES20, PAES98 and PAES98mut3p. In this chapter, we compare the solutions of two benchmark problems, produced by

our DE algorithm with all these methods, using a statistical comparison technique recently proposed by Knowles and Corne (1999, 2000). From the comparison, it is clear that our algorithm outperforms most algorithms when applied to these two test problems.

The chapter is organized as follows. After introducing the research context, previous research is scrutinized. This is followed by the proposed algorithm. Experiments are then presented and conclusions are drawn.

PREVIOUS RESEARCH

Existing MEAs

MEAs for solving MOPs can be categorized as plain aggregating, population-based non-pareto and pareto-based approaches (Coello Coello, 1999). In this section, we would briefly discuss several population-based approaches as they are more successful when solving MOPs, and are popular among researchers and practitioners.

The Random Sampling Evolutionary Algorithm (RAND) (Zitzler & Thiele, 1999) generates randomly a certain number of individuals per generation, according to the rate of crossover and mutation (though neither crossover, mutation nor selection are performed). Hence the number of fitness evaluations was the same as for the EAs. Another algorithm called Single Objective Evolutionary Algorithm (SOEA) (Zitzler & Thiele, 1999) uses the weighted-sum aggregation. In contrast to other algorithms, 100 independent runs were performed per test problem, each run being optimized towards another randomly chosen linear combination of the objectives. The non-dominated solutions among all solutions generated in the 100 runs form the trade-off frontier achieved on a particular test problem.

The Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985) is a population-based non-Pareto approach. In this approach, the total population is divided into a number of populations equal to the number of objective functions to be optimized. Each population is used to optimize each objective function independently. The populations are then shuffled together followed by conventional crossover and mutation operators. Schaffer (Schaffer, 1985) realized that the solutions generated by his system were non-dominated in a local sense, because their non-dominance was limited to the current population, and while a locally dominated individual is also globally dominated, the converse is not necessarily true.

Hajela and Lin's genetic algorithm (HLGA) (Hajela & Lin, 1992) is also a non-Pareto approach that uses the weighted-sum method for fitness assignment. Thereby, each objective is assigned a weight between zero and one, with the sum of all weights being exactly equal to one. To search for multiple solutions in parallel,

the weights are encoded in the genotype. The diversity of the weight combinations is promoted by phenotypic fitness sharing. As a consequence, the EA evolves solutions and weight combinations simultaneously.

In the pareto-based approaches, the dominated and non-dominated solutions in the current population are separated. Goldberg (1989) suggested a non-dominated ranking procedure to decide the fitness of the individuals. Later, Srinivas and Dev (1994) introduced Non-dominated Sorting Genetic Algorithms (NSGAs) based on the idea of Goldberg's procedure. In this method, the fitness assignment is carried out through several steps. In each step, the non-dominated solutions constituting a non-dominated frontier are assigned the same dummy fitness value. These solutions have the same fitness values and are ignored in the further classification process. Finally, the dummy fitness is set to a value less than the smallest shared fitness value in the current non-dominated frontier. Then the next frontier is extracted. This procedure is repeated until all individuals in the population are classified.

Fonseca and Fleming (1993) proposed a slightly different scheme which is known as Fonseca and Fleming's genetic algorithm (FFGA). In this approach, an individual's rank is determined by the number of individuals dominating it. Without using any non-dominated ranking methods, Horn, Nafpliotis and Goldberg (1994) proposed the Niche Pareto Genetic Algorithm (NPGA) that combines tournament selection and the concept of Pareto dominance. Two competing individuals and a comparison set of other individuals are picked at random from the population; the size of the comparison set is given by a user-defined parameter. If one of the competing individuals is dominated by any member of the set and the other is not, then the latter is chosen as the winner of the tournament. If *both* individuals are dominated (or not dominated), the result of the tournament is decided by sharing: the individual that has the least individuals in its niche (defined by the *niche radius*) is selected for reproduction. Horn et al. (1994) used phenotypic sharing on the objective vectors.

The common features of the Pareto-based approaches mentioned above are: (i) the Pareto-optimal solutions in each generation are assigned either the same fitness or rank, and (ii) some sharing and niche techniques are applied in the selection procedure. Recently, Zitzler and Thiele (1999) proposed a Pareto-based method, the Strength Pareto Evolutionary Algorithm (SPEA). The main features of this approach are: it (i) sorts the non-dominated solutions externally and continuously updates the population, (ii) evaluates an individual's fitness depending on the number of external non-dominated points that dominate it, (iii) preserves population diversity using the Pareto dominance relationship and (iv) incorporates a clustering procedure in order to reduce the non-dominated set without destroying its characteristics.

Most recently, Knowles and Corne (1999, 2000) proposed a simple Evolution Strategy (ES), (1+1)-ES, known as the Pareto Archived Evolution Strategy (PAES) that keeps a record of limited non-dominated individuals. The non-dominated individuals are accepted for recording based on the degree of crowding in their grid (defined regions on the Pareto frontier) location to ensure diversity of individuals in the final solution. They also proposed an extension to this basic approach, which results in some variants of a $(\mu + \lambda)$ -ES. These are recognized as PAES (on-line performance using an archive of 100 solutions), PAES20 (off-line performance using an archive of 20 solutions), PAES98 (off-line performance using an archive of 98 solutions) and PAES98mut3p (PAES98 but with a mutation rate of 3%).

Our algorithm is a Pareto-based approach using Differential Evolution (DE) for multi-objective optimization (Abbass, Sarker & Newton, 2001). This algorithm is briefly introduced in a later section.

Comparison Techniques

MOPs require multiple, but uniformly distributed, solutions to form a Pareto trade-off frontier. When comparing two algorithms, these two factors (number of alternative solution points and their distributions) must be considered in addition to the quality of solutions. There are a number of assessment methodologies available in the literature to compare the performance of different algorithms. The *error ratio* and the *generational distance* are used as the performance measure indicators when the Pareto optimal solutions are known (Veldhuizen & Mamont, 1999). The *spread* measuring technique expresses the distribution of individuals over the non-dominated region (Srinivas & Deb, 1994). The method is based on a chi-square-like deviation distribution measure, and it requires several parameters to be estimated before calculating the spread indicator. The method of *coverage metrics* (Zitzler & Thiele, 1999) compares the performances of different multi-objective evolutionary algorithms by indicating whether the outcomes of one algorithm dominate those of another without measuring how much better it is.

A statistical comparison method called “attainment surfaces” was introduced by Fonseca and Fleming (1996). For two objective problems, the *attainment surface* is defined as the lines joining the points (candidate solutions) on the Pareto frontier generated by an algorithm. Therefore, for two algorithms *A* and *B*, there are two attainment surfaces. An attainment surface divides the objective space into two regions: one containing vectors which are dominated by the results produced by the algorithm, and another that contains vectors that dominate the results produced by the algorithm. A number of sampling lines can be drawn from the origin, which intersects with the attainment surfaces, across the full range of the Pareto frontier.

For a given sampling line, the intersection of an algorithm closer to the origin (for both minimization) is the winner. Fonseca and Fleming's idea was to consider a collection of sampling lines which intersect the attainment surfaces across the full range of the Pareto frontier.

If MEAs are run r times, each algorithm will return r attainment surfaces, one from each run. Having these r attainment surfaces, some from algorithm A and some from algorithm B , a *single sampling line* yields r points of intersection, one for each surface. These intersections form a univariate distribution, and therefore, we can perform standard non-parametric statistical procedures to determine whether or not the intersections for one of the algorithms occurs closer to the origin with some statistical significance. Such statistical tests have been performed by Knowles and Corne (2000) for each of the lines covering the Pareto trade-off area. Insofar as the lines provide a uniform sampling of the Pareto surface, the result of this analysis yields two numbers: a percentage of the surface in which algorithm A outperforms algorithm B with statistical significance, and that when algorithm B outperforms algorithm A .

Knowles and Corne (2000) presented their results of a comparison in the form of a pair $[a, b]$, where a gives the percentage of the space (i.e., the percentage of lines) on which algorithm A was found statistically superior to B , and b gives the similar percentage for algorithm B . Typically, if both A and B are 'good,' then $a + b < 100$. The quantity $[100 - (a + b)]$, of course, gives the percentage of the space on which the results were statistically inconclusive. They use statistical significance at the 95% confidence level. Knowles and Corne (2000) also extended their comparison methodology to comparing more than two algorithms.

If the algorithms are competitive, the results of the statistical test may vary with the number of sampling lines drawn since the procedure considers only the intersection points of sampling lines and attainment surfaces. Knowles and Corne (2000) proposed that 100 lines should be adequate, although, obviously, more lines the better. They have shown experimentally that the percentage of the space $(a + b)$ increases, to give statistically significant results, with the increased number of lines.

Differential Evolution

DE is a branch of evolutionary algorithms developed by Storn and Price (1995) for optimization problems over continuous domains. In DE, each variable's value in the chromosome is represented by a real number. The approach works by creating a random initial population of potential solutions, where it is guaranteed, by some repair rules, that the value of each variable is within its boundaries. An individual is then selected at random for replacement and three different individuals are selected as parents. One of these three individuals is selected as the main parent.

With some probability, each variable in the main parent is changed but at least one variable should be changed. The change is undertaken by adding to the variable's value a ratio of the difference between the two values of this variable in the other two parents. In essence, the main parent's vector is perturbed by the other two parents' vectors. This process represents the crossover operator in DE. If the resultant vector is better than the one chosen for replacement, it replaces it; otherwise the chosen vector for replacement is retained in the population. Therefore, DE differs from GA in a number of points:

1. DE uses real number representation while conventional GA uses binary, although GA sometimes uses integer or real number representation as well.
2. In GA, two parents are selected for crossover and the child is a recombination of the parents. In DE, three parents are selected for crossover and the child is a perturbation of one of them.
3. The new child in DE replaces a randomly selected vector from the population only if it is better than it. In conventional GA, children replace the parents with some probability regardless of their fitness.

In DE, a solution, l , in generation k is a multi-dimensional vector $\vec{x}^l_{G=k} = (x^l_1, \dots, x^l_N)^T$. A population, $P_{G=K}$, at generation $G=k$ is a vector of M solutions ($M > 4$). The initial population, $P_{G=0} = \{\vec{x}^1_{G=0}, \dots, \vec{x}^M_{G=0}\}$, is initialized as

$$x^l_{i,G=0} = \text{lower}(x_i) + \text{rand}_i[0,1] \times (\text{upper}(x_i) - \text{lower}(x_i)), \quad l = 1, \dots, M, \\ i = 1, 2, \dots, N$$

where M is the population size, N is the solution's dimension, and each variable x_i in a solution vector l in the initial generation $G=0$, $x^l_{i,G=0}$, is initialized within its boundaries ($\text{lower}(x_i), \text{upper}(x_i)$). Selection is carried out to select four different solutions indices and $j \in [1, M]$. The values of each variable in the child are changed with some crossover probability, CR , to:

$$\forall i \leq N, x'_{i,G=K} = \begin{cases} x^{r_3}_{i,G=K-1} + F \times (x^{r_1}_{i,G=K-1} - x^{r_2}_{i,G=K-1}) & \text{if } (\text{random}[0,1] < CR \vee i = i_{\text{rand}}) \\ x^j_{i,G=K-1} & \text{otherwise} \end{cases}$$

where $F \in (0, 1)$ is a problem parameter representing the amount of perturbation added to the main parent. The new solution replaces the old one if it is better than it and at least one of the variables should be changed. The latter is represented in the algorithm by randomly selecting a variable, $i_{\text{rand}} \in (1, N)$. After crossover, if one

or more of the variables in the new solution are outside their boundaries, the following repair rule is applied until the boundary constraints are satisfied:

$$x'_{i,G=k} = \begin{cases} \frac{x_{i,G} + \text{lower}(x_i)}{2} & \text{if } x_{i,G+1}^j < \text{lower}(x_i) \\ \text{lower}(x_i) + \frac{x_{i,G}^j - \text{upper}(x_i)}{2} & \text{if } x_{i,G+1}^j > \text{upper}(x_i) \\ x_{i,G+1}^j & \text{otherwise} \end{cases}$$

ADIFFERENTIAL EVOLUTION ALGORITHM FOR MOPS

A generic version of the adopted algorithm is presented at the end of this chapter in Figure 3 with the following modifications:

1. The initial population is initialized according to a Gaussian distribution $N(0.5, 0.15)$.
2. The step-length parameter F is generated from a Gaussian distribution $N(0, 1)$.
3. Reproduction is undertaken only among non-dominated solutions in each generation.
4. Offspring are placed into the population if they dominate the main parent.
5. The boundary constraints are preserved either by reversing the sign if the variable is less than 0 or keep subtracting 1 if it is greater than 1 until the variable is within its boundaries.

The algorithm works as follows. An initial population is generated at random from a Gaussian distribution with mean 0.5 and standard deviation 0.15. All dominated solutions are removed from the population. The remaining non-dominated solutions are retained for reproduction. If the number of non-dominated solutions exceeds some threshold, a distance metric relation (Abbass, Sarker & Newton, 2001) is used to remove those parents who are very close to each other. Three parents are selected at random. A child is generated from the three parents and is placed into the population if it dominates the first selected parent; otherwise a new selection process takes place. This process continues until the population is completed.

A maximum number of non-dominated solutions in each generation was set to 50. If this maximum is exceeded, the following nearest neighborhood distance function is adopted:

$$D(x) = \frac{(\min \|x - x_i\| + \min \|x - x_j\|)}{2},$$

where $x \neq x_i \neq x_j$. That is, the nearest neighborhood distance is the average Euclidean distance between the closest two points. The non-dominated solution with the smallest neighborhood distance is removed from the population until the total number of non-dominated solutions is retained at 50.

EXPERIMENTS

Test Problems

The algorithm is tested using the following two benchmark problems from Zitzler and Thiele (1999):

Test Problem 1: Convex function

$$f_1(x) = x_1$$

$$f_2(x) = g \times (1 - \sqrt{\frac{f_1}{g}})$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^n x_i)}{(n-1)}$$

$$x_i \in [0,1], i = 1, \dots, 30$$

Test Problem 2: Discontinuous function

$$f_1(x) = x_1$$

$$f_2(x) = g * (1 - \sqrt{\frac{f_1}{g}} - (\frac{f_1}{g}) \sin(10\pi f_1))$$

$$g = 1 + 9 \times \frac{(\sum_{i=2}^n x_i)}{(n-1)}$$

$$x_i \in [0,1], i = 1, \dots, 30$$

Both test problems contain two objective functions and 30 variables. The computational results of these test problems are provided in the next section.

Experimental Setup

For our algorithm, the initial population size is set to 100 and the maximum number of generations to 200. Twenty different crossover rates changing from 0 to 1.00 with an increment of 0.05 are tested without mutation. The initial population is initialized according to a Gaussian distribution $N(0.5, 0.15)$. Therefore, with high probability, the Gaussian distribution will generate values between $0.5 \pm 3 \times 0.15$ which fits with the variables' boundaries. If a variable's value is not within its boundary, a repair rule is used to repair the boundary constraints. The repair rule is applied simply to truncate the constant part of the value; therefore if, for example, the value is 3.3, the repaired value will be 0.3 assuming that the variable is between 0 and 1. The step-length parameter F is generated for each variable from a Gaussian distribution $N(0, 1)$. The algorithm is written in standard C++ and run on a Sun Sparc 4.

Experimental Results and Discussions

In this section, the solutions of two test problems, provided by our PDE algorithm, are compared with the solutions of 12 other MEAs (FFGA, HLGA, NPGA, NSGA, RAND, SOEA, SPEA, VEGA, PAES, PAES20, PAES98 and PAES98mut3p) using a statistical comparison technique. The results of other algorithms, except PAESs, were obtained from the website <http://www.tik.ee.ethz.ch/~zitzler/testdata.html>. The results for all PAESs were obtained from <http://www.rdg.ac.uk/~ssr97jdk/multi/PAES.html>.

To perform the statistical analysis using the Knowles and Corne (2000) method, we used the solutions of 20 runs of the DE algorithm for each crossover rate. The results of the comparison are presented in the form of a pair $[a, b]$ for each crossover rate, where a gives the percentage of the space (i.e., the percentage of lines) on which PDE algorithm is found statistically superior to the other, and b gives the similar percentage for the other algorithm. For example, for test problem 1, the best result using PDE $[84.3, 15.1]$ is achieved with crossover rate 0.15 when compared to SPEA. This means, our algorithm outperforms SPEA on about 84.3% of the Pareto surface whereas SPEA is statistically superior than our algorithm for 15.1%. For problem 2, the best result is obtained with crossover 0.05 when compared to SPEA.

In Figures 1 and 2, the x-axis represents the crossover rate used in our PDE algorithm and the y-axis is the percentage of superiority. Each figure contains a plot of “ a ” for our PDE algorithm and “ b ” for one of the other existing algorithms for a

Figure 1: Test problem 1

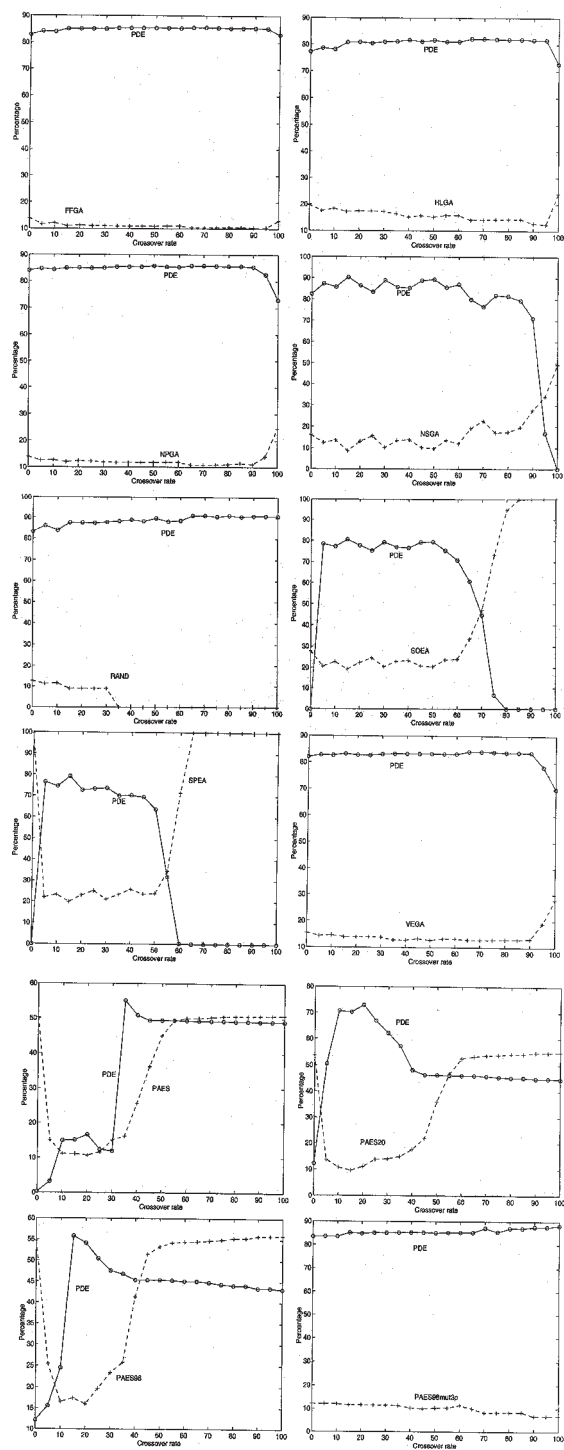
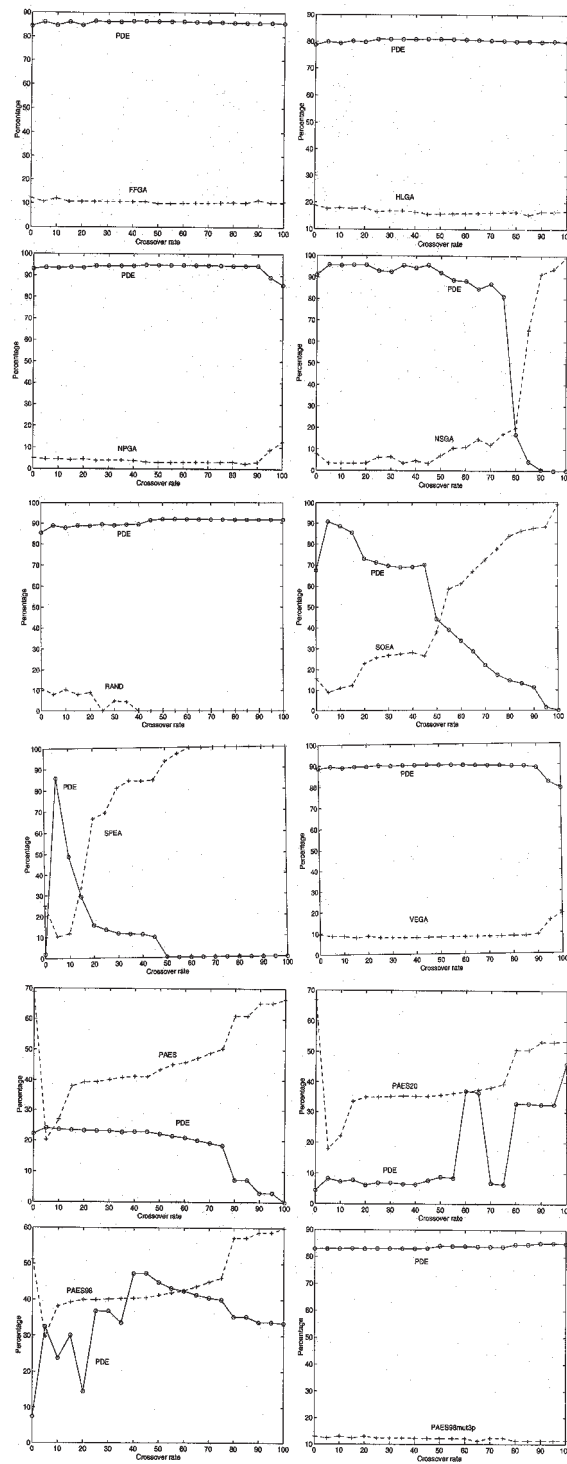


Figure 2: Test problem 2



given problem. Twelve plots in Figure 1 show the comparison of PDE with each of the other MEAs for test problem 1, and Figure 2 shows the same for test problem 2.

For both test problems, PDE is significantly better than FFGA, HLGA, NPGA, RAND and VEGA irrespective of the crossover rate. PDE is much better than NSGA for any crossover rate less than 0.85 for problem 1 and 0.8 for problem 2. PDE is superior than SOEA within the crossover rate 0.05 to 0.65 and SPEA within 0.05 to 0.5 for test problem 1. These figures for test problem 2 are 0 to 0.45 and 0.05 to 0.1 respectively. PDE is clearly better than PAES, PAES98 and PAES98mut3p for both test problems within a certain range of crossover rate. Although PDE shows superiority over PAES20 for test problem 1, it shows very little success for test problem 2. For test problem 1, a range of crossover rate for PDE can successfully challenge all other MEAs. For example, the solution of PDE at a crossover rate of 0.35 outperforms all other algorithms. From these results, it can be stated that no algorithm (out of 12) produces optimal solutions. However, PDE solutions could be close to the Pareto frontier though there is no guarantee. For problem 2, there is no single crossover rate for which PDE is superior than all the other MEAs. However such a rate can be found when we exclude one or two MEAs. That means, no one is close to optimal although PDE outperforms most algorithms.

CONCLUSIONS AND FUTURE RESEARCH

In this chapter, a novel differential evolution approach is discussed for multi-objective optimization problems. The approach generates a step by mutation, where the step is randomly generated from a Gaussian distribution. We tested the approach on two benchmark problems and it was found that our approach outperformed almost all existing MEAs. We also experimented with different crossover and mutation rates, on these two test problems, to find their best solutions. The crossover rates are found to be sensitive when compared with certain MEAs. However, a trend was found which suggests that a large number of non-dominated solutions were found with low-crossover rates. In future work, we intend to test the algorithm on more problems.

Also, the parameters chosen in this chapter were generated experimentally. It would be interesting to see the effect of these parameters on the problem.

REFERENCES

- Abbass, H., Sarker, R. & Newton, C. (2001). A Pareto differential evolution approach to vector optimization problems. *Congress on Evolutionary Computation*, 2, 971-978.
- Coello Coello, C. (1999). A comprehensive survey of evolutionary-based multi-objective optimization techniques. *Knowledge and Information Systems I* (3), 269-308.
- Fonseca, C. & Fleming, P. (1993). Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, 416-423.
- Goldberg, D. (1989). *Genetic Algorithms: In Search, Optimisation and Machine Learning*. Addison Wesley.
- Hajela, P. & Lin, C. (1992). Genetic search strategies in multi-criterion optimal design. *Structural Optimization*, 4, 99-107.
- Horn, J., Nafpliotis, N. & Goldberg, D. (1994). A niched Pareto genetic algorithm for multi-objective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1, 82-87.
- Knowles, J. & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for multi-objective optimization. *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington DC, IEEE Service Centre, 98-105.
- Knowles, J. & Corne, D. (2000). Approximating the non-dominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2), 149-172.
- Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, 93-100.
- Srinivas, N. & Deb, K. (1994). Multi-objective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation* 2(3), 221-248.
- Storn, R. & Price, K. (1995). Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012*, International Computer Science Institute, Berkeley.
- Veldhuizen, D. V. & Mamont, G. (1999). Multi-objective evolutionary algorithm test suites. *Proceedings of the 1999 ACM Symposium on Applied Computing*, San Antonio, Texas, ACM, 351-357.
- Zitzler, E. & Thiele, L. (1999). Multi-objective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257-271.

APPENDIX: THE PARETO DIFFERENTIAL EVOLUTION ALGORITHM

Figure 3: The Pareto frontier Differential Evolution Algorithm (PDE)

Let G denote a generation, P a population of size M , and $\vec{x}_{G=k}^j$ the j^{th} individual of dimension N in population P in generation k , and CR denotes the crossover probability

input $N, M \geq 4, \alpha, CR \in [0,1]$, and initial bounds: $\text{lower}(x_i), \text{upper}(x_i), i = 1, \dots, N$

initialize

$$P_{G=0} = \{\vec{x}_{G=0}^1, \dots, \vec{x}_{G=0}^M\}$$

for each individual $j \in P_{G=0}$

$$x_{i,G=0}^j = \text{Gaussian}(0.5, 0.15), i = 1, \dots, N$$

repair $\vec{x}_{G=k}^j$ if any variable is outside its boundaries

end for each

evaluate $P_{G=0}$

$k = 1$

while the stopping criterion is not satisfied **do**

remove all dominated solutions from $P_{G=k-1}$

if the number of non-dominated solutions in $P_{G=k-1} > \alpha$, **then** apply the neighborhood rule

for $j = 0$ to number of non-dominated solutions in $P_{G=k-1}$ $\vec{x}_{G=k}^j \leftarrow \vec{x}_{G=k-1}^j$

while $j \leq M$

randomly select $r_1, r_2, r_3 \in (1, \dots, \alpha)$, from the non-dominated solutions of

$P_{G=k-1}$, where $r_1 \neq r_2 \neq r_3$

randomly select $i_{rand} \in (1, \dots, N)$

forall $i \leq N, x_{i,G=k}^j =$

$$\begin{cases} x_{i,G=k-1}^{r_3} + \text{Gaussian}(0,1) \times (x_{i,G=k-1}^{r_1} - x_{i,G=k-1}^{r_2}) & \text{if } (\text{random}[0,1] < CR \wedge i = i_{rand}) \\ x_{i,G=k-1}^j & \text{otherwise} \end{cases}$$

end forall

Repair $\vec{x}_{G=k}^j$ if any variable is outside its boundaries

If \vec{x}' dominates $\vec{x}_{G=k-1}^{r_3}$ **then**

$$\vec{x}_{G=k}^j \leftarrow \vec{x}'$$

$$j = j + 1$$

end if

end while

$k = k + 1$

end while

return the set of non-dominated solutions.

Chapter XIV

Flexible Job-Shop Scheduling Problems: Formulation, Lower Bounds, Encoding and Controlled Evolutionary Approach

Imed Kacem, Slim Hammadi and Pierre Borne
Laboratoire d'Automatique et Informatique de Lille, France

ABSTRACT

The Job-shop Scheduling Problem (JSP) is one of the hardest problems; it is classified NP-complete (Carlier & Chretienne, 1988; Garey & Johnson, 1979). In the most part of cases, the combination of goals and resources can exponentially increase the problem complexity, because we have a very large search space and precedence constraints between tasks. Exact methods such as dynamic programming and branch and bound take considerable computing time (Carlier, 1989; Djerid & Portmann, 1996). Front to this difficulty, meta-heuristic techniques such as evolutionary algorithms can be used to find a good solution. The literature shows that they could be successfully used for combinatorial optimization such as wire routing, transportation problems, scheduling problems, etc. (Banzhaf, Nordin, Keller & Francone, 1998; Dasgupta & Michalewicz, 1997).

In this chapter we deal with the problem of flexible JSP which presents two difficulties: the first one is the assignment of each operation to a machine, and the second one is the scheduling of this set of operations in order to minimize a global criterion defined by a combination of many criteria (the makespan, the workload of the critical machine and the total workload of the machines). Practical and theoretical aspects of this problem are presented and carefully studied. Then we describe the state of the art concerning scheduling problems and evolutionary techniques. The evaluation function will be constructed by combination of the criteria and the corresponding lower bounds. The resolution method is based on many original direct chromosome representations. Also, based on practical examples, we present the efficiency of the suggested approach and some discussions about this research work.

INTRODUCTION

Several problems in various industrial environments are combinatorial. This is the case of numerous scheduling and planning problems. Generally, it is extremely difficult to solve this type of problem in its general form. Scheduling can be defined as a problem of finding the optimal sequence to execute a set of operations respecting the different problem's constraints. The problem set is extremely difficult to solve, it consists generally in a simultaneous optimization of a set of conflicting and concurrent goals. Therefore, the exact methods such as branch and bound, dynamic programming and constraint logic programming need a lot of time to find an optimal solution. So, we expect to find not necessary the optimal solution, but a good one to solve the problem. New search techniques such as genetic algorithms (Banzhaf, Nordin, Keller & Francone, 1998), simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), tabu search (Glover, Taillard & De werra, 1993) are able to reach our aim: find near optimal solutions for a wide range of combinatorial optimization problems.

In this work, we propose a new controlled evolutionary approach for solving a JSP and we describe the incorporation of the scheduling specific knowledge in the genetic operators and in the different chromosome representations.

This chapter is organized as follows: the first section presents the formulation of our flexible job-shop scheduling problem. In the second section, we present the lower bounds and we construct a global fitness function. Some definitions and a short description of genetic and evolutionary algorithms are presented in the third section. In the section four, the different codings and the implemented operators of the proposed methodology are described. Finally, the experimental results, discussions and conclusions are presented in the last section.

PROBLEM FORMULATION

The structure of the flexible job-shop scheduling problem can be described as follows:

- Set of N jobs $\{J_j\}_{1 \leq j \leq N}$, these jobs are independent of each other.
- Each job J_j represents a number of n_j ordered operations.
- The execution of each operation $O_{i,j}$ requires a resource or machine selected from a set of machines, $U = \{M_k\}_{1 \leq k \leq M}$. M is the total number of machines existing in the shop, this implies the existence of the assignment problem.
- For each job, the order of operations is fixed (precedence constraints).
- All machines are available at $t = 0$ and each job J_j can be started at $t = r_j$.
- There is a predefined set of processing times; for a given machine M_k and a given operation $O_{i,j}$, the processing time is defined and called $d_{i,j,k}$.
- An operation which has started runs to completion (non-preemption condition).
- Each machine can perform operations one after the other (resource constraints).
- To each operation $O_{i,j}$, we associate an earliest starting time $r_{i,j}$ calculated by the following formula:

$$r_{1,j} = r_j \quad \forall 1 \leq j \leq N \quad \text{and} \quad r_{i+1,j} = r_{i,j} + \gamma_{i,j} \quad \forall 1 \leq i \leq n_j - 1, \forall 1 \leq j \leq N$$

where $\gamma_{i,j} = \min_k (d_{i,j,k})$,

- For each flexible job-shop problem, we can associate a table D of processing times as in the following example:

$$D = \{d_{i,j,k} \in \mathbb{N}^* \mid 1 \leq j \leq N; 1 \leq i \leq n_j; 1 \leq k \leq M\}.$$

		$M1$	$M2$	$M3$	$M4$
$J1$	$O1,1$	1	3	4	1
	$O2,1$	3	8	2	1
	$O3,1$	3	5	4	7
$J2$	$O1,2$	4	1	1	4
	$O2,2$	2	3	9	3
	$O3,2$	9	1	2	2
$J3$	$O1,3$	8	6	3	5
	$O2,3$	4	5	8	1

The flexible job-shop scheduling problems present two difficulties. The first one is to assign each operation $O_{i,j}$ to a machine M_k (selected from the set U). The second one is the computation of the starting times $t_{i,j}$ and the completion times $tf_{i,j}$ of each operation $O_{i,j}$.

The considered objective is to globally minimize the following criteria:

- The makespan: $C_{r1} = \max_j (f_{n,j})$.
- The workload of the most loaded machine: $C_{r2} = \max_k (W_k)$ (W_k is the workload of M_k).
- The total workload of machines: $C_{r3} = \sum_k W_k$.

LOWER BOUNDS

Theorem

C_{r1}^* , C_{r2}^* and C_{r3}^* are respectively lower bounds for the considered criteria C_{r1} , C_{r2} and C_{r3} , where:

$$C_{r1}^* = \max \left(\max_j \left(r_j + \sum_i \gamma_{i,j} \right), \tilde{E} \left(\frac{R_M + \sum_j \sum_i \gamma_{i,j}}{M} \right), \theta_{k0, \tilde{N}} \right)$$

$$C_{r2}^* = \max \left(\tilde{E} \left(\frac{\sum_j \sum_i \gamma_{i,j}}{M} \right), \delta_{k0, \tilde{N}} \right) \text{ and } C_{r3}^* = \sum_j \sum_i \gamma_{i,j}$$

The others variables are defined as follows:

- \tilde{E} is a numerical function defined as follows:

if x is integer, $\tilde{E}(x) = x$, else $\tilde{E}(x) = E(x) + 1$ with $E(x)$ is the integer part of x

- $\tilde{N} = \tilde{E} \left(\frac{N_t}{M} \right)$ with $N_t = \sum_{j=1}^{j=N} n_j$
- $\delta_{k0, \tilde{N}} = \min_k \left(D_{\tilde{N}}^k \right)$ with $D_{\tilde{N}}^k$ is the sum of the \tilde{N} shortest processing times of the operations that we can execute on the machine M_k
- R_M is the sum of the M little values of the starting times ($r_{i,j}$)
- $\theta_{k0, \tilde{N}} = \min_k \left[\min_{1 \leq z \leq N_t - \tilde{N} + 1} \left(r_{i_z, j_z} + d_{i_z, j_z, k} + \min_{C_{z, \tilde{N}} \in E_{z, \tilde{N}}} \left[\Delta^k \left(C'_{z, \tilde{N}} \right) \right] \right) \right]$
- $\Delta^k \left(C'_{z, \tilde{N}} \right)$ is the sum of the processing times of the operations of $C'_{z, \tilde{N}}$ on M_k

- $C'_{z, \tilde{N}}$ is an element of $E'_{z, \tilde{N}}$ and $E'_{z, \tilde{N}}$ is the set of the combinations of $(\tilde{N}-1)$ operations chosen among the $(N_t - z)$ operations of V_z
- V_z is a part of the operations set defined as follows:

$$V_z = \left\{ O_{i_{z+1}, j_{z+1}}, O_{i_{z+2}, j_{z+2}}, \dots, O_{i_{N_t}, j_{N_t}} \right\} \text{ for } z \in \left[1, N_t - \tilde{N} + 1 \right]$$

where $\left\{ r_{i_1, j_1}, r_{i_2, j_2}, \dots, r_{i_{N_t}, j_{N_t}} \right\}$ are ranged in the ascending order.

→ **Proof:** see previous work (Kacem, Hammadi & Borne, 2002).

Fitness Function

So, we can reduce the multi-objective optimization to the minimization of the following global criterion (w_q is the importance weight of the criterion Cr_q):

$$C_g = \sum_{1 \leq q \leq 3} w_q \frac{Cr_q}{Cr_q^*} \text{ (with } \sum_{1 \leq q \leq 3} w_q = 1 \text{)}$$

This formulation is inspired of a fuzzy evaluation technique presented in a previous work (Kacem, Hammadi, & Borne 2001, a). In the ideal case, we obtain $C_g = 1$ if $Cr_q = Cr_q^* \forall 1 \leq q \leq 3$.

GENETIC AND EVOLUTIONARY ALGORITHMS: THE STATE OF THE ART

Evolutionary algorithms are general-purpose search procedures based on the mechanisms of natural selection and genetic evolution. These algorithms are applied by many users in different areas of engineering framework, computer science and operation research. Current evolutionary approaches included evolutionary programming, evolutionary strategies, genetic algorithms and genetic programming (Banzhaf, Nordin, Keller & Francone, 1998; Dasgupta & Michalewicz, 1997; Quagliarella, Périaux, Poloni & Winter, 1998; Goldberg, 1989; Fonseca & Fleming, 1998).

What Genetic Algorithms Are

Genetic algorithms enable to make evolve an initial set of solutions to a final set of solutions bringing a global improvement according to a criterion fixed at the beginning (Quagliarella, Périaux, Poloni & Winter, 1998). These algorithms function with the same usual genetic mechanisms (crossover, mutation, selection, etc.). In the genetic algorithms, the solutions set is called population. Each population is constituted of chromosomes whose each represents a particular coding of a solution. The chromosome is constituted of a sequence of genes that can take some values called alleles. These values are taken from an alphabet that has to be judiciously chosen to suit the studied problem. The classic coding corresponds to the binary alphabet: $\{0, 1\}$. In this case, the chromosome represents simply a finished table of 0 and 1. The operators that intervene in the genetic algorithms are selection, crossover and mutation.

A genetic algorithm is an algorithm that represents a special architecture. It operates on data without using preliminary knowledge on the problem processed. In fact, it consists of the following stages:

- The genesis: it's the generation phase of the initial population.
- The evaluation: in this stage, we compute the value of criterion for each individual of the current population.
- The selection: after the evaluation, we choose better adapted elements for the reproduction phase.
- The reproduction: we apply genetic operators (crossover, mutation...) on the selected individuals.
- The test: in this phase, we evaluate the improvement and decide if the solution is efficient. If the criterion reaches a satisfactory value, we take the current solution. If the result is insufficient, we return to the second stage and we repeat the same process until reaching the maximal iterations number.

Encoding Requirements

The implementation difficulty of these algorithms is in the conception of the gene content in order to describe all data of the problem and to represent the solutions. Choosing a good representation is a main stage of solving any optimization problem. However, choosing a good representation for a problem is as difficult as choosing a good search algorithm to solve it. Care must be taken to adopt both representational schemes and the associated genetic operators for an efficient genetic search.

Problems of encoding have been observed in the genetic algorithms literature (Dasgupta & Michalewicz, 1997). Traditionally, chromosomes are simple binary vectors. This simple representation is an excellent choice for the problems in which

solutions can be represented by lists of zeros and ones. Unfortunately, this approach cannot usually be used for real-world engineering problems such as a combinatorial one (Portmann, 1996). Many modifications should be made, such as the permutation of a basic string like that used for a Travelling Salesman Problem (Della Croce, Tadei & Volta, 1995). An illegal solution can obviously be obtained by applying traditional genetic operators (crossover and mutation). Some different encodings are proposed in the literature (Baghi, Uckun, Miyab & Kawamura, 1991; Uckun, Baghi & Kawamura, 1993; Bruns, 1993). The encoding is presented in two categories. The first one is the direct chromosome representation; we can represent a scheduling problem by using the schedule itself as a chromosome; this method generally requires developing specific genetic operators. The second one is the indirect chromosome representation; the chromosome does not directly represent a schedule, and transition from the chromosome representation to a legal schedule decoder is needed.

Concerning evolutionary algorithms and flexible job-shop scheduling problems, the literature presents many interesting propositions. Some of them can be used to solve the considered optimization problem. As examples, we have chosen to present the following codings:

1) PMR (Parallel Machine Representation) (Mesghouni, 1999)

The chromosome is a list of machines placed in parallel. For each machine, we associate operations to execute. Each operation is coded by three elements:

- i : the operation index
- J_j : the corresponding job
- t_{ij} : starting time of O_{ij} on the corresponding machine M_k

$M 1$	$(i, J_j, t_{i,j})$...
$M 2$...
$M 3$	$(i', J_{j'}, t_{i',j'})$	
....		
$M n$

This representation is based on the Parallel Machine Encoding (PME) which represents directly feasible schedules, gives all the necessary information to the foreman, and also enables us to treat the assignment problem. But it represents many difficulties to be implemented (Mesghouni, 1999). In fact, by using this representation, it is possible to obtain illegal solutions. Then, a corrective algorithm is needed. Unfortunately, these corrections increase the computation time and reduce the representation efficiency.

2) *PJsR (Parallel Jobs Representation) (Mesghouni, Hammadi & Borne, 1997)*

The chromosome is represented by a list of jobs. Each job is represented by the correspondent row where each place is constituted of two terms. The first term represents the machine that executes the operation. The second term represents the corresponding starting time (see the following figure).

$J1$	$(M_1, t_{1,1})$	$(M_2, t_{2,1})$...
$J2$	$(M_5, t_{1,2})$	$(M_1, t_{2,2})$	$(M_2, t_{3,2})$
$J3$			
...			
Jn	

This representation is a direct encoding which permits us to solve some problems met in the first encoding such as illegal solution (schedule) after a crossover operation and the creation of the first population (Mesghouni, 1999). This encoding integrates the precedence constraints, consequently we can create randomly the first population, and the genetic operators are very simple and give a feasible schedule, but we will see in the fifth section that this coding has a limited exploration capacity of the search space compared to other possible codings.

Mesghouni (1999) has proposed crossover and mutation operators for the two precedent chromosome representations, but, they are completely based on the exchanging of assignment informations and are not able to deal with the problem part of the tasks sequencing.

Portmann (1996) has presented other interesting coding possibilities for scheduling problems with or without assignment. As an example, Portmann et al. have proposed to use “ternary permutation matrix” with an “Assignment Vector.” But, these codings are not specified for the flexible JSP.

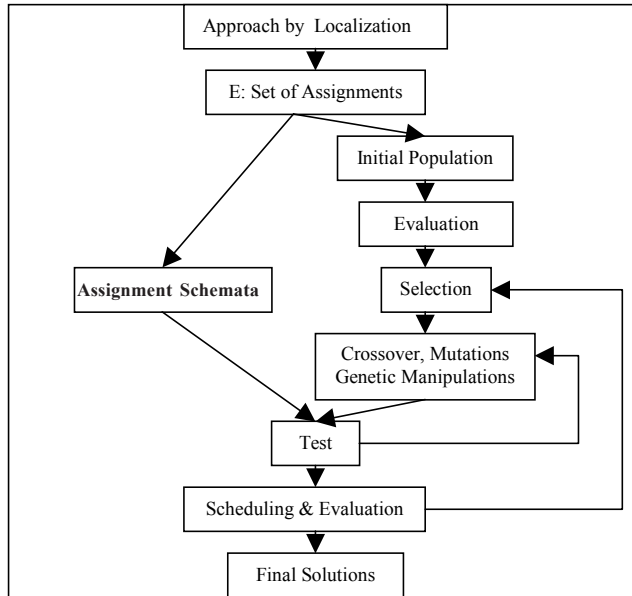
In this paper, we have chosen to use a direct representation to give conviviality and legibility to a chromosome and simplicity of use for a foreman. We suggest three new direct chromosome representations with their genetic operators.

CONTROLLED EVOLUTIONARY APPROACH

In this section, we present a new controlled evolutionary approach to solve the flexible job-shop scheduling problems.

The first stage of this method makes it possible to solve the problem of resources allocation and to build an ideal assignments model (assignments schemata).

Figure 1: Controlled evolutionary approach



The second stage is an evolutionary approach controlled by the assignment model (generated in the first step). In this approach, we apply advanced genetic manipulations in order to enhance solution quality and accelerate the convergence.

In the next paragraphs, we explain in details the different stages of this approach (see Figure 1).

First Stage: Approach by Localization

1) Resolution of the assignment problem

In order to solve this problem, the Approach by Localization (AL) is based on a set of assignment heuristics. These heuristics enable us to assign each operation to the suitable machine, taking into account the processing times and workloads of machines on which we have already assigned operations (Kacem, Hammadi & Borne, 2001, c). The obtained solutions can be represented in a table with the same size that the processing times table as for the following example:

$$S = \{S_{i,j,k} \in \mathbb{N}^* \mid 1 \leq j \leq N; 1 \leq i \leq n_j; 1 \leq k \leq M\}$$

		M1	M2	M3	M4
J 1	O 1,1	0	0	0	1
	O 2,1	0	0	0	1
	O 3,1	1	0	0	0
J 2	O 1,2	0	1	0	0
	O 2,2	1	0	0	0
	O 3,2	0	1	0	0
J 3	O 1,3	0	0	1	0
	O 2,3	0	0	0	1

Each case $S_{i,j,k}$ of the assignment S can take 0 or 1 as value:

- $S_{i,j,k}=1$, means that $O_{i,j}$ is assigned to the machine M_k .
- $S_{i,j,k}=0$, means that $O_{i,j}$ is not assigned to the machine M_k .

The AL enables us to construct a set of good assignments in balancing the machines' workloads (Kacem, Hammadi & Borne, 2001, c). We note E the set of these assignments: $E = \{S^z, \text{ such as } 1 \leq z \leq \text{cardinal}(E)\}$.

2) Resolution of the scheduling problem

The resolution of this problem is based on a modular algorithm called “Scheduling Algorithm” which calculates the starting times $t_{i,j}$ by taking into account the availabilities of the machines and the precedence constraints. The conflicts are solved by applying traditional rules of priority (SPT, LPT, FIFO, LIFO, FIRO... (Boucon, 1991), thus, we obtain a schedule set according to the applied priority rules (Kacem, Hammadi & Borne, 2001, c). This set will represent the initial population used by the Controlled Evolutionary Algorithm (see Figure 1).

3) Generation of an assignment model

The AL enables us to construct a set E of assignments in minimizing the sum of machines' workloads. The idea is to generate, from the set E , an assignment schemata that will serve us to control the genetic algorithm. This schemata is going therefore to represent a constraint which new created individuals must respect. The construction of this schemata consists of collecting the assignments S^z ($1 \leq z \leq \text{cardinal}(E)$) given by the AL and to determine (for each operation) the set of possible machines according to a procedure called “Schemata Generation Algorithm” (Kacem, Hammadi & Borne, 2001, 6). As an example, for the following Job-shop problem D (with total flexibility), we obtain the schemata S^{ch} as follows:

$$D = \{d_{i,j,k} / 1 \leq j \leq N; 1 \leq i \leq n_j; 1 \leq k \leq M\}$$

		M1	M2	M3	M4
J_1	$O_{1,1}$	1	3	4	1
	$O_{2,1}$	3	8	2	1
	$O_{3,1}$	3	5	4	7
J_2	$O_{1,2}$	4	1	1	4
	$O_{2,2}$	2	3	9	3
	$O_{3,2}$	9	1	2	2
J_3	$O_{1,3}$	8	6	3	5
	$O_{2,3}$	4	5	8	1



$$S^{ch} = \{S_{i,j,k}^{ch} / 1 \leq j \leq N; 1 \leq i \leq n_j; 1 \leq k \leq M\}$$

		M1	M2	M3	M4
J_1	$O_{1,1}$	*	0	0	*
	$O_{2,1}$	0	0	*	*
	$O_{3,1}$	*	0	*	0
J_2	$O_{1,2}$	0	*	*	0
	$O_{2,2}$	*	*	0	*
	$O_{3,2}$	0	*	*	*
J_3	$O_{1,3}$	0	0	1	0
	$O_{2,3}$	0	0	0	1

The value “ $S_{i,j,k}^{ch}=0$ ” indicates that the assignment of the operation $O_{i,j}$ to the machine M_k is *forbidden*. The value “ $S_{i,j,k}^{ch}=1$ ” indicates that the assignment of the operation $O_{i,j}$ to the machine M_k is *obligatory*, in this case, all values of the other

elements of the row (i, j) are inevitably equal to “0”. The symbol: “*” indicates that the assignment is *possible*, in this case, we cannot have the value “1” for the other elements of the row (i, j) .

In conclusion, this schemata covers the majority of the interesting assignment possibilities and avoids expensive prohibitions in terms of machine workloads (Kacem, Hammadi & Borne, 2001, b).

4) Results given by the AL

The results show that the AL enables us to construct solutions as interesting as solutions obtained using the classic genetic method (Mesghouni, Hammadi & Borne, 1997) or the Temporal Decomposition (Chetouane, 1995). The large advantage of this method is the important reduction of the computation time. In fact, the assignment procedures localize most interesting zones of the search space. Thus the scheduling is increasingly easy and becomes more efficient.

In general, the solutions of the previously evoked approach are acceptable and satisfactory. Therefore, it is worthwhile to investigate possible gains from the Controlled Evolutionary Algorithm which can be used to produce appropriate solutions for our problem while the other techniques do not guarantee the optimality of the final solution.

Remark: Case of a partial flexibility: in this case, some tasks can only be executed on a part of the available machines set. In the following example, the symbol “X” indicates that the assignment is impossible:

		M1	M2	M3	M4
J 1	O 1,1	1	X	4	1
	O 2,1	X	X	2	1
	O 3,1	3	5	4	7
J 2	O 1,2	4	1	X	X
	O 2,2	X	3	9	3
	O 3,2	9	X	2	X
J 3	O 1,3	8	6	3	5
	O 2,3	4	X	X	1

According to some authors (Mesghouni, Hammadi & Borne, 1997), this constraint is going to make the problem more difficult, complicate the search space and increase the computation time. But, in Kacem, Hammadi and Borne (2001, c), we show that our assignment procedures are applicable too in this case and we have shown the equivalence between the two problems.

Second Stage: Controlled Evolutionary Approach (CEA)

In this stage, we apply an advanced evolutionary approach on the initial solution set given by the AL. This approach is based on the application of the

schemata theorem. It consists of conceiving a model of chromosomes that suits the problem. This model is going to serve us in the construction of new individuals in order to integrate the good properties contained in the schemata. The objective is to make genetic algorithms more efficient and more rapid in constructing the solution by giving the priority to the reproduction of individuals respecting the model generated by the schemata and not from the whole set of chromosomes (Kacem, Hammadi & Borne, 2001, b).

In the case of scheduling problems, the implementation of this technique necessitates to elaborate a particular coding that could have described the problem data and exploited the schemata theorem that we propose in the next paragraph.

1) Modeling

In this paragraph, we present three direct chromosome representations suitable for the considered problem:

- a) Coding 1: Operations-Machines Coding (OMC)** (Kacem, Hammadi & Borne, 2001, b): it consists to represent the schedule in the same assignment table S . We replace each place $S_{i,j,k}=1$ by the couple $(t_{i,j}, tf_{i,j})$ where $t_{i,j}$ is the starting time and $tf_{i,j}$ is the completion time. The places $S_{i,j,k}=0$ are unchanged. As an example, the following schedule S is a possible solution of the job-shop problem D (already presented in the second section):

		$M1$	$M2$	$M3$	$M4$
J_1	$O_{1,1}$	0	0	0	0, 1
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	3, 6	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	1, 3	0	0	0
	$O_{3,2}$	0	3, 4	0	0
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

Remark: We use the same example D to explain the different genetic operators in the next paragraphs.

- b) Coding 2: List Operations Coding (LOC)** (Kacem, Hammadi & Borne, 2001, a): it consists to represent the schedule in a 3 columns-table. In the first column we put the operations list ($O_{i,j}$). The second indicates the machine M_k selected to execute the operation $O_{i,j}$ and the third column is reserved for the starting and completion times $(t_{i,j}, tf_{i,j})$. The precedent example can be represented in LOC as follows on the next page:

O_{ij}	M_k	t_{ij}, tf_{ij}
$O_{1,1}$	4	0, 1
$O_{2,1}$	4	1, 2
$O_{3,1}$	1	3, 6
$O_{1,2}$	2	0, 1
$O_{2,2}$	1	1, 3
$O_{3,2}$	2	3, 4
$O_{1,3}$	3	0, 3
$O_{2,3}$	4	3, 4

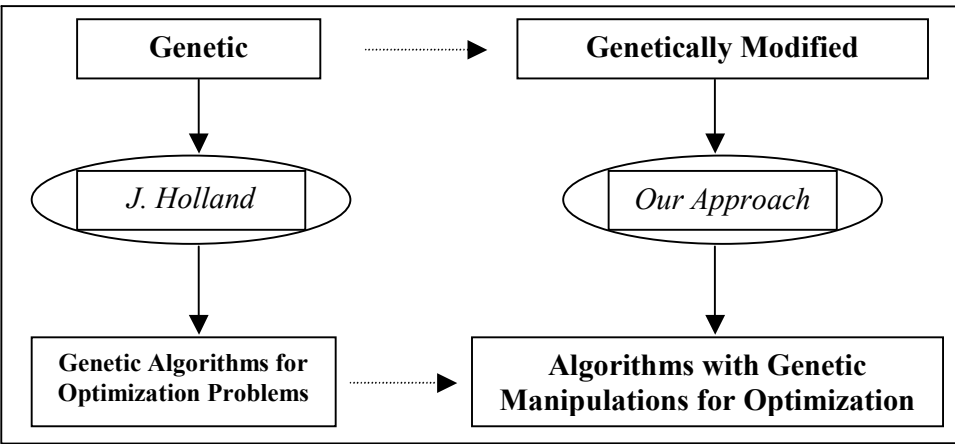
c) **Coding 3: Jobs Sequencings List Coding (JSLC)** (Kacem, Hammadi & Borne, 2001,c): it consists in representing the schedule in a table with z columns, ($z = \text{Max}(n_j)$). Each column will represent a jobs sequencing in the form of an N -cells list. Each cell is coded in the following way: (j, k, t_{ij}, tf_{ij}) :

Task 1	...	Task i	...	Task z
3, 3, 0, 3				
2, 2, 0, 1				
1, 1, 0, 1		j, k, t_{ij}, tf_{ij}		
....			

As an example, the precedent schedule can be presented in JSLC as follows:

Task 1	Task 2	Task 3
1, 4, 0, 1	1, 4, 1, 2	2, 2, 3, 4
2, 2, 0, 1	3, 4, 3, 4	1, 1, 3, 6
3, 3, 0, 3	2, 1, 1, 3	*****

Figure 2: Genetic manipulations algorithm



2) Genetic Manipulations, Crossover and Mutation Operators

- a) **Genetic Manipulations Operators:** these operators of mutation present a new way of application of the evolutionary algorithms: its the way of “genetic manipulations.” In genetic biology, these manipulations enable us to generate GMO (Genetically Modified Organisms). Our method is inspired by this principle and intervenes in the construction phase of the new chromosomes by applying the “artificial manipulations” in order to accelerate the convergence and insure a high quality of final solutions (see Figure 2).
- Manipulation reducing the Effective Processing Time ($\zeta_j = \sum_i \sum_K S_{i,j,k} \cdot d_{i,j,k}$) of a job J_j :

Manipulation 1	
- Select randomly an individual S_i ;	
- Choose the job J_j whose Effective Processing Time is the greatest;	
- $i=1$; $r=0$;	
- WHILE ($i \leq n_j$ And $r=0$)	
• Find K_0 such that $S_{i,j,K_0}=I$;	
• FOR ($k=1, k \leq M$)	
IF ($d_{i,j,k} < d_{i,j,k_0}$) Then $\{S_{i,j,K_0}=0; S_{i,j,k}=I; r=I\}$	
End IF	
End FOR	
• $i=i+1$;	
End WHILE	
- Calculate starting and completion times according to the algorithm "Scheduling Algorithm."	

Example: In this example S_i , the job J_1 has the greatest value of the ζ_j ($\zeta_1 = 6$ units of time). We have therefore to cover the list of its operations to reduce this duration. The operation $O_{2,1}$, can be assigned to the machine M_4 instead of the machine M_3 (because $d_{2,1,4} < d_{2,1,3}$), and thereafter we reduce the ζ_1 to 5 units of time and the makespan to 6 units instead of 8. So, we obtain the schedule S_j .

<i>S: before manipulation</i>					
		<i>M1</i>	<i>M2</i>	<i>M3</i>	<i>M4</i>
J_1	$O_{1,1}$	0	0	0	0, 1
	$O_{2,1}$	0	0	3, 5	0
	$O_{3,1}$	5, 8	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	0	1, 4	0	0
	$O_{3,2}$	0	0	0	4, 6
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

<i>S_j: after manipulation</i>					
		<i>M1</i>	<i>M2</i>	<i>M3</i>	<i>M4</i>
J_1	$O_{1,1}$	0	0	0	0, 1
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	2, 5	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	0	1, 4	0	0
	$O_{3,2}$	0	0	0	4, 6
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

The same manipulation can be applied for the JSLC (Kacem, Hammadi & Borne, 2001, c). In this case, the “Scheduling Algorithm” must be used without priority rule. For the same example, we obtain the following results:

<i>before manipulation</i>			<i>after manipulation</i>		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, 0, 1	3, 4, 3, 4	2, 4, 4, 6	1, 4, 0, 1	3, 4, 3, 4	2, 4, 5, 7
2, 2, 0, 1	1, 3, 3, 5	1, 1, 5, 8	2, 2, 0, 1	1, 4, 4, 5	1, 1, 5, 8
3, 3, 0, 3	2, 2, 1, 4	*****	3, 3, 0, 3	2, 2, 1, 4	*****

The same manipulation can be completely applied for the LOC (Kacem, Hammadi & Borne, 2001, a). Using the same example, we obtain the following results:

<i>before manipulation</i>			<i>after manipulation</i>		
$O_{i,l}$	M_K	$t_{i,h} \quad t_{f,i}$	$O_{i,l}$	M_K	$t_{i,h} \quad t_{f,i}$
$O_{1,1}$	4	0, 1	$O_{1,1}$	4	0, 1
$O_{2,1}$	3	3, 5	$O_{2,1}$	4	1, 2
$O_{3,1}$	1	5, 8	$O_{3,1}$	1	2, 5
$O_{1,2}$	2	0, 1	$O_{1,2}$	2	0, 1
$O_{2,2}$	2	1, 4	$O_{2,2}$	2	1, 4
$O_{3,2}$	4	4, 6	$O_{3,2}$	4	4, 6
$O_{1,3}$	3	0, 3	$O_{1,3}$	3	0, 3
$O_{2,3}$	4	3, 4	$O_{2,3}$	4	3, 4

- Manipulation balancing workloads of machines $W_K = \sum_j \sum_i S_{i,j,k} \cdot d_{i,j,k}$:

Manipulation 2
<ul style="list-style-type: none"> - Select randomly an individual S; - Find the most loaded machine M_{k1}; - Find the less loaded machine M_{k2}; - Choose randomly an operation $O_{i,j}$ such that $S_{i,j,k1} = I$; - Assign this operation to the less loaded machine: $S_{i,j,k1} = 0$; $S_{i,j,k2} = I$; - Calculate the starting and completion times according to the algorithm "Scheduling Algorithm."

Example: In this example S' , the workload of the critical machine is $W_4 = 5$ units of time (M_4). The less loaded machine is M_1 ($W_1 = 3$ units). We suppose that the operation $O_{1,1}$ has randomly been chosen among operations executed on M_4 . This operation will be therefore assigned to M_1 . So, we obtain the schedule S_{m2} :

<i>S': before manipulation</i>		$M1$	$M2$	$M3$	$M4$
J_1	$O_{1,1}$	0	0	0	0, 1
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	2, 5	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	0	1, 4	0	0
	$O_{3,2}$	0	0	0	4, 6
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

<i>S_{m2}: after manipulation</i>		$M1$	$M2$	$M3$	$M4$
J_1	$O_{1,1}$	0, 1	0	0	0
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	2, 5	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	0	1, 4	0	0
	$O_{3,2}$	0	0	0	4, 6
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

Workloads are therefore balanced, and the two machines M_1 and M_4 work during the same working time $W_1 = W_4 = 4$ units of time. The same manipulation can be applied for the JSLC (Kacem, Hammadi & Borne, 2001, c). In this case, the “Scheduling Algorithm” must be used without priority rule. For the same example D , we can obtain the following schedule O^2 starting from O^1 :

O^1			O^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, 0, 1	3, 4, 3, 4	2, 4, 5, 7	1, 1, 0, 1	3, 4, 3, 4	2, 4, 5, 7
2, 2, 0, 1	1, 4, 4, 5	1, 1, 5, 8	2, 2, 0, 1	1, 4, 4, 5	1, 1, 5, 8
3, 3, 0, 3	2, 2, 1, 4	*****	3, 3, 0, 3	2, 2, 1, 4	*****

The same manipulation can be applied for the LOC (Kacem, Hammadi & Borne, 2001, a). For the same example, we obtain the following results:

<i>before manipulation</i>			<i>after manipulation</i>		
$O_{i,j}$	M_k	$t_{i,j}, tf_{i,j}$	$O_{i,j}$	M_k	$t_{i,j}, tf_{i,j}$
$O_{1,1}$	4	0, 1	$O_{1,1}$	1	0, 1
$O_{2,1}$	4	1, 2	$O_{2,1}$	4	1, 2
$O_{3,1}$	1	2, 5	$O_{3,1}$	1	2, 5
$O_{1,2}$	2	0, 1	$O_{1,2}$	2	0, 1
$O_{2,2}$	2	1, 4	$O_{2,2}$	2	1, 4
$O_{3,2}$	4	4, 6	$O_{3,2}$	4	4, 6
$O_{1,3}$	3	0, 3	$O_{1,3}$	3	0, 3
$O_{2,3}$	4	3, 4	$O_{2,3}$	4	3, 4

Remark: Other manipulations are derived of this considered one and used to enhance solutions quality. As an example, we can exchange assignment between the most loaded machine and another or between the less loaded machine and another.

- b) Crossover Operators:** These operators are conceived in order to explore the search space and to offer more diversity by exchanging information between two individuals.
- **OMC Assignment Crossover**

OMC Crossover Algorithm
<ul style="list-style-type: none"> - Select randomly 2 parents S^1 and S^2; - Select randomly 2 integers j and j' such that $j \leq j' \leq N$; - Select randomly 2 integers i and i' such that $i \leq n_j$ and $i' \leq n_{j'}$ (in the case where $j=j'$, $i \leq i'$); - The individual e^1 receives the same assignments from the parent S^1 for all operations between the rows (i,j) and (i',j'); - The remainder of assignments for e^1 is obtained from S^2; - The individual e^2 receives the same assignments from the parent S^2 for all operations between the row (i,j) and the row (i',j'); - The remainder of assignments for e^2 is obtained from S^1; - Calculate the starting and completion times according to the algorithm "Scheduling Algorithm."

Example: For S^1 and S^2 , we suppose that we have randomly chosen $j = I$, $j' = 2$, $i = 2$, $i' = 2$:

S^1 : first parent

		M1	M2	M3	M4
J_1	$O_{1,1}$	0	0	0	0, 1
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	3, 6	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	1, 3	0	0	0
	$O_{3,2}$	0	3, 4	0	0
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

S^2 : second parent

		M1	M2	M3	M4
J_1	$O_{1,1}$	0, 1	0	0	0
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	3, 6	0	0	0
J_2	$O_{1,2}$	0	0	0, 1	0
	$O_{2,2}$	1, 3	0	0	0
	$O_{3,2}$	0	3, 4	0	0
J_3	$O_{1,3}$	0	0	1, 4	0
	$O_{2,3}$	0	0	0	4, 5

Copying of assignments:

e^1 : in construction

		M1	M2	M3	M4
J_1	$O_{1,1}$?, ?	0	0	0
	$O_{2,1}$	0	0	0	?, ?
	$O_{3,1}$?, ?	0	0	0
J_2	$O_{1,2}$	0	?, ?	0	0
	$O_{2,2}$?, ?	0	0	0
	$O_{3,2}$	0	?, ?	0	0
J_3	$O_{1,3}$	0	0	?, ?	0
	$O_{2,3}$	0	0	0	?, ?

e^2 : in construction

		M1	M2	M3	M4
J_1	$O_{1,1}$	0	0	0	?, ?
	$O_{2,1}$	0	0	0	?, ?
	$O_{3,1}$?, ?	0	0	0
J_2	$O_{1,2}$	0	0	?, ?	0
	$O_{2,2}$?, ?	0	0	0
	$O_{3,2}$	0	?, ?	0	0
J_3	$O_{1,3}$	0	0	?, ?	0
	$O_{2,3}$	0	0	0	?, ?

Computation of starting and completion times:

e^1 : first offspring

		M1	M2	M3	M4
J_1	$O_{1,1}$	0, 1	0	0	0
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	3, 6	0	0	0
J_2	$O_{1,2}$	0	0, 1	0	0
	$O_{2,2}$	1, 3	0	0	0
	$O_{3,2}$	0	3, 4	0	0
J_3	$O_{1,3}$	0	0	0, 3	0
	$O_{2,3}$	0	0	0	3, 4

e^2 : second offspring

		M1	M2	M3	M4
J_1	$O_{1,1}$	0	0	0	0, 1
	$O_{2,1}$	0	0	0	1, 2
	$O_{3,1}$	3, 6	0	0	0
J_2	$O_{1,2}$	0	0	0, 1	0
	$O_{2,2}$	1, 3	0	0	0
	$O_{3,2}$	0	3, 4	0	0
J_3	$O_{1,3}$	0	0	1, 4	0
	$O_{2,3}$	0	0	0	4, 5

- OMC Vertical Assignment Crossover

It consists of choosing randomly 2 machines (M_{k1} and M_{k2}) and exchanging all assignments between the two selected machines. That means, if the operation O_{ij} has been already assigned to M_{k1} , then the same operation will be assigned to M_{k2} and vice versa.

Remark: We cannot apply the Vertical Crossover to LOC.

• LOC Assignment Crossover

The same crossover operator can be used for the LOC. Using the same example, we obtain the following results:

S^1 :			S^2 :			e^1 :			e^2 :		
O_{ij}	M_k	t_{ij}, t'_{ij}	O_{ij}	M_k	t_{ij}, t'_{ij}	O_{ij}	M_k	t_{ij}, t'_{ij}	O_{ij}	M_k	t_{ij}, t'_{ij}
$O_{1,1}$	4	0, 1	$O_{1,1}$	1	0, 1	$O_{1,1}$	1	0, 1	$O_{1,1}$	4	0, 1
$O_{2,1}$	4	1, 2	$O_{2,1}$	4	1, 2	$O_{2,1}$	4	1, 2	$O_{2,1}$	4	1, 2
$O_{3,1}$	1	3, 6	$O_{3,1}$	1	3, 6	$O_{3,1}$	1	3, 6	$O_{3,1}$	1	3, 6
$O_{1,2}$	2	0, 1	$O_{1,2}$	3	0, 1	$O_{1,2}$	2	0, 1	$O_{1,2}$	3	0, 1
$O_{2,2}$	1	1, 3	$O_{2,2}$	1	1, 3	$O_{2,2}$	1	1, 3	$O_{2,2}$	1	1, 3
$O_{3,2}$	2	3, 4	$O_{3,2}$	2	3, 4	$O_{3,2}$	2	3, 4	$O_{3,2}$	2	3, 4
$O_{1,3}$	3	0, 3	$O_{1,3}$	3	1, 4	$O_{1,3}$	3	0, 3	$O_{1,3}$	3	1, 4
$O_{2,3}$	4	3, 4	$O_{2,3}$	4	4, 5	$O_{2,3}$	4	3, 4	$O_{2,3}$	4	4, 5

• JSLC Sequencing Crossover

JSLC Sequencing Crossover Algorithm	
- Select randomly 2 parents O^1 and O^2 ;	
- Select randomly z integers $\{j_i \leq N, 1 \leq i \leq z\}$;	
- For ($1 \leq i \leq z$)	
Exchange the sequencing between the parents (corresponding to the task i) using the same way than 1OX crossover: j_i will represent the cut point;	
End For	
- The individual e^1 receives the same assignments from the parent O^1 for all operations;	
- The individual e^2 receives the same assignments from the parent O^2 for all operations;	
- Calculate the starting and completion times in applying the algorithm "Scheduling Algorithm" according to the sequencing lists (without priority rule).	

We consider the following examples and we suppose that $j_1=1, j_2=2, j_3=1$:

O^1			O^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, 0, 1	1, 4, 1, 2	2, 2, 3, 4	3, 3, 0, 3	2, 4, 1, 4	1, 1, 5, 8
2, 2, 0, 1	3, 4, 3, 4	1, 1, 3, 6	2, 2, 0, 1	3, 4, 4, 5	2, 2, 4, 5
3, 3, 0, 3	2, 1, 1, 3	*****	1, 1, 0, 1	1, 3, 3, 5	*****

Exchange of sequencings:

e^1			e^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, ?, ?, ?	1, ?, ?, ?	2, ?, ?, ?	3, ?, ?, ?	2, ?, ?, ?	1, ?, ?, ?
3, ?, ?, ?	3, ?, ?, ?	1, ?, ?, ?	1, ?, ?, ?	3, ?, ?, ?	2, ?, ?, ?
2, ?, ?, ?	2, ?, ?, ?	*****	2, ?, ?, ?	1, ?, ?, ?	*****

Copying of assignments:

e^1			e^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, ?, ?	1, 4, ?, ?	2, 2, ?, ?	3, 3, ?, ?	2, 4, ?, ?	1, 1, ?, ?
3, 3, ?, ?	3, 4, ?, ?	1, 1, ?, ?	1, 1, ?, ?	3, 4, ?, ?	2, 2, ?, ?
2, 2, ?, ?	2, 1, ?, ?	*****	2, 2, ?, ?	1, 3, ?, ?	*****

Computation of starting and completion times:

e^1			e^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, 0, 1	1, 4, 1, 2	2, 2, 3, 4	3, 3, 0, 3	2, 4, 1, 4	1, 1, 5, 8
3, 3, 0, 3	3, 4, 3, 4	1, 1, 3, 6	1, 1, 0, 1	3, 4, 4, 5	2, 2, 4, 5
2, 2, 0, 1	2, 1, 1, 3	*****	2, 2, 0, 1	1, 3, 3, 5	*****

- JSLC Sequencing and Assignment Crossover

JSLC Sequencing and Assignment Crossover Algorithm
- Select randomly 2 parents O^1 and O^2 ;
- Select randomly z integers $\{j_i \leq N, 1 \leq i \leq z\}$;
- For ($1 \leq i \leq z$)
Exchange the sequencings and assignments (the couples (j, k)) between the parents (corresponding to the task i) using the same way than 1OX crossover: j_i will represent the cut point;
End For
- Calculate the starting and completion times in applying the algorithm "Scheduling Algorithm" according to the sequencing lists (without priority rule).

Example: We consider the same precedent examples and we suppose that $j_1 = I$, $j_2 = 0$, $j_3 = I$:

Exchange of sequencings and assignments:

e^1			e^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, ?, ?	2, 4, ?, ?	2, 2, ?, ?	3, 3, ?, ?	1, 4, ?, ?	1, 1, ?, ?
3, 3, ?, ?	3, 4, ?, ?	1, 1, ?, ?	1, 4, ?, ?	3, 4, ?, ?	2, 2, ?, ?
2, 2, ?, ?	1, 3, ?, ?	*****	2, 2, ?, ?	2, 1, ?, ?	*****

Computation of starting and completion times:

e^1			e^2		
Task 1	Task 2	Task 3	Task 1	Task 2	Task 3
1, 4, 0, 1	2, 4, 1, 4	2, 2, 4, 5	3, 3, 0, 3	1, 4, 1, 2	1, 1, 3, 6
3, 3, 0, 3	3, 4, 4, 5	1, 1, 5, 8	1, 4, 0, 1	3, 4, 3, 4	2, 2, 4, 5
2, 2, 0, 1	1, 3, 3, 5	*****	2, 2, 0, 1	2, 1, 1, 3	*****

Remark: It has been demonstrated that 1OX crossover preserves the sequencing propriety (Portmann, 1996), that is why we choose to apply it in our problem.

c) **Random Mutation Operators:** these operators represent some random changes that can be applied on the solutions set in order to avoid premature convergence.

- Random Sequencing Mutation

This operator is based on exchanging two random chosen couples (j, k) of twocells belonging to the same column (we do not have to change the assignments of the considered individual) and computing the starting and completion times using the algorithm “Scheduling Algorithm” according to the sequencing lists (without priority rule). It can only be applied in the case of JSLC. The other codings are not adapted for this kind of transformation.

- Random Assignment Mutation

This operator is based on a random change that we apply on the assignment $S_{i,j,k}$ of a chosen operation O_{ij} on a machine M_k to another $M_{k'}$. After this mutation, we obtain $S_{i,j,k}=0$ and $S_{i,j,k'}=1$. The starting and completion times are calculated in applying “Scheduling Algorithm.” This operator is used with all the three proposed codings.

3) Remarks

The crossover probabilities are fixed in a traditional way ($P_c=0.90$). The remainder of probability is allocated for the random mutation and the genetic manipulations (with the same rate).

The used selection mechanism gives the priority to the reproduction of the best individuals according to the global criterion C_g . The criteria of stop are the following:

- The maximum number of the iterations is reached,
- The threshold of satisfaction is reached ($C_g < 1.1$).

COMPUTATIONAL EXPERIMENTS

Computational experiments are carried out to evaluate the efficiency of our method with a large set of representative problem instances based on practical data. The obtained results are summarized in the next paragraphs.

Encoding Performance

Through the description of the different codings and the obtained computational experiments, many conclusions related to the encoding performance can be made and are summarized in the following table.

Encoding performances			
	OMC	LOC	JSLC
Simplicity and significance	Simple	Simple	Difficult
Exploration of sequencings space	Very difficult: only a single possibility: the use of priority rules	Very difficult: only a single possibility: the use of priority rules	Very good
Exploration of assignments space	Good	Good	Good
Implementation	Easy	Easier	Difficult
Computation time	Correct	Correct	Correct but needs more time because of its complexity
Quality of the obtained solutions	Good	Good	Generally the best solutions

In fact, we can notice a great similarity between OMC and LOC. On the one hand, all genetic operators are equivalent and can be used for the two proposed codings. On the other hand, exploration assignment and sequencing search spaces have the same size too. The only difference is in the representation form. This difference give more simplicity and more exploration possibilities (vertical cross-over) for OMC.

Concerning JSLC, although it is relatively difficult to be designed and difficult to be implemented, this coding represents the most efficient representation. In fact, it presents the same possibilities of the exploration of the assignment space search and offers more possibilities to explore the sequencing one. It enables us to consider jointly or separately the assignment and the scheduling problems and avoid the limited use of the priority rules; that is why it generally gives the best results.

Solutions Quality

In this paragraph, we present numerous examples that we have simulated to test the approach efficiency. Also, we present the lower bounds values of the different criteria to give a clear idea of the solution quality. For reasons of representation simplicity, solutions are presented in LOC:

1) Example 1: (little size 4 jobs/12 operations/ 5 machines):

$r_1=3, r_2=5, r_3=1$ and $r_4=6$. $w_1=0.1, w_2=0.1$ and $w_3=0.8$.

Processing times table (Example 1)							Obtained solution			
		M1	M2	M3	M4	M5	O_{ij}	M_k	$t_{i,p}$	tf_{ij}
J 1	O 1,1	2	5	4	1	2	O 1,1	4	3, 4	
	O 2,1	5	4	5	7	5	O 2,1	2	9, 13	
	O 3,1	4	5	5	4	5	O 3,1	4	13, 17	
J 2	O 1,2	2	5	4	7	8	O 1,2	1	7, 9	
	O 2,2	5	6	9	8	5	O 2,2	5	9, 14	
	O 3,2	4	5	4	54	5	O 3,2	1	14, 18	
J 3	O 1,3	9	8	6	7	9	O 1,3	3	1, 7	
	O 2,3	6	1	2	5	4	O 2,3	2	7, 8	
	O 3,3	2	5	4	2	4	O 3,3	4	8, 10	
	O 4,3	4	5	2	1	5	O 4,3	4	17, 18	
J 4	O 1,4	1	5	2	4	12	O 1,4	1	6, 7	
	O 2,4	5	1	2	1	2	O 2,4	2	8, 9	
$C_{r1}^*=16$, $C_{r2}^*=7$ and $C_{r3}^*=32$							$Cr_1=18$, $Cr_2=8$ and $Cr_3=32$			

2) Example 2: (middle size 10 jobs/29 operations/ 7 machines):

$r_1=2, r_2=4, r_3=9, r_4=6, r_5=7, r_6=5, r_7=7, r_8=4, r_9=1$ and $r_{10}=0$.
 $w_1=0.511, w_2=0.322$ and $w_3=0.167$.

Processing times table (Example 2)									Obtained solution			
		M1	M2	M3	M4	M5	M6	M7	O_{ij}	M_k	t_{ij}	tf_{ij}
J1	O 1,1	1	4	6	9	3	5	2	O 1,1	1	2, 3	
	O 2,1	8	9	5	4	1	1	3	O 2,1	5	5, 6	
	O 3,1	4	8	10	4	11	4	3	O 3,1	7	12, 15	
J2	O 1,2	6	9	8	6	5	10	3	O 1,2	7	4, 7	
	O 2,2	2	10	4	5	9	8	4	O 2,2	1	10, 12	
	O 1,3	15	4	8	4	8	7	1	O 1,3	7	9, 10	
J3	O 2,3	9	6	1	10	7	1	6	O 2,3	3	10, 11	
	O 3,3	11	2	7	5	2	3	14	O 3,3	5	11, 13	
	O 1,4	2	8	5	8	9	4	3	O 1,4	1	6, 8	
J4	O 2,4	5	3	8	1	9	3	6	O 2,4	4	13, 14	
	O 3,4	1	2	6	4	1	7	2	O 3,4	5	14, 15	
	O 1,5	7	1	8	5	4	3	9	O 1,5	2	7, 8	
J5	O 2,5	2	4	5	10	6	4	9	O 2,5	1	8, 10	
	O 3,5	5	1	7	1	6	6	2	O 3,5	4	14, 15	
	O 1,6	8	7	4	56	9	8	4	O 1,6	3	5, 9	
J6	O 2,6	5	14	1	9	6	5	8	O 2,6	3	9, 10	
	O 3,6	3	5	2	5	4	5	7	O 3,6	3	11, 13	

Continued on next page

Example 2, continued from previous page

J 7	O 1,7	5	6	3	6	5	15	2	O 1,7	7	7, 9
	O 2,7	6	5	4	9	5	4	3		6	9, 13
	O 3,7	9	8	2	8	6	1	7		6	13, 14
J 8	O 1,8	6	1	4	1	10	4	3	O 1,8	2	4, 5
	O 2,8	11	13	9	8	9	10	8		4	5, 13
	O 3,8	4	2	7	8	3	10	7		2	13, 15
J 9	O 1,9	12	5	4	5	4	5	5	O 1,9	5	1, 5
	O 2,9	4	2	15	99	4	7	3		2	8, 10
	O 3,9	9	5	11	2	5	4	2		7	10, 12
J 10	O 1,10	9	4	13	10	7	6	8	O 1,10	2	0, 4
	O 2,10	4	3	25	3	8	1	2		6	4, 5
	O 3,10	1	2	6	11	13	3	5		1	12, 13
$C_{r1}^*=15$, $C_{r2}^*=9$ and $C_{r3}^*=60$									$Cr_1=15$, $Cr_2=11$ and $Cr_3=61$		

3) Example 3: (great size 15 jobs/56 operations/ 10 machines) is located on the next page. The result for this example is presented in the last of the current subsection.

Values of the different criteria show the efficiency of the controlled genetic algorithm. In fact, this method enables us to have good results in a polynomial computation times. This efficiency is explained by the judicious choice of the search zone (using the AL) and by the contribution of genetic manipulations in the optimization of solutions.

Although we cannot demonstrate the solution optimality, this method makes it possible to ensure a good threshold of satisfaction since its solutions are always very near to the optimal one.

Robustness of the Global Evaluation Function

In this paragraph, we show, by the following example, how the evaluation function yields satisfactory results according to the preferences of the decision-makers and their weights for each criterion.

Example: We deal with the same “Example 1” already presented in the precedent paragraph and we vary the preferences of the decision makers (we remind that the lower bounds are $C_{r1}^*=16$, $C_{r2}^*=7$ and $C_{r3}^*=32$), so we obtain the schedules found on page 257.

These obtained results show the robustness of the form of the proposed evaluation function. The choice of this function enables us to obtain satisfactory solutions ($C_g < 1.1$) according to the desired preferences and the associated weights.

Others Results

- In a previous work, the CEA has been compared to other methods like Temporal Decomposition (Chetouane, 1995) and Classic Genetic Algorithm

3)Example 3: (great size 15 jobs/56 operations/ 10 machines)

$r_1=5, r_2=3, r_3=6, r_4=4, r_5=9, r_6=7, r_7=1, r_8=2, r_9=8, r_{10}=0, r_{11}=14, r_{12}=13, r_{13}=11, r_{14}=12$ and $r_{15}=5$. $w_1=0.511, w_2=0.322$ and $w_3=0.167$.

Processing times table											Obtained solution			
		M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	O_{ij}	M_k	t_{ij}, t'_{ij}
J 1	O 1,1	1	4	6	9	3	5	2	8	9	4	O 1,1	1	15, 16
	O 2,1	1	1	3	4	8	10	4	11	4	3	O 2,1	2	16, 17
	O 3,1	2	5	1	5	6	9	5	10	3	2	O 3,1	3	18, 19
	O 4,1	10	4	5	9	8	4	15	8	4	4	O 4,1	9	19, 23
J 2	O 1,2	4	8	7	1	9	6	1	10	7	1	O 1,2	4	3, 4
	O 2,2	6	11	2	7	5	3	5	14	9	2	O 2,2	3	11, 13
	O 3,2	8	5	8	9	4	3	5	3	8	1	O 3,2	10	18, 19
	O 4,2	9	3	6	1	2	6	4	1	7	2	O 4,2	4	22, 23
J 3	O 1,3	7	1	8	5	4	9	1	2	3	4	O 1,3	7	15, 16
	O 2,3	5	10	6	4	9	5	1	7	1	6	O 2,3	9	16, 17
	O 3,3	4	2	3	8	7	4	6	9	8	4	O 3,3	2	20, 22
	O 4,3	7	3	12	1	6	5	8	3	5	2	O 4,3	4	23, 24
J 4	O 1,4	6	2	5	4	1	2	3	6	5	4	O 1,4	5	4, 5
	O 2,4	8	5	7	4	1	2	36	5	8	5	O 2,4	5	19, 20
	O 3,4	9	6	2	4	5	1	3	6	5	2	O 3,4	6	21, 22
	O 4,4	11	4	5	6	2	7	5	4	2	1	O 4,4	10	22, 23
J 5	O 1,5	6	9	2	3	5	8	7	4	1	2	O 1,5	9	9, 10
	O 2,5	5	4	6	3	5	2	28	7	4	5	O 2,5	6	16, 18
	O 3,5	6	2	4	3	6	5	2	4	7	9	O 3,5	2	18, 20
	O 4,5	6	5	4	2	3	2	5	4	7	5	O 4,5	4	20, 22
J 6	O 1,6	4	1	3	2	6	9	8	5	4	2	O 1,6	2	13, 14
	O 2,6	1	3	6	5	4	7	5	4	6	5	O 2,6	1	16, 17
	O 1,7	1	4	2	5	3	6	9	8	5	4	O 1,7	1	14, 15
	O 2,7	2	1	4	5	2	3	5	4	2	5	O 2,7	2	15, 16
J 7	O 1,8	2	3	6	2	5	4	1	5	8	7	O 1,8	7	2, 3
	O 2,8	4	5	6	2	3	5	4	1	2	5	O 2,8	8	17, 18
	O 3,8	3	5	4	2	5	49	8	5	4	5	O 3,8	4	18, 20
	O 4,8	1	2	36	5	2	3	6	4	11	2	O 4,8	1	20, 21
J 9	O 1,9	6	3	2	22	44	11	10	23	5	1	O 1,9	10	8, 9
	O 2,9	2	3	2	12	15	10	12	14	18	16	O 2,9	3	9, 11
	O 3,9	20	17	12	5	9	6	4	7	5	6	O 3,9	7	16, 20
	O 4,9	9	8	7	4	5	8	7	4	56	2	O 4,9	8	20, 24
J 10	O 1,10	5	8	7	4	56	3	2	5	4	1	O 1,10	10	0, 1
	O 2,10	2	5	6	9	8	5	4	2	5	4	O 2,10	8	15, 17
	O 3,10	6	3	2	5	4	7	4	5	2	1	O 3,10	10	17, 18
	O 4,10	3	2	5	6	5	8	7	4	5	2	O 4,10	1	21, 24
J 11	O 1,11	1	2	3	6	5	2	1	4	2	1	O 1,11	7	14, 15
	O 2,11	2	3	6	3	2	1	4	10	12	1	O 2,11	6	15, 16
	O 3,11	3	6	2	5	8	4	6	3	2	5	O 3,11	3	16, 18
	O 4,11	4	1	45	6	2	4	1	25	2	4	O 4,11	7	23, 24
J 12	O 1,12	9	8	5	6	3	6	5	2	4	2	O 1,12	8	13, 15
	O 2,12	5	8	9	5	4	75	63	6	5	21	O 2,12	5	15, 19
	O 3,12	12	5	4	6	3	2	5	4	2	5	O 3,12	6	19, 21
	O 4,12	8	7	9	5	6	3	2	5	8	4	O 4,12	7	21, 23
J 13	O 1,13	4	2	5	6	8	5	6	4	6	2	O 1,13	2	11, 13
	O 2,13	3	5	4	7	5	8	6	6	3	2	O 2,13	10	13, 15
	O 3,13	5	4	5	8	5	4	6	5	4	2	O 3,13	10	15, 17
	O 4,13	3	2	5	6	5	4	8	5	6	4	O 4,13	2	22, 24
J 14	O 1,14	2	3	5	4	6	5	4	85	4	5	O 1,14	1	12, 14
	O 2,14	6	2	4	5	8	6	5	4	2	6	O 2,14	9	14, 16
	O 3,14	3	25	4	8	5	6	3	2	5	4	O 3,14	8	18, 20
	O 4,14	8	5	6	4	2	3	6	8	5	4	O 4,14	5	22, 24
J 15	O 1,15	2	5	6	8	5	6	3	2	5	4	O 1,15	1	5, 7
	O 2,15	5	6	2	5	4	2	5	3	2	5	O 2,15	6	7, 9
	O 3,15	4	5	2	3	5	2	8	4	7	5	O 3,15	3	13, 15
	O 4,15	6	2	11	14	2	3	6	5	4	8	O 4,15	5	20, 22

$C_{r1}=23, C_{r2}=10$ and $C_{r3}=91$

$C_{r1}=24, C_{r2}=11$ and $C_{r3}=94$

(Mesghouni, 1999). In Kacem, Hammadi and Borne (2001, b), we can find others results about this comparison that confirm the efficiency of the suggested approach.

- Other results concerning the use of CEA for solving Parallel Machines Problems (in particular, the case of the flexible JSP) are presented in Kacem,

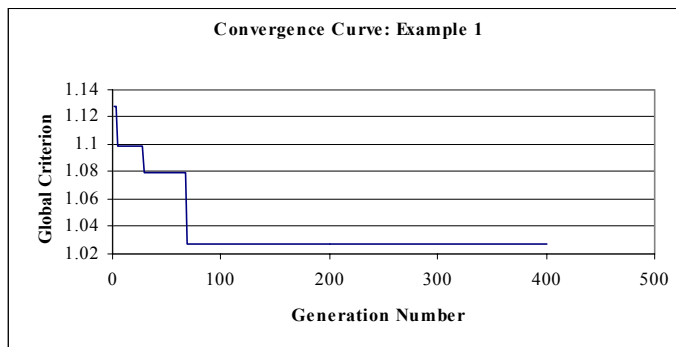
Schedules from the Example on page 255

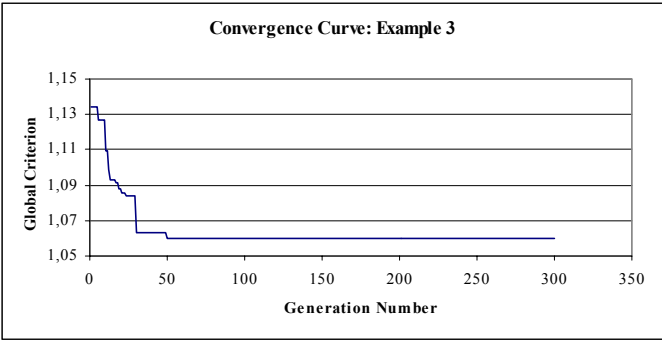
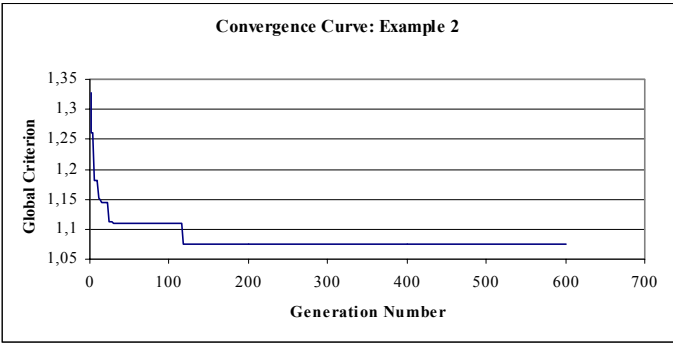
$w_1=0.1, w_2=0.5 \text{ and } w_3=0.4$			$w_1=0.8, w_2=0.2 \text{ and } w_3=0$		
$O_{i,j}$	M_k	t_{ijk}, ff_{ij}	$O_{i,j}$	M_k	t_{ijk}, ff_{ij}
$O_{1,1}$	5	3, 5	$O_{1,1}$	4	3, 4
$O_{2,1}$	2	9, 13	$O_{2,1}$	4	4, 11
$O_{3,1}$	4	13, 17	$O_{3,1}$	5	11, 16
$O_{1,2}$	1	7, 9	$O_{1,2}$	1	5, 7
$O_{2,2}$	5	9, 14	$O_{2,2}$	1	7, 12
$O_{3,2}$	1	14, 18	$O_{3,2}$	3	12, 16
$O_{1,3}$	3	1, 7	$O_{1,3}$	2	1, 9
$O_{2,3}$	2	7, 8	$O_{2,3}$	3	9, 11
$O_{3,3}$	4	8, 10	$O_{3,3}$	1	12, 14
$O_{4,3}$	4	17, 18	$O_{4,3}$	4	14, 15
$O_{1,4}$	1	6, 7	$O_{1,4}$	3	6, 8
$O_{2,4}$	2	8, 9	$O_{2,4}$	2	9, 10
$Cr_1=18, Cr_2=7 \text{ and } Cr_3=33$			$Cr_1=16, Cr_2=9 \text{ and } Cr_3=40$		

$w_1=0.6, w_2=0.1 \text{ and } w_3=0.3$			$w_1=0.79, w_2=0.01 \text{ and } w_3=0.2$		
$O_{i,j}$	M_k	t_{ijk}, ff_{ij}	$O_{i,j}$	M_k	t_{ijk}, ff_{ij}
$O_{1,1}$	4	3, 4	$O_{1,1}$	4	3, 4
$O_{2,1}$	5	4, 9	$O_{2,1}$	5	4, 9
$O_{3,1}$	4	10, 14	$O_{3,1}$	1	12, 16
$O_{1,2}$	1	5, 7	$O_{1,2}$	1	5, 7
$O_{2,2}$	1	7, 12	$O_{2,2}$	1	7, 12
$O_{3,2}$	3	12, 16	$O_{3,2}$	3	12, 16
$O_{1,3}$	3	1, 7	$O_{1,3}$	3	1, 7
$O_{2,3}$	2	7, 8	$O_{2,3}$	2	7, 8
$O_{3,3}$	1	12, 14	$O_{3,3}$	4	10, 12
$O_{4,3}$	4	14, 15	$O_{4,3}$	4	12, 13
$O_{1,4}$	4	6, 10	$O_{1,4}$	4	6, 10
$O_{2,4}$	2	10, 11	$O_{2,4}$	2	10, 11
$Cr_1=16, Cr_2=10 \text{ and } Cr_3=36$			$Cr_1=16, Cr_2=11 \text{ and } Cr_3=36$		

Hammadi and Borne (2001, a) and show the excellent performance of CEA: for these problems, we obtain $C_g \approx 1$ in the majority of cases.

- Concerning convergence speed, in all the tested numerical instances, the criterion of stop is obtained in a short computation time. At convergence, the number of iterations is few in most cases (< 200). This efficiency is explained by the reduction of the problem complexity (using the AL) and by the contribution of the genetic manipulations operators. As an example, we present the convergence curves of the instances already introduced in the current section (Example 1, Example 2 and Example 3):





CONCLUSION

In this chapter, we deal with one of the hardest combinatorial problems (the flexible JSP) and we propose a new evolutionary approach to solve it.

This approach is based on a controlled evolutionary optimization in which some efficient direct chromosome representations and advanced genetic operators are carefully chosen.

The multi-objective evaluation of the solutions quality is reduced to a single criterion that measures this quality according to the lower bound values of the different criteria. The theoretical formulas of these lower bounds are presented too.

The obtained results show the efficiency of the proposed approach. Although it does not guarantee the optimality, this approach provides good quality solutions in a reasonable time limit. Also, the general aspect of the considered formulation presents a large methodological advantage that makes it possible to solve other particular problems like Parallel Machines Problems.

The originality of this approach is in the application of a new biological concept in the optimization of computing problems. This concept concerns the Genetically

Modified Organisms; thus, we apply genetic manipulations to control the individual's evolution and reduce the blind aspect of classic genetic algorithm in order to accelerate the convergence and enhance the final solutions quality.

As future research direction, the study of the other multi-objective considerations in the global evaluation (like Pareto principle (Fonseca, & Fleming, 1998; Sarker, Abbas & Newton, 2001) seems an interesting subject which can enrich the proposed approach and give scientific benefits.

ACKNOWLEDGMENT

This work is integrated in the group “TACT” of a regional research program entitled “MOST” (research group on integrated manufacturing and man-machine systems). This program is supported by the “Conseil Régional du Nord Pas de Calais” and the “FEDER.” It involves several laboratories in the north of France. One of the goals of this program is to increase the competitiveness of industries by designing new tools and methods.

Also, we thank in particular Professor Jacques Carlier for giving us interesting references about his work related to lower bounds for scheduling problems.

REFERENCES

- Baghi, S., Uckun, S., Miyab, Y. & Kawamura, K. (1991). Exploring problem-specific recombination operators for job shop scheduling. *Proceedings of the 4th International Conference on Genetic Algorithms*. University of California, San-Diego, July 13-16, 10-17.
- Banzhaf, W., Nordin, P., Keller, R.E. & Francone, F.D. (1998). *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Application*. San Francisco, CA: Morgan Kaufmann.
- Boucon, D. (1991). *Ordonnancement d'Atelier: Aide au Choix de Règles de Priorité*. Ph D Thesis ENSAE, Toulouse, FRANCE.
- Bruns, R. (1993). Direct chromosome representation and advanced genetic operators for production scheduling. *Proceedings of the 5th International Conference on Genetic Algorithms*, University of Illinois at Urbana-Champaign, July 17-21, 352-359.
- Carlier, J. (1989). An algorithm for solving the job shop problem. *Management Science*, (35), 164-176.
- Carlier, J. & Chretienne, P. (1988). *Problèmes d'Ordonnancement: Modélisation / Complexité / Algorithmes*. Editions Masson.

- Chetouane, F. (1995). Ordonnancement d'atelier à tâches généralisées, perturbations, réactivité. *Rapport de DEA de l'Institut National Polytechnique de Grenoble*.
- Dasgupta, D. & Michalewicz, Z. (1997). *Evolutionary Algorithms in Engineering Applications*. Berlin: Springer-Verlag.
- Della Croce, F., Tadei, R. & Volta, G. (1995). A genetic algorithm for job shop problem. *Computers Ops Res*, 22(1), 15-24.
- Djerid, L. & Portmann, M-C. (1996). Genetic algorithm operators restricted to precedent constraints set: Genetic algorithm designs with or without branch and bound approach for solving scheduling problems with disjunctive constraints. *Proceedings of the International IEEE/SMC'96 Conference* October, 14-17 Pekin, China.
- Fonseca, C.M. & Fleming, P.J. (1998). Multi-objective optimization and multiple constraint handling with evolutionary algorithms—Part I: Unified formulation. *IEEE Trans/SMC, Part A*, 28(1), 26-37.
- Garey, M.R. & Johnson, D.S. (1979). *Computers and Intractability: A Guide to Theory of NP-Completeness*. New York: W.H. Freeman and Co.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.
- Golver, F., Taillard, E. & De werra, D. (1993). A user's guide to tabu search. *Annals of operations research*, (41), 3-28.
- Kacem, I., Hammadi, S. & Borne, P. (2001, b). Multi-objective optimization for flexible job-shops scheduling problem: Hybridization of genetic algorithms with fuzzy logic. *Proceedings of IFDICON European Workshop*, June 27-29, Santorini, Gr.
- Kacem, I., Hammadi, S. & Borne, P. (2001, b). Direct chromosome representation and advanced genetic operators for flexible job-shop problems. *Proceedings of CIMCA International Conference*, July 9-11, Las Vegas, Nevada, USA.
- Kacem, I., Hammadi, S. & Borne, P. (2001, c). Approach by localization and genetic manipulations algorithm for flexible job-shop problems. *Proceedings of International IEEE Conference on Systems, Man, and Cybernetics*, October 7-10, Tucson, Arizona, U.S.A. 2599-2604.
- Kacem, I., Hammadi, S., & Borne, P. (2002). Bornes inférieures pour les problèmes d'ordonnancement des job-shops flexibles. *CIFA '02*, 7-10 July 7-10, Nantes, Fr.
- Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680.

- Mesghouni, K. (1999). *Application des Algorithmes Évolutionnistes dans les Problèmes d'Optimisation en Ordonnancement de la Production*. PhD Thesis, Lille I University, Fr.
- Mesghouni, K., Hammadi, S. & Borne, P. (1997). Evolution Programs for Job-Shop Scheduling. *Proceedings of IEEE/SMC conference*, (1), 720-725. Orlando, FL.
- Portmann, M-C. (1996). Genetic algorithms and scheduling: A state of the art and some proposition. *Proceedings of the Workshop on Production Planning and Control*, September 9-11, Mons, Belgium, p i-xxiv.
- Quagliarella, D., Périaux, J., Poloni, C. & Winter, G. (1998). *Genetic Algorithms and Evolution Strategies in Engineering and Computer Sciences*. John Wiley & Sons.
- Sarker, R., Abbas, H.A. & Newton, C. (2001). Solving multi-objective optimization problems using evolutionary algorithm. *Proceedings of International CIMCA Conference*, July 9-11, Las Vegas, Nevada.
- Uckun, S., Baghi, S. & Kawamura, K. (1993). Managing genetic search in job-shop scheduling. *IEEE Expert*, 8(5), 15-24.

Chapter XV

The Effect of Multi-Parent Recombination on Evolution Strategies for Noisy Objective Functions

Yoshiyuki Matsumura, Kazuhiro Ohkura and Kanji Ueda
Kobe University, Japan

ABSTRACT

In this chapter we apply $(\mu / \mu, \lambda)$ -ES to noisy test functions, in order to investigate the effect of multi-parent versions of both intermediate recombination and discrete recombination. Among the many formulations of ES, we test three in particular; Classical-ES (CES), i.e., Schwefel's original ES (Schwefel, 1995, Bäck, 1996); Fast-ES (FES), i.e., Yao and Liu's extended ES (Yao & Liu, 1997); and Robust-ES (RES), i.e., our extended ES (Ohkura, 2001). Computer simulations are used to compare the performance of multi-parent versions of intermediate recombination and discrete recombination in CES, FES and RES. We saw that the performance of the $(\mu / \mu, \lambda)$ -ES algorithms depended on the particular objective functions. However, the FES and RES algorithms were seen to be improved by multi-parent versions of discrete recombination applied to both object parameters and strategy parameters.

INTRODUCTION

Noise is a common phenomenon in many real-world problems. For example, in the field of information engineering, any signal returned from the real world usually includes a significant amount of noise. Also in the field of Evolutionary Robotics (Harvey et al., 1997), simulation models are developed by taking noise into account in order to decrease the gap between simulated and real-world robot performance (Jacobi et al., 1995). In such cases, Evolutionary Algorithms (EAs) work well even in the presence of noise.

EAs have three main approaches, namely Evolutionary Programming (EP), Evolution Strategies (ES) and Genetic Algorithms (GAs). ES has several formulations (Schwefel, 1995, Bäck, 1996). $(\mu / \rho, \lambda)$ -ES is the general form for real-valued parameter optimization problems, in which μ parents generate λ offspring through recombination and mutation at each generation, and the best μ offspring are selected deterministically from the λ offspring to replace the current set of parents. ρ determines the number of parents to form one new offspring, with the case where $\rho > 2$ known as multi-recombination (Beyer, 2001).

In $(\mu / \rho, \lambda)$ -ES, Beyer (1995) theoretically investigated the case of $\rho = \mu$ for the sphere function, finding a λ -fold speedup compared to ESs without recombination. For ESs, each individual has a pair of real-valued vectors, i.e., the object parameters and strategy parameters, with strategy parameters roughly determining the size of mutation applied to object parameters. Beyer used recombination only on the object parameters, however it is necessary for ES researchers to investigate the effect of recombination on not only object parameters but also strategy parameters, both empirically and theoretically.

There are two popular recombination operators, namely intermediate recombination and discrete recombination. Many ES researchers (Bäck & Schwefel, 1993, Bäck & Eiben, 1998; Eiben & Bäck, 1998) often apply only intermediate recombination to strategy parameters due to Schwefel's general recommendations (Schwefel, 1995). However, Chang et al. (2001) experimentally investigated multi-parent versions of both intermediate recombination and discrete recombination on strategy parameters, and showed the advantages of not only intermediate recombination but also discrete recombination. They used 11 standard test functions and tested ES with Gaussian mutation, or Classical-ES (CES). However, the test functions they used did not incorporate noise. Thus we must investigate the performance of ESs with multi-parent recombination on noisy test functions in order to apply ESs to real world optimization problems.

In this chapter we apply $(\mu / \mu, \lambda)$ -ES to noisy test functions, in order to investigate the effect of multi-parent versions of both intermediate recombination and discrete recombination. Among the many formulations of ESs, we test three in

particular; CES, i.e., Schwefel's original ES (Schwefel, 1995; Bäck, 1996); Fast-ES (FES), i.e., Yao and Liu's extended ES (Yao & Liu, 1997); and Robust-ES (RES), i.e., our extended ES (Ohkura, 2001). Computer simulations of $(\mu/\mu, \lambda)$ -ES are conducted using both Gaussian and Cauchy mutation.

RELATED WORKS

Many types of ESs have been applied to noisy objective functions. Beyer (1993, 1998) analyzed the $(1, \lambda)$ -ES for the noisy sphere function. Bäck and Hammel (1994) and Hammel and Bäck (1994) empirically investigated the performance of the (μ, λ) -ES using discrete recombination on object parameters, and global intermediate recombination on strategy parameters. Nissen and Propach (1998) empirically compared the performance of point-based methods, e.g., Threshold Accepting and Pattern Search, with population-based methods, e.g., ES and GA. They employed the same ES as Bäck and Hammel (1994). Gruenz and Beyer (1999) investigated the $(\mu/\mu, \lambda)$ -ES for the noisy sphere function using both discrete and intermediate recombination on objective parameters, and intermediate recombination on strategy parameters. Arnold and Beyer (2001) investigated the $(\mu/\mu, \lambda)$ -ES for the noisy sphere function using intermediate recombination; while different kinds of recombination were applied to object parameters, only intermediate recombination was applied to strategy parameters.

In this chapter we empirically investigate the $(\mu/\mu, \lambda)$ -ES using both intermediate recombination and discrete recombination, applied to both object parameters and strategy parameters.

EVOLUTION STRATEGIES ALGORITHMS

Classical Evolution Strategies (CES)

The Classical ES (CES) algorithms adopted in this chapter are described as follows (Schwefel, 1995; Bäck, 1996):

1. Generate an initial population of $1/4$ individuals, and set $g=1$. Each individual is taken as a pair of real-valued vectors $(\mathbf{x}_i, \boldsymbol{\eta}_i)$, where \mathbf{x}_i and $\boldsymbol{\eta}_i$ are the i -th coordinate value in R and its strategy parameters (larger than zero), respectively.
2. Evaluate the objective value for each individual $(\mathbf{x}_i, \boldsymbol{\eta}_i)$ in the population, based on the objective function $f(\mathbf{x}_i)$.
3. Each parent $(\mathbf{x}_i, \boldsymbol{\eta}_i)$, $i=1, \dots, \mu$, creates λ/μ offspring on average, so that a total of λ offspring are generated. At that time, offspring are calculated as follows: for $i=1, \dots, \mu$, $j=1, \dots, n$, and $p=1, \dots, \lambda$

$$\eta p(j) = \eta i(j) \exp \{ \tau' N(0, 1) + \tau N_j(0, 1) \} \quad (1)$$

$$xp(j) = xi(j) + \eta p(j) N_j(0, 1) \quad (2)$$

where $xi(j)$, $xp(j)$, $\eta i(j)$ and $\eta p(j)$ denote the j -th component values of the vectors $\mathbf{x}i$, $\mathbf{x}p$, $\boldsymbol{\eta}i$ and $\boldsymbol{\eta}p$, respectively. $N(0, 1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0, 1)$ indicates that the random number is generated anew for each value of j . The factors τ and τ' are commonly set to constant (Bäck, 1996). Various types of recombination operators can also be applied before calculating Equations (1) and (2).

4. Calculate the fitness of each offspring ($\mathbf{x}'i$, $\boldsymbol{\eta}'i$), according to $f(\mathbf{x}'i)$.
5. Sort offspring ($\mathbf{x}'i$, $\boldsymbol{\eta}'i$) according to their fitness values, and select the μ best offspring out of λ to be parents of the next generation.
6. Stop if the halting criterion is satisfied; otherwise, $g = g + 1$ and go to Step 3.

Fast Evolution Strategies (FES)

Yao and Liu (1997) proposed the Fast ES (FES) algorithm variant of (μ, λ) -ES. In FES, the Gaussian mutation (Step 3 above) is replaced by Cauchy mutation, using the following Cauchy distribution function:

$$Ft(x) = 1/2 + (1/\pi) \arctan(x/t) \quad (3)$$

where $t=1$. The success of FES is explained as a result of a larger probability of escaping from local optima, due to the fatter convergence trails of the Cauchy mutation operator. In other words the Cauchy distribution has a higher probability than the Gaussian distribution of producing large mutations. Yao and Liu (1997) conducted empirical experiments using a number of test functions, demonstrating an improvement in performance especially on multi-modal problems.

Robust Evolution Strategies (RES)

When ESs are applied to an optimization problem successfully, the observed evolutionary dynamics show qualitatively similar behavior to that of other evolutionary algorithms: over generations the focus of the search shifts from global regions to smaller local regions. This arises from the gradual convergence of the population due to the direct effects of natural selection. Associated with this, the strategy parameters $\boldsymbol{\eta}i$ tend to zero. This is the process of “self-adaptation,” which is considered to be one of the major attractive features of ES. This may be useful for unimodal functions, however in many multi-modal functions, ESs are often trapped in local optima.

Robust-ES (RES) (Ohkura, 2001) was designed to avoid this problem of entrapment. The key idea of RES is to utilize selectively neutral mutations (Kimura, 1983) on strategy parameters so that the algorithm is capable of rapidly increasing or decreasing strategy parameters, irrespective of natural selection. RES follows the same procedure as CES or FES except for the following two points:

- A different individual representation is used, incorporating redundant strategy parameters, i.e., inactive strategy parameters, which have no effect on the selection process.
- Extra stochastic mutation mechanisms are used to change the original strategy parameters. These mutations replace, swap or copy active strategy parameters with inactive strategy parameters.

An individual \mathbf{X}_i is represented as follows, assuming that $i=1,2,\dots,\mu$, $j=1,2,\dots,n$, $k=0,1,\dots,m$:

$$\mathbf{X}_i = [\mathbf{x}_i, (\boldsymbol{\eta}_{i0}, \dots, \boldsymbol{\eta}_{ik}, \dots, \boldsymbol{\eta}_{im})] \quad (4)$$

$$\mathbf{x}_i = (x_i(1), \dots, x_i(j), \dots, x_i(n)) \quad (5)$$

$$\boldsymbol{\eta}_{ik} = (\eta_{ik}(1), \dots, \eta_{ik}(j), \dots, \eta_{ik}(n)) \quad (6)$$

where $x_i(j)$ and $\eta_{ik}(j)$ denote the j -th component values of the vectors \mathbf{x}_i and $\boldsymbol{\eta}_{ik}$, respectively. Note that each $\mathbf{x}_i(j)$ has $(m+1)$ strategy parameters.

We define D as same the mutation mechanism given in Equation (1). In addition, $\boldsymbol{\eta}_{ik}$ is modified stochastically, according to the following new mutation operators:

- *Odup* shifts all of $\eta_{ik}(j)$ into the adjacent position of $(k+1)$ and removes $\eta_{im}(j)$ from the list. Then, *Odup* mutates all $\boldsymbol{\eta}_{ik}$ with D .
- *Odel* discards $\eta_{i0}(j)$ and moves $\eta_{ik}(j)$ to the adjacent position of $(k-1)$. At the m -th position η_L is calculated as the smaller value either η_{max} or $\Sigma \eta_{ip}(j)$. Then, *Odel* mutates all $\boldsymbol{\eta}_{ik}$ with D .
- *Oinv* swaps $\eta_{i0}(j)$ with one of $\eta_{ik}(j)$, $k=1, \dots, m$ and mutates $\eta_{i0}(j)$ and $\eta_{ik}(j)$ with D .

Note that RES using Gaussian mutation is referred to as gRES, and RES using Cauchy mutation is referred to as cRES. When the probabilities of *Odup*, *Odel* and *Oinv* are set at 1.0, 0.0 and 0.0, gRES and cRES are equivalent to CES or FES, respectively.

Multi-Parent Recombination

Typically, recombination operators have been investigated empirically, due to their mathematical intractability. Traditional recombination operators reproduce one offspring using two parents, however more recent work tends to use the multi-parent versions of recombination operators. Where the multi-parent version of intermediate recombination is applied to both the object parameters and strategy parameters, the recombination operator is known as “Multi-Parent Intermediary Recombination.” Where the multi-parent version of discrete recombination is applied to both the object parameters and strategy parameters, it is known as “Multi-Parent Discrete Recombination” and “Global Combined Discrete Recombination” (Chang et al., 2001).

Multi-Parent Intermediary Recombination

Intermediate recombination is some kind of averaging across parent solutions. This can be formulated as follows:

$$\eta'_{ik}(j) = \frac{1}{\mu} \sum_{i=1}^{\mu} \eta_{ik}(j) \quad (7)$$

$$x'_{i}(j) = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i}(j) \quad (8)$$

Following Chang et al. (2001), this type of recombination is referred to as II.

Multi-Parent Discrete Recombination

In discrete recombination, the j -th component of the offspring is equal to the j -th component from a randomly selected parent:

$$\eta'_{ik}(j) = \eta_{yjk}(j) \quad (9)$$

$$x'_{i}(j) = x_{y'j}(j) \quad (10)$$

yj and $y'j$ denote uniformly distributed random integers in $\{1, \dots, \mu\}$, respectively, and are generated anew for each value of j . Following Chang et al. (2001), this type of recombination is referred to as DD.

Global Combined Discrete Recombination

Multi-parent discrete recombination is separately applied to object parameters and strategy parameters in the DD-ES introduced above. However, the two parameter sets may be strongly coupled to each other, as the strategy parameters determine the mutability of object parameters. Due to this possibility, a new recombination which regards a pair of an object parameter and a strategy parameter as a unit of recombination, can be formulated as follows:

$$\eta'_{ik}(j) = \eta_{yjk}(j) \quad (11)$$

$$x'_{ij}(j) = x_{yji}(j) \quad (12)$$

Following Chang et al. (2001), this type of recombination is referred to as D.

COMPUTER SIMULATION

Test Functions and Conditions

Following Bäck and Hammel (1994) and Hammel and Bäck (1994), we calculate the noisy objective function $F(\mathbf{x}_i)$ at each generation as follows:

$$F(\mathbf{x}_i) = f(\mathbf{x}_i) + \sigma Ni(0, 1) \quad (13)$$

where the test functions $f(\mathbf{x}_i)$ are Sphere Model (f_1), Ackley's Function (f_2) and Generalized Rastrigin's Function (f_3) (Table 1). All the test functions define 30 dimensional problems ($n=30$) with f_1 a unimodal functions, and f_2 and f_3 multimodal functions. $Ni(0, 1)$ denotes a normally distributed n -dimensional random number with mean zero and standard deviation one which is generated anew for each value of i . σ is the noise level, and set it at either 0.0, 0.001, 0.01, 0.1 or 1.0.

Table 1: Test functions

Functions ($n = 30, 100$)	(Range)
$f_1(x) = \sum_{i=1}^n x_i^2$	$(-100 \leq x_i \leq 100)$
$f_2(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i \right) + 20 + e$	$(-32 \leq x_i \leq 32)$
$f_3(x) = \sum_{i=1}^n \{x_i^2 - 10 \cos(2\pi x_i) + 10\}$	$(-5.12 \leq x_i \leq 5.12)$

The experimental setup is based on Yao and Liu (1997): $(\mu, \lambda) = (30, 200)$ with Gaussian mutation or Cauchy mutation, recombination and no correlated mutations. CES, FES and RES use the same initial populations, and all simulations are independently repeated for 50 runs. The upper bound of strategy parameters η_{max} is set at 3.0 for $f1$ and $f2$ and 1.0 for $f3$. In RES, the number of inactive strategy parameters m for each variable is set at 5. *Odup*, *Odel* and *Oinv* are applied with the probabilities of 0.6, 0.3 and 0.1, respectively. The main purpose of our computer simulations is to investigate the effect of multi-parent recombination on ESs for noisy objective function. Thus, the parameters are not fully tuned.

Results

The averaged best function values of CES, gRES, FES and cRES when applied to the sphere model $f1$ are shown in Figure 1 to Figure 4 for the five different noise levels, 0.0, 0.001, 0.01, 0.1 and 1.0. These results clearly demonstrate that the different multi-parent recombination operators have a different effect on the different types of ES. In Figure 1(a), CES does not find function values less than 1.0. In Figure 1(b), (c) and (d), II-CES, DD-CES and D-CES can find function values less than $1e-10$ even in the presence of noise. These results suggest that three multi-parent recombination operators, i.e., II, DD and D recombination, improve the performance of CES on $f1$. In Figure 2(a), gRES does not find function values less than 0.01 except when the noise level is either 0.1 or 1.0. In Figure 2(b), II-gRES does not evolve in the early generations. In Figure 2(c) and (d), DD-gRES and D-gRES can find function values less than $1e-10$. These results suggest that gRES on $f1$ prefers DD and D recombination to II recombination. In Figure 3 and Figure 4, FES and cRES show the same tendencies as gRES on $f1$, with both preferring DD and D recombination to II recombination.

The averaged best function values of CES, gRES, FES and cRES when applied to Ackley's function $f2$ are shown in Figure 5 to Figure 8 for the five different noise levels, 0.0, 0.001, 0.01, 0.1 and 1.0. The results are different from $f1$ for all noise levels. In Figure 5(a), CES does not find function values less than 1.0. In Figure 5(b), II-CES can find function values less than $1e-10$ except when the noise level is 1.0. In Figure 5(c) and (d), DD-CES and D-CES can find function values less than $1e-10$. These results suggest that CES on $f2$ prefers DD and D recombination to II recombination. In Figure 6, gRES on $f2$ shows the same tendencies as gRES on $f1$, preferring DD and D recombination to II recombination. In Figure 7(a) and (b), FES does not find function values less than 1.0. In Figure 7(c) and (d), DD-FES and D-FES can find function values less than $1e-10$ except when the noise level is 1.0. These results suggest that FES on $f2$ prefers DD and D recombination to II recombination. In Figure 8(a) and (b), cRES does not find function values less than 0.1. In Figure 8(c), DD-cRES can find function values less

Figure 1 and Figure 2: Averaged best results for $f1$ when the noise level is 0.0, 0.001, 0.01, 0.1 and 1.0

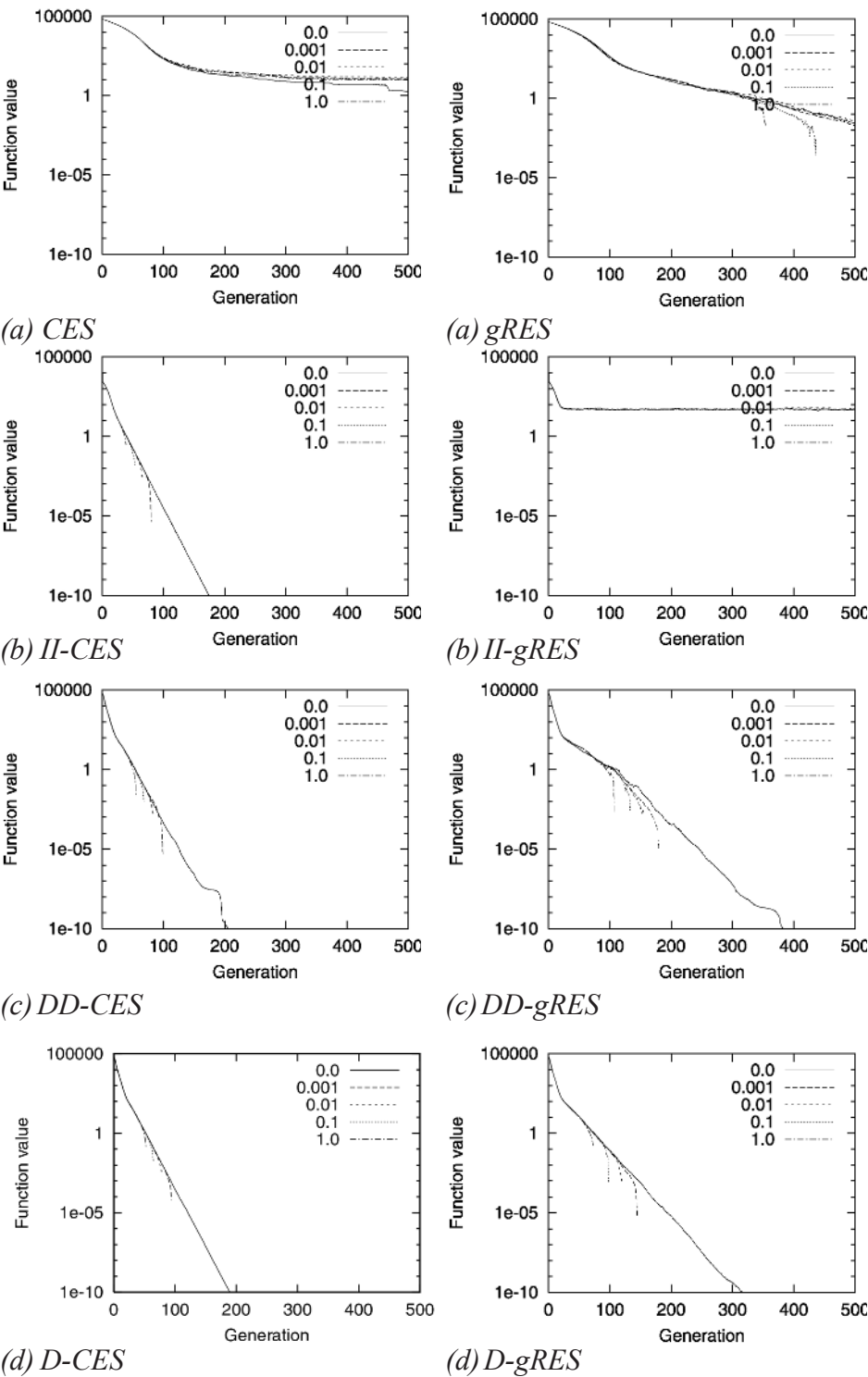


Figure 3 and Figure 4: Averaged best results for $f1$ when the noise level is 0.0, 0.001, 0.01, 0.1 and 1.0

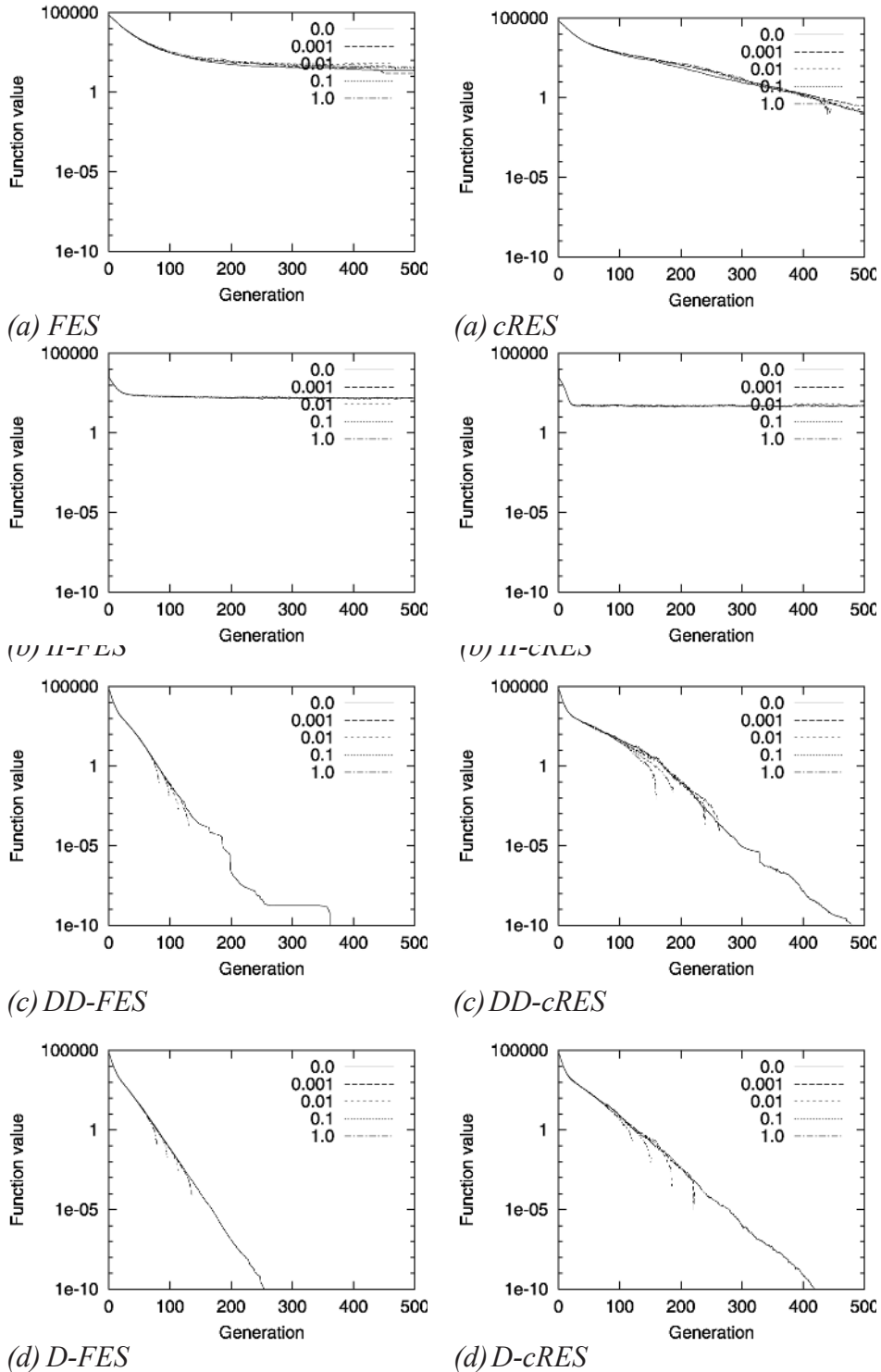
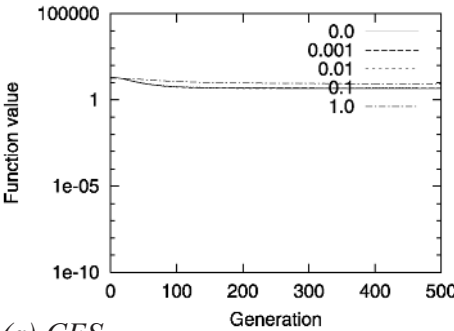
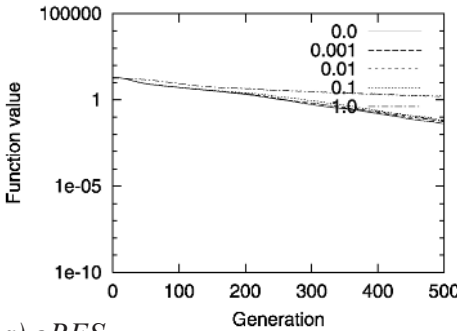


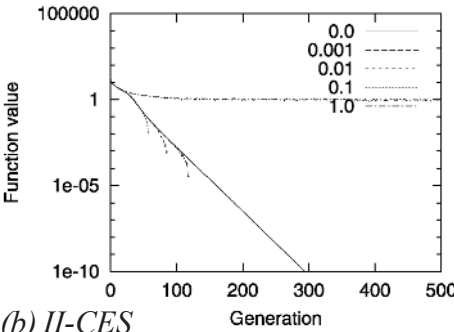
Figure 5 and Figure 6: Averaged best results for f_2 when the noise level is 0.0, 0.001, 0.01, 0.1 and 1.0



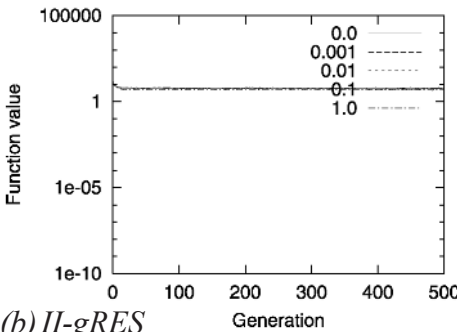
(a) CES



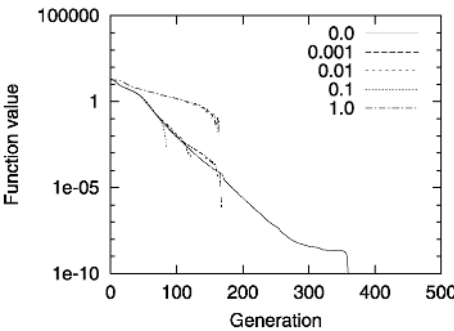
a) gRES



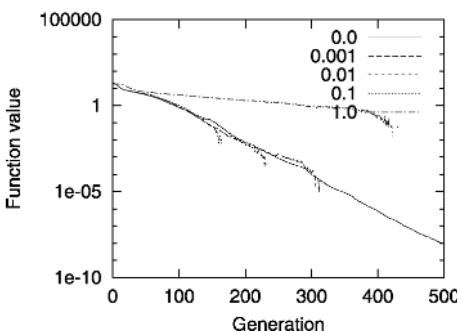
(b) II-CES



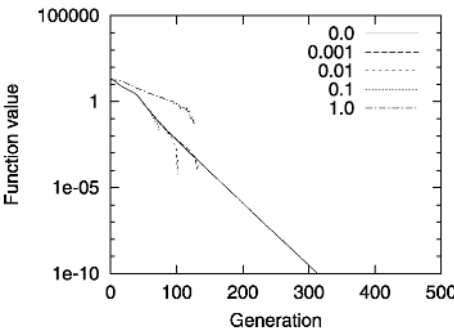
(b) II-gRES



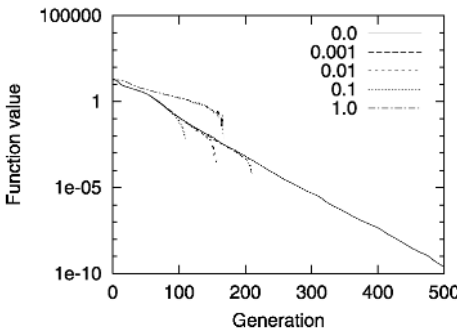
(c) DD-CES



(c) DD-gRES



(d) D-CES



(d) D-gRES

Figure 7 and Figure 8: Averaged best results for f_2 when the noise level is 0.0, 0.001, 0.01, 0.1 and 1.0

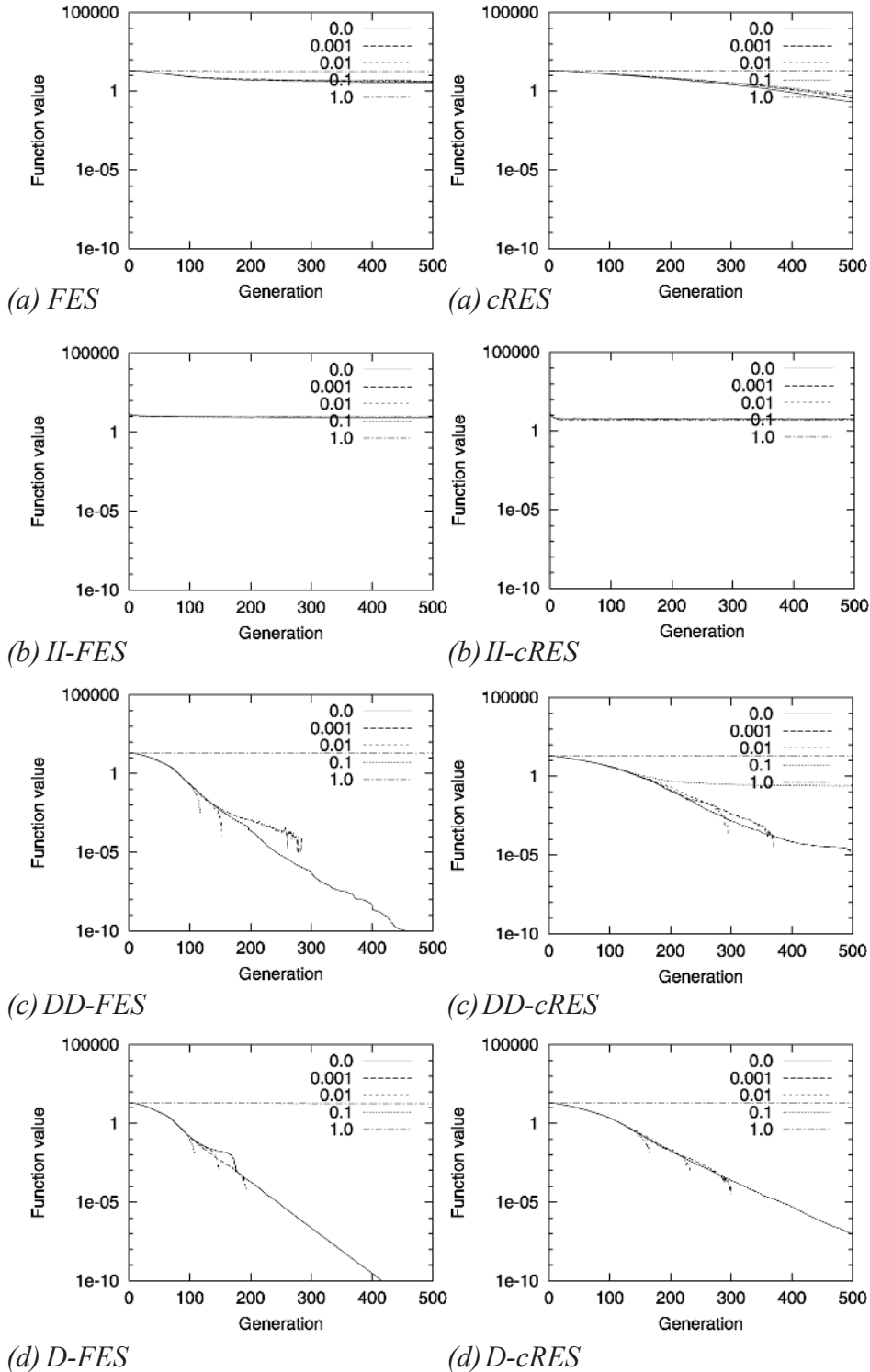


Figure 9 and Figure 10: Averaged best results for f_3 when the noise level is 0.0, 0.001, 0.01, 0.1 and 1.0

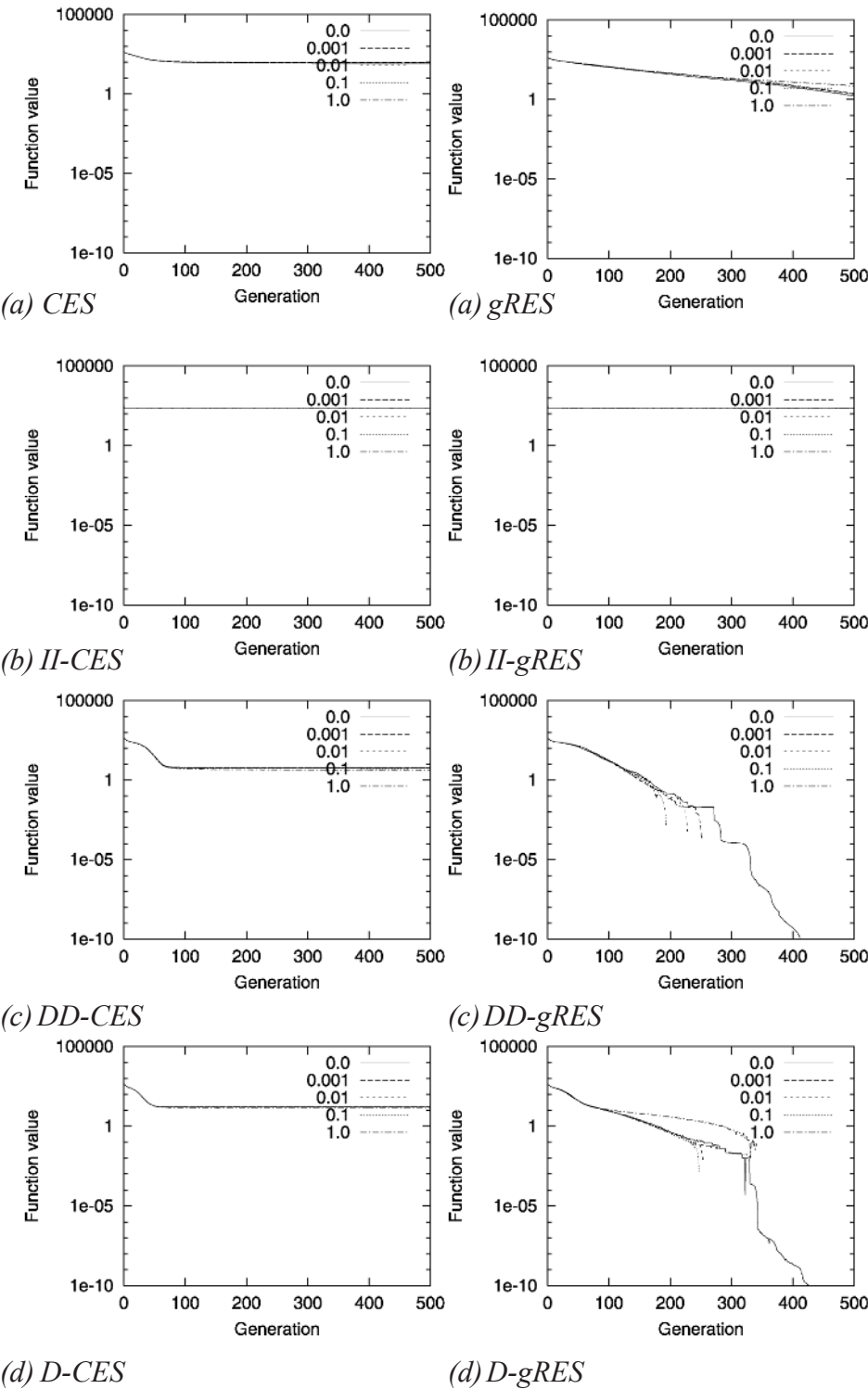
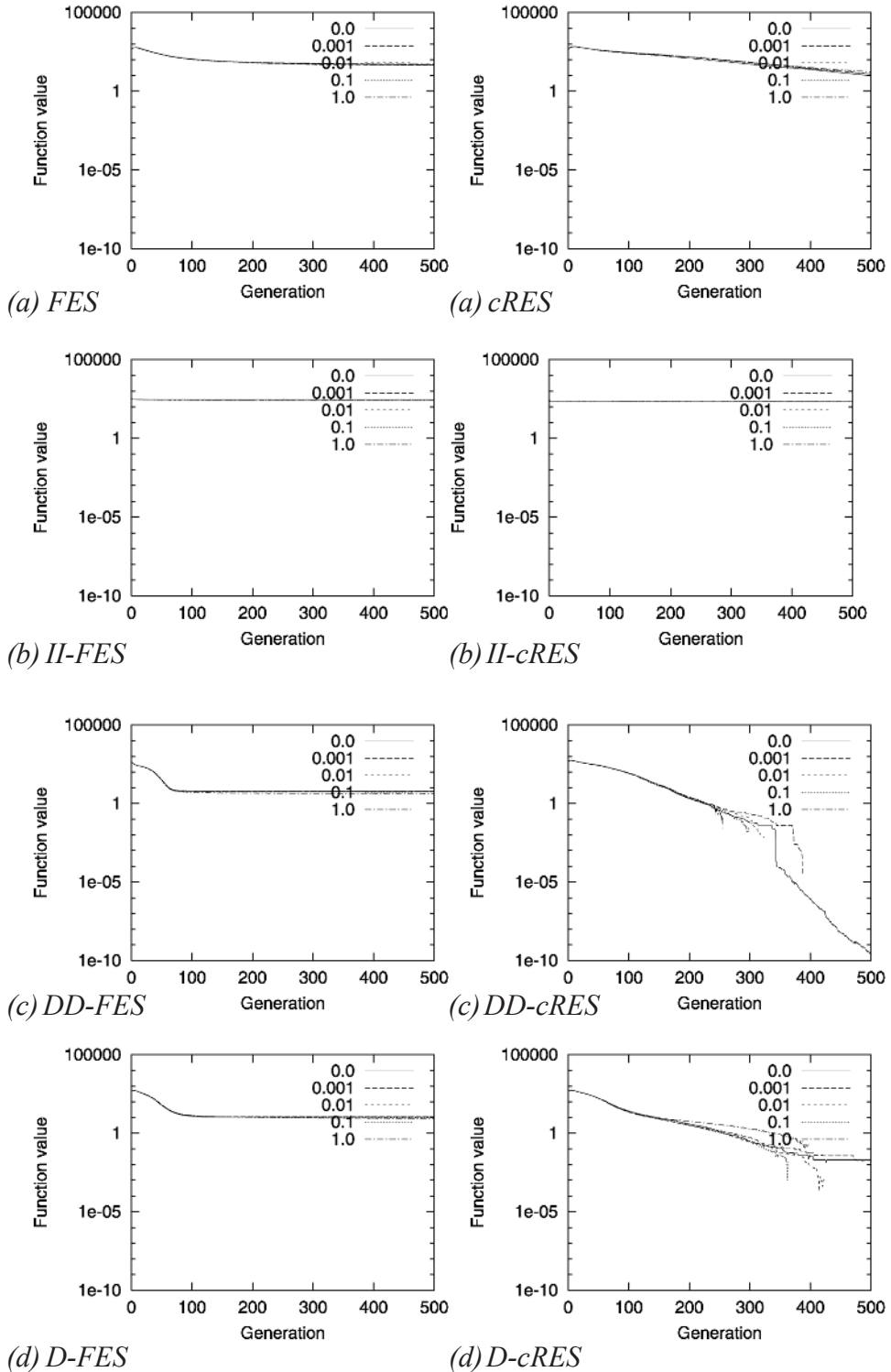


Figure 11 and Figure 12: Averaged best results for f_3 when the noise level is 0.0, 0.001, 0.01, 0.1 and 1.0



than $1e-10$ except when the noise level is either 0.0, 0.1 or 1.0. In Figure 8(d), D-cRES can find function values less than $1e-10$ except when the noise level is either 0.0 or 1.0. These results suggest that cRES on f_2 prefers D recombination to II and DD recombination.

The averaged best function values of CES, gRES, FES and cRES when applied to the generalized Rastrigin's function f_3 are shown in Figure 9 to Figure 12 for the five different noise levels, 0.0, 0.001, 0.01, 0.1 and 1.0. These results are different from f_1 and f_2 due to cases where the ESs are trapped in local optima. In Figure 9(a), CES does not find function values less than 10.0. In Figure 9(b), II-CES does not evolve at all. In Figure 9(c) and (d), DD-CES and D-CES get trapped in local optima. These results suggest that CES on f_3 prefers DD and D recombination to II recombination. In Figure 10, gRES on f_3 shows the same tendencies as gRES on f_1 and f_2 , preferring DD and D recombination to II recombination. In Figure 11, FES shows same tendencies as CES on f_3 , preferring DD and D recombination to II recombination. In Figure 12(a), cRES does not find function values less than 10.0. In Figure 12(b), II-CES does not evolve at all. In Figure 12(c), DD-cRES can find function values less than $1e-9$. In Figure 12(d), D-cRES can find function values less than $1e-10$ except when the noise level is 0.0. These results suggest that cRES on f_3 prefers DD recombination to II and D recombination.

Table 2 summarizes the results for all algorithms over all three noisy test functions f_1, f_2, f_3 . An "x" indicates that the algorithm failed to evolve at all, a triangle indicates that the algorithm found only poor solutions, while a circle indicates that the algorithm found good solutions in reasonable time. As can be seen, the DD and Drecombination shows improvement over the II recombination in almost all cases.

CONCLUSIONS

We have empirically investigated the effect of multi-parent recombination over three versions of the $(\mu / \mu, \lambda)$ -ES algorithm with noisy objective functions. Computer simulations were used to compare the performance of multi-parent versions of intermediate recombination and discrete recombination in CES, FES and RES. We saw that the performance of the $(\mu / \mu, \lambda)$ -ES algorithms depended

Table 2: Summary of experiments

	CES			GRES			FES			cRES		
	II	DD	D	II	DD	D	II	DD	D	II	DD	D
f_1	○	○	○	∈	○	○	∈	○	○	∈	○	○
f_2	○	○	○	∈	○	○	∈	○	○	∈	△	○
f_3	∈	△	△	∈	○	○	∈	△	△	∈	○	○

on the particular objective functions. However, the FES, gRES and cRES algorithms were seen to be improved by multi-parent versions of discrete recombination applied to both object parameters and strategy parameters, i.e., DD recombination and D recombination are better than II recombination.

ACKNOWLEDGMENT

We would like to thank Tom Smith, CCNR, University of Sussex, for his proofreading, and Matt Quinn, CCNR, University of Sussex, for his kindness. The first author acknowledges financial support through JSPS (the Japan Society for the Promotion of Science) Research Fellowship for Young Scientists (200004999).

REFERENCES

- Arnold, D.V. & Beyer, H.G. (2000). Efficiency and mutation strength adaptation of the $(\mu/\mu_I, \lambda)$ -ES in a noisy environment, *Proceedings of the 6th Conference on Parallel Problem Solving from Nature -PPSN VI*, Lecture Notes in Computer Science, vol. 1917, 39-48, Springer.
- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*, Oxford University Press.
- Bäck, T. & Eiben, A.E. (1999). Generalizations of intermediate recombination in evolution strategies. *Proceedings of Congress on Evolutionary Computation (CEC'99)*, 1566-1573, IEEE Press.
- Bäck, T. & Hammel, U. (1994). Evolution strategies applied to perturbed objective functions. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 40-45, IEEE Press.
- Bäck, T. & Schwefel, H.P. (1993). An overview of evolutionary algorithms for parameter optimization, *Evolutionary Computation*, 1(1), 1-23.
- Beyer, H.-G. (1993). Toward a theory of evolution strategies: Some asymptotical results from the $(1, +\lambda)$ -theory, *Evolutionary Computation*, 1(2), 165-188.
- Beyer, H.-G. (1995). Toward a theory of evolution strategies: On the benefits of $(\mu/\mu, \lambda)$ -theory, *Evolutionary Computation*, 3(1), 81-111.
- Beyer, H.-G. (1998). Mutate large, but inherit small! On the analysis of rescaled mutations in $(1, \lambda)$ -ES with noisy fitness data, *Proceedings of 5th Conference on Parallel Problem Solving from Nature -PPSN V*, 109-118, Springer.
- Beyer, H.-G. (2001). *The Theory of Evolution Strategies*, Springer.
- Chang, M., Ohkura, K. & Ueda, K. (2001). Some experimental observation of $(\mu$

- $/\mu, \lambda$)-evolution strategies. *Proceedings of the Congress on Evolutionary Computation (CEC'01)*, 663-670, IEEE Press.
- Eiben, A.E. & Bäck, T. (1998), Empirical investigation of multi-parent recombination operators in evolution strategies. *Evolutionary Computation*, 5(3), 347-365.
- Gruenz, L. & Beyer, H.G. (1999). Some observations on the interaction of recombination and self-adaptation in evolution strategies, *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, 639-645, IEEE Press.
- Hammel, U. & Bäck, T. (1994). Evolution strategies on noisy functions: How to improve convergence properties, *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature—PPSN III*, Lecture Notes in Computer Science, vol. 866, 418-427, Springer.
- Harvey, I., Husbands, P., Cliff, D., Thompson, A. & Jakobi, N. (1997). Evolutionary robotics: The Sussex approach, *Robotics and Autonomous Systems*, vol. 20, 205-224.
- Jacobi, N., Husbands, P. & Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics, *Proceedings of the Third European Conference on Artificial Life (ECAL95)*, 704-720, Springer.
- Kimura, M. (1983), *The Neutral Theory of Molecular Evolution*, Cambridge University Press.
- Nissen, V. & Propach, J. (1998). Optimization with noisy function, *Proceedings of the 5th Conference on Parallel Problem Solving from Nature—PPSN V*, Lecture Notes in Computer Science, vol. 1498, 159-168, Springer.
- Ohkura, K., Matsumura, Y. & Ueda, K. (2001), Robust evolution strategies, *Applied Intelligence*, 15(3), 153-169, Kluwer Academic Publishers.
- Schwefel, H.-P. (1995), *Evolution and Optimum Seeking*, John Wiley & Sons.
- Yao, X. & Liu, Y. (1997), Fast evolution strategies, *Control and Cybernetics*, 26(3), 467-496.

Chapter XVI

On Measuring the Attributes of Evolutionary Algorithms: A Comparison of Algorithms Used for Information Retrieval

J. L. Fernández-Villacañas Martín
Universidad Carlos III, Spain

P. Marrow and M. Shackleton
Intelligent Systems Laboratory, BTextextract Technologies, UK

ABSTRACT

In this chapter we compare the performance of two contrasting evolutionary algorithms addressing a similar problem, of information retrieval. The first, BTGP, is based upon genetic programming, while the second, MGA, is a genetic algorithm. We analyze the performance of these evolutionary algorithms through aspects of the evolutionary process they undergo while filtering information. We measure aspects of the variation existing in the population undergoing evolution, as well as properties of the selection process. We also measure properties of the adaptive landscape in each algorithm, and quantify the importance of neutral evolution for each algorithm. We choose measures of these properties because they appear generally important in evolution. Our

results indicate why each algorithm is effective at information retrieval, however they do not provide a means of quantifying the relative effectiveness of each algorithm. We attribute this difficulty to the lack of appropriate measures available to measure properties of evolutionary algorithms, and suggest some criteria for useful evolutionary measures to be developed in the future.

INTRODUCTION

Evolutionary methods have been the focus of much attention in computer science, principally because of their potential for performing a partially directed search in very large combinatorial spaces (Sloman, 1998). Evolutionary algorithms (EAs) have the potential to balance exploration of the search space with exploitation of useful features of that search space. However the correct balance is difficult to achieve and places limits on what can be predicted about the algorithm's behaviour. In addition, EAs are often implemented in system-specific ways, making it very difficult to predict and evaluate performance on different implementations. This makes the need for measures to evaluate and compare different algorithms all the more urgent.

In this chapter we focus upon the comparison between algorithms for information retrieval. This is one of the tasks at which evolutionary algorithms have been found particularly effective. Such algorithms deal with the situation where a relevant sub-set of documents or records must be isolated from a larger pool. This chapter considers two such algorithms which were developed for the task of information filtering in a telecommunications context. The BTGP is a genetic programming system where the programs produced execute Boolean searches through keywords (Fernández-Villacañas & Exell, 1996). The MGA is a genetic algorithm which also uses a Boolean tree representation, through a relatively complicated mapping between genotype and phenotype.

We compare the performance of these algorithms using a collection of measures chosen from consideration of evolutionary processes. Such measures have been developed within an evolutionary computation context and also within evolutionary biology. To understand why such measures might be useful, we first consider the evolutionary process itself.

Evolution can be described as “...any net directional change or any cumulative change in the characteristics of organisms or populations over many generations ...” (Endler, 1986).

But this evolutionary change may occur as the consequence of a number of different processes, acting to differing extent. Comparison of biological and

computational evolution shows the importance of three classes of phenomena in making natural and artificial evolutionary systems evolvable. These are variation, selection and adaptive landscape structure.

The existence of variation is crucially important for evolutionary processes because there would otherwise be no possibility for the selection scheme to exploit the search space. Measuring the amount of variation gives an indication of the potential of the population to be selected, although it does not of course tell about the potential of the population to vary in the future. Ideally we need to know about the propensity of the population to vary in the future in order to get a full picture of the evolvability of the system. This distinction between the amount of variation and the variability of a population has been emphasized, in the context of evolvability by Wagner and Altenberg (1996).

Variation may be measured through genetic variance, which can be calculated provided it is possible to set values on the different genetic variants present (Falconer, 1989; Lynch & Walsh, 1998). Depending on the evolutionary algorithm under consideration, it may be more appropriate to take a phenotypic variance measure, as the representation of the genotype in the phenotype may crucially affect the way in which available variation influences the selective process. The method of measurement of phenotypic variation will depend upon the representation used.

Mutation is an important means of generating further variation, and acts in part to counteract the loss of variation through selection. It must therefore be important for evolvability. It is with this in mind that Wagner and Altenberg (1996) have proposed mutational variance, the variation in effect of possible mutants that can arise in a population, as a measure of the evolvability or evolutionary performance of a system. While mutation variance may be very difficult to calculate in natural populations, it is at least in principle derivable for a given evolutionary algorithm.

While variation may be essential for evolutionary change, it is also the case that some means of searching through the variation is necessary for the evolutionary algorithm to have some practical application. Most frequently this means some sort of selection or evaluation function applied to the population of each generation, which allows only a subset to reproduce. Fortunately there is already a large literature dealing with the properties and performance of selection functions (Altenberg, 1994, 1995; Blickle & Thiele, 1997; Manderick et al., 1991; Mühlenbein & Schlierkamp-Voosen, 1993; Mühlenbein, 1998). We can also draw upon the theoretical tools for the analysis of natural and artificial selection which have been developed by quantitative geneticists and animal breeders (Falconer, 1989; Lynch & Walsh, 1998; Roff, 1998).

There are a wide range of measures which have been used to characterize selection (Brodie et al., 1995; Blickle & Thiele, 1997). In this study we focus on two measures which indicate important features of selection; it will be possible to

extend this analysis to other measures in the future. The first, opportunity for selection (derived from Crow, 1958; Arnold & Wade, 1984), measures the potential of a population to respond to selection. The second, called here intensity of selection (although this term has been applied to other measures, e.g., Brodie et al., 1995; Endler, 1986) measures the strength of selection on characters under selection. These measures allow us to build up a comparative picture of the nature of selection in different systems.

Variation and selection do not provide a complete picture of evolutionary progress; we also need to know something about the process of adaptation. The metaphor of the adaptive landscape has provided a useful means of studying the process of adaptation and has led to a large range of measures of evolutionary processes (Gavrilets, 1997; Hordijk, 1992; Kauffman, 1993). A central problem with the view of evolution taking place on an adaptive landscape is that selection is envisaged as driving populations up gradients of increasing fitness. Any reasonably complex adaptive landscape, with multiple fitness peaks, will result in populations reaching local optima below the global optimum, from which they cannot escape. It is therefore useful to obtain measures of the likelihood of transitions on the adaptive landscape leading to fitness increase, as these may give an indication of the likelihood of adaptation proceeding without the population being stuck in local maxima. In this chapter we implement a variant of epistasis variance (after Davidor, 1991), which may give such an indication. We also measure the proportion of mutants fitter than the current variant.

Escape from local maxima may also be possible through the intermediary of adaptively neutral mutations, which may change the genome without changing the individual's fitness, and thus create the circumstances in which further advantageous mutants can invade. There has been much interest in neutral evolution as a facilitator of adaptive change in recent evolutionary computation research (Schuster, 1996). Our aim here is not to focus specifically on this area of study, but to quantify in a simple way the potential for neutral change. We do this by calculating the proportion of neutral mutants that are possible; this allows us to establish further the properties of the adaptive landscape.

We are thus able to apply measures of variation, of selection and of properties of adaptive landscapes to two different evolutionary algorithms developed to solve equivalent problems. This allows us to gain insights into the evolutionary process as each algorithm converges to a solution, and to identify the most effective features of each algorithm.

There are many more measures that could be applied to the evolutionary algorithms BTGP and MGA: our choice of measures here was dictated by relevance to some of the most important components of the evolutionary process, and feasibility of implementation in the given algorithms. However the need to

understand the evolutionary process in EAs is sufficiently urgent to suggest that research involving measurement of evolutionary phenomena is essential in the development of EAs for real-world applications. As Mitchell and Forrest (1995), discussing the relation of genetic algorithms to artificial life, write: “...*the formulation of macroscopic measures of evolution and adaptation, as well as descriptions of the microscopic mechanisms by which the macroscopic quantities emerge, is essential if artificial life is to be made into an explanatory science ...*” and “...*we consider it an open problem to develop adequate criteria and methods for evaluating artificial life systems.*” Their comments still apply strongly to evolutionary computation and artificial life.

The outline of the chapter is as follows. After this introduction, we introduce the evolutionary measures that we will be using, and explain their derivation. Later we give the details of the two algorithms we consider, MGA and BTGP. Another sections follows with the definition of the problem which the algorithms are used to solve, information retrieval. Furthermore, the implementation-specific features of the measures we have used are presented. We also present the results of applying the above measures, and interpret these results in the context of each algorithm. Finally, we draw conclusions regarding the performance of each algorithm relative to the other, and suggest some guidelines regarding the definition of measures for more general evolutionary algorithms.

EVOLUTIONARY MEASURES

The performance of an EA is a balance between the exploration of its search space and the exploitation of features of the same space (Holland, 1992), such as local minima. We have identified a number of parameters that are representative of features of the evolutionary process, in particular features that affect, or are consequences of the balance between exploration and exploitation. These features are:

MEASURES OF VARIATION

Genetic Diversity

This measures the amount of variation among the genotypes present in a population. Some variation is essential for an evolutionary algorithm, for without it selection cannot take place. This measure of diversity depends upon the search space being represented in a gene string or equivalent. Since both BTGP and MGA depend upon such a representation, this measure is appropriate. We use a measure of genetic diversity, rather than the more usual genetic variance, because of the

greater difficulty involved in deriving genetic variance in this computational context. Measurement of genetic variance in a biological population is discussed by Falconer (1989); calculation of genetic variance in EAs will depend upon the representation.

Mutation Variance

This measures the potential degree of variation in magnitude of mutation effect. Such a measure assumes that mutation effect can be quantified on an ordinal scale. Mutational variance gives an indication of the extent to which mutation can explore the search space. It has been proposed as a measure of the performance of evolutionary systems by Wagner and Altenberg (1996). The actual, as opposed to potential, variation in effect among mutants will depend upon the population size, the time period under consideration and the mutation rate, and is not considered here. Mutation variance is typically difficult to calculate in biological systems (Warner & Altenberg, 1996), but should be easy to identify in EAs through definition in the algorithm.

Phenotypic Diversity

This measures the degree of variation in fitness between pairs of randomly generated individuals across the fitness landscape. Phenotypic diversity is a quantity difficult to relate to more biologically conventional measures of diversity, such as variance. It may measure some of the potential for evolution, given that genetic variation is required and the one-to-one mapping between genotype and phenotype. Its implementation will be explained later on.

MEASURES OF SELECTION

Opportunity for Selection

This gives an estimate of the upper limit of the strength of selection in a system (Brodie *et al.*, 1995). It is measured by the variance in relative fitness in the population (Arnold & Wade, 1984; Brodie *et al.*, 1995; Crow, 1958); relative fitness being fitness scaled by mean fitness. Relative fitness is used in order to be able to compare measures in different populations on an equivalent scale. Variance in fitness gives a measure of the maximum strength of selection in a system since fitness differences are required for selection to occur (Endler, 1986) and the size of these differences determines the consequences that selection can have on the system. It can be shown that variance in relative fitness bounds the effect of selection (e.g., Arnold & Wade, 1984). The magnitude of the opportunity for selection will depend crucially on the way in which fitness is measured. There are of course many different ways of doing this in EAs as in biological populations, and the usefulness

of opportunity for selection as a measure will relate to the appropriateness of the fitness measure used.

Intensity of Selection

This measures the overall strength of selection in a system (Brodie et al., 1995). It is calculated (following Arnold & Wade, 1984, who proposed this method of measurement in a biological context) by taking the slope of the linear regression of fitness on a phenotypic trait. This calculation differs from the more usual calculation of the effect of selection via the Breeder's equation (Mühlenbein, 1998). The measurement used here was derived for use upon natural populations, where it is more flexible, allowing for arbitrary genetic correlations between characters. Consequently, we believe that it will also be useful for evolutionary algorithms, away from the specific circumstances in which methods based upon the Breeder's equation are most appropriate (Mühlenbein & Schlierkamp-Voosen, 1993; Mühlenbein, 1998). Our calculation relies on the observation that if selection were more intense, we would expect a closer association between fitness and the value of a phenotypic trait under selection. This measure requires that fitness be measured as relative fitness, in order to provide comparable measurements, and that phenotypic values are standardized by subtracting their mean and dividing by their standard deviation (Arnold & Wade, 1984). The regression coefficient measures the total selection differential, an estimate of the intensity of selection on that particular trait which combines both direct selection on that trait and indirect selection on other traits which affect that trait. Other methods which can be used to calculate direct selection (Arnold & Wade, 1984) are not considered here. In an EA, there may not be much distinction between total and direct selection, depending upon the representation, but this is likely to be very different in biological scenarios. The implementation of this measure depends upon fitness being measurable, as well as some phenotypic character. These conditions should be satisfied in many EAs, although the details of the measurement process will usually be implementation-dependent.

Effect of Genetic Drift

Not directly related to selection but otherwise important as a means of cancelling variation in the population is the effect of genetic drift. This process occurs because the gene pool of the offspring in a finite population is unlikely to be the same as that of their parents. It is particularly important in small populations where sampling acts on all variants of the population, under selection or not, reduce the amount of variation. Its effect will be obscured by selection in most EAs. A practical implementation of the effect caused by this process will be discussed later.

MEASURES OF ADAPTIVE LANDSCAPES

Epistasis Variance

This measures the degree of variation in epistasis in an evolutionary system (Davidor, 1991). Epistasis is interaction between genes; its consequence is that measurement of the fitness resulting from individual genes will not accurately predict the overall fitness resulting from the complete genotype. Thus epistasis variance is a measure of the predictability of the fitness landscape derived from the fitness of genes which make it up. As such, it is a useful measure of the performance of EAs because it quantifies how easy it will be to explore genotype space. Epistasis variance depends upon what definition is given to a gene, a difficult enough problem in biological systems, likely to be implementation-specific in EAs. Consequently its derivation is described in more detail later.

Proportion of Fitter Mutants

This measures the proportion of mutants which are fitter than the current fittest individual, averaged over the fitness landscape. Although fitness increase is not universal under selection either in EAs or biological systems, some fitness increase is likely in the search for a solution in an EA, and measures of the ability to produce fitter mutants have been proposed as an indication of the evolvability of an evolutionary system (Altenberg, 1994).

MEASURES OF NEUTRAL EVOLUTION

Proportion of Neutral Mutants

This measures the proportion of feasible mutants which do not change individual fitness away from its current value, and thus are adaptively neutral. This measure gives a preliminary indication of the possibility of neutral transitions within genotype space, and thus may give an indication of the potential for populations to escape from local maxima which are not global maxima. The proportion of neutral mutants does not directly tell us about neutral networks that may percolate through genotype space; this would require substantially more analysis beyond the scope of this chapter (Huynen, 1995; Nimwegen & Crutchfield, 1998).

This is by no means a complete list of possible measures of the performance of EAs. However, these measures do encompass a range of the attributes of evolutionary systems, and are general enough to apply to different EAs. Furthermore, they have specific links to equivalent measures in biological systems. Implementation-specific details are given later on.

THE ALGORITHMS

BTGP

The genetic programming system (BTGP) maintains a population of phenotypes (decision trees) which it operates on directly. In this sense the genotype and phenotype can be considered to be one and the same.

After generating the initial population, the BTGP performs the genetic programming cycle of fitness evaluation, selection of parents and reproduction with application of the genetic operators to produce the children of the next generation. The BTGP has many configuration options (see Fernández-Villacañas & Exell, 1996), but for the experiments described in this chapter the following options were used:

- “Ramped growth” of the initial population’s trees
- Fitness proportionate (roulette wheel) selection
- Genetic operators: copy, crossover, mutation

In addition to the above settings, the following parameters can be experimentally varied:

- Rates at which each genetic operator is applied
- Maximum tree depth
- Node branching factor

Ramped growth means that the generated trees are uniformly distributed in depth up to the maximum tree depth. Crossover is performed by randomly choosing nodes from each parent and exchanging them, but avoiding exchanges which would exceed the maximum tree depth. Mutation consists of replacing a node with a randomly grown sub-tree up to the maximum depth. Further details regarding the BTGP and the information retrieval task are given elsewhere (Fernández-Villacañas & Exell, 1996).

MGA

The MGA is based on the Simple Genetic Algorithm described by Goldberg (1989). The SGA together with a relatively complex genotype-phenotype mapping comprise the Mapping Genetic Algorithm (MGA). The mapping takes an unrestricted bit-string genome of fixed length from the genetic algorithm and parses it sequentially to create a list of node descriptions which are then assembled to form a tree.

Each node is described by a fixed number of bits (a gene, typically 45 bits) of the genome. The fields encoded are: root and child labels; function type, negation

flag and used flag; reference type; reference value. The meaning of the fields is as follows: the root label provides a bit pattern against which child labels are matched in order to assemble a list of nodes into a tree. The node's function is specified by the function type, negation and used flags providing for the functions AND, OR, NAND, NOR and NIL. The latter function allows sub-trees to be switched on or off via mutation. Reference values are interpreted either as child labels or keywords depending on the reference type. The reference type field uses 3 bits, building extra redundancy into the genetic code, potentially requiring more than a single bit mutation to change the reference type. The number of references encoded per node defines the maximum tree branching factor (typically 4). Once the individual genes have been decoded to generate a list of their corresponding node descriptions, these nodes are assembled to form a decision tree comprising the phenotype by matching child labels to root labels of other nodes. Where a match is found, a sub-tree is formed. Every node which remains ultimately un-referenced by a parent node forms the root node of a tree, the largest of which is taken as the phenotypic tree.

The genetic algorithm uses fitness proportionate (roulette wheel) selection. The mutation operator used is a simple bit flip mutation. Single point crossover was available and was extended to respect gene boundaries, but was rarely used as it was found to be too disruptive in general. Further investigations of the role of crossover within the MGA are left to future research.

Possible effects on the phenotype of a point mutation of the genotype include: addition, deletion or change of sub-tree; switching off/on a sub-tree; change of function; replacement of a sub-tree by a keyword reference (leaf node) or vice versa; creation or change of a keyword reference. Mutations which change labels can effect quite large changes similar to those produced by the BTGP crossover and mutation operators. There are several sources of redundancy in the genotype-phenotype mapping; many different node list arrangements could code for the same phenotypic tree and many apparently different decision trees may be logically equivalent when evaluated.

INFORMATION RETRIEVAL

The task on which BTGP and MGA have been applied is to evolve a Boolean decision tree capable of discriminating between two document classes, those sought in a retrieval task and those which are of no interest. The data used is generated in a pre-processing step from Internet documents which have been labeled by a user as either of interest (positive) or of no interest (negative). Pre-processing consists of extraction of a set of keywords across all the documents, and then recording for each document whether it is a positive or negative example, and

whether each keyword is present or absent. The resulting data records, one per document, are then separated into training and test sets.

THE PHENOTYPE AND FITNESS FUNCTION

The phenotypic representation is a Boolean decision tree. Each node of this tree is either a function node taking one of the values AND, OR, NOR, NAND or a leaf node variable which references a particular keyword. For a given training or test case, each keyword variable will be instantiated to the value 1 or 0 denoting the presence or absence (respectively) of the corresponding keyword for that case. A tree which evaluates TRUE for a positive case or FALSE for a negative case has thus correctly classified that case.

The fitness function is evaluated over a set of training or test cases. It is parameterized by the following values: the number of correctly identified positives n_{pos} , the number of negatives falsely identified as positive n_{neg} , the total number of positives N_{pos} and the total number of negatives N_{neg} . The fitness function is designed to minimize both the number of missed positives and the number of false positives:

$$f = \alpha \frac{(N_{pos} - n_{pos})}{N_{pos}} + \beta \frac{n_{neg}}{N_{neg}}$$

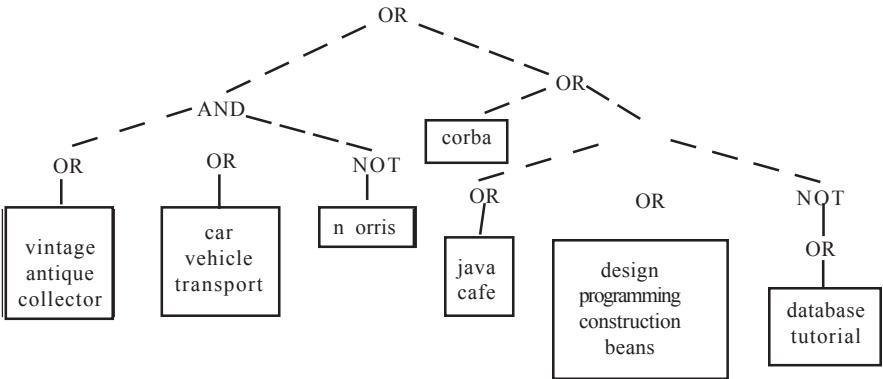
Note that α and β and the function lie in the range $[0, 1]$ with 0 being the best possible fitness, 1 the worst. The aim is therefore to minimize its value.

The data set was generated from a known decision tree illustrated in Figure 1. It has 16 keywords, a training set of 200 cases and a test set of 50 cases. The training and test cases were chosen randomly from the 2¹⁶ possible keyword configurations such that each set contained an equal number of positive and negative cases.

IMPLEMENTATION OF MEASURES

In this section we will specify how the different evolutionary measures and processes previously identified are applied to our algorithms (BTGP and MGA) and task (information retrieval). On one hand, some of the measures' specifications will depend on the nature of the algorithm itself and its representation of solutions (Boolean trees) where phenotype and genotype are the same, while, on the other hand, some will be influenced by the definition of fitness and sampling of our fitness landscape derived from the task. Let's analyze them in turn.

Figure 1: Decision tree corresponding to data set



MEASURES OF VARIATION

Genetic Diversity

In BTGP each active genomic unit that contributes to the fitness of the individual is a function node; that is, our genes are AND, OR, NAND, NOR. These ‘genes’ can only have two allele values, either 0 or 1 after evaluation of their leave nodes.

Genetic diversity, σ_g^2 is measured as the sum of the differences between the average fitness for a particular combination of alleles in the population and the mean fitness value for all combinations of alleles. This quantity is also normalized by the number of possible combinations of alleles, N , as,

$$\sigma_g^2 = \frac{1}{N} \sum_{i=1}^N (f_i - \bar{f})$$

where, for instance, $N=16$ for BTGP with four genes.

The above measure of genetic diversity would not be computationally feasible to calculate over the 900 bits of the MGA genome. Consequently we have developed a second measure of genetic diversity based on Hamming distance between genomes. The genetic diversity is the sum of Hamming distances between the genomes of a sample of randomly chosen pairs of individuals, normalized by the number of samples and the genome length.

Mutation Variance

This measure is implemented by first calculating the average of the differences in fitness that result from having one gene switched on or off, Δf_i (alleles values of

1 or 0,) in the population. These are later subtracted from the mean difference, Δf for each combination. For instance, in BTGP, the number of combinations for genes on or off is $N=4$. The expression for mutation variance is:

$$\sigma_m^2 = \frac{1}{N} \sum_{i=1}^N (\Delta f_i - \overline{\Delta f})$$

Phenotypic Diversity

Ph_{div} is the result of randomly generating pairs of trees and measuring the Hamming distance over a number of randomly generated environment loci (typically 1,000 pairs over 200 cases). One locus corresponds to a combination of the 16 different keywords in the data set. Each tree returns either 1 or 0 for each locus; different returns are counted for each pair of trees and this result is normalized in the scale $[0,1]$.

MEASURES OF SELECTION

Opportunity for Selection

As described earlier on, Op_{sel} measures the variance in the relative fitness of the whole population: fitness is measured for each individual in the population for a training set (typically 200 cases) and then is normalized by the mean fitness of the whole population. Variance of this relative fitness is then calculated across the whole population.

High values of Op_{sel} show potential for evolution through natural selection, hence evolvability. In the presence of strong selection, Op_{sel} should decline over time.

Intensity of Selection

The phenotypic traits available for choice are the number of nodes, N_{nod} , and levels, N_{lev} , in the BTGP trees. At each generation, fitness is evaluated on the training set and a linear correlation coefficient, r^2 , and the slope, b (regression coefficient), from the fit $y=a+bx$, are calculated for the fitness to each phenotypic trait diagram. For a description of these coefficients, see Press et al. (1994). In order to compare between different values, of phenotypic traits with their also different fitness values we have scaled the x-data to the deviation from the mean phenotypic trait, N_{nod} , and divided by the trait's standard deviation as:

$$\frac{N_{nod} - \overline{N_{nod}}}{\sigma_{N_{nod}}}$$

and the fitness (y-data) as the relative fitness, f/\overline{f} .

If N_{nod} and N_{lev} were very good phenotypic traits under selection, we would expect a correlation coefficient, $r^2 \approx 1$ and strong values for b . The real situation is far from that; N_{nod} and N_{lev} are not good phenotypic traits as, in the majority of runs of our algorithm, the correlation is very poor (resulting from sparse clouds of points in the diagrams), and only in some cases we can establish some conclusions as to the sign of the correlation coefficient, indicating predilection for bigger or deeper trees ($r > 0$) and smaller or shorter trees ($r < 0$).

The algorithms used to calculate these coefficients are derived from Press *et al.* (1994). In calculating intensity of selection in this way over successive generations, we depart from the usual use of this measure, which is usually calculated once for each population (Arnold & Wade, 1984). We do this in order to investigate whether the intensity of selection varies during the run of the EA.

Effect of Genetic Drift

Genetic drift is a process, not a measure; its effect on the total genetic variation may be obscured by selection in most EAs. We have decided to get a phenotypic diversity measure in the absence of mutation and selection to get an insight on the effect of sampling in the population from generation to generation. As previously discussed this effect is on fitness, not on genic content, so we should refer to it as fitness sampling drift.

MEASURES ON ADAPTIVE LANDSCAPES

Epistasis Variance

Following Davidor's (1991) approach, epistasis variance is defined as:

$$\sigma_{\varepsilon}^2 = \frac{1}{N_{\Gamma}} \sum_{S \in \Gamma} (f(S) - A(S))^2$$

where Γ is the grand population of all possible strings $\{0,1\}^4$, $f(S)$ is the fitness of string S and $A(S)$ is the predicted string value from the alleles separate contribution.

In turn, if we define the excess fitness of a string S as, $E(S) = f(S) - \bar{f}$, where \bar{f} is the average fitness and:

$$E(A) = \sum_{i=1}^4 \left(\frac{1}{N_i(a)} \sum_{S \in Pop_{s_i=a}} (f(S)) - \bar{f} \right)$$

as the excess genic value for each allele, $N_i(a)$ being the number of strings that match allele a at position i ; then we can define the predicted string value, $A(S) = E(A) + \bar{f}$ that we need in the epistasis variance definition.

Other variances that will be used are the fitness variance:

$$\sigma_f^2 = \frac{1}{N} \sum_{S \in Pop} (E(S))^2$$

and genic variance:

$$\sigma_A^2 = \frac{1}{N} \sum_{S \in Pop} (E(A))^2$$

Normally, the grand population and our sample population, Pop , are not equal; this results in different values for epistasis variance when we sum Γ or Pop . When we are missing some of the combinations of alleles from the grand population, the statistic is subject to sampling error; the distinction between base epistasis and sample population epistasis variance is thus very important as the latter can sometimes be equal or bigger than the former. When there are no sampling errors, $\sigma_{\epsilon}^2 = \sigma_f^2 - \sigma_A^2$.

Each generation epistasis variance is calculated; first the fitness of the individuals are computed together with the average population fitness; later the population is sampled for having an instance of allele i active and counted, their excess value calculated and all these added up to form the excess genic value $E(A)$; the predicted string value from the individual contributions from the alleles, $A(S)$, is then used in conjunction with the real fitness of that string, $f(S)$, to calculate the epistasis of each string. Finally these values are squared and averaged for the whole population.

The measure of bit-wise epistasis we use in MGA is that suggested by Fonlupt et al. (1998). Bit-wise epistasis was measured in MGA for all 45 bits of a single “gene” (corresponding to one potential tree node) at each generation of each run. The values were then averaged across all runs and generations for each bit to obtain an overall measure of epistasis per bit.

Proportion of Fitter Mutants

At each generation each individual in the population is mutated and the mutant and original tested over the training data set. We count a mutant as fitter when the mutant is fitter than the fittest tree in the previous generation. We repeat this process for all training cases and each individual. Finally, the total count is normalised. As previously discussed the ability to produce fitter mutants at each generation does not relate directly to evolvability but it is a requirement for EAs to solve problems.

MEASURES OF NEUTRAL EVOLUTION

Proportion of Neutral and Non-Neutral Mutants

A number of randomly generated trees are generated and each tree is mutated and evaluated for a number of random environment loci. Equivalent phenotypic responses are counted and normalized. The resulting measure, Pr_{nm} , gives us the proportion of mutation events which have identical phenotypic effect, and is thus a measure of neutrality with respect to phenotypic behaviour. We also measure the proportion of mutants which are neutral with respect to fitness.

The proportion of non-neutral mutants, Pr_{nnm} , is therefore, $Pr_{nnm} = 1 - Pr_{nm}$.

RESULTS AND DISCUSSION

Results were obtained for both BTGP and MGA, subject to the differences in implementation described above. In particular, the interpretation of a gene in the context of each algorithm differs, with the BTGP having 4 ‘genes’ and the MGA having 20 genes, each comprising 45 bits (a total of 900 bits). For instance, calculation of epistasis variance is not computationally feasible over 900 bits, so we instead calculated bit-wise epistasis for the MGA, sampled over the population.

Results were generated over 10 runs of each algorithm, for 1000 generations, with a population size of 1,000 individuals. The settings we used for BTGP were: mutation probability of 0.05 per tree; maximum tree depth of 4 levels; roulette wheel selection; branching factor between 2 and 4. The MGA settings were: probability of single bit mutation per genome of 0.9; roulette wheel selection. Neither algorithm used elitism.

Each algorithm succeeded in reducing the mean fitness over the run. The BTGP typically reached a mean fitness of 0.25 (best run: 0.2) at the end of 1,000 generations, while the MGA achieved better fitness, typically 0.09 (best run: 0.05). The MGA mean fitness changed in a punctuated fashion with periods of rapid improvement interspersed with relatively slow change, whilst the BTGP mean fitness improved more evenly. Both algorithms maintained relatively constant levels of fitness variance, with the MGA having the highest level (0.09) and BTGP a lower level (0.005). This suggests that the MGA offers more scope for exploration.

The genetic diversity in BTGP showed a slow increase on average through the runs. In MGA the equivalent measure increased steadily, interspersed with rapid temporary reductions in diversity. This reflects high fitness neutrality in the genetic representation allowing increases in diversity in periods when selection is not operating strongly. The corresponding phenotypic diversity measure was recorded for both algorithms and found to be very similar in both cases, at a level of approximately 0.1, constant through each run. The difference between phenotypic and genetic diversity measures was a result of the complex mapping between genotype and phenotype encoded in each algorithm.

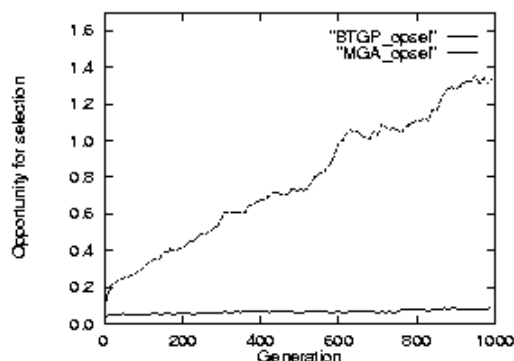
The effect of genetic drift was also studied in different runs with different population sizes; the results indicate that the smaller the population size, the faster is the reduction of the variation and, even for some small population sizes, variation is cancelled completely.

Mutation variance increased in BTGP from $\sigma_m^2 \approx 0.002$ to $\sigma_m^2 \approx 0.01$ after 1,000 generations; this implies a greater role for mutation in generating and changing variation towards the end of the run. However, it may be that this is an artefact caused by the convergence of mean fitness towards a fixed lower boundary of zero.

Opportunity for selection (Figure 2) differed markedly between the two algorithms. In BTGP it increased from 0.05 to 0.09, while in MGA the corresponding increase was from 0.1 to 1.4. The more rapid increase measured in the MGA is a consequence of mean fitness decreasing more rapidly. These observations suggest that the MGA offers more opportunity for selection to act. It is interesting however that opportunity for selection does not decline during the run of either algorithm, suggesting that the redundancy built into each algorithm prevents early convergence.

We considered intensity of selection with reference to tree size in a number of nodes. If the number of nodes was under strong directional selection, we would expect a correlation coefficient, $r \approx \pm 1$, and high values for the regression coefficient. The absolute value of the correlation coefficient for BTGP reaches 0.9 and the corresponding regression coefficient is +0.3. In contrast, for MGA the correlation coefficient reaches a maximum absolute value of 0.6 with a corresponding regression coefficient of -0.6. In addition, the sign of each correlation coefficient is always positive for BTGP and almost always negative for MGA. The implication

Figure 2: Opportunity for selection for BTGP and MGA averaged over 10 runs



is that in BTGP larger trees tend to have higher fitness, while in MGA this is true of smaller trees. This characteristic probably results from the different manner in which trees are generated in the respective algorithms, and indicates that it may be easy to include implicit biases in the details of algorithm construction.

In BTGP epistasis variance displays the same spiky behaviour as the sampling error, $\sigma_v^2 - \sigma_A^2$, with similar magnitude. This means that the epistasis variance is being masked by sampling errors in the allele population and is in consequence unreliable. Measurement of bit-wise epistasis in MGA revealed that certain bits had high epistasis values relative to the others (approximately 7 times the magnitude of the average). On inspection it was found these bits played a crucial role in tree construction, linking nodes to one another. Thus changes in these bits relate strongly to bits determining other (linked) parts of the tree. The measure is useful in determining relative relatedness between bits in the encoding.

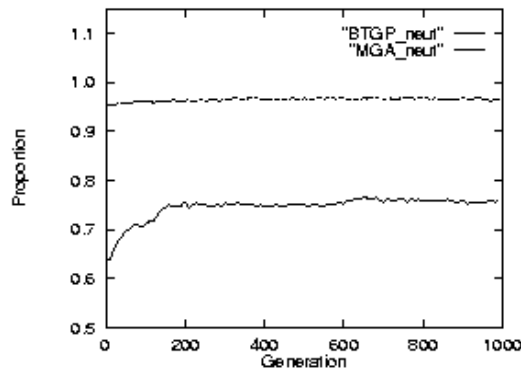
The proportion of fitter mutants (those fitter than the previous best individual) was found to be around 0.02 for BTGP. However, this reflects the fact that elitism was not used; if elitism is used the proportion of fitter mutants typically falls to 0.001 or less for BTGP (corresponding to a single individual). We would need to use a larger population size to generate a more meaningful and accurate measure.

The level of phenotypic neutrality (Figure 3) was very high for MGA (in excess of 0.95) whilst it was approximately 0.75 for BTGP. This permits higher levels of neutral exploration of genotype space using MGA, but suggests that neutral transitions in genotype space may be important in the function of both algorithms.

CONCLUSIONS AND FUTURE WORK

In the preceding sections we have compared the BTGP and MGA, and shown that their relative performance at information retrieval derives from different aspects

Figure 3: Proportion of neutral mutants that have no phenotypic effect averaged over 10 runs



of the algorithms as implemented (this needs to be specified and confirmed). In this chapter we have only provided the preliminary steps of such a comparative analysis, however the results presented here give indicators for the direction of future work.

MGA can be regarded as performing better as it attains a lower mean fitness at the end of its run. The measures we have calculated suggest that this may be a consequence of greater opportunity for selection, combined with increased neutrality with respect to phenotypic effect (Figure 3). However this does not mean that BTGP is ineffective at the information retrieval task: it shows high values of the above parameters as well.

Our results also indicate how we may learn about possible changes in the representation to improve algorithm performance. The calculation of bit-wise epistasis in MGA shows how the specific representation used gives us the structure of the fitness landscape: this information could be used to design a better representation for solving the current problem.

We have used some measures inspired by biological evolution. The difficulty of deriving and interpreting the results of the measures here highlights the differences between biological and computational evolution. In biological systems, the situation is typically one of limited information, both about the gene pool and individual genotypes, and about the phenotypes of organisms. In evolutionary computation much more complete information is available, in principle, although it may be computationally intensive to obtain it.

Despite this, measures derived from biological sources may be difficult to use, for several reasons. Evolutionary computation typically uses small populations, which may not include all the possible variation that could occur with respect to the character or gene under consideration. This can lead to differences between the properties of particular populations, as compared to the theoretical grand population, which reduce the usefulness of measures such as epistasis variance.

Furthermore, evolutionary algorithms typically are initialized with a random, diverged population, and spend much of their time away from convergence. This is in contrast to biological populations, in which most loci have converged to fixation, allowing the relatively few polymorphic ones to be considered in isolation. Evolutionary measures inspired by biology which focus on relatively few characters may need changing as a consequence.

An additional problem for measurement of evolutionary algorithms comes from the definition of genotype and phenotype, and their constituents, genes and phenotypic traits. Although the definition of a gene in a living organism is not entirely without ambiguity, there are generally agreed-upon protocols for identifying and defining genes. By contrast, the very flexibility of specifying the genotype and genotype-phenotype mapping in a genetic algorithm makes it difficult to define a gene. This problem is exacerbated in genetic programming systems such as BTGP, where the programs implement Boolean trees. Is a gene one type of Boolean function, with every instance of that function the same gene, or are we dealing with distinct genes? We have adopted a particular view in this chapter in order to implement our measures, but do not believe that this will be a universal solution.

In BTGP and MGA, it is also difficult to identify what relevant phenotypic traits to measure are. Indeed, in BTGP it could be argued that there is no distinction between phenotype and genotype. Problems of identifying appropriate phenotypic traits for measurement are likely to arise often when there is a complex mapping between genotype and phenotype. Complicated genotype-to-phenotype mappings have been identified by several authors (e.g., Bäck et al., 1997) as a likely route to more versatile applications of evolutionary algorithms. Consequently the identification of methods which focus upon relevant phenotypic variation is an important task for the future.

This chapter develops a range of evolutionary measures in order to gain insight into the workings of EAs. Since our measures derive from fundamental evolutionary attributes, this methodology should be extensible to a wide range of EAs.

Finally, we emphasize that we are interested in application of EAs to real-world problems. We do not see measures such as those we have investigated here as an end for evolutionary computation research in themselves, only as a preliminary contribution to a better understanding of problem solving by evolutionary algorithms.

REFERENCES

- Altenberg, L. (1994). The evolution of evolvability in genetic programming. In Kinnear Jr, K.E. (Ed.) *Advances in Generic Programming*, 47-74, Cambridge, MA: MIT Press.

- Altenberg, L. (1995). The schema theorem and Price's theorem. In L. D. Whitley & M. D. Vose (Eds.), *Foundations of Genetic Algorithms*, 23-50. San Francisco, CA: Morgan Kauffman.
- Arnold, S. J. & Wade, M. J. (1984). On the measurement of natural and sexual selection: Theory. *Evolution*, 38, 709-719.
- Bäck, T., Hammel, U. & Schwefel, H.-P. (1997). Evolutionary computation: Comments on the history and current state. *IEEE Transactions in Evolutionary Computation*, 1, 3-17.
- Bedau, M.A. & Packard, N.H. (1991). Measurement of evolutionary activity, teleology and life. In C.G. Langton, C. Taylor, J.D. Farmer & S. Rasmussen (Eds.), *Artificial Life II*, 431-461, Redwood City, CA: Addison-Wesley.
- Blickle, T., & Thiele, L. (1997) A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4, 361-394.
- Brodie, E. J. III, Moore, A. J. & Janzen, F. J. (1995). Visualising and quantifying natural selection. *Trends in Ecology and Evolution*, 10, 313-318.
- Crow, J. F. (1958). Some possibilities for measuring selection intensities in man. *Human Biology*, 30, 1-13.
- Davidor, Y. (1991). Epistasis variance: A viewpoint on GA-hardness, In G.J.E. Rawlins, (Ed.), *Foundations of Genetic Algorithms*, 1, 23-35, San Mateo, CA: Morgan Kauffman.
- Endler, L., (1986) *Natural Selection in the Wild*. Princeton, NJ: Princeton University Press.
- Falconer, D.S. (1989). *Introduction to Quantitative Genetics*, (3rd. ed.) Harlow, Longman.
- Fernández-Villacañas, J.L. & Exell, J. (1996). BTGP and information retrieval. *Proceedings of the Second International Conference ACEDC'96, PEDC*, University of Plymouth.
- Fonlupt, C., Robilliard, D. & Preux, P. (1998). A bit-wise epistasis measure for binary search spaces. In A. E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature—PPSN V*. 47–56. Berlin, Springer.
- Gavrilets, S. (1997). Evolution and speciation on holey landscapes. *Trends in Ecology and Evolution*, 12, 307-317.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*, Reading, MA: Addison-Wesley.
- Hofbauer, J. & Sigmund, K. (1988). *The Theory of Evolution and Dynamical Systems*, Cambridge: Cambridge University Press.
- Holland, J.H, (1992). *Adaptation in Natural and Artificial Systems*, Cambridge, MIT Press.

- Hordijk, W. (1997). A measure of landscapes, *Evolutionary Computation*, 4, 335-360.
- Huynen, M.A. (1995). *Exploring Phenotype Space Through Neutral Evolution*. Santa Fe Working Paper 95-10-100.
- Kauffman, S. A. (1993). *The Origins of Order*, New York: Oxford University Press.
- Lynch, M., Walsh, B. (1998). *Selection and Evolution of Quantitative Traits*. Sunderland, MA: Sinauer.
- Manderick, B., deWegner, M. & Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In R.K. Belew, & L.B. Booker, (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, 143-150. San Mateo, CA: Morgan Kaufman.
- Marrow, P. (1999). Evolvability: Evolution, computation, biology. In A. S. Wu, (Ed.), *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO-99) Workshop Program*, 30-33.
- Mitchell, M. and Forrest, S., (1995). Genetic algorithms and Artificial Life, *Artificial Life*, 1, 267-289.
- Mühlenbein, H. & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm. I. Continuous parameter optimization. *Evolutionary Computation* 1, 25-49.
- Mühlenbein, H. (1998). The equation for the response to selection and its use for prediction, *Evolutionary Computation* 5, 303-346.
- Nimwegen, E. V. & Crutchfield, J.P. (1998). *Optimizing Epochal Evolutionary Search: Population-Size Independent Theory*, Santa Fe Working Paper 98-06-046.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1994). *Numerical Recipes in C*, Cambridge University Press, 662.
- Roff, D. (1998). *Evolutionary Quantitative Genetics*, London: Chapman and Hall.
- Schuster, P. (1996). Landscapes and molecular evolution. In *Landscape Paradigms in Physics and Biology*.
- Sloman, (1998). Some notes on combinatorial search and how (not?) to tame it.
- Wagner, G.P. & Altenberg, L. (1996). Complex adaptations and the evolution of evolvability, *Evolution*, 50, 967-976.

Chapter XVII

Design Wind Speeds Using Fast Fourier Transform: A Case Study

Z. Ismail, N.H. Ramli and Z. Ibrahim,
Universiti Malaya, Malaysia

T.A. Majid and G. Sundaraj
Universiti Sains Malaysia, Malaysia

W.H.W. Badaruzzaman
Universiti Kebangsaan Malaysia, Malaysia

ABSTRACT

In this chapter, a study on the effects of transforming wind speed data, from a time series domain into a frequency domain via Fast Fourier Transform (FFT), is presented. The wind data is first transformed into a stationary pattern from a non-stationary pattern of time series data using statistical software. This set of time series is then transformed using FFT for the main purpose of the chapter. The analysis is done through MATLAB software, which provides a very useful function in FFT algorithm. Parameters of engineering significance such as hidden periodicities, frequency components, absolute magnitude and phase of the transformed data, power spectral density and cross spectral density can be obtained. Results obtained using data from case studies involving thirty-one weather stations in Malaysia show great potential for application in verifying the current criteria used for design practices.

INTRODUCTION

In the design of civil engineering structures, the effects of the natural environment on these structures have to be taken into consideration. Some examples of these are the effects of wind, current and waves on offshore structures; and the effects of wind and seismic activities on buildings. This chapter focuses mainly on the effects of wind on buildings. The parameter of interest in the design and construction of a structure is the design wind speed. This can be obtained from fundamental principles backed by verification through field studies of the dynamic characteristics of the structure. In many cases involving large structures, the input force cannot be created at will or be controlled. This shortcoming is overcome through ambient vibration testing and the use of Fast Fourier Transform (FFT) to convert the raw wind data into wind loads.

The chapter starts with a review of the general effects of wind on structures and the inhabitants. The parameters used in the design of structures including buildings are discussed. It is shown here that Fast Fourier Transform (FFT) which include Power Spectral Density (PSD), Cross Spectral Density (CSD) and Turbulence Intensities (TI) can be applied to derive the design parameters and subsequently, improve a wind code for structures. Examples of early studies on wind loads are given, and the limits of tolerance for civil engineering structures and the inhabitants therein are mentioned. It is noted that the criteria for design are more concerned with the human tolerance rather than the structural tolerance. In the design of structures, there is a need for a full understanding of the effects of wind at each stage of construction since the tolerance for the final structure could vary appreciably with the tolerance at each intermediate stage of the structure during construction. Vibration effects due to wind on structures are given, and examples of vibration effects on mechanical structures are also given as a comparison.

Methods of structural analysis and structural monitoring including vibration analysis and modal analysis are mentioned. Field tests including forced vibration methods as well as ambient vibration methods are described. Factors affecting the design of structures as well as the incentives to better understand the complex effects of wind on structures result in the approach to simplify these effects into components. These components can then be utilized in defining design wind speed and deriving design wind load, which can be used to develop the local design wind code for civil engineering structures such as buildings.

The analysis using FFT in this chapter can be taken one step further through frequency response function (FRF). FRF is the ratio of the output response to the input excitation force. This measurement is typically acquired using a dedicated instrument such as an FFT analyzer or a data acquisition system with software that performs the FFT. The input data in this case would be the measured wind speed using several anemometers, which can be converted into dynamic pressure

experienced by the structure. The output data would be the dynamic response of the structure measured using several transducers such as accelerometers. Once the data are sampled, the FFT is computed to form linear spectra of the input excitation and output response. Typically, averaging is performed on power spectra obtained from the linear spectra. The main averaged spectra computed are the input power spectrum, the output and input signals.

These functions are averaged and used to compute two important functions that are used for modal data acquisition, which are the FRF and the coherence. The coherence function is used as a data quality assessment tool which identifies how much of the output signal is related to the measured input signal. The FRF contains information regarding the system frequency and damping, and a collection of FRF contains information regarding the mode shape of the system at the measured locations. This is the most important measurement related to experimental modal analysis.

The FRF can be viewed in the form of acceleration or displacement experienced by the structure due to the wind speed. Information such as this as well as the mode shape obtained provides vital information from the design aspect. This will help to provide additional meaningful and significant engineering data to structural design engineers (Avitable, 2001).

BACKGROUND

Early Studies

Wind loads on structures have been studied over the last 400 years or so, starting with some empirical work by Newton. During the next 100 years, some work was done particularly on the determination of wind forces on objects of different shapes. By the middle of the eighteenth century, correlation for many structures had already been developed. Research work on windmills by Smeaton focused on wind moments, rather than wind forces. Experiments into wind loads on static lattice frameworks by Baker followed the Tay Bridge disaster in 1879. This demonstrated the need for more precise wind data. The present focus of interest by researchers is the study of atmospheric turbulence in wind tunnels, and in the investigation of the dynamic response of structures due to wind. In addition, the development of automatic aircraft landing systems demands research and understanding of the vertical and horizontal components of wind-speed correlation (Sachs, 1972).

Limits of Tolerance

In the past, the vibration of tall buildings has merely been considered in terms of its structural implications. However, there are structures today that vibrate in

certain wind conditions, which have not shown any structural ill effect but has given concern in other aspects. The vibration of buildings can have undesirable effects on the well being of its inhabitants. Although the natural bending frequencies of many tall structures are within the limits of between 0.1 Hz and 0.3 Hz, it is known that these oscillations can induce sickness.

Throughout the period when a structure is being erected, attention should be paid to the limits of structural tolerance for each stage of construction. A stable final structure may be susceptible to damaging wind loads at some stage in its erection, and this should be taken into account in its design and construction procedure. For a construction stage, a lower design wind speed than that for the completed structure may be used with the same element of risk, since each erection stage lasts for a relatively short period of time.

Civil Engineering Structures

Civil engineering structures like lattice towers which are used for a number of diverse purposes such as antennae for telecommunication, radio and television broadcasting, power transmission and lighting supports, electric power transmission lines or skyscrapers have always been affected by wind loads. A standard method on determining how these loads act and the effects of certain frequency of wind on high-rise structures need therefore to be carefully examined and calculated. This is to ensure structural safety. It is also for the purpose of monitoring the continued usefulness of the structures. A lattice tower needs to be designed taking into consideration the resonant dynamic response due to wind load, which arises when the natural vibration frequency (fundamental frequency) of the structure is low enough to be excited by the frequency of turbulence in the natural wind. However, in the design of lattice towers, there is a further requirement for extending the basis of design to include, more explicitly, the load effects, such as top deflection, bending moment and shear force on the structure. In addition, wind behavior depends on the structure and topology of the terrain, therefore for safe design of all these structures, knowledge of wind characteristics in uneven terrain is of vital importance (Simiu & Scanlan, 1977).

The British Standards and the Australian Standards have recommended Gust Response Factors (GRFs) for lattice towers for different load effects like bending moment for the design of main leg members, shear force for the design of the main bracing members and top deflection for the serviceability criteria.

Structural Analysis

Vibration measurements and analysis are made for a variety of reasons. It may be done to determine the natural frequencies of a structure. It may be done to verify an analytical model of a structure. It may be done to determine the dynamic

durability under various environmental conditions, or it may be done to monitor the condition of a structure under various loading conditions. As structural analysis techniques continually evolve and become increasingly sophisticated, awareness of the potential shortcomings in their representation of structural behavior also grows. This is especially so in the field of structural dynamics. The justification and technology exists for vibration testing and analysis of large civil engineering structures. However, large civil engineering structures are usually too complex for accurate dynamic analysis using manual computation. It is usual to use matrix algebra-based solution methods, employing the finite element method of structural modeling and analysis, on digital computers. All linear models have dynamic properties, which can be evaluated using techniques of dynamic analysis, such as modal analysis. The modal analysis technique can provide the natural frequencies and corresponding mode shapes for a numerical model of a structure. For an existing structure, the accuracy of such a linear finite element method model can be validated through comparison of these dynamic properties with those obtained from testing the actual structure. Vibration testing and analysis of an existing structure can therefore provide a quantitative evaluation of its dynamic properties.

Mechanical Engineering Systems

The theories of vibration testing and analysis are well established. Common applications are found in mechanical engineering, where it is used to study industrial machinery noise and vibration problems. The techniques commonly use some device whose express purpose is to artificially induce a force or displacement to excite the structure. Usually a controlled periodic, random, transient or impact force is used.

In the field of mechanical engineering, there are a number of integrated systems, which can handle the experimental testing, system identification and modal refinement. These systems are mostly based on forced vibration tests, which range from simple impact tests to complex test setups involving several exciters. Due to their relatively small size, most mechanical specimens can be tested in laboratories under controlled conditions. There is, however, no such luxury or advantage for the verification of dynamic models of large civil engineering structures. The procedure is relatively expensive, and with very long or massive structures, such as dams, it may be necessary to use more than one exciter, thereby increasing the costs for testing.

Large Structures

The integrated systems, developed for mechanical engineering applications, cannot be applied economically to large civil engineering structures such as bridges and buildings. Bridges form vital links in transportation networks and therefore

traffic shutdowns associated with forced vibration testing would be costly. Controlled forced vibration tests of buildings may disturb the occupants and may have to be conducted after working hours, which would also increase the cost of the testing. Therefore, routine dynamic tests of bridges and buildings must be based on ambient methods, which do not interfere with the normal operation of the bridge or the building.

Dynamic properties of a structure may vary over time. This could be as a result of changing material properties due to weathering or as a result of response to load history. Determination of dynamic properties before and after an extreme load event, such as an earthquake, may indicate changed conditions not evident by conventional means of evaluation, such as visual inspection or standardized material non-destructive testing. Vibration testing can therefore be considered for use as a monitoring tool, for providing dynamic properties over time, which can be studied to identify structural changes.

Generally, analytical models of existing large structures are based on geometric properties taken from old drawings and material properties obtained from the structure. A series of assumptions are also made to account for the surrounding medium and its interaction with the structure such as soil-structure interaction in the case of buildings and bridges, soil-water-structure interaction in the case of dams, wharves and bridges and the composite behavior of structural elements. This, in general, is not the case for mechanical systems.

Vibration Testing Techniques

Two techniques are available for vibration testing of large structures: forced and ambient vibration techniques. Both forced vibration and ambient methods have been used in the past and are capable of determining the dynamic characteristics of structures. Forced vibration methods can be significantly more complex than ambient vibration tests, and are generally more expensive than ambient vibration tests. The main advantage of forced vibration over ambient vibration is that for the former, the level of excitation and induced vibration can be carefully controlled, while for the latter, one has to rely on the forces of nature and uncontrolled artificial forces such as vehicle traffic on bridges. Sometimes the structure can only be excited to a very low level of vibration. As a result, the sensitivity of sensors used for ambient vibration measurements needs to be much higher than those required for forced vibration tests.

Forced Vibration Testing

By definition, a forced vibration test constitutes the use of any source of controlled excitation applied to a large structure in order to induce vibrations. Ambient tests may be used to test bridges, nuclear power plants, offshore platforms

and buildings. Although ambient tests do not require traffic shut downs or interruptions of normal operations, the amount of data to be collected is significant and it can take several weeks to analyze all these data thoroughly. The techniques for data analysis are also different from the forced vibration analysis technique. The theory for forced vibration tests of large structures is well developed and is a natural extension of the techniques used in forced vibration tests of mechanical systems. In contrast, the theory and technique for ambient vibration tests still requires some further development.

Forced vibration tests are generally conducted to determine the dynamic characteristics of both simple and complex systems. In these tests, controlled forces are applied to a structure to induce vibrations. By measuring the structure's response to these known forces, the dynamic properties of the structure can be determined. Controlled excitation forces can be applied to a structure using several different methods. The three most popular methods are:

- i) *Shaker Tests*: Shakers are used to produce sufficiently large forces, to effectively excite a large structure in a frequency range of interest. For large structures, such as bridges or tall buildings, the frequencies of interest are commonly less than 1 Hz. At such low frequencies, a shaker cannot generate sufficiently large forces, and while it may be possible to build such massive, low frequency shakers, these are expensive to construct, transport and mount.
- ii) *Impact Tests*: Impact testing is to identify the dynamic characteristics of machine components and small assemblies. The test article is attached with an accelerometer, and the hammer, which is used to strike it, is attached with a force transducer. The impact force and acceleration response time histories are used to compute frequency response functions (FRFs) between a measured point and the point of impact. These FRFs can be used to determine the natural frequencies, mode shapes and damping values of the structure using well-established methods of analysis of impact test data.
- iii) *Pull Back Tests*: The pull back or quick-release testing method generally involves inducing a prescribed displacement to a structure and quickly releasing it, causing the structure to vibrate freely. The objective of this technique is to quickly release the load and record the free vibrations of the structure, as it tends to return to its position of static equilibrium.

Ambient Vibration Testing

Ambient vibration analysis is a vibration testing and analysis technique, which can be targeted for large civil engineering structures. Since the method requires no artificial excitation to be imparted to the structure being tested, and relies on the naturally occurring ambient vibrations, this provides a distinct cost advantage over other methods.

When ambient vibration testing is considered, a structure may be excited by wind, by micro tremors, by machinery or by traffic. Unlike forced vibration testing, the excitation forces cannot be controlled. Natural frequencies and mode shapes are obtained by measuring the vibrations of the structure simultaneously at several locations on the structure. Natural frequencies cannot be evaluated for each independent degree of freedom since information on the excitation is not known. Alternative methods of analysis of ambient vibration data are then to be utilized to identify relevant dynamic characteristics of the structure.

Severe environmental natural excitations such as earthquakes, windstorms and large waves can also be considered ambient vibrations, except that the level of motion is much higher and that the source of the excitation can be known. In some cases it can also be measured. However, the occurrence of such severe excitations cannot be controlled or predicted, and the vibrations of the structure can only be captured if permanent instrumentation is placed in the structure and set-up to record vibrations at prescribed levels of shaking. In many seismically active areas, buildings, bridges and dams are fixed with instruments capable of measuring severe shaking at different locations within the structure. Modal analysis techniques can be used to identify the dynamic properties of such structures from recorded earthquake motions.

The methods that have been developed for analyzing data from forced and ambient vibration tests range from linear deterministic models to nonlinear stochastic models. The applications range from improving mathematical models of systems to damage detection, identifying the input of a system to controlling its response. Parameter estimation methods using dynamic signals can be classified as time-domain methods, frequency-domain methods and joint frequency-time domain methods (Ventura, 2001).

Effects of Wind

Over the centuries, wind has caused considerable damage to buildings and structures and induced collapse in many. The trend to more economic design using less material accentuates the relative importance of wind loads as compared to gravitational loads, and this trend has greatly accelerated over the past few decades. The situation with structural design currently is such that the assessment of environmental loading may represent the greatest unknowns in the design of a proposed structure. The nature of the wind in the earth's boundary layer is complex. The variation of the average wind speed is usually approximated in calculations by a power law and depends on such factors as the surrounding ground and buildings and the general synoptic weather pattern (Houghton & Carruthers, 1977).

The prediction of the wind response of a skeletal structure is becoming more and more important in structural design due to the sensitivity of such structure to

wind loads. Some data have been reported based on wind tunnel tests. However, reliable data from full-scale measurements are scarce. Hence a full-scale field experimental program using instruments such as anemometers, accelerometers and strain gauges should be taken up using a lattice tower to study the dynamic behavior of the structure under wind loads.

For convenience of analysis, wind speed is broken down into a mean component and a fluctuating component. While the mean speed component is assumed to result in static wind pressure and corresponding steady deflection, the fluctuating component gives rise to dynamic amplification. Modern codes of practice provide criteria for determining the design wind speed depending on ground roughness, building size, height above ground, required life span of the structure and the topography of the site. These are intended to cover the contingencies affecting the incident wind and the way in which gustiness influences loading. However, the magnitude of the design wind speed may well be the most uncertain element of a wind-load calculation.

Turbulence in the wind produces fluctuating loads, which can cause motion that is mainly in the wind direction. Vortices shed alternately from either side of a tall structure produce a force in the crosswind direction that is roughly sinusoidal and this can cause vibration of a structure of corresponding natural frequency. When the structure is stationary, the vortex-induced force has a broad bandwidth but when it is oscillating above certain amplitude, the vortex shedding locks-on to the vibration and the exciting force has an almost sinusoidal form (i.e., a sharp peak in its spectrum). Winds that are not necessarily strong can cause structures to vibrate in ways such that, even if not disastrous, can still cause structural deterioration, fatigue problems or human discomfort.

The main objective is to always provide, by the most economical means, predictions of wind effects that are sufficiently accurate for the civil engineer. In a significant number of cases, the criteria are set by the tolerance of humans and as such are not well defined. In the case of wind speeds around pedestrian areas for example, values between 2 m/s and 9 m/s have been suggested as upper limits. However, the turbulence, the air temperature and humidity are also important. A steady breeze of 2 m/s may be of no consequence whereas a breeze of 2 m/s fluctuating in direction would be intolerable.

Design Wind Speed

The design wind speed at any given station in the United States is defined as the peak gust at 30 ft above ground record at that station. This definition was adopted in the Uniform Building Code. It follows from an equation that the design wind speed implicit in the uniform Building Code is approximately equal to the 39-year gust wind.

The design wind speed specified by the BS and ANSI are 50-year fastest miles for most permanent structures, 100-year fastest miles for structures with an unusually high degree of hazard to life and property in case of failure, and 25-years fastest miles for structures having no human occupant or where there is negligible risk to human life. In the light of past experience, it may be stated, however, that both the 39-year peak gust and the 50-year fastest mile criteria result in wind loads that appear to ensure a reasonable degree of structural safety. Some criteria to be adopted in obtaining the corresponding values for the Malaysian Standard (the case study) for Wind Loads have to be developed.

Gust Response Factors

Initially, the design wind speeds are obtained by multiplying the mean wind speeds with gust velocity factors. The mean wind speeds are used to allow for fluctuations in the wind speed. However, neglecting both the dynamic properties and size of the structure could result in an unsafe structure or a conservative over-designs of a structure. The structural loads produced by wind gusts depend on the size, natural frequency and damping of the structure. The structural failure, which is directly attributable to gust action, emphasizes the importance of these parameters in arriving at the gust wind load. The gust response factors (GRFs) that will account for influence of these important parameters, are a measure of the effective dynamic load produced by gusts, and are intended to translate the dynamic response phenomena produced by gust loading into a simpler factored static design criteria.

Currently, the wind sensitive structures are designed using a semi-analytical approach with a simple model relating the upwind turbulent velocity fluctuations and the fluctuating forces on the structure. In this approach, the dynamic response is treated using random vibration theory and modal analysis. In most of the international design codes and standards, the GRF for the modal coordinate is computed using the above approach and the same value is considered for all other load effects such as bending moment, shear force, etc. It is also assumed that the first mode shape of the structure varies linearly with height, and the contribution of higher modes of vibration is neglected. This makes the GRF constant for the whole height of the structure.

Malaysian Wind Code (The Case Study)

Malaysia is situated close to the equator and is outside the belt of severe tropical cyclones. The monsoon winds are also mild. Therefore, winds due to these weather systems are relatively not severe. However, equatorial regions are prone to tropical thunderstorms. Wind speeds of the gust fronts of these thunderstorms can be relatively high.

There are several codes of practice currently adopted in Malaysia for wind loads. These are AS117-2-1989 (SAA Loading Code: Part 2 Wind Loads), ASCE 7-95: Minimum Design Loads For Building and Other Structures and BS 6399. Generally, for wind load determination on buildings, the British Standard (CP3: Chapter V: Part 2: Sept. 1972, Code of Basic Data For the Design Of Buildings Chapter V, Loading, Part 2: Wind Loads) is adopted.

COMPUTATIONAL CONCEPTS

Fourier Transform

Time and frequency are two different ways of expressing the characteristics of a signal. Both the time signal itself and a frequency analysis of it present the same information since each frequency domain point is derived from the entire time domain signal. The frequency domain provides an alternative perspective for characterizing the behavior of oscillating and vibrating functions.

Waveform that exists could be generated through adding sine waves. Real-world signals can be broken down into these same sine waves and it can be shown that this combination of sine waves is unique. For measurement of any parameter, to detect a small sine wave in the presence of large signals, it is better to use frequency domain. When these components are separated in the frequency domain, the small components are easily seen because larger ones do not mask them.

Fourier transform/integral is the primary tool to analyze a periodic waveform, the waveforms that do not repeat themselves regularly. The essence of the Fourier transform of a waveform is to decompose or separate the waveform into a sum of sinusoids of different frequencies. If these sinusoids sum to the original waveform then Fourier transform of the waveform has been determined. The pictorial representation of the Fourier transform is a diagram that displays the amplitude and frequency of each of the determined sinusoids. The Fourier transform identifies or distinguishes the different frequency sinusoids (and their respective amplitudes) that combine to form an arbitrary waveform.

A physical process x , in the *time domain* given by $x(t)$, can also be described in the *frequency domain* with its amplitude X and frequencies f , by $X(f)$. Mathematically, this relationship is stated as:

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{-2\pi i f t} df$$

which is a linear operation.

Fourier transform equation transforms functions coming from, and extending to infinity. In real-world application, however, this is not practical. More often than not, only a small section of this continuum is to be transformed. Furthermore, analysis on a digital computer requires the function to be sampled discretely in time. Thus, a continuous waveform will be represented as a series of impulses whose magnitude is equal to the amplitude of the waveform for that time step, each separated by a constant interval determined by the sampling rate.

The sampling rate is a very important factor when considering analyzing some function of time. The Nyquist theorem, $f_c = 1/2$, states that the critical frequency f_c , or the maximum frequency seen in the sampling process, is half that of the sampling rate. If the signal on samples is not bandwidth limited to f_c , a process called aliasing occurs from which information from above f_c is folded back into the sampling bandwidth. This results in an incorrect transform. High-quality-low-pass filters must be used to artificially bandwidth limit the waveform before sampling.

There are two types of Fourier transforms: Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT). For N values of data, the DFT requires N^2 complex operation. For data samples of moderate size, the direct determination of the DFT can be extremely time-consuming. The FFT is an algorithm that has been developed to compute the DFT in economical fashion. It utilizes the results of previous computation to reduce the number of operations. In particular, it exploits the periodicity and symmetry of trigonometric functions to compute the transform with approximately $N \log_2 N$ operations (Brigham, 1974).

Power Spectral Density

Spectral analysis, sometimes called 'spectrum analysis,' is the name given to methods of estimating the spectral density function, or spectrum of a given time series. The power spectral density, PSD, describes how the power (or variance) of a time series is distributed with frequency. Mathematically, it is defined as the Fourier Transform of the auto correlation sequence of the time series. Spectral analysis is not only concerned with looking for 'hidden periodicities' in the data, but also with estimating the spectrum over the whole range of frequencies.

Spectral analysis is mainly concerned with purely indeterministic or stochastic series, which have a continuous spectrum, but also can be used for deterministic series to pick out periodic component in the presence of noise. Deterministic series is referred to as a time series, which can be predicted exactly. But most time series are stochastic, that is the future is only partly determined by past values. For stochastic series, exact predictions are impossible and must be replaced by the idea that future values have a probability distribution, which is conditioned by knowledge of past values. Wind speed data are categorized as stochastic series.

A natural way of estimating the power spectral density function is by using the periodogram. The averaged periodogram is called the spectrum of the data provided, in this case, the wind speed. It gives the distribution of the variance of wind speed as function of frequency. The variance of each point is equal to the expected value at the point. By averaging together 10-30 periodograms, the uncertainty in the value at each frequency can be reduced.

In this chapter, interpretation of the wind speed power spectrum plots consists of identifying the frequency ranges over which noticeable peaks in power or energy occur, observing any trends and comparing energy levels for different wind records (Nezih & Davras, 1983).

Cross-Spectral Density

In another context, the cross-spectral density function is a technique for examining the relationship between two time series over a range of frequencies in the frequency domain. The cross-spectrum of a discrete bivariate process measured at unit intervals of time is defined as the Fourier transform of the cross-covariance function.

The ordinate value in all the cross-spectrum plots represents the gain factor, which is essentially a regression coefficient of the second time series on the first. For this project, cross-spectral density is applied in the situation of wind speed records at two different heights, i.e., 43.9m and 28.1m for both type of data: daily maximum and 10 minute intervals.

Turbulence Intensity

Turbulence intensity is a measure of the amplitude of the velocity fluctuations, which occur in the flow. It is proportional to the frequency of turbulent or eddies angular velocities, which is a rough estimate of the degree of violence of turbulent fluctuation. The energy content of a large eddy is much greater than that of a small one for a given intensity, and energy is transferred from larger eddies to smaller ones.

Turbulence intensity is calculated by dividing the standard deviation of the wind speed with the mean value of the wind speed. One of the causes of atmospheric turbulence is terrain roughness; hence it must also be an important factor affecting the intensity of turbulence. Its effect is twofold. Firstly, the increase of site roughness will increase the turbulent intensity. Secondly, for the same terrain roughness, the intensity of turbulence is height dependent: that is, it decreases with increase of altitude.

Applications

Adherence to current Code of Practice has not completely protected structures from failures. Codes have to be continually updated and improved. At the same

time, there is a need to be a balance between degree of safety and economics. In hurricane areas for instance, it may be uneconomical to build against a storm, which may never happen. The approach here is to see if Fast Fourier Transform (FFT) which includes Power Spectral Density (PSD), Cross Spectral Density (CSD), and Turbulence Intensities (I), could produce results that could improve the basis of the code. Meteorological data, including wind speed, is a local phenomenon, which differs from one place to another. Design codes developed elsewhere, such as the British Standard (BS) and the American National Standard (ANSI), are therefore not suitable for the Malaysian case. Geological differences in terms of different climates and topography, and choice of life duration of a structure, are significant features that cannot be applied globally. We need to establish a code of practice of our own in order to derive more realistic structural analysis and design.

The study of wind loads by using computational intelligence algorithm is a new method to be explored and applied in structural engineering field. Norville et al. conducted such a study. The work concentrated on validation of wind data recorded at the Moro test site by the Bonneville Power Administration. The scope of the project was mainly concerned with determining wind and response characteristics. The analysis includes plots of time histories, histograms, power spectra, cross spectra and determination of statistical properties. The analysis in frequency domain is done by FFT algorithm using the International Mathematical and Statistical Libraries (IMSL) routines (Norville et al., 1985).

Liew also conducted the same study. The work focused on frequency domain analysis of the energy contents of the wind speed. The analysis was based on actual time series and on the selected model of time series and was performed in power spectrum plot (Liew, 1977).

DATA COLLECTION

Two sources of data were studied in this project. First, data of 95% confidence level of maximum wind speed, which recorded annually, for 30 different stations throughout the whole Malaysia were obtained from Malaysian Meteorological Stations (MMSs). Second, data recorded in an interval of every 10 minutes and a daily maximum within a certain period in the year 1998 and 1999 obtained from a meteorological station, which is situated in Universiti Malaya (UM).

However, only a total of eleven case studies were carried out to see the effect of FFT on wind speed data. Seven cases came from the MMS, which were collected for the years 1948-1998; and four came from the UM station, which has a wind tower with two anemometers, one at 28.1 meter (WS1) and another one at 43.9 meter (WS2). These four data consisted of two daily maximal data, and another two were based on wind speed data with 10-minute intervals. The data

used are not raw data; instead, a model data modeled as ARIMA (1,1,0) with the parameter equal to -0.29451631, which according to Liew is generally sufficient as a representative of the actual power spectrum (Cheong, 2000).

DATA PROCESSING AND ANALYSIS

MATLAB was utilized in this study.

Fast Fourier Transform (FFT)

In MATLAB, FFT is a built-in function. FFT computes the discrete Fourier transform of a vector or matrix. When the sequence length is a power of two, FFT uses a high-speed radix-2 FFT algorithm. The radix-2 FFT routine is optimized to perform a real FFT if the input sequence is purely real; otherwise it computes the complex FFT. This causes a real power-of-two FFT to be about 40% faster than a complex FFT of the same length. When the sequence length is not an exact power of two, a sequence algorithm finds the prime factors of the sequence length and computes the mixed-radix discrete Fourier transform of the shorter sequences.

Power Spectral Density (PSD) and Cross-Spectral Density (CSD)

In MATLAB, PSD estimates power spectral density of a signal and CSD estimates cross-spectral density of two signals. $Pxx=psd(x)$ estimates the power spectrum of the sequence x . $Pxy=csd(x,y)$ estimates the cross-spectral density of the length n sequences x and y . Both of the functions above use the Welch method of spectral estimation, which uses certain values of $nFFT$, F_s , $window$ and $noverlap$. $nFFT$ specifies the FFT length that PSD or CSD uses to determine the frequencies at which the power spectrum or cross spectrum is estimated. F_s is a scalar that specifies the sampling frequency. $window$ specifies a windowing function and the number of samples PSD or CSD uses in its segmenting of the x and y vectors. $noverlap$ is the number of samples by which the segments overlap. Welch method applied Hanning window with non-overlap and 95% confidence interval (Thomas et al., 1994).

RESULTS AND DISCUSSIONS

The data processing and analysis included periodogram, PSD, turbulence energy, TI and cycle lengths. Figure 1- Figure 3 are examples of results from one of the seven stations, namely Kuala Terengganu.

Figure 1: Periodogram of wind speed data (Kuala Terengganu, from years 1948 to 1998)

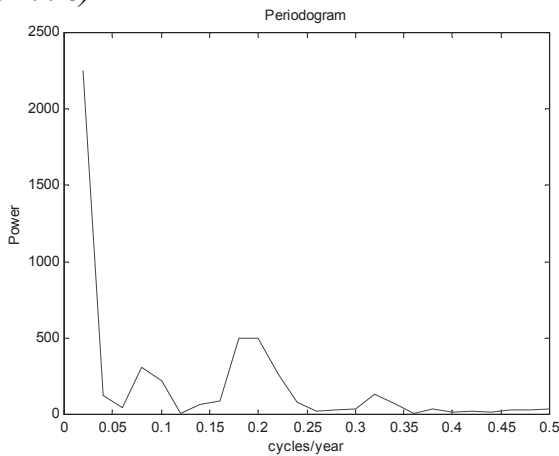
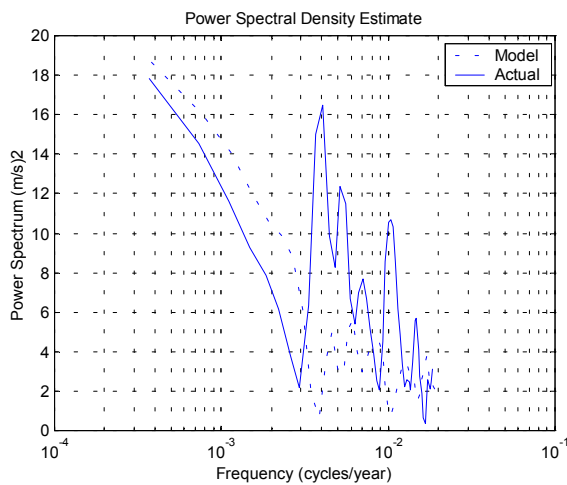


Figure 2: Plot of power against period (Kuala Terengganu, from years 1948 to 1998)



The periodogram is an initial means of estimating PSD function. The average of 10-30 periodograms is the spectrum of the wind speed. The PSD indicates the change of variance of fluctuations in the winds with frequency of contributory wave. In this study, interpretation of wind speed PSD plots consists of identifying the frequency ranges over which noticeable peaks in energy occur, observing any trends and comparing energy levels for different wind records.

Variation of TI with height above mean sea level for these stations cannot be compared due to different terrain roughness. From the point view of safety, wind

Figure 3: Plot of power against period (Kuala Terengganu, from year 1948 to 1998)

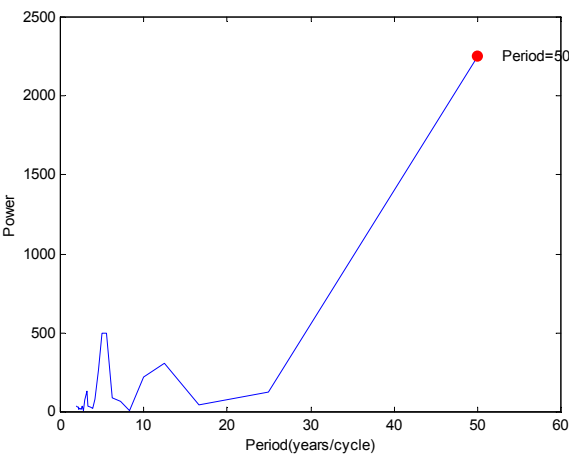


Table 1: Results of cycle length of wind speed recorded for all types of data

Station	Cycle length
Kuala Terengganu	50 years
Bayan Lepas	2.2727 years
Ipoh	4.4545 years
Mersing	47 years
Alor Star	49 years
Sandakan	44 years
Sitiawan	50 years
Daily maximum WS1	2.4545 days
Daily maximum WS2	2.44 days
10-minute interval WS1	143.6667 min
10-minute interval WS2	143.3333 min

velocity used in the analysis of a structure subjected to wind force should not be the maximum observed so far, but one, which may occur once in “cycle length” years depending on the importance of the structure. The cycle lengths for all data have been computed by FFT algorithm. The results are tabulated in Table 1.

CSD function is a tool for examining the linear relationship between two time series over a range of frequencies. CSD is only applied in the situation of wind speed records at two different heights, which are 43.9m and 28.1m for both types of data: daily maximum and 10-minute intervals. Direct comparison can be done for the UM data. Both data show that at the higher level, WS1, energy spectrum is greater due to lower viscosity effects and smaller site roughness, i.e., less obstruction.

The results for maximum power, power spectral density and frequency recorded for all stations are given in Table 2. The power spectral density indicates the change in variance of the fluctuations in the signal with the frequency of the contributory wave. It is useful to plot the product of power spectra against the logarithm of the frequency. The advantage is that the area under the curve between any two frequencies gives the true measure of the energy in that frequency range. Generally, the maximum turbulent energy for these stations is between the power spectral range of 12 to 24 m/s². Comparison between these seven stations shows that wind data in Sitiawan contains the maximum turbulent energy; meanwhile the minimum energy observed is at Bayan Lepas. The results for mean wind speed, standard deviation and turbulence intensity obtained for all stations are given in Table 3.

Turbulence intensity (TI) is calculated by dividing the standard deviation of the wind speed with the mean wind speed. TI is used to measure the amplitude of the

Table 2: Results of MP, PSD and frequency recorded from all stations

Station	Maximum Power (MP)	Frequency of MP (cycles/year)	Power Spectral Density (PSD) (m/s) ²	Frequency of PSD (cycles/year)
Bayan Lepas	2250	0.025	16	0.004
Ipoh	1500	0.044	12	0.0012
Mersing	3000	0.022	24	0.002
Alor Setar	175	0.025	15	0.008
Sandakan	4500	0.025	24	0.003
Sitiawan	2400	0.018	15	0.0025
Kuala Terengganu	1900	0.4	24	0.0015
	Maximum Power (MP)	Frequency of MP (cycles/day)	Power Spectral Density (PSD) (m/s) ²	Frequency of PSD (cycles/day)
Daily Maximum WS1	700	0.4	7	0.0017–0.0028
Daily Maximum WS2	550	0.4	7	0.0017–0.0028
	Maximum Power (MP)	Frequency of MP (cycles/10 min)	Power Spectral Density (PSD) (m/s) ²	Frequency of PSD (cycles/10 min)
WS1 (10 minute)	30 000	0.008	2.25	0.0001
WS2 (10 minute)	18 000	0.008	1.4	0.0001

Table 3: Results of mean wind speed, standard deviation and turbulence intensity obtained from all stations

Station/Type of Data	Mean Wind Speed (m/s)	Standard Deviation	Turbulence Intensity
Mersing	23.3	1.0	0.044
Ipoh	23.2	4.0	0.174
Sandakan	17.1	1.8	0.104
Sitiawan	18.3	2.7	0.149
Kuala Terengganu	20.8	2.0	0.095
Alor Setar	20.4	2.3	0.115
Bayan Lepas	20.7	2.6	0.125
Daily maximum WS1	6.5	1.8	0.272
Daily maximum WS2	6.2	1.7	0.276
WS1 (10 minute)	1.3	1.0	0.722
WS2 (10 minute)	1.2	0.7	0.597

velocity fluctuations which occur in the flow and proportional to the angular velocities of the eddies, which is a rough estimate of the degree of violence of turbulent fluctuation.

One more important feature shows by almost all data that most of the energy of the turbulence eddies is concentrated in the lower frequency range. The spectra obtained for low frequencies, therefore, results in a more rational static analysis than the one based on extreme mean winds. Mean wind spectra are useful for arriving at design mean wind speed required for computing basic pressure as incorporated by various design codes.

CONCLUSION

From the study, mean wind spectra covering the entire range of frequencies have been obtained. The presentation of mean wind speed in the form of spectra provides important information in respect to energy at different frequencies at which the spectral values are the highest. The maximum scale of eddy motion that contains the maximum energy is one of the most important indices for structural design purpose. The major factor affecting the response of structures under wind loading is the dynamic characteristic of atmospheric turbulence. This can best be represented by its correlation functions and spectral functions.

FUTURE WORK

The following study gives a clear view on the transformation of wind speed data into a more useful engineering tool. This result can also be applied in wind energy

studies. Further studies and research are required on different types of wind conditions to see the effects of FFT.

REFERENCES

- Avitable, P. (2001). Experimental modal analysis. *Sound & Vibration: Structural Analysis*, 35(1), 20–31.
- Brigham, E.O. (1974). *The Fast Fourier Transform*. NJ: Prentice Hall.
- Cheong, W.V. (2000). *Time Series Modelling of Wind Speed in Malaysia*. Graduation Thesis. Department of Civil Engineering, University of Malaya, Kuala Lumpur.
- Houghton, E.L. & Carruthers, N.B. (1976). *Wind Forces on Buildings and Structures*. London: Edward Arnold Ltd.
- Liew, S.H. (1997). *Time series analysis and frequency contents of wind loads*. *Journal of Institution of Engineers, Malaysia*, 58(1).
- Nezih, C.G. & Davras, Y. (1983). *Discrete Fourier Transformation and Its Application to Power Spectra Estimation*. Amsterdam, Netherlands: Elsevier Scientific Publishing.
- Norville, H.S., Metha, K.C. & Farwagi, A.F. (1985). *500 kV Transmission Tower/Conductor Wind Response*. Report Submitted to the Bonneville Power Administration, Texas Tech University, Lubbock, Texas.
- Sachs, P. (1972). *Wind Forces in Engineering*. Oxford, UK: Pergamon Press.
- Simiu, E. & Scanlan, R.H. (1977). *Wind Effects On Structures*. CA: John Wiley & Sons.
- Thomas, P.K, Loren, S. & John, N.L. (1994). *Signal Processing Toolbox User's Guide*. The MathWorks. Inc.
- Ventura, C.E. (2001). *Overview of Vibration Testing of Large Structures*. Course Notes on Modal Identification of Output-Only Systems, Orlando, Florida, USA.

About the Authors

Masoud Mohammadian has completed his Bachelor, Master and PhD in Computer Science. His research interests lie in adaptive self-learning systems, fuzzy logic, genetic algorithms, neural networks and their applications in robotics, control, industrial automation, financial and business problems which involve real time data processing, planning and decision making. He is a member of over 30 international conferences and he has chaired several international conferences in computational intelligence and intelligent agents. He is currently a senior lecturer at the school of computing at the University of Canberra in Australia. He is a member of many professional (computing and engineering) organizations. He is also currently the vice chair of the Institute of Electrical and Electronic Engineering (IEEE) ACT section.

Ruhul Sarker received his PhD in 1991 from DalTech, Dalhousie University, Halifax, Canada, and is currently a senior lecturer in Operations Research at the School of Computer Science, University of New South Wales, ADFA Campus, Canberra, Australia. Before joining at UNSW in February 1998, Dr. Sarker worked with Monash University, Victoria, and the Bangladesh University of Engineering and Technology, Dhaka. His main research interests are Evolutionary Optimization, Data Mining and Applied Operations Research. He was involved with three edited books either as editor or co-editor, and has published more than 80 refereed papers in international journals and conference proceedings. He is also the editor of ASOR Bulletin, the national publication of the Australian Society for Operations Research.

Xin Yao received the BSc degree in computer science from the University of Science and Technology of China (USTC), Hefei, the MSc degree in computer science from the North China Institute of Computing Technologies (NCI), Beijing, and the PhD degree in computer science from the USTC, Hefei, in 1982, 1985, and 1990, respectively. He is currently a professor of computer science at the University

of Birmingham, Birmingham, England. Xin Yao is an associate editor or a member of the editorial board of six international journals, including IEEE Transactions on Evolutionary Computation, and an editor/co-editor of nine journal special issues. His major research interests include combinations between neural and evolutionary computation techniques, evolutionary learning, co-evolution, evolutionary design and evolvable hardware, neural network ensembles, global optimization, simulated annealing, computational time complexity and data mining.

* * *

Hussein A. Abbass gained his PhD in Computer Science from the Queensland University of Technology, Brisbane, Australia. He also holds several degrees including Business, Operational Research, and Optimisation and Constraint Logic Programming, from Cairo University, Egypt, and Artificial Intelligence, from the University of Edinburgh, UK. He started his career as a systems administrator. In 1994, he was appointed associate lecturer at the Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Egypt. In 2000, he was appointed lecturer at the School of Computer Science, University of New South Wales, ADFA Campus, Australia. His research interests include Swarm Intelligence, Evolutionary Algorithms and Heuristics where he develops approaches for the Satisfiability problem, Evolving Artificial Neural Networks, and Data Mining. He has gained experience in applying Artificial Intelligence Techniques to different areas including Budget Planning, Finance, Chemical Engineering (heat exchanger networks), Blood Management, Scheduling, and Animal Breeding and genetics.

C. Alippi obtained the DrIng degree in Electronic Engineering *summa cum laude* in 1990 and the PhD in Computer Engineering in 1995, both from Politecnico di Milano, Milano, Italy. His further education includes research work in computer sciences carried out at the University College London and the Massachusetts Institute of Technology. Currently, C. Alippi is an associate professor in Information Processing Systems at the Politecnico di Milano. His interests include neural networks (learning theories, implementation issues and applications), composite systems and high level analysis and design methodologies for embedded systems. His research results have been published in more than 80 technical papers in international journals and conference proceedings. He is a senior member of IEEE.

T. G. B. Amaral, received the DiplIng and MS degrees in Electrical Engineering from the Faculty of Science and Technology (FCT), University of Coimbra, Portugal, in 1993 and 1997, respectively. He is currently pursuing the PhD degree,

at the FCT. Since 1996 he is a member of the teaching staff at Electrical Engineering Department of Superior Technical School of Setúbal – Polytechnic Institute of Setúbal. He is currently adjoint professor in the Electrical Engineering Department of Superior Technical School of Setúbal – Polytechnic Institute of Setúbal. His interests include computer vision, image processing, modeling and control of dynamic system.

W.H.W. Badaruzzaman obtained his BSc (Hons) Civil & Structural Engineering and MSc Structural Engineering from the University of Bradford, UK, in 1984 and 1986, respectively. Completed his PhD degree in Structural Engineering at University of Wales, Cardiff, UK, in 1994. Currently, the head of the Department of Civil & Structural Engineering, University Kebangsaan Malaysia. A corporate member of the Institution of Engineers Malaysia (IEM) and a registered professional engineer with the Board of Engineers Malaysia. Is actively involved with the working Group Committee in Wind Loads for building structures in Malaysia.

J.-M. Bauschat, a native of Germany, studied Aircraft Engineering and completed his final examination at the University of Braunschweig, Germany. In 2001 he joined the teaching staff at the Technical University of Berlin, leading subjects in flight mechanics, experimental flight mechanics, and flight testing using a Dornier 128 and the German Aerospace Center (DLR) flight test-bed ATTAS (Advanced Technologies Testing Aircraft System). Prior to his current position, he served as a scientist at the Institute of Flight Systems of the DLR and project director of the DLR project ATTAS In-Flight Simulation. He also previously served as head of the DLR Group of Applied Flight Control. He has published articles in 18 referred publications.

Judith Bishop is professor of Computer Science at the University of Pretoria, South Africa, a position she has held since 1991. She has a PhD from Southampton University, UK, in the area of code generation for new computer architectures. Her research interests are programming languages, distributed systems and web technology. She is co-editor of *IEEE Software* and on the editorial board of several other journals. She chairs the IFIP committee WG2.4 on Software Implementation Technology and is South Africa's representative on IFIP Technical Committee 2 on Programming. She has served on many international and local programme committees and advisory boards and is a strategic advisor on Information Technology to the National Research Foundation.

Pierre Borne is professor “de Classe Exceptionnelle” at the “Ecole Centrale de Lille”; director of Research of this institution; and head of the Automatic Control

Department. He has been president of IEEE/SMC society (2000-2001) and has been IMACS vice president (1988-1994). He is chairman of the IMACS Technical Committee on “Robotics and Control Systems.” He was nominated a fellow of IEEE in 1996 and received the IEEE Norbert Wiener Award in 1998. He is author or co-author of more than 250 journal articles, book chapters, a scientific dictionary and communications in international conferences and 14 books on automatic control. He is a fellow of the Russian Academy of Non-Linear Sciences. He is listed in the *Who's Who in the World*. In 1997, he was nominated for the “Tunisian National Order of Merit in Education” by the president of the Tunisian Republic, and in 1997 he was named an honorary member of the IMACS board of directors. In 1999, he was promoted in France to “Officier dans l’ordre des Palmes Académiques.” In 2000, he received the IEEE Third Millennium Medal. His activities concern automatic control, robust control and optimization in planning and scheduling, including implementation of fuzzy logic, neural nets and genetic algorithms. He can be reached at: p.borne@ieee.org.

M. M. Crisóstomo was born in Coimbra, Portugal, in 1952. He received his BSc degree from the Department of Electrical Engineering and Computer Science of the University of Coimbra in 1978, his MSc from the Technical University of Lisbon, Portugal, in 1987 and his PhD from Brunel University, UK, in 1992. He is currently a lecturer in the Department of Electrical Engineering and Computer Science of the University of Coimbra and a researcher at the Institute for Systems and Robotics in Coimbra, Portugal. His main research interests are robotics, sensors and actuators for robots, classical and fuzzy control systems.

J.L. Fernández-Villacañas Martín graduated in Physics from the Complutense University in Madrid and received his PhD in Astrophysics in 1989. He then served as a member of the Theoretical Physics Department in Oxford University until he moved to British Telecom Research Labs in 1992. At BT he was a senior researcher in Artificial Life and Evolutionary Computation. He left the labs to join the European Commission in 1999 as a project officer in Future Emerging Technologies. Since October 2000 he is a visiting professor at the Charles III University in Madrid at the Department of Signal Theory and Communications. Dr. Martín has published extensively and has been chair and invited speaker of a number of conferences and events. His current field of work is GA theory, information ecosystems and complexity in physical systems.

M. Gestwa was born in Gelsenkirchen in 1966. From 1988 to 1991 he was trained as a computer assistant at the Institute of Flight Research. Subsequently he studied Computer Science at the Technical University of Braunschweig with a focus on

computational intelligence. During his studies he worked as a freelance software developer in the field of real-time application. Since 1997 he has worked as a Scientist at the Institute of Flight Research. His current research project deals with the cognitive pilot simulation (CoPS).

Hongfei Gong, born in 1972, graduated from the Department of Plant Protection, Zhejiang University, China, in 1993. After graduation, he became a research assistant at the Biotechnology Institute of Zhejiang University. His research work involved biological control, protein purification and gene isolation in plant protection activities. In 1999, he came to the Instituto Superior Tecnico, Technical University of Lisbon, Portugal, for his PhD study in Computer Science. His current research interests are: Artificial Life, Genetic Algorithms, Decision System and its application in Agriculture and Ecosystem.

Slim Hammadi is an associate professor of Production Planning and Control at the Ecole Centrale de Lille where he obtained a PhD in 1991. He is a member of IEEE/SMC and has served as a referee for numerous journals, including the *IEEE Transactions on SMC*. He was co-organizer of a Symposium (IMS) of the IMACS/IEEE SMC Multiconference CESA '98 held in Hammamet (Tunisia) in April 1998. He has organized several invited sessions in different SMC conferences where he was session chairman. His research is related to production control, production planning, computer science and computer-integrated manufacturing. He can be reached at: slim.hammadi@ec-lille.fr

Tetsuya Higuchi received BE, ME, and PhD degrees in Electrical Engineering from Keio University, Japan. He heads the New Circuits / System Technology Group in MIRAI Project, Advanced Semiconductor Research Center (ASRC), National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan. His current interests include evolvable Hardware Systems, Parallel Processing Architecture in Artificial Intelligence, and Adaptive Systems. Dr. Higuchi received the Ichimura Award in 1994 and the ICES Best Paper Award in 1998. He is a member of the Japanese Society for Artificial Intelligence (JSAI) and of the Institute of Electronics, Information and Communication Engineers (IEICE).

Z. Ibrahim graduated in 1990 with a BSc (Hons) in Civil Engineering from Middlesex University, London, UK. She continued with her MSc degree at Liverpool University, UK, in Structural Engineering (1994). Currently a lecturer in Structural Analysis and Dynamics at the University of Malaya, she is actively involved with the working Group Committee in Wind Loads for building structures in Malaysia.

Z. Ismail graduated in 1985 with a BA (Hons) in Mathematics from SUNY, Newpaltz, USA. She completed her MA in Applied Mathematics from Temple University, in 1987. She was a lecturer teaching mathematics at the Institute of Technology MARA, Malaysia, between 1985 to 1992, before joining the University of Malaya as a lecturer in the Department of Civil Engineering, a position she still holds. Currently, pursuing her PhD degree in Structural Dynamics at the Universiti Malaya, Malaysia, and is actively involved with the working Group Committee in Wind Loads for building structures in Malaysia.

Imed Kacem was born in Eljem, Tunisia, in 1976. He received the EngDipl degree of the ENSAIT (French “Grande Ecole”) and the DEA degree (MSc degree) from the University of Lille1, France, in Control and Computer Sciences, both in 2000. He is currently pursuing a PhD in Automatic and Computer Science at “Laboratoire d’Automatique et Informatique de Lille” of the “Ecole Centrale de Lille,” France. Mr. Kacem was selected from among the young Tunisian engineers of the “Grandes Ecoles” to receive the Tunisian Presidential Prize for 2001. He has served as a referee for the Int CIMCA’01, the Int SMC’02 Conferences and the IEEE/SMC Transactions. His research is related to the evolutionary optimization methods for discrete events system, computer science and operational research. He can be reached at: imed.kacem@ec-lille.fr.

Rens Kortmann studied Cognitive Science and Engineering at the Rijksuniversiteit Groningen and finished his Master’s dissertation in 1998. The dissertation was written at the Artificial Intelligence Laboratory of the University of Edinburgh, where he performed a one-year research internship. Since 1998 he is appointed at the Universiteit Maastricht, The Netherlands, as a PhD student, where he works on the modelling of visually guided behaviour in computer simulations and robots.

Yong Liu received his BSc degree from Wuhan University, Wuhan, in 1988; his MSc degree from Huazhong University of Science and Technology, Wuhan, in 1988; his PhD from Wuhan University, Wuhan in 1994; and the University of New South Wales, Canberra, in 1999. He is currently an associate professor at the University of Aizu, Japan. He was a research fellow at AIST Tsukuba Central 2, National Institute of Advanced Industrial Science and Technology, Japan, in 1999. He was a lecturer in the State Key Laboratory of Software Engineering, Wuhan University in 1994. His research interests include evolutionary algorithms, neural networks and evolvable hardware.

Taksiah A. Majid graduated in 1990 with a BSc (Hons) in Civil Engineering from Middlesex University, London, UK. She completed her MSc and PhD degrees at

Liverpool University, UK, in Structural Dynamics (1996). Currently a lecturer in Structural Analysis and Dynamics at Universiti Sains Malaysia, she is actively involved with the working Group Committee in Wind Loads for building structures in Malaysia.

P. Marrow began his career as a biologist, gaining a First Degree in Biology from Oxford University and a Coctorate in Mathematical Biology from York University. Postdoctoral research at Leiden and Cambridge Universities addressed evolutionary dynamics, coevolutionary theory and the evolution of reproductive strategies. In 1997 he joined a research group established by BT to focus on biologically inspired solutions to computing and telecommunications problems. Since then his research has drawn upon various aspects of biological systems in developing computational applications. Now a senior research scientist in the Intelligent Systems Laboratory BTexttract, UK, he leads a team investigating software agent systems for information management, inspired by interactions between organisms in natural ecosystems.

Yoshiyuki Matsumura received BS and MS degrees in Mechanical Engineering from Kobe University, Kobe, Japan, in 1998 and 2000, respectively. He is currently working toward the PhD in the Graduate School of Science and Technology, Kobe University. Also, he is a research fellow of the Japan Society of the Promotion of Science (DC1). His research interests are evolutionary computation, evolutionary artificial neural networks and evolutionary robotics. Mr. Matsumura is a student member of the IEEE, SICE (Society of Instrument and Control Engineers), ISCIE (Institute of System, Control and Information Engineers) and JSPE (Japan Society of Precision Engineering).

Charles S. Newton is the head of Computer Science, University of New South Wales (UNSW), at the Australian Defence Force Academy (ADFA) campus, Canberra. Dr. Newton is also the deputy rector (Education). He obtained his PhD in Nuclear Physics from the Australian National University, Canberra, in 1975. He joined the School of Computer Science in 1987 as a senior lecturer in Operations Research. In May 1993, he was appointed head of School and became professor of Computer Science in November 1993. Prior to joining ADFA, Prof. Newton spent nine years in the Analytical Studies Branch of the Department of Defence. In 1989-91, he was the national president of the Australian Society for Operations Research. His research interests encompass Group Decision Support Systems, Simulation, Wargaming, Evolutionary Computation, Data Mining and Operations Research Applications. He has published extensively in national and international journals, books and conference proceedings.

A. D. Nurse was appointed to a lectureship in Stress Analysis in the Department of Mechanical Engineering at Loughborough University, UK, in 1992 and was promoted to senior lecturer in 2000. He has published over 25 journal papers mainly in the application of inverse techniques for extracting information from experimental data. He has also prepared over 40 conference contributions involving work on Photoelasticity, Bimaterial Interface Cracks, Adhesive Joints, Finite Elements, and Damage Detection in Composites. He sits on the advisory board for FEA Ltd. (www.lusas.com), which produce the finite element software *Lusas*. He is also secretary of the Plastics Design Committee for the Institute of Materials. Dr. Nurse's research interests include Computer-Aided Engineering, Inverse Analysis, and Composite Materials.

Kazuhiro Ohkura received BS, MS, and PhD degrees in Computer Science from Hokkaido University, Sapporo, Japan, in 1988, 1990 and 1997, respectively. He is an associate professor in the Department of Mechanical Engineering, Kobe University, Japan. Before joining Kobe University as a research associate in 1993, he was with Fujitsu Laboratories, Ltd. for three years. In 1998, he was a visiting research fellow in the School of Cognitive and Computing Science, University of Sussex, UK. His research interests are evolutionary computation, reinforcement learning, artificial life, robotics and manufacturing systems. He is a member of the SICE, ISCIE, JSPE, JSME (Japan Society of Mechanical Engineers) and RSJ (Robotics Society of Japan).

D. C. Panni is a PhD candidate in the Wolfson School of Mechanical and Manufacturing Engineering at Loughborough University, UK. His research interests cover the fields of Genetic Algorithms, Finite Element Analysis, Inverse Analysis and The Design of Advanced Composite Materials. In particular he has specialised in the use of novel methods of integrating GAs and the FE method to solve a range of structural engineering problems.

V. Fernão Pires received his BS degree in Electrical Engineering from the Institute Superior of Engineering of Lisbon, Portugal, in 1988 and his MS and PhD degrees in Electrical and Computer Engineering from the Technical University of Lisbon, Portugal, in 1995 and 2000, respectively. Since 1991 he is a member of the teaching staff in the Electrical Engineering Department of Superior Technical School of Setúbal – Polytechnic Institute of Setúbal, Portugal. Presently he is a professor, teaching Power Electronics and Control of Power Converters. He is also researcher at Centro de Automática of UTL. His present research interests are in the areas of Low-Distortion Rectifier topologies, Converter Control, Modelling and Simulation.

Eric Postma studied Cognitive Science at the Catholic University of Nijmegen and received his Master's degree in 1989. Since that year he is affiliated with the Computer Science Department of the Universiteit Maastricht, The Netherlands, where he is currently appointed as associate professor and coordinating the Neural Networks and Adaptive Behaviour group. Dr. Postma has published about neural networks and adaptive behaviour in many international journals and conference proceedings.

Anet Potgieter is currently employed by CoreProcess (Pty) Ltd. where she holds the position of systems architect. She has extensive industrial experience in embedded distributed applications as well as supply chain management applications. She received her MSc (Computer Science) from the University of Pretoria, South Africa, in 1994, and is currently pursuing her PhD in the area of software engineering and component-based systems, under the instruction of Professor Judith Bishop. Her research interests include software engineering, distributed artificial intelligence, data mining and web-technology. She is a student member of the IEEE and the ACM.

N. H. Ramli graduated in 2001 with a First Class Degree in Civil Engineering from the University of Malaya, Malaysia. Currently pursuing her PhD degree in Steel Structures at Sheffield University, UK, she is also a tutor at the Department of Civil Engineering, University of Malaya, Malaysia.

Agostinho Claudio da Rosa is director of Evolutionary Systems and Biomedical Engineering Lab at the Institute for Systems and Robotics (LaSEEB-ISR), Lisbon, Portugal, and associate professor of the Department of Electrical Engineering and Computers of Technical University of Lisbon (UTL). She previously served as visiting professor at the School of Medicine Stanford University in 99-00. After graduation from the Electrical Engineering & Computing at Instituto Superior Tecnico (IST) in 1978, MSc and PhD degrees in Electronic Engineering and Computers, in 1984 and 1990 from IST-UTL. Her main research interests include: Artificial Life, Biomedical Engineering, Signal and Image Processing, Evolutionary Computation and Computational Intelligence.

M. Shackleton graduated from Sheffield University in Computer Science. He joined the Image Processing and Computer Vision group at BT in 1989. In this group he carried out research into novel computer vision algorithms, many inspired by natural computation techniques. In 1996 Mr. Shackleton moved across to the Future Technologies Group at Adastral Park, whose remit is to develop novel solutions to BT's problems using a nature-inspired approach. Within this group he

has carried out research including, amongst other projects, developing a novel “information chemistry” architecture, and evolutionary computation techniques. This work has led to patents, international papers and book chapters. He also recently edited a special issue of the *BT Technology Journal* (October 2000) on nature-inspired computation. He now leads the future technologies group, where he works on research and applications within the domain of evolutionary computation and adaptive systems, and exploitation of these technologies within the business.

D. P. Solomatine received the MS degree in Systems Engineering from the Moscow Aviation Institute (University) in 1979. From 1979 to 1990 he worked at the Institute for Systems Analysis of the Russian Academy of Sciences (from 1986 as a Senior Researcher). He received his PhD in Systems and Management Sciences in 1984. He actively collaborated with the International Institute for Applied Systems Analysis (Austria). In 1989-90 he spent a year as a Researcher at the Delft University of Technology, and since March 1990 he is a staff member (from 2000, Associate Professor) of the Hydroinformatics section of the International Institute for Infrastructural, Hydraulic and Environmental Engineering (IHE-Delft), The Netherlands. His research interests include machine learning, data-driven modeling, applications of chaos theory, global optimization and Internet-based computing.

Ida Sprinkhuizen-Kuyper studied Applied Mathematics at the Universiteit van Amsterdam and received her Master’s degree cum laude in 1973. She received her PhD in Mathematics in 1979. From 1984 until 1999 she worked for the Computer Science Department of the Universiteit Leiden, and since 1999 she is affiliated with the Computer Science Department of the Universiteit Maastricht, The Netherlands. Her main research interests are neural networks and evolutionary algorithms.

Pieter Spronck studied Computer Science at Delft University of Technology and received his Master’s degree cum laude in 1996. Since 2001 he is affiliated as a researcher with the Computer Science department of the Universiteit Maastricht, The Netherlands, where he is also working on a PhD thesis. Before that, he worked for 15 years in the field of Computer and Information Science as a developer, researcher and project leader for several companies and a research institute.

R. J. Stonier is an associate professor at Central Queensland University, Australia. He received his Bachelor’s of Science (1968) and Honours (1969) in Mathematics and completed a PhD (1978) in Mathematics at the University of Queensland. His research interests are in non-linear control of multi-robot systems using Liapunov theory and sliding mode, fuzzy logic and neural networks, evolutionary computation

including evolution algorithms to learn fuzzy logic controllers in robot soccer and fuzzy image enhancement filters, solutions to continuous nonlinear constrained optimal control problems, problems of optimisation in VLSI, CPI prediction, and irrigation strategies for water flow control in cropped soils.

G. Sundaraj obtained a Bachelor's degree in Civil Engineering with honours in 1999 from the Universiti Sains Malaysia. He is currently pursuing a master's degree in Civil Engineering from the same university. He is employed by the Construction Industry Board of Development (CIDB), Malaysia, as a manager of the Research and Development Unit, Technology Division at CIDB Malaysia. He is committee member of the Malaysian Standard of Wind Loading Working Group and a member of the Australasian Wind Engineering Society.

P.J. Thomas obtained his Engineering degree from Central Queensland University, Australia, with First Class Honours in 1998. His current PhD research is centered on the evolutionary learning of fuzzy control in robot-soccer. Other research interests include digital image processing, digital signal processing, participation in robot-soccer competitions, and fostering community awareness of science and technology through robot-soccer.

Kanji Ueda is a professor of Mechanical Engineering at Kobe University, Kobe, Japan. He has been engaged in research and teaching in the fields of manufacturing engineering and systems for more than 25 years, during which time he has authored more than 300 published papers. He has led the IMS Program Next Generation Manufacturing Systems and the international project "Biological Manufacturing Systems" of the Consortium for Advanced Manufacturing. He has been chairman of the Committee on Manufacturing Systems of the College International for l'Etude Scientifique des Techniques de Production Mecanique (CIRP) since 1998. His research interests include biological manufacturing systems, intelligent artifacts, robotics, emergent synthesis and artificial life. Professor Ueda is a member of CIRP, JSPE, JSME, SICE, ISCIE, RSJ, Society for Manufacturing Engineers, and Danube Adria Association for Automation and Manufacturing.

Simon X. Yang received his BSc degree in Engineering Physics from Peking University, China; his first MSc degree in Biophysics from Academia Sinica in Beijing; his second MSc degree in Electrical Engineering from the University of Houston, USA, and his PhD in Electrical and Computer Engineering from the University of Alberta, Canada. He has been an assistant professor of Engineering Systems and Computing at the University of Guelph, Canada, since 1999. Currently he is the director of the Advanced Robotics & Intelligent Systems (ARIS)

Lab at the University of Guelph. His research areas include Robotics, Intelligent Systems, Control Systems and Computational Neuroscience. He has published more than 100 journal papers, book chapters and conference proceedings.

Index

A

adaptive landscapes 288
 adaptive learning 130
 agent 169
 agricultural production 184
 AIDA 149
 aircraft cockpit 149
 airplane system technology 150
 ambient vibration testing 307
 ant colony 171
 Approach by Localization (AL) 241
 artificial intelligence 170
 Artificial Life modeling approach 185
 Artificial Neural Networks (ANN) 197
 attainment surfaces 222
 ATTAS 149
 autonomous agents 168, 170

B

Bactrocera oleae 184
 Bayesian agencies 168
 Bayesian agents 168
 Bayesian networks 168, 169
 behavior networks 170
 binary alphabet 238
 binary string 142
 box-pushing controller 107
 BTGP 280

C

Cartesian workspace 71
 Chaos theory 215

civil engineering structures 304
 classic coding 238
 Classical Evolution Strategies (CES)
 264
 classification 200
 climatic data 184
 clustering 200
 clusters 175
 complex control systems 122
 complex interaction protocols 170
 computational intelligence 169
 computer simulation 268
 control vector 44
 Controlled Evolutionary Approach
 (CEA) 233, 243
 conventional control difficult 43
 coverage metrics 222
 critical machine 234
 Cross-Spectral Density (CSD) 302, 313
 crossover 287
 cycle-cutset conditioning 175
 cycle-cutsets 175

D

damage detection 137
 design wind load 302
 Differential Evolution (DE) 219, 222
 Discrete Fourier Transform (DFT) 312
 doping 118

E

ecological system analysis 185
 ensemble learning system 2

epistasis variance 286
 epoch 7
 error ratio 222
 Evolution Strategies (ES) 263
 evolution strategies algorithms 264
 Evolutionary Algorithm (EAs) 88, 104,
 110, 116, 218, 219, 280
 evolutionary biology 280
 evolutionary computation context 280
 Evolutionary Programming (EP) 263
 existence of variation 281

F

Fast Evolution Strategies (FES) 265
 Fast Fourier Transform (FFT) 302, 312
 Finite Element (FE) 136
 fitter mutants 286
 flight simulation 150
 Flight Training Devices (FTDs) 149
 forced vibration methods 302
 forced vibration testing 306
 Fourier Transform 311
 Frequency Response Function (FRF)
 302
 fuzzy amalgamation 89
 fuzzy logic 122
 fuzzy logic application 88
 fuzzy logic controllers 89
 Fuzzy logic systems 122
 fuzzy pilot 151
 fuzzy rule-based systems 197

G

general regression 43
 General Regression Network (GRNN)
 43
 generational distance 222
 generic neural network 25
 genetic algorithm (GA) 123, 137, 238,
 263
 genetic diversity 283
 genetic drift 285
 genetic manipulations 246
 Genetic Programming System (BTGP)
 287
 genetic reinforcement learning 106
 Genetically Modified Organisms (GMO)

246

genotype 138
 German Aerospace Center (DLR) 149
 glide slope 154
 Global Combined Discrete Recombina-
 tion 268
 Global External (GE) 112
 Global Internal (GI) 112
 global semantics 173
 golden unit 23
 ground based simulators 150
 Gust Response Factors (GRF) 304

H

heat unit accumulation concept 187
 helicopter control 43
 hidden periodicities 312
 Hierarchical Fuzzy Logic Systems 126
 human decision maker 219

I

ILS tracking task 151
 impact tests 307
 information retrieval 279
 intelligent components 168
 intelligent control systems 43
 interaction 170
 inverse analyses 136
 inverse system identification 137

J

Job-shop Scheduling Problem (JSP)
 233
 join-tree propagation 175

K

Khepera type 107
 knowledge base (KB) 88
 knowledge space 23

L

lay-up design 143
 learning 2
 Local External (LE) 112
 Local Internal (LI) 112
 local semantics 173

Lyapunov stability theory 71

M

machine learning 199
 main rotor 45
 makespan 234
 Malaysian Wind Code 310
 Man/Machine Interface (MMI) 148
 Mapping Genetic Algorithm (MGA) 280, 287
 Mixtures-of-Experts (ME) 4
 model 198
 model tree 208
 Most Probable Explanation (MPE) 174
 Multiobjective Evolutionary Algorithms (MEAs) 218
 Multiobjective Optimization Problems (MOPs) 218
 mutation 281
 mutation variance 284

N

negative correlation learning 5
 neighboring neurons 73
 neural computation 23
 neural controller configuration 114
 neural network 42, 104
 neural network methods 2, 70
 neural variables 22
 neutral mutants 286
 next-generation 168
 NN-based controllers 44
 Noise 263
 nominal data 200
 Non-dominated Sorting Genetic Algorithms (NSGA) 221
 non-linear dynamics 215
 non-serial dynamic programming algorithms 175
 nonstationary environment 69
 nonstationary statistics 54

O

olive fly's life cycle 186
 olive trees 184
 Operational Research (OR) 219

output vector 44

P

parameter sensitivity 82
 Pareto Archived Evolution Strategy (PAES) 222
 pest management 183
 phenotype 138
 phenotypes (decision trees) 287
 phenotypic diversity 284
 pilot model approach 151
 Plasmopara viticola 185
 poikilothermic animal 188
 population dynamics analysis methods 184
 Power Spectral Density (PSD) 302, 312, 315
 pre-imaginary phases 186
 probability 28
 pruning 209
 pull back tests 307
 push 164

R

Radial-Basis Function (RBF) neural network 205
 Ramped growth 287
 Random Sampling Evolutionary Algorithm (RAND) 220
 randomized algorithms 22
 real-time path planning 69
 real-time trajectory 70
 regression tree 208
 robot soccer system 89
 robot's proximity sensors 107
 Robust Evolution Strategies (RES) 265
 robustness index 27
 root locus 47

S

scheduling algorithm 242
 schemata generation algorithm 242
 sequential training methods 3
 shaker tests 307
 simple agents 171
 simulation models 198

- Single Objective Evolutionary Algorithm (SOEA) 220
- single sampling line 223
- spectrum analysis 312
- splines 205
- spread 222
- state vector 44
- statistical learning theory 200
- Strength Pareto Evolutionary Algorithm (SPEA) 219
- strength-to-weight ratio 143
- support vector machine (SVM) 200

T

- task instances 111
- thrust increase 164
- thrust reduction 164
- training set 201
- travelling salesman problem 239
- Turbulence Intensities (TI) 302, 313

U

- uncertain environments 176

V

- variation 281
- Vector Evaluated Genetic Algorithm (VEGA) 220
- verification set 201
- vibration testing 305

W

- wind speed 309